K-MEANS: CRITICAL ANALYSIS ON THE TECHNIQUES USED TO DETERMINE THE OPTIMAL VALUE OF *K* IN HIGH-DIMENSIONAL DATASETS

GIKERA RUFUS KINYUA (MSC)

J57 /26723 / 2018

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF DOCTOR OF PHILOSOPHY (COMPUTER SCIENCE) IN THE SCHOOL OF PURE AND APPLIED SCIENCES OF KENYATTA UNIVERSITY

DECLARATION

This thesis is my original work and has not been presented for a degree in any other university or for any other academic award.

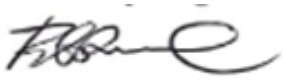
Signature:  Date: 21st November 2024

GIKERA RUFUS KINYUA - J57 /26723 / 2018

Department of Computing & Information Science, Kenyatta University

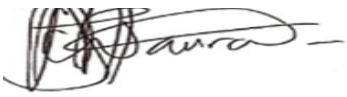
SUPERVISORS:

We confirm that the work reported in this thesis was carried out by the candidate under our supervision:

Signature:  Date: 22nd November 2024

PROF. ELIZAPHAN MAINA

Department of Computing & Information Science, Kenyatta University, Nairobi, Kenya

Signature:  Date: 21st November 2024

PROF. JONATHAN MWAURA

Khoury College of Computer Sciences, Roux Institute At Northeastern University, Portland, ME, 04101, United State of America

Signature:  Date: 21st November 2024

ENG. DR. SHADRACK MAMBO

Department of Electrical Engineering, Walter Sisulu University, Eastern Cape, South Africa

DEDICATION

This thesis is dedicated to my wife Kathambi, my daughters Kendi and Makena and my son Gikera. Their moral support and encouragements in the entire process gave me perseverance to try my best.

ACKNOWLEDGEMENTS

I thank God for His grace and strength during my entire PhD journey. Writing this thesis would not have been possible without the help and assistance of my supervisors, colleagues and important people in my life. I would like to start by thanking my three supervisors, Prof. Jonathan Mwaura of Khoury College of Computer Sciences, Roux Institute At Northeastern University, Portland, ME, 04101. I also thank Dr Shadrack Mambo of the Electrical Engineering Department at Walter Sisulu University, Eastern Cape, South Africa and Prof. Maina Muuro of Computing & Information Science Department at Kenyatta University, Kenya. Their valuable insights, suggestions, advice and criticism improved my PhD work tremendously. They edited my manuscripts, suggested several improvements and criticism, and made my work scientifically and academically sound, both in content and in style. I appreciate Prof. Musau of Riara University and Ken Mwirigi of Kenyatta University for their support during the empirical set ups. I also appreciate Prof John Kihoro of Cooperative University, Kenya for his guidance in the data analysis process. I appreciate my Dad Godfrey Gikera, my brothers Thomas, Dickens, John and my sister Eunice for their moral support during my PhD journey. Finally, I am extremely indebted to my wife Kathambi and our three children; Kendi, Makena and Gikera. They urged me to never give up!

TABLE OF CONTENTS

DECLARATION ..... **i**
DEDICATION ..... **ii**
ACKNOWLEDGEMENTS ..... **iii**
TABLE OF CONTENTS ..... **iv**
LIST OF TABLES ..... **xii**
LIST OF FIGURES ..... **xvi**
LIST OF EQUATIONS ..... **xviii**
ABBREVIATIONS & TERMS ..... **xx**
OPERATIONAL DEFINITION OF TERMS ..... **xxii**
ABSTRACT ..... **xxx**
CHAPTER ONE: INTRODUCTION **1**
1.1 Background Information 1
1.2 Statement of the Problem 4
1.3 Justification 10
1.4 Research Objectives 11
1.4.1 General Objective 11
1.4.2 Specific Objectives 11
1.5 Research Questions 12
1.6 Research Hypothesis 12
1.7 Scope of the Study 12
1.8 Significance of the Study 13
1.9 Thesis Outline 14
2.0 Language Conventions 15
CHAPTER TWO: LITERATURE REVIEW **17**
2.1 Introduction 17
2.2 Clustering Theory 18
2.3 Fundamentals of the *K*-means Clustering Algorithms 19
2.3.1 Structure of the *K*-means Algorithms 20
2.3.2 K-means Hyperparameters 22
2.3.3 Traditional Elbow Method in *K*-hyperparameter Tuning 24
2.4 High dimensional Datasets and the associated Cluster Analysis Challenges 26
2.5 Challenges of Tuning the *K*-hyperparameter in High Dimensional Spaces 29

2.6 Dimensionality Reduction Methods in High Dimensional Space Clustering 31
2.6.1 Principal Component Analysis 33
2.6.2 Kernel PCA 35
2.6.3 Factorial Analysis 36
2.6.4 Singular Value Decomposition 38
2.6.5 Uniform Manifold Approximation and Projection 39
2.6.6 t-Distributed Stochastic Neighborhood Embedding 40
2.6.7 Locally Linear Embedding 41
2.6.8 Autoencoders 43
2.6.9 Kernel Functions and Variations for Dimensionality Reduction Methods 45
2.6.9.1 Linear Kernel 46
2.6.9.2 Polynomial Kernel 46
2.6.9.3 Radial Basis Function (RBF) Kernel 47
2.6.9.4 Sigmoid Kernel 47
2.6.9.5 Laplacian Kernel 48
2.6.9.6 Bessel Kernel 48
2.6.9.7 ANOVA Kernel 49
2.6.9.8 Special Variations for Variety of Dimensionality Reduction Methods 50
2.7 Evaluation metrics for *K*-hyperparameter Tuning Techniques in High Dimensional Space Clustering 56
2.7.1 Internal Validation Indexes 57
2.7.1.1 Dunn Index (DI) 57
2.7.1.2 Calinski-Harabasz Index (CH) 58
2.7.1.3 Davies Bouldin Index (DB) 59
2.7.1.4 Silhouette Index (SI) 59
2.7.1.5 Bayesian Information Criterion (BIC) 61
2.7.1.6 Point Bi-serial 63
2.7.1.7 Sum of squares 64
2.7.2 Accuracy 66
2.7.2.1 Adjusted Rand Index (ARI) 66
2.7.2.2 Normalized Mutual Information (NMI) 67
2.7.2.3 Homogeneity, Completeness, and V-measure 67

2.7.3 Jaccard Coefficient 68
2.7.4 F1-Score 70
2.7.5 Cochran's Q score 71
2.7.6 Chi Square 72
2.7.7 T-test 73
2.7.8 Sum of Squared Error (SSE) 74
2.8 Critical Review of the *K*-hyperparameter Tuning Techniques in High Dimensional Space Clustering 76
2.8.1 Analysis of the *K*-hyperparamter Tuning Techniques, their Tuning Strategies and Dimensionality Reduction Methods 78
2.8.2 Discussions on the Critical Review Analysis 110
2.9 Conceptual Framework, Variables and Operationalization of Variables 132
2.9.1 Conceptual Elements 133
2.9.1.1 Independent Variables 133
2.9.1.2 Moderating Variables 134
2.9.1.3 Dependent Variables 134
2.9.2 Operationalization of the Variables 135
2.10 Summary 136
CHAPTER THREE: RESEARCH METHODOLOGY 139
3.1 Introduction 139
3.2 Research Philosophy 140
3.3 Research Design 141
3.4 Experimental Design for the Empirical Analysis 143
3.4.1 Research Hypothesis 143
3.4.2 Experiments 145
3.4.3 Integrated Development Environment and the Libraries 157
3.4.4 Research Instruments 158
3.4.5 Population, Sampling Strategy and Sample size 159
3.4.5.1 High Dimensional *K*-hyperparameter Tuning Techniques 160
3.4.5.2 High Dimensional Input Datasets 162
3.4.5.3 Pre-training Datasets 167
3.4.5.4 Experimental Dimensionality Reduction Methods 171
3.4.6 Experimental Data Collection Process and Techniques 173
3.4.7 Analysis techniques on the Experimental Data 174

3.5 Model Development 184
3.5.1 Conceptual Design 185
3.5.2 Model Architecture 185
3.5.3 Model Design 190
3.5.4 Model Prototyping 202
3.5.5 Evaluation Experiments 203
3.6 Research Ethics 205
CHAPTER FOUR: RESULTS 206
4.1 Introduction 206
4.2 Experimental Results 207
4.2.1 Two-way ANOVA analysis on the AIPS of the *k*HTs in different DTs 213
4.2.2 Two-way ANOVA analysis on the CBTs of the kHTs in different DTs 215
4.2.3 Two-way ANOVA analysis on AIPS of the *k*HTs in different RMs 216
4.2.4 Two-way ANOVA analysis on CBTs of the *k*HTs in different RMs 218
4.2.5 Two-way ANOVA analysis on CBTs of *k*HTs in HDDs of varied DLs 220
4.2.6 Two-way ANOVA analysis on the interaction between DTs and DLs of HDDs on AIPS of the *k*HTs. 221
4.2.7 Two-way ANOVA analysis on the interaction between DTs and DLs of HDDs on CBTs of the *k*HTs 221
4.2.8 Two-way ANOVA analysis on the interaction between DTs and RMs on AIPS of the *k*HTs 224
4.2.9 Two-way ANOVA analysis on the interaction between DTs and RMs on CBTs of the *k*HTs 238
4.2.10 Two-way ANOVA analysis on the interaction between DLs and RMs of HDDs on the AIPS of the *k*HTs. 250
4.2.11 Two-way ANOVA analysis on the interaction between DLs and RMs of HDDs on the CBTs of the *k*HTs. 250
4.2.12 Three-way ANOVA on the AIPS based on the *k*HTs, DTs and DLs of HDDs 255
4.2.13 Three-way ANOVA analysis on the AIPS based on the choice of the *k*HTs, DLs of the HDDs and RMs applied 257
4.2.14 Three-way ANOVA analysis on the AIPS based on the DT of HDDs, DLs of HDDs and RMs applied 258

4.2.15 Three-way ANOVA analysis on the AIPS based on the choice of the *k*HTs, DTs of HDDs and RMs applied. 261
4.2.16 Three-way ANOVA analysis on CBTs of the *k*HTs, DTs and DLs of HDDs 276
4.2.17 Three-way ANOVA analysis on the CBTs of the *k*HTs, DLs of HDDs and the RMs applied 278
4.2.18 Three-way ANOVA analysis on CBTs of the DTs of HDDs, DLs of HDDs and RMs applied 281
4.2.19 Three-way ANOVA analysis on the CBTs of the *k*HTs, DTs of HDDs and RMs applied 285
4.3 Key findings from the empirical study and how they guided the system development methodology on the new *k*-hyperparameter tuning technique. ... 286
4.4 Evaluation on the effectiveness of the newly developed *k*-hyperparameter tuning technique against the existing ones in a variety of high-dimensional datasets. 292
4.4.1 Kruskal-Wallis H statistic on the newly developed *k*HT and the existing ones in a variety of high-dimensional datasets. 293
4.4.2 One-way ANOVA analysis on the EIVI of the newly developed *k*HT against the existing ones in a variety of HDDs 295
4.4.3 One -way ANOVA analysis on CBTs of the newly developed *k*HT against the existing ones in a variety of HDDs. 297
4.5 Whisker-box plots of the different k-hyperparameter tuning techniques in a variety of high dimensional datasets. 298
4.6 Comparison of the cluster visualizations in different HDDs between the standard autoencoder and the improved one in the new *k*HT. 299
4.6.1 Visualization of the Tox-171 dataset using the newly developed *k*HT and the standard autoencoder. 301
4.6.2 Visualization of the Tox-171 dataset using the newly developed *k*HT and an improved autoencoder. 301
4.6.3 Visualization of the Yale image dataset using the newly developed *k*HT and a standard autoencoder. 302
4.6.4 Visualization of the Yale image dataset using the newly developed *k*HT and an improved autoencoder. 302

4.6.5 Visualization of the Reuters dataset using the newly developed *k*HT and a standard autoencoder. 303
4.6.6 Visualization of the Reuters dataset using the newly developed *k*HT and an improved autoencoder. 303
4.6.7 Visualization of the Lung Cancer dataset using the newly developed *k*HT and a standard autoencoder. 304
4.6.8 Visualization of the Lung Cancer dataset using the newly developed *k*HT and an improved autoencoder. 304
4.6.9 Visualization of the Covid-19 coughs dataset using the newly developed *k*HT and a standard autoencoder. 305
4.6.10 Visualization of the Covid-19 coughs dataset using the newly developed *k*HT and improved autoencoder. 305
4.6.11 Visualization of the Heart beat sounds dataset using the newly developed *k*HT and a standard autoencoder. 306
4.6.12 Visualization of the Heart beat sounds dataset using the newly developed *k*HT and an improved autoencoder. 306
4.6.13 Visualization of the YouCook dataset using the newly developed *k*HT and a standard autoencoder. 308
4.6.14 Visualization of the YouCook dataset using the newly developed *k*HT and an improved autoencoder. 309
4.6.15 Visualization of the YouTube vlogs for cooking beef in Kenya using the newly developed *k*-hyperparameter tuning technique and an improved autoencoder. 309
CHAPTER FIVE: DISCUSSION 312
5.1 Introduction 312
5.2 Discussion on the results 312
5.2.1 Discussion on the Empirical Analysis Results on the two-way ANOVA and the Interactions 314
5.2.1.1 Analysis of Two-way ANOVA Across Varied Data Types 314
5.2.1.2 Two-way ANOVA and interactions on the CBTs of the *k*HTs in different DTs 315
5.2.1.3 Two-way ANOVA and interactions on the AIPS of the *k*HTs in different RMs 317

5.2.1.4 Two-way ANOVA and interactions on the CBTs of the *k*HTs in different RMs applied 320
5.2.1.5 Two-Way ANOVA and interactions on the AIPS of the *k*HTs in HDDs of varied DLs 321
5.2.1.6 Two-way ANOVA and interactions of the DTs and DLs of HDDs on the AIPS of the *k*HTs 323
5.2.1.7 Two-way ANOVA and interactions of the DTs and DLs of HDDs on the CBTs of the *k*HTs 325
5.2.1.8 Two-way ANOVA and interactions of the DTs of HDDs and RMs on the AIPS of the *k*HTs 326
5.2.1.9 Two-way ANOVA and interactions of the DTs of HDDs and RMs on the CBTs of the *k*HTs 329
5.2.1.10 Two-way ANOVA and interactions of the DLs of HDDs and RMs on the AIPS of the *k*HTs 331
5.2.1.11 Two-way ANOVA and interactions of the DLs and RMs on the CBTs of the *k*HTs 333
5.2.1.12 Two-way ANOVA analysis and interactions on the CBTs of *k*HTs in HDDs of varied DLs 334
5.2.2 Empirical Analysis Results on the three-way ANOVA and Interactions ... 336
5.2.2.1Three-way ANOVA on the AIPS based on the DT of HDDs, DLs of HDDs and RMs applied 336
5.2.2.2 Three-way ANOVA on the AIPS based on the choice of the *k*HTs, DTs of HDDs and RMs applied. 337
5.2.2.3 Three-way ANOVA on the CBTs of the *k*HTs, DTs and DLs of HDDs 338
5.2.2.4 Three-way ANOVA on the CBTs of *k*HTs, DLs of HDDs and RMs applied 339
5.2.2.5 Three-way ANOVA on CBTs of the DTs of HDDs, DLs of HDDs and RMs applied 340
5.2.2.6 Three-way ANOVA on the CBTs of the *k*HTs, DTs of HDDs and RMs 341
5.2.2.7 Three-way ANOVA on the AIPS based on the choice of the *k*HTs, DLs of the HDDs and RMs applied 341

5.2.2.8 Three-way ANOVA on the AIPS based on the *k*HTs, DTs and DLs of HDDs 342
5.2.3 Evaluation Results on the Effectiveness of the new technique 343
5.3 Conclusion 345
CHAPTER SIX: CONCLUSION & RECOMMENDATION FOR FUTURE WORK 349
6.1 Introduction 349
6.2 Summary and conclusion 350
6.3 Research contributions 351
6.3.1 Technical contribution 352
6.3.2 Theoretical Contribution 353
6.4 Limitations 354
6.5 Recommendations for further research 355
REFERENCES 357
APPENDICES 383
Appendix 1: Work schedule 383
Appendix 2: Budget 384
Appendix 3: Publications 385
Appendix 4: Program Code Segments 386
Appendix 5: The R-based Jamovi interface for the Analysis of Experimental Data 416
Appendix 6: NACOSTI Research license 417
Appendix 7: Sample Experimental Check List 418

LIST OF TABLES

Table 2.1 Example of a High Dimensional Dataset from a Clinical Data..... 29
Table 2.2: Description of the *K*-hyperparameter Tuning Techniques, Dimensionality Reduction Methods as well as their Tuning Strategies and Limitations 79
Table 2.3: Nature of the Datasets used with the *K*-hyperparameter Tuning Techniques in High Dimensional Space Clustering..... 93
Table 2.4: Metrics and Scores in the Evaluation of the existing *K*-hyperparameter Tuning Techniques in High Dimensional Space Clustering..... 101
Table 2.5: Key Findings from the Literature Review Analysis and how they guided the Methodology..... 128
Table 3.1: Comparative results' analysis of the pilot study against the theoretical analysis 146
Table 3.2: Experiments on the various techniques and datasets using the PCA methods.... 148
Table 3.3: Validation Experiments on the KHT techniques using the Best Performing Dimensionality Reduction Methods in a Variety of Datasets 153
Table 3.4: Python Libraries for the Experimental High Dimensional Input Datasets..... 157
Table 3.4: Description of the Experimental High Dimensional Datasets..... 162
Table 4.1: Summary of the set of variables, interactions and the data analysis method applied in the different set of experiments. 209
Table 4.2: Two-Way ANOVA analysis on AIPS of *k*HTs in different DTs. 213
Table 4.3: Post-hoc test of *k*HTs in the two-way ANOVA analysis on the AIPS of the *k*HTs in different DTs. 214
Table 4.4: Post-hoc test of DTs in the two way ANOVA analysis on the AIPS of the *k*HTs in different DTs. 214
Table 4.5: Two-Way ANOVA analysis on CBTs of the *k*HTs in different DTs..... 215
Table 4.6: Post-hoc test of DTs in the ANOVA analysis on the CBTs of *k*HTs in different DTs. 215
Table 4.7: Two-way ANOVA analysis on AIPS of the *k*HTs in different RMs. 216
Table 4.8: Posthoc test of the *k*HTs in the ANOVA analysis on AIPS of *k*HTs in different RMs. 216
Table 4.9: Post-hoc test of RMs in the ANOVA analysis on AIPS of the *k*HTs in different RMs. 217
Table 4.10: Two-way ANOVA analysis on CBTs of *k*HTs in different RMs. 218

Table 4.11: Post-hoc test of RMs in the ANOVA analysis on CBTs of the *k*HTs in different RMs. 218
Table 4.12: Two-Way ANOVA analysis on CBTs of the *k*HTs in HDDs of varied DLs. ... 220
Table 4.13: Posthoc test of DLs in the two-Way ANOVA analysis on CBTs of the *k*HTs in HDDs of varied DLs. 220
Table 4.14: Two-Way ANOVA analysis on the interaction between DTs and DLs of HDDs on AIPS of the *k*HTs..... 221
Table 4.15: Two-Way ANOVA analysis on the interaction between DTs and DLs of HDDs on CBTs of the *k*HTs. 221
Table 4.16: Posthoc test of DTs in the two-Way ANOVA analysis on the interaction between DTs and DLs of HDDs on CBTs of the *k*HTs..... 222
Table 4.17: Posthoc test of DLs in the two-Way ANOVA analysis on the interaction between DTs and DLs of HDDs on CBTs of the *k*HTs..... 222
Table 4.18: Posthoc test of the interaction between DTs and DLs in the two-Way ANOVA analysis on the interaction between DTs and DLs on CBTs of the *k*HTs..... 223
Table 4.19: Two-Way Anova analysis on the interaction between DTs and RMs on AIPS of the *k*HTs. 224
Table 4.20: Post-hoc test of DTs in the two-Way ANOVA analysis on the interaction between DTs and RMs on AIPS of the *k*HTs. 225
Table 4.21: Post-hoc test of the interaction between DTs and RMs in the two-Way ANOVA analysis on the interaction between DTs and RMs on AIPS of the *k*HTs..... 225
Table 4.22: Two-Way ANOVA analysis on the interaction between DTs and RMs on CBTs of the *k*HTs. 238
Table 4.23: Post hoc test of the interaction between DTs and RMs in the two-Way ANOVA analysis on the interaction between DTs and RMs on CBTs of the *k*HTs..... 238
Table 4.24: Two-Way ANOVA analysis on the interaction between DLs and RMS on AIPS of the *k*HTs..... 250
Table 4.25: Two-Way ANOVA analysis on the interaction between the DLs and RMs on CBTs of the *k*HTs. 250

Table 4.26: Post-hoc analysis of the interaction between DLs and RMs of HDDs in the two-way ANOVA analysis on the interaction between DLs and RMs of HDDs on CBTs of the *k*HTs. 251
Table 4.27: Three-way ANOVA analysis on the AIPS based on the choice of *k*HT, DTs and DLs of the HDDs. 255
Table 4.28: Post-hoc analysis of the *k*HT techniques in the three-way ANOVA analysis on the AIPS based on the choice of *k*HT, DTs and DLs of HDDs. 256
Table 4.29: Post-hoc analysis of DTs in the three-way ANOVA analysis of the AIPS based on the choice of *k*HT, DT and DL of the HDDs. 256
Table 4.30: Three-way ANOVA analysis on the AIPS based on the choice of the *k*HT, DLs of HDDs and RMs applied. 257
Table 4.31: Post-hoc analysis of the *k*HTs in the three-way ANOVA analysis on the AIPS based on the choice of *k*HT, DLs of HDDs and RMs applied. 258
Table 4.32: Three-way ANOVA analysis on the AIPS based on the DTs of HDDs, DLs of HDDs and RMs applied. 258
Table 4.33: Post-hoc analysis of the DTs in the three-way ANOVA analysis on the AIPS based on the DTs of HDDs, DLs of HDDs and RMs applied. 259
Table 4.34: Posthoc analysis of the RMs in the three-way ANOVA analysis on the AIPS based on the DTs of HDDs, DLs of HDDs and the RMs applied. 260
Table 4.35: Three-way ANOVA analysis on the AIPS based on the choice of the *k*HTs, DTs of HDDs and RMs applied. 261
Table 4.36: Posthoc analysis of the *k*HTs in the three-way ANOVA analysis on the AIPS based on the choice of the *k*HTs, DTs of HDDs and RMs applied. 262
Table 4.37: Posthoc analysis of the RMs in the three-way ANOVA analysis on the AIPS based on the choice of the *k*HTs, DTs of HDDs and RMs applied. 263
Table 4.38: Post-hoc analysis of the DTs in HDDs in the three-way ANOVA analysis on the AIPS based on the choice of the *k*HTs, DTs of HDDs and RMs applied. 264
Table 4.39: Post-hoc analysis of interaction between DTs and RMs in the three-way ANOVA analysis on the AIPS based on the choice of the *k*HTs, DTs of HDDs and RMs. 264
Table 4.40: Three-way ANOVA analysis on CBTs of the *k*HTs, DTs and DLs of HDDs. 276

Table 4.41: Post-hoc analysis of the *k*HTs in the three-way ANOVA analysis on CBTs of the *k*HTs, DTs and DLs of HDDs. 277
Table 4.42: Post-hoc analysis of the DTs of HDDs in the three-way ANOVA analysis on CBTs of the *k*HTs, DTs and DLs of HDDs. 277
Table 4.43: Post-hoc analysis of the DLs of HDDs in the three-way ANOVA analysis on CBTs of the *k*HTs, DTs and DLs of HDDs. 278
Table 4.44: Three-way ANOVA analysis on CBTs of the *k*HT techniques, DLs of HDDs and RMs applied. 278
Table 4.45: Post hoc analysis of the DLs of HDDs in the three-way ANOVA analysis on CBTs of the *k*HTs, DLs of HDDs and RMs applied. 279
Table 4.46: Post hoc analysis of the RMs in the three-way ANOVA analysis on CBTs of the *k*HTs, DLs of HDDs and RMs applied. 280
Table 4.47: Three-way ANOVA analysis on CBTs of the DTs of HDDs, DLs of HDDs and RMs applied. 281
Table 4.48: Post-hoc analysis of the DTs in HDDs in the three-way ANOVA analysis on CBTs of the DTs of HDDs, DLs of HDDs and RMs applied. 282
Table 4.49: Post-hoc analysis of the DLs of HDDs in the three-way ANOVA analysis on CBTs of DTs of HDDs, DLs of HDDs and RMs applied. 283
Table 4.50: Posthoc analysis of the RMs in the three-way ANOVA analysis on CBTs of the DTs of HDDs, DLs of HDDs and RMs applied. 283
Table 4.51: Three-way ANOVA analysis on the CBTs of the *k*HTs, DTs of HDDs and RMs applied. 285
Table 4.52: Summary of empirical study findings and how they guided the system development methodology of the new *k*HT technique 287
Table 4.53: One-way ANOVA analysis on the EIVI of the newly developed *k*HT against the existing ones in a variety of HDDs 296
Table 4.54: Posthoc analysis on the *k*HTs on the one-way ANOVA analysis on the EIVI of the new *k*HT against the existing ones in a variety of HDDs. 296
Table 4.55: One -way ANOVA analysis on the CBTs of the newly developed *k*HT against the existing ones in a variety of HDDs. 297
Table 4.56: Posthoc analysis of *k*HTs in the one -way ANOVA analysis on CBTs of the new *k*HT against the existing ones in a variety of HDDs. 297

LIST OF FIGURES

Figure 1.1: Taxonomy of Clustering Algorithms. Source: Hussain & Haris (2019) 2
Figure 1.2: Application of *K*-means in Computational Anatomy by Clustering Lung Images into Normal, Covid-19, Bacterial Pneumonia and Viral Pneumonia. Source: Wang et al., (2021) 4
Figure 2.1: Standard Pseudocode of the *K*-means Clustering Algorithm 22
Figure 2.2: Elbow Method for Selecting the *K*-hyperparameter value 25
Figure 2.3: Diagrammatic Representation of an Autoencoder 45
Figure 2.4: Illustration of the Relationship between Independent, Moderating and Dependent Variables 135
Figure 3.1: Mapping of research objectives and questions to research methods 142
Figure 3.2: Activity Diagram for the First Set of Experiments 151
Figure 3.3: Activity Diagram for Validation Experiments 156
Figure 3.4: Architecture of the Ensemble based *k*-hyperparameter tuning technique on high-dimensional space using a self-adapting autoencoder and internal validation indexes 186
Figure 3.5: Research methodology for system development 204
Figure 4.1: Whisker-box plots of the different *k*-hyperparameter tuning techniques in a variety of high-dimensional datasets 298
Figure 4.2: Cluster visualization of the Tox-171 dataset using the new *k*HT and the standard autoencoder 301
Figure 4.3: Cluster visualization of the Tox-171 dataset using the new *k*HT and the improved autoencoder 301
Figure 4.4: Cluster visualization of the Yale dataset using the new *k*HT and the standard autoencoder 302
Figure 4.5: Cluster visualization of the Yale dataset using the new *k*HT and the improved autoencoder 302
Figure 4.6: Cluster visualization of the Reuters dataset using the new *k*HT and the standard autoencoder 303
Figure 4.7: Cluster visualization of the Reuters dataset using the new *k*HT and the improved autoencoder 303
Figure 4.8: Cluster visualization of the Lung Cancer dataset using the new *k*HT and the standard autoencoder 304

Figure 4.9: Cluster visualization of the Lung Cancer dataset using the new *k*HT and the improved autoencoder 304
Figure 4.10: Cluster visualization of the Covid-19 Coughs dataset using the new *k*HT and the standard autoencoder..... 305
Figure 4.11: Cluster visualization of the Covid-19 Coughs dataset using the new *k*HT and the improved autoencoder..... 305
Figure 4.12: Cluster visualization of the Heart beats sounds dataset using the new *k*HT and the standard autoencoder..... 306
Figure 4.13: Cluster visualization of the Heart beats sounds dataset using the new *k*HT and the improved autoencoder..... 306
Figure 4.14: Cluster visualization of the YouCook dataset using the new *k*HT and the standard autoencoder 308
Figure 4.15: Cluster visualization of the YouCook dataset using the new *k*HT and the improved autoencoder 309
Figure 4.16: Cluster visualization of the YouTube vlogs for cooking beef in Kenya using the new *k*-hyperparameter tuning technique and an improved autoencoder..... 310

LIST OF EQUATIONS

Equation 2.1: Mathematical formulation of elbow point in an elbow curve 26
Equation 2.2: Derivative of the Mathematical formulation of elbow point 26
Equation 2.3: Projection formula for PCA 34
Equation 2.4: Projection formula for Kernel PCA 36
Equation 2.5: Formula for Factor Analysis 37
Equation 2.6: Formula for Singular Value Decomposition 38
Equation 2.7: Weighted Least Squares equation for Locally Linear Embedding 42
Equation 2.8: Formula for low-dimensional embeddings on Locally Linear Embedding 42
Equation 2.9: Formula for encoding process on the autoencoder 43
Equation 2.10: Formula for decoding process on the autoencoder 44
Equation 2.11: Formula for the MSE loss function on the autoencoder 44
Equation 2.12: Formula for Linear Kernel Function 46
Equation 2.13: Formula for Polynomial Kernel function 46
Equation 2.14: Formula for Radial Basis Function 47
Equation 2.15: Formula for Sigmoid Kernel Function 47
Equation 2.16: Formula for Laplacian Kernel Function 48
Equation 2.17: Formula for Bessel Kernel function 48
Equation 2.18: Formula for ANOVA Kernel function 49
Equation 2.19: Dunn index formula 57
Equation 2.20: Calinski-Harabsz index formula 58
Equation 2.21: Inter cluster divergence formula for CI index 58
Equation 2.22: Inter cluster divergence formula for CI index 58
Equation 2.23: Davies Bouldin index formula 59
Equation 2.24: Silhoutte index formula 59
Equation 2.25: Bayesian Information Criterion index formula 61
Equation 2.26: Point bi serial formula 63
Equation 2.27: Sum of squares formula 64
Equation 2.28: Adjusted Rand index formula 66
Equation 2.29: Normal Mutual Information formula 67
Equation 2.30: V-measure formula 67
Equation 2.31: Harmonic Mean formula 68
Equation 2.32: Jaccard index formula 68
Equation 2.33: F1 score formula 70

Equation 2.34: Cochran’s Q statistic formula 71
Equation 2.35: Chi Square formula 73
Equation 2.36: T-test formula 73
Equation 2.37: Sum of squares error formula 74
Equation 3.1: Normalization formula 176
Equation 3.2: Ensemble index formula 176
Equation 3.3: Cosine similarity check formula 204
Equation 4.1: Kruskal-Wallis H statistic formula 293

ABBREVIATIONS & TERMS

ACM:	Association of Computing Machinery
AIC:	Akaike's Information Criterion
AIPS:	Average Index Performance Score
ANOVA:	Analysis of Variance
APA:	American Psychological Association
ARI:	Adjusted Rand Index
BIC:	Bayesian Information Criterion
CBT:	Cluster Building Time
CI:	Calinski harabsz Index
DB:	Davies-Bouldin index
DI:	Dunn Index
EDA:	Exploratory Data Analysis
EFA:	Exploratory Factor Analysis
EI:	Ensemble Index
FA:	Factor Analysis
FiBiNET:	Feature Importance and Bilinear feature Interaction NETwork
FS:	Folkes Russe
GPU:	Graphics Processing Unit
HD:	High-Dimensional
ICA:	Independent Component Analysis
IDE:	Integrated Development Environment
IEEE:	Institute of Electrical & Electronics Engineers
JC:	Jaccard Coefficient
kHT:	k-hyperparameter tuning Technique
KL:	Kull back
LLE:	Locally Linear Embedding
MFCC:	Mel-frequency Cepstral Coefficients
MSE:	Mean Squared Error
NLP:	Natural Language Processing
NMI:	Normalized Mutual Information
NOTK:	Number of iterations using *K*-means
PCA:	Principal Component Analysis
RI:	Rand Index

SI:	Silhouette Index
SSE:	Sum of squared errors
SVD:	Singular Value Decomposition
t-SNE:	t-distributed Stochastic Neighborhood Embedding
PCA:	Principal Component Analysis
PC:	Principal Components
RBF:	Radial Basis Function
SI:	Silhouette Index
TPU:	Tensor Processing Unit
t-SNE:	t-Distributed Stochastic Neighbourhood Embedding
UMAP:	Uniform Manifold Approximation and Projection
WCSS:	Within Cluster Sum of Squared distances
ZCA:	Zero Component Analysis

OPERATIONAL DEFINITION OF TERMS

Bayesian Inference Criterion: Bayesian inference refers to a statistical based framework that updates probabilities for hypotheses based on new evidence or data (Kodinariya & Makwana, 2013). It is founded on Bayes' theorem that relates conditional probabilities of events. In Bayesian inference, the posterior probability of a hypothesis is computed, given observed data, by combining both the prior beliefs (prior probability) with the likelihood of the data given the hypothesis. Finally, the resulting posterior probability becomes the updated belief (Kodinariya & Makwana, 2013).

Akaike's Information Criterion: Akaike's Information Criterion (AIC) is a statistical criterion applied in the selection and comparison of different models (Kodinariya & Makwana, 2013). AIC provides a way of assessing the trade-off balance between a model's goodness-of-fit to the data as well as its complexity, with an aim of choosing a model that effectively achieves a balance on these factors (Kodinariya & Makwana, 2013).

Autoencoder: Autoencoders are a type of neural networks applied in unsupervised learning and dimensionality reduction (Zhou et al., 2017). They aim to learn a compressed representation of input data, reffered to as the latent space (Zhou et al., 2015). They then reconstruct the original data from the representation on the latent space. Autoencoders typically comprise of two components i.e. encoder and decoder separated by a latent space (Jia et al., 2017). There are different variations of autoencoders, each with a specific objective.

For example, variational autoencoders are used for generative modelling (Aljalbout et al., 2018).

Calinski-Harabasz index: Calinski-Harabasz index, also reffered to as the Variance Ratio Criterion, refers to a metric used for evaluating the quality of clustering results (Kumar & Reddy, 2017). Calinski Harabsz index measures the ratio of the between-cluster variance to the within-cluster variance (Kumar & Reddy, 2017). This ratio indicates the separation between different clusters in a dataset. Higher scores on the Calinski-Harabasz index are an indication of better-defined clusters (Kumar & Reddy, 2017).

Centroid: Centroid, in the context of *k*-means clustering, refers to a data point that represents the center or mean of a cluster (Nazeer & Sebastian, 2009). The centroid point minimizes the sum of squared distances (Euclidean distances) between itself and all the data points in its cluster (Nazeer & Sebastian, 2009). The centroid is used to define each cluster in K-Means algorithm (Nazeer & Sebastian, 2009).

Cluster: Cluster, in the context of *k*-means clustering, refers to a combination or subset of *n* object that share similar characteristics. The combination of the *n* objects is normally from one group of N objects (Nazeer & Sebastian, 2009).

Cluster Building Time: Cluster Building Time is referred to as the amount of time taken for a clustering algorithm to generate clustering results from a set of N objects on a dataset (Kumar & Reddy, 2017).

Cluster inertia: Cluster inertia is the total squared error for each cluster (Li, 2015). It is also referred to as the "sum of squared" error (Li, 2015). Lower values of the cluster inertia demonstrates that the points are close while higher values of demonstrates that the points are further apart (Li, 2015).

Cluster validation: Cluster validation is the process of evaluating the quality of the clustering results generated by the clustering algorithms (Rendón et al., 2011).

Clustering: Clustering, in the context of *k*-means, is the process of grouping similar data points together in such a way that points in the same group (cluster) are more similar to each other than to those in other groups. (Nazeer & Sebastian, 2009).

Conceptual model: A conceptual model is an information system's framework that simulates and helps users to understand the real appearance and the working of the actual system looks like (Terell, 2012).

Cost function: Cost function, in the context of *k*-means, refers to the parameter that quantifies the overall dissimilarity or variance between data points and their assigned cluster centroids (Zhao et al., 2015). The cost function, also reffered to as an objective function, represents how well the clusters are fitted to the data (Zhao et al., 2015). K-means is an optimization problem where the algorithm seeks to find centroids that minimize the cost function (Zhao et al., 2015).

Cross entropy: Cross-entropy is a concept from information theory and machine learning that aims at measuring the difference between two probability distributions or the difference between predicted and actual outcomes (Guerin et al., 2017). Cross entropy is commonly used as a loss function in various machine learning tasks. It's used to assess the quality of the model's predictions and to update the model's parameters during training in order to minimize the difference between predicted and actual outcomes (Gueri et al., 2017).

Cross-validation: Cross-validation is a machine learning technique that is used to assess the performance of a model and estimate its ability to perform generalization on unseen data. It involves dividing the available dataset into multiple subsets inorder to train and evaluate the model for a multiple times. This strateghy assists in obtaining a more robust and reliable estimate of the model's performance (Kodinariya & Makwana, 2013)

Dataset: Dataset, in the context of machine learning, is a collection of structured data used for training, evaluating, and testing models in the machine learning space (Dilokthanakul et al., 2016). They are an essential machine learning components as they provide the input data on which models are trained to make predictions, classifications, clustering or other tasks. Datasets typically consist of features (input variables) and labels (target variables), and they come in various formats depending on the nature of the problem (Dilokthanakul et al., 2016).

Davies-Bouldin index: The Davies-Bouldin index is an internal validation metric that is used to evaluate the quality of the clustering results. It computes the average similarity between each cluster and its most similar cluster, considering both the size and dispersion of clusters. Lower Davies-Bouldin index scores are an indication of a better clustering as opposed to higher scores (Shah & Koltun, 2018).

Dimensionality: In machine learning space, dimensionality refers to the number of features or variables that are used to represent each data point in a dataset. In other words, it's the number of columns or attributes in a dataset. High dimensionality datasets have large numbers of features as compared to the low dimensionality datasets (Badrinarayanan et al., 2015).

Dunn index: The Dunn index is a metric used to evaluate the quality of clusters in data clustering tasks and aims to measure the compactness and separation between clusters in a clustering solution (Shah & Koltun, 2018). The Dunn index is defined as the ratio of the minimum inter-cluster distance to the maximum intra-cluster distance and quantifies the inter-cluster distance relative to intra-cluster distance (Shah & Koltun, 2018). The inter-cluster distance refers to how far apart the clusters are from each other while the inter-cluster distance refers to how compact the individual clusters are to each other (Shah & Koltun, 2018).

Elbow method: The Elbow Method is a graphical based benchmark technique for finding the optimal number of clusters, the *k*-hyperparameter value, in a dataset when performing a clustering analysis, such as K-means clustering (Kodinariya & Makwana, 2013).

It's a simple but effective way to determine a suitable number of clusters without relying solely on arbitrary choices or trial and error (Kodinariya & Makwana, 2013).

Embedded latent space: An embedded latent space is a lower-dimensional representation of data that captures meaningful features and relationships present in the original high-dimensional dataset (Saito & Tan, 2017). A latent space is not directly observable in the data, but it captures hidden or latent features that are not explicitly present in the original data. This lower-dimensional space is chosen or learned in a way that maximizes the amount of useful information while minimizing redundancy and noise (Saito & Tan, 2017)

High dimensional datasets: High-dimensional datasets refer to those datasets whose number of features, *P*, is greater, in one or several orders of magnitude, than the number of samples, *N*, i.e. $P>N$ (Bickel et al., 2009). High-dimensional datasets are common in various fields, including image analysis, genomics, text processing, and sensor data, among others (Bickel et al., 2009).

KL- divergence: The Kullback-Leibler (KL) divergence, also known as relative entropy, is a measure of the difference between two probability distributions. In the context of clustering tasks, KL divergence can be used to quantify the dissimilarity or difference between probability distributions that represent different clusters or groups of data points. In clustering tasks, KL divergence is often used in the context of evaluating the quality of a clustering solution or as part of a clustering algorithm (Brock et al., 2011).

Learning representation: Learning representation, in the context of machine learning and artificial intelligence, refers to the process of automatically discovering meaningful and informative features or patterns in raw data (Xie et al., 2019). The goal of learning representation is to transform the original data into a new, more compact, and more useful representation that captures the underlying structure of the data and is suitable for downstream tasks like classification, regression, clustering, and more (Xie et al., 2019). Effective representation learning can significantly improve the performance of machine learning models by enabling them to work with more relevant and discriminative features (Xie et al., 2019).

Low dimensional space: A low-dimensional space is a representation of data that has been reduced from its original high-dimensional form to a smaller number of dimensions (El-Mandpuh et al., 2019). In contrast to high-dimensional spaces where each data point is represented by a large number of features, low-dimensional spaces use a reduced set of features or dimensions to capture the most relevant information while discarding some of the less important details (Rathod & Garg, 2017).

Noisy data: Noisy data refers to data that contains errors, inconsistencies, or irrelevant information, which can negatively impact the quality of analysis, modeling, and decision-making (Ahmed, 2015). Noise can arise from various sources, including measurement errors, data entry mistakes, sensor inaccuracies, and environmental factors (Ahmed, 2015). Dealing with noisy data is a common limitation in data analysis and machine learning, and it requires careful preprocessing and handling techniques (Kodinariya & Makwana, 2013; Ahmed, 2015).

Optimization: Optimization, in the context of machine learning and data analysis, refers to the process of finding the best possible solution or configuration for a problem from a set of available options (Zhu et al., 2013). It involves adjusting parameters, variables, or settings to achieve a specific goal, such as minimizing an error function, maximizing a reward, or improving the performance of a model (Zhu et al., 2013).

Purity: Purity is a metric used to evaluate the quality of a clustering algorithm's performance, particularly in situations where there are ground truth labels available for the data (Shah & Koltun, 2018). Purity metric measures how well the algorithm's clusters match the true class labels of the data points and assesses the homogeneity of clusters, indicating how well the algorithm has grouped similar data points together (Shah & Koltun, 2018). It doesn't take into account the size or distribution of clusters. Purity is a simple and intuitive metric, but it has limitations and might not capture all aspects of cluster quality (Shah & Koltun, 2018).

Reconstruction Loss: Reconstruction loss, also known as reconstruction error, is a term commonly used in the context of autoencoders and other unsupervised learning techniques. It quantifies the difference between the input data and the output of a model's reconstruction process. The goal of minimizing the reconstruction loss is to make the model learn to reconstruct the input data as accurately as possible. (Dilokthanakul et al., 2016)

ABSTRACT

Clustering is one of the main goals of exploratory data analysis. It has an extensive and wealthy history in a variety of fields. The methods used to perform clustering have been evolving over time. Among these methods, *k*-means is still the most popular clustering algorithm because of its ability to adapt to new examples and to scale up to large datasets. It is also easy to understand and implement and is computationally faster and more efficient compared to other algorithms. However, with *k*-means, selecting the correct *k*-hyperparameter, i.e. the number of clusters in a dataset, has a long standing challenge and has a significant effect on the clustering results. Although a number of *k*-hyperparameter tuning techniques in high-dimensional space clustering have been proposed, to help in the selection of the correct *k*-value, these techniques still face performance limitations in a variety of high dimensional datasets and dimensionality reduction methods. This makes the *k*-hyperparameter tuning problem intractable and an open research challenge. In light of this, this research firstly aims at investigating the existing *k*-hyperparameter tuning techniques in high dimensional space clustering through the literature review analysis. Secondly, an investigation on the dimensionality reduction methods used with the high dimensional spaces is also done via the same process. The results of the first two steps provide key findings and a conceptual framework that acts as the road map and the foundation for the subsequent empirical investigations in the third step. These investigations are guided by a comprehensive methodology based on mixed research methods for validation triangulation. Experiments are conducted on techniques that demonstrate methodological rigour and novelty, in a variety of datasets and dimensionality reduction methods. Empirical research design guides the process of conducting these experiments. The invaluable insights based on the results' analysis of the experimental data, evinces the significance of the feature extraction process as a critical leverage point in the effective *k*-hyperparameter tuning process in high dimensions. This guides the implementation of a novel generalizable technique, through a multi-methodological system development methodology. This technique is then validated against the existing ones, using similar metrics, in order to evaluate its effectiveness. Statistical significance tests, using the ANOVA and the Kruskal-Wallis H statistic, demonstrate that the new technique is more superior. This is also evinced by the improved internal index scores, cluster visualizations as well as the presence of shorter whiskers and higher median (Q2) values in the whisker-box plots, in a variety of datasets. The new technique handles a variety of datasets, using an improved self-adapting autoencoder based on an unsupervised transfer learning strategy and a thoughtful configuration of both the architectural and training-related hyperparameter settings. This makes it effective in handling data sparsity and curse of dimensionality limitations inherent in high dimensional spaces. Future research aims at evaluating its efficacy in wider application domains, including a further comparative analysis of hybrid sets of best performing dimensionality reduction methods.

CHAPTER ONE: INTRODUCTION

1.1 Background Information

Clustering is one of the main goals of the exploratory data analysis and with an extensive and wealthy history in a number of disciplines (Franti & Sieranoja, 2018). Some of these include: Psychology, Mathematics, Engineering, Anthropology, Medicine, Biology, Statistics, and Computer Science (Aghabozorgi et al., 2015). A number of clustering algorithms have been suggested since the nineteenth century (Franti & Sieranoja, 2018). *K*-means is one such popular algorithm (Alsabti et al., 1997). This is because of its ability to produce tighter clusters as compared to other clustering algorithms (Khan et al., 2017). It is also simple to understand and implement (Celebi, 2014; Khan et al., 2017).

K-means is an unsupervised clustering algorithm that aim at partitioning *n* number of data objects into *k* clusters, where each data object belongs to the cluster with the nearest cluster centroid or mean (Shiudkar & Takmare, 2017). Data objects are aggregated into one cluster because they share common similarities (Gupta & Panda, 2019; Park et al., 2013). The number of clusters into which the data is partitioned is referred to as the *k*, a value that has to be supplied to the algorithm in advance (Pandey & Kumar, 2018; Qi et al., 2017). The *k* value is a global parameter in *k*-means and has a major effect on the clustering results (Ortego, et al., 2018). Selecting the correct *k* value from a given dataset, a process reffered to as the *k*-hyperparameter tuning, has a long standing challenge (Franti & Sieranoja, 2018).

There are different types of clustering algorithms. These include: partition-based clustering algorithms, hierarchy-based clustering algorithms, density-based clustering algorithms, grid-based clustering algorithms and the model-based clustering algotithms (Hussain & Haris, 2019).

The choice of a particular type of clustering algorithm depends on how well it scales to a particular dataset and its distribution (Hussain & Haris, 2019). K-means algorithms fall under the partition-based clustering algorithms (Hussain & Haris, 2019). In partition-based clustering algorithms, data objects are classified into multiple groups called clusters, based on the characteristics and similarity of the dataset (Hussain & Haris, 2019). Each partition contains data objects with common characteristics and represents a cluster (Hussain & Haris, 2019).

The number of clusters for a given dataset, the *k*-hyperparameter, is a value that users have to supply in advance, to the *k*-means algorithm. Figure 1.1 shows the taxonomy of the different clustering algorithms.

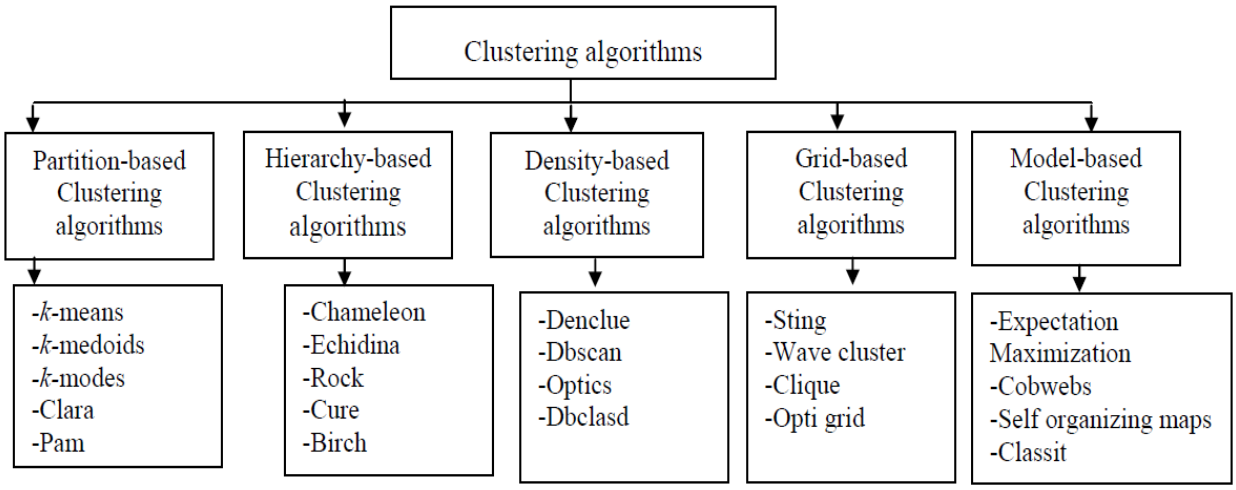


Figure 1.1: Taxonomy of Clustering Algorithms. Source: (Hussain & Haris, 2019)

The modern technology trends have proliferated the generation of massive high-dimensional datasets, both in volume and dimensionality, across diverse scientific disciplines (Yang & Shami, 2020). Examples include datasets in computational medicine, genomics, computer vision and beyond (Yang & Shami, 2020). K-hyperparameter tuning in high-dimensional space clustering poses several challenges to data scientists due to a number of issues.

The curses of dimensionality as well as the sparse and redundant nature of these datasets are such main issues (Dubey & Choubey, 2018). The current *k*-hyperprameter tuning techniques are faced with significant challenges in tuning for the *k*-hyperparameter value because of lack of effective strategies for extracting meaningful structures from such high-dimensional data spaces, among others (Dubey & Choubey, 2018). For example, the curse of dimensionality, the exponential growth of feature space in high dimensions, amplifies the computational complexity and diminishes the clustering quality. This makes the exhaustive exploration for the optimal *k*-hyperparameter value a computationally expensive process (Dubey & Choubey, 2018). Moreover, in high-dimensional data spaces, data points tend to be sparse, making it challenging for the existing *k*-hyperprameter tuning techniques to discern meaningful clusters and estimate the optimal *k*-hyperparameter value (Dubey & Choubey, 2018). Selection of an overly large *k*-hyperparameter value from a high dimensional data space leads to overfitting while the selection of a small *k*-hyperparameter value leads to underfitting. Overfitting is characterized by fragmented or trivial clusters while the underfitting results into merging genuine clusters or overlooking distinct patterns (Yang & Shami, 2020).

Computational medicine, a discipline that is devoted to quantitative approaches for diagnosis and treatment of human diseases, is one of the areas that have led to generation of massive datasets in high-dimensions (Yang & Shami, 2020). Computational Medicine comprises of Computational Molecular Medicine, Computational Anatomy, Computational Health-care, Computational Neuroscience and Computational Physiological Medicine. The adoption of effective *k*-hyperparameter tuning techniques to identify the correct *k*-hyperparameter value from such datasets is therefore of paramount importance (Dubey & Choubey, 2018). Clustering infected lung images is one of the successes of the application of the *k*-means algorithm in Computational Anatomy (Wang et al., 2021).

Inspired by the application of statistical, mathematical and data analytical methods to model human anatomy, this clustering algorithm detects the subtle differences from the computed tomography scans of lung patients and visualizes them into four optimal clusters, as shown in Figure 1.2. These include: Normal, Viral pneumonia, Bacterial pneumonia and Covid-19 infections (Wang et al, 2021). Detecting the correct number of clusters from such images using a naked eye is an arduous task for the radiologists and this can easily interfere with the accurate diagnosis (Wang et al., 2021)

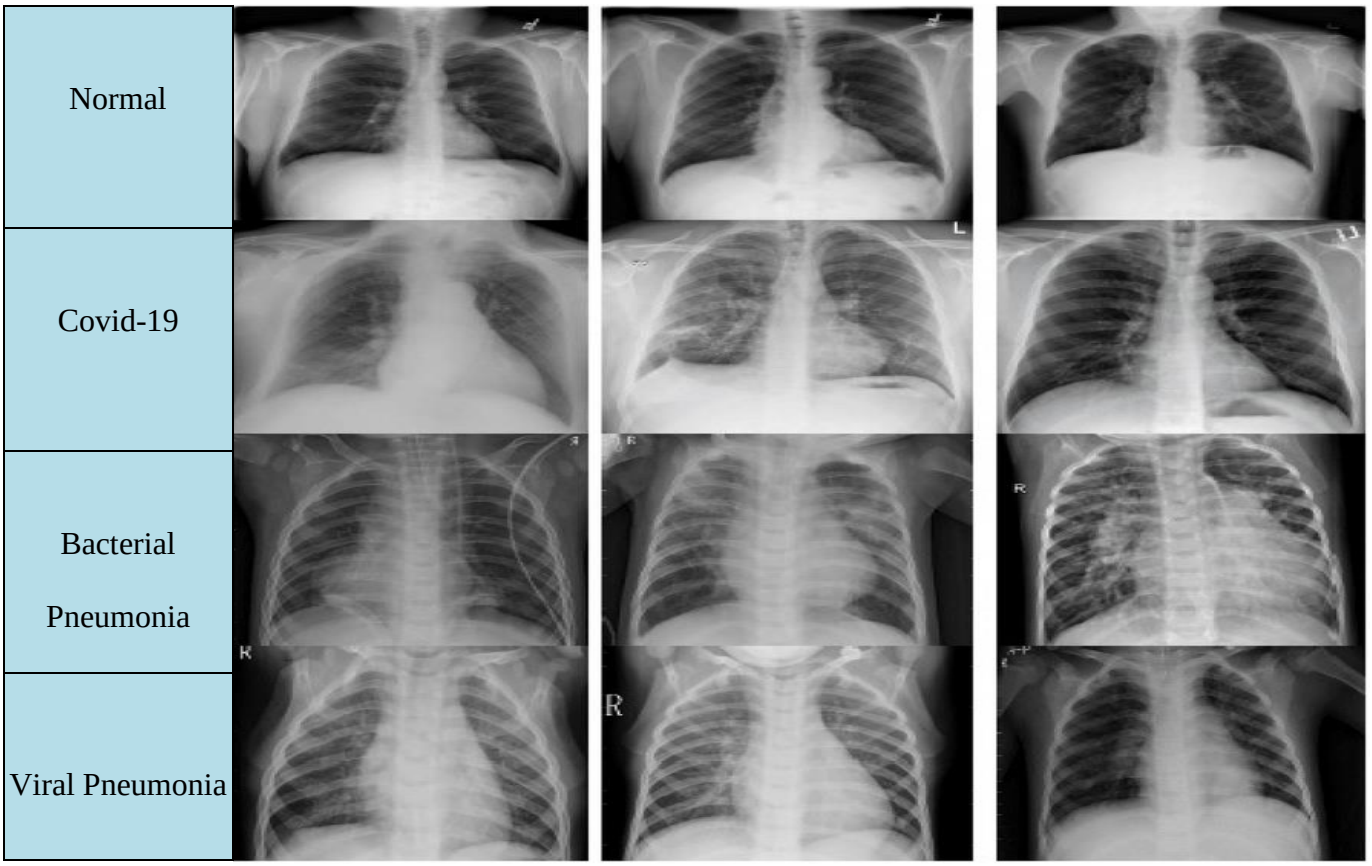


Figure 1.2: Application of *K*-means in Computational Anatomy by Clustering Lung Images into Normal, Covid-19, Bacterial Pneumonia and Viral Pneumonia. Source: (Wang et al., 2021)

1.2 Statement of the Problem

The prevalence of high-dimensional datasets, generated by high-throughput technologies, has become common in the modern era of technology (Moon et al., 2019; Onumanyi et al., 2022).

Clustering datasets in high dimensions has given rise to new computational challenges experienced by both the data scientists and the world of scholars (Kampman & Elomaa, 2018; Xia et al., 2020). The curse of dimensionality and data sparsity are the two such challenges (Kampman & Elomaa, 2018; Wainwright, 2019). K-means is a popular clustering algorithm. Although it has remained popular and attracted immense research interest, the identification of the correct *k*-hyperparameter value has been a long-standing challenge (Brodinová et al., 2019; Ikotun et al., 2022). The value of the *k*-hyperparameter must be provided in advance to the *k*-means algorithm, and its proper tuning has a significant impact on the clustering results, particularly in high-dimensional spaces (Chakraborty & Das, 2020; Yuan &Yang, 2019). Several variants of *k*-means, which incorporate *k*-hyperparameter tuning mechanisms in high-dimensional space clustering, have been developed to aid in determining the correct value of the *k*-hyperparameter (Yuan & Yang, 2019; Rezaee et al., 2021). However, due to the challenges of the curse of dimensionality and the data sparsity in the cluster analysis in high-dimensions, these techniques still face limitations in their *k*-hyperparameter tuning processes (Sanapala et al., 2023; Murugesan & Murugesan, 2020).

Onumanyi et al. (2022) proposed an AutoElbow, a *k*-hyperparameter tuning technique that applies the normalization strategy to tune for the *k*-value in high dimensional spaces. Through normalization, each feature contributes more equally to the distance calculations. Without normalization, features with larger scales in the high dimensional spaces may dominate the distance computations, leading to biased results and incorrect *k*-value. Normalization helps prevent this bias, allowing for more meaningful distance measures. In Autoelbow, the elbow graph is normalized using the lowest and the highest values along the coordinates of both the *Y*-axis and the *X*-axis.

The estimated elbow, also known as the *k*-hyperparameter, is the point on the graph that maximizes the distance between each point, the minimum and maximum reference points, as well as the "heel" of the elbow graph. Although this *k*-hyperparameter tuning technique performs relatively well, it has the limitation of the fact that in high dimensions, where complex non-linear relationships exist between variables, normalization might not adequately capture these relationships. This affects effective tuning of the *k*-value. Moreover, the auto-elbow graph may not depict a sharp elbow with some high-dimensional datasets.

This increases the variability in high-dimensional datasets, making it challenging to identify a clear "elbow point" in the within-cluster sum of squares (WCSS). The WCSS tends to decrease continuously as '*k*' increases, leading to a smoother curve with no distinct elbow (Saputra et al., 2020). This subsequently makes the identification of the correct *k*-hyperparameter value a challenging task when using this technique. For this technique to be effective, it is critical to devise a strategy that handles the inherent challenges in high dimensional spaces in a more effective manner and subsequently generate an elbow point that is sharp and clearly visible. This makes it possible to identify the *k*-value with ease.

Xia et al. (2020) proposed a technique for tuning the *k*-hyperparameter in high dimensional space clustering, using the neighbour searching strategy and a ball to represent clusters, divided into both stable and active regions. By defining clusters as balls, this minimizes the number of distance calculations needed for clustering, thereby reducing the computational load, especially in high-dimensional spaces where distance calculations become costly. By dividing each cluster into stable and active regions, the stable region holds centroids that are less prone to change, providing a foundation for the clustering process.

The active region, on the other hand, is more flexible and is further subdivided into equal portions of annular area in order to refine the clustering accuracy. This subdivision helps in refining the clustering process by focusing efforts on the specific regions where the data points are denser or where clusters need more distinction. This strategy helps to manage sparsity and addresses the curse of dimensionality by strategically focusing computational resources where they are most needed. Employing a neighbor searching strategy assists in identifying relevant data points within the defined clusters, allowing for a more nuanced understanding of the local density and cluster characteristics. This can be particularly helpful in high-dimensional spaces where sparsity might affect the *k*-hyperparameter tuning. However, this technique faces the limitation of dependency-on-parameter tuning effect. In specific, this effect involves parameters like the size of the balls, subdivision criteria, and the number of annular areas, which require proper tuning. Selecting appropriate parameter values can be non-trivial and might vary in a variety of datasets of different data types and dimensionality levels. Moreover, in the presence of noisy or outlier-laden high dimensional dataset, the ball *k*-means technique might struggle to define accurate cluster boundaries. Outliers can affect the centroid calculations and the stability of the cluster areas, impacting the *k*-hyperparameter tuning process and the subsequent quality of clustering.

Wang et al. (2019) proposed another technique for *k*-hyperparameter tuning using the "Fast Adaptive *k*-Means Subspace Clustering for High-Dimensional Data" and the PCA. In this technique, an adaptive loss is created to provide an adjustable cluster indicator computation approach for handling high-dimensional datasets with different distributions. By adapting to these distributions, it can provide a more robust *k*-hyperparameter tuning approach that is less affected by the curse of dimensionality challenge.

The technique performs both the feature extraction, using PCA, and clustering simultaneously, enhancing the quality of the clusters by focusing on the most informative features, while at the same time minimizing the impact of irrelevant and noisy dimensions. This counteracts the data sparsity challenge. However, using this technique can lead to excessive feature reduction, which in turn can affect the *k*-hyperparameter tuning process and subsequently degrade the quality of clustering results. The reduction of the original high dimensional space to an optimal number of features is a crucial parameter that needs to be tuned when using this technique. The adoption of a reduction strategy that strikes a balance between having enough features to capture meaningful patterns from a specific variety of a high-dimensional dataset, and not having too many features that could lead to computational and modeling issues like overfitting, is crucial for handling curse of dimensionality and data sparsity challenges when using this technique. This observation is also in line with the one made by Musco (2015) and Nguyen and Holmes (2019).

Brodinová et al. (2019) proposed a *k*-hyperparameter tuning technique in high dimensional space clustering using a "Robust and Sparse K-means clustering for High-dimensional Data." This technique uses a weighting function that assigns weights to each observation in the high dimensional space. By automatically assigning weights to observations, it aims to emphasize relevant data points and reduce the influence of noise, thereby handling data sparsity to some extent. The incorporation of a lasso-type penalty in the objective function promotes sparsity by encouraging some feature weights to be exactly zero, effectively performing feature selection. This strategy helps in reducing the impact of irrelevant or noisy dimensions during the *k*-hyperparameter tuning process. The mechanism for determining the number of clusters is achieved through a modified gap statistic. The gap statistic compares the within-cluster dispersion to that expected under a null reference distribution.

This helps in identifying the optimal number of clusters by considering the data's inherent structure while mitigating the curse of dimensionality. However, determining the sparsity parameter *s* for high-dimensional datasets poses computational challenges with this technique. The choice of this parameter affects the balance between sparsity and the information retention.

While reducing features can address the balance between the sparsity and information retention, excessive feature reduction might lead to loss of critical information, impacting clustering quality when using this technique. Feature sparsity parameter challenges in high dimensional spaces have motivated the use of effective feature reduction strategies to help capture the most relevant features from a specific data type of a certain number of features (dimensionality) and consequently support the *k*-hyperparameter tuning process (Bohn et al., 2016; Jia et al., 2022; Han & Ge, 2020). Therefore, it is crucial to make the best strategy for a specific variety of high-dimensional dataset in order to effectively deal with the high feature sparse parameter challenge when using this *k*-hyperparameter tuning technique.

Chakraborty and Das (2020) proposed a Lasso Weighted *k*-means, a variant of the *k*-means algorithm that incorporates a *k*-hyperparameter tuning mechanism in high-dimensional space clustering. It incorporates an *L1* regularization term directly on the feature weights. This regularization encourages sparsity by driving some feature weights to zero, effectively performing feature reduction. By assigning non-zero weights to only relevant features, it reduces the impact of irrelevant or noisy dimensions on the clustering process. By applying regularization on the feature weights, this technique induces dimensionality reduction within a sparse clustering framework. This process effectively reduces the influence of less informative dimensions, helping to handle the curse of dimensionality challenge.

However, deploying weights on the numerous features present in high-dimensional datasets presents challenges related to computational complexity. Therefore, the adoption of an effective feature reduction strategy with regularization is crucial when working with this specific technique (Prabhu & Anbazhagan, 2011).

Based on this, it is clear that the current techniques still face limitations in the *k*-hyperparameter tuning process in high dimensional space clustering. This is due to the inherent challenges of the high dimensional datasets, mainly the data sparsity and the curse of dimensionality. This observation is also in line with the one made by Berisha et al. (2021), Han and Ge (2020), Springenburg (2015), Musco (2015) and Nguyen and Holmes (2019). The effectiveness in handling these hurdles is critical to the identification of the correct *k*-hyperparameter in a variety of high dimensional spaces. This makes the *k*-hyperparameter tuning problem in high-dimensions intractable and an open research challenge. This is the motivation behind this research work.

1.3 Justification

Clustering is a significantly important process in variety of applications in real life. These applications cut across data mining, pattern recognition, data analysis, image processing etc. Considering the fact that the reliability of the clustering results obtained from the *k*-means clustering algorithms in high-dimensional datasets heavily depend upon the correctness of the *k*-hyperparameter value, the results of this research study are therefore important. These results enriches the data scientists' tool box, showing the most appropriate set of data dimensionality reduction methods and *k*-hyperparameter tuning technique for a specific variety of high-dimensional dataset.

Besides this, the experimental results form the basis for developing an improved *k*-means based technique for *k*-hyperparameter tuning in high-dimensional space clustering. The development of an improved internal validation index to evaluate *k*-means based techniques used in the *k*-hyperparameter tuning is also a valuable addition to the existing literature of cluster validation. Moreover, the future recommendations in this research work gives a strong foundation that inspires the conduct of research work that is geared towards solving the *k*-hyperparameter tuning problem in high-dimensional space clustering.

1.4 Research Objectives

1.4.1 General Objective

This research aims to empirically analyze the existing *k*-hyperparameter tuning techniques for high-dimensional clustering across various datasets and dimensionality reduction methods, with an aim of developing a novel technique based on the experimental insights.

1.4.2 Specific Objectives

The specific research objectives of this research includes, to:

i. Perform a theoretical and empirical analysis on the *k*-hyperparameter tuning techniques in high-dimensional space clustering.
ii. Develop an improved *k*-hyperparameter tuning technique in high-dimensional space clustering, based on the results of both the theoretical and empirical analysis.
iii. Evaluate the newly developed *k*-hyperparameter tuning technique in high-dimensional space clustering.

1.5 Research Questions

The specific research questions are then formulated as follows:

i. How do the *k*-hyperparameter tuning techniques in high-dimensional space clustering, compare, in their performance?

ii. How can an improved *k*-hyperparameter tuning technique in high-dimensional space clustering be developed, based on the results of the theoretical and empirical analysis?

iii. How effective is the newly developed *k*-hyperparameter tuning technique in high-dimensional space clustering?

1.6 Research Hypothesis

In this research study, both the null hypothesis (H_0) and the alternate hypothesis (H_a), as proposed by Shaffer (1995) are formulated as follows:

H_1: "There are no statistically significant differences in the performance of the *k*-hyperparameter tuning techniques in high-dimensional space clustering"

H_a: "There are statistically significant differences in the performance of the *k*-hyperparameter tuning techniques in high-dimensional space clustering"

1.7 Scope of the Study

The research focuses on the *k*-hyperparameter tuning problem in *k*-means clustering algortihms applied in high dimensional datasets, addressing challenges such as the curse of dimensionality and data sparsity. The study evaluates the efficacy of the existing *k*-huperparpameter tuning techniques in a variety of high dimensional datasets and dimensionality reduction methods, both theoretically and empirically. A novel technique is then developed based on the empirical insights.

Lastly, the effectiveness of the proposed novel technique is evaluated using the quantitative and qualitative metrics like ANOVA tests, Kruskal-Wallis H statistics, whisker plots as well as the qualitative cluster visualizations. This ensures a rigorous and comprehensive validation process including a check on the degree of alignment of both the quantitative as well as the qualitative validation methods.

1.8 Significance of the Study

The research study on the *k*-hyperparameter tuning in high dimensional space clustering is not about treading waters. It has significant implications and benefits to the field of *k*-means algorithms and the data analysis in high dimensional spaces. Improved *k*-hyperparameter tuning in high dimensional data clustering algorithms is an important aspect of this research. These data clustering algorithms have a wide variety of applications in real life, including data analysis, image processing, natural language processing, computational medicine and bioinformatics. In specific, the specific beneficiaries of this research study include: Artificial intelligence and big data companies, bioinformatics and genomics researchers as well as both the data scientists and machine learning engineers whose one of the major research interests is *k*-hyperparameter tuning on high dimensional spaces. A research study on the *k*-hyperparameter tuning in these application areas, therefore, leads to more accurate and efficient solutions since the choice of the number of clusters, the *k*-hyperparameter, is a fundamental decision when performing clustering. A better understanding of how to tune this *k*-hyperparameter value in high-dimensional spaces leads to the development of a more effective clustering algorithm that, in turn, leads to improved data organization and pattern recognition in these applications areas. Lastly, the scientific contribution of this research study to the body of knowledge and understanding of how *k*-hyperparameter tuning is achieved in high-dimensional spaces is also significant.

In specific, this research study is valuable for the advancement of the domain knowledge in the area of *k*-hyperparameter tuning in high dimensional space clustering, leading to new discoveries and insights by the future scholars focusing on this landscape.

1.9 Thesis Outline

Chapter one presents a brief introduction to the background of the study, problem statement, justification of the study, research questions, hypothesis, research objectives, conceptual framework and the research purpose and scope. Chapter Two presents literature review of the existing *k*-means based techniques used in the *k*-hyperparameter tuning in high-dimensional space clustering. Firstly, the fundamentals of *k*-means, structure of high-dimensional datasets and the challenges of tuning the *k*-hyperparameter value in high-dimensional datasets are discussed. During the review on the *k*-hyperparameter tuning techniques, the data dimensionality reduction methods used with them are also discussed. The metrics used in the evaluation of these techniques are discussed. At the end, the identification of gaps from the literature review and the key findings culminates this chapter. Chapter Three presents the methodology in this research work. The first part investigates on the current status of the existing techniques used in *k*-hyperparameter tuning in high dimensional space clustering, through a literature analysis. During this, both the data-dimensionality reduction methods and high-dimensional datasets used with these techniques are also investigated. The culmination of this analysis is a conceptual model that subsequently guided the empirical research design. The experimental results form the basis for the development of an improved new technique. This development is guided by a multi-methodological system development methodology. At the end, the effectiveness of the newly developed technique is evaluated against the existing ones using a similar set of high-dimensional datasets and evaluation metrics. Chapter Four presents the experimental results based on the experimental design methodology.

In this methodology, the feasibility of the empirical study, using a pilot study, is initially conducted based on the conceptual framework. The results of the various experiments conducted on the different *k*-means based techniques, in a variety of data dimensionality reduction methods and high-dimensional datasets are presented. The experimental results are used as the basis for developing the new *k*-hyperparameter tuning technique. ANOVA tests are used for the analysis of the statistical significance tests and the interactions that results from the manipulation of the experimental variables. The effectiveness of the newly developed technique, for the *k*-hyperparameter tuning in high-dimensional space clustering, is then evaluated against the existing ones using a similar set of high-dimensional datasets and evaluation metrics. Kruskall-Wallis H statistic, the ANOVA test as well as an improved ensemble internal validation index are performance scores that are used during this evaluation process. Chapter five presents the discussions based on the experimental results. Chapter six concludes the thesis, highlighting both the technical and theoretical contributions as well as research limitations. The recommendations for future research work, in regards to the *k*-hyperparameter tuning in high-dimensional space clustering, are also suggested in this chapter.

2.0 Language Conventions

In this study, a number of sets of words and phrases are used interchangeably. "*K* hyperparameter tuning technique", "*K* means variant that incorporates a *k*-hyperparameter tuning mechanism" and "*K*-means based techniques used for *k* hyperparameter tuning" are used interchangeably. "Unsupervised learning" is used interchangeably with clustering. Data object is used interchangeably with data point. "Number of features" is used interchangeably "dimensionality level". "High-dimensional dataset" is used interchangeably with "high-dimensional space".

Elbow is used interchangeably with Knee. Attribute is used interchangeably with Feature and Column. “Cluster building time” is used interchangeably with “run time”. Limitations is used interchangeably challenges. Lastly, the AIPS is used interchangeably with the EIVI.

CHAPTER TWO: LITERATURE REVIEW

2.1 Introduction

The literature review focuses on a comprehensive exploration of the *k*-hyperparameter tuning techniques in high-dimensional space clustering. Firstly, the structure of the *k*-means algorithms as well as their related hyperparameters is discussed. Each technique's strategy, based on the clustering theory, is then evaluated for its effectiveness in tuning for the *k*-hyperparameter value under the constraints and challenges inherent in the high-dimensional spaces. The comparative study, based on a theoretical analysis, presents a number of the *k*-hyperparameter tuning techniques and dimensionality reduction methods, offering insights into their relative strengths and limitations. These dimensionality reduction methods and their variations, including Principal Component Analysis (PCA), t-distributed Stochastic Neighbor Embedding (t-SNE), and Uniform Manifold Approximation and Projection (UMAP), are explored as crucial tools to alleviate the impact of the cluster analysis challenges in high dimensional spaces during the *k*-hyperparameter tuning process. The literature also delves into the metrics used to evaluate the effectiveness of the *k*-hyperparameter tuning techniques, including the common metrics such as the Silhouette index, Davies-Bouldin index, Calinski Harabsz index, Dunn index, among others. The discussion on these metrics is done in the context of their relevance and reliability in the evaluation process. The culmination of the literature review process is a discussion on the results and how the key findings from the literature analysis guide both the experimental and the system development methodology, through a conceptual framework. A summary, at the end, lays the groundwork for future research directions in the *k*-hyperparameter tuning problem in high-dimensional spaces. This chapter is organized as follows: Section 2.2 describes the theory behind clustering. Section 2.3 introduces the *k*-means algorithm, its fundamental structure, hyperparameters and the traditional elbow method for tuning the *k*.

Section 2.4 discusses the structure of high-dimensional datasets and their cluster analysis challenges. Section 2.5 discusses the limitations of tuning the *k*-hyperparameter in high-dimensional spaces. Section 2.6 discusses the dimensionality reduction methods used in a variety of high dimensional spaces. Section 2.7 discusses the metrics applied in the evaluation of the *k*-hyperparameter tuning techniques in high-dimensional space clustering. Section 2.8 reviews the techniques, highlighting both their successes and limitations. At the end of this review, the key findings from the literature analysis are highlighted including how they guided the methodology. Section 2.9 discusses the conceptual framework, research variables and the operationalization of these variables. Lastly, section 2.10 gives the summary of this chapter.

2.2 Clustering Theory

The theory of clustering is a fundamental concept in data analysis and machine learning, and encompasses the principles and methodologies behind the process of grouping data points into clusters, based on their similarity or dissimilarity. Clustering theory is widely applied in various domains, including data analysis, image segmentation, natural language processing, customer segmentation, anomaly detection, and much more. Understanding the principles and concepts in the clustering theory is essential for selecting the appropriate clustering algorithm, selecting for the optimal *k*-hyperparameter value, and interpreting the results of clustering analysis. The selection of the optimal *k*-hyperparameter value is based on the specific characteristics of the dataset and the goals of the data analysis. Clustering is an unsupervised learning problem, meaning that it does not rely on labeled data. Instead, it aims at discovering the natural patterns, structures, or groupings within an unlabelled dataset (Hartigan, 1985). It is a good practice to ground any research on an existing theory in order to provide the context of the respective research study (Creswell & Poth, 2016).

2.3 Fundamentals of the *K*-means Clustering Algorithms

K-means clustering algorithm, sometimes called Lloyd-Forgy method, was developed by James MacQueen in the year 1967 (Celebi et al., 2013). It was developed as a simple centroid-based method for clustering objects (Naeem & Wumaier, 2018). Over time, it has become one of the most famous clustering algorithms (Perez-Ortega, 2018). *K*-means algorithm, when given a data set, loops continuously until it finds the *k* centroids (Nguyen & Duong, 2018). After this, every object in the dataset is assigned to the neighboring centroid, until the average of the coordinates of the N dataset objects, in each cluster, stabilizes (Dubey & Choubey, 2018). *K*-means clustering algorithm is relatively efficient in regards to computational times for high dimensional spaces. However, it is sensitive to the selection of the *k*-hyperparameter value, at the first instance (Chinthra, 2017). *K*-Means Clustering is a strategy used to group unlabeled *N* number of objects in an unlabeled data set into *n* number of objects in smaller separate clusters (Yuan & Yang, 2019; Lu et al., 2018). *K*-means clustering algorithms takes the *k* number of clusters and its arrangement of centroids as hyperparameters (Bala et al., 2015; Sandhya & Sekar, 2018; Zhao et al., 2018). The separation of all objects in the dataset is figured out with each of the centroids of the particular cluster (Zhong et al., 2015). This separation is referred to as the Euclidean Distance (Bao & Huang, 2017).

The aim of the *k*-means clustering algorithm is to get an extremely small value of the squared difference between the cluster centroid and the one in the unlabeled dataset (Raissi et al., 2017; Salve & Sinha, 2017). The number of clusters that are required must be selected first (Dubey & Choubey, 2018). This is assumed to be the number of clusters, i.e. the *k*-hyperparameter value (Dubey & Choubey, 2018). The next step requires the selection of the centroids for each of the sets of the *k* clusters.

The third step involves considering every component of the given set and contrasting its separation with every other centroid of the *k* clusters (Dubey & Choubey, 2018). The component is added to the cluster whose centroid is nearest to the component based on the separation (Dubey & Choubey, 2018). Cluster centroids are re-computed after every task or after an arrangement of the assignments (Sharma & Suji, 2016). This strategy is iterative and persistently refreshed until stability is achieved (Beaulieu-Jones & Greene, 2016; Sharma & Suji, 2016; Ulloa, 2017).

2.3.1 Structure of the *K*-means Algorithms

The *k*-means algorithm is an iterative clustering technique used to partition a dataset into *k* clusters based on their similarity (Dwivedi & Bhaiya, 2019). It is a widely used unsupervised machine learning algorithm (Dwivedi & Bhaiya, 2019). The first step of the *k*-means is initialization where the number of clusters, *k,* to be formed is chosen (Dwivedi & Bhaiya, 2019). At the initialization step, the *k* cluster centroids are also randomly initialized with each centroid representing the center of a cluster. The second step involves assigning data points to clusters (Dwivedi & Bhaiya, 2019). In this stage, each data point in the dataset is iterated through while at the same time computing the distance between each data point and each cluster centroid. After this, the data point to the cluster whose centroid is closest to it (based on distance) is assigned. This step creates *k* clusters. The third step involves recomputing cluster centroids (Dwivedi & Bhaiya, 2019). After all data points are assigned to clusters, the new centroid for each cluster is computed. The new centroid is the mean (average) of all the data points assigned to that cluster. This step moves the cluster centers to the mean of the data points in each cluster. The fourth step is the convergence check that investigates if the algorithm has converged, i.e., if the cluster assignments remain unchanged or changes minimally between iterations (Dwivedi & Bhaiya, 2019).

Convergence means that the clusters are stable, and the algorithm can stop. The fifth step is the iteration where if the algorithm has not converged, both the second and the third steps are repeated until convergence is achieved. In each iteration, the data points are reassigned to clusters based on the current centroids, and new centroids are calculated based on the updated cluster assignments.

The final result is the last step of the *k*-means where after the algorithm converges (the clusters are stable), the final cluster centroids and assignments are obtained (Dwivedi & Bhaiya, 2019). At this stage, the data points are partitioned into *k* clusters, and each data point belongs to the cluster with the closest centroid (Dwivedi & Bhaiya, 2019). It's important to note that the initialization step can have an impact on the final clustering result (Dwivedi & Bhaiya, 2019). The *k*-means algorithm may converge to different local optima depending on the initial placement of the centroids (Dwivedi & Bhaiya, 2019). To mitigate this, it is common to run the *k*-means algorithm multiple times with different initializations and choose the clustering result with the lowest cost (sum of squared distances from data points to their respective cluster centroids) (Dwivedi & Bhaiya, 2019).

K-means is sensitive to the choice of the *k*- hyperparameter value, the number of clusters (Gikera et al., 2020). This is the biggest challenge that a typical *k*-means user today would face (Gikera et al., 2020). There are various methods, such as the elbow method and silhouette analysis, that have a long standing literature and success in the determination of the optimal *k*-hyperparameter value for a given dataset. Figure 2.1 is a standard pseudo code of the *k*-means clustering algorithm (Dwivedi & Bhaiya, 2019).

1. Start: *The algorithm starts.*
2. Input: *The dataset and the number of clusters (k) are provided as input to the algorithm*.
3. Initialization: *Randomly or strategically initialize k cluster centroids*.
4. Iteration Loop:
    - Assign Data Points to Clusters: *Iterate through each data point in the dataset and calculate the distance to each centroid. Assign the data point to the cluster with the nearest centroid.*
    - Re-compute Cluster Centroids: *Calculate new centroids by taking the mean of all data points in each cluster.*
    - Convergence Check: *Check if the algorithm has converged. If the centroids have not changed significantly, exit the loop. Else, go back to the start of step 4.*
5. Output: *After the algorithm converges, we obtain the final cluster centroids and the clustering result.*
6. Stop: *The algorithm stops*.

Figure 2.1: Standard Pseudocode of the *K*-means Clustering Algorithm

2.3.2 K-means Hyperparameters

Hyperparameters in machine learning are parameters that are set before the learning process begins and control various aspects of the learning algorithm (Probst et al., 2019). Unlike model parameters, which are learned during training, hyperparameters are usually set by the data scientist or a machine learning engineer based on their expertise, experience, and experimentation (Probst et al., 2019). The choice of hyperparameters can significantly impact the performance and generalization ability of the model (Jia & Song, 2016).

K-means hyperparameters provide control over the behavior of the *k*-means algorithm and its final output (Blomer et al., 2016). Although *k*-means is a simple clustering algorithm, it does have a couple of hyperparameters that can affect its performance and results (Blomer et al., 2016). These include the number of clusters *k*, maximum iterations, initialization method, and tolerance (Thorpe et al., 2015).

The number of clusters *k* is one of the most important hyperparameters in the *k*-means algorithm (Kant & Ansari, 2016). The *k*-value determines the number of clusters the algorithm will create. The choice of the *k*-hyperparameter value depends on the nature of the dataset and the problem being solved. Selecting the correct *k*-value is crucial, as too few or too many clusters can lead to suboptimal results (Patil & Baidari, 2019).

The maximum number of iterations, or max iterations, aims at avoiding infinite loops where the algorithm is defined to stop if convergence is not reached within the specified iterations (Li, 2015). The algorithm iteratively assigns data points to clusters and updates centroids until convergence. However, in some cases, convergence may take a long time or may not be achieved at all (Li, 2015).

The initialization method refers to the way in which the initial centroids are selected, which can impact the final clustering results as well. The default approach is to randomly initialize the centroids, but there are other methods like *k*-means++ that can be used to choose initial centroids more strategically, improving the chances of finding better solutions (Li, 2015).

Lastly, tolerance, the minimum amount of centroid movement between iterations that is considered as convergence, is also another hyperparameter in the *k*-means algorithm (Li, 2015).

Experimenting with different values for these hyperparameters and evaluating the clustering results using appropriate evaluation metrics can help find the best set of hyperparameters for a specific dataset and clustering task at hand (Gikera et al., 2023a). Each algorithm may have its own set of hyperparameters, and choosing the right values for them often requires experimentation and tuning to achieve optimal performance for a given task and dataset (Probst et al., 2019). If the centroid movements fall below this tolerance value, the algorithm assumes convergence and stops (Li, 2015). A smaller tolerance value leads to more precise clusters but may require more iterations on the other hand (Dasgupta, 2003; Li, 2015)**.**

2.3.3 Traditional Elbow Method in *K*-hyperparameter Tuning

The elbow method is a long standing technique that has been successfully used to determine the optimal *k*-hyperparameter value in the *k*-means clustering algorithm (Syakur et al., 2018). It helps to find a suitable value for *k* by analyzing the relationship between the number of clusters and the within-cluster sum of squared distances (WCSS) (Bholowalia & Kumar, 2014). The elbow method involves five steps (Bholowalia & Kumar, 2014). The first step involves running the *k*-means algorithm for different values of *k,* typically in a range from one to a predefined maximum value. In the second step, the sum of squared distances from each data point to its assigned cluster centroid is computed, for each value of *k.* This is the within-cluster sum of squared distances (WCSS) for that value of *k*. The third step involves plotting a graph with the number of clusters (*k*) on the x-axis and the corresponding WCSS on the y-axis. Step four involves looking for the the "elbow" point on the graph.

The elbow is the point where the WCSS starts to level off and represents the value of *k* where adding more clusters does not significantly reduce the WCSS. Lastly, the value of *k*-hyperparameter is choosen at the elbow point as the optimal number of clusters for the *k*-means algorithm. In the figure below, the elbow point, indicated by the arrow, is the point where the curve starts to level off, indicated by the "+" symbol (Bholowalia & Kumar, 2014). In this case, the optimal *k* –hyperparameter value is chosen as the number of clusters at the knee. Figure 2.2 shows the elbow method (Bholowalia & Kumar, 2014).

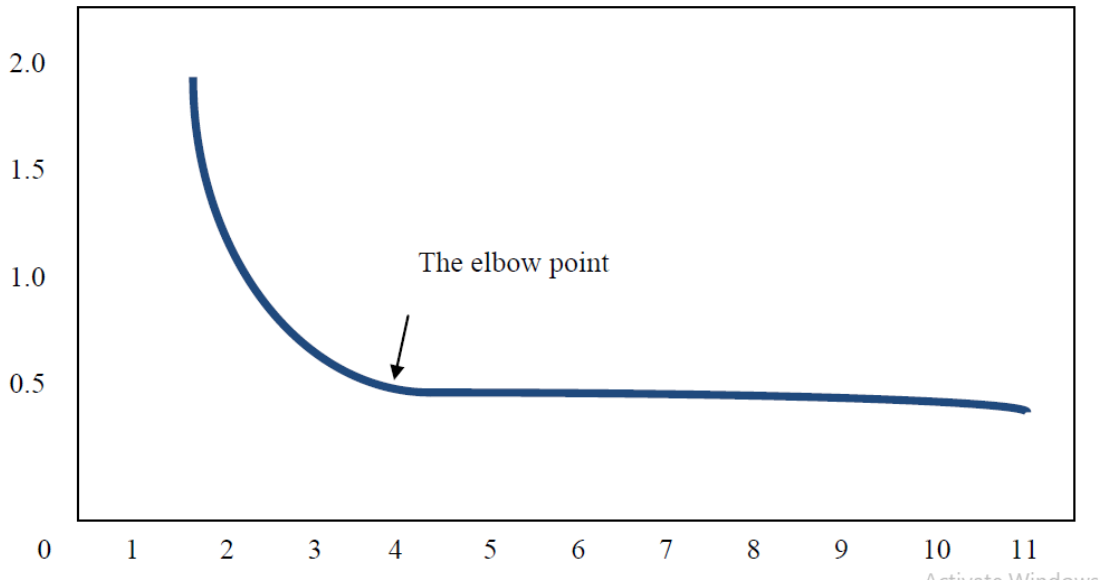


Figure 2.2: Elbow Method for Selecting the *K*-hyperparameter value.

It's important to note that the elbow method is not always definitive, and the choice of *k* can sometimes be subjective (Bholowalia & Kumar, 2014). In some cases, there might be a smooth unclear elbow point, or the plot might have multiple potential elbow points (Bholowalia & Kumar, 2014). In such situations, domain knowledge and other clustering evaluation metrics (like silhouette analysis) can also be used to help make an informed decision on the appropriate value of *k* (Ismkhan, 2018). Adopting a mathematical equation in the definition of an elbow is because of the fact that defining an elbow point in one situation may not be equally a good elbow point in another situation (Bholowalia & Kumar, 2014).

For this reason, authors have adopted the mathematical definition of the curvature on a function as the foundation of the mathematical formulation of an elbow point in an elbow curve (Bholowalia & Kumar, 2014). This is defined as follows:

$$L_{K=}\frac{f''(k)}{((1+f'(k)^2)^{3/2}} \quad (2.1)$$

Where L*k* is the curvature of the function *f(x)* at the point *x*, *f'k* is the first derivative of the function f(x) evaluated at *x* = *k* and represents the slope of the tangent to the curve at the point *(k, f(k)).* The f''k is the second derivative of the function f (x) evaluated at *x* = *k* and represents the concavity or convexity of the function at the point *(k, f(k)).* The $((1 + f'(k)^2)^{3/2}$ represents the normalization factor that accounts for the slope of the curve at the point *(k, f(k))* and ensures that the curvature is independent of the scaling of the *x*-axis and reflects the intrinsic curvature of the graph of the function. The elbow point, is the optimal number of clusters *k*, is the point of the highest curvature. It is the mathematical measure of the extent to which the function varies from a straight line (Bholowalia & Kumar, 2014). The maximum curvature is calculated as the derivative of the above equation and is taken as the optimal *k*-hyperparameter value (Bholowalia & Kumar, 2014). This derivative is stated as follows:

$$L'f\,(k) = f''(k)(1+ f'(k)2)\text{-}3\ f''(k)2\ f''((k)2\ f'(k)=0 \quad (2.2)$$

2.4 High dimensional Datasets and the associated Cluster Analysis Challenges

The modern technology trends have led to the generation of massive high-dimensional datasets (Brodinová et al., 2019). The study of high dimensional statistics has attracted immense interest from the world of scholars and data scientists (Hess & Duivesteijn, 2019).

The high-dimensional datasets refer to those datasets whose number of features / attributes, *P*, is greater than or close to, in one or several orders of magnitude, the number of instances / observations, *N*, i.e. $P > N$ (El-Mandpuh et al., 2019; Gikera et al., 2023a). Mathematically, "orders of magnitude" refers to a system of classification determined by size, typically in powers of ten (Gikera et al., 2023a). This is contrary to the belief that high-dimensional datasets is similar to big data comprised of a high number of instances (Brodinová et al., 2019).

For example, a dataset that has five features / attributes ($P=5$) and four instances / observations ($N=4$) would be considered a high dimensional dataset while a dataset with twenty thousand features (P=20,000) and eighty thousands instances /data points ($N=80,000$) would be considered low dimensional big data (Gikera et al., 2023a). It is mostly common to find high dimensional datasets in the field of medicine (Brodinová et al., 2019). An example is where the numbers of attributes for a particular patient are many i.e. body-mass index, blood pressure values, diagnosis history, family history on illnesses, height, weight, status of immune system etc. (Wang et al., 2016). In genomics and proteomic, each sample can be defined by multiple measurements up to a thousand (Khanmohammadi et al., 2017). Computational molecular medicine, computational anatomy, computational health care, computational neuroscience and the computational physiological medicine are some of the most common areas that demonstrate heavy utilization of the high dimensional datasets in their analysis (Khanmohammadi et al., 2017). Computational molecular medicine aims at constructing a deep understanding of the molecular networks through gaining insights to the concentrations of biomolecules and their time-based variations that assist in more informed clinical decisions (Khanmohammadi et al., 2017).

Computational physiological medicine aims at creating disease models that put together information from a number of levels of biological organization i.e. molecules, cells, tissues and organ systems, and apply these computational models to the patients' care (Chowdhury et al., 2022). Computational anatomy aims at utilizing the mathematical theories to model structures of the anatomy and their variations in health and disease (Chowdhury et al., 2022). An example of computational anatomy is the identification of changes in the shape and motion of a heart and using this data to characterize cardiac diseases (Chowdhury et al., 2022).

Computational healthcare integrates biomedical signal processing computational modeling, machine learning and health informatics in order to create new strategies in personalized medicine through e-health records physiology-based time series data as well as genomics (Chowdhury et al., 2022). Both the computational molecular medicine and computational healthcare apply the highest number of high dimensional datasets in their models (Chowdhury et al., 2022). The curse of dimensionality, sparse and redundant natures of the high-dimensional datasets pose the greatest data mining challenges to data scientists (Chowdhury et al., 2022). An "enhanced deterministic *k*-means clustering algorithm for cancer subtype prediction from gene expression data" is an example of a *k*-means based computational medicine based model that has been successfully applied in cluster analysis (Haraty et al., 2015). Table 2.1 is a structural example of a high dimensional dataset based on clinical data. The high dimensionality of the clinical datasets as opposed to other domains is due to the complexity of biological systems, the detailed nature of diagnostic and imaging data, and the increasing use of both the longitudinal and the multimodal data coming from multiple sources including the popular genetic profiles (Khanmohammadi et al., 2017)

Table 2.1 Example of a High Dimensional Dataset from a Clinical Data

P / N	BP	Height	Weight	Diagnosis	…	…
Patient 1						
Patient 2						
Patient 3						

The common types of such datasets include text, images, audio and video (Haraty et al., 2015; Gikera et al., 2023*a*). Most high-dimensional datasets exhibit both linear and non-linear characteristics (El-Mandpuh et al., 2019). For this reason, it is important to adopt data dimensionality reduction methods that are able to capture both the linear and non-linear relationships inherent in such datasets (Kung, 2014). At the same time, standard data dimensionality reduction methods do not usually perform well across the different data types. The use of variations, extensions or the incorporation of the kernel functions into the standard data dimensionality reduction methods makes them more effective in handling specific datasets (Kung, 2014). Although these variations and kernel functions come with computational challenges, they are versatile, providing a powerful way of capturing and exploiting non-linear relationships present in high-dimensional datasets (Kung, 2014).

2.5 Challenges of Tuning the *K*-hyperparameter in High Dimensional Spaces

Cluster analysis, the process of unveiling meaningful and hidden patterns in datasets, poses a number of challenges for the data science researchers (Chakraborty & Das, 2020). When performing clustering in high dimensional spaces using the *k*-means clustering algorithms, users experience a number of challenges which have had a heavy influence on the appearance of clusters (Syakur et al., 2018; Thorpe et al., 2015; Xing & Gu, 2014).

Some of these challenges include: curse of dimensionality, overfitting, computational complexity, feature extraction, visualization, data sparsity and feature engineering (Chakraborty & Das, 2020). As the number of dimensions increases, the volume of the space increases in an exponential manner. This leads to the curse of dimensionality effect where distances between data points become less meaningful and sparse, making it challenging to find meaningful patterns (Brodinova et al., 2019). Overfitting is where a model learns noise in the data rather than the underlying patterns. Models can become overly complex when they have too many features, making them less generalizable to new, unseen data (Brodinova et al., 2019). As the number of dimensions increases, the computational requirements for training and evaluating models also increase (Chakraborty & Das, 2020). High-dimensional datasets can be computationally expensive to process and may require more memory and processing power. Dealing with high-dimensional data often involves selecting relevant features (feature selection) or performing dimensionality reduction techniques (feature reduction). Feature selection involves choosing a subset of the most relevant features, while the dimensionality reduction techniques like Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE) aim to transform the data into a lower-dimensional space while preserving important information (Chakraborty & Das, 2020).

Visualizing high-dimensional data is challenging, as human brains can only perceive three dimensions directly (Chakraborty & Das, 2020). Techniques like dimensionality reduction are used to project data into lower-dimensional spaces for visualization purposes (Brodinova et al., 2019). In high-dimensional spaces, data points can become sparse, meaning that most data points are far apart from each other. This can affect the performance of clustering algorithms that rely on the density of data points. The quality of features is crucial in machine learning. High-dimensional data might contain redundant, irrelevant features, noise and outliers (Brodinova et al., 2019). This can negatively impact the performance of clustering models.

Feature engineering involves creating informative, relevant, and discriminative features (Chakraborty & Das, 2020). The choice of the *k*-hyperparameter value makes most of the high-dimensional *k*-means algorithms sensitive to the clustering performance (Chakraborty & Das, 2020). In data dimensionality reduction methods, there is need to adopt those methods that efficiently represent a high-dimensional dataset in low-dimensions, with as much minimal information loss as possible (Brodinova et al., 2019). Some *k*-means algorithms are sensitive to both the dimensionality redundancy and biasing and the need to invent algorithms that are independent of such limitations is critical to the success in this area (Brodinova et al., 2019). Due to these challenges, the identification of the optimal *k*-hyperparameter value in high-dimensional clustering is still a challenging task motivating the researchers to invent mechanisms that aid in more accurate methods of identifying this *k*-value (Brodinova et al., 2019). The correct identification of the *k*-hyperparameter value has a significant effect on the performance of the high-dimensional *k*-means models (Chakraborty & Das, 2020).

2.6 Dimensionality Reduction Methods in High Dimensional Space Clustering

In the recent past, high volumes of data are being generated from different sources. The size, complexity and dimensionality of this data have continually increased exponentially (Ayesha et al., 2020). Dimensionality reduction methods convert the high dimensional data spaces into low dimensional data spaces ahead of the clustering process by the clustering algorithms. The learning process of the machine learning models, in high dimensional spaces, is a difficult task with high computational complexity. If the size of data for training a machine learning model is fixed, raising the dimensionality will lead to over fitting. This is called the curse of dimensionality (Ma & Zhu, 2013); (Patil & Karthikeyan, 2020). In order to increase the efficiency of a machine learning model in a high dimensional dataset, it is imperative that it is pre-processed by an efficient and the most suitable data dimensionality reduction method (Ayesha et al., 2020).

Dimensionality reduction process can be accomplished through feature selection or feature extraction (Jia et al., 2022; Velliangiri & Alagumuthukrishnan, 2019). The feature extraction process converts data points in the original high space into newly reduced data points through eliminating redundant and irrelevant features (Reddy et al., 2020). On the other hand, feature selection picks a subset of features from the original high dimensional data space, i.e. the ones that are more relevant to the problem (Jung et al., 2017). Dimensionality reduction methods play a critical role in data compressions (efficient information storage and retrieval) and noise removal (Sorzano et al., 2014). Other advantages of the data dimensionality reduction methods include: lesser training time of a machine learning model as well as computational resources due to lower number of dimensions which in turn improves the algorithm's performance, reduced over fitting problems on the training datasets, improved data visualization since it's very hard to visualize data in high dimensions, lower levels of noise and outliers on a dataset as well as an aid in the transformation of non-liner data into a liner data (Springenburg, 2015). Essentially, a data dimensionality reduction technique converts a high dimensional space $X = [x_1, x_2, \cdots, x_m] \in \mathbb{R}^{m \times p}$,containing p dimensions and m observations, into low dimensional data $Y = [y_1, y_2, \cdots, y_m] \in \mathbb{R}^{m \times k}$, where ideally k (the reduced number of dimensions) is significantly smaller than p (Sorzano et al., 2014). Here, the X represents the high dimensional dataset containing m data points (observations) and p features (dimensions). On the other hand, Y is the lower dimensional dataset containing the same m data points but only k features where $k<<p$. In an ideal scenario, the dimensionality reduction technique preserves the essential structure of the data, meaning the number of data points (observations) remains the same, but the dimensionality k in Y is much lower than the original p in X. For reconstructing the low dimensional data back into the high dimensional form after processing, dimensionality reduction methods can use implicit, explicit or inverse mapping strategies (Sorzano et al., 2014).

Classified to either linear or non-linear data dimensionality reduction technique, the former uses simple linear functions to convert high dimensional spaces into lower dimensional spaces, while the latter uses more complex functions (Siblini et al., 2019). Principal Component Analysis (PCA), Kernel PCA, Factor Analysis, Singular Value Decomposition (SVD), Uniform Manifold Approximation and Projection (UMAP), t-Distributed Stochastic Neighbourhood Embedding (t-SNE) and Locally Linear Embedding (LLE) are the common data dimensionality reduction methods used with the clustering tasks within the unsupervised learning domain (Ayesha et al., 2020). Autoencoders, on the other hand, have in the recent past gained popularity in the area of data dimensionality reduction (Ayesha et al., 2020).

2.6.1 Principal Component Analysis

Principal Component Analysis (PCA) is a dimensionality reduction method used to transform a high-dimensional dataset into a lower-dimensional space while preserving the most important patterns and variances in the data (Sorzano et al., 2014). It accomplishes this by identifying the principal components, which are orthogonal (uncorrelated) linear combinations of the original features. By selecting a subset of these principal components, lower-dimensional representation of the data can be created while retaining most of the important information (Rehman et al., 2020). This lower-dimensional representation is useful for data visualization, compression, and feature extraction tasks (Hasan & Abdulazeez, 2021). Data dimensionality reduction process using the Principal Component Analysis involves five steps, namely: standardization, covariance matrix, Eigen value decomposition, selecting principal components and lastly, projection (Sorzano et al., 2014). In the first step, the data is standardized by subtracting the mean and dividing by the standard deviation for each feature (Rehman et al., 2020). This step ensures that all features have comparable scales. In the second step, covariance matrix of the standardized data is computed.

The covariance matrix captures the relationships and variances between different features. In the third step, eigenvalue decomposition is performed on the covariance matrix to find its eigenvectors and eigenvalues (Rehman et al., 2020). Eigenvectors represent the directions (principal components), and eigenvalues represent the amount of variance explained by each principal component.

In the fourth step, the eigenvectors based on their corresponding eigenvalues in descending order are selected (Rehman et al., 2020). The eigenvector with the highest eigenvalue represents the first principal component; the one with the second highest eigenvalue represents the second principal component, and so on (Rehman et al., 2020). The number of principal components can be chosen to retain based on the amount of variance that needs to be preserved (Rehman et al., 2020). Usually, a significant segment of 95% or more of the total variance is retained. In the last step, the original data onto the selected principal components to obtain the lower-dimensional representation is projected (Rehman et al., 2020). The projection is achieved by taking the dot product between the standardized data and the principal components (Rehman et al., 2020). The formula for the projection of the original data (X) onto the selected principal components (PC) can be represented as:

$$\text{Projected Data} = X * PC \quad (2.3)$$

Where projected data is the transformed lower-dimensional representation of the original data. X is the standardized original data matrix, where each row corresponds to a data point, and each column corresponds to a feature. PC is the matrix of selected principal components, where each column represents a principal component (eigenvector) (Rehman et al., 2020).

2.6.2 Kernel PCA

Kernel Principal Component Analysis (kPCA) is an extension of the standard Principal Component Analysis (PCA) that allows nonlinear dimensionality reduction (Rehman et al., 2020). It uses kernel functions to implicitly map the original data into a higher-dimensional space, where linear PCA can be applied to find principal components (Rehman et al., 2020). This dimensionality reduction method is particularly useful when the data cannot be effectively separated by linear transformations (Rehman et al., 2020). It has applications in image processing, bioinformatics, and pattern recognition. The choice of the kernel function plays a significant role in the performance of Kernel PCA, and selecting an appropriate kernel function is crucial for obtaining meaningful results (Rehman et al., 2020). Kernel PCA involves six steps i.e. Kernel function, Kernel matrix, centering the kernel matrix, eignen value decomposition, selecting principal components and lastly, projection. The first step involves choosing an appropriate kernel function (e.g., polynomial kernel, radial basis function (RBF) kernel, sigmoid kernel, etc.). The kernel function computes the dot product of the data points in a higher-dimensional feature space without explicitly transforming the data into that space (Rehman et al., 2020).

The second step involves the computation of the kernel matrix, which is an N x N symmetric matrix, where N is the number of data points. The element (i, j) of the kernel matrix is the kernel function evaluated for data points xi and xj (Rehman et al., 2020). The third step involves centering the Kernel Matrix to ensure that the data is represented in a zero-mean feature space (Rehman et al., 2020). This step is important for the mathematical properties of PCA (Rehman et al., 2020). The fourth step involves performing eigenvalue decomposition on the centered kernel matrix to find its eigenvectors and eigenvalues (Rehman et al., 2020).

The fifth step involves selecting Principal Components and sorting the eigenvectors based on their corresponding eigenvalues in descending order. The eigenvector with the highest eigenvalue represents the first principal component; the one with the second highest eigenvalue represents the second principal component, and so on. The last step involves the projection of the original data onto the selected principal components to obtain the lower-dimensional representation. The projection is achieved by taking the dot product between the centered kernel matrix and the selected principal components. The formula for the projection of the original data onto the selected principal components (PC) can be represented as follows:

Projected Data = Centered_Kernel_Matrix * PC (2.4)

Where Projected Data is the transformed lower-dimensional representation of the original data. Centered_Kernel_Matrix is the kernel matrix after being centered to have zero mean. PC is the matrix of selected principal components, where each column represents a principal component (eigenvector) (Rehman et al., 2020).

2.6.3 Factorial Analysis

Factor Analysis (FA) is a statistical method used for data dimensionality reduction and to identify underlying factors (latent variables) that explain the observed correlations among variables (Ayesha et al., 2020). It is particularly useful when dealing with a large number of correlated variables and aims to capture the essential information while reducing the dimensionality. Factor Analysis can be applied to a variety of fields, such as Psychology, Social Sciences, Finance, and Marketing (Ghodsi, 2006). It helps identify the underlying structure of observed variables, reducing the number of dimensions and simplifying data interpretation (Ghodsi, 2006).

However, it's essential to be cautious and interpret the factors carefully, as they are not always directly interpretable and might require domain knowledge for meaningful understanding. Factor analysis involves four steps, namely: factor loadings, Eigen decomposition, determining the number of factors and lastly, the computation of the factor scores (Ayesha et al., 2020). In step one, the factor loadings matrix are estimated. The factor loadings represent the strength of the relationship between each observed variable and the underlying latent factors. The factor loadings matrix provides insights into which variables are most influenced by each factor. In the second step, eigenvalue decomposition is performed depending on the type of Factor Analysis (e.g., Principal Component Analysis (PCA) based FA or Maximum Likelihood Estimation based FA).

In the third step, the numbers of factors are determined in order to decide on the number of underlying factors to retain. This decision can be based on eigenvalues, scree plots, or other statistical criteria, like the Kaiser-Guttman criterion or the explained variance. Lastly, once the factor loadings are estimated, the latent factors' scores for each observation can be calculated. These scores represent the transformed lower-dimensional representation of the data. The formula for the Factor Analysis is represented as follows:

$$X = \mu + LF + \varepsilon \tag{2.5}$$

Where X is the data matrix of the observed variables. μ is the vector of means for each observed variable. LF is the matrix of factor loadings that represents the relationships between the observed variables and the latent factors. ε is the matrix of unique (error) terms.

2.6.4 Singular Value Decomposition

Singular Value Decomposition (SVD) is a linear matrix factorization technique used for data dimensionality reduction, feature extraction, and data compression (Ayesha et al., 2020). It decomposes a matrix into three matrices, where the middle matrix represents the singular values and captures the most important information about the original data. Singular Value Decomposition is widely used in various fields, including image processing, recommendation systems, natural language processing, and data compression (Robinson et al., 2019).

It helps to extract meaningful information from high-dimensional datasets and allows for efficient data representation and analysis (Ayesha et al., 2020). Dimensionality reduction method using the Singular Value Decomposition involves two steps i.e. creation of a data matrix as well as the SVD decomposition. In the first step, an m x n data matrix X is considered, where each row represents an observation (data point), and each column represents a feature (Ayesha et al., 2020). The SVD decomposition process involves performing Singular Value Decomposition on the data matrix X. The formula for the Single Value Decomposition is represented as follows:

$$X = U * \Sigma * V^T \quad (2.6)$$

Where U is an *m x m* matrix, containing the left singular vectors (columns) representing the relationships between the observations and the latent features. These vectors are orthogonal to each other, and they form an orthonormal basis for the row space of X (Ayesha et al., 2020). Σ is an *m x n* diagonal matrix, containing the singular values (non-negative) along its diagonal. The singular values are sorted in descending order.

They represent the importance of each latent feature (Ayesha et al., 2020). V^T is an *n x n* matrix, containing the right singular vectors (rows) representing the relationships between the features and the latent features. These vectors are orthogonal to each other and form an orthonormal basis for the column space of X (Ayesha et al., 2020).

2.6.5 Uniform Manifold Approximation and Projection

UMAP (Uniform Manifold Approximation and Projection) is a nonlinear dimensionality reduction technique that aims to preserve local and global structure in high-dimensional dataset (Ghojogh, 2021). It is particularly useful for visualizing complex datasets and discovering underlying patterns. Its ability to preserve both local and global structures makes it effective for data visualization and clustering (Ghojogh, 2021). It has become a popular technique for exploratory data analysis and has found applications in various fields, such as bioinformatics, image analysis, and natural language processing (Ghojogh, 2021). Data dimensionality reduction using the Uniform Manifold Approximation and Projection involves three steps, namely: constructing the nearest neighbour graph, converting the nearest neighbour graph into fuzzy-simplified set approximation and lastly, low dimensional embedding.

In the first step, the nearest neighbors for each data point in the high-dimensional space, based on some distance metric (e.g., Euclidean distance) are identified (Ghojogh, 2021). The result is a weighted graph where each data point is connected to its nearest neighbors. In the second step, the nearest neighbor graph is converted into a fuzzy-simplicial set representation. This step involves transforming the distances between data points into probabilities representing the likelihood of a data point being a neighbor to another point (Ghojogh, 2021).

The probabilities are calculated using a fuzzy topological set, which allows a data point to be a neighbor to multiple points with different probabilities. In the last third step, the optimization of the low-dimensional embedding is performed using a stochastic gradient descent. The optimization process minimizes the cross-entropy between the fuzzy-simplified set and the low-dimensional embedding.

This encourages data points that are close in the high-dimensional space to be close in the low-dimensional space as well (Ghojogh, 2021). The reduced dimensional representation of the data (in the low-dimensional space) is the final output of Uniform Manifold Approximation and Projection (Ghojogh, 2021). This representation retains the underlying structure of the original data while reducing its dimensionality. The low-dimensional embedding can then be visualized, analyzed, or used as input for other machine learning tasks (Ghojogh, 2021).

2.6.6 t-Distributed Stochastic Neighborhood Embedding

T-SNE (t-Distributed Stochastic Neighbor Embedding) is another popular nonlinear dimensionality reduction method used for visualizing high-dimensional into a low-dimensional space (Ghodsi, 2006). It is particularly effective at preserving local and global structures in the data and is commonly used for visualizing complex datasets (Ghodsi, 2006). T-SNE involves three steps, i.e: computing pairwise similarities, constructing conditional probability distributions and lastly, the optimization of the low-dimensional embedding (Ghodsi, 2006). In the first step, the pairwise similarity for each data point in the high-dimensional space is computed to all the other data points. The similarity is typically based on a Gaussian kernel function, which measures the similarity between two data points based on their distances.

In the second step, two conditional probability distributions are constructed using the pairwise similarities. One distribution represents the similarity of each data point to all other data points in the high-dimensional space. The other distribution represents the similarity of each data point to all other points in the low-dimensional space.

In the last step, the low-dimensional embedding is optimized in such a way that the data points that are similar in the high-dimensional spaces have similar probabilities of being neighbors in the low-dimensional spaces. The optimization is achieved using the gradient descent, where the algorithm iteratively updates the positions of data points in the low-dimensional space to minimize the divergence between the two distributions (Ghodsi, 2006). The reduced dimensional representation of the data (in the low-dimensional space) is the final output of t-SNE. This representation retains the local and the global structures of the original data while reducing its dimensionality. The low-dimensional embedding can then be visualized to gain insights into the data's underlying patterns and relationships.t-SNE is computationally intensive and can be slow for large datasets (Ghodsi, 2006). Therefore, it is commonly used for visualizing relatively small to moderate-sized datasets (Ghodsi, 2006).

2.6.7 Locally Linear Embedding

Locally Linear Embedding (LLE) is a nonlinear dimensionality reduction technique that aims to preserve local linear relationships in high-dimensional data (Ghodsi, 2006). It is particularly effective at capturing the intrinsic geometry of the data manifold. LLE works by modeling each data point as a linear combination of its neighbors and then reconstructing the lower-dimensional representation using these linear combinations (Ghodsi, 2006). The process of data dimensionality reduction using the Locally Linear Embedding involves neighborhood selection and local reconstruction weights (Ghodsi, 2006).

In the first step, the *k*-nearest neighbors based on some distance metric (e.g., Euclidean distance) are identified for each data point in the high-dimensional space (Ghodsi, 2006). In the second step, the linear combination of its neighbors that best reconstructs the data point itself for each data point is found out. This is achieved by solving a weighted least squares problem shown using the equation below:

$$\min \sum (x\_i - \sum W\_ij * x\_j)^2 \quad (2.7)$$

Where x_i is the data point to be reconstructed. x_j are the neighboring data points. W_ij are the weights that represent the linear combination coefficients.W_ij are constrained to ensure that the reconstruction of each data point is locally linear. Specifically, these constraints ensure that the data point can be accurately reconstructed as a linear combination of its neighboring data points, preserving the local structure of the dataset. The weight matrix W is organized such that each row corresponds to a data point, with each column containing the weights that represent the linear combination coefficients for reconstructing that data point from its neighbors (Ghodsi, 2006). The low-dimensional embedding is computed by solving an optimization problem that preserves the local relationships of the data points. This optimization aims to minimize the difference between the pairwise distances in the high-dimensional space and the pairwise distances in the low-dimensional space for the neighboring data points using the formula below:

$$\min \sum (||z\_i - z\_j|| - ||x\_i - x\_j||)^2 \quad (2.8)$$

where z_i and z_j are the low-dimensional embeddings of data points x_i and x_j, respectively. The optimization problem is subject to certain constraints to ensure that the low-dimensional embedding preserves the local relationships in the data (Ghodsi, 2006).

By solving the optimization problem, LLE finds a low-dimensional embedding that captures the local linear relationships between data points, effectively reducing the dimensionality while preserving the local geometry of the data manifold (Ghodsi, 2006). LLE has been widely used for various tasks, such as data visualization, manifold learning, and image processing. However, it can suffer from computational and memory limitations for large datasets due to the need to solve the weight matrix and optimization problems (Ghodsi, 2006).

2.6.8 Autoencoders

Autoencoders are a type of neural network used for unsupervised learning and data dimensionality reduction (Li et al., 2018). They are part of the broader family of neural networks called auto associative neural networks, and their primary goal is to reconstruct the input data at the output layer, usually through a bottleneck layer with reduced dimensionality (Li et al., 2018). Autoencoders can be used for various tasks, including feature extraction, denoising, and anomaly detection (Li et al., 2018). Autoencoders consists of mainly the encoder and the decoder. The encoder part of the autoencoder takes the input data and maps it to a lower-dimensional representation, also known as the encoding or latent space. It is represented by the following function:

$$z = f\_encoder(x) \quad (2.9)$$

Where x is the input data (e.g., a vector of features). z is the encoding / latent representation of x. f_encoder represents the mapping performed by the encoder neural network. On the other hand, the decoder part of the autoencoder takes the encoding/latent representation and reconstructs the input data. It is represented by the following function:

x_hat = f_decoder(z) (2.10)

Where x_hat is the reconstructed data, which should ideally match the input data *x*. The *z* is the encoding / latent representation of x. f_decoder represents the mapping performed by the decoder neural network. The autoencoder's objective is to minimize the reconstruction error between the original input data x and the reconstructed data x_hat. This is usually measured using a loss function such as Mean Squared Error (MSE), shown below:

Loss = MSE(x, x_hat) (2.11)

The loss function quantifies how well the autoencoder can reconstruct the input data. During training, the autoencoder adjusts its parameters (weights and biases) using optimization techniques like gradient descent to minimize the reconstruction error (Li et al., 2018). By using an autoencoder, the input data x is effectively compressed and represented in a lower-dimensional space z. This compressed representation can then be used for various tasks, such as visualization, feature extraction, or anomaly detection (Li et al., 2018). Autoencoders come in various architectures, ranging from shallow autoencoders with a single hidden layer to deep autoencoders with multiple hidden layers. These different architectures enable the autoencoders to learn increasingly complex representations of the data. Additionally, variants such as denoising autoencoders and variational autoencoders introduce additional constraints and regularization techniques to further refine the encoding process. Denoising autoencoders are designed to improve robustness by reconstructing data from noisy inputs, while VAEs incorporate probabilistic models to capture complex data distributions and enhance the generative capabilities of the autoencoder (Li et al., 2018).

These modifications not only improve the quality of the learned representations but also expand the range of applications for autoencoders in tasks such as anomaly detection, data generation, and more. Figure 2.3 shows the diagrammatic representation of an autoencoder.

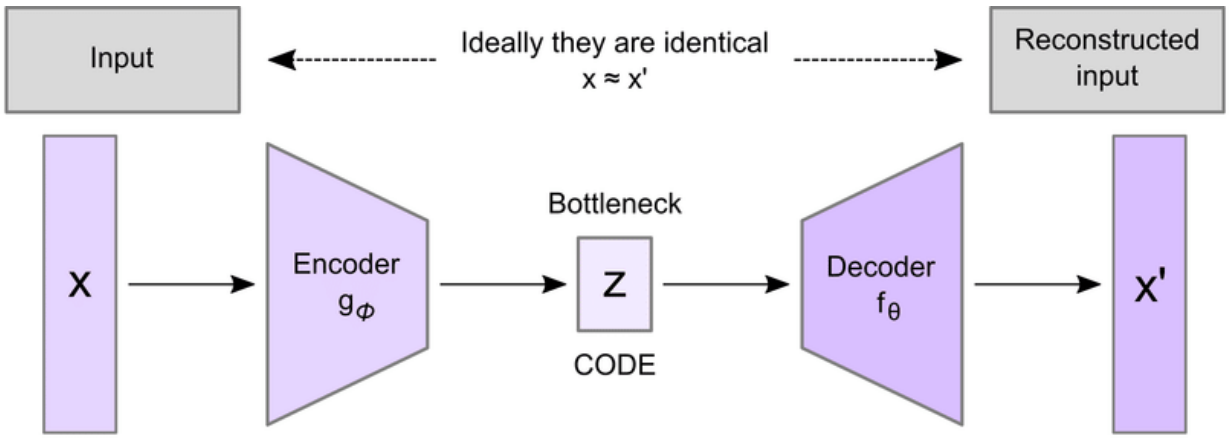


Figure 2.3: Diagrammatic Representation of an Autoencoder

Besides performing dimensionality reduction, the autoencoder's latent space can be leveraged to perform both the feature learning and the transfer learning. Autoencoders can learn meaningful features in an embedded latent space while at the same time train for specific tasks, leading to more efficient and effective representations for downstream tasks like classification. The autoencoder's embedded latent space can also facilitate transferring knowledge from one task or dataset to another. By learning a shared representation in the latent space, models can leverage knowledge learned from one domain to improve performance in another.

2.6.9 Kernel Functions and Variations for Dimensionality Reduction Methods

Kernel methods are powerful techniques for data dimensionality reduction, especially when dealing with non-linear relationships in high-dimensional datasets (De et al., 2015; Kung, 2014). They are highly effective at capturing complex structures in dataset and finding meaningful low-dimensional representations (De et al., 2015; Kung, 2014).

These methods project the data into a higher-dimensional feature space where it may become more separable or linearly approximated (De et al., 2015; Kung, 2014). Radial Basis Function, linear kernel, polynomial kernel, sigmoid kernel, laplacian kernel, Bessel kernel, ANOVA kernel and sigmoid kernel are the common functions used in kernalizing data dimensionality reduction methods (De et al., 2015; Kung, 2014).

2.6.9.1 Linear Kernel

The Linear Kernel is one of the simplest and most commonly used kernel functions. It computes the inner product of the original data points without any transformation. The following equation represents a linear kernel function:

$$K(x, y) = x^T y \quad (2.12)$$

Where K(x,y) represents the result of applying the linear kernel to the input vectors *x* and *y*. The *x* the first input vector in the feature space while the *y* is the second input vector in the feature space. The x^T denotes the transpose of the vector *x*, an operation that converts a column vector into a row vector.Linear Kernels are mostly used in methods like Principal Component Analysis (De et al., 2015; Kung, 2014).

2.6.9.2 Polynomial Kernel

The Polynomial Kernel allows for non-linear transformations by introducing polynomial terms. The following equation represents a polynomial kernel function:

$$K(x, y) = (x^T y + c)^d \quad (2.13)$$

K(x, y) represents the result of applying the polynomial kernel to the input vectors. The x and y are the input vectors in the feature space. Polynomial Kernel can capture non-linear relationships in a dataset (De et al., 2015; Kung, 2014). $x^T y$ denotes the dot product of the vectors. C is a constant while d is a polynomial that controls the complexity of the polynomial transformation applied to the data.

2.6.9.3 Radial Basis Function (RBF) Kernel

The RBF Kernel, also known as the Gaussian kernel, is a widely used kernel method. It is especially useful for capturing non-linear patterns in a dataset. The following equation represents the Radial Basis Function kernel function.

$$K(x, y) = \exp(-\gamma \|x - y\|^2) \quad (2.14)$$

Here, the K(x, y) represents the result of applying the RBF kernel to the input vectors x and y. The x and y are the input vectors into the feature space. The $\|x - y\|^2$ denotes the squared Euclidean distance between the vectors. The 'γ' is a hyperparameter that controls the width of the kernel, determining the influence of each data point on others (De et al., 2015).

2.6.9.4 Sigmoid Kernel

The Sigmoid Kernel is another non-linear kernel that is often used in support vector machines (SVMs) and other dimensionality reduction techniques. The following equation represents a sigmoid kernel function

$$K(x, y) = \tanh(\alpha x^T y + \beta) \quad (2.15)$$

Where K(*x*,*y*) represents the result of applying the Sigmoid Kernel to the input vectors *x* and *y*. The x and y are the inut vectors into the feature space. x^T y denotes the dot product of the vectors *x* and *y*. The α is the scaling factor that controls the influence of the dot product term in the sigmoid function while β is a constant that shifts the sigmoid function and influences its output range. The tan*h* is a hyperbolic tangent function maps the input to a value between -1 and 1 (De et al., 2015; Kung, 2014).

2.6.9.5 Laplacian Kernel

The Laplacian Kernel is similar to the RBF kernel but with a different shape. The following equation represents a Laplacian kernel function

$$K(x, y) = \exp(-\gamma \|x - y\|) \quad (2.16)$$

Where K(*x*, *y*) represents the result of applying the Laplacian Kernel to the input vectors *x* and *y*. The *x* and *y* are the input vectors into the feature space. The ||x - y||) is the the Euclidean distance between the vectors *x* and *y*. The γ is the hyperparameter that controls the width or spread of the kernel function (De et al., 2015; Kung, 2014).

2.6.9.6 Bessel Kernel

The Bessel kernel is a modified version of the RBF kernel and is used in some dimensionality reduction techniques. The following equation represents a Bessel kernel function.

$$K(x, y) = J_{-0}(\gamma \|x - y\|) \quad (2.17)$$

Where K(x, y) represents the result of applying the Bessel Kernel to the input vectors *x* and *y*. The x and y are the input vectors into the feature space.

The ||x - y||) denotes the Euclidean distance between the vectors *x* and *y*. The γ is the hyperparameter that scales the distance ||x - y|| before applying the Bessel function. 'J_0' represents the Bessel function of the first kind of order zero (De et al., 2015; Kung, 2014).

2.6.9.7 ANOVA Kernel

The ANOVA Kernel is used for feature selection and dimensionality reduction tasks. It is designed to capture interactions between different features and considers all possible feature interactions.

$$K(x, y) = \exp(-\gamma \sum(x\_i - y\_i)^2) \quad (2.18)$$

Where K(*x*,*y*) represents the result of applying the ANOVA Kernel to the input vectors *x* and *y*. The *x* and *y* are the input vectors into the feature space. The *x*-*i* and *y*-*i* represents the *i*-th component of the vectors *x* and *y*, respectively. The γ is the hyperparameter that scales the sum of squared differences and controls the width of the kernel function. Kernel methods have been successfully applied in data dimensionality reduction methods during machine learning and pattern recognition tasks (De et al., 2015; Kung, 2014). Examples of machine learning models that have successfully used kernel methods include the kernel factor analysis in the "new probabilistic kernel factor analysis for multisensory data fusion" by Wang et al. *(*2016) and the "incremental kernel SVD for face recognition with image sets" by Chin et al. (2006). The selection of the appropriate kernel function and tuning of its parameters depend on the data and the specific dimensionality reduction task at hand (De et al., 2015; Kung, 2014). Kernel methods, however, can be computationally intensive, particularly when the data dimensionality is high (De et al., 2015; Kung, 2014).

2.6.9.8 Special Variations for Variety of Dimensionality Reduction Methods

Some data dimensionality reduction methods, in their standard forms, may not be the most suitable for handling specific datasets (De et al., 2015). For example, PCA is one of the most widely applied data dimensionality reduction method but it may not be the best choice for image dataset in its standard form. It may not be effective in preserving important features and structures in images. In order to curb such limitations and to make such dimensionality reduction methods more effective in handling specific datasets, a number of variations and extensions, based on their original standard forms, have been proposed. The PCA with ZCA whitening is an extension of the popular PCA that is effective with image data. The original PCA identifies orthogonal directions in the data with the highest variance, but it doesn't take into account the correlations between pixels in the image (Yong-lian, 2020). ZCA whitening (Zero Component Analysis) technique addresses this issue by reducing the correlations between the features (pixels) in the dataset (Yong-lian, 2020). The "Multi-feature data mining for CT image recognition" is an example of an algorithm that has successfully used this specific ZCA based PCA variant to reduce the dimensionality of image dataset.

The Principal Component Analysis with Spatiotemporal Cubes is an extension of PCA that is effective with video data. In this variant, each frame of the video data is treated as a vectorized representation. After this, the multiple consecutive frames are stacked together to form 3D spatio-temporal cubes. PCA is then applied to these cubes to capture both the spatial and temporal variations (Bueso et al., 2020). The "Non-linear PCA for Spatio-Temporal Analysis of Earth Observation Data" is an example of an algorithm that has successfully used this specific PCA variant to reduce the dimensionality of video dataset. PCA Filter Banks is an extension of PCA that is effective with audio data. In this variant, PCA is applied to filter bank coefficients; bank coefficients are common feature representation for audio data.

Filter banks capture spectral information at different frequency bands, making it effective for feature learning from an audio dataset. The “Speaker recognition using PCA-based feature transformation” is an example of an algorithm that has successfully used this specific PCA variant to reduce the dimensionality of audio datasets. Polynomial Function based Kernel PCA is a variant of the Kernel PCA that is effective for text data. This variant allows you to capture relationships and patterns in the text data, making it suitable for various text analysis tasks (Mohammed, 2022). The “Improved Regularized Multi-class Logistic Regression for Gene Classification with Optimal Kernel PCA and HC Algorithm” is an example of an algorithm that has successfully used this specific Polynomial Function based Kernel PCA variant to reduce the dimensionality of text datasets. Radial Basis Function based Kernel PCA is the variant of the Kernel PCA that is effective for image, audio and video datasets. This variant allows one to capture complex relationships and patterns in both the image, audio and the video datasets, making it suitable for various images, audio and video analysis tasks (Sawssen et al., 2020).

The “Mammographic images classification technique via the Gaussian Radial Basis Kernel ELM and kPCA” is an example of an algorithm that has successfully used this specific Radial Basis Function based Kernel PCA variant to reduce the dimensionality of image datasets. The “Metric learning-based multimodal audio-visual emotion recognition” is an algorithm that has used Radial Basis Function based Kernel PCA variant to reduce the dimensionality of audio datasets. The “ML Project 2 Motion-based Similarity Search in Videos of Confucian Rituals” is an example of an algorithm that has successfully used this specific Radial Basis Function based Kernel PCA variant to reduce the dimensionality of video datasets.

Exploratory Factor Analysis (EFA) is the variant of the Factor Analysis that is effective for text data. It explores the underlying structure or latent factors in a numeric dataset. It is suitable for uncovering relationships among observed variables and identifying factors that explain the observed correlations. EFA assumes that the observed variables are linear combinations of the latent factors plus error terms (Finch, 2020). "Using Fit Statistic Differences to Determine the Optimal Number of Factors to Retain in an Exploratory Factor Analysis" is an example of an algorithm that has successfully used this specific Factor Analysis variant to reduce the dimensionality of text datasets.

Factorization machine is the variant of the Factor Analysis that is effective for image, audio and video datasets. They can be adapted for these datasets by encoding interactions between image, audio and video features, making them suitable for various classification and recommendation tasks involving any type of these datasets (Huang et al., 2019). For example, the "FiBiNET (Feature Importance and Bilinear feature Interaction NETwork)" is an example of a study that has successfully used this specific Factor Analysis variant to reduce the dimensionality of image datasets.

Randomized SVD is a variant of the Singular Value Decomposition that is effective for text datasets. This is an approximation technique that uses random sampling to compute an approximate SVD of text numeric data. It is useful for large-scale text datasets when exact SVD computation is computationally expensive and provides a good approximation to the full SVD (Zhang and Tang, 2022). The "Perturbation Analysis of Randomized SVD and its Applications to High-dimensional Statistics" is an example of a study that has successfully used this specific SVD variant to reduce the dimensionality of text datasets.

Economy SVD, also known as thin SVD, is a variant of the Singular Value Decomposition that is effective for image, audio and video datasets. The Economy SVD computes the Singular Value Decomposition in a way that avoids explicitly computing the zero singular values and corresponding vectors. It results in smaller matrices that are more efficient when working with large image datasets. For example, the "Facial Image Characterization and Reconstruction Using Singular Value Decomposition" is an example of an algorithm that has successfully used this specific SVD variant to reduce the dimensionality of image datasets (Mallick & Mukhopadhyay, 2022). On the other hand, "Video retrieval framework based on color co-occurrence feature of adaptive low rank extracted keyframes and graph pattern matching" is an example of an algorithm that has successfully used this specific SVD variant to reduce the dimensionality of video datasets.

The "UMAP-learn for images" is a variant of the Uniform Manifold Approximation and Projection method that is effective for image datasets. It provides a general-purpose implementation of UMAP. The UMAP-learn is applied to image data by first converting images into appropriate numeric representations like the commonly used approach of flattening the pixel values (Tetef et al., 2023). The "Manifold Projection Image Segmentation for Nano-XANES Imaging" is an example of an algorithm that has successfully used this specific UMAP variant to reduce the dimensionality of image datasets.

Spectrogram UMAP is a variant of the Uniform Manifold Approximation and Projection method that is effective for audio datasets. It converts the audio signals into spectrograms, which are representations of audio in the time-frequency domain. After this, the standard UMAP is applied to reduce the dimensionality of the spectrogram data. This helps in visualizing and exploring audio patterns and similarities (Thomas et al., 2022).

The “practical guide for generating unsupervised, spectrogram-based latent space representations of animal vocalizations” is an example of a study that has successfully applied this specific UMAP variant to reduce the dimensionality of audio datasets.

Spatio-Temporal UMAP is a variant of the Uniform Manifold Approximation and Projection method that is effective for video datasets. It creates spatio-temporal representations of video data by considering both spatial and temporal information. Features from video frames can be extracted and temporal dependencies incorporated in order to create feature vectors for segments of video clips. After this, a standard UMAP is applied to these spatio-temporal feature vectors for dimensionality reduction and visualization (Mayer et al., 2023). The “Characterization of Interactive Visual Data Stories with a Spatio-Temporal Context” is an example of a study that has successfully applied this specific UMAP variant to reduce the dimensionality of video datasets.

Multicore t-SNE is a variant of the t-Distributed Stochastic Neighbor Embedding method that is effective for image, audio and video datasets. Multicore t-SNE is an optimized version of t-SNE that leverages parallel processing to speed up the computation. This can be particularly useful when working with large datasets, as it can significantly reduce the computation time (Hozumi et al., 2021). For example, the “UMAP-assisted K-means clustering of large-scale SARS-CoV-2 mutation datasets” is an example of an algorithm that has successfully applied this specific t-SNE variant to reduce the dimensionality of image datasets.

The Superpixel-Based LLE is a variant of the Locally Linear Embedding method that is effective for image datasets. Superpixels are compact and perceptually meaningful image regions.

LLE is applied to the superpixels instead of individual pixels in order to reduce the dimensionality while preserving local relationships within these regions. This helps in object segmentation and image categorization (Liu et al., 2022). The “Superpixel-guided locality quaternion representation for color face hallucination” is an example of an algorithm that has successfully applied this specific LLE variant to reduce the dimensionality of image datasets.

Spectrogram-Based LLE is a variant of the Locally Linear Embedding method that is effective for audio datasets. The audio signals are converted into spectrograms, which represent the audio in the time-frequency domain. After this, the standard LLE is applied to the spectrogram data to capture local linear relationships between time-frequency components.

This helps in visualizing and exploring patterns in audio signals (Altaf et al., 2019). The “Automatic and efficient fault detection in rotating machinery using sound signals” is an example of an algorithm that has successfully applied this specific LLE variant to reduce the dimensionality of audio datasets.

Spatio-Temporal LLE is a variant of the Locally Linear Embedding method that is effective for video datasets. The spatio-temporal representations of video data are created by considering both the spatial and temporal information. Features from the video frames are extracted and temporal dependencies incorporated in order to create feature vectors for segments of video clips. Finally, the spatio-temporal feature vectors are applied to the LLE for dimensionality reduction process while preserving local relationships (Erekat, 2021). The “Spatio-Temporal Assessment of Pain Intensity through Facial Transformation-Based Representation Learning” is an example of an algorithm that has successfully applied this specific LLE variant to reduce the dimensionality of video datasets.

Autoencoders have been successfully used for unsupervised learning and data dimensionality reduction method, creating cluster friendly spaces when using *k*-means (Fard et al., 2020). The deep *k*-means model is an example of the algorithm that has successfully achieved cluster friendly spaces in *k*-means, making it easier to identify the correct *k*-hyperparameter value. The traditional data dimensionality reduction methods have been successfully applied in improving the performance of an autoencoder using transfer learning, by serving as an effective dimensionality reduction technique for feature extraction on the source task. After this, the extracted features are then transferred and used as inputs to the autoencoder in the target task (Yan & Zhang, 2016).

2.7 Evaluation metrics for *K*-hyperparameter Tuning Techniques in High Dimensional Space Clustering

In unsupervised clustering algorithms, for example *k*-means, the ground truth about the *k*-hyperparameter value, the number of clusters in a specific dataset, relies on the prior knowledge of the problem (Rendon et al., 2011). In most cases, there is no prior knowledge or intuition about the clustering dataset, at hand, and at times, the domain knowledge is required (Rendon et al., 2011). For this reason, it is important to use the metrics that give some intuition about the best or the optimal value of *k* on any clustering high dimensional dataset (Rendon et al., 2011). Such a standard cluster validation process and set of internal validation metrics is highly critical to assessing the quality of the *k* clusters generated as the output from the high-dimensional *k*-means algorithms (Rendon et al., 2011). The optimal *k*-value, in *k*-means, dictates the best clustering results (Rendon et al., 2011). At this optimal, the variance within a cluster is normally low, while the separation between clusters is normally high (Rendon et al., 2011). Some of the most commonly applied metrics are discussed in the subsequent sections.

2.7.1 Internal Validation Indexes

The choice of the internal validation metrics, as opposed to the external and relative validation metrics, is based on the fact that the internal metrics are purely based on the information intrinsic to the data alone with no clue on prior information about the dataset (Rendon et al., 2011). The Internal indexes are known to be better while applied in the determination of the quality of the clustering results because they are purely based on the information intrinsic to the data alone (Rendon et al., 2011). The most commonly used internal validity metrics, in the clustering literature; include Dunn index, calinski harabsz index, Davies Bouldin index and the silhouette index. However, bayesian information criterion, point bi-serial and sum-of-squares have also been used to some extent. Comparing the scores of the different pairs of internal validation metrics, for one dataset, as well as comparing their consistency using Kendall's index, each at a time, would be computationally expensive (Liu et al., 2013). For this reason, an ensemble validation metric is proposed, whose components exercise equal sensitivity to the varied conditions present in the high dimensional datasets. This type of ensemble could either be bagging (bootstrap aggregating) or boosting (Liu et al., 2013). The most commonly used internal validity metrics, in the clustering literature, include:

2.7.1.1 Dunn Index (DI)

Dunn Index, an internal validity metric, indicates a high degree of compactness of the objects belonging to the same cluster and a high degree of separation between objects in different clusters (Deborah et al., 2010; Saito et al., 2012). Dunn index is defined mathematically as follows:

$$\mathrm{DI} = \frac{\underset{1\le i\le j\le m}{Min\delta(C_i,C_j)}}{\underset{1\le k\le m}{\max \Delta_k}} \tag{2.19}$$

Where m is the total numner of clusters, distance between clusters i and j are denoted by δ (C_i, C_j) and the Δk is the size of cluster (Saito et al., 2012). Higher values of the Dunn index indicate the minimum intra-cluster distances as well as the maximum inter cluster distance (Saito et al., 2012). A number of *k*-means algorithms have successfully used the Dunn index to validate the clustering results that they generate (Deborah et al., 2010; Saito et al., 2012).

2.7.1.2 Calinski-Harabasz Index (CH)

Calinski-Harabasz Index (CH), an internal validity metric, is referred to as the ratio between the "sums of between-clusters dispersion" and "inter-cluster dispersion" for all clusters (Deborah et al., 2010; Rendon et al., 2011). Calinski-Harabasz index, as follows, mathematically:

$$\text{CH(K)}=\frac{B(K)(N-K)}{W(K)(K-1)} \tag{2.20}$$

$$\text{B(K)} = (\Sigma_{k=1}^{k}\ a_k \parallel \bar{x}_k - \bar{x} \parallel^2) \tag{2.21}$$

$$W(K) = (\Sigma_{k=1}^{k}\Sigma_{c(j)=k}\|x_j - \overline{x_k} \|^2) \tag{2.22}$$

Where *k* is the corresponding number of clusters, B(K) is the inter-cluster divergence, also called the inter-cluster covariance, W(K) is the intra-cluster divergence, also called the intra-cluster covariance, and N is the number of samples (Deborah et al., 2010; Rendon et al., 2011). The larger the B(K) is, the higher the degree of dispersion between clusters is (Deborah et al., 2010; Rendon et al., 2011). The smaller the W(K) is, the closer the relationship in the cluster (Rendon et al., 2011). Higher CH values are better because they are an indication of a good quality clustering performance and results (Deborah et al., 2010; Rendon et al., 2011).

2.7.1.3 Davies Bouldin Index (DB)

Davies-Bouldin Index (DB), an internal validity metric, is used to identify cluster overlap by measuring the ratio of the sum of the "within-cluster scatters" to the "between-cluster separations" (Rendon et al., 2011). Davies-Bouldin index is defined as follows:

$$DB=\frac{1}{k}\sum_{i=1}^{k}\max_{j\neq i}\left(\frac{\bar{c}_i+\bar{c}_j}{\|w_i-W_j\|_2}\right) \quad (2.23)$$

Where *k* is the total number of clusters, *wi* and *wj* are the cluster centroids, C*i* is the measure of scatter of all points in cluster *i* to the centroid *wi*. C*j* is the measure of scatter of all points in cluster *j* to the centroid *wj*. The ||wi-wj|| is the distance between the two centroids. A DB index that is close to zero (0) is an indication that the clusters are compact and far apart from each other (Deborah et al., 2010; Rendon et al., 2011). The implementation of *K*-Medoids algorithm with Davies-Bouldin-Index evaluation for Clustering Postoperative Life Expectancy in Patients with Lung Cancer is an example of a *k*-means algorithm that has applied Davies Bouldin index in its cluster analysis (Deborah et al., 2010; Rendon et al., 2011).

2.7.1.4 Silhouette Index (SI)

Silhouette Index, an internal validity metric, is referred to as the optimal clustering number derived from the difference between the average distance within the cluster and the minimum distance between the clusters (Deborah et al., 2010; Rendon et al., 2011). Silhouette index is defined mathematically as follows:

$$\bar{s}=\frac{1}{n}\sum_{i=1}^{n}\left(\frac{b(i)-a(i)}{\max\{a(i),b(i)\}}\right) \quad (2.24)$$

Where a(*i*) represents the average distance of sample *i* to other samples in the cluster, b(*i*) represents the minimum distance of the sample from the sample *i* to the other clusters (Mamat et al., 2018). The Silhouette index for determining optimal *k*-means clustering on images in different color models is an example of an algorithm that has applied silhouette index in its cluster analysis (Deborah et al., 2010; Rendon et al., 2011). However, it is important to note that Calinski harabsz index, Silhouette index, Dunn index and Davies Boulldin index poses some limitations at the individual level. These limitations mainly include: Sensitivity to cluster density, dependency on cluster size, lack of ground truth labels, interpretation challenges and sensitivity to data scaling and data dimensionality (Deborah et al., 2010; Rendon et al., 2011). For example, the calinski harabsz indexs tend to favor clusters with similar densities (Deborah et al., 2010; Rendon et al., 2011). If the clusters have significantly different densities, the index may not accurately capture the clustering quality (Deborah et al., 2010; Rendon et al., 2011). It may give higher scores to clusters that are denser, even if they are not necessarily well-separated. The index is sensitive to the number of clusters and the size of the dataset. It tends to favor solutions with a larger number of clusters, which may lead to overfitting or over-segmentation of the data (Deborah et al., 2010; Rendon et al., 2011). This can result in inflated index values, making it challenging to determine the optimal number of clusters (Deborah et al., 2010; Rendon et al., 2011). These indexes do not rely on ground truth labels (Deborah et al., 2010; Rendon et al., 2011).

They assess the clustering quality based on the internal structure of the data, without considering the true underlying classes (Rendon et al., 2011). Therefore, their effectiveness may be limited when compared to metrics that utilize ground truth information. Similar to the Dunn index, interpreting the absolute value of the Calinski-Harabasz index can be difficult (Deborah et al., 2010; Rendon et al., 2011).

It lacks a clear threshold or guideline to determine what constitutes a good or bad clustering result (Deborah et al., 2010; Rendon et al., 2011). It is often used comparatively, comparing different clustering solutions or tuning the number of clusters to find an optimal solution (Rendon et al., 2011). The Calinski-Harabasz index can be sensitive to the scaling of the data and the number of dimensions (Rendon et al., 2011). Inconsistent scaling on high-dimensional data can impact the distances used in the calculation and lead to biased results (Deborah et al., 2010; Rendon et al., 2011). Proper data preprocessing and dimensionality reduction techniques may be necessary to address these issues. It is therefore important to consider these limitations when using the each of these internal validation indexes and complement them with each other inorder to gain a more comprehensive understanding of the clustering quality. At the same time, Dunn index, Silhoutte index, Calinski Harabsz index as well as the Davies Bouldin index focuses on different aspect / perspective about the cluster quality at the individual level. For this reason, it is important to use a combination of these indexes in order to have a more comprehensive understanding of the the quality of the clustering results through trade offs and comparative analysis.

2.7.1.5 Bayesian Information Criterion (BIC)

Bayesian information criterion (BIC), an internal validity metric, is referred to as a strategy for model selection among a finite set of models (Clarke, 2019; Deborah et al., 2010). The model with the lowest BIC value is the most preferred as it is an indication of good clustering results (Clarke, 2019). Bayesian information criterion index is calculated as follows:

$$BIC=2\log(L) + q\log(N) \tag{2.25}$$

L is the maximum likelihood function of the model and N is the number of data points in a dataset (Clarke, 2019). In evaluating clustering quality using the bayesian information criterion, some limitations crop up (Clarke, 2019). These include: sensitivity to model complexity, limited consideration of cluster structure, dependency on initialization and algorithm choice and limited comparability across datasets (Clarke, 2019; Deborah et al., 2010).

BIC penalizes complex models to avoid overfitting, determining the appropriate number of clusters where the BIC tends to favor more clusters as it penalizes model complexity, potentially leading to over-segmentation or identifying spurious clusters (Clarke, 2019). Interpreting the BIC values and identifying the appropriate number of clusters often requires domain knowledge and additional analysis (Clarke, 2019). The BIC primarily focuses on the goodness of fit and model complexity but may not fully capture the underlying structure of the clusters (Clarke, 2019). It does not explicitly account for cluster separability, irregular cluster shapes, or overlapping clusters (Clarke, 2019). Therefore, the BIC alone may not provide a comprehensive assessment of clustering quality, and it is advisable to consider other metrics or visual inspections to validate the results (Clarke, 2019). The BIC can be sensitive to the initialization of clustering algorithms and the choice of algorithm itself (Clarke, 2019). Different initializations or algorithms may yield different BIC values and clustering results (Clarke, 2019). It is important to be aware of this dependency and perform multiple runs with different initializations or algorithms to obtain more robust results (Clarke, 2019). The BIC values may not be directly comparable across different datasets due to variations in data characteristics and underlying distributions (Clarke, 2019). It is more appropriate to compare BIC values within the same dataset or similar datasets rather than between different datasets (Clarke, 2019).

2.7.1.6 Point Bi-serial

This looks for the difference between the mean intra-cluster distance and the mean inter-cluster distance (Rendon et al., 2011). The point bi serial's formula is formulated as follows:

$$\bar{d}_s - \bar{d}_c * \frac{\sqrt{(\alpha * \beta / x^2}}{\alpha} \quad (2.26)$$

d_c is the distance from each data point and every other data point within the cluster while d_s refer to the distance from each data point and every other data point that is not within its cluster (Rendon et al., 2011). The α refers to the intra-cluster distances while the β refers to the number of the inter-cluster distances (Rendon et al., 2011). The *x* refers to the actual number of the point-pairs within a clustering dataset (Rendon et al., 2011). The σ refers to the standard deviation of all the distances (Rendon et al., 2011). Point bi-serial internal validation metric resembles the popular Silhouette metric, except the fact that it computes the separation from all the non-cluster sharing points, instead of only those that have the closest cluster (Rendon et al., 2011). When using point bi-serial to assess the quality of clusters, some limitations come up. These include: limited applicability to clustering, lack of ground truth labels, inadequate representation of cluster quality and difficulty in interpretation (Rendon et al., 2011). Point bi-serial is primarily designed to evaluate relationships between binary and continuous variables (Rendon et al., 2011). It may not directly apply to assessing clustering results, which involve grouping similar data points together (Rendon et al., 2011). Clustering typically deals with unsupervised learning and does not involve predefined binary variables that it requires (Rendon et al., 2011). Point bi-serial requires a binary variable as a reference point to calculate the correlation coefficient (Rendon et al., 2011). However, in clustering, there are typically no ground truth labels available to construct such a binary variable (Rendon et al., 2011).

Clustering is an unsupervised learning task, and the absence of ground truth labels makes it challenging to apply point bi-serial directly (Rendon et al., 2011). Point bi-serial assesses the strength and direction of the relationship between a binary variable and a continuous variable (Rendon et al., 2011). While this can be useful in certain analyses, it may not adequately capture the quality of clustering results (Rendon et al., 2011). The evaluation of the clustering quality often focuses on factors such as compactness, separation, or similarity within and between clusters, which are not directly addressed by it (Rendon et al., 2011). Lastly, the interpretation of the point-biserial can be challenging, as it measures the strength and direction of a relationship. In clustering, the goal is to assess the quality of clusters rather than the correlation between variables (Rendon et al., 2011). Therefore, the interpretation of point bi-serial in the context of clustering may not provide meaningful insights.

2.7.1.7 Sum of squares

This method adapts to the Calinski-Harabasz Index of the CH method:

$$\frac{trace(WCSM)}{trace(BCSM)} * k \tag{2.27}$$

Where trace (WCSM) refers to the trace of the within-cluster scatter matrix and measures the sum of squared deviations of points within each cluster from their respective cluster centroids. On the other hand, trace (BCSM) refers to the trace of the between-cluster scatter matrix and measures the sum of squared deviations of each cluster centroid from the overall mean (centroid) of the data.The *k* is the number of clusters in the dataset. Therefore, this metric is a reverse of the separation-compactness relationship (Rendon et al., 2011). For this reason, the change on the factor of normalization factor is drastic when the value of the *k* increases (Rendon et al., 2011).

Just like the Davies Bouldin index, the sum of squares metric divides compactness by separation (Rendon et al., 2011). Lower values in this metric indicate better clustering on a particular dataset (Rendon et al., 2011).

Limitations of using the sum of squares metric to assess the cluster quality include: sensitivity to cluster size and dimensionality, lack of normalization, dependency on initialization as well as the insensitivity to cluster shape (Rendon et al., 2011). The sum of squares is influenced by the number of data points in each cluster and the dimensionality of the data. In *k*-means clustering, for instance, clusters with a larger number of data points tend to have higher sum of squares values (Rendon et al., 2011). Similarly, in high-dimensional data, the sum of squares can be inflated due to the curse of dimensionality (Rendon et al., 2011). As a result, the sum of squares may not accurately reflect the quality of clusters in scenarios where the number of points or the dimensionality varies significantly (Rendon et al., 2011). The sum of squares does not inherently account for the scale or variance of the data (Rendon et al., 2011). It treats each feature equally and does not consider differences in magnitude or variability between features (Rendon et al., 2011). Consequently, clusters may be biased towards variables with larger scales or higher variances, potentially leading to a misleading assessment of cluster quality (Rendon et al., 2011). In iterative clustering algorithms like *k*-means, the initialization of cluster centroids can significantly affect the resulting sum of squares (Rendon et al., 2011). Different initializations can yield different cluster assignments and, consequently, different sum of squares values (Rendon et al., 2011). Consequently, the choice of initialization can impact the interpretation of cluster quality based on the sum of squares (Rendon et al., 2011). The sum of squares primarily measures the dispersion of data points within clusters (Rendon et al., 2011). However, it does not explicitly capture the shape or structure of the clusters (Rendon et al., 2011).

Clusters with different shapes, such as elongated or irregular clusters, may have similar sum of squares values despite their inherent structural differences (Rendon et al., 2011). Therefore, the sum of squares alone may not provide a comprehensive evaluation of cluster quality in terms of shape or compactness (Rendon et al., 2011). To address these limitations, it is often recommended to use the sum of squares in combination with other clustering evaluation metrics.

2.7.2 Accuracy

In machine learning, clustering algorithms are often evaluated using different metrics rather than a single accuracy score (Gikera et al.,2023a). Clustering is an unsupervised learning task, meaning that there are no ground truth labels available to directly calculate accuracy. Instead, various evaluation measures are used to assess the quality of clustering results (Gikera et al.,2023a). Here are some commonly used metrics:

2.7.2.1 Adjusted Rand Index (ARI)

Adjusted rand index measures the similarity between the true cluster assignments and the predicted clusters, considering all pairs of samples. ARI ranges from -1 to 1, where a higher value indicates better clustering quality (Robert et al., 2021). The following is the equation for the adjusted rand index:

$$ARI = (RI - E) / (max(RI) - E) \quad (2.28)$$

Where RI is the Rand Index that measure the similarity between two clusterings by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in both clusterings.

E is the expected value of the Rand Index under the hypothesis of random labeling and max (RI) is the maximum possible value of the Rand Index, which represents perfect agreement between the cluster assignments.

2.7.2.2 Normalized Mutual Information (NMI)

Normalized Mutual Information computes the mutual information between the true labels and the predicted clusters, normalized by entropy measures. NMI ranges from 0 to 1, with 1 indicating perfect clustering (Wang et al., 2019). The following is the equation for the normalized mutual information:

$$\text{NMI} = \frac{2.I(C;K)}{H(C)+H(K)} \tag{2.29}$$

Where set of true clusters, *k* is the set of predicted clusters, H(C) is the entropy of true clusters, measuring the uncertainty of true cluster assignments, H(K) is the entropy of predicted clusters, measuring the uncertainty of predicted cluster assignments. I(C;K) is the mutual information between true and predicted clusters, measuring the amount of shared information.

2.7.2.3 Homogeneity, Completeness, and V-measure

These three metrics provide a detailed evaluation of clustering. Homogeneity measures the extent to which each cluster contains only samples from a single class (Vysala and Gomes, 2020). Completeness measures the extent to which all samples from a given class are assigned to the same cluster. V-measure is the harmonic mean of homogeneity and completeness (Vysala & Gomes, 2020).

$$h = 1 - \frac{H(C|K)}{H(C)} \tag{2.30}$$

Where *h* is the homogeneity score of the clustering, H(C|K) is the conditional entropy of the class labels given the cluster assignmentsand the H(C) is the entropy of the class labels alone

$$H(C|K) = -\sum_{c.k} \frac{nck}{N} \log(\frac{nck}{nk}) \quad (2.31)$$

Where H(C|K) refers to the the conditional entropy of the class labels C given the cluster assignments K, *nck* is the number of samples that belong to class c and are assigned to cluster *k*, *nk* is the is the total number of samples assigned to cluster k while N is the total number of samples across all the clusters. It's important to note that the choice of metric depends on the specific problem and the nature of the data (Vysala & Gomes, 2020). Additionally, these metrics may not always capture all aspects of clustering quality, so it's often recommended to consider multiple metrics and interpret the results collectively (Gikera et al., 2023a).

2.7.3 Jaccard Coefficient

Jaccard coefficient is a cluster quality assessment metric that shows the degree of closeness between the clustered values and the actual values, and evaluates the ability of the clustering algorithm (Kongsin & Klongboonjit, 2020). Jaccard coefficient is used to investigate similarities between data points and the evaluation based on these similarities (Kongsin & Klongboonjit, 2020).

Jaccard Index = (the number in both sets) / (the number in either set) * 100

$$J(X,Y) = |X \cap Y| / |X \cup Y| \quad (2.32)$$

When using the Jaccard coefficient to assess cluster quality, some limitations prop up. These include: binary data requirement, sensitivity to set size, lack of ground truth labels, interpretation limitations and limited consideration of cluster structure (Kongsin & Klongboonjit, 2020).

The Jaccard coefficient assumes that the data is binary or can be converted into binary form. It calculates the similarity between sets by measuring the intersection over union (Kongsin & Klongboonjit, 2020). If the data is not naturally binary or cannot be converted to binary representation, the Jaccard coefficient may not be applicable (Kongsin & Klongboonjit, 2020). The Jaccard coefficient is sensitive to the size of the sets being compared (Kongsin & Klongboonjit, 2020). It tends to yield higher similarity values for smaller sets, even if they share a relatively small number of elements (Kongsin & Klongboonjit, 2020). As a result, the Jaccard coefficient may bias towards smaller clusters or clusters with fewer data points (Kongsin & Klongboonjit, 2020). The Jaccard coefficient, like other unsupervised clustering evaluation metrics, does not rely on ground truth labels (Kongsin & Klongboonjit, 2020). It assesses the similarity between clusters without considering the true underlying classes (Kongsin & Klongboonjit, 2020). This can limit its effectiveness as a standalone measure of clustering quality, as it does not account for the true clustering structure (Kongsin & Klongboonjit, 2020).

Interpreting the absolute value of the Jaccard coefficient can be challenging (Kongsin & Klongboonjit, 2020). It measures the similarity between sets, ranging from 0 to 1, where 1 indicates identical sets (Kongsin & Klongboonjit, 2020). However, determining an appropriate threshold or guideline to define good or bad clustering results based on the Jaccard coefficient alone can be subjective and context-dependent (Kongsin & Klongboonjit, 2020). The Jaccard coefficient focuses on measuring the similarity between sets or clusters without considering the internal structure of the clusters (Kongsin & Klongboonjit, 2020). It does not explicitly account for factors such as compactness, separation, or cluster shapes. Therefore, using the Jaccard coefficient alone may not provide a comprehensive assessment of clustering quality.

2.7.4 F1-Score

F1 score refers to the clustering metric that assesses the clustering algorithm's accuracy on a dataset (Yacouby & Axman, 2020). F1 score can be applied in the assessment of the systems used in binary classifications that cluster data points into two, for example, e.g. true / false or yes / no (Yacouby & Axman, 2020). Larger F1 scores are better than lower F1 scores, on a range of 0 and 1 (Yacouby & Axman, 2020).

The F1 score formula is as follows:

F-score = 2 * (precision * recall) / (precision + recall) (2.33)

Some of the limitations posed with the use of the F1-score in assessing the quality of clusters include: dependency on predefined metrics, sensitivity to imbalance, inability to capture cluster structure and difficulty in interpretation (Yacouby & Axman, 2020). To calculate the F1 score for clustering, it is necessary to define criteria for true positives, false positives, and false negatives (Yacouby & Axman, 2020). This requires specifying rules or thresholds to determine if two clusters should be considered as matching or not (Yacouby & Axman, 2020). Selecting appropriate criteria can be subjective and may vary depending on the specific clustering task and the nature of the data (Yacouby & Axman, 2020). The F1 score is influenced by the balance or imbalance of cluster sizes (Brodinová et al., 2019). If the clusters are imbalanced, with significantly different numbers of data points, it can affect the precision and recall values and consequently impact the F1 score (Yacouby & Axman, 2020). Imbalanced clusters can lead to biased F1 score results, as the metric may be more influenced by the larger clusters (Yacouby & Axman, 2020).

The F1 score evaluates the agreement between the predicted and ground truth cluster assignments based on a flat comparison (Yacouby & Axman, 2020). It does not explicitly consider the structure, shape, or relationships between clusters (Yacouby & Axman, 2020). Consequently, the F1 score may not fully capture the quality of clustering in terms of compactness, separation, or cluster interdependencies (Yacouby & Axman, 2020). Similar to other evaluation metrics, interpreting the absolute value of the F1 score can be challenging (Yacouby and Axman, 2020). There is no universally defined threshold or guideline to determine what constitutes a good or bad clustering result based on the F1 score alone (Yacouby and Axman, 2020). The interpretation of the F1 score should be performed in conjunction with other metrics, domain knowledge, and visual inspection to gain a comprehensive understanding of clustering quality (Yacouby & Axman, 2020).

2.7.5 Cochran's Q score

Cochran's Q test is a non-parametric statistical test applied in heterogeneous meta-analyses (Stephen & Saz, 2018). Cochran's Q score is based on the chi-square distribution (Stephen & Saz, 2018). It creates a probability that when maximized, it indicates high variation across the subjects of study as opposed to the variations of the subjects within a study (Rendon et al., 2011). In the evaluation process of clustering algorithms, Cochran's Q score can be used to investigate if different algorithms give different quality of clusters based on the same dataset (Stephen & Saz, 2018). Cochran's Q statistic is computed as follows:

$$T = k(k-1) = \frac{\sum_{j=1}^{k}(Xj - N/k)^2}{\sum_{i=1}^{b} Xi(k - Xi)} \quad (2.34)$$

Where, k is the number of treatments. $X\,j$ is the column total for the j^{th} treatment, The b is the number of blocks. Xi is the row total for the ith block and N is the grand total.

When using Cochran’s Q score to assess the quality of clusters, some limitations crop up. These include: difficulty in interpreting significance, sensitivity to sample size and cluster imbalance and insensitivity to cluster structure (Stephen & Saz, 2018). Cochran's Q test determines whether there are significant differences between groups (Stephen & Saz, 2018). However, it does not provide insights into the nature or magnitude of these differences (Stephen & Saz, 2018). Interpreting the significance of the test results in the context of clustering quality can be challenging without additional information or domain knowledge (Stephen & Saz, 2018).

Cochran's Q test can be sensitive to the sample size and the distribution of data across clusters (Stephen & Saz, 2018). Unequal cluster sizes or imbalanced data can influence the test results and potentially bias the assessment of clustering quality (Stephen & Saz, 2018). Cochran's Q test primarily evaluates the differences between groups without explicitly considering the structure or internal characteristics of clusters (Stephen & Saz, 2018). It does not account for factors such as compactness, separation, or cluster shapes (Stephen & Saz, 2018). Therefore, relying solely on Cochran's Q score may not provide a comprehensive assessment of clustering quality in terms of these important cluster characteristics (Stephen & Saz, 2018). Given these limitations, Cochran's Q test may not be the most suitable method for assessing the quality of clusters.

2.7.6 Chi Square

Chi-square is a non-parametric statistical metric used to measure of the variance between the observed and the expected recurrence of the results of a set of variables (McHugh, 2013; Whatley, 2022). Chi-square is important in the analysis of such differences in categorical variables of nominal in nature (Gaddis, 2015). A Chi square is computed as follows:

$$x^2 = \sum \frac{(Oi-Ei)^2}{Ei} \quad (2.35)$$

Where, X^2 represents the chi-squared statistic, *Oi* represents the observed frequency in the *i-th* category and the *Ei* represents the expected frequency in the *i-th* category.

2.7.7 T-test

T-test refers to a statistical test applied in the comparison of two categories (Semenick, 1990). In clustering algorithms, T-tests are normally applied in hypothesis testing to find out if an algorithm actually has an effect on a dataset, or whether two clustering algorithms are dissimilar from each other in terms of the performance and evaluation metrics used to in the assessment process in clustering a specific dataset (DeCoster, 2006). T-test, *t*, is computed as follows:

$$t = \frac{(\mathrm{X} - \mu 0)}{(\mathrm{s} / \sqrt{\mathrm{n}})} \quad (2.36)$$

Where *X* is the sample mean, $\mu 0$ represents the population mean, *s* is the standard deviation of the sample and *n* stands for the size of the sample. When using the Chi squre and T-test metrics to assess the quality of clusters, some limitations come up. These include: interpretation limitations, sensitivity to sample size and cluster imbalance and limited consideration of cluster structure (Park, 2009). These tests assess the independence between categorical variables or groups (Semenick, 1990). While statistical significance can be determined, interpreting the practical significance and meaningfulness of the results in the context of clustering quality can be challenging (Semenick, 1990). The tests provide insights into the presence or absence of associations between variables but do not provide direct information about the quality of clustering (Semenick, 1990).

These tests can be influenced by the sample size and the distribution of data across clusters (Semenick, 1990). Small sample sizes or imbalanced data may affect the test's power and potentially bias the assessment of clustering quality (Semenick, 1990). Moreover, unequal cluster sizes can impact the statistical significance of the the test results (Semenick, 1990). These tests focus on measuring associations between categorical variables but do not explicitly consider the structural characteristics of clusters (Semenick, 1990). They do not account for factors such as compactness, separation, or cluster shapes (Semenick, 1990). Therefore, relying solely on the Chi-square or T-test may not provide a comprehensive evaluation of clustering quality in terms of these important clusters' attributes. Considering these limitations, the Chi-square test may not be the most appropriate method for assessing the quality of clusters**.**

2.7.8 Sum of Squared Error (SSE)

The sum of squared error refers to the variation between the perceived values and the foreseen values (Nainggolan et al., 2019). Sum of squared error can also be referred to the difference between the foreseen values and the actual values (Nainggolan et al., 2019). In the evaluation of clustering algorithms, an example of the sum of squared error would be identifying the variation between the expected running time values of a clustering algorithms against the actual running time values of the same algorithm (Nainggolan et al., 2019). The sum of squared error is computed as follows:

$$SE = \sum_{i=1}^{n} (yi - f\ (xi))^2 \qquad (2.37)$$

Where *yi* is the *i*th value of the variable to be predicted, *f(xi)* is the predicted value and *xi* is the *ith* value of the explanatory variable.

The i is the index of summation, ranging from 0 to n. The squared difference between the actual value and the predicted value for the *i-th* data point is$(yi - f\ (xi))^2$. When using the sum of squared error, some limitations come up. These include: sensitivity to cluster size and density, dependency on initialization, lack of normalization, insensitivity to cluster shape and structure and lack of external validation (Nainggolan et al., 2019). The SSE is influenced by the number of data points in each cluster (Nainggolan et al., 2019). Clusters with larger numbers of data points tend to have higher SSE values, even if the clustering is of good quality (Nainggolan et al., 2019). Consequently, the SSE may be biased towards larger clusters and may not accurately reflect the overall clustering performance, especially in scenarios with varying cluster sizes or densities (Nainggolan et al., 2019). The SSE is sensitive to the initialization of cluster centroids, particularly in iterative algorithms like *k*-means clustering (Nainggolan et al., 2019). Different initializations can lead to different SSE values and, consequently, different cluster assignments (Nainggolan et al., 2019). The choice of initialization can impact the interpretation of clustering quality based on the SSE, making it less reliable as a standalone metric (Nainggolan et al., 2019). The SSE does not inherently account for the scale or variance of the data (Nainggolan et al., 2019). It treats each feature equally and does not consider differences in magnitude or variability between features (Nainggolan et al., 2019). Therefore, the SSE may be biased towards variables with larger scales or higher variances, potentially leading to a distorted assessment of clustering quality (Nainggolan et al., 2019). The SSE measures the dispersion of data points within clusters but does not explicitly capture the shape, structure, or inter-relationships between clusters (Nainggolan et al., 2019). Consequently, clusters with different shapes, densities, or structural complexities may have similar SSE values, even if they differ significantly in terms of their quality or characteristics (Nainggolan et al., 2019).

The SSE is an internal evaluation metric that assesses the compactness of clusters based on the distances between data points and their assigned cluster centroids (Nainggolan et al., 2019). However, it does not consider external validation or ground truth labels (Nainggolan et al., 2019). Without external validation, it is challenging to determine if the obtained clustering solution is meaningful or corresponds to the true underlying structure of the data (Nainggolan et al., 2019). To overcome these limitations, it is recommended that the SSE is used in conjunction with other clustering evaluation metrics, such as silhouette coefficient, adjusted Rand index, or other suitable measures (Gikera et al., 2023a).

2.8 Critical Review of the *K*-hyperparameter Tuning Techniques in High Dimensional Space Clustering

The critical review strategy on the *k*-means based techniques for the *k*-hyperparameter tuning techniques in high-dimensional space clustering focuses on the identification of the relevant literature, assessment on the quality of articles, critical review summary of the reviewed literature, and lastly, the interpretation of results. This approach is aligns with the proposal by Khan, Kunz, Kleijnen, and Antes for conducting critical review (Khan et al., 2003). The meta-search based strategy is applied to gather relevant recent articles focusing on *k*-hyperparameter tuning techniques within high-dimensional datasets from Google Scholar, ACM, Research Gate, IEEE Xplore, and Springer. The approach of narrowing the focus on these articles, in the critical review analysis, is determined by the level of methodological rigour, evaluation rigour as well as the significance and novelty. After the critical review process, a number of recommendations for the future research directions, in the last section are proposed. This provides a guide and a foundation for further work on solving the *k*-hyperparameter tuning problems in high-dimensional space clustering.

The critical review process is based on the desktop research design. Besides identifying gaps with the existing techniques and the dimensionality reduction methods, this critical review process is also used in the development of a conceptual framework for this research study. Table 2.5 presents the key findings from the critical analysis on these techniques and dimensionality reduction mtrthods, and how these findings guides the development of the conceptual framework, experimental and the model development methodology in chapter three.

Table 2.2 presents the description of the *k*-hyperparameter tuning techniques in high-dimensional space clustering, data dimensionality reduction methods used with these techniques as well as their tuning strategies and limitations. Table 2.3 presents the analysis of the nature, description and dimensionality of the datasets used with the existing *k*-hyperparameter tuning techniques in high dimensional space clustering. Moreover, a theoretical analysis of the performance of the existing *k*-hyperparameter tuning techniques is performed as a preliminary to the empirical analysis outlined in the research methodology. This theoretical analysis uses the key algorithm performance and statistical metrics for evaluating the existing *k*-hyperparameter tuning techniques in high-dimensional space clustering. The results on these evaluation metrics are reported in Table 2.4. The key findings from the critical review are presented and how these findings guide the development of both the experimental research design and model development methodology in this research study. Theoretical analysis focuses on studying the performance and behavior of algorithms in a formal and abstract manner. During the theoretical analysis on the existing *k*-hyperparameter tuning techniques, formal proofs and reasoning are used to establish the efficacy of these techniques. The key findings are presented in Table 2.5.

2.8.1 Analysis of the *K*-hyperparamter Tuning Techniques, their Tuning Strategies and Dimensionality Reduction Methods

The analysis process begins with tabulating the *k*-hyperparameter tuning techniques, dimensionality reduction methods, and their respective tuning strategies, providing a structured overview and summary. Secondly, to identify a clear research gap, the author engages in a detailed discussion and critique of the tabulated analysis, evaluating both the strengths and limitations in the existing approaches. This analysis is concluded by highlighting key findings that underscore the limitations in the current approaches, reinforcing the identified research gap and guiding the direction for further investigation in this research study.

Table 2.2 presents the summary results of the names of the state of the art *k*-hyperparameter tuning techniques, dimensionality reduction methods applied and a brief description of each technique including its strengths and limitations, as proposed by Gikera et al. (2023a). Table 2.3 presents the names and nature of the high-dimensional datasets used with the techniques, as proposed by Gikera et al. (2023a).

Lastly, Table 2.4 presents a summary of the algorithms' performance and statistical metrics as well as the specific metrics 'scores for each and every technique, as proposed by Gikera et al., (2023a). Each of the results' findings is aligned to the research questions in this research study. Table 2.2 provides an in-depth overview of advanced techniques for *k*-hyperparameter tuning. Each technique is described with its specific methods for determining the optimal *k*-value. The table also details associated data dimensionality reduction methods, explaining their role in the tuning process. Furthermore, it highlights both the strengths and limitations of each technique.

Strengths are outlined to showcase the advantages while limitations are noted to identify areas of improvement. To address these limitations, the table offers targeted recommendations for future research and development. These suggestions aim to address current challenges and drive improvements in the *k*-hyperparameter tuning methodologies, providing valuable guidance for future research work in the *k*-hyperparameter tuning in high dimensional space clustering.

Table 2.2: Description of the *K*-hyperparameter Tuning Techniques, Dimensionality Reduction Methods as well as their Tuning Strategies and Limitations .

Reference	Technique	Description, tuning strategy and limitations	Dimensionality Reduction Method
(Onumanyi et al., 2022)	AutoElbow	In this technique, the elbow graph is normalized using the lowest and the highest values along the coordinates of both the ordinate and abscissa. The estimated elbow, i.e. the *k*-hyperparameter, is the point that maximizes the distance between each point on the graph to the minimum, maximum reference points as well as the "heel" of the elbow graph. Although the technique performs relatively well, it has the limitation of the fact that the auto-elbow graph may not depict a sharp elbow with some high-dimensional datasets. Also, as the number of dimensions increases, the space becomes increasingly sparse, leading to limitations in adequately representing the data. Normalization might not effectively address this issue, as distances between points can become less meaningful in high-dimensional spaces. Extreme values in high-dimensional spaces can significantly influence normalization.	None

		Outliers along different dimensions might distort the normalization process, affecting the effectiveness of the *k*-hyperparameter tuning technique.	
(Yan et al., 2020)	Adaptive Multi-view Subspace Clustering for High-dimensional Data	This technique is an extension of the canonical *k*-means algorithm where the feature learning mechanism is integrated in order to handle the high-dimensional space with multiple perspectives. Although the experimental results with four different datasets shows that the *k*-hyperparameter technique is relatively effective, its limitation is the fact that the effectiveness of *k*-hyperparameter tuning may vary across its different views due to their distinct characteristics. Finding an optimal *k* value that fits all views simultaneously can be challenging and is a limitation for this technique. Also, as the number of dimensions increases, determining relevant subspaces and optimizing *k* in each subspace becomes more computationally demanding. Lastly, integrating information from multiple views while at the same time tune the *k*-hyperparameter can increase the complexity of the tuning process. Combining diverse views effectively to determine the optimal number of clusters (*k*) becomes more intricate.	None
(Tao et al., 2020)	An Intelligent Clustering Algorithm for High-Dimensional Multiview Data in Big Data Applications	This technique uses intelligent weighting *k*-means clustering approach to deal with the limitations of having to consider all features of a high-dimensional dataset with equal relevance. At first, the coupling degree between clusters is presented in the clustering model in order to increase the level of dissimilarity among the clusters. Several features are applied in the computation of the weighting distance function used to identify the objects' clusters.	PCA

		In the second step, swarm intelligence is used to minimize the sensitivity of the initial cluster centers and the weights of features via a global search. However, the performance of the swarm intelligence algorithms can be sensitive to the choice of parameters, population size, and tuning parameters, especially in high-dimensional spaces, making their behavior less predictable Extracting meaningful tuning feature parameters from such datasets requires efficient dimensionality reduction methods.	
(Xia et al., 2020)	Ball *k*-means	The technique uses a ball to describe a cluster with the intention of minimizing the point-centroid distance calculation through enclosing data points within more effective and efficient hyperspheres or "balls" in high dimensional spaces. Each cluster is separated into a stable area and an active area; active area is then subdivided into equal portions of annular area. This *k*-hyperparameter tuning technique uses the ball clusters and neighbor searching strategy along with a number of novel stratagems to lower the computations of the centroid distances. This technique's iteration time complexity drops to sub linear levels as the progress on the iterations is made. This makes it more efficient. However, the performance degradation on some datasets with high distance computations is a draw back with this technique. Moreover, the optimal arrangement of hyperspheres to encapsulate data points becomes complex and challenging in high-dimensional spaces, affecting the accuracy of the cluster representation and subsequently the *k*-hyperparameter tuning process.	None

(Wang et al., 2019)	Fast Adaptive K-Means Subspace Clustering for High-Dimensional Data	In this *k*-hyperparameter tuning technique, an adaptive loss is created to give an adjustable cluster indicator computation approach in order to handle high-dimensional datasets that possess different distributions. In this technique, the feature selection processes as well as the clustering process are done simultaneously. However, identifying relevant subspaces for clustering within high-dimensional spaces is a non-trivial exercise. Efficiently determining which lower-dimensional structures contain meaningful clusters, and adapting the algorithm accordingly, adds complexity to the tuning process. Moreover, with this *k*-hyperparameter tuning technique, excessive feature reduction on the original high-dimensional datasets degrades the quality of clustering results. An effective feature reduction method that strikes a balance between the reduced features and the cluster quality is critical to the success of this technique.	PCA
(Lu, 2019)	Improved K-Means Clustering Algorithm for Big Data Mining under Hadoop Parallel Framework	At first, the data points' density is computed and each cluster contains center points whose density is not lower than the threshold and the supplied density range. The basic cluster is combined in relation to the distance between the two cluster centers while the points that are not divided into any clusters are divided into the clusters near to them. However, effectively partitioning high-dimensional data across the Hadoop cluster and managing the communication overhead during the *k*-hyperparameter tuning process can impact the performance of the algorithm.	PCA

		Ensuring the algorithm's robustness, in handling high dimensional dataset with irrelevant features and noise, within a distributed framework is a significant limitation. Adapting the improved K-Means Clustering Algorithm to work efficiently within the distributed and parallel architecture of the Hadoop framework while considering the specifics of high-dimensional data introduces limitations during the *k*-hyperparameter tuning process. Moreover, selecting the initial cluster centers is a limitation with this technique.	
(Xie et al., 2019)	Improving K-means clustering with enhanced Firefly Algorithms	In this *k*-hyperparameter tuning technique, two variants of firefly based algorithms are proposed. These include the inward intensified exploration firefly algorithm as well as the compound intensified exploration firefly algorithm. These variants are meant to solve the limitations of initialization sensitivity and trappings of the local optima on *k*-means. A matrix-based search parameters and dispersing strategies are used with the two firefly models in order to improve the capability of exploitation and exploration processes. The attractiveness coefficient with a randomized control matrix in the inward intensified exploration firefly algorithm model is first replaced in order to release it from the biological law constraints. The exploitation in the neighborhood is lifted from a one-dimensional to multiple dimensional search strategy with improved diversification in terms of search scopes, scales, as well as directions. A dispersing strategy in the compound intensified exploration Firefly Algorithm is employed to generate similar fireflies to new positions out of the close neighborhood in order to execute the global exploration process.	PCA

		In order to increase the efficiency in the search process, a sufficient variance between fireflies is created. The strength of the use of firefly algorithms in the *k*-hyperparameter tuning process is the fact that firefly algorithms possess attraction movements that help swarm to subdivide into smaller groups, automatically. In this case, each smaller group swarms around one mode or a local optimum solution. However, Firefly Algorithms are sensitive to the choice of parameters and settings. In high-dimensional spaces, finding appropriate parameters and optimization settings becomes more complex, affecting the overall performance of the algorithm in *k*-hyperparameter tuning. Fine-tuning the parameters of the Firefly Algorithms through effective dimensionality reduction method is critical. The adaptability of Firefly Algorithms to effectively explore and optimize the search space for *k*-values from a high dimensional space can be intricate. Moreover, the presence of redundant, noisy, and irrelevant features in this technique severely affects the model's performance.	
(Rustam et al., 2019)	Kernel Spherical K-Means and Support Vector Machine for Acute Sinusitis Classification	In this *k*-hyperparameter tuning technique, the popular machine learning based Kernel Spherical K-Means (KSKM) as well as the Support Vector Machine (SVM) have been utilized. The technique was improved through the modification of the inner product with the kernel function so as to make sure that the separation of the linear data in high dimensions is achieved and thereby enhance the *k*-hyperparameter tuning performance of this technique.	kPCA

		On the other hand, the Support Vector Machine is a binary based classification strategy that assists in developing models that poses good generalization ability. The results, evaluated and compared on a number of datasets is a confirmation that this *k*-hyperparameter tuning technique is superior as compared to the baseline models. Both the clustering accuracy and the running time are better as compared to the baseline models. However, the tuning process of the kernel parameters using the grid search method is a challenging task when handling datasets of high-dimensionality. For such relatively high-dimensional spaces, it would be important to apply more efficient search methods as opposed to the grid search. Moreover, choosing the appropriate kernel function and its parameters for both the KSKM and SVM becomes challenging in high-dimensional spaces. Different kernels might perform differently, and finding the best kernel configuration becomes complex. The best choice on the kernel function and its parameters play a crucial role in the dimensionality reduction process, allowing this algorithm to capture intricate relationships and patterns into a new space where the *k*-hyperparameter tuning exercise becomes effective.	
(Hussain & Haris, 2019)	K-means based Co-clustering Algorithm for Sparse, High Dimensional Data	The uniqueness and significance of this *k*-hyperparameter tuning technique is that it embeds the higher order statistics as well as the co-clustering strategies in its tuning process as opposed to just using them as an external distance measure. This technique presents a unified framework referred to as "*K*-means based Co-clustering Algorithm for Sparse, High Dimensional Data".	PCA

		The step for the initialization process is modified to incorporate several points representing cluster centers in a way that the within-cluster points are near each other but far from the points in the other clusters. In an iterative process, the neighborhood based walk statistics is proposed as a semantic similarity technique for both the assignment as well as the re-estimation of the center in this algorithm. The results, on a number of standard datasets, demonstrate the effectiveness of this technique as compared to other baseline models and state-of-the-art improvements. However, interpreting co-clusters in sparse high-dimensional data presents limitations. Understanding the grouping of both rows and columns simultaneously for meaningful insights can be intricate, making the *k*-hyperparameter tuning a challenge. Adoption of effective dimensionality reduction method that merges correlated features into fewer representative features without losing essential information is critical to a successful *k*-hyperparameter tuning process when using this technique. Moreover, the running time is high in cases where the dataset has a large number of clusters.	
(Rezaee et al., 2020)	GBK-means clustering algorithm: An improvement to the K-means algorithm based on the bargaining game	The Game Based K-means (GBK-means) is a *k*-hyperparameter tuning technique that utilizes the strength of the bargaining game modelling to cluster high-dimensional datasets. With this technique, the cluster centers (players) compete to attract as many objects as possible to their own cluster. The payoff for the players is maximized after a successful bargaining with each other. For this reason, these centers keep moving from one position to another in a way that they possess lower distances with the highest possible data compared with other centers.	PCA

		The evaluation process demonstrates that this *k*-hyperparameter tuning technique clusters a dataset with relatively higher accuracy as compared to the other baseline models and the state-of-the-art tuning techniques. However GBK-means might be sensitive to the choice of its parameters and settings. Tuning these parameters effectively in high-dimensional spaces to achieve optimal clustering performance and hyperparameter tuning can be challenging. The adaptability of GBK-means to effectively explore the search space for *k*-values in high dimensional spaces can be intricate. Moreover, the continuous movement of the cluster centers (players) is an iterative process that could be computationally expensive for a dataset with high dimensionality.	
(Dey et al., 2020)	Lasso Weighted *k*-means	The Lasso Weighted *k*-means (LW-*k*-means) is a *k*-hyperparameter tuning technique that applies L_1 regularization term with feature weights that triggers feature selection within the framework of sparse clustering. A simple block-coordinate descent type algorithm is developed with a time-complexity that resembles Lloyd's strategy with an aim of optimizing the proposed objective. At the same time, strong consistency of the LW-*k*-means procedure is established. This technique, validated on several datasets via a rigorous experimentation process, shows that this *k*-tuning technique is extremely competitive in performing high-dimensional clustering as compared to the other baseline models as well as the other state-of-the-art models. The scores of both the clustering accuracy and the computational time are relatively good as compared to the ones in the other techniques.	t-SNE

		However, the adaptability of the Lasso Weighted *k*-means to explore the search space for *k*-values efficiently can be challenging. The performance of Lasso Weighted *k*-means might be sensitive to the choice of its parameters, such as the Lasso regularization parameter. Tuning these parameters effectively in high-dimensional spaces can be intricate, subsequently making the *k*-hyperparameter tuning process a challenging exercise. Moreover, deploying weights on datasets with extremely high dimensionality, for example, the gene microarray datasets is a challenge. Therefore, the use of a fuzzy system to help in assigning a probabilistic interpretation of the weights of features could be worth investigating in future research works.	
(Brodinová et al. , 2019)	Robust and sparse *k*-means clustering for high-dimensional data	This technique incorporates a weighting function that paves way to an automated assignment of weights on every observation. A high weight on an observation means that a data point belongs to a cluster while low weight means that a data point is a potential outlier. To cope with noisy variables, an objective function referred to as a lasso-type penalty is applied. A framework for determining both the number of clusters, *k*-hyperparameter and variables based on a modified gap statistic is introduced. However, the performance of the Robust and Sparse K-means algorithm might be sensitive to the choice of parameters, such as the sparsity-inducing penalties or robustness measures. Tuning these parameters effectively in high-dimensional spaces can be intricate. Combining Robust and Sparse K-means with a robust dimensionality reduction method that has effective regularization methods might offer more effective solutions for *k*-hyperparameter tuning in high-dimensional spaces	PCA

		when using this technique. Moreover, the process of applying weighting functions as well as updating the variable weights is repeated iteratively until there is stabilization of the variable weights. This could be computationally expensive for a dataset with relatively high dimensionality, for example, the proteomic and genes based datasets. A technique for pre-processing such a high-dimensional dataset before being clustered could be a solution for this. At the same time, identifying outliers from the noisy variables is challenging as the noisy variables are assigned a weight of zero.	
(Orkphol & Yang, 2019)	Sentiment Analysis on Micro blogging with *K*-Means Clustering and Artificial Bee Colony	This technique uses the frequency–inverse document frequency strategy to select the important features from a micro blogging high-dimensional dataset. The dimensionality reduction is performed using the Singular Value Decomposition method. In order to search for a global optimum, the popular artificial bee colony is applied in the determination of the best initial centroids. Silhouette internal validation index is then used to determine the optimal value of *k*. However, the adaptability of the Artificial Bee Colony to find the optimal *k*-value efficiently might be intricate in high-dimensional text spaces. Encoding text data into a format suitable for clustering while retaining its semantic meaning is complex. Converting text to numerical representations (e.g., using the term frequency-inverse document frequency) requires careful selection of the dimensionality reduction method that effectively captures the essence of the text, and the most relevant features.	SVD

		Moreocver, in order to use the Silhouette index in the determination of the optimal *k*-hyperparameter, several *k*-values have to be supplied in order to compare the one that returns the best Silhouette index score. This makes the technique computationally expensive. At the same time, inconsistencies may occur where the Silhouette index generates the best score at an optimal *k*-value that is different from another internal validation index e.g. Dunn index or Davies Bouldin index. In this case, it is recommended that an ensemble internal validation index is used, via boosting or bagging, whose components exercise equal sensitivity to the varied conditions present in the high dimensional datasets. The optimal *k*-hyperparameter value would then be reached at through the ensemble's voting scheme i.e. the one that is returned by most of the internal validation indexes.	
(Song et al., 2021)	A Fast Hybrid Feature Selection Based on Correlation-Guided Clustering and Particle Swarm Tuning for High-Dimensional Data	This technique utilizes an integrated three-phase hybrid system using the correlation-guided clustering and particle swarm tuning. During the first and the second steps, a filter mechanism and a feature clustering-based method, both of low computational costs, are developed in order to minimize the search space. During the third step, it finds an optimal feature subset using an evolutionary algorithm with the global mechanism for searching. However, correlation-guided clustering using this technique might face limitations due to the multitude of features and their interactions. Identifying and handling the correlated features effectively is crucial for the feature selection process.	Hybrid (PCA & (t-SNE)

		The adoption of an effective dimensionality reduction method that can capture correlated features in a more compact representation, reducing the impact of inter-feature correlations, is critical to the success of this technique in the *k*-hyperparameter tuning process. Moreover, this technique faces the drawback of the fact that for data with a huge number of samples, it faces the limitation of extremely high computational costs.	
(Dey et al., 2020)	The Sparse Min-Max *k*-Means Algorithm for High-Dimensional Clustering	The sparse Min-Max *k*-Means Clustering strategy reformulates the objective function of the Min-Max *k*-Means algorithm into a new form of weighted between-cluster sum of squares. The sparse regularization is imposed on these weights in order to make it useful in the clustering of high-dimensional space. However, the performance of the Sparse Min-Max *k*-Means Algorithm might be sensitive to the choice of its parameters, such as sparsity-inducing penalties. Tuning these parameters effectively in high-dimensional spaces can be intricate, subsequently making the *k*-hyperparameter tuning process a challenging task. The adoption of an Elastic Net Regularization combining L1 (Lasso) and L2 (Ridge) penalties in order to encourage sparsity while controlling for the sensitivity to the individual parameter choices is critical to the success of this technique. The technique, moreover, degrades in its performance when noisy and irrelevant variables are present in a high-dimensional dataset.	PCA

(Hozumi et al., 2021)	UMAP-assisted K-means clustering of large-scale SARS-CoV-2 mutation datasets	This technique uses the UMAP data dimensionality reduction method to convert the high-dimensional space into low dimensional space. The UMAP is nonlinear and uses three assumptions i.e. a dataset has uniform distribution on Riemannian manifold, the Riemannian metric has a local constant, and the there is a local connection of the manifold. UMAP creates a graph representation of each of the original highdimensional space in form of a predefined *k*-dimensional weighted UMAP. This is accomplished in such a way as to minimize the edge-wise cross-entropy between the weighted graph and the original data. Lastly, the UMAP graph's *k*-dimensional eigenvectors are used to represent each of the original data space. However, converting genetic mutation data into suitable feature representations for clustering while preserving biological relevance is complex. Mapping genetic features onto a space suitable for clustering via UMAP might require domain-specific knowledge and preprocessing. Genetic mutation datasets often contain noise, variability, irrelevant features and inherent complexity. Handling these characteristics to ensure effective dimensionality reduction of the original high dimensional features is crucial for meaningful biological interpretations when using this technique. Moreover, this technique does not perform well on a dataset that possess complex non-linear structure or on a dataset that possess non-uniform density.	UMAP

Table 2.3 provides detailed information on the high-dimensional datasets utilized by various state-of-the-art *k*-hyperparameter tuning techniques. For each technique, the table lists the dataset's name and characterizes its nature, including data type, dimensionality, number of instances, and number of variables. The dataset's dimensionality is expressed in terms of the order of magnitude of attributes, typically measured in powers of ten. This comprehensive overview helps to understand the scale and complexity of datasets involved in the tuning process, offering insights into how different techniques handle varying data dimensions and structures. By presenting these details, the table facilitates a better comparison of the techniques based on the nature of the datasets they are designed to work with.

Table 2.3: Nature of the Datasets used with the *K*-hyperparameter Tuning Techniques in High Dimensional Space Clustering.

Reference	Technique	Description of the Nature of Dataset
(Onumanyi et al., 2022)	AutoElbow	Cleveland heart disease dataset, containing 13 features, 303 instances and the *k*-hyperparameter value is 5. This dataset is of the type text and dimensionality of 10^1.
(Yan et al., 2020)	Adaptive Multi-view Subspace Clustering for High-dimensional Data	Caltech, Jaffe, handwritten and Yale datasets are used with this technique. Caltech contains 8677 images from 101 categories, Jaffe contains 213 samples and 10 classes, handwritten dataset contains 2000 samples and 10 classes while the Yale contains 165 gray images of 15 individuals. These datasets are of the type text and images with a dimensionality of 10^1 and 10^2.

(Tao et al., 2018)	An Intelligent Clustering Algorithm for High-Dimensional Multiview Data in Big Data Applications	Multiple Features (Mfeat) dataset, Internet Advertisement data set, Spambase data set, Segmentation data set and Cardiotocography data set are used with this technique. Mfeat contains 2,000 objects and 649 features in 10 clusters. Internet advertisement dataset contains 2359 objects in 2 clusters and 1557 features. Spam base dataset contains 4601 objects in 2 clusters and 57 features. Image segmentation dataset contains 2310 objects in 7 clusters and 57 features while the cardiotocography contains 2126 objects in 3 clusters and 21 features. These datasets are of the type text and images with a dimensionality of 10^1, 10^2 and 10^3.
(Xia et al., 2020)	Ball *k*-means	Four-class, svmguide1, codma, keg network, epileptic, birch and ijcnn are the datasets used in this technique. Four class dataset is of the size 862 with 3dimensions. Svmguide1 dataset is of the size 7088 and 5dimensions. Codrna dataset is of the size 59535 with 8 dimensions. Kegg-Network dataset is of the size 65554 with 28dimensions.Epileptic is of the size 11500 and 179 dimensions. Birch3 dataset is of the size 100000 and 2dimensions. Ijcnn dataset is of the size 141690 and 22 dimensions while RNA-Seq contains 20531 dimensions. These datasets are of the type text and images with a dimensionality of 10^1, 10^2, $10^3$and 10^4.
(Wang et al., 2019)	Fast Adaptive K-Means Subspace Clustering for High-Dimensional Data.	Glass, breast, vehicle, Umist, Yale, WebKB and TD2 are the datasets used in this technique. Glass contains 6 clusters, 214 instances and 9 features. Breast contains 2 clusters, 699 instances and 10 features. Vehicle contains 4 clusters, 846 instances and 18 features. Umist contains 20 clusters, 575 instances and 644 features. Yale contains 15 clusters, 165 instances and 1024 features.

		WebKB contains 7 clusters, 814 instances and 4029 features. TD2 contains 10 clusters, 653 instances and 36771 features. These datasets are of the type text and images with a dimensionality of 10^1, 10^2, 10^3 and 10^4.
(Lu, 2019)	Improved K-Means Clustering Algorithm for Big Data Mining under Hadoop Parallel Framework.	HIGGS data set containing 11 million records and 28 features. This dataset is of the type signal images and with a dimensionality of 10^1.
(Xie et al., 2019)	Improving K-means clustering with enhanced Firefly Algorithms.	Acute Lymphoblastic Leukaemia (ALL), Sonar, Ozone, Wisconsin breast cancer diagnostic data set (Wbc1), Wisconsin breast cancer original data set (Wbc2), Wine, Iris, Balance, Thyroid, E. coli, Drivface, Micromass, sensor, Human Activity, Skin Lesion, Mice Protein, and Libras are the datasets used with this technique. Sonar contains 60 features, 2 clusters, and 140 instances. Ozone contains 72 features, 2 clusters, and 196 instances. ALL contains 80 features, 2 clusters, and 100 instances. WBC1 contains 30 features, 2 clusters, and 561 instances. WBC2 contains 9 features, 2 clusters, and 683 instances. Wine contains 13 features, 3 clusters, and 178 instances. Iris contains 4 features, 3 clusters, and 150 instances. Balance contains 4 features, 2 clusters, and 576 instances. Thyroid contains 5 features, 3 clusters, and 90 instances. Ecoli contains 7 features, 3 clusters, and 150 instances. Drivface contains 6400 features, 3 clusters, and 81 instances.

		Micromass contains 1300 features, 5 clusters, and 180 instances. Sensor contains 128 features, 5 clusters, and 415 instances. Human Activity contains 560 features, 2 clusters, and 600 instances, Skin Lesion contains 98 features, 2 clusters, and 660 instances, Mice Protein contains 77 features, 2 clusters, and 300 instances. Libras contains 90 features, 2 clusters, and 72 instances. These datasets are of the type text and images with a dimensionality of 10^1, 10^2, and 10^3.
(Rustam et al., 2019)	Kernel Spherical K-Means and Support Vector Machine for Acute Sinusitis Classification	Acute Sinusitis Data which contains 4 features, 200 instances and 2 clusters of acute and non-acute sinusitis.
(Hussain & Haris, 2019)	K-means based Co-clustering Algorithm for Sparse, High Dimensional Data.	M2, M5, M10, Cornell, Washington and Cora datasets have been used with this *k*-hyperparameter tuning technique. M2 – This dataset contains 20,000 news group documents from 20 different newspapers and has 2 clusters. M5 – This dataset is similar to M2 but with more number of categories and less number of documents as compared to those in M2. M10 – This dataset is similar to M5 but with more number of categories and less number of documents as compared to those in M5 and M2. Cornell – This dataset contains 195 documents containing 1703 words and in 5 clusters.

		Cora – This dataset contains 2708 documents, 1433 words with each document belonging to one of 6 classes. Washington – This dataset contains 230 documents and has five classes. These datasets are of the type text and with a dimensionality of 10^3.
(Rezaee et al., 2020)	GBK-means clustering algorithm: An improvement to the K-means algorithm based on the bargaining game.	Australian Credit, Breast Cancer, Breast Wisconsin, Diabetes, Haberman's Survival, Heart Disease, Hepatitis, Ionosphere, Japanese Credit and Mammographic are the datasets used with this *k*-hyperparameter tuning technique. Australian Credit dataset contains 690 instances and 14 attributes. Breast Cancer dataset contains 569 instances and 30 attributes. Breast Wisconsin dataset contains 699 instances and 9 attributes. Diabetes dataset contains 768 instances and 8 attributes. Haberman's Survival dataset contains 306 instances and 3 attributes. Heart Disease dataset contains 303 instances and 13 attributes. Hepatitis dataset contains 155 instances and 20 attributes. Ionosphere dataset contains 351 instances and 34 attributes. Japanese Credit dataset contains 690 instances and 15 attributes. Mammographic dataset contains 961 instances. These datasets are of the type text with a dimensionality of 10^1.
(Dey et al., 2020)	Lasso Weighted k-means.	Brain, Leukemia, Lung cancer, Lymphoma, Wine, Coil_5, ORL_5, YALE_5, ALLAML, Appendicitis, SuCancer, Iris, Glass, Tae, Zoo, Cleveland, Leaf, Vowel, Ecoli, Hebaerman and the WDBC are the datasets used in this technique. Brain - Brain dataset has 5 clusters and contains 42 instances and 5,597 features Leukemia - Leukemia dataset has 2 clusters and contains 72 instances and 3,571 features.

		Lung cancer – Lung cancer dataset has 2 clusters and contains 181 instances and 12,533 features Lymphoma - Lymphoma dataset has 3 clusters and contains 62 instances and 4,026 features Wine - Wine contains 13 features, 3 clusters, and 178 instances Iris - Iris contains 4 features, 3 clusters, and 150 instances Cleveland - Cleveland heart disease dataset, containing 13 features, 303 instances and the *k*- value is 5 Ecoli - Ecoli contains 7 features, 3 clusters, and 150 instances. These datasets are of the type text and images with a dimensionality of 10^1, 10^3 and 10^4.
(Brodinová et al., 2019)	Robust and sparse k-means clustering for high-dimensional data.	Synthetic dataset was used in this technique. The synthetic dataset consists of 40 observations, 50 features and 3 clusters. This dataset is of the type text with a dimensionality of 10^1.
(Orkphol & Yang, 2019)	Sentiment Analysis on Micro blogging with K-Means Clustering and Artificial Bee Colony.	Twitter dataset was used to evaluate this algorithm. This dataset contains 1,000 non-redundant tweets and a polarity of 228 positive tweets, 104 negative tweets, and 668 neutral tweets. This dataset is of the type text with a dimensionality of 10^3.

(Song et al., 2021)	A Fast Hybrid Feature Selection Based on Correlation-Guided Clustering and Particle Swarm Tuning for High-Dimensional Data.	Arrhythmia, SCADI, GFE, Prostate, MFD, Coil 20, Yale_64, Colon, SRBCT, WrapAR10P, Leukemia_Small, DBWorld, DLBCL, Drv_face, Leukemia_Big, CNS, Lung and Ovarian are used as the datasets with this technique. Arrhythmia contains 195 features, 452 samples and 16 clusters. SCADI contains 205 features, 70 samples and 7 clusters. GFE contains 301 features, 743 samples and 2 clusters. Prostate contains 339 features, 102 samples and 2 clusters. MFD contains 649 features, 700 samples and 10 clusters. Coil 20 contains 1,024 features, 200 samples and 20 clusters. Yale_64 contains 1,024 features, 165 samples and 15 clusters. Colon contains 2,000 features, 62 samples and 2 clusters. SRBCT contains 2,304 features, 83 samples and 4 clusters. WrapAR10P contains 2,400 features, 130 samples and 10 clusters. Leukemia_Small contains 3,571 features, 72 samples and 2 clusters. DBWorld contains 4,703 features, 64 samples and 2 clusters. DLBCL contains 5,469 features, 77 samples and 2 clusters. Drv_face contains 6,400 features, 606 samples and 3 clusters. Leukemia_Big contains 7,128 features, 72 samples and 2 clusters. CNS contains 7,129 features, 60 samples and 2 clusters. Lung contains features 12,600, 203 samples and 2 clusters. Ovarian contains 15,154 features, 253 samples and 2 clusters. These datasets are of the type text and images with a dimensionality of 10^2, 10^3 and 10^4.
(Dey et al., 2020)	The Sparse MinMax *k*-Means Algorithm for High-Dimensional Clustering.	Brain, breast cancer, colon cancer, leukemia, lung cancer 1, lung cancer 2, lymphoma, prostate cancer, SRBCT and suCancer datasets have been used in this technique. Brain dataset has 5 clusters and contains 42 instances and 5,597 features. Breast cancer dataset has 2 clusters and contains 276 instances and 22,215 features.

		Colon cancer dataset has 2 clusters and contains 62 instances and 2,000 features. Leukemia dataset has 2 clusters and contains 72 instances and 3,571 features. Lung cancer 1 dataset has 2 clusters and contains 181 instances and 12,533 features. Lung cancer 2 dataset has 2 clusters and contains 203 instances and 12,600 features. Lymphoma dataset has 3 clusters and contains 62 instances and 4,026 features. Prostate cancer dataset has 2 clusters and contains 102 instances and 6,033 features. SRBCT dataset has 4 clusters and contains 63 instances and 2,308 features. SuCancer dataset has 2 clusters and contains 174 instances and 7,909 features. These datasets are of the type text and images with a dimensionality of 10^3 and 10^4.
(Hozumi et al., 2021)	UMAP-assisted *k*-means clustering of large-scale SARS-CoV-2 mutation datasets.	Global SARS-CoV-2 mutation dataset, Coil 20, Facebook network, original MNIST and Jaccard distance based MNIST are used as the datasets with the technique. Global SARS-CoV-2 mutation dataset contains 203,344 features, 203,344 instances and 6 clusters. Coil 20 contains 1,440 grey images, 20 different objects each with an orientation of 72. Each image is 128 x 128 with a total dimensionality of 16384. Facebook network contains 22,470 nodes with a feature size of the same amount. Original MNIST contains a sample of 70,000, 28 x 28 grey scale images with a dimensionality of 784.

	Jaccard distance based MNIST is similar to the original MNIST. These datasets are of the type text and images with a dimensionality of 10^3, 10^4. And 10^5.

Table 2.4 provides an extensive evaluation of the performance of various *k*-hyperparameter tuning techniques, based on a comprehensive set of statistical and performance metrics. Each technique is assessed according to these metrics, which are reported to highlight their effectiveness and efficiency. It is important to note that the metric scores included in the table are those reported directly by the authors of the respective studies. This ensures that the performance evaluations reflect the original findings and methodologies used in each study. The table aims to offer a detailed comparison of how different techniques perform across various criteria, enabling a deeper understanding of their relative strengths and weaknesses in the context of *k*-hyperparameter tuning

Table 2.4: Metrics and Scores in the Evaluation of the existing *K*-hyperparameter Tuning Techniques in High Dimensional Space Clustering.

Reference	Name of the technique	Algorithm's performance & statistical metrics & scores
(Onumanyi et al., 2022)	AutoElbow.	Clustering accuracy (100%). This has been computed as a percentage of the number of clusters generated to the number of actual clusters (ground truth)
(Yan et al., 2020)	Adaptive Multi-view Subspace Clustering for High-dimensional Data.	Normal mutual information (98.31), accuracy (99.83) as well as the purity (98.83).

(Tao et al., 2020)	An Intelligent Clustering Algorithm for High-Dimensional Multiview Data in Big Data Applications.	The Jaccard Coefficient (JC), Rand Index (RI) and Folkes Russe (FS) are used to evaluate this algorithm. Mfeat dataset – RI = 0.9586, JC=0.6820 and FS = 0.8116 Internet Advertisement data set- RI = 0.8179, JC= 0.7868and FS =0.8809 Spambase data set- RI = 0.5225, JC=0.5222 and FS = 0.7225 Segmentation data set - RI = 0.2297, JC=0.8047 and FS = 0.3750 Cardiotocography - RI = 0.3984, JC=0.5576 and FS = 0.5854
(Xia et al., 2020)	Ball *k*-means.	Run-time was used to evaluate this technique. Four-class dataset- 0.03 Svmguide1 dataset - 0.24 Codrna dataset – 3.15 Kegg Network dataset – 12.21 Epileptic dataset -15.58 Birch3 dataset -1.18 Ijcnn dataset –9.28 RNA-seq dataset – 500
(Wang et al., 2019)	Fast Adaptive K-Means Subspace Clustering for High-Dimensional Data.	Accuracy and NMI are used to evaluate this technique. Glass dataset – Accuracy 49.53% of and NMI of 33.81% Breast dataset- Accuracy 95.57% of and NMI of 71.92% Vehicle dataset- Accuracy of 44.13% and NMI of 17.87% Umist dataset- Accuracy of 44.35% and NMI of 63.89% Yale dataset- Accuracy of 48.56% and NMI of 54.63% WebKB dataset- Accuracy 67.09% of and NMI of 16.72% TD2 dataset - Accuracy of 36.22% and NMI of 31.13%

(Lu, 2019)	Improved K-Means Clustering Algorithm for Big Data Mining under Hadoop Parallel Framework.	Clustering accuracy and running time are the metrics used to evaluate this technique. Clustering accuracy – 98.8% Running time – 22 seconds
(Xie et al., 2019)	Improving K-means clustering with enhanced Firefly Algorithms	Sum of intra-cluster distances, also called fitness scores, accuracy, sensitivity, specificity, and macro-average F-score are used as the performance metrics to evaluate the performance of this *k*-hyperparameter tuning technique Fitness score- Acute Lymphoblastic Leukemia (293.53), Sonar (160.54), Ozone (514.11), Wisconsin breast cancer diagnostic data set Wbc1 (2280.8), Wisconsin breast cancer original data set Wbc2 (1092.1), Wine (456.78), Iris (130.24), Balance (1002.9), Thyroid (113.26), E. coli (257.63), Drivface (4849.4), Micromass (656.91), sensor (426,26), Human Activity (12785), Skin Lesion (5399.8), Mice Protein (2345.0), and Libras (466.05). Accuracy – Acute Lymphoblastic Leukemia (0.5137), Sonar, Ozone, Wisconsin breast cancer diagnostic data set Wbc1 (0.9147), Wisconsin breast cancer original data set Wbc2 (0.9693), Wine (0.9485), Iris (0.8818), Balance (0.8047), Thyroid (0.8235), E. coli (0.7739), Drivface (0.7687), Micromass (0.8582), sensor (0.8118), Human Activity (0.6436), Skin Lesion (0.7854), Mice Protein (0.7238), and Libras (0.7801).

		Sensitivity – Acute Lymphoblastic Leukemia (0.5187), Sonar, Ozone, Wisconsin breast cancer diagnostic data set Wbc1 (0.9056), Wisconsin breast cancer original data set Wbc2 (0.9667), Wine, Iris (0.8227), Balance (0.8038), Thyroid (0.8676), E. coli (0.6609), Human Activity (0.6303), Skin Lesion (0.7898), Mice Protein (0.6913), and Libras (0.8342) Specificity – Acute Lymphoblastic Leukemia (0.5087), Sonar, Ozone, Wisconsin breast cancer diagnostic data set Wbc1 (0.8990), Wisconsin breast cancer original data set Wbc2 (0.9667), Wine (0.9618), Iris (0.9113), Balance (0.8056), Thyroid (0.8676), E. coli (0.8304), Drivface, Micromass, sensor, Human Activity (0.6568), Skin Lesion (0.7800), Mice Protein (0.6913), and Libras (0.8342) Macro-average F-score – Acute Lymphoblastic Leukemia (0.5145), Sonar, Ozone, Wisconsin breast cancer diagnostic data set Wbc1(0.9092), Wisconsin breast cancer original data set (Wbc2), Wine, Iris (0.9295), Balance (0.8045), Thyroid (0.7539), E. coli (0.6992). Wilcoxon rank sum test -Acute Lymphoblastic Leukemia (1.18E-04), Sonar (1.07E-08), Ozone (2.88E-11), Wisconsin breast cancer diagnostic data set WBc1 (5.01E-13), Wisconsin breast cancer original data set Wbc2(3.10E-10), Wine (3.49E-08), Iris (1.00E-00), Balance (2.89E-05), Thyroid (2.02E-06), E. coli (2.15E-02), Drivface (3.44E-03), Micromass(3.32E-04).
(Rustam et al., 2019)	Kernel Spherical K-Means and Support Vector Machine for Acute Sinusitis Classification	Clustering accuracy and running time are the two metrics used to evaluate this *k*-hyperparameter tuning technique. Clustering accuracy – 90% Running time – 0.03 seconds

(Hussain & Haris, 2019)	K-means based Co-clustering Algorithm for Sparse, High Dimensional Data.	Accuracy, NMI, Sum of squared error, running time and the popular statistical t-test are used to evaluate the effectiveness of this technique. Accuracy – M2 (0.92), M5 (0.95), M10 (0.73), Cora (0.48), Cornell (0.63), Washington (0.66). NMI - M2 (0.53), M5 (0.91), M10 (0.69), Cora (0.28), Cornell (0.43), Washington (0.48). Sum of squared error - M2 (385), M5 (368), M10 (370), Cora (1960), Cornell (78.5), Washington (93). Running time (seconds) - M2 (0.05), M5 (0.08), M10 (0.36), Cora (0.72), Cornell (0.01), Washington (0.04). T-test – The technique is statistically significant with a 0.05 significance level on M2, M5, M10, Cora , Cornell and Washington datasets.
(Rezaee et al., 2020)	GBK-means clustering algorithm: An improvement to the K-means algorithm based on the bargaining game.	F-measure, Dunn index, Rand index, Jaccard index, Normalized Mutual Information, normalized variation of information, measure of concordance and Wilcoxon signed rank test have been used to evaluate this algorithm. F-measure - Australian Credit (0.849), Breast Cancer (0.933), Breast Wisconsin (0.948), Diabetes (0.660), Haberman's Survival (0.518), Heart Disease (0.78), Hepatitis (0.711), Ionosphere (0.759), Japanese Credit (0.839), Mammographic (0.764) Dunn index - Australian Credit (0.059), Breast Cancer (0.076), Breast Wisconsin (0.134), Diabetes (0.100), Haberman's Survival (0.072), Heart Disease (0.2), Hepatitis (0.443), Ionosphere (0.085), Japanese Credit (0.381), Mammographic (0.25). Rand index - Australian Credit (0.867), Breast Cancer (0.893), Breast Wisconsin (0.890), Diabetes (0.322), Haberman's Survival (0.742), Heart Disease (0.779), Hepatitis

		(0.848), Ionosphere (0.641), Japanese Credit (0.864), Mammographic (0.757) Jaccard index - Australian Credit (0.863), Breast Cancer (0.996), Breast Wisconsin (1.000), Diabetes (0.974), Haberman's Survival (0.962), Heart Disease (0.510), Hepatitis (0.971), Ionosphere (0.672), Japanese Credit (0.970), Mammographic (0.809) Normalized Mutual Information - Australian Credit (0.42), Breast Cancer (0.670), Breast Wisconsin (0.756), Diabetes (0.481), Haberman's Survival (0.632), Heart Disease (0.24), Hepatitis (0.553), Ionosphere (0.097), Japanese Credit (0.45922), Mammographic (0.245) Normalized variation of information - Australian Credit (0.712), Breast Cancer (0.511), Breast Wisconsin (0.530), Diabetes (0.765), Haberman's Survival (0.633), Heart Disease (0.850), Hepatitis (0.594), Ionosphere (0.951), Japanese Credit (0.702), Mammographic (0.872) Measure of concordance - Australian Credit (1), Breast Cancer (1), Breast Wisconsin (1), Diabetes (1), Haberman's Survival (1), Heart Disease (1), Hepatitis (1), Ionosphere (1), Japanese Credit (1), Mammographic (1).
(Dey et al., 2020)	Lasso Weighted *k*-means.	Running time in seconds, clustering error rate, rand index and Normalized Mutual Information are used in the evaluation process. Running time in seconds - Brain (2.407632), Leukemia (1.008672), Lung cancer (1.542459), Lymphoma (1.542459), Wine (0.219742), Coil_5 (4.165497), ORL_5 (0.609416), YALE_5 (0.704548), ALLAML (1.008672), Appendicitis (2.421305), SuCancer (Missing), Iris (Missing), Glass

		(Missing), Tae (Missing), Zoo (Missing), Cleveland (Missing), Leaf (Missing), Vowel (Missing), Ecoli (Missing), Hebaerman (Missing) WDBC (0.510246) Clustering error rate - Brain (0.2254), Leukemia (0.0250), Lung cancer (0.2196), Lymphoma (0.0161), Wine (0.0549), Coil_5 (0.4031), ORL_5 (0.2800), YALE_5 (0.3455), ALLAML (0.2492), Appendicitis (0.1917), SuCancer (0.4770), Iris (Missing), Glass (Missing), Tae (Missing), Zoo (Missing), Cleveland (Missing), Leaf (Missing), Vowel (Missing), Ecoli (Missing), Hebaerman (Missing) WDBC (0.0748) Rand index - Brain (Missing), Leukemia (Missing), Lung cancer (Missing), Lymphoma (Missing), Wine (0.9339), Coil_5 (Missing), ORL_5 (Missing), YALE_5 (Missing), ALLAML (Missing), Appendicitis (Missing), SuCancer (Missing), Iris (0.9495), Glass (0.6983), Tae (0.6181), Zoo (0.8886), Cleveland (0.6677), Leaf (0.9477), Vowel (0.8598), Ecoli (0.7984), Hebaerman (0.6220) WDBC (Missing) Normalized Mutual Information – Brain (0.6263), Leukemia (0.8056), Lung cancer (0.3078), Lymphoma (0.9255), Wine (0.8267), Coil_5 (0.4092), ORL_5 (0.7610), YALE_5 (0.5828), ALLAML (0.4298), Appendicitis (0.2502), SuCancer (Missing), Iris (Missing), Glass (Missing), Tae (Missing), Zoo (Missing), Cleveland (Missing), Leaf (Missing), Vowel (Missing), Ecoli (Missing), Hebaerman () WDBC (0.6215).
(Brodinová et al., 2019)	Robust and sparse *k*-means clustering for high-dimensional data.	Synthetic dataset was used in this technique. The synthetic dataset consists of 40 observations, 50 features and 3 clusters.

(Orkphol & Yang, 2019)	Sentiment Analysis on Micro blogging with K-Means Clustering and Artificial Bee Colony.	Clustering error rate was used to evaluate this algorithm. CER – 0.191
(Song et al., 2021)	A Fast Hybrid Feature Selection Based on Correlation-Guided Clustering and Particle Swarm Tuning for High-Dimensional Data.	Clustering accuracy in % and run-time in seconds are the two metrics used to evaluate this algorithm. Arrhythmia- Clustering accuracy (67.68), Run time (31.723 seconds) SCADI- Clustering accuracy (89.68), Run time (3.314 seconds) GFE- Clustering accuracy (85.13), Run time (43.571 seconds) Prostate- Clustering accuracy (97.49), Run time (3.550 seconds) MFD- Clustering accuracy (99.40), Run time (94.522 seconds) Coil 20- Clustering accuracy (100), Run time (237.147 seconds) Yale_64- Clustering accuracy (79.52), Run time (35.169 seconds) Colon- Clustering accuracy (92.47), Run time (5.974 seconds) SRBCT- Clustering accuracy (100), Run time (8.716 seconds) WrapAR10P- Clustering accuracy (100), Run time (13.104 seconds) Leukemia_Small- Clustering accuracy (100), Run time (5.274 seconds) DBWorld- Clustering accuracy (97.57), Run time (4.571 seconds)

		DLBCL- Clustering accuracy (100), Run time (5.821 seconds) Drv_face- Clustering accuracy (98.23), Run time (72.400 seconds) Leukemia_Big- Clustering accuracy (100), Run time (6.119) CNS- Clustering accuracy (85.91), Run time (7.400 seconds) Lung - Clustering accuracy (98.01), Run time (26.316) Ovarian- Clustering accuracy (100), Run time (9.826 seconds).
(Dey et al., 2020)	The Sparse MinMax *k*-Means Algorithm for High-Dimensional Clustering.	Retained features, run-time and Dunn index have been used in the evaluation of this technique Retained features – Brain (1,810), breast cancer (79), colon cancer (76), leukemia (148), lung cancer 1 (16), lung cancer 2 (5), lymphoma (717), prostate cancer (5,650), SRBCT (1,019) and suCancer (1,370) Dunn index - Brain (0.647), breast cancer (0.197), colon cancer (0.435), leukemia (0.621), lung cancer 1 (0.245), lung cancer 2 (0.548), lymphoma (0.616), prostate cancer (0.393), SRBCT (0.544) and suCancer (0.505) Runtime - Brain (1.386 seconds), breast cancer (7.712 seconds), colon cancer (0.341 seconds), leukemia (0.784 seconds), lung cancer 1 (5.305 seconds), lung cancer 2 (5.137 seconds), lymphoma (1.857 seconds), prostate cancer (4.293 seconds), SRBCT (1.126 seconds) and suCancer (2.918 seconds).
(Hozumi et al., 2021)	UMAP-assisted K-means clustering of large-scale SARS-CoV-2 mutation datasets.	Clustering accuracy and run-time are the three evaluation metrics used with this *k*-hyperparameter tuning technique. Clustering accuracy - Coil20 (0.853), Facebook network (0.786), original MNIST (0.919), Jaccard distanced-based MNIST (0.960), Global SARS-CoV-2 mutation dataset (0.617).

	Run-time - Coil20 (500 seconds), Facebook network (22,000 seconds), original MNIST(8,000 seconds), Jaccard distanced-based MNIST (25,000 seconds), Global SARS-CoV-2 mutation dataset (45,000s)

2.8.2 Discussions on the Critical Review Analysis

The culmination of the critical review analysis on the *k*-hyperparameter tuning techniques in high-dimensional space clustering unveils multifaceted insights and trends that shed light on the limitations and advancements within this domain. This provides valuable insights and implications for future research directions besides assisting in highlighting the key patterns, themes and the key theories from the literature review. In the subsequent sections, a critical evaluation and summary on the existing literature in regards to the *k*-hyperparameter tuning techniques in high-dimensional space clustering is done. Firstly, the review is organized based on topics, themes and key theories. Next, key findings are summarized, highlighting the main points. The identification of the gaps follows as well as the conduction of a study that compares and contrasts the different *k*-hyperparameter tuning techniques in high-dimensional space clustering. A discussion on the establishment of a theoretical framework that underpins the existing literature is done, clearly highlighting the significance of this research study as well as its implications in informing the objectives for new research. This approach is in line with the works of Snyder (2019) on literature review as a research methodology.

2.8.2.1 Architectural components, strategies and key theories of the existing *k*-hyperparameter tuning techniques

It is evident that most of the *k*-hyperparameter tuning techniques in high-dimensional space clustering are hybrid as opposed to stand-alone. The performance of the hybrid looks more promising as compared to the stand-alone ones.

This is evident on the “Improved K-means clustering with enhanced Firefly Algorithms” proposed by Xie et al. (2019) that demonstrates good performance scores on the Leukemia dataset as compared to other stand alone techniques. Across all the *k*-hyperparameter tuning techniques, the clustering theory is evident in their *k*-hyperparameter tuning processes. The architectural components of these techniques include the initialization and representation of each cluster by the centroids, iteration optimization towards convergence, minimization of the within-cluster variance, early stoppage and the number of clusters among others. The identification of these elements is in line with the survey works on the *k*-means clustering algorithms done by Blömer et al. (2016). In line with this work, it is observed that the performance of these techniques mainly depend on the *k*-hyperparameter, the number of clusters. Early stoppage helps the *k*-hyperparameter tuning algorithms not to get stuck at the local optima, generating sub-optimal results (Bai et al., 2013). Early stopping impacts the exploration and exploitation trade-off in an algorithm's search strategy across the entire search space. Therefore, adopting the right and effective search strategies goes along way with a successful early stoppage (Bai et al., 2013). In relation to the early stoppage strategy, it is observed that the superiority of the firefly algorithm is ascribed to its strong capability of exploration and exploitation through its search strategies on the entire search space. The effectiveness of this strategy is demonstrated in the accuracy, sensitivity, specifity and the F1-scores. At the same time, the automated subdivision coupled with global exploration and intensified neighboring lowers the probability of trapping at the local optima. Based on this, the proposed *k*-hyperparameter tuning technique is based on an effective early stoppage strategy. The Wilcoxon rank sum statistical test result of higher than 0.05 for the firefly based algorithm with the IRIS dataset is a demonstration that the firefly algorithms are more effective in the *k*-hyperparameter tuning problems in higher dimensional space as opposed to the low dimensional space, compared to other models.

This demonstrated the efectiveness of the early stoppage. This observation is in line with the one made by Steinley (2008). In the "Kernel Spherical *k*-means and Support Vector based clustering method for acute sinusitis", it is noted that the tuning of the kernel parameters using grid search is not efficient when dealing with dataset of relatively high dimensionality. For this reason, it is proposed that future works in this domain should focus on the application of more efficient search methods and strategies that are able to handle high-dimensional spaces. On another hand, it is noted that the choice of the data dimensionality reduction method is linked to the local optima problem that the early stoppage addresses. For example, in the "UMAP assisted *k*-means" technique, the superior scores in the clustering accuracy and run time across a number of datasets are an evidence of the high efficiency and stability of the UMAP dimensionality reduction method as opposed to both the PCA and the t-SNE. This is ascribed to the fact that the UMAP possess the capability of preserving the global structure of a dataset including its between-points structure and distances, making it one of the best options in visualization and exploration of the entire search space. This observation is in line with the one made by Bao et al. (2023). The interplay between early stopping and dimensionality reduction method might pose challenges in finding the optimal balance between the model's performance, convergence, and the quality of reduced dimensions (Bao, et al., 2023). The "iteration optimization towards convergence" is another component that is identified during the review process on the *k*-hyperparameter tuning techniques. The iterative optimization process aims to find the centroids that minimize the within cluster sum of squares, leading to well-separated and compact clusters (Bishno & Todwal 2018). For example, in the robust and sparse *k*-means clustering for high dimensional data, having to repeat the process of applying weighting functions and updating the variable weights iteratively until stabilization is attained is computationally expensive for datasets with extremely high dimensionality.

The evaluation results on this algorithm demonstrats a limitation in determining the sparsity parameter *s* for contaminated datasets of high dimensionality. An automated method of determining the sparse parameter value *s* in such high dimensionality datasets is therefore deemed critical in building effective high-dimensional K-means models when using this technique. This observation is in line with the one made by Qi et al. (2016).

The number of clusters *k* is another component that is identified across the different techniques. For example, in the "*k*-means based co-clustering algorithm for sparse high dimensional data", its running time showed that it increased when the number of clusters are large as is the case with the M10 and Cora datasets. To solve this limitation, developing parallel systems can significantly improve on the running time on the datasets with large number of clusters.These parallel systems would share the processing workload and subsequently improve on the running times.

This observation is in line with the one made in the research works of Casas et al. (2015). At the same time, this algorithm performed relatively lower particularly when the dataset is not well separated. For example, with the M2 and M5 datasets, the accuracy values of this algorithm is relatively lower, with other techniques performing significantly much better as compared to this technique. Initialization is another component that is identified during the reciew process. Initialization in *k*-means refers to the process of selecting the initial positions for the cluster centroids before starting the iterative optimization procedure (Agarwal et al., 2015). The initial centroids play a crucial role in determining the final clustering results because *k*-means is sensitive to their placement (Agarwal et al., 2015). Different initializations can lead to different local optima, resulting in different clusterings of the data (Agarwal et al., 2015).

From the review on the experimental results of the reviewed techniques, those that adopted the *k*-Means++ initialization strategy performed relatively better as compared to those that applied the random initialization strategy. For this reason, it is concluded that it is important for future researchers focusing on this problem domain to ensure that the initialization strategies adopted are oriented to finding the initial centers, more strategically, as opposed to doing finding them randomly. This observation is in line with the one made by Hämäläinen et al. (2020). Development of such new technique is therefore based on the similar architectural components, strategies and the key theories as those of the existing techniques.

2.8.2.2 Diversity, similarities, efficacy and limitations of the *k*-hyperparameter tuning techniques in high dimensional spaces and the future opportunities

The surveyed literature reveals spectrum of *k*-hyperparameter tuning techniques in high dimensional space clustering, ranging from optimization-based algorithms to validation indices and dimensionality reduction strategies. Each technique exhibits distinct strengths and limitations, highlighting the diversity and adaptability of the techniques in high-dimensional spaces. Firstly, it is observed that the traditional elbow had a long standing literature and success in the *k*-hyperparameter tuning process. However, in some datasets, a clear elbow does not form, making it difficult to identify the *k*-hyperparameter from the unclear elbow. The use of the internal validation indexes like the Silhouette index in order to solve the smooth elbow limitations do not bear fruits as such indeces, at times, generates their best scores on sub-optimal number of clusters. Future research in this domain needs to use several internal validation indexes, and compare them, instead of just using a single internal validation index. Considering that the elbow technique is a benchmark, it is important to look into ways that it could be improved based on its limitations. Despite the diversity shared among the different *k*-hyperparameter tuning techniques, some similarities are observed.

For example, the central objective, iterative optimization, use of an objective function in the tuning process, unsupervised learning approach to clustering, and the convergence are some of the key similarities that are identified. The central objective of all the techniques aims at generating an optimal *k*-hyperparameter value that generated good quality clustering results. The iterative optimization process across all the techniques aims at repeatedly refining the cluster assignments and centroids in order to minimize the within-cluster sum of squared distances (WCSS) or variance using an objective function. Lastly, the convergence across all the reviewed techniques aims at ensuring that the centroids no longer change significantly between the consecutive iterations or when the maximum number of iterations is reached. This observation is in line with the works done by Ikotun et al. (2022). However, amidst these similarities, diversity is observed, on the other hand. Firstly, the initialization strategies are different across the techniques and this had an effect on the performance of the techniques. For example, the "Sparse Min-Max *k*-Means Algorithm for High-Dimensional Clustering" proposed by Dey et al. (2020) and the "Fast Hybrid Feature Selection Based on Correlation-Guided Clustering and Particle Swarm Tuning for High-Dimensional Data" proposed by Song et al. (2021) had different initialization methods.

The dimensionality reduction methods as well as the high-dimensional input datasets are also different. Morever, the performance and the evaluation metrics applied in the evaluation process across the different techniques are also different. The way that the different *k*-hyperparameter tuning techniques handle the membership degrees of the high-dimensional dataset points is also different. Each and every *k*-hyperparameter tuning technique has its own strengths and limitations as shown in Table 2.2. Some techniques performed better with the linearly separable data while others worked well with the non-linear dataset.

For example, the t-SNE based “Lasso Weighted *k*-means” proposed by Chakraborty and Das (2020) versus the PCA based Autoelbow proposed by Onumanyi et al. (2022). Based on this observation, it is proposed that future research should endeavor on leveraging on the technologies that have the capabilities of accommodating both the linearly separable as well as the non-linearly separable high-dimensional datasets.

This observation is in line with the proposal made by Gikera et al. (2023a) in regards to leveraging on the ensemble based technique of self-adapting autoencoder in the *k*-hyperparameter tuning process on a high-dimensional space, a technique that has the ability to accommodate both the linearly separable and the the non-linearly separable high dimensional spaces. Certain methodology, such as the heuristic algorithms and adaptive clustering approaches, demonstrates promising results in mitigating challenges posed by high dimensionality. However, the trade-offs between the computational overhead and the clustering accuracy remains prevalent across the techniques. From the review, it is evident that the existing *k*-hyperparameter tuning techniques, in high-dimensional space clustering, faced some limitations. One, while they have been successful in the *k*-hyperparameter tuning processes with some high-dimensional datasets, these techniques face customization challenges in trying to accommodate other different high-dimensional datasets. For example, in the AutoElbow technique, proposed by Onumanyi et al. (2022), the auto-elbow graphs do not depict a sharp elbow with some imbalanced high-dimensional datasets. It is therefore concluded that future research in this domain should focus on the techniques that allow flexibility in accommodating a variety of high-dimensional datasets in the *k*-hyperparameter tuning process. This observation is in line with the one made by Liu et al. (2021) on the use of multiple kernel *k*-means algorithm to handle multiple varieties of datasets.

In some other techniques, selecting the optimal *k*-hyperparameter value is a data-dependent process. For example, in the "Intelligent Clustering Algorithm for High-Dimensional Multiview Data in Big Data Applications", proposed by Tao et al. (2018), the high-dimensional datasets consisting of features with equal relevance performed better than the ones without. Such future techniques must therefore be data-independent. This observation is in line with the the works of Feldman et al. (2020). With some other techniques, for example the "Kernel Spherical K-Means and Support Vector Machine for Acute Sinusitis Classification", proposed by Rustam et al. (2019), getting stuck at the local optima as opposed to the global optima is another limitation. Therefore, it is concluded that the future *k*-hyperparameter tuning techniques must aim at effectively exploring the entire search space in order to avoid the local optima limitations that subsequently lead to the generation of sub-optimal cluster results. The early stopping and meta-learning, using an autoencoder, is proposed as a leverage point for improving the efficiency and effectiveness of the *k*-hyperparameter tuning process of the future techniques and help avoid algorithms from getting stuck at the local optima. This observation is in line with the one made by Xie et al. (2019), in a technique that used firefly technology to avoid the local optimum problems.

The nature of the high-dimensional datasets used as the input into the *k*-hyperparameter tuning techniques also posed challenges. This is because of the fact that there is usually no priori information about such high-dimensional datasets, with no labels. Sparse, noisy and redundant features present in these high-dimensional datasets made the *k*-hyperparameter tuning process an even more challenging task. For example, in the "Robust and sparse *k*-means clustering for high-dimensional data", proposed by Brodinová et al. (2019), using this technique to identify outliers from the noisy variables is challenging as the noisy variables are assigned a weight of zero.

Based on this observation, it is concluded that such future techniques should have mechanisms to learn the data and pre-process it appropriately before feeding it into the *k*-hyperparameter tuning technique. This observation is in line with the one made by Zhao and Liu (2020) in their works on an adaptive graph regularized low–rank matrix factorization with noise and outliers for clustering, a model for managing high-dimensional data that contain outliers.

In conclusion, the focus on the diversity, similarities, efficacy and limitations on the reviewed techniques gives birth to new research opportunities. The need for effective and adaptive state of the art techniques capable of tuning for the *k*-hyperparameter value correctly, in high dimensional spaces, is evident. Furthermore, the scalability for such techniques to handle large-scale high-dimensional datasets is also a critical area for future research focus. The discussions on the diversity, similarities, efficacy and limitations on the different techniques highlighted avenues for future research, including the development of hybrid techniques, improved automated *k*-selection methodologies, and more robust clustering frameworks tailored specifically to high-dimensional space clustering.

2.8.2.3 Trends on the dimensionality reduction methods in unsupervised clustering

In the review, it is evident that the PCA is the popularly applied dimensionality reduction method across the various *k*-hyperparameter tuning techniques. This observation is in line with the work done by Huang et al. (2019) on a systematic review of the data dimensionality reduction methods. Kernel PCA, which is an extension of the canonical PCA, based on a kernel function, has also been popularly used across the techniques. Most of the other techniques do not adopt similar data dimensionality reduction method.

Based on this observation, it is concluded that developing an empirical approach to comparative analysis of several dimensionality reduction methods on a single *k*-hyperparameter tuning techniques would be crucial in revealing important knowledge. The performance of these dimensionality reduction methods may be relative to the specific nature and variety of a high-dimensional dataset. This observation is in line with the "no-free-lunch" theorem in machine learning where an algorithm may perform well in one application area and not the other (Huang et al., 2019). Such comparative analysis results, in form of a data scientist tool box, would inform the researchers on the most suitable combination of the *k*-hyperparameter tuning technique and the dimensionality reduction method for a specific variety of a high-dimensional dataset. This observation is in line with the one made by Gikera et al. (2023a). The aim of such an adoption would be driven by the need to handle large and high-dimensional datasets more efficiently and effectively, extracting meaningful features from a high-dimensional dataset. In the Lasso Weighted K-means technique, the results show that the t-distributed Stochastic Neighbour and Embedding (t-SNE) and Principal Component Analysis generated different qualities of clusters with the Leukemia dataset.

It is therefore concluded as an important step to perform further experiments on a number of other high dimensional datasets and dimensionality reduction methods, with an aim of establishing the best set of data dimensionality reduction methods for a specific variety of high dimensional dataset. Although most of the techniques uses only one dimensionality reduction method, others like the "Fast Hybrid Feature Selection Based on Correlation-Guided Clustering and Particle Swarm Tuning for High-Dimensional Data" proposed by Song et al. (2021) utilized a hybrid approach to dimensionality reduction method, and achieved good results. In this technique, a clustering accuracy of 100% is achieved on the COIL 20 and SBRT datasets as well as a clustering accuracy of 99.40% on the MFD dataset.

It is therefore concluded that such future new *k*-hyperparameter tuning techniques should focus on the adoption of a hybrid data dimensionality reduction method that complement the strengths and weaknesses of the individual dimensionality reduction methods, with the aim of effective and efficient extraction of meaningful features from a high-dimensional space. Morever, with such a hybrid, discovering complex patterns and representations from a high-dimensional space would be easier. Lastly, the review unveils the intricate interplay between high-dimensionality data challenges and the selection of the *k*-hyperparameter value, elucidating the impact of dimensionality reduction methods on high-dimensional spaces in the *k*-hyperparameter tuning process and subsequently on the quality of clustering results. It delineates the strengths and limitations of the different *k*-hyperparameter tuning techniques in a variety of dimensionality reduction methods and high dimensional datasets. It is concluded that in the choice of the data dimensionality reduction method, the aim should be to choose the most suitable method for a particular variety of a high-dimensional dataset, one that effectively and efficiently represents it in a low dimensional space and with minimal information loss.

2.8.2.4 Nature of datasets and the impact of their dimensionality in clustering quality

The reviewed literature consistently emphasizes the profound impact of high dimensionality on the quality and efficiency of clustering algorithms. Across all the techniques, the curse of dimensionality is identified as the main cluster analysis challenge, making the *k*-hyperparameter tuning process a non-trivial exercise. High dimensional datasets are characterized by a multitude of features and dimensions and pose cluster analysis challenges during the *k*-hyperparameter tuning process. This is usually attributed to the fact that high-dimensional datasets consists of high number of features that are both sparse and redundant. In these datasets, the numbers of features are usually more than the number of instances.

Text and image based datasets are the most commonly used datasets across the different *k*-hyperparameter tuning techniques. It is concluded that such future empirical study in this domain should therefore focus on incorporating video and audio based high-dimensional datasets, besides the text and images. It is concluded that such inclusion could inform the data scientists on the most suitable *k*-hyperparameter tuning technique for a specific variety of a high-dimensional dataset. The dimensionality of these high dimensional datasets ranges between tens of number of features to hundreds of thousands of number of features. However, datasets with hundreds of features, thousands of features as well as tens of thousands of features are the most prominent across all the techniques. Across the different techniques, the datasets with higher dimensionality shows increased run times as compared to the datasets with lower dimensionality levels. For example, in the Ball *k*-means algorithm, the runtime for the RNA seq dataset, with a relatively higher dimensionality, is higher as compared to the Epileptic dataset. Based on this observation, it is concluded that in the development of future *k*-hyperparameter tuning technique, it is critical to leverage on effective data dimensionality methods for such high-dimensional datasets. This observation is in line with the one made by Al-Daoud (1996), that alludes to the fact that the number of object to centroid point computations increases proportionally with the increasing dimensionality of a high dimensional dataset. The "Adaptive Multi-view Subspace Clustering for High-dimensional Data" technique performed relatively fast when handling Jaffe and Yale datasets.

This is because of the fact that the technique iteratively updates each cluster centroids in the embedded space as opposed to performing this on the original high dimensional space. It is also noted that that it is more effective to perform the *k*-hyperparameter tuning process from a reduced feature space as opposed to performing it on the original high-dimensional space.

This is evident on the relatively faster speed demonstrated by the "Fast Adaptive *k*-Means Subspace Clustering for High-Dimensional Data" technique when handling dataset with extremely high dimensionality, i.e. TDT2 and WebKB, where the *k*-hyperparameter tuning process is performed on the reduced feature space as opposed to on the original feature space. Based on this observation, it is concluded that such new techniques, dealing with high-dimensional statistics, should focus on performing the *k*-hyperparameter tuning process from a reduced feature space as opposed to a high-dimensional space. This observation is in line with the research works of Hussain and Haris (2019). However, it is noted that too much reduction on the number of features degraded the quality of clustering when a few features on the original dataset are preserved. This is evident in the work of Song et al. (2021) where the use of the hybrid set of data dimensionality reduction methods aimed at achieving a balanced trade-off on the number of reduced features from the high-dimensional space and the quality of clusters. Across all the datasets used with the firefly algorithm, the fitness scores, accuracy, F1 score, sensitivity and specifity has a significant improvement when the numbers of features in the original dataset are reduced to a lower number, but with minimal information loss. If this reduction has significant information loss in the original dataset, then these scores degrades due to the degradation of the clustering results. For this reason, it is suggested that future *k*-hyperparameter tuning techniques should focus on a balanced trade-off of the number of features during the *k*-hyperparameter tuning process. This observation is in line with the work of Ayesha (2020). In the "Sparse MinMax *k*-means algorithm for high dimensional clustering", all the datasets used with this technique contained a number of features that are greater than the number of instances. The evaluation scores on this technique with these datasets indicate that this technique is a state of the art in the *k*-hyperparameter tuning in high-dimensional spaces.

However, it is concluded that mechanisms needed to be put in place to assist the techniques in managing the challenges of noisy variables and redundant features in a high dimensional dataset. Redundant features can lead to increased computational complexity and may negatively impact the performance of machine learning models. Invention of an algorithm that can learn a compact representation of the data, by effectively eliminating or reducing the influence of redundant features, is worth investigating. Using multiple kernels to handle such challenges could also offer solution into this problem.

This observation is in line with the one made by Sun et al. (2021). In the Fast Hybrid technique based on the Feature Selection on Correlation Guided Clustering and Particle Swarm, it is evident that the run times are proportional to the number of features in a dataset. For this reason, it is proposed that the adoption of an efficient and effective data dimensionality reduction method with this technique had a significant effect on its performance. This observation is also in line with the works of Jamal et al. (2018). In the GBK means algorithm, it is noted that the dimensionalities of all the ten datasets used in the experimentation process are of relatively lower dimensionality as opposed to the datasets used with the other *k*-hyperparameter tuning techniques. It is therefore concluded as an important step to also undertake an empirical study using datasets of higher dimensionality and analyze the results in order to have a clearer understanding of the performance of this technique. This observation is in line with the one made by Ayesha et al. (2020).

2.8.2.5 Performance evaluation metrics on the *k*-hyperparameter tuning techniques

The existing *k*-hyperparameter tuning techniques do not adopt similar performance evaluation metrics. The internal and external validation indexes, clustering accuracy and the run times are the commonly applied metrics.

It is however concluded that in unsupervised clustering like the one by *k*--means, the internal validation indexes, as opposed to external validation indexes, are the best metrics for use in assessing the quality of clusters with unknown number of clusters. This observation is in line with the one made by Clarke (2019). External validation indexes are based on the previous knowledge about a dataset while the internal validation indexes are based on the information intrinsic to the data alone (Rendon et al., 2011).

Internal validation indexes assess cluster quality without requiring knowledge of true class labels, making them more versatile for unsupervised learning scenarios where true labels are unavailable. Among the internal validation indexes, Silhouette Index, Davies Bouldin index, Calinski Harabsz index as well as the Dunn Index are the most commonly applied metrics for evaluating the quality of clusters. This relates to the research done by Gikera et al. (2023a) that explains why the four are the most common internal validation metrics used in evaluation of clusters. This is also in line with the survey results of Deborah et al. (2010). Although these metrics successfully evaluated the performance of these techniques, it is concluded that the adoption of multiple metrics as opposed to a single metric only is a more effective approach. This is because of the fact that the Calinski Harabsz index, Silhouette index, Dunn index and Davies Bouldin index poses some limitations at the individual levels.

These limitations mainly include: Sensitivity to cluster density, dependency on cluster size, lack of ground truth labels, interpretation limitations and sensitivity to data scaling and data dimensionality (Gikera et al., 2023a). For example, the Calinski Harabsz index tends to favor clusters with similar densities (Gikera et al., 2023a). If the clusters have significantly different densities, the index may not accurately capture the clustering quality (Gikera et al., 2023a).

It may give higher scores to clusters that are denser, even if they are not necessarily well-separated. The index is sensitive to the number of clusters and the size of the dataset. It tends to favor solutions with a larger number of clusters, which may lead to overfitting or over-segmentation of the data (Gikera et al., 2023a). This can result in inflated index values, making it challenging to determine the optimal number of clusters (Gikera et al., 2023a). These indexes do not rely on ground truth labels. Instead, they assess the clustering quality based on the internal structure of the data, without considering the true underlying classes (Gikera et al., 2023a). Therefore, their effectiveness may be limited when compared to metrics that utilize ground truth information. Similar to the Dunn index, interpreting the absolute value of the Calinski-Harabasz index can be difficult as it lacks a clear threshold or guideline to determine what constitutes a good or bad clustering result. It is often used comparatively, comparing different clustering solutions or tuning the number of clusters to find an optimal solution (Gikera et al., 2023a). The Calinski-Harabasz index can be sensitive to the scaling of the data and the number of dimensions. Inconsistent scaling or high-dimensional data can impact the distances used in the calculation and lead to biased results (Gikera et al., 2023a). Proper data preprocessing and dimensionality reduction techniques may be necessary to address these issues. It is concluded as an important aspect to consider these limitations when using each of these internal validation indexes and complement them with other indexes in order to gain a more comprehensive understanding of the clustering quality.

For example, in the Sentiment Analysis on Micro blogging technique with *k*-means and Artificial Bee Colony, the use of Silhouette index only in the determination of the *k*-hyperparameter demonstrated limitations. This is because, inconsistencies may occur where the Silhouette index generates the best score at a different optimal *k*-value than the one generated by a different internal validation index like Dunn index or Davies Bouldin index.

In such a case, it is recommended that an ensemble internal validation index, via boasting or bagging, whose components exercise equal sensitivity to the varied conditions within a high dimensional dataset, is used. This is in line with the observation made by Rodríguez et al. (2018).

The optimal *k*-hyperparameter value generated through such an ensemble would be achieved through the voting scheme i.e. the one that is returned by most of the internal validation indexes. Chi-square and T-test are on the other hand the most commonly used statistical tests for hypothesis significance testing in the existing *k*-hyperparameter tuning techniques. However, it is noted that some limitations come up when using the Chi square and T-test metrics to assess the quality of clusters. These include: interpretation limitations, sensitivity to sample size and cluster imbalance and limited consideration of cluster structure (Gikera et al., 2023a). These tests assess the independence between variables or groups (Gikera et al., 2023a). While statistical significance can be determined, interpreting the practical significance and meaningfulness of the results in the context of clustering quality can be challenging (Gikera et al., 2023a). The tests provide insights into the presence or absence of associations between variables but do not provide direct information about the quality of clustering (Gikera et al., 2023a).

These tests can be influenced by the sample size and the distribution of data across clusters (Gikera et al., 2023a). Small sample sizes or imbalanced data may affect the test's power and potentially bias the assessment of clustering quality (Gikera et al., 2023a). Moreover, unequal cluster sizes can impact the statistical significance of the the test results (Gikera et al., 2023a). These tests focus on measuring associations between variables but do not explicitly consider the structural characteristics of clusters (Gikera et al., 2023a).

They do not account for factors such as compactness, separation, or cluster shapes (Gikera et al., 2023a). Therefore, it is concluded that relying solely on the Chi-square or T-test may not provide a comprehensive evaluation of clustering quality in terms of these important clusters attributes, besides the limitation of the fact that the two tests do not test interactions between variables.

In future empirical studies on the evaluation of the state-of-the-art *k*-hyperparameter tuning techniques, it is important to adopt a standard set of more reliable performance and statistical metrics (Gikera et al., 2023a). An option of ANOVA would be the most appropriate as it has the capability of dealing with multiple groups, factors, and interactions between variables, making it a more comprehensive and efficient tool of statistical analyses compared to T-tests and Chi-square tests. This observation is in line with the one made by Greenland et al. (2016). Lastly, it is a concern that the computations of someevaluation metrics in the table of results are missing. For example, the Normal Mutual Information for Cleveland dataset and SuCancer datasets are missing and had been on the contrary proposed in the methodology. It is concluded an important practice to include all the evaluation metrics proposed in the methodology had their results computed in the table of results so that the discussion and conclusion process is done fairly. This observation is in line with the one made by Mehrabi et al. (2021), in regards to ensuring fairness in the evaluation process of machine learning algorithms.

2.8.2.6 Key findings from the literature review analysis

The literature review chapter extensively investigates the *k*-hyperparameter tuning techniques in high-dimensional space clustering. The comprehensive review encompasses a diverse spectrum of these techniques, assessing their applicability, strengths, limitations, and relevance in the domain of *k*-hyperparameter tuning in high-dimensional space clustering.

This review is not limited to a rigorous examination of the dimensionality reduction methods that play a pivotal role in the *k*-hyperparameter tuning process by dealing with the cluster analysis challenges inherent in high-dimensional spaces. The critical analysis and comparative assessment on these techniques, data dimensionality reduction methods as well as the evaluation metrics provides valuable insights into a set of trends and key findings. Table 2.5, presents the key findings from the analysis of the literature and provided insights into how these findings provided guidance on both the experimental research design methodology and the system development methodology in this research study, based on a conceptual framework.

Table 2.5: Key Findings from the Literature Review Analysis and how they guided the Methodology

Key findings from the analysis on the literature review	How the key finding guided the methodology
The *k*-hyperparameter tuning is based on the clustering theory, with a wide application and significance in a variety of domains. The *k*-value is one of the most important hyperparameters in *k*-means, with a great level of significance on the clustering results.	The performance evaluation metrics of the experimental *k*-hyperparameter tuning techniques focused on how correctly they tuned the *k*-hyperparameter value in a variety of high dimensional datasets in different domains. Correct *k*-values lead to good quality clustering.
The *k*-hyperparameter tuning techniques are variations of the classical *k*-means algorithm, adapted and scaled to tune the *k*-hyperparameter value in high dimensional datasets.	The conceptual model used in the empirical analysis is based on the variables present in the classical *k*-means adapted to high dimensional spaces i.e. *k*-hyperparameter tuning techniques, feature extraction methods, data types and the dimensionality levels of the high dimensional datasets as well as internal validation indexes.

Although the elbow method showed a long standing literature and success in tuning for the *k*-value, the generation of smooth unclear elbow is identified as one of the main limitation.	The proposed *k*-hyperparameter tuning technique focused on a strategy of solving the smooth elbow limitation associated with the elbow methods.
The challenges of tuning the *k*-hyperparameter value in high dimensional spaces are due to the inherent nature of the high dimensional datasets, mainly the curse of dimensionality. Others included data sparsity, noise and outliers as well as the irrelevant features. The nature is characterized by the data type and dimensionality levels.	The proposed *k*-hyperparameter tuning technique focused on a strategy of solving the limitations due to the inherent nature of the high dimensional datasets. The insights on the leverage points to accomplish this on the new technique are to be picked from the analysis results of the experimental data from the empirical analysis.
The existing *k*-hyperparameter tuning techniques are diverse, with each technique exhibiting distinct set of efficacy and limitations. The performance metrics claimed some techniques better than others.	The empirical approach to the comparative analysis is designed in a manner to validate or challenge the theoretical expectations from the theoretical analysis.
The *k*-hyperparameter tuning techniques in high-dimensional spaces still face performance gaps. This is due to the cluster analysis challenges inherent in the high-dimensional space datasets.	The development of a conceptual framework used in the empirical analysis aimed at identifying invaluable insights and leverage points that guided the development of an improved *k*-hyperparameter tuning technique.
Key findings revealed a nuanced interplay between the dimensionality reduction method, *k*-hyperparameter tuning technique and the clustering quality. Notably, certain techniques exhibited promising results in reduced dimensions, effectively mitigating challenges associated with high dimensionality.	The empirical approach to the comparative analysis is designed in a manner that it incorporated state-of-the-art dimensionality reduction methods in a variety of high dimensional datasets, including their variations and specialized kernel functions. This is done in order to identify the most effective methods in a specific variety of high dimensional dataset.

PCA is the most commonly applied data dimensionality reduction method across the various techniques.	
Although different metrics have been used to evaluate the performance of the *k*-hyperparameter tuning techniques in the liteareture, it is envinved that the internal validation indexes are the most effective. Internal validation indexes assess cluster quality without requiring knowledge of true class labels, making them more versatile for unsupervised learning scenarios where true labels are unavailable e.g. in *k*-means. Moreover, using a combination of these indexes gives a more comprehensive analysis of the clustering quality through tradeoffs and comparative analysis as compared to the use of individual internal validation index.	The experimental methodology is designed in such a way that the performance of the different *k*-hyperparameter tuning techniques is measured based on a combination of metrics i.e. Silhoutte index, Calinski Harabsz index, Dunn index and Davies Bouldin index. Run time is also used as an extra metric besides the internal indexes.
For each and every *k*-hyperparameter tuning technique, different high-dimensional datasets, dimensionality reduction methods and evaluation metrics are used.	In order to ensure fairness in evaluation, similar high-dimensional datasets, dimensionality reduction methods and evaluation metrics are applied to the *k*-hyperparameter tuning techniques used in the empirical analysis.
Text and images datasets are the commonly used datasets with the existing *k*-hyperparameter tuning techniques. These datasets possess different dimensionality levels.	In the design of the experimental methodology, audio and video datasets are also included in the empirical analysis besides the popular text and the image datasets. Data types of varied dimensionality levels are also adopted in this empirical study.

Several *k*-hyperparameter tuning techniques faced the local optima problem.	The experimental methodology is designed in a way that the different techniques are run for multiple times so as to increase the chances of escaping the local optima problems.
In the review of the dimensionality reduction methods, it is noted that the utoencoders have the ability to learn meaningful features in the embedded latent space, while at the same time train for specific tasks, leading to more efficient and effective representation.	During the empirical analysis in this research study, the autoencoder is adapted for effective representation on the embedded latent space, based on the specific variety of the high dimensional input dataset i.e. text, audio, video and image.
Although the literature analysis revealed that the *k*—means ++ initialization method demonstrated a more superior centroid determination strategy as compared to other methods like the random initialization, the adoption of a scalable and adaptable *k*-means ++ method is more strategic.	In the design of experiments during the empirical process, the *k*-means ++ initialization method is adapted and scaled appropriately in line with the specific nature of the high dimensional input dataset.
The Early stopping strategy, for example in the firefly algorithms, had an impact on the exploration and exploitation trade-off in the algorithm's search strategy across the entire search space. This subsequently lowered the probability of trapping at the local optima.	The proposed development of the new *k*-hyperparameter tuning technique is based on an effective early stoppage strategy besides the improvement in other areas as guided by the invaluable insights from the experimental results of the empirical analysis.
There is an interplay between the dimensionality reduction problem and the local optima challenge that the early stoppage addresses in the *k*-hyperparameter tuning techniques.	The early stopping and meta-learning strategy, in the proposed technique, is proposed as one of the leverage points for improving the effectiveness of the *k*-tuning process. This helps avoid getting stuck at the local optima.

Some techniques performed better with the linearly separable data while others worked well with the non-linear datasets.	The development of the new *k*-hyperparameter tuning technique endeavoured to leverage on strategies that have the capabilities of accommodating both the linearly separable and the non-linearly separable datasets.
Although most of the techniques used only one dimensionality reduction method, others utilized a hybrid approach to dimensionality reduction method, and achieved better results.	The development of the new *k*-hyperparameter tuning technique endeavoured on the adoption of an effective dimensionality reduction method on the high dimensional input datasets. Its performance is complemented by the invaluable insights identified from the analysis results of the experimental data from the empirical analysis.

The key findings elucidate the evolving research landscape of *k*-hyperparameter tuning in high-dimensional space clustering, highlighting recent advancements such as hybrid algorithms that integrate manifold learning, regularization strategies, and metaheuristic optimization. These developments aim to enhance the robustness and accuracy of *k*-selection in complex high-dimensional spaces. By summarizing existing research and justifying the research design, these findings provide a clear direction for the subsequent empirical analysis in Chapter Three. The integration of literature findings into the methodology not only advances knowledge within this field but also contributes to the progression of advanced research methodologies.

2.9 Conceptual Framework, Variables and Operationalization of Variables

The conceptual framework is defined in terms of definition of conceptual elements, relationship between dependent and independent variables as well as the operationalization of the variables (Kaur, 2013).

Conceptual framework is aimed at helping the researcher to formulate important concepts and develop a theory to support the development of the proposed *k*-hyperparameter tuning technique for high dimensional datasets (Kaur, 2013). In this research study, the conceptual elements, independent variables, moderating variables and dependent variables are underpinned on the analysis of the literature review and the key findings. The conceptual elements are identified as follows, in the subsequent sections.

2.9.1 Conceptual Elements

Understanding the conceptual elements helped in understanding the mechanics of the existing *k*-hyperparameter tuning techniques and their impact on clustering in high dimensions. Moreover, it provides insights into the techniques' strengths and limitations, laying a strong foundation upon which new such techniques are proposed.

2.9.1.1 Independent Variables

The *k*-hyperparameter tuning techniques are identified as the independent variables. Three promising *k*-hyperparameter tuning techniques are studied in both the pilot study and the empirical analysis in the main experiments. These techniques includes: AutoElbow (Onumanyi et al., 2022), Fast Hybrid Feature Selection Based on Correlation-Guided Clustering and Particle Swarm Tuning for High-Dimensional Data (Song et al., 2021) and the Adaptive Multi-view Subspace Clustering for High-dimensional Data (Yan et al., 2020). In this conceptual framework, the three best performing *k*-hyperparameter tuning techniques are used to construct the independent variables. The new technique is only used in the validation process in order to evaluate its level of effectiveness against the existing best performing techniques, in a variety of high-dimensional datasets.

2.9.1.2 Moderating Variables

Moderating variables are variables that act between the independent variables and the dependent variables and change the strength of their relationships (Kaur, 2013). They provide a potential explanation or mechanism for how the independent variable influences the dependent variable (Kaur, 2013). In other words, moderating variables mediate or act as an intermediate step in the causal pathway between the predictor and the outcome (Kaur, 2013). In this research study, both the nature of the high-dimensional datasets as well as the different dimensionality reduction methods are adopted as the moderating variables. The nature of the high dimensional datasets is defined by both the data type and the dimensionality levels.

2.9.1.3 Dependent Variables

The dependent variables in this research are derived from the most effective performance scores of the different *k*-hyperparameter tuning techniques based on the literature review analysis. These metrics included the Silhouette index, Calinski Harabsz index, Dunn index, Davies Bouldin index as well as the cluster building times. The first four are generalized as the internal validation indexes for measuring the quality of clustering results while the cluster building time is the duration taken by the algorithm to generate clusters after a high-dimensional dataset input. The five metrics are identified as the dependent variables. During the validation process on the new technique, the four internal validation indexes are aggregated into a single index, the ensemble validation index, and used to evaluate on the effectiveness of the new technique against the existing best performing ones in a variety of high-dimensional datasets. Figure 2.4 below illustrates the relationship between the independent variables, moderating variables and the dependent variables in this research's conceptual model.

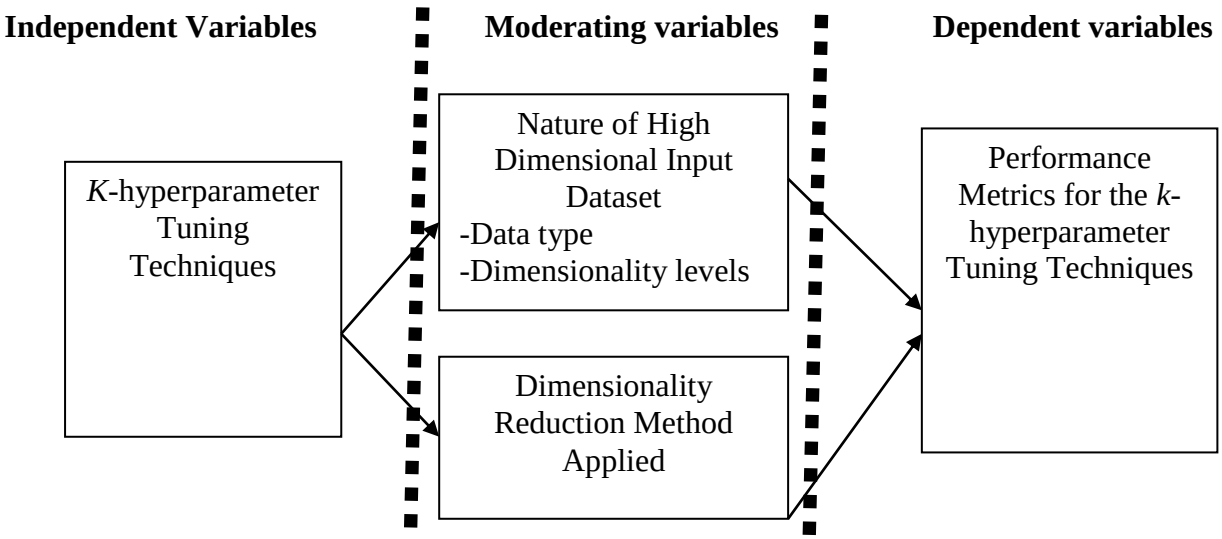


Figure 2.4: Illustration of the Relationship between Independent, Moderating and Dependent Variables

2.9.2 Operationalization of the Variables

Operationalization of variables is the process of defining and translating abstract concepts or constructs into measurable and observable variables or indicators (Herek, 2013). In research and data collection, operationalization is crucial as it allows researchers to quantify and gather empirical data to test the hypotheses and draw meaningful conclusions (Herek, 2013). Through the operationalization of variables, researchers are able to make the concepts of their study measurable, making them to put together an empirical data that can be analyzed and interpreted with ease (Herek, 2013). Clear operational definitions and valid measurement instruments are essential for ensuring the accuracy and rigor of the research findings. In light of this, this research uses the following steps in the operationalization of variables: identification of the constructs, operational definitions, selection of the measurement scales, development of the measurement instruments, pilot testing, data collections and the data analysis. This approach is in line with the one proposed by Herek (2013). The constructs identified in this research includes the specific scores of the evaluation metrics from the different *k*-hyperparameter tuning techniques in high-dimensional space clustering.

The independent and the moderating variables both used the nominal measurement scale while the dependent variables used the ratio measurement scales. Experiments are used to collect data on the operationalized variables. The initial experiments are performed in pilot testing in order to demonstrate feasibility for conducting this research study. In order to assess relationships between variables, test hypotheses, and draw conclusions based on the operationalized variables, data analysis is done using the R-based JAMOVI software interface and appropriate statistical techniques. In analyzing the *k*-hyperparameter tuning techniques across high-dimensional datasets, JAMOVI's R-based software is used for 2-way and 3-way ANOVA in order to evaluate how multiple factors such as tuning technique, dimensionality reduction method, dataset's dimensionality level and type affect clustering performance. This software enables researchers to statistically validate the significance of observed performance differences, assess interactions among these variables, and visualize results effectively via the whisker plots. By leveraging on the R's analytical power, the study offers empirical insights for improved *k*-hyperparameter tuning in high-dimensional spaces.

2.10 Summary.

The comprehensive analysis on the literature review focuses on a critical examinination of the *k*-means based techniques used to tune for the *k*-hyperparameter value in high-dimensional spaces. It also delves into a critical examination of the hyperparameters in *k*-means, cluster analysis challenges in tuning for the *k*-hyperparameter in high dimensional spaces, dimensionality reduction methods for unsupervised clustering as well as the metrics used in the performance evaluation of these techniques. The discussion of the results from the literature review analysis, identification of the key findings that guides the research methodology, formulation of the conceptual framework as well as the summary culminates the literature review analysis process.

The review underscores the trade-offs and limitations associated with the *k*-hyperparameter tuning techniques, emphasizing their efficacy and applicability in specific contexts. Although the various *k*-hyperparameter tuning techniques are rooted in their own strengths and limitations, the theoretical analysis and the performance scores demonstrates that four techniques are promising. It is noted that the *k*-hyperparameter value is one of the most important hyperparameters among the hyperparameters in *k*-means. Although the K-means++ initialization method demonstrates theoretical effectiveness in choosing the initial centroids more strategically, it is noted that the adoption of a scalable and adaptable K-means ++ initialization methods in a variety of the high dimensional datasets is a more strategic approach. This is based on the need to efficiently capture the data-specific features and scale them appropriately. The curse of dimensionality is identified as the main cluster analysis challenge in tuning for the *k*-hyperparameter value in high dimensional spaces. However, data sparsity and feature redundancy are also identified. Devising an effective strategy for handling the curse of dimensionality challenge is therefore a leverage point in the development of a new state of the art *k*-hyperparameter tuning technique.

The review on the dimensionality reduction methods used with the *k*-hyperparameter tuning techniques revealed that the PCA is the most popularly applied across most of the techniques. This review also demonstrates a highly competitive set of variations and kernel functions for handling specific variety of datasets. It is however noted as an important step to conduct a comprehensive comparative analysis in order to understand the trade-offs, capabilities, and applicability of these different variations, guiding researchers and practitioners in choosing the most appropriate and competitive dimensionality reduction method based on the specific variety of dataset at hand.

The empirical approach to this comparative analysis also aims at helping in validating or challenging the theoretical expectations of the high popularity of the PCA based on the theoretical analysis. In the review on the performance evaluation metrics for the different *k*-hyperparameter tuning techniques, it is examined that the internal validation indexes assesses the cluster quality without requiring knowledge of the true class labels. This makes them more versatile for unsupervised learning scenarios, e.g. in *k*-means clustering problems, where the true labels are unavailable. Moreover, using a combination of the internal validation indexes is crucial in order to have a more comprehensive understanding of the the quality of the clustering results, through trade offs and comparative analysis.

In conclusion, this review not only provides a comprehensive panorama of an in-depth literature on the *k*-hyperparameter tuning in high dimensional spaces, but also identified gaps, key findings, limitations, conceptual framework as well as the opportunities for further research in this domain. The conceptual framework acts as a roadmap, guiding the methodology in this research, integrating knowledge from the literature to address key research objectives in *k*-hyperparameter tuning. This critical synthesis, therefore, serves as the foundation for the subsequent empirical investigations in the subsequent chapters in this research thesis.

CHAPTER THREE: RESEARCH METHODOLOGY

3.1 Introduction

This chapter discusses the methodology that is applied in this research work. Mixed research method is adopted. The desktop research design is used to conduct and analyze the literature review, in a qualitative manner. On the other hand, the empirical research design is used to conduct and analyze the experiments in a quantitative manner. The first research objectives in this study guides both the desktop research and the experimental research design processes while the first and the third research objectives, as well as the hypothesis, guides the experimental research design process. The design science research design guided the development and implementation of the new technique in the second objective. This study is based on the positivism research philosophy, using mixed research methods for validation triangulation. In the research design, the overall strategy that guides the execution of this study is described. The experimental research design is composed of both the empirical analysis process and the validation experiments. Empirical analysis process focuses on conducting experiments to investigate on the end to end behaviour of the existing *k*-hyperparameter tuning techniques in a variety of high dimensional datasets and dimensionality reduction methods. Validation experiments, on the other hand, aims at evaluating the performance of the newly developed *k*-hyperparameter tuning technique against the existing ones in high dimensional space clustering. All experiments are conducted in the cloud-based Google Colab's Python Integrated Development Environment (IDE). The analysis results on the experimental data during the empirical analysis informed the design process of the newly developed technique, through a multi-methodological model development methodology. In this methodology, a conceptual design of the proposed *k*-hyperparameter tuning technique is discussed as well as its architectural design. After this, the process of developing and testing the technique is discussed.

In the last section, validation experiments are performed with an aim of evaluating the effectiveness of the newly developed *k*-hyperparameter tuning technique against the existing ones in high-dimensional space clustering. This is based on a set of evaluation metrics and the cluster visualizations in a variety of high-dimensional datasets. This chapter is organized as follows: research philosophy, research design, empirical analysis, research hypothesis, initial empirical study, experiments, integrated development environment, research instruments, sampling strategy, experimental data collection and techniques, analysis techniques on the experimental data, model development, and lastly, research ethics.

3.2 Research Philosophy

This work is based on the positivism research philosophy, using mixed research methods for validation triangulation. In this research, experiments on the performance of the existing best performing *k*-hyperparameter tuning techniques, using a similar set of standard high-dimensional benchmark datasets and dimensionality reduction methods are performed using an experimental check list. This approach confines to the positivism research philosophy where knowledge of a specific phenomenon is based to what can be observed, measured and recorded, in the same way as in natural science (Benbasat & Zmud, 1999). The measurements on the performance of these techniques, using a standard set of evaluation metrics, are recorded in a table of results before being analyzed using the R software. The choice of the Jamovi R based software is due to the fact that it has an intuitive, point-and-click interface that simplifies performing ANOVA and other statistical tests, making it less time consuming as opposed to Python that requires extensive coding which can be time-consuming for complex analyses such as the ones in this research study. The results of the analysis on the experimental data from the main experiments of the empirical analysis are used to guide the development of the new *k*-hyperparameter tuning technique.

Based on this research philosophy, the discussions and conclusions on the performance of both the newly developed *k*-hyperparameter tuning technique and the existing ones are done based on the ground truth.

3.3 Research Design

This research adopts the mixed methods research design, using both the qualitative and quantitative data as prosposed by Creswell and Poth (2016). The qualitative research design focuses on exploring and understanding the depth and nuances of the *k*-hyperprameter tuning space in high dimensions. It entails analyzing literature review on the existing *k*-hyperparameter tuning techniques in high dimensional spaces, alongside different dimensionality reduction methods used with these techniques. During the review, the five step methodology, as proposed by Khan et al. (2003), is adopted as explained in Section 2.8. On the other hand, the quantitative research design focuses on systematically collecting and analyzing numerical data to answer research questions, test hypotheses, and make statistical inferences in this methodology. It entails the collection and analysis of the experimental data from the main experiments through an experimental design methodology. Using the design science approach that incorporates the quantitative and the qualitative approach, this research aims at solving the problem of the *k*-hyperparameter tuning in high-dimensional space clustering through the design of a new improved technique. Since the research design also entails a strategy on how the research objectives are achieved, a mapping process of the research objectives and questions to the research methods is done. In the validation process of the new technique, both the quamtitative methods and qualitative methods are used. The 1-way ANOVA, H-statistic and the whisker plots are employed as the quantitative methods in the validation process while the cluster visualizations are employed as the qualitative methods. The degree of alignment between these two methods was assessed.

The mapping process involves identifying the appropriate research methods to help in achieving the research objectives in this study. This process ensures that the research process is both effective and efficient. The chosen methods are well-suited to address the research objectives while also taking into account both the practical constraints and ethical considerations within the *k*-hyperparameter tuning research domain. This alignment ensures that the research study is focused, rigorous, and capable of generating valid and valuable insights. Figure 3.1 shows how the research objectives and questions in this research study are mapped to the respective research methods as proposed by Järvinen (2008).

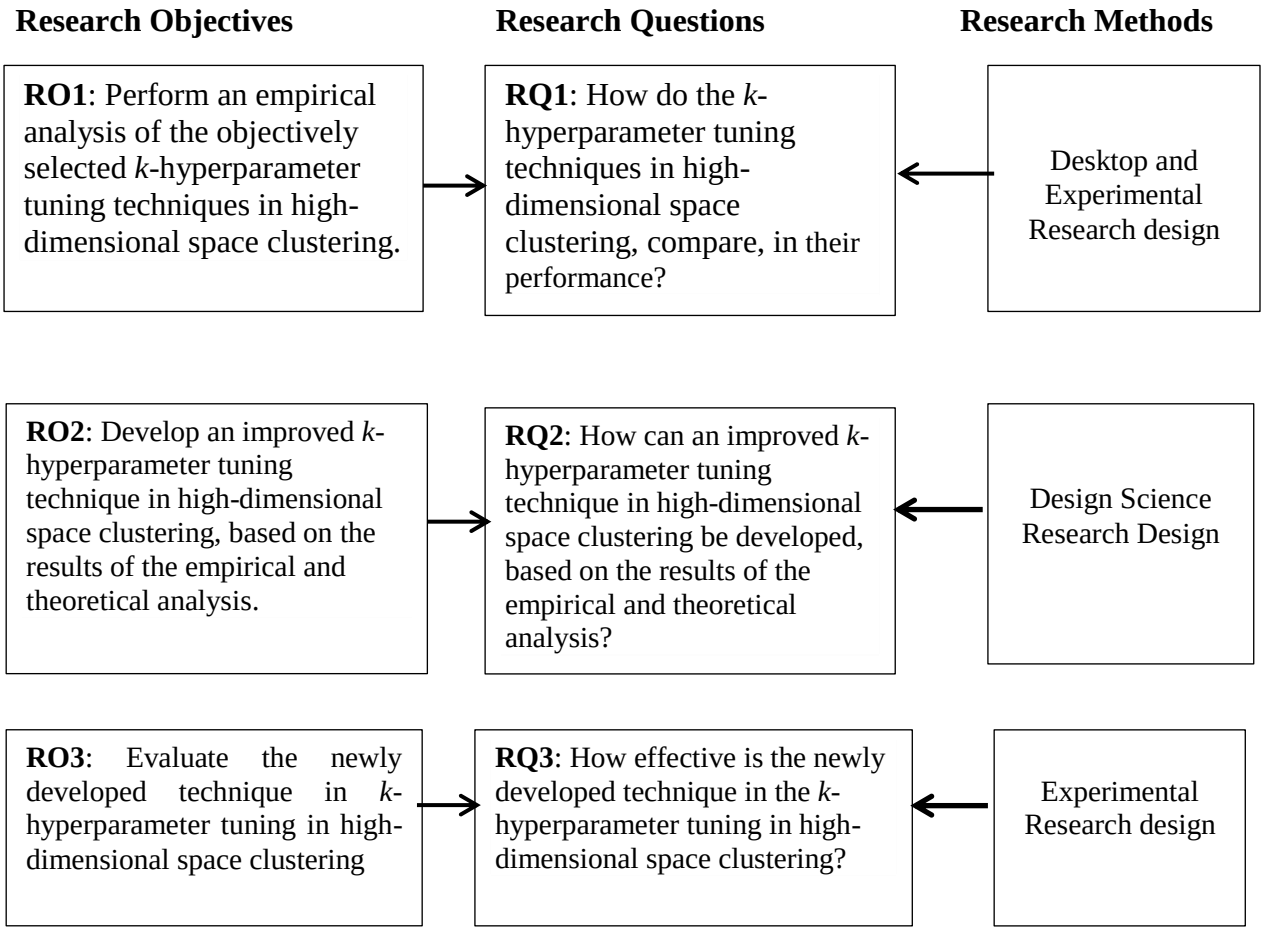


Figure 3.1: Mapping of research objectives and questions to research methods

3.4 Experimental Design for the Empirical Analysis

The experimental research design for the empirical analysis process in this research study focused on planning and conducting experiments, with an aim of investigating the end to end behaviour, and the interplay, of the existing best performing *k*-hyperparameter tuning techniques in a variety of high dimensional datasets and dimensionality reduction methods. In this empirical analysis, the existing best performing techniques used in the main experiments were identified through both the theoretical analysis process and the pilot study. Based on the experimental design, research hypothesis was first formulated in line with the research questions, before conducting the pilot study and the experiments. The Integrated Development Environment (IDE) for conducting the experiments was described. The sampling strategy used in the selection of the input high dimensional datasets, pre-training datasets, *k*-hyperparameter tuning techniques and the dimensionality reduction methods used in the various experiments was also described. Lastly, the experimental data collection process and techniques were discussed, including the data analysis techniques for the experimental data.

3.4.1 Research Hypothesis

In this research, the hypothesis is formulated as follows:

H_0: "There are no statistically significant differences between the performances of the *k*-hyperparameter tuning techniques in high-dimensional space clustering, in a variety of datasets and data dimensionality reduction methods."

H_a: "There are statistically significant differences between the performances of the *k*-hyperparameter tuning techniques in high-dimensional space clustering, in a variety of datasets and data dimensionality reduction methods."

In light of this research hypothesis, the experiments in this research study aims at investigating on the following:

1. If there are statistically significant differences in the performance of the *k*-hyperparameter tuning techniques in:
    a) Datasets of varied dimensionality
    b) Datasets of varied data types
    c) Different data dimensionality reduction methods
2. If there are statistically significant interactions between the following, in the performance of the *k*-hyperparameter tuning techniques in high-dimensional space clustering:
    a) The type of a dataset and its level of dimensionality.
    b) The type of a dataset and the data dimensionality reduction method.
    c) The level of dimensionality of a dataset and the data dimensionality reduction method used
    d) The *k*-hyperparameter tuning technique, type of a dataset and its level of dimensionality.
    e) The *k*-hyperparameter tuning technique, type of a dataset and the data dimensionality reduction method used.
    f) The *k*-hyperparameter tuning technique, the level of dimensionality of a dataset and the data dimensionality reduction method used
    g) Type of a dataset, its level of dimensionality and the data dimensionality reduction method used.
3. If there is a statistically significant difference between the performance of the newly developed *k*-hyperparameter tuning technique and the existing ones, in a variety of datasets.

3.4.2 Experiments

Conducting main experiments during the empirical analysis involves a systematic and rigorous process that firstly aims at identifying the independent variables, moderating variables and the dependent variables. The ground work of the experimental design is established through a small-scale preliminary investigation of a pilot study as proposed by Soin et al. (2022). In specific, the pilot study consists of three key steps. Firstly, a thorough review of the existing *k*-hyperparameter tuning techniques is conducted in order to determine their efficacy. Secondly, a series of preliminary experiments are executed using the PCA dimensionality reduction method applied to the Leukemia dataset to assess on the techniques' performance. Lastly, the silhouette index scores are used to compare these preliminary findings and helps in validating or challenging the theoretical expectations from the theoretical analysis. For example, during the theoretical analysis, the Lasso weighted *k*-means seems extremely competitive, in regards to its evaluation metrics scores, contrary to the observation made during the actual pilot study.

In fact, the AutoElbow, A Fast Hybrid Feature Selection Based on Correlation-Guided Clustering and Particle Swarm Tuning for High-Dimensional Data as well as the Adaptive Multi-view Subspace Clustering for High-dimensional Data gives more promising results than both the Lasso weighted *k*-means algorithms. The pilot study also assesses the feasibility of the experiments conducted during the empirical analysis process. Table 3.1 shows the comparative analysis of the results from the pilot study against the theoretical analysis results. This pilot study demonstrates the criterion that underpins the selection of the best three performing techniques used in the empirical analysis.

Table 3.1: Comparative results' analysis of the pilot study against the theoretical analysis

K-hyperparameter tuning technique in high dimensional space clustering	Performance scores from the theoretical analysis	Performance scores from the pilot study
Lasso Weighted *k*-means.	-Clustering accuracy of 98.39% on Lymphoma	Silhouette index score of 0.76 on the Leukemia dataset.
Adaptive Multi-view Subspace Clustering for High-dimensional Data	-Normal mutual information of 98.31 -Clustering accuracy of 99.83% -Purity of 98.83	Silhouette index score of 0.85 on the Leukemia dataset.
Improving K-means clustering with enhanced Firefly Algorithms	-Clustering accuracy of 91.47% on Wisconsin breast cancer diagnostic data set Wbc1 - Clustering accuracy of on Iris (0.8818)	Silhouette index score of 0.81 on the Leukemia dataset.
Fast Adaptive K-Means Subspace Clustering for High-Dimensional Data	-Clustering accuracy of 67.09% on WebKB dataset - Clustering accuracy of 95.57% on Breast cancer dataset	Silhouette index score of 0.79 on the Leukemia dataset
AutoElbow	Clustering accuracy of 100% on iris dataset	Silhouette index score of 0.86 on the Leukemia dataset
A Fast Hybrid Feature Selection Based on Correlation-Guided Clustering and Particle Swarm Tuning for High-Dimensional Data	- Clustering accuracy of 97.49% on Prostate - Clustering accuracy of 100% on Coil 20 - Clustering accuracy of 100% on Leukemia_Small - Clustering accuracy of 100% on DLBCL	Silhouette index score of 0.84 on the Leukemia dataset
Kernel Spherical K-Means and Support Vector Machine for Acute Sinusitis Classification	Clustering accuracy of 90% on Acute Sinusitis Dataset	Silhouette index score of 0.81 on the Leukemia dataset.

The challenges identified during the pilot study are used as the basis for improving the experimental design in the experiments. For example, through the pilot study, it is possible to note that running the experiments on the Google Colab's cloud environment is more computationally efficient as opposed to running them on the Jupyter Notebook on a local machine. The pilot study provides an opportunity to refine both the experimental procedures in the experimental checklists as well as the observation check lists used as the main research instruments during the empirical analysis process. During the pilot study, a cross sectional survey on the independent, moderating and dependent variables are investigated in order to refine the experimental research design of the experiments in the empirical analysis process. The *k*-hyperparameter tuning techniques are identified as the independent variables while the dimensionality reduction methods and the high dimensional input datasets are identified as the moderating variables. The dependent variables are identified as the performance indicators of the *k*-hyperparameter tuning techniques, measured using a set of the four common internal validation indexes and the cluster building times. Using the cross sectional survey, an analytical approach is adopted to help focus on the cause and effect of the moderating variables on the dependent variables, derived from both the performance indicators and the quality of clustering results from the *k*-hyperparameter tuning techniques, as proposed by Buresh et al. (1982).

In line with the research hypothesis, the existing best performing *k*-hyperparameter tuning techniques are identified as the independent variables, high dimensional datasets as the moderating variables while the internal validation indexes and the algorithm's run times are identified as the dependent variables. The datasets used in the experiments are prepared through pre-processing in order to ensure quality, and split into both the training and the testing datasets in the ratio 70:30.

The various techniques in the various experiments are trained and run in specified number of iterations and runs, recording both the internal validation index scores and the run times for each set of experiments, in a table of results. With an aim of identifying the main effects and interactions among the different variables in the *k*-hyperparameter tuning process, the factorial design on these variables results into a number of experiments. In Table 3.2, the first set of experiments, based on the existing three best performing techniques, a variety of datasets and the PCA based dimensionality reduction methods, are presented.

Table 3.2: Experiments on the various techniques and datasets using the PCA methods

Experiment Serial Number	Technique	High dimensional input dataset	Dimensionality reduction method used
1	*k*HT1	Tox171 dataset	Standard PCA
2	*k*HT1	Reuters dataset	Standard PCA
3	*k*HT1	Yale Image dataset	PCA with ZCA whitening
4	*k*HT1	Lung Cancer dataset	PCA with ZCA whitening
5	*k*HT1	Heart Beat sounds dataset	PCA Filter Bank
6	*k*HT1	Covid19 Coughs dataset	PCA Filter Bank
7	*k*HT1	YouCook dataset	PCA with Spatiotemporal Cubes

8	*k*HT1	YouCook2 dataset	PCA with Spatiotemporal Cubes
9	*k*HT2	Tox171 dataset	Standard PCA
10	*k*HT2	Reuters dataset	Standard PCA
11	*k*HT2	Yale Image dataset	PCA with ZCA whitening
12	*k*HT2	Lung Cancer dataset	PCA with ZCA whitening
13	*k*HT2	Heart Beat sounds dataset	PCA Filter Bank
14	*k*HT2	Covid19 Coughs dataset	PCA Filter Bank
15	*k*HT2	YouCook dataset	PCA with Spatiotemporal Cubes
16	*k*HT2	YouCook2 dataset	PCA with Spatiotemporal Cubes
17	*k*HT3	Tox171 dataset	Standard PCA
18	*k*HT3	Reuters dataset	Standard PCA
19	*k*HT3	Yale Image dataset	PCA with ZCA whitening

20	*k*HT3	Lung Cancer dataset	PCA with ZCA whitening
21	*k*HT3	Heart Beat sounds dataset	PCA Filter Bank
22	*k*HT3	Covid19 Coughs dataset	PCA Filter Bank
23	*k*HT3	YouCook dataset	PCA with Spatiotemporal Cubes
24	*k*HT3	YouCook2 dataset	PCA with Spatiotemporal Cubes

The following activity diagram represents all the experiments in the first set of experiments, using the PCA based dimensionality reduction methods across all high dimensional datasets.

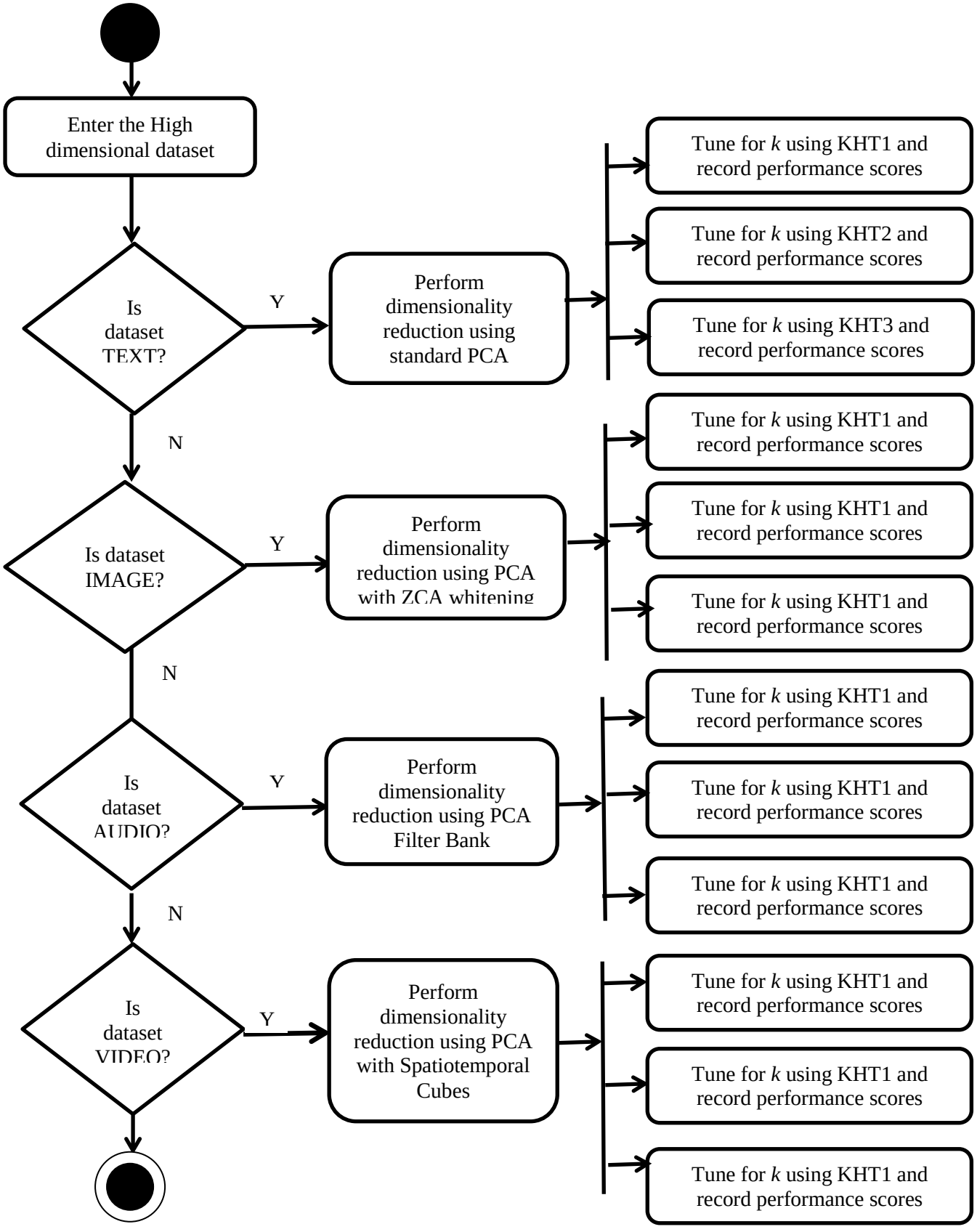


Figure 3.2: Activity Diagram for the First Set of Experiments

In the second set of the experiments, the Randomized Singular Value Decomposition is applied to the text datasets while the Economy Singular Value Decomposition is applied to the image, audio and video datasets. In the third set of experiments, the Exploratory Factor Analysis is applied to the text datasets while the Factorization machine is applied to image, audio and the video datasets. In the fourth set of experiments, the standard Locally Linear Embedding (LLE) is applied to the text datasets. The Superpixel-Based LLE is applied to the images. The Spectrogram-Based LLE is applied to the audio datasets while the Spatial Temporal LLE is applied to the video datasets. In the fifth set of experiments, the standard t-Distributed Stochastic Neighbourhood Embedding (t-SNE) is applied to the text datasets while the Multicore t-SNE is applied to the images, audio and the video datasets. In the sixth set of experiments, the Polynomial Function based Kernel PCA is applied to the text datasets while the Radial Basis Function based Kernel PCA is applied to the images, audio and the video datasets. In the seventh set of experiments, the standard UMAP is applied to text datasets. The "UMAP-learn for images" is applied to the image datasets. The Spectrogram UMAP is applied to the audio datasets while the Spatio-Temporal UMAP is applied to the video datasets. In the eighth set of experiments, the autoencoder's architecture is made flexible to accommodate the different datasets i.e. text, image, video and audio. This flexibility involved changing both the architectural-related and the training related hyperparameter settings using the Keras Tuner, the Hyperband and the Bayesian Optimization libraries. The TensorFlow, Pytorch and Scikit-Learn Python libraries are also used. The last column in Table 3.2 as well as the dimensionality reduction methods in the activity diagram in Figure 3.2 is changed, after each set of experiment, based on this narrative. The preprocessing of the image datasets involves normalization of pixel values in the range of 0 and 1. Noise is removed using wiener filtering, providing a good balance between noise reduction and signal preservation through minimization of the mean square error.

Preprocessing of the text datasets involves tokenization, stop word removal, stemming, vectorization and normalization. Vectorization is done using both the Term Frequency-Inverse Document Frequency (TF-IDF) and the word2vec, forming a combined feature vector for each text dataset. Preprocessing of the audio datasets involves both the normalization and extraction of spectrograms. Preprocessing of the video datasets involves frames extraction, normalization, temporal sampling as well as resizing frames to standard resolution. For all the experiments, the number of iterations is set to 100 while the number of runs is set to 15. Also, the Scalable and adaptable *k*-means ++ initialization method is used across all the techniques. In all the experiments, data is split into both the training and testing datasets at a ratio of 70:30. For each experiment, the Silhouette Index, Dunn Index, Calinski Harabsz Index, Davies Bouldin Index as well as the run time scores are recorded as the performance metrics. In the validation experiments, the various techniques and datasets using the best performing data dimensionality reduction methods are presented in Table 3.3.

Table 3.3: Validation Experiments on the KHT techniques using the Best Performing Dimensionality Reduction Methods in a Variety of Datasets

Experiment Serial Number	Technique Used	High Dimensional Input Dataset	Dimensionality Reduction Method Used
1	*k*HT1	Tox171 dataset	Best performing method on text datasets
2	*k*HT1	Reuters dataset	Best performing method on text datasets
3	*k*HT1	Yale Image dataset	Best performing method on image datasets
4	*k*HT1	Lung Cancer dataset	Best performing method on image datasets

5	*k*HT1	Heart Beat sounds dataset	Best performing method on audio datasets
6	*k*HT1	Covid19 Coughs dataset	Best performing method on audio datasets
7	*k*HT1	YouCook dataset	Best performing method on video datasets
8	*k*HT2	Tox171 dataset	Best performing method on text datasets
9	*k*HT2	Reuters dataset	Best performing method on text datasets
10	*k*HT2	Yale Image dataset	Best performing method on image datasets
11	*k*HT2	Lung Cancer dataset	Best performing method on image datasets
12	*k*HT2	Heart Beat sounds dataset	Best performing method on audio datasets
13	*k*HT2	Covid19 Coughs dataset	Best performing method on audio datasets
14	*k*HT2	YouCook dataset	Best performing method on video datasets
15	*k*HT3	Tox171 dataset	Best performing method on text datasets
16	*k*HT3	Reuters dataset	Best performing method on text datasets
17	*k*HT3	Yale Image dataset	Best performing method on images
18	*k*HT3	Lung Cancer dataset	Best performing method on images
19	*k*HT3	Heart Beat sounds dataset	Best performing method on audio datasets

20	*k*HT3	Covid19 Coughs dataset	Best performing method on audio datasets
21	*k*HT3	YouCook dataset	Best performing method on video datasets
22	*k*HT4	Tox171 dataset	Improved autoencoder
23	*k*HT4	Reuters dataset	Improved autoencoder
24	*k*HT4	Yale Image dataset	Improved autoencoder
25	*k*HT4	Lung Cancer dataset	Improved autoencoder
26	*k*HT4	Heart Beat sounds dataset	Improved autoencoder
27	*k*HT4	Covid19 Coughs dataset	Improved autoencoder
28	*k*HT4	YouCook dataset	Improved autoencoder

The following is an activity diagram for validation experiments on the *k*-hyperparameter tuning techniques, using the best performing dimensionality reduction methods in a variety of high dimensional datasets

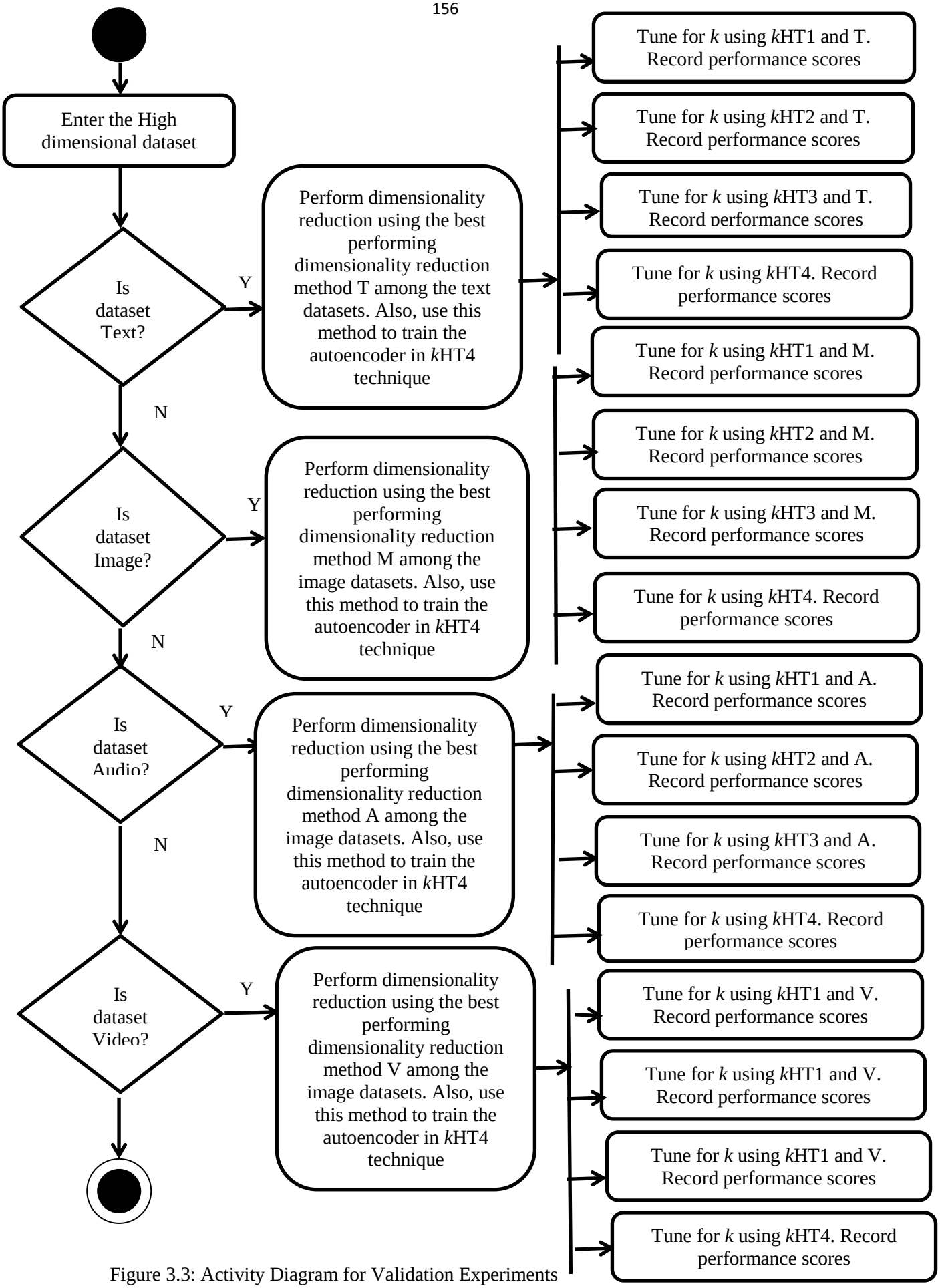


Figure 3.3: Activity Diagram for Validation Experiments

3.4.3 Integrated Development Environment and the Libraries

The Google Colab's Python IDE is used to implement the various models used in the experiments. The choice of the Google Colab as opposed to the Jupyter Notebook is due to the fact that the former provides a free cloud-based environment with access to both the Graphics Processing Unit (GPU) and Tensor Processing Unit (TPU) resources. Both the GPU and TPU are powerful resources which are beneficial when working with large high-dimensional datasets and computationally intensive algorithms. Different Python libraries have different functionalities (Saabith et al., 2020). Table 3.4 shows the different Python libraries that are used for the different high dimensional input datasets.

Table 3.4: Python Libraries for the Experimental High Dimensional Input Datasets

Type of High Dimensional Input Dataset	Python libraries used
Text	The Natural Language Toolkit, Pandas, Scikit-Learn and Numpy Python libraries were used for the purpose of data loading, pre-processing and analysis in the text datasets.
Image	The Open Source Computer Vision Library (OpenCV), Scikit-Image, Matplotlib, and Numpy Python libraries were used for the data loading, pre-processing and analysis of the image datasets.
Audio	The Librosa, Scipy and the Moviepy Python libraries were used for the data loading, pre-processing and analysis of the video datasets.

	In some instances, the TorchAudio python library was used in the place of the Librosa library when analysing the audio datasets
Video	The OpenCV, Librosa and Moviepy Python libraries were used for the data loading, pre-processing and analysis of the audio datasets

3.4.4 Research Instruments

In this research study, experimental checklists, observation check lists and the Python script to the YouTube Application Interface (API) are used as the main research instruments to facilitate the collection of data and information in an accurate and systematic manner. The choice of these research instruments is adapted on the research objectives, playing a critical role in the quality, reliability and validity of the research findings as proposed by Hinds (2002). The check lists ensure that the experiments on the performance of the different *k*-hyperparameter tuning techniques are conducted systematically, consistently, and with attention to important details. They also help in effective planning, executing, and documenting the results from the various experiments on the *k*-hyperparameter tuning techniques, in a variety of high dimensional datasets and dimensionality reduction methods. The observation check lists provide a mechanism through which the data from the various experiments are systematically recorded and documented in order to maintain consistency, objectivity, and thoroughness in the observations. This enhances the rigour, validity and reliability of the research study. Lastly, the Python script to the YouTube API provides a mechanism to collect primary video data from the popular Chef Raphael's Vlogs through an automated data mining strategy.

3.4.5 Population, Sampling Strategy and Sample size

In line with the purposive sampling strategy proposed by Etikan et al. (2016), the results of the theoretical analysis and the initial empirical study in form of a pilot study, a set of variables for use in the main experiments of the empirical analysis process are selected. These include the three best performing *k*-hyperparameter tuning techniques from a population of thirty eight, eight benchmark high dimensional datasets from a population of twenty three as well as eight state of the art dimensionality reduction methods from a population of fifteen. These variables have been presented in the subsequent Sections *3.4.5.1*, *3.4.5.2*, *3.4.5.3* and *3.4.5.4*. The purposeful sampling strategy adopted in this research study is helpful for both the cost and time efficiency as opposed to large random samples that can be expensive and time-consuming to sample, as proposed by Banerjee et al. (2023). The strategy for the adoption of the benchmark datasets, reffered to as the standardized collection of datasets that are commonly used to evaluate and compare the performance of different algorithms in machine learning, is in line with proposal by Homburg et al. (2005). In this research, these benchmark datasets provide a common basis for evaluating the performance of various *k*-hyperparameter tuning algorithms. By using the benchmark datasets, it is possible to objectively compare the effectiveness of the different techniques in the variety of high dimensional datasets and dimensionality reduction techniques against the baseline models. Moreover, the replication of experiments and results is easy since the benchmark datasets are publicly available and well-documented. This paves way for reliability and reproducibility of the research findings besides enhancing the real-world relevance. The selection of papers from the initial thirty eight to the final three is guided by a focus on methodological rigor and reproducibility, emphasizing studies with strong experimental setups, evaluation rigor, and relevance to *k*-hyperparameter tuning within high-dimensional spaces.

Given the limited number of articles specifically addressing the *k*-hyperparameter tuning in this context, the need for an in-depth analysis necessitates a shift towards quality over quantity. The final selection is narrowed down to three papers that demonstrate significant contributions and novelty in the research domain, including techniques that generalize well. Purposive sampling of these high-quality articles is deemed more relevant to the research objectives compared to random sampling, which might not have adequately addressed the specific aims of the study. This approach also extends to the dimensionality reduction methods analyzed. Additionally, resource efficiency plays a crucial role in the selection process. Evaluating algorithms can be computationally intensive, and analyzing all possible combinations can be both resource-heavy and time-consuming. By employing purposive sampling, the focus on a manageable subset of combinations yields high-quality data while maintaining satisfactory statistical power. This approach is not only practical but also feasible within the constraints of available resources and the timeframe for this doctoral thesis. Finally, concentrating on the most promising and effective techniques and dimensionality reduction methods provides deeper insights, sparking new ideas for potential novel techniques to address existing gaps in the research landscape of *k*-hyperparameter tuning in high-dimensional space clustering. This focused analysis enhances the overall quality and relevance of the empirical study, contributing to meaningful advancements in the field.

3.4.5.1 High Dimensional *K*-hyperparameter Tuning Techniques

The AutoElbow (Onumanyi et al., 2022), Fast Hybrid Feature Selection Based on Correlation-Guided Clustering and Particle Swarm Tuning for High-Dimensional Data (Song et al., 2021) and the Adaptive Multi-view Subspace Clustering for High-dimensional Data (Yan et al., 2020) are the three sample techniques picked from sixteen techniques.

In order to enhance the data analysis process, these techniques are encoded as kHT1, kHT2 and kHT3 respectively. The choice on the sample techniques is based on a purposive strategy where the techniques with the most promising results are identified during both the theoretical analysis process and the pilot study. Besides this, the choice on the Autoelbow technique is based on the fact that the traditional elbow method, one of the *k*-hyperparameter tuning methods with a long-standing literature. During the evaluation of the effectiveness of the new technique against the existing ones in a variety of datasets, the newly developed technique is encoded as kHT4. The new technique is referred to as the Ensemble based *k*-hyperparameter tuning technique on high-dimensional space using a self-adapting autoencoder and internal validation indexes (Gikera et al., 2023b). This technique is developed from the insights of the ANOVA analysis of the experimental data from the main experiments, besides a theoretical analysis and the initial empirical study in form of a pilot study. Across all the techniques, an adaptable scalable *k*-means++ initialization method is used as the initial centroids' determination strategy, in line with Hämäläinen et al., (2020). The adaptability of the *k*-means++ to capture spatial relationships in the video datasets is intended to capture the temporal relationships among frames and video segments in the determination of the centroids. The adaptability of the *k*-means++ to capture the temporal nature of the audio signals is intended to intelligently initialize the cluster centroids based on the frequency and time-domain characteristics of the audio datasets. The adaptability of the *k*-means ++ to capture the image-specific features such as color histograms and texture descriptors is intended to ensure that the centroid initialization method aligns well with the complexities of the image representations. The adoption of the scalable and adaptable *k*-means ++ initialization methods for each variety of the high dimensional datasets is based on the need to efficiently capture the data-specific features and scale appropriately.

3.4.5.2 High Dimensional Input Datasets

Tox-171, Yale Image, Reuters, YouCook, Heart Beat Sounds, Lung Cancer and the Covid-19 Coughs datasets are the high dimensional datasets used in this empirical study. The purposive sampling strategy on the datasets is based on the four common data types i.e. text, image, video and audio, besides being benchmark datasets with high dimensionality. Table 3.4 shows the description of the different high-dimensional bench mark datasets downloaded from the various machine learning repositories including Kaggle.

Table 3.4: Description of the Experimental High Dimensional Datasets

High Dimensional Dataset	Description and Type of the Dataset, Number of Features (P) and Number of Instances (N)
Tox-171 Dataset	Tox -171 dataset is a high-dimensional benchmark dataset based on gene expression. It contains 171 instances, 5,748 features and 4 clusters. The clusters contained in this gene expression dataset include: patients without cancer, patients whose radiation-therapy have been controlled, patients who are suffering from skin cancer, and patients who have suffered from the radiation-therapy (Moslemi & Ahmadian, 2023).
Yale Image Dataset	The Yale Image Dataset is a widely used benchmark dataset in the field of computer vision and facial recognition. It contains 1,024 features, 165 instances and 15 clusters. It was created by researchers at Yale University and consists of gray scale images of human faces, each captured under different lighting conditions, facial

	expressions, and orientations. For each person, there are different images with various expressions (such as smiling or frowning) and 64 different lighting conditions (from different angles and intensities). The resolution of the images in the original Yale Face Database is 320 x 243 pixels, but they are often resized or cropped for specific experiments or applications. The images are typically stored in a matrix format, where each pixel's intensity value is represented by a gray scale value ranging from 0 to 255. The Yale Face Database has been widely and successfully used for research and evaluation of face recognition algorithms, illumination modeling, facial expression analysis, and other related tasks. Researchers use this dataset to develop and test algorithms and models for facial recognition, emotion recognition, and other applications in the field of computer vision (Calli et al., 2017).
Reuters-21578 dataset	Reuters-21578 dataset is a well-known benchmark text based dataset in the field of natural language processing (NLP) and text classification. It is widely used for tasks such as document categorization, topic modeling, and text clustering. This dataset contains 18,933 features, 8,293 instances and 20 clusters. The Reuters dataset consists of news articles collected from the Reuters news agency. It covers various topics, including business, finance, politics, sports, and more. The articles are clustered into different categories or topics. The Reuters dataset is often used in research and evaluation of text clustering algorithms and models.

	It offers a diverse collection of documents with different topics and enables researchers to develop and test algorithms that can automatically cluster news articles into the relevant categories (Fürnkranz, 1998).
Lung Cancer Dataset	Lung cancer dataset is a computational medicine benchmark dataset based on images and includes clinical attributes such as patient age, smoking history, and tumor characteristics, along with the target variable indicating whether the patient has lung cancer or not. Lung cancer dataset consists of 12,533 features, 181 instances and 2 clusters. It contains radiological imaging features from CT scans and genetic mutation data, with clinical information on patient demographics, tumor characteristics, genetic markers, and treatment outcomes. Researchers use this dataset to investigate risk factors, develop diagnostic models, predict treatment response, or know biomarkers for lung cancer (Naseriparsa & Kashani, 2014).
Covid-19 Coughs Dataset	Covid-19 Coughs Dataset is an audio data, including cough sounds, to aid in the detection and diagnosis of COVID-19. Fever, fatigue and dry coughs are the most common symptoms of COVID-19. The shortness of breath, joint pain, muscle pain, gastrointestinal symptoms and loss of smell or taste are also other symptoms. However, among all these symptoms, coughing is the most predominant symptom which this specific dataset addresses.

	The high dimensional dataset contains 1179 instances, separated among 2 clusters i.e. 92 Covid-19 positive patients and 1079 Covid-19 negative patients. This dataset was collected from across all continents except Africa (Laguarta et al., 2020).
Heart Beat Sounds Dataset	This dataset contains heart sound recordings from patients with various heart conditions. The dataset contains 4 clusters. These clusters are normal and abnormal heart sounds (murmers) associated with various cardiac pathologies. The abnormal heart sounds are: aortic valve stenosis, aortic insufficient and ventricular septum defect. The dataset has 50,000 features (Amit et al., 2009).
YouCook and YouCookII datasets	The YookCook and YouCookII datasets are popular benchmark datasets in the field of computer vision and video understanding. They are specifically focused on the task of action recognition and anticipation in cooking videos. The datasets provide a collection of instructional cooking videos along with associated annotations. The YouCook datasets consist of 2,000 cooking videos, where each video demonstrates a step-by-step process of preparing a recipe. The videos are sourced from YouTube and a wide variety of 89 recipes and cooking styles. The datasets consists of 40,096 features including a diverse range of ingredients, cooking techniques, and culinary actions. For each video, the datasets provide detailed annotations that include temporal annotations and action labels.

	Temporal annotations mark the start and end times of specific actions within each video. These annotations indicate when a particular cooking action occurs, such as chopping vegetables, mixing ingredients, or frying. Action labels have each action within the video labeled with a descriptive action name. These labels specify the type of action being performed in the corresponding video segment. The YouCook datasets have been widely used for various tasks, such as action recognition, action anticipation, and video understanding. In this research, it was used to cluster the different recipes based on the temporal annotations and action plans (Yagcioglu et al., 2018).
Recipes for cooking beef in Kenya	Recipes for cooking beef in Kenya were mined from Chef Raphael's video blogs as a source of primary data in this research. This was done using a Python script to the YouTube API.

The numbers of features present in a dataset, in most cases, have a more significant impact on the algorithm's run time as compared to the numbers of records in a dataset. With respect to the the number of instances in a dataset, most algorithms exhibits both the logarithmic (O(log n)) and the linear time complexities (O(n)). On the other hand, with respect to the number of features in a dataset, most algorithms exhibit both the polynomial (O(n^k) and the exponential time complexities (O(2^n)). The time complexities of the latter are longer than the ones of the former. This observation is in line with the one made by Bian and Qian (2022).

3.4.5.3 Pre-training Datasets

In the pre-training during the unsupervised transfer learning process of the new *k*-hyperparameter tuning technique's autoencoder, the cosine similarity between the feature vectors of both the source dataset and the target dataset guides the selection process. Tox21 and BBC news datasets are used on the text datasets. The Olivetti Research Laboratory (ORL) face dataset and the Non Small Cell Lung Cancer (NSCLC) Radiogenomics datasets are used for the image datasets. The Stethoscope Heart Sounds and the Coswara datasets are used for the audio datasets while the MPII cooking activities dataset is used in the pre-training for the video datasets. Tox21 dataset is a widely used toxicology dataset that plays a significant role in predictive toxicology and chemical risk assessment. It was initiated as a collaborative effort by multiple United State (U.S.) federal agencies, including the National Institutes of Health (NIH), the U.S. Environmental Protection Agency (EPA), and the National Institute of Environmental Health Sciences (NIEHS). Tox21 is based on high-throughput screening (HTS) that involve the rapid testing of a large number of chemical compounds for their effects on biological systems. The dataset includes results from thousands of such assays and focuses on various toxicological endpoints or adverse effects, including cytotoxicity, genotoxicity, endocrine disruption, and more.

It provides data on the potential toxicity of chemical compounds. Tox21 covers a diverse set of chemical compounds, including environmental pollutants, pharmaceuticals, and other chemicals. Researchers use this data to assess the safety and potential hazards associated with these compounds. Tox21 is known for its large-scale data, making it a valuable resource for the development and validation of computational models and machine learning algorithms in toxicology (Idakwo et al., 2020).

The BBC News Summary Dataset is a collection of news articles and their corresponding article summaries, often used for text summarization and clustering tasks. This dataset contains news articles from the BBC News website. The dataset contains news articles from various categories, including politics, technology, entertainment, sports, and more. Each news article is associated with a human-generated summary, typically a shorter version of the article. The dataset includes both the full news articles and their corresponding summaries, making it suitable for tasks such as extractive or abstractive summarization. News articles are categorized into different topics or sections, allowing researchers to focus on specific domains of interest. The dataset primarily contains news articles in English (Urologin, 2018).

The ORL face dataset contains grayscale facial images of several individuals, each captured under controlled and consistent lighting conditions. There are ten different images for each individual, capturing various facial expressions and poses. The images in the ORL Face Database, typically stored in a standard format, such as JPEG or BMP, provide variations in facial expressions (e.g., neutral, smiling), head poses (e.g., left and right poses), and lighting conditions. The controlled environment ensures that the lighting variations are minimal and consistent across the images. It was primarily developed for face detection, face recognition, and facial expression analysis (Saraswathi & Sivakumari, 2015).

The NSCLC Radiogenomics dataset is a specialized dataset that combines radiological imaging data with genomic information, specifically for non-small cell lung cancer. This dataset is designed for research into the relationship between the radiological features observed in medical imaging, such as Computed Tomography (CT) scans, and the genomic characteristics of NSCLC tumors. NSCLC is the most common type of lung cancer, accounting for the majority of lung cancer cases.

The dataset is focused on NSCLC tumors, which can be further classified into various subtypes. The dataset also includes radiological features extracted from CT scans of NSCLC patients. These features describe the size, shape, texture, and other characteristics of tumors as observed in medical images on the lungs. Genomic data is a critical component of the dataset, allowing researchers to study the genetic profile of NSCLC tumors. This information includes Dioxyribo Nucleic Acid (DNA) and molecular data related to the tumors' genetic makeup. The NSCLC Radiogenomics dataset was first downloaded as a MATLAB MATrix (MAT) file but converted to Comma Separated File (CSV) file using the Scipy and Panda libraries. The scipy.io module in the Scipy Python library provides a function called loadmat that reads MAT files, and converted to CSV file using Panda library (Shiri et al., 2020).

The Stethoscope Heart Sound datasets refer to collections of audio recordings of heart sounds (phonocardiograms) obtained using a stethoscope. These datasets are valuable resources for research and clinical applications in cardiology and healthcare. They may contain recordings of various heart sounds, including the normal heart sounds and abnormal heart sounds associated with different cardiac conditions (Schmidt et al., 2008).

The Coswara Dataset is a collection of respiratory sound recordings and metadata that has been widely used in research related to respiratory health, including the analysis of cough and breathing sounds. This dataset is notable for its relevance to respiratory conditions and has been used for tasks such as COVID-19 detection, cough classification, and the study of respiratory diseases (Chaudhari et al., 2020).

The MPII Cooking Activities dataset is a benchmark dataset in the field of computer vision and action recognition, particularly in the context of culinary research. It is rich with intricate details of cooking video activities. These activities are a collection of video recordings that demonstrate various cooking activities and tasks such as chopping, stirring, frying, and assembling dishes. The videos in the dataset are typically annotated to identify and label specific actions and activities. These annotations help researchers train and evaluate action recognition models. The dataset covers a variety of recipes and cooking styles, making it a valuable resource for understanding how cooking activities are performed in different contexts and culinary traditions (Kuehne et al., 2015).

Pre-training the autoencoder using one dataset and then applying the unsupervised transfer learning with the best performing dimensionality reduction methods, in respect to the specific variety of the high dimensional dataset, is a multi-step process that involves feature extraction, representation learning, and knowledge transfer. When using the MPII cooking activities dataset to pre-train the autoencoder for clustering task on the Youcook dataset, the Spatio-Temporal UMAP is first applied to the MPII cooking activities dataset in order to reduce its dimensionality while retaining the most important features. This UMAP transformation serves as the feature extraction step. After this, the autoencoder architecture is designed for feature extraction and representation learning. The encoder has the same input dimension as the Spatio-Temporal UMAP -transformed features from the MPII cooking activities dataset. Then, the unsupervised transfer learning takes place. Here, the encoder part of the autoencoder with the weights learned from the Spatio-Temporal UMAP on the MPII cooking activities dataset is initialized. This initialization transfers knowledge from the MPII Cooking Activities dataset to the YouCook dataset. Using the Spatio-Temporal UMAP-transformed features of the YouCook dataset as input, the autoencoder is trained.

The goal is to learn a meaningful representation of the YouCook data. The learned representations in the latent space of the autoencoder based on the YouCook dataset are extracted. These representations capture the essential features of the YouCook data. Evaluation and fine tuning of the model continues until the best clustering scores are output. At such best clustering scores, the *k*-hyperparameter is picked as optimal. All the data is divided into two subsets i.e. training and testing. The training data is used to train the model while the testing data is used to evaluate the final model's performance.

3.4.5.4 Experimental Dimensionality Reduction Methods

The choice of the data dimensionality reduction methods is limited to the unsupervised methods only as clustering is an unsupervised task. The data dimensionality reduction methods used in this empirical study included: Principal Component Analysis (PCA) based methods, Kernel PCA based methods, Singular Value Decomposition (SVD) based methods, Factor Analysis based methods, Locally Linear and Embedding methods, t-Distributed Stochastic Neighborhood Embedding (t-SNE) based methods, Uniform Manifold Approximation and Projection (UMAP) based methods, and an autoencoder. The choice of the best variation, extension or the kernel function for every data dimensionality reduction method is made based on both the expert judgment, theoretical analysis as well as the experimentation process, to help in finding the most suitable method for a particular variety of a high-dimensional dataset. In Principal Component Analysis (PCA), the standard PCA is applied to text datasets. The PCA with ZCA whitening is applied to the image datasets.

The Principal Component Analysis with Spatiotemporal Cubes is applied to the video datasets while the PCA Filter Bank is applied to the audio datasets. The various variants of the Principal Component Analysis are applied because of their effectiveness in handling the respective types of the high dimensional datasets.

In Singular Value Decomposition (SVD), the Randomized SVD is applied to the text datasets. The Economy SVD is applied to the image, audio and video datasets. The various variants of the Singular Value Decomposition are applied because of their effectiveness in handling the respective types of the high dimensional datasets.

In the Factor Analysis (FA), the Exploratory Factor Analysis is applied to the text datasets. The Factorization machine is applied to image, audio and the video datasets. The various variants of the Singular Value Decomposition are applied because of their effectiveness in handling the respective types of the high dimensional datasets.

In the Locally Linear Embedding (LLE), the standard LLE is applied to the text datasets. The Superpixel-Based LLE is applied to the images. The Spectrogram-Based LLE is applied to the audio datasets while the Spatial Temporal LLE is applied to the video datasets. The various variants of the Locally Linear Embedding are applied because of their effectiveness in handling the respective types of the high dimensional datasets.

In the t-Distributed Stochastic Neighborhood Embedding (t-SNE), the standard t-SNE is applied to the text datasets. The Multicore t-SNE is applied to the images, audio and the video datasets.

The various variants of the t-Distributed Stochastic Neighborhood Embedding are applied because of their effectiveness in handling the respective data types. In the Kernel Principal Component Analysis (Kernel PCA), the Polynomial Function based Kernel PCA is applied to the text datasets. The Radial Basis Function based Kernel PCA is applied to the images, audio and the video datasets. The various variants of the Kernel Principal Component Analysis are applied because of their effectiveness in handling the respective types of the high dimensional datasets.

In the Uniform Manifold and Projection (UMAP), the standard UMAP is applied to text datasets. The "UMAP-learn for images" is applied to images. The Spectrogram UMAP is applied to the audio datasets while the Spatio-Temporal UMAP is applied to video datasets. The various variants of the Uniform Manifold and Projection are applied because of their effectiveness in handling the respective types of the high dimensional datasets. Lastly, on the autoencoder, its architecture is made flexible to accommodate the different datasets i.e. text, image, video and audio. This flexibility mainly involved changing both the architectural-related and the training related hyperparameter settings using the Keras Tuner, the Hyperband and the Bayesian Optimization Python libraries.

3.4.6 Experimental Data Collection Process and Techniques

In this research study, three specific data collection techniques are employed i.e. content analysis, experimental methods, and data mining using the YouTube API. Firstly, content analysis is utilized to systematically and objectively evaluate on the existing the *k*-hyperparameter tuning techniques. This involves analyzing datasets from machine learning repositories, such as Kaggle and the University of Michigan, in order to extract meaningful patterns and insights related to the effectiveness and application of these techniques.

The aim is to provide a comprehensive understanding of the current methodologies in the field. Secondly, experimental methods are applied to assess the performance of different experiments on the *k*-hyperparameter tuning techniques across various high-dimensional datasets and dimensionality reduction methods. In specific, the data collection process entails running these experiments and carefully recording the results i.e. the four internal validation index scores as well as the run time scores. These results, guided by both the observation and experimental checklists, are recorded in a table, which facilitates an organized approach to data analysis. This allows for a clear comparison of the effectiveness of each technique in different contexts. Lastly, data mining is conducted using the YouTube API to gather primary data. This involves extracting metadata from Chef Raphael's popular video blogs (vlogs), specifically focusing on recipes for cooking beef in Kenya. A Python script is employed to mine relevant video metadata, including titles and descriptions, which serve to validate the effectiveness of the newly developed technique. This technique is particularly valuable for collecting qualitative data that complements the quantitative findings from the content analysis and experimental methods. Overall, the selection of these data collection techniques is aligned with the research objectives, the specific types of data required, the target population, and ethical considerations, as proposed by Cln and Iro (2013). The experimental check lists for the experimental methods have been added to the appendix of this thesis document.

3.4.7 Analysis techniques on the Experimental Data

The data analysis is done using the JAMOVI R-based software, a tool proposed in the works of Musa et al., 2023 and aims at comparing the performance of the different *k*-hyperparameter tuning techniques.

The analysis is based on the one-way ANOVA, two-way ANOVA and the three-way ANOVA on the performance metrics of the different *k*-hyperparameter tuning techniques in high-dimensional space clustering, recorded in a table of results. ANOVA, or Analysis of Variance, is a statistical technique used to analyze the differences among group means in a sample. It is commonly used in hypothesis testing to determine whether there are statistically significant differences between the means of three or more independent groups (DeCoster, 2006).

The two-way and the three-way ANOVA analysis are applied in the analysis of the experimental data results from the main experiments. The metrics used in the evaluation process of the existing techniques include the Silhouette Index, Dunn Index, Calinski Harabsz Index, Davies Bouldin Index and the Cluster Building Times in the different *k*-hyperparameter tuning techniques. The scores of each of the four internal validation indexes are first normalized before computing the average index score based on their mean scores. However, the evaluation of the newly developed technique is evaluated against the existing *k*-hyperparameter tuning technique using the Kruskall Wallis H statistic and the one-way ANOVA on the ensemble validation index scores, in a variety of high-dimensional datasets. The proposed ensemble validation index is in line with the improved index for clustering validation based on Silhouette Index and Calinski-Harabasz Index only, as proposed by Wang and Xu (2019). The new ensemble validation index is also based on the Dunn Index and the Davies Bouldin Index besides the Silhoutte Index and the Calinski Harabsz Index scores. The comparison of the cluster visualizations of the different high-dimensional datasets between the standard autoencoder and the improved autoencoder on the new *k*-hyperparameter tuning technique is performed as part of evaluating the effectiveness of the new technique.

The analysis results on the performance of the existing three *k*-hyperparameter tuning techniques are used as the foundation in the design and the development of the newly developed *k*-hyperparameter tuning technique in high dimensional space clustering. Spreadsheets software, the current version of the Google workspace's Google sheets at the time of data analysis on January 2023 is used to perform the feature scaling of the different indexes through the normalization. The normalization process is done in order to standardize the scale of the internal validation indexes without distorting the differences in their values.

In statistics, normalization refers to the process of rescaling data to bring it within a common scale or range, typically for the purpose of making different variables or datasets comparable or suitable for analysis. It involves transforming data so that it conforms to a standard scale and prevents one variable from being overly influential, especially if it's measured in different units Misra (2020). The formula for the Min-max normalization is as shown below:

$$X_{normalized} = \frac{(X - X_{min})}{(X_{max} - X_{min})} \quad (3.1)$$

Once normalized, the mean score of the four commonly used internal validation indexes is used as the Average Index Performance Score (AIPS) in order to achieve a comprehensive evaluation of the performance of the *k*-hyperparameter tuning techniques. This Average Index Performance Score is translated to the Ensemble Internal Validation Index (EIVI) during the validation experiments.

$$EIVI = AVG\ (SI_{Normalized} + DI_{Normalized} + CI_{Normalized} + 1\text{-}DB_{Normalized}) \quad (3.2)$$

In clustering high-dimensional datasets, the use of multiple validation indices is crucial for comprehensively assessing the quality and effectiveness of the clustering results.

The Silhouette index provides insight into how similar an object is to its own cluster compared to the other clusters, thereby measuring the cohesion and separation of clusters. A higher silhouette score indicates better-defined clusters. The Davies-bouldin index evaluates the average similarity ratio of each cluster with its most similar cluster, effectively quantifying the separation and compactness of clusters. In this internal validation index, lower values denote better clustering performance while higher values denote poor clustering performance. The Calinski-harabasz index assesses the ratio of the sum of between cluster dispersion to within cluster dispersion. In this index, a higher score suggests that the clusters are well-separated and compact, indicating better-defined groupings. Lastly, the Dunn index focuses on identifying clusters that are well-separated and compact by comparing the minimum inter-cluster distance to the maximum intra-cluster distance. A higher Dunn index signifies improved clustering quality. Utilizing these indices collectively allows for a robust evaluation of the *k*-hyperparameter tuning techniques, ensuring that the technique not only identifies distinct groups effectively but also maintains the integrity and interpretability of the underlying high dimensional data.

In the two way ANOVA analysis, the objective is to investigate on the following:

1. If there are statistically significant differences in the performance of the *k*-hyperparameter tuning techniques based on the dimensionality levels of the high dimensional input datasets. Through this, it is possible to investigate if using a high dimensional input dataset of a particular dimensionality level, over the other, has a significant impact on the performance of the *k*-hyperparameter tuning techniques.
2. If there are statistically significant differences in the performance of the *k*-hyperparameter tuning techniques based on the choice of the dimensionality reduction method.

Through this, it is possible to investigate if using a particular dimensionality reduction method, over another, has a significant impact on the performance of the *k*-hyperparameter tuning techniques.

3. If there are statistically significant differences in the performance of the *k*-hyperparameter tuning techniques based on the data types of the high dimensional input datasets. Through this, it is possible to investigate on whether the use of a high dimensional input dataset of a particular data type, over another, has a statistically significant impact on the performance of the *k*-hyperparameter tuning techniques.
4. If there are statistically significant interactions between the dimensionality levels of the high dimensional input datasets and the *k*-hyperparameter tuning techniques, in their performance. Through this, it is possible to investigate if the above interaction, if any, is due to the dimensionality levels of the high dimensional input datasets at their individual levels, or if it is due to the *k*-hyperparameter tuning techniques at their individual levels, or if it is due to a combination of both the two variables.
5. If there are statistically significant interactions between the choice of the dimensionality reduction methods and the the *k*-hyperparameter tuning techniques, in their performance. Through this, it is possible to investigate if the above interaction, if any, is due to the dimensionality reduction methods at their individual levels, or if it is due to the *k*-hyperparameter tuning techniques at their individual levels, or if it is due to a combined effect of both the two variables.
6. If there are statistically significant interactions between the data type of the high dimensional input datasets and the *k*-hyperparameter tuning techniques, in their performance.

Through this, it is possible to investigate if the above interaction, if any, is due to the data types at their individual levels, or if it is due to the *k*-hyperparameter tuning techniques at their individual levels, or if it is due to a combined effect of both the two variables.

7. If there are statistically significant differences between the dimensionality levels of the high dimensional input datasets and their data types, on the performance of the *k*-hyperparameter tuning techniques. Through this, it is possible to investigate if the choice of a particular data type and dimensionality level of the high dimensional input dataset, over others, has a statistically significant impact on the performance of the *k*-hyperparameter tuning techniques.
8. If there are statistically significant interactions between the dimensionality levels of the high dimensional input datasets and their data type, in the performance of the *k*-hyperparameter tuning techniques. Through this, it is possible to investigate if the above interaction, if any, is due to the dataset's dimensionality levels at the individual level, dataset's type at the individual level, or if it is due to a combined effect of both the two variables.
9. If there are statistically significant differences between the data type of the high dimensional input datasets and the dimensionality reduction methods used, on the performance of the *k*-hyperparameter tuning techniques.

   Through this, it is possible to investigate if the choice of a particular dimensionality reduction method on a dataset of a particular data type, over another, has a significant impact on the performance of the *k*-hyperparameter tuning techniques.
10. If there are statistically significant interactions between the data type of the high dimensional input datasets and the data dimensionality reduction method used, on the performance of the *k*-hyperparameter tuning techniques.

Through this, it is possible to investigate if the above interaction, if any, is due to the data types at their individual levels, dimensionality reduction methods at their individual levels, or if it is due to a combined effect of both the two variables.

11. If there are statistically significant differences between the dimensionality levels of the high dimensional input datasets and the data dimensionality reduction methods, on the performance of the *k*-hyperparameter tuning techniques. Through this, it is possible to investigate if the choice of a particular dimensionality reduction method on a dataset of a particular dimensionality level, over another, has a significant impact on the performance of the *k*-hyperparameter tuning techniques.
12. If there are statistically significant interactions between the dimensionality levels of the high dimensional input datasets and the dimensionality reduction method used, in the performance of the *k*-hyperparameter tuning techniques.

    Through this, it is possible to investigate if the above interaction, if any, is due to the dimensionality levels of the datasetat their individual levels, or if it is due to the dimensionality reduction methods at their individual levels, or if is due to a combined effect of both the two variables.

In the three way ANOVA analysis, the objective was to investigate on the following:

1. If there are statistically significant differences among the performance of the different *k*-hyperparameter tuning techniques.
2. If there are statistically significant differences among the different data types in the high dimensional input datasets, in the performance of the *k*-hyperparameter tuning techniques.
3. If there are statistically significant differences among the different dimensionality levels of the high dimensional input datasets, in the performance of the *k*-hyperparameter tuning techniques.

4. If there are statistically significant differences among the different dimensionality reduction methods, in the performance of the *k*-hyperparameter tuning techniques.
5. If there are statistically significant interactions between the data types of the high dimensional input datasets and the *k*-hyperparameter tuning techniques, in their performance.
6. If there are statistically significant interactions between the data types of the high dimensional input datasets and their dimensionality levels, in the performance of the *k*-hyperparameter tuning techniques.
7. If there are statistically significant interactions between the dimensionality levels of the high dimensional input datasets and the *k*-hyperparameter tuning techniques, in their performance.
8. If there are statistically significant interactions among the *k*-hyperparameter tuning techniques, data types and dimensionality levels of the high dimensional input datasets, considering all the variables combined together.
9. If there are statistically significant interactions between the *k*-hyperparameter tuning techniques and the dimensionality reduction methods applied.
10. If there are statistically significant interactions between the dimensionality levels of the high dimensional input datasets and the *k*-hyperparameter tuning techniques, in their performance.
11. If there are statistically significant interactions between the dimensionality reduction methods and the dimensionality levels of the high dimensional input datasets, in the performance of the *k*-hyperparameter tuning techniques.
12. If there are statistically significant interactions between the dimensionality levels of the high dimensional input datasets and the *k*-hyperparameter tuning techniques, in their performance.

13. If there are statistically significant interactions among the dimensionality reduction methods, dimensionality levels of the high dimensional input datasets and the *k*-hyperparameter tuning techniques, considering all the variables combined together.
14. If there are statistically significant interactions between the dimensionality reduction methods used and the dimensionality levels of the high dimensional input datasets, in the performance of the *k*-hyperparameter tuning techniques.
15. If there are statistically significant interactions between the data types of the high dimensional input datasets and their dimensionality levels, in the performance of the *k*-hyperparameter tuning techniques.
16. If there are statistically significant interactions between the data types of the high dimensional input datasets and the dimensionality reduction methods, in the performance of the *k*-hyperparameter tuning techniques.
17. If there are statistically significant interactions among the dimensionality reduction methods, dimensionality levels of the high dimensional input datasets and their data types, considering all the variables combined together.
18. If there are statistically significant interactions between the dimensionality reduction methods and the *k*-hyperparameter tuning techniques, in their performance.
19. If there are statistically significant interactions between the data types of the high dimensional datasets and the dimensionality reduction methods, in the performance of the *k*-hyperparameter tuning techniques.
20. If there are statistically significant interactions between the data types of the high dimensional datasets and the *k*-hyperparameter tuning techniques, in their performance.

21. If there are statistically significant interactions among the *k*-hyperparameter tuning techniques, data types of the high dimensional input datasets and dimensionality reduction methods applied, considering all the variables combined together.

It is important to note that the high dimensional datasets involved in this study demonstrate intricate behaviours and undertaking a comprehensive 2-way and 3-way ANOVA analysis of both the significant differences and interactions is quite necesary. This approach provides a satisfactory analytical depth that befits this kind of a multifaceted exploration. Also, checking the lower levels of ANOVA first (2-way) is simpler and easier to verify assumptions like normality and homogeneity of variance early enough before moving into the more complex analysis in the 3-way ANOVA. In this case, the 2-way ANOVA analysis is done as a preliminary exploration of the experimental data before gradually increasing the complexity to the 3-way ANOVA analysis. Normality checks are ususally done by looking into whether whether the points lie on a straight line on the Q-Q residual plots. Lastly, the step wise analysis usually helps in better understanding the hierarchical relationships and helps in validation of findings where the consistency across multiple analyses usually indicates the reliability of the conclusions drawn from such experimental data analysis. After the data analysis process, discussions are made, with an objective of accepting or rejecting the hypothesis that was earlier formulated. The invaluable insights from the analyses on these results provides a strong foundation upon which the the new *k*-hyperparameter tuning technique is developed. These discussions are done in a prose format and are critical to the development of the summary and recommendations for future research.

3.5 Model Development

Based on the data analysis results of the experimental data from the main experiments, the newly developed *k*-hyperparameter tuning technique, in high-dimensional space clustering, is designed and implemented using a multi-methodological approach. The new technique is based on both the Autoelbow technique and the autoencoder dimensionality reduction method. The Autoelbow is based on the traditional elbow method that has a long standing literature of success in the *k*-hyperparameter tuning processes. On the other hand, the use of the autoencoder in the new technique, as opposed to the other data dimensionality reduction methods, is because of the fact that it demonstrated flexibility and promising results in accommodating a variety of high dimensional datasets. By adjusting both the architectural-related and the training-related hyperparameter settings as well as the adoption of an effective unsupervised transfer learning strategy, the new technique's improved autoencoder worked relatively well in a variety of high-dimensional datasets.

Multi-methodological system methodology approach to this development is composed of four strategies, namely: theory building, experimentation, observation and system development (Nunamaker et al., 1990). In this research study, the experimentation process provides an avenue to experiment the different *k*-hyperparameter tuning techniques in a variety of high-dimensional datasets and dimensionality reduction methods. The experimentation process also provides an avenue to evaluate on the effectiveness of the new technique against the existing ones, in a variety of both secondary and primary high-dimensional datasets. It also helps to validate the underlying theories in *k*-hyperparameter tuning. A model development process consisting of five stages, as proposed by Nunamaker et al. (1990) is adopted. These stages include: conceptual design, model architecture, model design, prototyping and experiment. These have been explained in detail in the subsequent sections.

3.5.1 Conceptual Design

This stage involves the development of a conceptual framework. This framework helps the researcher to formulate the important concepts and develop a theory to support the system development as proposed by Nunamaker et al. (1990). In this research study, the conceptual design consists of three variables i.e. independent variables, moderating variables and the dependent variables. The *k*-hyperparameter tuning techniques are used as the independent variables. The data dimensionality reduction methods as well as the high-dimensional input datasets are used as the moderating variables while the performance evaluation metrics are used as the dependent variables in the conceptual design of the new technique's model. Through the conceptual design, the blue print of the new model is defined as follows: High-dimensional datasets are first input into the model. After this, the dimensionality reduction process on the input dataset is performed using the most effective hyperparameter settings and unsupervised transfer learning strategy, based on the specific variety of the dataset. This is performed through a greedy approach with the best clustering accuracy as the target. The qualities of clusters generated through the optimal *k*-hyperparameter value are evaluated against a set of evaluation metrics i.e. the Ensemble Internal Validation Index scores. If the average Ensemble Internal Validation Index scores are low, the model auto adjusts the hyperparameter settings in order to search for a *k*-hyperparameter value that is optimal to the clustering performance.

3.5.2 Model Architecture

This stage involves the development of a model architecture that provides a blue print for the model building process. It also defines the model components in terms of functionality, structural relationships and dynamic interactions among the components (Nunamaker et al., 1990).

In this research, the high dimensional dataset is first input into the model. At the input stage, the technique is able to detect the type of the high dimensional input dataset i.e. text, image, video or audio. Based on the detected data type, the self-adapting autoencoder engine in the new technique selects the most effective hyperparameter settings and unsupervised transfer learning strategy for that particular data type. The Figure 3.4 shows the architecture of the newly developed *k*-hyperparameter tuning technique using an Ensemble Based Technique of a Self-adapting Autoencoder and Internal Validation Indexes (Gikera et al., 2023b).

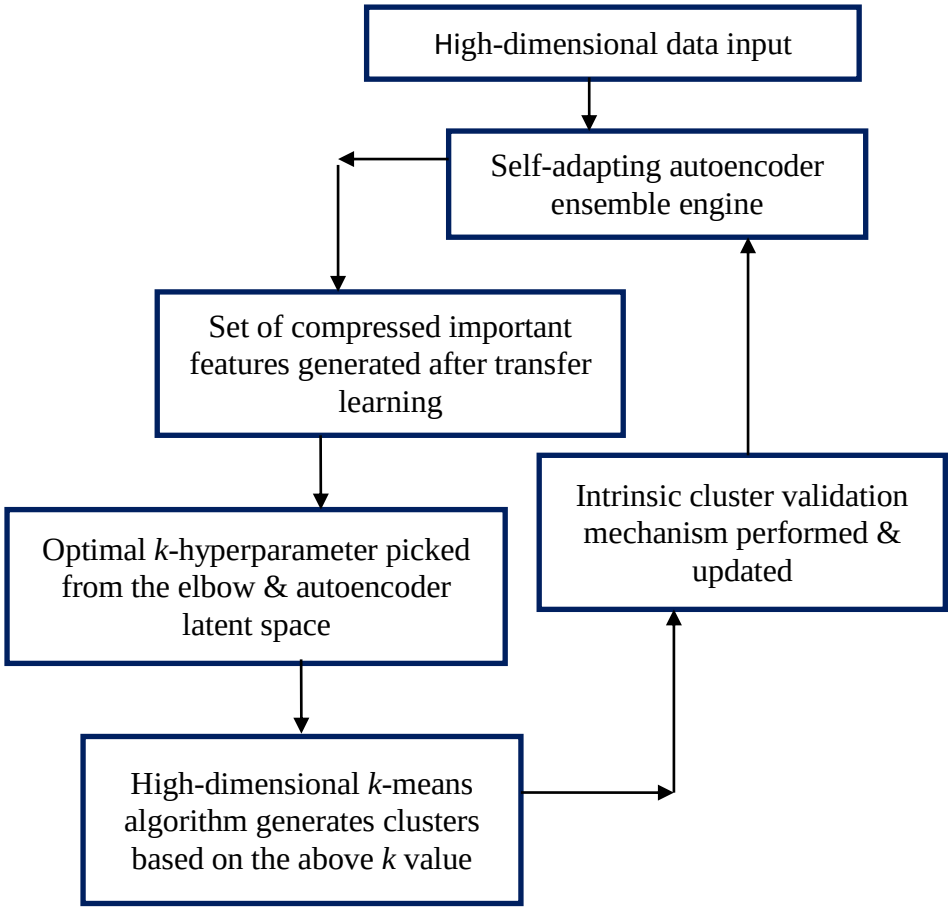


Figure 3.4: Architecture of the Ensemble based *k*-hyperparameter tuning technique on high-dimensional space using a self-adapting autoencoder and internal validation indexes

The intrinsic cluster validation mechanism validates the quality of clusters by comparing a set of average internal index scores using the ensemble's voting scheme. The *k*-value with the best score is picked as the correct optimal *k*-hyperparameter tuning technique for that particular dataset. However, if an optimal *k*-hyperparameter value is picked as the correct one, and the quality of clusters still seem low, the technique triggers the autoencoder to perform further adjustments on both the hyperparameter settings and unsupervised transfer learning so as to tune the *k*-hyperparameter value to a more optimal value. The start of the self-adapting autoencoder ensemble engine to the end of the stage where the *k*-hyperparameter value is generated forms the top level model, while the actual clustering process by the *k*-means algorithm forms the baseline model. The Pseudocode for the newly developed *k*-hyperparameter tuning technique was formulated as follows:

1. *k*HT ← [HD] # Input high-dimensional dataset.
2. HDNature ← HD #Check for the nature of the high-dimensional dataset.
3. Adjust the autoencoder architectural and training related hyperparameter settings in line with the nature of the dataset
4. A ← [E] # Initialize the autoencoder.
5. If HD = = Text of dimensionality X then
6. Perform transfer learning, on the autoencoder based on the most suitable dimensionality reduction method for the text dataset, based on the experimental results, and initialize weights.
7. A ← [K*,HSTX] #Activate further set of hyperparameter settings for the text dataset of X dimensions (K)
8. HD ← LD # Map the high-dimensional features into low dimensional
9. Else if

10. If HD = = Image of dimensionality X then
11. Perform transfer learning, on the autoencoder based on the most suitable dimensionality reduction method for image dataset, based on the experimental results, and initialize weights.
12. A ← [K*,HSIX] #Activate further set of hyperparameter settings for the image dataset of X dimensions (K)
13. HD ← LD # Map the high-dimensional features into low dimensional
14. Else if
15. If HD = = Audio of dimensionality X then
16. Perform transfer learning, on the autoencoder based on the most suitable dimensionality reduction method for audio dataset, based on the experimental results, and initialize weights.
17. A ← [K*,HSAX] #Activate further set of hyperparameter settings for the Audio dataset of X dimensions (K)
18. HD ← LD # Map the high-dimensional features into low dimensional
19. Else if
20. If HD = = Videos of dimensionality X then
21. Perform transfer learning, on the autoencoder based on the most suitable dimensionality reduction method for video dataset, based on the experimental results, and initialize weights.
22. A ← [K*,HSVX] #Activate further set of hyperparameter settings for the Video dataset of X dimensions (K)
23. HD ← LD # Map the high-dimensional features into low dimensional
24. End if
25. End if

26. End if
27. End if
28. C ← [ ] # Initialize the elbow curve as empty
29. C* ← [K-values, inertia] # Generate the elbow curve based on LD
30. If
31. K*←[k1] #optimal *k* is f′′(k)(1+ f′(k)2)-3 f′′(k)2 f′′((k)2 f′(k) ≈0 at elbow
32. AND
33. K* ← [k2]optimal *k* aligning to best scores of the validation indexes
34. AND
35. K* ← [k3] optimal *k* aligning to the visualization of the autoencoder's latent space
36. Then
37. CI,DI,DB,SI ← [ ] #Initialize all internal validation indexes as empty
38. [Clusters] ← K-means(HD,K*) # Do *k*-means clustering based on K*
39. [SI, CI,DBn,DI] ← Clusters # Compute internal index scores for clusters
40. If SI,CI,DBn, DI ≠ SI*,CI*,DBn*,DI* # Check if the index scores are best
41. Then
42. Go to step 5
43. Else
44. EI $[_{SI,\ CI,DBn,DI}]$ ← EI # Compute the average ensemble index based on the four internal indexes
45. End

3.5.3 Model Design

This stage involves analyzing and designing the model. This involves understanding the studied domain, the application of relevant scientific and technical knowledge, the creation of various alternatives and the synthesis and evaluation of proposed alternative solutions (Nunamaker et al., 1990). In this research, the model design mainly focuses on the setting up of detailed specifications for the model, including its architecture and its components. In the model design, the motivation is to create an effective *k*-hyperparameter tuning technique in high-dimensional space clustering, in a variety of high dimensional datasets. The model design process for this technique mainly focuses on the architecture and the interaction of different components of the self-adapting autoencoder ensemble engine, including the improved autoencoder. Firstly, the autoencoder's weights are first initialized through unsupervised transfer learning strategy during the pre-training process, using the respective best performing dimensionality reduction methods, based on the experimental results. The evaluation and the fine tuning process continues until the optimal *k*-hyperparameter value is obtained. The standard PCA on both the Tox21 and BBC news datasets are used for the pre-training of the autoencoder on text datasets. The UMAP learn on both the NSCLC Radiogenomics dataset and the ORL face dataset are used for the pre-training of the autoencoder on the image datasets. The Multi-Core t-SNE on both the Stethoscope Heart Sounds and the Coswara datasets are used for the pre-training of the autoencoder on the audio datasets. The Spatial-Temporal UMAP on the MPII Cooking Activities dataset is used for the pre-training of the autoencoder on video datasets.

Unsupervised transfer learning on the autoencoder using the standard PCA as a feature extractor, BBC News dataset as the source dataset and the Reuters dataset as the target dataset involves a number of steps.

Firstly, both the BBC News and Reuters-21578 datasets are pre-processed to ensure that they have a consistent format and structure. This mainly involves text cleaning, tokenization and potentially the removal of stop words. The autoencoder is trained on the BBC News dataset. The encoder part of the autoencoder learns to encode meaningful representations of text data in an unsupervised manner. The standard PCA is applied to the encoded features extracted from the BBC News dataset.

The standard PCA reduces the dimensionality of the features while preserving the most important variance. The Term Frequency-Inverse Document Frequency (TF-IDF) is used to convert text data into a numerical format before applying the standard PCA. The Reuters dataset is pre-processed and structured in a format that is compatible with the autoencoder's input, making it ready for unsupervised learning process. The encoder part of the pre-trained autoencoder is loaded and fine-tuned on the Reuters-21578 target dataset using the PCA-transformed features from the BBC news source dataset, in an unsupervised manner. Based on the evaluation metrics, further hyperparameter tuning is performed in an iterative manner in order to optimize the model's performance on the Reuters-21578 dataset.

Unsupervised transfer learning on the autoencoder using the standard PCA as a feature extractor, Tox21 as a source dataset and the Tox 171 as a target dataset involves a number of steps. First, the two datasets are consistently pre-processed including format and structure. This includes the imputation of the missing values as well as performing both the scaling and the normalization processes. The autoencoder is then pre-trained on the Tox21 dataset. The encoder part of the autoencoder learns to extract meaningful features from the Tox21 toxicology data in an unsupervised manner.

The standard PCA is then applied to the encoded features extracted from the Tox 21 toxicology dataset. The standard PCA reduces the dimensionality of the features while preserving the most important variance. The TOX 171 dataset is then pre-processed and structured in a format that is compatible with the autoencoder's input in order to ensure that it is ready for unsupervised learning. The encoder part of the pre-trained autoencoder is loaded and fine-tuned on the TOX 171 dataset, using the PCA-transformed features from the Tox21 dataset. Based on the evaluation metrics, further hyperparameter tuning is performed in an iterative manner in order to optimize the model's performance on the TOX 171 target dataset.

Unsupervised transfer learning on the autoencoder using the UMAP-learn as a feature extractor, NSCLC Radiogenomics dataset as the source dataset and the Lung Cancer dataset as the target dataset involves a number of steps. Firstly, both the NSCLC Radiogenomics and Lung Cancer datasets are pre-processed in order to ensure that they have a consistent format and structure. This includes resizing images, normalizing pixel values, and dealing with any present class imbalances. The autoencoder is trained on the NSCLC Radiogenomics dataset. The encoder part of the autoencoder learns to extract features from the CT scan images in an unsupervised manner.

The UMAP-learn is applied to the encoded features extracted from the NSCLC Radiogenomics source dataset and reduced the dimensionality of features while preserving their relevant characteristics. The Lung Cancer dataset is pre-processed and structured in a format that is compatible with the autoencoder's input in order to ensure that it is ready for unsupervised learning. The encoder part of the pre-trained autoencoder is loaded and fine-tuned on the Lung Cancer dataset, using the UMAP-learn features extracted from the NSCLC Radiogenomics dataset for weight initialization.

Based on the evaluation metrics, further hyperparameter tuning is performed in an iterative manner in order to optimize the model's performance on the Lung Cancer target dataset.

Unsupervised transfer learning on the autoencoder using the UMAP-learn as a feature extractor, ORL face dataset as the source dataset and the Yale image dataset as the target dataset involves a number of steps. Firstly, both the ORL face dataset and Yale image dataset are pre-processed in order to ensure that they have a consistent format and structure.

This includes resizing images, normalizing pixel values, and dealing with any present class imbalances. The autoencoder is trained on the ORL face dataset. The encoder part of the autoencoder learns to extract features from the images in an unsupervised manner. The UMAP-learn is applied to the encoded features extracted from the ORL face dataset and reduced the dimensionality of features while preserving their relevant characteristics. The Yale image dataset is pre-processed and structured in a format that is compatible with the autoencoder's input in order to ensure that it is ready for unsupervised learning. The encoder part of the pre-trained autoencoder is loaded and fine-tuned on the Yale Image dataset, using the UMAP-learn features extracted from the ORL face dataset for initialization of weights. Based on the evaluation metrics, further hyperparameter tuning is performed in an iterative manner in order to optimize the model's performance on the Yale Image target dataset.

Unsupervised transfer learning on the autoencoder using the Multi-core t-SNE as the feature extractor, Coswara dataset as the source dataset and the COVID-19 Cough Sounds dataset as the target dataset involves a number of steps. Firstly, both the Coswara dataset and COVID-19 Cough Sounds target datasets are pre-processed to ensure they have a consistent format and structure. This includes the audio feature extraction, normalization, and cleaning.

The autoencoder is then trained on the Coswara dataset. The encoder part of the autoencoder learns to encode meaningful features from the Coswara audio data in an unsupervised manner. The Multi-core t-SNE is applied to the encoded features extracted from the Coswara dataset and reduces the dimensionality of features, while preserving their structure. The COVID-19 Cough Sounds dataset is pre-processed and structured in a format that is compatible with the autoencoder's input in order to ensure that it is prepared for unsupervised learning. The encoder part of the pretrained autoencoder is loaded and fine-tuned on the COVID-19 Cough Sounds dataset using the Multi-core t-SNE-transformed features from the Coswara dataset. Based on the evaluation metrics, further hyperparameter tuning is performed in an iterative manner in order to optimize the model's performance on the COVID-19 Cough Sounds target dataset.

Unsupervised transfer learning on the autoencoder using the Multi-core t-SNE as the feature extractor, the Stethoscope Heart Sounds dataset as the source dataset and the HeartBeat Sounds dataset as the target dataset involves a number of steps. Firstly, both the Stethoscope Heart Sound source dataset and Heart Beat Sounds target datasets are pre-processed to ensure they have a consistent format and structure. This includes the audio feature extraction, normalization, and cleaning. The autoencoder is then trained on the Stethoscope heart Sounds dataset. The encoder part of the autoencoder learns to encode meaningful features from the Stethoscope Heart Sounds audio data in an unsupervised manner. The Multi-core t-SNE is applied to the encoded features extracted from the Stethoscope Heart Sounds dataset and reduces the dimensionality of features while preserving their structure. The Heart Beat Sounds dataset is pre-processed and structured in a format that is compatible with the autoencoder's input in order to ensure that it is prepared for unsupervised learning.

The encoder part of the pretrained autoencoder is loaded and fine-tuned on the Heart Beat Sounds dataset using the Multi-core t-SNE-transformed features from the Stethoscope Heart Sounds dataset. Based on the evaluation metrics, further hyperparameter tuning is performed in an iterative manner in order to optimize the model's performance on the Heart Beat sounds target dataset.

Unsupervised transfer learning on the autoencoder using the Spatial-Temporal UMAP as a feature extractor, the MPII cooking activities dataset as the source dataset and the YouCook dataset as the target dataset comprises of a number of steps. Firstly, both the MPII Cooking Activities dataset and the YouCook datasets are pre-processed in order to ensure that they have a consistent format. This includes the extraction of the spatial-temporal features using the histogram of oriented gradients. The autoencoder is then pre-trained on the MPII Cooking Activities dataset. The encoder part of the autoencoder learns to extract spatial-temporal features from the dataset. The Spatial-Temporal UMAP is applied to the encoded features from the MPII Cooking Activities dataset and aims at reducing the dimensionality of the features while preserving their spatial-temporal characteristics. The YouCook dataset is pre-processed to make it compatible with the autoencoder's input format in order to ensure that it is structured in a way that allows the extraction of the spatial-temporal features in an unsupervised manner. The pre-trained autoencoder's encoder part is loaded and fine-tuned on the YouCook dataset using the spatial-temporal features extracted by the Spatial-Temporal UMAP method. Based on the evaluation metrics, further hyperparameter tuning is performed in an iterative manner in order to optimize the model's performance on the YouCook target dataset.

The engine's capability to automatically adjust the hyperparameter settings in a greedy manner, and depending on the nature of the high-dimensional dataset input, involves changes in a number of important architectural and training-related hyperparameter settings, performing further evaluation on the generated clusters and fine tuning processes. These hyperparameter settings includes: the number of hidden layers, number of nodes, choice on the activation function, choice on the loss function, learning rate and optimization, regularization, batch size, epochs, initializer, learning rate schedule, dimensionality's bottleneck dimensions, batch normalization and the early stopping. The improved autoencoder in the new *k*-hyperparameter tuning technique aims at combining the best set of the hyperparameter settings based on the nature of the high-dimensional input dataset. After the high-dimensional dataset input is input into the improved autoencoder of the new technique, hyperparameter tuning is performed in order to generate the optimal hyperparameter settings based on the high dimensional input dataset. During the tuning process, both the maximum number of trials and the number of executions per each and every trial are set out.

Once the best combinations of the hyperparameter settings are identified, and the effective unsupervised transfer learning strategy is adopted, the new technique projects the high-dimensional dataset into the latent space of low-dimensions. This information is part of the decision that the technique uses to identify the correct *k*-hyperparameter value for a specific high dimensional dataset. The Ensemble Internal Validation Index scores from the clustering result determines whether the autoencoder is to perform further adjustments on the hyperparameter settings or whether the algorithm will stop i.e. if the internal validation index scores were relatively the best.

In clustering the different high-dimensional datasets, a thoughtful configuration of an architecture that can effectively learn and represent the underlying structure of a particular high-dimensional dataset is critical to the success of the development of the new *k*-hyperparameter tuning technique. The thoughtful configuration in the design also incorporates the use of experimentation process in order to find the best architecture and hyperparameter settings for a specific variety of a high-dimensional dataset. The settings incorporate sparsity constraints, regularization mechanisms, loss functions, activations functions, learning rates, initialization rates e.t.c. The new technique is a greedy algorithm that makes choices based on the best options available, with the clustering quality being the target. After the encoding process, the decoder is discarded and the learning representation is optimized on the autoencoder's latent space (Gikera et al., 2023b).

In building the design for handling text datasets, the features are standardized in order to ensure that they have similar scales, as this improved the training and the clustering process during the data pre-processing. The data pre-processing also entails handling any missing values appropriately, either by imputing them or encoding them as a separate category. The architecture consists of multiple encoding and decoding layers using fully connected parts for both the encoder and the decoder in order to capture the intricate patterns in the dataset. Drop out, experimented against *L1* and *L2* regularization techniques are used in order to avoid over-fitting. The architectural design is also experimented with three activation functions i.e. Rectified Linear Unit (ReLU), Leaky ReLU, and the Scaled Exponential Linear Units (SELU) in order to fine tune the model to effectively handle such datasets. The Mean Squared Error (MSE) is used for the reconstruction loss because of its popularity in such applications.

Experiments using different regularization techniques are done in order to incorporate a regularization term that encourages the learned embeddings to be suitable for clustering the text datasets. Experimentation with different latent space dimensions is done in order to identify the latent space size that retains the most relevant information while discarding noise and irrelevant features from the specific text dataset. The autoencoder is trained to minimize the reconstruction loss while regularizing the embeddings. In order to improve on the training performance, learning rate scheduling and early stopping techniques are used. Although the Adam and the AdaGrad are used as the adaptive learning rate strategies to dynamically adjust the learning rates based on the gradients, the model's performance is validated with different learning rates on a validation set, monitoring both the training and the validation loss curves. Based on this, the choice on the best learning rate is based on the one that provides the best trade-off between the convergence speed and the performance. Although the mini-batch gradient descent strategy is adopted to balance the advantages of the stochastic gradient descent and the full-batch gradient descent, experiments are performed in order to select the optimal batch size that demonstrates a good trade-off between the convergence speed, memory efficiency, and the generalization performance. In order to determine the optimal number of epochs, experiments are conducted with different epoch values, observing the trade-off between the training time and the model's performance. Depending on the complexity of the dataset, the changes on the architectural and the training-related hyperparameter settings are experimented against the internal validation index scores. They are also based on the experimentation of the different layer configurations, loss functions and regularization techniques (Gikera et al., 2023b).

In building the design for handling image datasets, the image data is normalized to a common scale during the data pre-processing stage. At the same time, the images are flattened and reshaped into a vector suitable for feeding into the autoencoder. The architecture's depth is set to strike a balance between capturing more complex features from an image dataset while at the same time minding the computational constraints. For the encoder layers, the numbers of neurons starts with fewer neurons and are gradually increased in deeper layers in order to capture the hierarchical features. Convolutional layers are used for both the encoder and the decoder due to their ability to capture the spatial hierarchies. The design starts with a sequence of convolutional layers followed by pooling layers to capture image features effectively. Towards the bottleneck layer, the dimensionality is reduced gradually. Experimentation with different latent space dimensions is done in order to identify the latent space size that retains the most relevant information while discarding noise and irrelevant features from the specific dataset. Although the Adam and the AdaGrad are used as the adaptive learning rate strategies to dynamically adjust the learning rates based on the gradients, finding the optimal learning rate involves experimentation and validation, ensuring that the model converges effectively without overfitting or getting stuck in the local minima. The model is tested with various batch sizes of 32, 64, 128 e.t.c. and the impact assessed based on the convergence speed, stability, and reconstruction quality using validation datasets. The hyperparameter settings are experimented with Rectified Linear Unit (RELU), Leaky RELU and Swish activation functions. The batch normalization of layers is incorporated in order to stabilize training and improve convergence.

The Kullback-Leibler (KL) Divergence is adopted as the loss function as it encourages the encoder to generate embeddings that are suitable for clustering. A regularization term, to enforce a compact clustering-friendly space, is also added.

The combined reconstruction and clustering loss are minimized whereas learning rate and early stopping are used to improve the training stability and convergence. During training, both the loss and the internal validation index scores are monitored to avoid over-fitting (Gikera et al., 2023b).

In building the design for handling audio datasets, the motivation is to design an architecture that can effectively capture the unique patterns and features within audio data, enabling better clustering performance. During the pre-processing stage, audio files are converted into spectrograms that capture the frequency and the time-domain characteristics of the audio signals. The spectrograms are also normalized to a common scale. In order to increase the diversity of the training data, data augmentation techniques are incorporated. The architectural design is tailored for audio data. Convolutional layers ire used in order to capture local patterns and hierarchical features in the spectrogram data. Pooling convolutions are included in order to reduce the spatial dimensions while retaining important features. Experiments with varying layer sizes are conducted in order to identify the optimal layer size that efficiently captures the complex audio features from the audio dataset. The experiment process using the skip connections and the U-Net architectures are done in order to enhance feature preservation. Mean Squared Error (MSE) between the input spectrogram and the reconstructed spectrogram is used as the reconstruction loss. Kullback-Leibler Divergence is also used as the loss function as it encourages the encoder to produce embeddings that are friendly to clustering (Gikera et al., 2023b). Although the RMSprop is used as the adaptive learning rate strategy to dynamically adjust the learning rates based on the gradients, the model's performance is validated with different learning rates on a validation set, monitoring both the training and the validation loss curves.

The appropriate number of epochs is determined through monitoring the validation loss where the training process is stopped when the validation loss stabilized or when it starts increasing in order to prevent overfitting. The training strategy combines loss, focusing on minimizing the reconstruction error and encouraging clustering-friendly embeddings. During training, using audio-specific data augmentation is considered as the best option as it increases the robustness of the model. The fine tuning strategy involves the experimentation process with various audio preprocessing techniques, architectures, and loss combinations. The specific audio pre-processing steps incorporate the application of the Mel-frequency Cepstral coefficients (MFCCs) in order to enhance the quality of the input data (Gikera et al., 2023b).

In building the design for handling the video datasets, the aim is to design an architecture that could effectively capture the temporal and spatial features within video datasets, enabling better capturing of the motion information and clustering performance. The video frames are converted into suitable representations such as image frames, optical flow, or spatiotemporal volumes. The data pre-processing involves normalization of the pixel values to a common scale. Both the random cropping and flipping are used as the data augmentation techniques in order to increase the diversity of the training data. A three-dimension (3D) architecture consisting of a convolutional autoencoder is used, aimed at capturing both the spatial and temporal features of video data. The convolutional layers are incorporated for spatial processing while the recurrent layers are incorporated for temporal processing. Kullback-Leibler Divergence is used as the loss function as it encourages the encoder to produce embeddings that are friendly to clustering. The training strategy incorporates frame skipping and temporal jittering techniques in order to enhance temporal diversity.

Although the RMSprop is used as the adaptive learning rate strategy to dynamically adjust the learning rates based on the gradients, the model's performance is validated with different learning rates on a validation set, monitoring both the training and the validation loss curves. Fine tuning is accomplished through iterations on the different hyperparameters as well as through an experimentation process using different video pre-processing techniques. It also involves an evaluation process against the internal validation scores on the clustering results. In order to enhance the quality of input data, optical flow estimation technique is incorporated. After training the improved autoencoder, a reconstruction error threshold is set in order to identify the noisy data and the outliers. Data points with reconstruction errors above this threshold are considered outliers or potentially noisy data points (Gikera et al., 2023b).

3.5.4 Model Prototyping

This stage involves the development of a prototype model in order to test the new *k*-hyperparameter tuning technique in a real world setting. This prototype can further be developed into a final product and implemented to a fully functional system (Nunamaker et al., 1990). In this research, the rapid application development methodology is adopted to create the model's prototype in order to assess the technical feasibility of the newly developed *k*-hyperparameter tuning technique. This prototype also provides an avenue for further improvements on the technique, especially on its design and implementation processes (Gikera et al., 2023b).

3.5.5 Evaluation Experiments

This stage involves the evaluation process through experiments. This helps the researchers to observe the functionality, performance and the impacts of the model on individuals or groups (Nunamaker et al., 1990). In this research, different experiments are used in order to improve on the performance of the new *k*-hyperparameter tuning technique. Firstly, experiments are done on the performance of the different dimensionality reduction methods in a variety of high dimensional datasets. The best dimensionality reduction method, for a specific variety of high dimensional dataset is used to perform the unsupervised transfer learning process on the new technique's autoencoder. At the same time, various experiments on the new technique, based on the different hyperparameter settings, are performed in order to identify the most suitable set of hyperparameter settings for a specific variety of high dimensional dataset. Lastly, the new technique is fed with a variety of high-dimensional datasets in order to evaluate its effectiveness against the existing techniques. After this, the performance results are recorded in a table of results followed by an analysis using the R software. Figure 3.5 gives an illustration of the five stages together with a description on how these stages are adopted to fit the development of the new *k*-hyperparameter tuning technique. To this end, the model development methodology in this chapter addresses the first four stages in details while the fifth stage is addressed in the next chapter. Besides adopting domain similarity, the cosine similarity i.e. a measure of how similar two datasets are based on their patterns, helps in checking of the feature vectors in the selection process of the source datasets for the respective target datasets in the new *k*-HT technique (Park et al., 2020). In high-dimensional pretraining, cosine similarity helps select a source dataset closely aligned with the target dataset by measuring the angle between their feature vectors, ensuring semantic alignment while disregarding magnitude. This alignment improves knowledge transfer efficiency, as a similar source dataset enables the model to carry over more relevant information, reducing

fine-tuning requirements and the risks of negative transfer. Additionally, cosine similarity ensures that dimensionality reduction techniques maintain the essential structure of high-dimensional data, aiding in model generalization and minimizing overfitting risks, ultimately leading to a more focused, resource-efficient pretraining process. For instance, the MPII cooking activities dataset served as the source dataset in the pre-training process when analyzing the YouCook dataset as the target high-dimensional dataset, both of which contain sequences of features related to the visual and temporal aspects of cooking food. The NumPy and SciPy libraries are used in the development of the submodule used for these computations. The formula for the cosine similarity check is formulated as follows:

$$Cos(A, B) = \frac{A.B}{\|A\|\|B\|} \quad (3.3)$$

Where A represents the features extracted from the source dataset while B from target dataset.

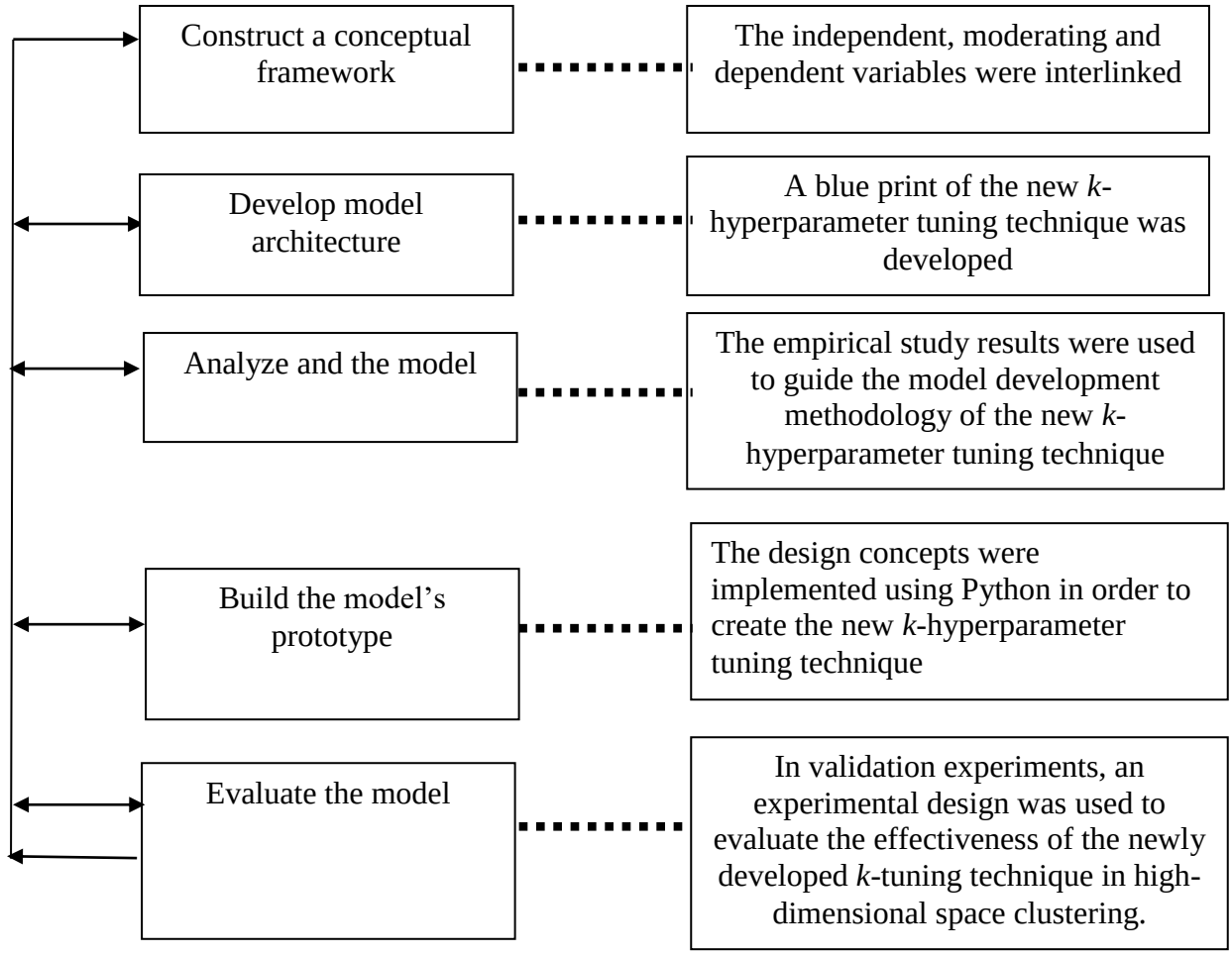


Figure 3.5: Research methodology for system development

3.6 Research Ethics

In this research, the guidelines associated with authorship, copyright and patenting policies and data sharing policies are closely adhered to. During the collection of the primary data from the YouTube Vlogs, the YouTube terms of service are observed before mining data using the Python script. The YouTube community guidelines, copyright and data privacy policies are adhered to. Moreover, every part of this thesis that is not part of the author's work is referenced in the American Psychological Association in line with Cooper (2012).

CHAPTER FOUR: RESULTS

4.1 Introduction

In this chapter, the obtained results are first analyzed, followed by the discussion ahead of drawing conclusion. As discussed in chapter one, the main objective of this empirical study is to perform an empirical analysis on the objectively selected existing *k*-hyperparameter tuning techniques. After this, the experimental data is analyzed and the results used as the basis of developing a new such technique. After the presentation of the analyzed results from the empirical analysis, the effectiveness of the new technique is evaluated against the existing ones in a variety of datasets. The results are presented in the order in which the experiments are carried out. Based on the research objectives, the findings are achieved as follows:

As discussed in chapter one, the third objective is to perform an empirical analysis of the objectively selected *k*-hyperparameter tuning techniques. To achieve this, experiments are set up to investigate their performance in a variety of datasets and dimensionality reduction methods. This is based on an empirical research design method and the conceptual framework. Section 3.4.2 provides details of how the empirical experiments are set up. The analyzed results from the experimental data, through invaluable insights, are used as a guide to the model development methodology of the new technique. Section 4.2 presents the analysis results of the experimental data from the empirical analysis. As discussed in chapter one, the fourth objective is to develop an improved *k*-hyperparameter tuning technique in high-dimensional space clustering, based on the results of the empirical analysis. Section 4.3 presents the key findings from the empirical study and how the experimental results guided the model development methodology on the new technique. The fifth objective was to evaluate on the effectiveness of the newly developed technique using high dimensional benchmark datasets.

Section 4.4.1, 4.4.2 and 4.4.3 presents the analysis results on the Kruskal-Wallis H statistic and the ANOVA in the evaluation of the effectiveness and flexibility of the new technique. This comparison is done against the existing techniques in a variety of datasets. Section 4.5 presents further comparison of the cluster visualizations in a variety of datasets, using both the standard and the improved autoencoder in the newly developed *k*-hyperparameter tuning technique.

4.2 Experimental Results

Empirical algorithmics involves the empirical analysis of algorithms to understand their behavior, performance, and efficiency (Prefect & Prosser, 2014). It focuses on experimentally evaluating algorithms using real-world data rather than solely relying on theoretical analysis (McGeoch, 2012). Empirical algorithmics involves conducting experiments on algorithms using various datasets to evaluate their performance under different conditions, comparing their performance on standardized datasets to establish benchmarks. Through this, it optimizes the algorithmic parameters to enhance performance on specific datasets (Prefect & Prosser, 2014). Empirical algorithmics bridges the gap between theoretical analysis and real-world applicability by providing empirical evidence of how algorithms behave in practice (McGeoch, 2012). Empirical analysis in machine learning is conducted in order to provide a reference for models selection based on their empirical applicability on a real world dataset (Zhang et al., 2017). In light of this, the experiments in this research are conducted in order to answer the first research question formulated in chapter 1 and aimed at investigating on the following:

1. The performance and interactions of the *k*-hyperparameter tuning techniques in a variety of datasets.
2. The performance and interactions of the *k*-hyperparameter tuning techniques in a variety of dimensionality reduction methods.
3. The performance of the newly developed *k*-hyperparameter tuning technique and the existing ones, in a variety of datasets.

Table 4.1 is a summary of the set of independent variables, moderating variables, interactions as well as the data analysis method used in the analysis of the different sets of experiments. The scores for the Silhouette index, Dunn index, Calinski Harabsz index, Davies Bouldin index as well as the ones for the cluster building times are the dependent variables. The four internal validation indexes are first normalized before being reported as a single performance score referred to as an Average Index Performance Score (AIPS).

This score is based on 15 runs and 100 iterations for each experiment. Equation 3.1 in section 3.4.7 is used in the normalization process. During normalization, the range is first identified followed by the identification of both the minimum and maximum values. For each data point x, the minimum value is subtracted from it. This formulates the denominator. The difference between the maximum and the minimum values is formulated as the denominator. The numerator divided by the denominator results into the normalized score for a single index.

Table 4.1: Summary of the set of variables, interactions and the data analysis method applied in the different set of experiments.

Independent variable	Moderating variable	Interaction	Data analysis method used
K-hyperparameter tuning techniques	a) A variety of Data types	None	2-way ANOVA
K-hyperparameter tuning techniques	a) A variety of data dimensionality reduction methods	None	2-way ANOVA
K-hyperparameter tuning techniques	a) Datasets of different dimensionality levels	None	2-way ANOVA
None	a) A variety of Data types b) Datasets of different dimensionality levels	Data type *and* dimensionality level	2-way ANOVA
None	a) A variety of Data types b) Different data dimensionality reduction methods	Data type *and* data dimensionality reduction method	2-way ANOVA

None	a) Datasets of different dimensionality levels b) Different data dimensionality reduction methods	Dimensionality level *and* data dimensionality reduction method	2-way ANOVA
K-hyperparameter tuning techniques	a) A variety of Data types b) Datasets of different dimensionality levels	*k*-hyperparameter tuning technique *and* Data type *and* dimensionality level	3-way ANOVA
K-hyperparameter tuning techniques	a) A variety of Data types b) Different data dimensionality reduction methods	*k*-hyperparameter tuning technique *and* Data type *and* data dimensionality reduction method	3-way ANOVA
K-hyperparameter tuning techniques	a) Datasets of different dimensionality levels b) Different data dimensionality reduction methods	*k*-hyperparameter tuning technique *and* Dimensionality level *and* data dimensionality reduction method	3-way ANOVA

None	a) A variety of data types b) Datasets of different dimensionality levels c) Different data dimensionality reduction methods	Data type *and* Dimensionality level *and* data dimensionality reduction method	3-way ANOVA
K-hyperparameter tuning techniques	None	None	a) Kruskal Wallis H statistic b) 1-way ANOVA c) Visualization

The experimental data is analyzed using ANOVA. The analyzed results for the experimental data from the empirical analysis are presented in sections 4.2.1 to 4.2.19. A confidence level of 95% is used. This indicates the level of certainty in the statistical analysis where if the same study is repeated multiple times, the result would fall within the given confidence interval. The sum of squares in the ANOVA tables is a measure of the variation (dispersion) in the experimental data that is attributed to different sources. It evaluates the significance of the techniques, datasets and the dimensionality reduction methods in explaining the variability in the internal validation indexes and the cluster building times. Df (degree of freedom) are associated with different sources of variation in the internal validation indexes and the run times, and are critical for determining the significance of those sources. Mean square is a measure of the average variance of the squared differences within a specific source of variation e.g. dimensionality reduction method.

It is a key component in the F-ratio calculation in this research study, which is used to determine the statistical significance of the sources of variation. The F-statistic is the ratio of the variance between groups to the variance within groups. It assesses whether the means of the different *k*-hyperparameter tuning techniques being compared, dimensionality reduction methods, dimensionality levels and data types of datasets are significantly different. It helps determine whether there are significant differences. The *P*-value is a statistical measure that helps assess the significance of the F-statistic. The *p*-value provides the probability of observing an F-statistic as extreme as the one calculated, assuming that the null hypothesis is true. In this context, the *P*-value determines the significance of the effect of the *k*-hyperparameter tuning techniques, dimensionality reduction methods, dimensionality levels and the data types on the performance. In this context, the residuals represent the discrepancies between the actual and predicted performance based on the *k*-hyperparameter tuning techniques. The normalities of the residuals are tested using the Q-Q plots where the presence of any deviations from the diagonal lines is verified. For convenience in the presentation of the analyzed experimental results, the following key denotations are used:

1. *k*HT is used to denote the *k*-hyperparameter tuning technique. *k*HT1, *k*HT2, *k*HT3 and *k*HT4 are the four *k*-hyperparameter tuning techniques used in this empirical study. *k*HTs therefore refer to *k*-hyperparameter tuning techniques.
2. AIPS is used to denote the average index performance score. The AIPS is an average of the individualized normalized scores of the Silhouette index, Dunn index, Davies Bouldin index as well as the Calinski Harabsz index.
3. CBTs is used to denote the cluster building times, which is the algorithm's run time during the experiments.

4. DTs is used to denote the data types of the high dimensional datasets used in the empirical study. DataType aslo refer to the data types
5. RMs is used to denote the dimensionality reduction methods used in the empirical study. Reduction_Method also refer to the dimensionality reduction methods.
6. DLs is used to denote the dimensionality levels of the high dimensional datasets used in the empirical study. Order_magnitude also refer to the dimensionality levels.
7. The P3 and P4 have been used to denote the dimensionality levels of the high dimensional datasets used in the empirical study, in magnitudes of the powers of ten. P4 datasets have a multiple of 10^1 more features than the P3 datasets.
8. HDDs is used to denote the high dimensional datasets.

4.2.1 Two-way ANOVA analysis on the AIPS of the *k*HTs in different DTs

The table 4.2 shows the results of the two-way ANOVA analysis on the performance scores of the existing *k*-hyperparameter tuning techniques in different types of datasets.

Table 4.2: Two-Way ANOVA analysis on AIPS of *k*HTs in different DTs.

ANOVA - AIPS

	Sum of Squares	**df**	**Mean Square**	**F**	**p**
*k*HT Technique	0.05472	2	0.02736	3.203	**0.043**
DataType	0.25959	3	0.08653	10.132	**< .001**
*k*HT Technique ✻ DataType	0.00616	6	0.00103	0.120	0.994
Residuals	1.53726	180	0.00854		

The low *p*-value of 0.043 on the *k*HT technique is a clear indication that there are significant variations in the performance scores across the existing *k*-hyperparameter tuning techniques in different types of datasets. On the other hand, the low *p*-value of less than 0.01 on the data type indicates strong significance. Based on this evidence, a further post-hoc analysis is performed and presented in tables 4.3 and 4.4, respectively.

Table 4.3: Post-hoc test of *k*HTs in the two-way ANOVA analysis on the AIPS of the *k*HTs in different DTs.

Post Hoc Comparisons - kHT Technique

Comparison							
kHT Technique		***k*HT Technique**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
kHT 1	-	kHT 2	0.0166	0.0163	180	1.01	0.569
	-	kHT 3	0.0411	0.0163	180	2.52	**0.034**
kHT 2	-	kHT 3	0.0245	0.0163	180	1.50	0.293

From the post-hoc analysis, it is clear that the specific variations on the *k*-hyperparameter tuning techniques are due to significant differences between the techniques *k*HT1 and *k*HT3 only.

Table 4.4: Post-hoc test of DTs in the two way ANOVA analysis on the AIPS of the *k*HTs in different DTs.

Post Hoc Comparisons - DataType

Comparison							
DataType		**DataType**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
Images	-	Text	-0.01188	0.0189	180	-0.630	0.922
	-	Video	0.06375	0.0189	180	3.379	**0.005**
	-	Audio	0.07021	0.0189	180	3.722	**0.001**
Text	-	Video	0.07563	0.0189	180	4.009	**< .001**
	-	Audio	0.08208	0.0189	180	4.351	**< .001**
Video	-	Audio	0.00646	0.0189	180	0.342	0.986

From the post-hoc analysis, it is clear that the specific variations on the data types are due to significant differences between the images and video data types, images and audio data types, text and video data types as well as between text and audio data types. The *p*-value of less than 0.001 on the text and video as well as the text and audio is an indication of a strong significance.

4.2.2 Two-way ANOVA analysis on the CBTs of the kHTs in different DTs

The table 4.5 shows the results of the two-way ANOVA analysis on the performance scores of the existing *k*-hyperparameter tuning techniques on different types of datasets.

Table 4.5: Two-Way ANOVA analysis on CBTs of the *k*HTs in different DTs.

ANOVA - Time(s)

	Sum of Squares	df	Mean Square	F	p
kHT Technique	419.050	2	209.5250	2.78	0.065
DataType	5147.279	3	1715.7595	22.79	**< .001**
kHT Technique ✻ DataType	0.215	6	0.0358	4.75e-4	1.000
Residuals	13554.043	180	75.3002		

The low *p*-value of less than 0.01 on the data type is a clear indication that there are significant variations in the cluster building times across the different types of datasets. Based on this evidence, a further post-hoc analysis is performed and presented in table 4.6.

Table 4.6: Post-hoc test of DTs in the ANOVA analysis on the CBTs of *k*HTs in different DTs.

Post Hoc Comparisons - DataType

Comparison							
DataType		**DataType**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
Images	-	Text	10.208	1.77	180	5.763	**< .001**
	-	Video	-3.458	1.77	180	-1.952	0.210
	-	Audio	-0.638	1.77	180	-0.360	0.984
Text	-	Video	-13.666	1.77	180	-7.715	**< .001**
	-	Audio	-10.846	1.77	180	-6.123	**< .001**
Video	-	Audio	2.820	1.77	180	1.592	0.386

From the post-hoc analysis, it is clear that the specific variations on the cluster building times are due to significant differences between the images and text data types, text and video data types as well as the text and audio data types. The *p*-value of less than 0.001 on all of them is evidence strong significance.

4.2.3 Two-way ANOVA analysis on AIPS of the *k*HTs in different RMs

The table 4.7 shows the results of the two-way ANOVA analysis on the performance index scores of the existing *k*-hyperparameter tuning techniques in different data dimensionality reduction methods.

Table 4.7: Two-way ANOVA analysis on AIPS of the *k*HTs in different RMs.

ANOVA - AIPS

	Sum of Squares	df	Mean Square	F	p
kHT Technique	0.0547	2	0.02736	6.180	**0.003**
Reduction_Method	1.0173	7	0.14532	32.828	**< .001**
kHT Technique ✻ Reduction_Method	0.0421	14	0.00300	0.679	0.793
Residuals	0.7437	168	0.00443		

The low *p*-value of 0.003 on both the *k*-hyperparameter tuning techniques and less than 0.001 on the data dimensionality reduction methods is a clear indication that there are significant variations for both. Based on this evidence, a further post-hoc analysis is performed and presented in tables 4.8 and 4.9, respectively.

Table 4.8: Posthoc test of the *k*HTs in the ANOVA analysis on AIPS of *k*HTs in different RMs.

Post Hoc Comparisons - kHT Technique

Comparison							
kHT Technique		kHT Technique	Mean Difference	SE	df	t	p_{tukey}
kHT 1	-	kHT 2	0.0166	0.0118	168	1.41	0.339
	-	kHT 3	0.0411	0.0118	168	3.49	**0.002**
kHT 2	-	kHT 3	0.0245	0.0118	168	2.09	0.096

From the above posthoc analysis, it is clear that the specific variations on the *k*-hyperparameter tuning techniques are due to significant differences between the techniques *k*HT1 and *k*HT3 only.

Table 4.9: Post-hoc test of RMs in the ANOVA analysis on AIPS of the *k*HTs in different RMs.

Post Hoc Comparisons - Reduction_Method

Comparison							
Reduction_Method		**Reduction_Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
Autoencoder	-	FA	0.09458	0.0192	168	4.9245	**< .001**
	-	LLE	0.24500	0.0192	168	12.7559	**< .001**
	-	PCA	0.14583	0.0192	168	7.5928	**< .001**
	-	SVD	0.16167	0.0192	168	8.4172	**< .001**
	-	UMAP	0.04500	0.0192	168	2.3429	0.277
	-	kPCA	0.16292	0.0192	168	8.4823	**< .001**
	-	t-SNE	0.07583	0.0192	168	3.9483	**0.003**
FA	-	LLE	0.15042	0.0192	168	7.8315	**< .001**
	-	PCA	0.05125	0.0192	168	2.6683	0.140
	-	SVD	0.06708	0.0192	168	3.4927	**0.014**
	-	UMAP	-0.04958	0.0192	168	-2.5816	0.170
	-	kPCA	0.06833	0.0192	168	3.5578	0.011
	-	t-SNE	-0.01875	0.0192	168	-0.9762	0.977
LLE	-	PCA	-0.09917	0.0192	168	-5.1631	**< .001**
	-	SVD	-0.08333	0.0192	168	-4.3388	**< .001**
	-	UMAP	-0.20000	0.0192	168	-10.4130	**< .001**
	-	kPCA	-0.08208	0.0192	168	-4.2737	**< .001**
	-	t-SNE	-0.16917	0.0192	168	-8.8077	**< .001**
PCA	-	SVD	0.01583	0.0192	168	0.8244	0.992
	-	UMAP	-0.10083	0.0192	168	-5.2499	**< .001**
	-	kPCA	0.01708	0.0192	168	0.8894	0.987
	-	t-SNE	-0.07000	0.0192	168	-3.6446	0.008
SVD	-	UMAP	-0.11667	0.0192	168	-6.0743	**< .001**
	-	kPCA	0.00125	0.0192	168	0.0651	1.000
	-	t-SNE	-0.08583	0.0192	168	-4.4689	**< .001**
UMAP	-	kPCA	0.11792	0.0192	168	6.1393	**< .001**
	-	t-SNE	0.03083	0.0192	168	1.6053	0.747
kPCA	-	t-SNE	-0.08708	0.0192	168	-4.5340	**< .001**

From the posthoc analysis, it is clear that the specific variations on the data dimensionality reduction methods are due to significant differences between the autoencoders and FA methods, autoencoders and LLE methods, autoencoders and PCA methods, autoencoders and SVD methods, autoencoders and kPCA methods as well as between the autoencoders and the t-SNE methods. There are significant variations between the perfromances of FA and LLE, FA and SVD as well as between the FA and kPCA.

There are significant variations between LLE and PCA, LLE and SVD, LLE and UMAP, LLE and kPCA as well as between LLE and t-SNE. There are significant variations between PCA and UMAP, as well as between the PCA and t-SNE. There are significant differences between SVD and UMAP as well as between SVD and t-SNE.

4.2.4 Two-way ANOVA analysis on CBTs of the *k*HTs in different RMs

The table 4.10 shows the results of the two-way ANOVA analysis on the Cluster Building Times of the existing *k*-hyperparameter tuning techniques in different data dimensionality reduction methods.

Table 4.10: Two-way ANOVA analysis on CBTs of *k*HTs in different RMs.

ANOVA - Time(s)

	Sum of Squares	df	Mean Square	F	p
kHT Technique	419.05	2	209.525	2.40679	0.093
Reduction_Method	4074.74	7	582.106	6.68658	**< .001**
kHT Technique ✻ Reduction_Method	1.41	14	0.101	0.00116	1.000
Residuals	14625.38	168	87.056		

The low *p*-values of less than 0.001 on the data dimensionality reduction methods are a clear indication that there are statistically significant variations in the cluster building times of the existing techniques in different dimensionality reduction methods. Based on this evidence, a further post-hoc analysis is performed and presented in table 4.11.

Table 4.11: Post-hoc test of RMs in the ANOVA analysis on CBTs of the *k*HTs in different RMs.

Post Hoc Comparisons - Reduction_Method

Comparison							
Reduction_Method		Reduction_Method	Mean Difference	SE	df	t	p_{tukey}
Autoencoder	-	FA	0.3004	2.69	168	0.1115	1.000
	-	LLE	-5.4317	2.69	168	-2.0166	0.474
	-	PCA	4.4467	2.69	168	1.6509	0.719
	-	SVD	2.9908	2.69	168	1.1104	0.954

Post Hoc Comparisons - Reduction_Method

Comparison							
Reduction_Method		**Reduction_Method**	**Mean Difference**	**SE**	**df**	**t**	p_{tukey}
	-	UMAP	-4.0813	2.69	168	-1.5153	0.798
	-	kPCA	4.4933	2.69	168	1.6682	0.708
	-	t-SNE	-8.9117	2.69	168	-3.3086	**0.025**
FA	-	LLE	-5.7321	2.69	168	-2.1282	0.401
	-	PCA	4.1462	2.69	168	1.5394	0.785
	-	SVD	2.6904	2.69	168	0.9989	0.974
	-	UMAP	-4.3817	2.69	168	-1.6268	0.733
	-	kPCA	4.1929	2.69	168	1.5567	0.775
	-	t-SNE	-9.2121	2.69	168	-3.4202	**0.018**
LLE	-	PCA	9.8783	2.69	168	3.6675	**0.008**
	-	SVD	8.4225	2.69	168	3.1270	**0.042**
	-	UMAP	1.3504	2.69	168	0.5014	1.000
	-	kPCA	9.9250	2.69	168	3.6849	**0.007**
	-	t-SNE	-3.4800	2.69	168	-1.2920	0.901
PCA	-	SVD	-1.4558	2.69	168	-0.5405	0.999
	-	UMAP	-8.5279	2.69	168	-3.1662	0.038
	-	kPCA	0.0467	2.69	168	0.0173	1.000
	-	t-SNE	-13.3583	2.69	168	-4.9596	**< .001**
SVD	-	UMAP	-7.0721	2.69	168	-2.6257	0.154
	-	kPCA	1.5025	2.69	168	0.5578	0.999
	-	t-SNE	-11.9025	2.69	168	-4.4191	**< .001**
UMAP	-	kPCA	8.5746	2.69	168	3.1835	**0.036**
	-	t-SNE	-4.8304	2.69	168	-1.7934	0.625
kPCA	-	t-SNE	-13.4050	2.69	168	-4.9769	**< .001**

From the posthoc analysis, it is clear that the specific variations in cluster building times on the dimensionality reduction methods are due to significant differences between the autoencoders and t-SNE methods, FA and t-SNE, LLE and PCA, LLE and SVD, LLE and kPCA, PCA and UMAP, PCA and t-SNE, SVD and t-SNE, UMAP and kPCA as well as between kPCA and t-SNE.

4.2.5 Two-way ANOVA analysis on CBTs of *k*HTs in HDDs of varied DLs

The table 4.12 shows the results of the two-way ANOVA analysis on the cluster building times of the existing *k*-hyperparameter tuning techniques in datasets of varied dimensionality.

Table 4.12: Two-Way ANOVA analysis on CBTs of the *k*HTs in HDDs of varied DLs.

ANOVA - Time(s)

	Sum of Squares	df	Mean Square	F	p
kHT Technique	419.0500	2	209.52498	2.28	0.105
Order_magnitude	1616.1123	1	1616.11230	17.59	**< .001**
kHT Technique ✻ Order_magnitude	0.0190	2	0.00949	1.03e-4	1.000
Residuals	17085.4048	186	91.85701		

There are significant variations in the cluster building times of the different dimensionality levels of the input datasets. Based on this evidence, the post-hoc analysis on the dimensionality levels is performed presented in table 4.13.

Table 4.13: Posthoc test of DLs in the two-Way ANOVA analysis on CBTs of the *k*HTs in HDDs of varied DLs.

Post Hoc Comparisons - Order_magnitude

Comparison							
Order_magnitude		Order_magnitude	Mean Difference	SE	df	t	p_{tukey}
P3	-	P4	-5.80	1.38	186	-4.19	**< .001**

From the post hoc analysis, it is evident that the significant variations in cluster building times of the different dimensionality levels are due to differences between the datasets of dimensionality level P3 and those of the dimensionlaity level P4.

4.2.6 Two-way ANOVA analysis on the interaction between DTs and DLs of HDDs on AIPS of the *k*HTs.

The table 4.14 shows the results of the two-way ANOVA analysis on the interaction between the type of dataset and its level of dimensionality on the performance techniques.

Table 4.14: Two-Way ANOVA analysis on the interaction between DTs and DLs of HDDs on AIPS of the *k*HTs.

ANOVA - AIPS

	Sum of Squares	df	Mean Square	F	p
Data Type	0.2596	3	0.08653	10.1241	**< .001**
Order_magnitude	6.02e-4	1	6.02e-4	0.0704	0.791
Data Type ✻ Order_magnitude	0.0249	3	0.00829	0.9702	0.408
Residuals	1.5727	184	0.00855		

4.2.7 Two-way ANOVA analysis on the interaction between DTs and DLs of HDDs on CBTs of the *k*HTs.

The table 4.15 shows the results of the two-way ANOVA analysis on the interaction between the data types and thelevel of dimensionality on the cluster building times of the existing techniques.

Table 4.15: Two-Way ANOVA analysis on the interaction between DTs and DLs of HDDs on CBTs of the *k*HTs.

ANOVA - Time(s)

	Sum of Squares	df	Mean Square	F	p
DataType	5147	3	1715.8	28.77	**< .001**
Order_magnitude	1616	1	1616.1	27.10	**< .001**
DataType ✻ Order_magnitude	1386	3	462.0	7.75	**< .001**
Residuals	10971	184	59.6		

The results show that there are significant differences between the different data types and the dimensionality levels datasets, at their individual levels in regards to the cluster building times.

There are also significant interactions between the data types and the level of dimensionality on the cluster building times of the techniques. Based on this evidence, the posthoc analyses are performed and reported in tables 4.16, 4.17 and 4.18, in order to show the specific variations.

Table 4.16: Posthoc test of DTs in the two-Way ANOVA analysis on the interaction between DTs and DLs of HDDs on CBTs of the *k*HTs.

Post Hoc Comparisons - DataType

Comparison							
DataType		**DataType**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
Images	-	Text	10.208	1.58	184	6.476	**< .001**
	-	Video	-3.458	1.58	184	-2.194	0.129
	-	Audio	-0.638	1.58	184	-0.405	0.978
Text	-	Video	-13.666	1.58	184	-8.670	**< .001**
	-	Audio	-10.846	1.58	184	-6.881	**< .001**
Video	-	Audio	2.820	1.58	184	1.789	0.282

Based on the post-hoc analysis on the data types, it is evident that the specific interactions between the data types and the levels of dimensionality on the cluster building times of the existing techniques are due to significant differences between different data types. These include images and text, text and video as well as between text and audio.

Table 4.17: Posthoc test of DLs in the two-Way ANOVA analysis on the interaction between DTs and DLs of HDDs on CBTs of the *k*HTs.

Post Hoc Comparisons - Order_magnitude

Comparison							
Order_magnitude		**Order_magnitude**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
P3	-	P4	-5.80	1.11	184	-5.21	**< .001**

From the post hoc analysis, it is evident that the significant variations in cluster building times of the different dimensionality levels are due to differences between the datasets of dimensionality level P3 and those of the level P4.

Table 4.18: Posthoc test of the interaction between DTs and DLs in the two-Way ANOVA analysis on the interaction between DTs and DLs on CBTs of the *k*HTs.

Post Hoc Comparisons - DataType ✻ Order_magnitude

Comparison					Mean Difference	SE	df	t	p_{tukey}
DataType	**Order_magnitude**		**DataType**	**Order_magnitude**					
Images	P3	-	Images	P4	-3.579	2.23	184	-1.6057	0.746
		-	Text	P3	15.795	2.23	184	7.0856	**< .001**
		-	Text	P4	1.043	2.23	184	0.4677	1.000
		-	Video	P3	-5.018	2.23	184	-2.2509	0.327
		-	Video	P4	-5.478	2.23	184	-2.4576	0.221
		-	Audio	P3	-0.219	2.23	184	-0.0981	1.000
		-	Audio	P4	-4.637	2.23	184	-2.0801	0.432
	P4	-	Text	P3	19.374	2.23	184	8.6913	**< .001**
		-	Text	P4	4.622	2.23	184	2.0733	0.436
		-	Video	P3	-1.438	2.23	184	-0.6453	0.998
		-	Video	P4	-1.899	2.23	184	-0.8520	0.990
		-	Audio	P3	3.360	2.23	184	1.5075	0.803
		-	Audio	P4	-1.058	2.23	184	-0.4744	1.000
Text	P3	-	Text	P4	-14.752	2.23	184	-6.6179	**< .001**
		-	Video	P3	-20.812	2.23	184	-9.3365	**< .001**
		-	Video	P4	-21.273	2.23	184	-9.5433	**< .001**
		-	Audio	P3	-16.013	2.23	184	-7.1838	**< .001**
		-	Audio	P4	-20.431	2.23	184	-9.1657	**< .001**
	P4	-	Video	P3	-6.060	2.23	184	-2.7186	0.124
		-	Video	P4	-6.521	2.23	184	-2.9253	0.073
		-	Audio	P3	-1.261	2.23	184	-0.5658	0.999
		-	Audio	P4	-5.679	2.23	184	-2.5477	0.182
Video	P3	-	Video	P4	-0.461	2.23	184	-0.2067	1.000
		-	Audio	P3	4.799	2.23	184	2.1528	0.385
		-	Audio	P4	0.381	2.23	184	0.1708	1.000
	P4	-	Audio	P3	5.260	2.23	184	2.3595	0.268
		-	Audio	P4	0.842	2.23	184	0.3776	1.000
Audio	P3	-	Audio	P4	-4.418	2.23	184	-1.9819	0.497

Based on the post-hoc analysis on the interaction between the data types and the levels of dimensionality on the cluster building times of the existing techniques, it is evident that the differences are due to significant differences in a number of data types and their dimensionality levels. There are significant differences between the image datasets of the dimensionality level P3 and the text datasets of dimensionality level P3. There are significant differences between the image datasets of the dimensionality level P4 and the text datasets of dimensionality level P3.

There are significant differences between the text datasets of the dimensionality level P3 and the text datasets of dimensionality level P4. There are significant differences between the text datasets of the dimensionality level P3 and the video datasets of both the dimensionality level P3 and P4. There are significant differences between the text datasets of the dimensionality level P3 and the audio datasets of both the dimensionality level P3 and P4.

4.2.8 Two-way ANOVA analysis on the interaction between DTs and RMs on AIPS of the *k*HTs

The table 4.19 shows the results of the two-way ANOVA analysis on the interaction between the data types of input datasets and the data dimensionality reduction methods applied, on the performance scores of the existing *k*-hyperparameter tuning techniques.

Table 4.19: Two-Way Anova analysis on the interaction between DTs and RMs on AIPS of the *k*HTs**.**

ANOVA – AIPS

	Sum of Squares	df	Mean Square	F	p
Data Type	0.260	3	0.08653	39.90	**< .001**
Reduction Method	1.017	7	0.14532	67.01	**< .001**
Data Type ✻ Reduction Method	0.234	21	0.01114	5.14	**< .001**
Residuals	0.347	160	0.00217		

The results show that there are significant performance differences on both the data types and the dimensionality reduction methods applied, at their individual levels. The results also show that there are significant interactions between the data types and the dimensionality reduction methods, on the performance scores of the techniques. Based on this evidence, further post hoc analyses are performed and reported in tables 4.20 and 4.21. The post-hoc analysis and description of the dimensionality reduction methods are similar to the ones presented in table 4.9.

Table 4.20: Post-hoc test of DTs in the two-Way ANOVA analysis on the interaction between DTs and RMs on AIPS of the *k*HTs.

Post Hoc Comparisons - Data Type

Comparison							
Data Type		**Data Type**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
Images	-	Text	-0.01188	0.0189	180	-0.630	0.922
	-	Video	0.06375	0.0189	180	3.379	0.004
	-	Audio	0.07021	0.0189	180	3.722	**0.001**
Text	-	Video	0.07563	0.0189	180	4.009	**< .001**
	-	Audio	0.08208	0.0189	180	4.351	**< .001**
Video	-	Audio	0.00646	0.0189	180	0.342	0.986

From the post-hoc analysis, it is clear that the specific variations on the data types are due to significant differences between the images and video data types, images and audio data types, text and video data types as well as between text and audio data types. The *p*-value of less than 0.001 on the text and video as well as the text and audio is an indication of a strong significance.

Table 4.21: Post-hoc test of the interaction between DTs and RMs in the two-Way ANOVA analysis on the interaction between DTs and RMs on AIPS of the *k*HTs.

Post Hoc Comparisons - Data Type ✻ Reduction Method

Comparison									
Data Type	**Reduction Method**		**Data Type**	**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
Images	Autoencoder	-	Images	FA	0.12167	0.0269	160	4.5253	**0.005**
		-	Images	LLE	0.19667	0.0269	160	7.3149	**< .001**
		-	Images	PCA	0.07833	0.0269	160	2.9136	0.490
		-	Images	SVD	0.13833	0.0269	160	5.1452	**< .001**
		-	Images	UMAP	0.06500	0.0269	160	2.4176	0.849
		-	Images	kPCA	0.10667	0.0269	160	3.9674	**0.034**
		-	Images	t-SNE	0.09000	0.0269	160	3.3475	0.205
		-	Text	Autoencoder	0.00333	0.0269	160	0.1240	1.000
		-	Text	FA	0.09667	0.0269	160	3.5955	0.107
		-	Text	LLE	0.20333	0.0269	160	7.5628	**< .001**
		-	Text	PCA	0.06500	0.0269	160	2.4176	0.849
		-	Text	SVD	0.10000	0.0269	160	3.7194	0.075
		-	Text	UMAP	0.05500	0.0269	160	2.0457	0.975
		-	Text	kPCA	0.09667	0.0269	160	3.5955	0.107

Post Hoc Comparisons - Data Type ✻ Reduction Method

Comparison									
Data Type	**Reduction Method**		**Data Type**	**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	Text	t-SNE	0.08167	0.0269	160	3.0375	0.396
		-	Video	Autoencoder	0.02333	0.0269	160	0.8679	1.000
		-	Video	FA	0.10167	0.0269	160	3.7814	0.062
		-	Video	LLE	0.32167	0.0269	160	11.9642	**< .001**
		-	Video	PCA	0.24167	0.0269	160	8.9886	**< .001**
		-	Video	SVD	0.23167	0.0269	160	8.6167	**< .001**
		-	Video	UMAP	0.05333	0.0269	160	1.9837	0.983
		-	Video	kPCA	0.24167	0.0269	160	8.9886	**< .001**
		-	Video	t-SNE	0.09167	0.0269	160	3.4095	0.176
		-	Audio	Autoencoder	0.02833	0.0269	160	1.0538	1.000
		-	Audio	FA	0.11333	0.0269	160	4.2154	**0.015**
		-	Audio	LLE	0.31333	0.0269	160	11.6542	**< .001**
		-	Audio	PCA	0.25333	0.0269	160	9.4226	**< .001**
		-	Audio	SVD	0.23167	0.0269	160	8.6167	**< .001**
		-	Audio	UMAP	0.06167	0.0269	160	2.2937	0.908
		-	Audio	kPCA	0.26167	0.0269	160	9.7325	**< .001**
		-	Audio	t-SNE	0.09500	0.0269	160	3.5335	0.127
	FA	-	Images	LLE	0.07500	0.0269	160	2.7896	0.589
		-	Images	PCA	-0.04333	0.0269	160	-1.6118	0.999
		-	Images	SVD	0.01667	0.0269	160	0.6199	1.000
		-	Images	UMAP	-0.05667	0.0269	160	-2.1077	0.963
		-	Images	kPCA	-0.01500	0.0269	160	-0.5579	1.000
		-	Images	t-SNE	-0.03167	0.0269	160	-1.1778	1.000
		-	Text	Autoencoder	-0.11833	0.0269	160	-4.4013	0.007
		-	Text	FA	-0.02500	0.0269	160	-0.9299	1.000
		-	Text	LLE	0.08167	0.0269	160	3.0375	0.396
		-	Text	PCA	-0.05667	0.0269	160	-2.1077	0.963
		-	Text	SVD	-0.02167	0.0269	160	-0.8059	1.000
		-	Text	UMAP	-0.06667	0.0269	160	-2.4796	0.813
		-	Text	kPCA	-0.02500	0.0269	160	-0.9299	1.000
		-	Text	t-SNE	-0.04000	0.0269	160	-1.4878	1.000
		-	Video	Autoencoder	-0.09833	0.0269	160	-3.6574	0.090
		-	Video	FA	-0.02000	0.0269	160	-0.7439	1.000
		-	Video	LLE	0.20000	0.0269	160	7.4389	**< .001**
		-	Video	PCA	0.12000	0.0269	160	4.4633	**0.006**
		-	Video	SVD	0.11000	0.0269	160	4.0914	**0.023**
		-	Video	UMAP	-0.06833	0.0269	160	-2.5416	0.774
		-	Video	kPCA	0.12000	0.0269	160	4.4633	0.006
		-	Video	t-SNE	-0.03000	0.0269	160	-1.1158	1.000
		-	Audio	Autoencoder	-0.09333	0.0269	160	-3.4715	0.150
		-	Audio	FA	-0.00833	0.0269	160	-0.3100	1.000
		-	Audio	LLE	0.19167	0.0269	160	7.1289	**< .001**
		-	Audio	PCA	0.13167	0.0269	160	4.8973	**0.001**
		-	Audio	SVD	0.11000	0.0269	160	4.0914	**0.023**
		-	Audio	UMAP	-0.06000	0.0269	160	-2.2317	0.930
		-	Audio	kPCA	0.14000	0.0269	160	5.2072	**< .001**

Post Hoc Comparisons - Data Type ✻ Reduction Method

Comparison									
Data Type	**Reduction Method**		**Data Type**	**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	Audio	t-SNE	-0.02667	0.0269	160	-0.9918	1.000
	LLE	-	Images	PCA	-0.11833	0.0269	160	-4.4013	**0.007**
		-	Images	SVD	-0.05833	0.0269	160	-2.1697	0.949
		-	Images	UMAP	-0.13167	0.0269	160	-4.8973	**0.001**
		-	Images	kPCA	-0.09000	0.0269	160	-3.3475	0.205
		-	Images	t-SNE	-0.10667	0.0269	160	-3.9674	**0.034**
		-	Text	Autoencoder	-0.19333	0.0269	160	-7.1909	**< .001**
		-	Text	FA	-0.10000	0.0269	160	-3.7194	0.075
		-	Text	LLE	0.00667	0.0269	160	0.2480	1.000
		-	Text	PCA	-0.13167	0.0269	160	-4.8973	0.001
		-	Text	SVD	-0.09667	0.0269	160	-3.5955	0.107
		-	Text	UMAP	-0.14167	0.0269	160	-5.2692	**< .001**
		-	Text	kPCA	-0.10000	0.0269	160	-3.7194	0.075
		-	Text	t-SNE	-0.11500	0.0269	160	-4.2773	**0.012**
		-	Video	Autoencoder	-0.17333	0.0269	160	-6.4470	**< .001**
		-	Video	FA	-0.09500	0.0269	160	-3.5335	0.127
		-	Video	LLE	0.12500	0.0269	160	4.6493	**0.003**
		-	Video	PCA	0.04500	0.0269	160	1.6737	0.999
		-	Video	SVD	0.03500	0.0269	160	1.3018	1.000
		-	Video	UMAP	-0.14333	0.0269	160	-5.3312	**< .001**
		-	Video	kPCA	0.04500	0.0269	160	1.6737	0.999
		-	Video	t-SNE	-0.10500	0.0269	160	-3.9054	**0.042**
		-	Audio	Autoencoder	-0.16833	0.0269	160	-6.2610	**< .001**
		-	Audio	FA	-0.08333	0.0269	160	-3.0995	0.352
		-	Audio	LLE	0.11667	0.0269	160	4.3393	**0.009**
		-	Audio	PCA	0.05667	0.0269	160	2.1077	0.963
		-	Audio	SVD	0.03500	0.0269	160	1.3018	1.000
		-	Audio	UMAP	-0.13500	0.0269	160	-5.0212	**< .001**
		-	Audio	kPCA	0.06500	0.0269	160	2.4176	0.849
		-	Audio	t-SNE	-0.10167	0.0269	160	-3.7814	0.062
	PCA	-	Images	SVD	0.06000	0.0269	160	2.2317	0.930
		-	Images	UMAP	-0.01333	0.0269	160	-0.4959	1.000
		-	Images	kPCA	0.02833	0.0269	160	1.0538	1.000
		-	Images	t-SNE	0.01167	0.0269	160	0.4339	1.000
		-	Text	Autoencoder	-0.07500	0.0269	160	-2.7896	0.589
		-	Text	FA	0.01833	0.0269	160	0.6819	1.000
		-	Text	LLE	0.12500	0.0269	160	4.6493	0.003
		-	Text	PCA	-0.01333	0.0269	160	-0.4959	1.000
		-	Text	SVD	0.02167	0.0269	160	0.8059	1.000
		-	Text	UMAP	-0.02333	0.0269	160	-0.8679	1.000
		-	Text	kPCA	0.01833	0.0269	160	0.6819	1.000
		-	Text	t-SNE	0.00333	0.0269	160	0.1240	1.000
		-	Video	Autoencoder	-0.05500	0.0269	160	-2.0457	0.975
		-	Video	FA	0.02333	0.0269	160	0.8679	1.000
		-	Video	LLE	0.24333	0.0269	160	9.0506	**< .001**
		-	Video	PCA	0.16333	0.0269	160	6.0751	**< .001**

Post Hoc Comparisons - Data Type ✻ Reduction Method

Comparison									
Data Type	**Reduction Method**		**Data Type**	**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	Video	SVD	0.15333	0.0269	160	5.7031	**< .001**
		-	Video	UMAP	-0.02500	0.0269	160	-0.9299	1.000
		-	Video	kPCA	0.16333	0.0269	160	6.0751	**< .001**
		-	Video	t-SNE	0.01333	0.0269	160	0.4959	1.000
		-	Audio	Autoencoder	-0.05000	0.0269	160	-1.8597	0.993
		-	Audio	FA	0.03500	0.0269	160	1.3018	1.000
		-	Audio	LLE	0.23500	0.0269	160	8.7407	**< .001**
		-	Audio	PCA	0.17500	0.0269	160	6.5090	**< .001**
		-	Audio	SVD	0.15333	0.0269	160	5.7031	**< .001**
		-	Audio	UMAP	-0.01667	0.0269	160	-0.6199	1.000
		-	Audio	kPCA	0.18333	0.0269	160	6.8190	**< .001**
		-	Audio	t-SNE	0.01667	0.0269	160	0.6199	1.000
	SVD	-	Images	UMAP	-0.07333	0.0269	160	-2.7276	0.637
		-	Images	kPCA	-0.03167	0.0269	160	-1.1778	1.000
		-	Images	t-SNE	-0.04833	0.0269	160	-1.7977	0.996
		-	Text	Autoencoder	-0.13500	0.0269	160	-5.0212	**< .001**
		-	Text	FA	-0.04167	0.0269	160	-1.5498	1.000
		-	Text	LLE	0.06500	0.0269	160	2.4176	0.849
		-	Text	PCA	-0.07333	0.0269	160	-2.7276	0.637
		-	Text	SVD	-0.03833	0.0269	160	-1.4258	1.000
		-	Text	UMAP	-0.08333	0.0269	160	-3.0995	0.352
		-	Text	kPCA	-0.04167	0.0269	160	-1.5498	1.000
		-	Text	t-SNE	-0.05667	0.0269	160	-2.1077	0.963
		-	Video	Autoencoder	-0.11500	0.0269	160	-4.2773	**0.012**
		-	Video	FA	-0.03667	0.0269	160	-1.3638	1.000
		-	Video	LLE	0.18333	0.0269	160	6.8190	**< .001**
		-	Video	PCA	0.10333	0.0269	160	3.8434	0.051
		-	Video	SVD	0.09333	0.0269	160	3.4715	0.150
		-	Video	UMAP	-0.08500	0.0269	160	-3.1615	0.311
		-	Video	kPCA	0.10333	0.0269	160	3.8434	0.051
		-	Video	t-SNE	-0.04667	0.0269	160	-1.7357	0.998
		-	Audio	Autoencoder	-0.11000	0.0269	160	-4.0914	**0.023**
		-	Audio	FA	-0.02500	0.0269	160	-0.9299	1.000
		-	Audio	LLE	0.17500	0.0269	160	6.5090	**< .001**
		-	Audio	PCA	0.11500	0.0269	160	4.2773	0.012
		-	Audio	SVD	0.09333	0.0269	160	3.4715	0.150
		-	Audio	UMAP	-0.07667	0.0269	160	-2.8516	0.539
		-	Audio	kPCA	0.12333	0.0269	160	4.5873	**0.004**
		-	Audio	t-SNE	-0.04333	0.0269	160	-1.6118	0.999
	UMAP	-	Images	kPCA	0.04167	0.0269	160	1.5498	1.000
		-	Images	t-SNE	0.02500	0.0269	160	0.9299	1.000
		-	Text	Autoencoder	-0.06167	0.0269	160	-2.2937	0.908
		-	Text	FA	0.03167	0.0269	160	1.1778	1.000
		-	Text	LLE	0.13833	0.0269	160	5.1452	**< .001**
		-	Text	PCA	-1.73e−16	0.0269	160	-6.45e−15	1.000

Post Hoc Comparisons - Data Type ✻ Reduction Method

Comparison									
Data Type	**Reduction Method**		**Data Type**	**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	Text	SVD	0.03500	0.0269	160	1.3018	1.000
		-	Text	UMAP	-0.01000	0.0269	160	-0.3719	1.000
		-	Text	kPCA	0.03167	0.0269	160	1.1778	1.000
		-	Text	t-SNE	0.01667	0.0269	160	0.6199	1.000
		-	Video	Autoencoder	-0.04167	0.0269	160	-1.5498	1.000
		-	Video	FA	0.03667	0.0269	160	1.3638	1.000
		-	Video	LLE	0.25667	0.0269	160	9.5465	**< .001**
		-	Video	PCA	0.17667	0.0269	160	6.5710	**< .001**
		-	Video	SVD	0.16667	0.0269	160	6.1991	**< .001**
		-	Video	UMAP	-0.01167	0.0269	160	-0.4339	1.000
		-	Video	kPCA	0.17667	0.0269	160	6.5710	**< .001**
		-	Video	t-SNE	0.02667	0.0269	160	0.9918	1.000
		-	Audio	Autoencoder	-0.03667	0.0269	160	-1.3638	1.000
		-	Audio	FA	0.04833	0.0269	160	1.7977	0.996
		-	Audio	LLE	0.24833	0.0269	160	9.2366	**< .001**
		-	Audio	PCA	0.18833	0.0269	160	7.0049	**< .001**
		-	Audio	SVD	0.16667	0.0269	160	6.1991	**< .001**
		-	Audio	UMAP	-0.00333	0.0269	160	-0.1240	1.000
		-	Audio	kPCA	0.19667	0.0269	160	7.3149	**< .001**
		-	Audio	t-SNE	0.03000	0.0269	160	1.1158	1.000
	kPCA	-	Images	t-SNE	-0.01667	0.0269	160	-0.6199	1.000
		-	Text	Autoencoder	-0.10333	0.0269	160	-3.8434	0.051
		-	Text	FA	-0.01000	0.0269	160	-0.3719	1.000
		-	Text	LLE	0.09667	0.0269	160	3.5955	0.107
		-	Text	PCA	-0.04167	0.0269	160	-1.5498	1.000
		-	Text	SVD	-0.00667	0.0269	160	-0.2480	1.000
		-	Text	UMAP	-0.05167	0.0269	160	-1.9217	0.989
		-	Text	kPCA	-0.01000	0.0269	160	-0.3719	1.000
		-	Text	t-SNE	-0.02500	0.0269	160	-0.9299	1.000
		-	Video	Autoencoder	-0.08333	0.0269	160	-3.0995	0.352
		-	Video	FA	-0.00500	0.0269	160	-0.1860	1.000
		-	Video	LLE	0.21500	0.0269	160	7.9968	**< .001**
		-	Video	PCA	0.13500	0.0269	160	5.0212	**< .001**
		-	Video	SVD	0.12500	0.0269	160	4.6493	0.003
		-	Video	UMAP	-0.05333	0.0269	160	-1.9837	0.983
		-	Video	kPCA	0.13500	0.0269	160	5.0212	**< .001**
		-	Video	t-SNE	-0.01500	0.0269	160	-0.5579	1.000
		-	Audio	Autoencoder	-0.07833	0.0269	160	-2.9136	0.490
		-	Audio	FA	0.00667	0.0269	160	0.2480	1.000
		-	Audio	LLE	0.20667	0.0269	160	7.6868	**< .001**
		-	Audio	PCA	0.14667	0.0269	160	5.4552	**< .001**
		-	Audio	SVD	0.12500	0.0269	160	4.6493	0.003
		-	Audio	UMAP	-0.04500	0.0269	160	-1.6737	0.999
		-	Audio	kPCA	0.15500	0.0269	160	5.7651	**< .001**
		-	Audio	t-SNE	-0.01167	0.0269	160	-0.4339	1.000
	t-SNE	-	Text	Autoencoder	-0.08667	0.0269	160	-3.2235	0.272

Post Hoc Comparisons - Data Type ✻ Reduction Method

Comparison									
Data Type	**Reduction Method**		**Data Type**	**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	Text	FA	0.00667	0.0269	160	0.2480	1.000
		-	Text	LLE	0.11333	0.0269	160	4.2154	0.015
		-	Text	PCA	-0.02500	0.0269	160	-0.9299	1.000
		-	Text	SVD	0.01000	0.0269	160	0.3719	1.000
		-	Text	UMAP	-0.03500	0.0269	160	-1.3018	1.000
		-	Text	kPCA	0.00667	0.0269	160	0.2480	1.000
		-	Text	t-SNE	-0.00833	0.0269	160	-0.3100	1.000
		-	Video	Autoencoder	-0.06667	0.0269	160	-2.4796	0.813
		-	Video	FA	0.01167	0.0269	160	0.4339	1.000
		-	Video	LLE	0.23167	0.0269	160	8.6167	**< .001**
		-	Video	PCA	0.15167	0.0269	160	5.6411	**< .001**
		-	Video	SVD	0.14167	0.0269	160	5.2692	**< .001**
		-	Video	UMAP	-0.03667	0.0269	160	-1.3638	1.000
		-	Video	kPCA	0.15167	0.0269	160	5.6411	**< .001**
		-	Video	t-SNE	0.00167	0.0269	160	0.0620	1.000
		-	Audio	Autoencoder	-0.06167	0.0269	160	-2.2937	0.908
		-	Audio	FA	0.02333	0.0269	160	0.8679	1.000
		-	Audio	LLE	0.22333	0.0269	160	8.3067	**< .001**
		-	Audio	PCA	0.16333	0.0269	160	6.0751	**< .001**
		-	Audio	SVD	0.14167	0.0269	160	5.2692	**< .001**
		-	Audio	UMAP	-0.02833	0.0269	160	-1.0538	1.000
		-	Audio	kPCA	0.17167	0.0269	160	6.3850	**< .001**
		-	Audio	t-SNE	0.00500	0.0269	160	0.1860	1.000
Text	Autoencoder	-	Text	FA	0.09333	0.0269	160	3.4715	0.150
		-	Text	LLE	0.20000	0.0269	160	7.4389	**< .001**
		-	Text	PCA	0.06167	0.0269	160	2.2937	0.989
		-	Text	SVD	0.09667	0.0269	160	3.5955	0.107
		-	Text	UMAP	0.05167	0.0269	160	1.9217	0.908
		-	Text	kPCA	0.09333	0.0269	160	3.4715	0.150
		-	Text	t-SNE	0.07833	0.0269	160	2.9136	0.490
		-	Video	Autoencoder	0.02000	0.0269	160	0.7439	1.000
		-	Video	FA	0.09833	0.0269	160	3.6574	0.090
		-	Video	LLE	0.31833	0.0269	160	11.8402	**< .001**
		-	Video	PCA	0.23833	0.0269	160	8.8646	**< .001**
		-	Video	SVD	0.22833	0.0269	160	8.4927	**< .001**
		-	Video	UMAP	0.05000	0.0269	160	1.8597	0.993
		-	Video	kPCA	0.23833	0.0269	160	8.8646	**< .001**
		-	Video	t-SNE	0.08833	0.0269	160	3.2855	0.237
		-	Audio	Autoencoder	0.02500	0.0269	160	0.9299	1.000
		-	Audio	FA	0.11000	0.0269	160	4.0914	0.023
		-	Audio	LLE	0.31000	0.0269	160	11.5302	**< .001**
		-	Audio	PCA	0.25000	0.0269	160	9.2986	**< .001**
		-	Audio	SVD	0.22833	0.0269	160	8.4927	**< .001**
		-	Audio	UMAP	0.05833	0.0269	160	2.1697	0.949
		-	Audio	kPCA	0.25833	0.0269	160	9.6085	**< .001**
		-	Audio	t-SNE	0.09167	0.0269	160	3.4095	0.176

Post Hoc Comparisons - Data Type ✻ Reduction Method

Comparison									
Data Type	**Reduction Method**		**Data Type**	**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
	FA	-	Text	LLE	0.10667	0.0269	160	3.9674	**0.034**
		-	Text	PCA	-0.03167	0.0269	160	-1.1778	1.000
		-	Text	SVD	0.00333	0.0269	160	0.1240	1.000
		-	Text	UMAP	-0.04167	0.0269	160	-1.5498	1.000
		-	Text	kPCA	2.08e-17	0.0269	160	7.74e-16	1.000
		-	Text	t-SNE	-0.01500	0.0269	160	-0.5579	1.000
		-	Video	Autoencoder	-0.07333	0.0269	160	-2.7276	0.637
		-	Video	FA	0.00500	0.0269	160	0.1860	1.000
		-	Video	LLE	0.22500	0.0269	160	8.3687	**< .001**
		-	Video	PCA	0.14500	0.0269	160	5.3932	**< .001**
		-	Video	SVD	0.13500	0.0269	160	5.0212	**< .001**
		-	Video	UMAP	-0.04333	0.0269	160	-1.6118	0.999
		-	Video	kPCA	0.14500	0.0269	160	5.3932	**< .001**
		-	Video	t-SNE	-0.00500	0.0269	160	-0.1860	1.000
		-	Audio	Autoencoder	-0.06833	0.0269	160	-2.5416	0.774
		-	Audio	FA	0.01667	0.0269	160	0.6199	1.000
		-	Audio	LLE	0.21667	0.0269	160	8.0588	**< .001**
		-	Audio	PCA	0.15667	0.0269	160	5.8271	**< .001**
		-	Audio	SVD	0.13500	0.0269	160	5.0212	**< .001**
		-	Audio	UMAP	-0.03500	0.0269	160	-1.3018	1.000
		-	Audio	kPCA	0.16500	0.0269	160	6.1371	**< .001**
		-	Audio	t-SNE	-0.00167	0.0269	160	-0.0620	1.000
	LLE	-	Text	PCA	-0.13833	0.0269	160	-5.1452	**< .001**
		-	Text	SVD	-0.10333	0.0269	160	-3.8434	0.051
		-	Text	UMAP	-0.14833	0.0269	160	-5.5172	**< .001**
		-	Text	kPCA	-0.10667	0.0269	160	-3.9674	0.034
		-	Text	t-SNE	-0.12167	0.0269	160	-4.5253	0.005
		-	Video	Autoencoder	-0.18000	0.0269	160	-6.6950	**< .001**
		-	Video	FA	-0.10167	0.0269	160	-3.7814	0.062
		-	Video	LLE	0.11833	0.0269	160	4.4013	0.007
		-	Video	PCA	0.03833	0.0269	160	1.4258	1.000
		-	Video	SVD	0.02833	0.0269	160	1.0538	1.000
		-	Video	UMAP	-0.15000	0.0269	160	-5.5791	**< .001**
		-	Video	kPCA	0.03833	0.0269	160	1.4258	1.000
		-	Video	t-SNE	-0.11167	0.0269	160	-4.1534	**0.018**
		-	Audio	Autoencoder	-0.17500	0.0269	160	-6.5090	**< .001**
		-	Audio	FA	-0.09000	0.0269	160	-3.3475	0.205
		-	Audio	LLE	0.11000	0.0269	160	4.0914	**0.023**
		-	Audio	PCA	0.05000	0.0269	160	1.8597	0.993
		-	Audio	SVD	0.02833	0.0269	160	1.0538	1.000
		-	Audio	UMAP	-0.14167	0.0269	160	-5.2692	**< .001**
		-	Audio	kPCA	0.05833	0.0269	160	2.1697	0.949
		-	Audio	t-SNE	-0.10833	0.0269	160	-4.0294	**0.028**
	PCA	-	Text	SVD	0.03500	0.0269	160	1.3018	1.000
		-	Text	UMAP	-0.01000	0.0269	160	-0.3719	1.000
		-	Text	kPCA	0.03167	0.0269	160	1.1778	1.000

Post Hoc Comparisons - Data Type ✻ Reduction Method

Comparison									
Data Type	**Reduction Method**		**Data Type**	**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	Text	t-SNE	0.01667	0.0269	160	0.6199	1.000
		-	Video	Autoencoder	-0.04167	0.0269	160	-1.5498	1.000
		-	Video	FA	0.03667	0.0269	160	1.3638	1.000
		-	Video	LLE	0.25667	0.0269	160	9.5465	**< .001**
		-	Video	PCA	0.17667	0.0269	160	6.5710	**< .001**
		-	Video	SVD	0.16667	0.0269	160	6.1991	**< .001**
		-	Video	UMAP	-0.01167	0.0269	160	-0.4339	1.000
		-	Video	kPCA	0.17667	0.0269	160	6.5710	**< .001**
		-	Video	t-SNE	0.02667	0.0269	160	0.9918	1.000
		-	Audio	Autoencoder	-0.03667	0.0269	160	-1.3638	1.000
		-	Audio	FA	0.04833	0.0269	160	1.7977	0.996
		-	Audio	LLE	0.24833	0.0269	160	9.2366	**< .001**
		-	Audio	PCA	0.18833	0.0269	160	7.0049	**< .001**
		-	Audio	SVD	0.16667	0.0269	160	6.1991	**< .001**
		-	Audio	UMAP	-0.00333	0.0269	160	-0.1240	1.000
		-	Audio	kPCA	0.19667	0.0269	160	7.3149	**< .001**
		-	Audio	t-SNE	0.03000	0.0269	160	1.1158	1.000
	SVD	-	Text	UMAP	-0.04500	0.0269	160	-1.6737	0.999
		-	Text	kPCA	-0.00333	0.0269	160	-0.1240	1.000
		-	Text	t-SNE	-0.01833	0.0269	160	-0.6819	1.000
		-	Video	Autoencoder	-0.07667	0.0269	160	-2.8516	0.539
		-	Video	FA	0.00167	0.0269	160	0.0620	1.000
		-	Video	LLE	0.22167	0.0269	160	8.2447	**< .001**
		-	Video	PCA	0.14167	0.0269	160	5.2692	**< .001**
		-	Video	SVD	0.13167	0.0269	160	4.8973	**0.001**
		-	Video	UMAP	-0.04667	0.0269	160	-1.7357	0.998
		-	Video	kPCA	0.14167	0.0269	160	5.2692	**< .001**
		-	Video	t-SNE	-0.00833	0.0269	160	-0.3100	1.000
		-	Audio	Autoencoder	-0.07167	0.0269	160	-2.6656	0.685
		-	Audio	FA	0.01333	0.0269	160	0.4959	1.000
		-	Audio	LLE	0.21333	0.0269	160	7.9348	**< .001**
		-	Audio	PCA	0.15333	0.0269	160	5.7031	**< .001**
		-	Audio	SVD	0.13167	0.0269	160	4.8973	**0.001**
		-	Audio	UMAP	-0.03833	0.0269	160	-1.4258	1.000
		-	Audio	kPCA	0.16167	0.0269	160	6.0131	**< .001**
		-	Audio	t-SNE	-0.00500	0.0269	160	-0.1860	1.000
	UMAP	-	Text	kPCA	0.04167	0.0269	160	1.5498	1.000
		-	Text	t-SNE	0.02667	0.0269	160	0.9918	1.000
		-	Video	Autoencoder	-0.03167	0.0269	160	-1.1778	1.000
		-	Video	FA	0.04667	0.0269	160	1.7357	0.998
		-	Video	LLE	0.26667	0.0269	160	9.9185	**< .001**
		-	Video	PCA	0.18667	0.0269	160	6.9429	**< .001**
		-	Video	SVD	0.17667	0.0269	160	6.5710	**< .001**
		-	Video	UMAP	-0.00167	0.0269	160	-0.0620	1.000
		-	Video	kPCA	0.18667	0.0269	160	6.9429	**< .001**
		-	Video	t-SNE	0.03667	0.0269	160	1.3638	1.000

Post Hoc Comparisons - Data Type ✻ Reduction Method

Comparison									
Data Type	**Reduction Method**		**Data Type**	**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	Audio	Autoencoder	-0.02667	0.0269	160	-0.9918	1.000
		-	Audio	FA	0.05833	0.0269	160	2.1697	0.949
		-	Audio	LLE	0.25833	0.0269	160	9.6085	**< .001**
		-	Audio	PCA	0.19833	0.0269	160	7.3769	**< .001**
		-	Audio	SVD	0.17667	0.0269	160	6.5710	**< .001**
		-	Audio	UMAP	0.00667	0.0269	160	0.2480	1.000
		-	Audio	kPCA	0.20667	0.0269	160	7.6868	**< .001**
		-	Audio	t-SNE	0.04000	0.0269	160	1.4878	1.000
	kPCA	-	Text	t-SNE	-0.01500	0.0269	160	-0.5579	1.000
		-	Video	Autoencoder	-0.07333	0.0269	160	-2.7276	0.637
		-	Video	FA	0.00500	0.0269	160	0.1860	1.000
		-	Video	LLE	0.22500	0.0269	160	8.3687	**< .001**
		-	Video	PCA	0.14500	0.0269	160	5.3932	**< .001**
		-	Video	SVD	0.13500	0.0269	160	5.0212	**< .001**
		-	Video	UMAP	-0.04333	0.0269	160	-1.6118	0.999
		-	Video	kPCA	0.14500	0.0269	160	5.3932	**< .001**
		-	Video	t-SNE	-0.00500	0.0269	160	-0.1860	1.000
		-	Audio	Autoencoder	-0.06833	0.0269	160	-2.5416	0.774
		-	Audio	FA	0.01667	0.0269	160	0.6199	1.000
		-	Audio	LLE	0.21667	0.0269	160	8.0588	**< .001**
		-	Audio	PCA	0.15667	0.0269	160	5.8271	**< .001**
		-	Audio	SVD	0.13500	0.0269	160	5.0212	**< .001**
		-	Audio	UMAP	-0.03500	0.0269	160	-1.3018	1.000
		-	Audio	kPCA	0.16500	0.0269	160	6.1371	**< .001**
		-	Audio	t-SNE	-0.00167	0.0269	160	-0.0620	1.000
	t-SNE	-	Video	Autoencoder	-0.05833	0.0269	160	-2.1697	0.949
		-	Video	FA	0.02000	0.0269	160	0.7439	1.000
		-	Video	LLE	0.24000	0.0269	160	8.9266	**< .001**
		-	Video	PCA	0.16000	0.0269	160	5.9511	**< .001**
		-	Video	SVD	0.15000	0.0269	160	5.5791	**< .001**
		-	Video	UMAP	-0.02833	0.0269	160	-1.0538	1.000
		-	Video	kPCA	0.16000	0.0269	160	5.9511	**< .001**
		-	Video	t-SNE	0.01000	0.0269	160	0.3719	1.000
		-	Audio	Autoencoder	-0.05333	0.0269	160	-1.9837	0.983
		-	Audio	FA	0.03167	0.0269	160	1.1778	1.000
		-	Audio	LLE	0.23167	0.0269	160	8.6167	**< .001**
		-	Audio	PCA	0.17167	0.0269	160	6.3850	**< .001**
		-	Audio	SVD	0.15000	0.0269	160	5.5791	**< .001**
		-	Audio	UMAP	-0.02000	0.0269	160	-0.7439	1.000
		-	Audio	kPCA	0.18000	0.0269	160	6.6950	**< .001**
		-	Audio	t-SNE	0.01333	0.0269	160	0.4959	1.000
Video	Autoencoder	-	Video	FA	0.07833	0.0269	160	2.9136	0.490
		-	Video	LLE	0.29833	0.0269	160	11.0963	**< .001**
		-	Video	PCA	0.21833	0.0269	160	8.1208	**< .001**
		-	Video	SVD	0.20833	0.0269	160	7.7488	**< .001**
		-	Video	UMAP	0.03000	0.0269	160	1.1158	1.000

Post Hoc Comparisons - Data Type ✻ Reduction Method

Comparison									
Data Type	**Reduction Method**		**Data Type**	**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	Video	kPCA	0.21833	0.0269	160	8.1208	**< .001**
		-	Video	t-SNE	0.06833	0.0269	160	2.5416	0.774
		-	Audio	Autoencoder	0.00500	0.0269	160	0.1860	1.000
		-	Audio	FA	0.09000	0.0269	160	3.3475	0.205
		-	Audio	LLE	0.29000	0.0269	160	10.7864	**< .001**
		-	Audio	PCA	0.23000	0.0269	160	8.5547	**< .001**
		-	Audio	SVD	0.20833	0.0269	160	7.7488	**< .001**
		-	Audio	UMAP	0.03833	0.0269	160	1.4258	1.000
		-	Audio	kPCA	0.23833	0.0269	160	8.8646	**< .001**
		-	Audio	t-SNE	0.07167	0.0269	160	2.6656	0.685
	FA	-	Video	LLE	0.22000	0.0269	160	8.1828	**< .001**
		-	Video	PCA	0.14000	0.0269	160	5.2072	**< .001**
		-	Video	SVD	0.13000	0.0269	160	4.8353	**0.001**
		-	Video	UMAP	-0.04833	0.0269	160	-1.7977	0.996
		-	Video	kPCA	0.14000	0.0269	160	5.2072	**< .001**
		-	Video	t-SNE	-0.01000	0.0269	160	-0.3719	1.000
		-	Audio	Autoencoder	-0.07333	0.0269	160	-2.7276	0.637
		-	Audio	FA	0.01167	0.0269	160	0.4339	1.000
		-	Audio	LLE	0.21167	0.0269	160	7.8728	**< .001**
		-	Audio	PCA	0.15167	0.0269	160	5.6411	**< .001**
		-	Audio	SVD	0.13000	0.0269	160	4.8353	0.001
		-	Audio	UMAP	-0.04000	0.0269	160	-1.4878	1.000
		-	Audio	kPCA	0.16000	0.0269	160	5.9511	**< .001**
		-	Audio	t-SNE	-0.00667	0.0269	160	-0.2480	1.000
	LLE	-	Video	PCA	-0.08000	0.0269	160	-2.9755	0.442
		-	Video	SVD	-0.09000	0.0269	160	-3.3475	0.205
		-	Video	UMAP	-0.26833	0.0269	160	-9.9805	**< .001**
		-	Video	kPCA	-0.08000	0.0269	160	-2.9755	0.442
		-	Video	t-SNE	-0.23000	0.0269	160	-8.5547	**< .001**
		-	Audio	Autoencoder	-0.29333	0.0269	160	-10.9103	**< .001**
		-	Audio	FA	-0.20833	0.0269	160	-7.7488	**< .001**
		-	Audio	LLE	-0.00833	0.0269	160	-0.3100	1.000
		-	Audio	PCA	-0.06833	0.0269	160	-2.5416	0.774
		-	Audio	SVD	-0.09000	0.0269	160	-3.3475	0.205
		-	Audio	UMAP	-0.26000	0.0269	160	-9.6705	**< .001**
		-	Audio	kPCA	-0.06000	0.0269	160	-2.2317	0.930
		-	Audio	t-SNE	-0.22667	0.0269	160	-8.4307	**< .001**
	PCA	-	Video	SVD	-0.01000	0.0269	160	-0.3719	1.000
		-	Video	UMAP	-0.18833	0.0269	160	-7.0049	**< .001**
		-	Video	kPCA	2.78e-17	0.0269	160	1.03e-15	1.000
		-	Video	t-SNE	-0.15000	0.0269	160	-5.5791	**< .001**
		-	Audio	Autoencoder	-0.21333	0.0269	160	-7.9348	**< .001**
		-	Audio	FA	-0.12833	0.0269	160	-4.7733	**0.002**
		-	Audio	LLE	0.07167	0.0269	160	2.6656	0.685
		-	Audio	PCA	0.01167	0.0269	160	0.4339	1.000
		-	Audio	SVD	-0.01000	0.0269	160	-0.3719	1.000

Post Hoc Comparisons - Data Type ✻ Reduction Method

Comparison									
Data Type	**Reduction Method**		**Data Type**	**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	Audio	UMAP	-0.18000	0.0269	160	-6.6950	**< .001**
		-	Audio	kPCA	0.02000	0.0269	160	0.7439	1.000
		-	Audio	t-SNE	-0.14667	0.0269	160	-5.4552	**< .001**
	SVD	-	Video	UMAP	-0.17833	0.0269	160	-6.6330	**< .001**
		-	Video	kPCA	0.01000	0.0269	160	0.3719	1.000
		-	Video	t-SNE	-0.14000	0.0269	160	-5.2072	**< .001**
		-	Audio	Autoencoder	-0.20333	0.0269	160	-7.5628	**< .001**
		-	Audio	FA	-0.11833	0.0269	160	-4.4013	**0.007**
		-	Audio	LLE	0.08167	0.0269	160	3.0375	0.396
		-	Audio	PCA	0.02167	0.0269	160	0.8059	1.000
		-	Audio	SVD	-9.63e−17	0.0269	160	-3.58e−15	1.000
		-	Audio	UMAP	-0.17000	0.0269	160	-6.3230	**< .001**
		-	Audio	kPCA	0.03000	0.0269	160	1.1158	1.000
		-	Audio	t-SNE	-0.13667	0.0269	160	-5.0832	**< .001**
	UMAP	-	Video	kPCA	0.18833	0.0269	160	7.0049	**< .001**
		-	Video	t-SNE	0.03833	0.0269	160	1.4258	1.000
		-	Audio	Autoencoder	-0.02500	0.0269	160	-0.9299	1.000
		-	Audio	FA	0.06000	0.0269	160	2.2317	0.930
		-	Audio	LLE	0.26000	0.0269	160	9.6705	**< .001**
		-	Audio	PCA	0.20000	0.0269	160	7.4389	**< .001**
		-	Audio	SVD	0.17833	0.0269	160	6.6330	**< .001**
		-	Audio	UMAP	0.00833	0.0269	160	0.3100	1.000
		-	Audio	kPCA	0.20833	0.0269	160	7.7488	**< .001**
		-	Audio	t-SNE	0.04167	0.0269	160	1.5498	1.000
	kPCA	-	Video	t-SNE	-0.15000	0.0269	160	-5.5791	**< .001**
		-	Audio	Autoencoder	-0.21333	0.0269	160	-7.9348	**< .001**
		-	Audio	FA	-0.12833	0.0269	160	-4.7733	**0.002**
		-	Audio	LLE	0.07167	0.0269	160	2.6656	0.685
		-	Audio	PCA	0.01167	0.0269	160	0.4339	1.000
		-	Audio	SVD	-0.01000	0.0269	160	-0.3719	1.000
		-	Audio	UMAP	-0.18000	0.0269	160	-6.6950	**< .001**
		-	Audio	kPCA	0.02000	0.0269	160	0.7439	1.000
		-	Audio	t-SNE	-0.14667	0.0269	160	-5.4552	**< .001**
	t-SNE	-	Audio	Autoencoder	-0.06333	0.0269	160	-2.3556	0.880
		-	Audio	FA	0.02167	0.0269	160	0.8059	1.000
		-	Audio	LLE	0.22167	0.0269	160	8.2447	**< .001**
		-	Audio	PCA	0.16167	0.0269	160	6.0131	**< .001**
		-	Audio	SVD	0.14000	0.0269	160	5.2072	**< .001**
		-	Audio	UMAP	-0.03000	0.0269	160	-1.1158	1.000
		-	Audio	kPCA	0.17000	0.0269	160	6.3230	**< .001**
		-	Audio	t-SNE	0.00333	0.0269	160	0.1240	1.000
Audio	Autoencoder	-	Audio	FA	0.08500	0.0269	160	3.1615	0.311
		-	Audio	LLE	0.28500	0.0269	160	10.6004	**< .001**
		-	Audio	PCA	0.22500	0.0269	160	8.3687	**< .001**
		-	Audio	SVD	0.20333	0.0269	160	7.5628	**< .001**

Post Hoc Comparisons - Data Type ✻ Reduction Method

Comparison									
Data Type	**Reduction Method**		**Data Type**	**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	Audio	UMAP	0.03333	0.0269	160	1.2398	0.813
		-	Audio	kPCA	0.23333	0.0269	160	8.6787	**< .001**
		-	Audio	t-SNE	0.06667	0.0269	160	2.4796	0.957
	FA	-	Audio	LLE	0.20000	0.0269	160	7.4389	**< .001**
		-	Audio	PCA	0.14000	0.0269	160	5.2072	**< .001**
		-	Audio	SVD	0.11833	0.0269	160	4.4013	0.007
		-	Audio	UMAP	-0.05167	0.0269	160	-1.9217	0.989
		-	Audio	kPCA	0.14833	0.0269	160	5.5172	**< .001**
		-	Audio	t-SNE	-0.01833	0.0269	160	-0.6819	1.000
	LLE	-	Audio	PCA	-0.06000	0.0269	160	-2.2317	0.930
		-	Audio	SVD	-0.08167	0.0269	160	-3.0375	0.396
		-	Audio	UMAP	-0.25167	0.0269	160	-9.3606	**< .001**
		-	Audio	kPCA	-0.05167	0.0269	160	-1.9217	0.989
		-	Audio	t-SNE	-0.21833	0.0269	160	-8.1208	**< .001**
	PCA	-	Audio	SVD	-0.02167	0.0269	160	-0.8059	1.000
		-	Audio	UMAP	-0.19167	0.0269	160	-7.1289	**< .001**
		-	Audio	kPCA	0.00833	0.0269	160	0.3100	1.000
		-	Audio	t-SNE	-0.15833	0.0269	160	-5.8891	**< .001**
	SVD	-	Audio	UMAP	-0.17000	0.0269	160	-6.3230	**< .001**
		-	Audio	kPCA	0.03000	0.0269	160	1.1158	1.000
		-	Audio	t-SNE	-0.13667	0.0269	160	-5.0832	**< .001**
	UMAP	-	Audio	kPCA	0.20000	0.0269	160	7.4389	**< .001**
		-	Audio	t-SNE	0.03333	0.0269	160	1.2398	1.000
	kPCA	-	Audio	t-SNE	-0.16667	0.0269	160	-6.1991	**< .001**

Based on the post-hoc analysis, the significant differences in these interactions are due to significant differences among a number of them. There are significant differences between the autoencoder and the FA based factorization machines, autoencoder and Superpixel-based LLE methods as well as between the economy SVD and Radial Basis Function based kPCA on the images datasets. There are significant differences between the multi-core t-SNE method and the PCA with ZCA whitening as well as between the multicore t-SNE and UMAP learn for images methods on the images datasets. There are significant differences between the autoencoder method and the standard LLE method on the text datasets. There are significant differences between the standard LLE method and the standard PCA, standard LLE and standard UMAP, standard LLE and Polynomial Function based kPCA as well as between

standard LLE and standard t-SNE methods on the text datasets. There are significant differences between the autoencoder method and the Spatial Temporal LLE, autoencoder and PCA with Spatiotemporal Cubes, autoencoder and economy SVD as well as between the autoencoder and Radial Basis Function based kPCA methods on the video datasets. There are significant differences between the Spatial Temporal LLE method and the Spatio-Temporal UMAP as well as between the Spatial Temporal LLE and multi-core t-SNE methods on the video datasets. There are significant differences between the Spatial Temporal LLE method and Spatio-Temporal UMAP as well as between the Spatial Temporal LLE and multi-core t-SNE methods on the video datasets. There are significant differences between the PCA with Spatiotemporal Cubes method and the Spatio-Temporal UMAP as well as between PCA with Spatiotemporal Cubes and the multi-core t-SNE methods on the video datasets. There are significant differences between the SVD method and the UMAP as well as between the SVD and t-SNE methods on the video datasets. There are significant differences between the kPCA method and the t-SNE method on the video datasets. There are significant differences between the autoencoder method and the Spectrogram-Based LLE, autoencoder and PCA Filter Bank, autoencoder and economy SVD as well as between the autoencoder and Radial Basis Function based kPCA methods on the audio datasets. There are significant differences between the FA based Factorization machine and the Spectrogram-Based LLE, FA based Factorization machine and PCA Filter Bank, FA based Factorization machine and the economy SVD as well as between FA based Factorization machine and Radial Basis Function based kPCA methods on the audio datasets. There are significant differences between the PCA Filter Bank method and the Spectrogram UMAP as well as between the PCA Filter Bank and multi-core t-SNE methods on the audio datasets. There are significant differences between the economy SVD method and the Spectrogram UMAP as well as between the economy SVD and multi-core t-SNE methods on the audio datasets.

4.2.9 Two-way ANOVA analysis on the interaction between DTs and RMs on CBTs of the *k*HTs

The table 4.22 shows the results of the two-way ANOVA analysis on the interactions between the data types of input datasets and the data dimensionality reduction methods applied on the cluster building times of the existing *k*-hyperparameter tuning techniques.

Table 4.22: Two-Way ANOVA analysis on the interaction between DTs and RMs on CBTs of the *k*HTs.

ANOVA - Time(s)

	Sum of Squares	df	Mean Square	F	p
DataType	5147	3	1715.8	35.35	**< .001**
Reduction_Method	4075	7	582.1	11.99	**< .001**
DataType ✻ Reduction_Method	2132	21	101.5	2.09	**0.006**
Residuals	7767	160	48.5		

The results show that there are significant differences in the cluster building times among both the data types and the dimensionality reduction methods. Moreover, the results show that there are significant interactions between the type of datasets and the dimensionality reduction methods on the cluster building times of the techniques. Based on this evidence, a further post-hoc analysis is performed. The post hoc results on the data types are similar to the ones reported in table 4.16. The post hoc results on the dimensionality reduction methods are similar to the ones reported in table 4.11. The post hoc results on the interaction between the data types and the dimensionality reduction methods are presented in table 4.23.

Table 4.23: Post hoc test of the interaction between DTs and RMs in the two-Way ANOVA analysis on the interaction between DTs and RMs on CBTs of the *k*HTs.

Post Hoc Comparisons - Data Type ✻ Reduction Method

Comparison					Mean Difference	SE	df	t	p_{tukey}
Data Type	**Reduction Method**		**Data Type**	**Reduction Method**					
Images	Autoencoder	-	Images	FA	0.80341	0.418	160	1.9224	0.989
		-	Images	LLE	2.03476	0.418	160	4.8687	**0.001**
		-	Images	PCA	0.50731	0.418	160	1.2139	1.000

Post Hoc Comparisons - Data Type ✻ Reduction Method

Comparison									
Data Type	**Reduction Method**		**Data Type**	**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	Images	SVD	0.77427	0.418	160	1.8527	0.994
		-	Images	UMAP	-0.36231	0.418	160	-0.8669	1.000
		-	Images	kPCA	-0.22839	0.418	160	-0.5465	1.000
		-	Images	t-SNE	0.29248	0.418	160	0.6998	1.000
		-	Text	Autoencoder	-0.59557	0.418	160	-1.4251	1.000
		-	Text	FA	-1.50414	0.418	160	-3.5991	0.106
		-	Text	LLE	-1.48510	0.418	160	-3.5535	0.121
		-	Text	PCA	-1.36081	0.418	160	-3.2561	0.253
		-	Text	SVD	-1.28489	0.418	160	-3.0745	0.370
		-	Text	UMAP	-1.28496	0.418	160	-3.0746	0.370
		-	Text	kPCA	-1.04035	0.418	160	-2.4893	0.807
		-	Text	t-SNE	-0.99206	0.418	160	-2.3738	0.872
		-	Video	Autoencoder	0.26725	0.418	160	0.6395	1.000
		-	Video	FA	-0.44503	0.418	160	-1.0649	1.000
		-	Video	LLE	-0.17921	0.418	160	-0.4288	1.000
		-	Video	PCA	-0.55318	0.418	160	-1.3236	1.000
		-	Video	SVD	-0.40452	0.418	160	-0.9679	1.000
		-	Video	UMAP	-0.26198	0.418	160	-0.6269	1.000
		-	Video	kPCA	-0.75685	0.418	160	-1.8110	0.995
		-	Video	t-SNE	-0.10275	0.418	160	-0.2459	1.000
		-	Audio	Autoencoder	-0.83269	0.418	160	-1.9924	0.982
		-	Audio	FA	-1.09590	0.418	160	-2.6223	0.717
		-	Audio	LLE	-0.81316	0.418	160	-1.9457	0.987
		-	Audio	PCA	-0.56967	0.418	160	-1.3631	1.000
		-	Audio	SVD	-0.63364	0.418	160	-1.5162	1.000
		-	Audio	UMAP	-0.42702	0.418	160	-1.0218	1.000
		-	Audio	kPCA	-0.80765	0.418	160	-1.9325	0.988
		-	Audio	t-SNE	-0.77091	0.418	160	-1.8446	0.994
	FA	-	Images	LLE	1.23136	0.418	160	2.9464	0.465
		-	Images	PCA	-0.29610	0.418	160	-0.7085	1.000
		-	Images	SVD	-0.02913	0.418	160	-0.0697	1.000
		-	Images	UMAP	-1.16572	0.418	160	-2.7893	0.589
		-	Images	kPCA	-1.03179	0.418	160	-2.4689	0.820
		-	Images	t-SNE	-0.51093	0.418	160	-1.2225	1.000
		-	Text	Autoencoder	-1.39898	0.418	160	-3.3474	0.205
		-	Text	FA	-2.30755	0.418	160	-5.5214	**< .001**
		-	Text	LLE	-2.28851	0.418	160	-5.4759	**< .001**
		-	Text	PCA	-2.16422	0.418	160	-5.1785	**< .001**
		-	Text	SVD	-2.08830	0.418	160	-4.9968	**< .001**
		-	Text	UMAP	-2.08837	0.418	160	-4.9970	**< .001**
		-	Text	kPCA	-1.84376	0.418	160	-4.4117	**0.007**
		-	Text	t-SNE	-1.79547	0.418	160	-4.2961	**0.011**
		-	Video	Autoencoder	-0.53616	0.418	160	-1.2829	1.000
		-	Video	FA	-1.24844	0.418	160	-2.9872	0.434
		-	Video	LLE	-0.98261	0.418	160	-2.3512	0.883
		-	Video	PCA	-1.35659	0.418	160	-3.2460	0.259

Post Hoc Comparisons - Data Type ✻ Reduction Method

Comparison									
Data Type	**Reduction Method**		**Data Type**	**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	Video	SVD	-1.20793	0.418	160	-2.8903	0.509
		-	Video	UMAP	-1.06539	0.418	160	-2.5492	0.769
		-	Video	kPCA	-1.56025	0.418	160	-3.7333	**0.072**
		-	Video	t-SNE	-0.90615	0.418	160	-2.1682	0.949
		-	Audio	Autoencoder	-1.63610	0.418	160	-3.9148	**0.041**
		-	Audio	FA	-1.89931	0.418	160	-4.5446	**0.004**
		-	Audio	LLE	-1.61657	0.418	160	-3.8681	**0.048**
		-	Audio	PCA	-1.37307	0.418	160	-3.2855	0.237
		-	Audio	SVD	-1.43705	0.418	160	-3.4385	0.164
		-	Audio	UMAP	-1.23043	0.418	160	-2.9441	0.466
		-	Audio	kPCA	-1.61106	0.418	160	-3.8549	0.050
		-	Audio	t-SNE	-1.57431	0.418	160	-3.7670	0.065
	LLE	-	Images	PCA	-1.52745	0.418	160	-3.6549	0.091
		-	Images	SVD	-1.26049	0.418	160	-3.0161	0.412
		-	Images	UMAP	-2.39707	0.418	160	-5.7357	**< .001**
		-	Images	kPCA	-2.26315	0.418	160	-5.4152	**< .001**
		-	Images	t-SNE	-1.74228	0.418	160	-4.1689	**0.017**
		-	Text	Autoencoder	-2.63033	0.418	160	-6.2938	**< .001**
		-	Text	FA	-3.53890	0.418	160	-8.4678	**< .001**
		-	Text	LLE	-3.51987	0.418	160	-8.4222	**< .001**
		-	Text	PCA	-3.39558	0.418	160	-8.1249	**< .001**
		-	Text	SVD	-3.31965	0.418	160	-7.9432	**< .001**
		-	Text	UMAP	-3.31972	0.418	160	-7.9434	**< .001**
		-	Text	kPCA	-3.07512	0.418	160	-7.3581	**< .001**
		-	Text	t-SNE	-3.02682	0.418	160	-7.2425	**< .001**
		-	Video	Autoencoder	-1.76751	0.418	160	-4.2293	**0.014**
		-	Video	FA	-2.47980	0.418	160	-5.9336	**< .001**
		-	Video	LLE	-2.21397	0.418	160	-5.2975	**< .001**
		-	Video	PCA	-2.58795	0.418	160	-6.1924	**< .001**
		-	Video	SVD	-2.43928	0.418	160	-5.8367	**< .001**
		-	Video	UMAP	-2.29674	0.418	160	-5.4956	**< .001**
		-	Video	kPCA	-2.79161	0.418	160	-6.6797	**< .001**
		-	Video	t-SNE	-2.13751	0.418	160	-5.1146	**< .001**
		-	Audio	Autoencoder	-2.86746	0.418	160	-6.8612	**< .001**
		-	Audio	FA	-3.13067	0.418	160	-7.4910	**< .001**
		-	Audio	LLE	-2.84793	0.418	160	-6.8145	**< .001**
		-	Audio	PCA	-2.60443	0.418	160	-6.2318	**< .001**
		-	Audio	SVD	-2.66840	0.418	160	-6.3849	**< .001**
		-	Audio	UMAP	-2.46178	0.418	160	-5.8905	**< .001**
		-	Audio	kPCA	-2.84242	0.418	160	-6.8013	**< .001**
		-	Audio	t-SNE	-2.80567	0.418	160	-6.7133	**< .001**
	PCA	-	Images	SVD	0.26697	0.418	160	0.6388	1.000
		-	Images	UMAP	-0.86962	0.418	160	-2.0808	0.969
		-	Images	kPCA	-0.73570	0.418	160	-1.7604	0.997
		-	Images	t-SNE	-0.21483	0.418	160	-0.5140	1.000
		-	Text	Autoencoder	-1.10288	0.418	160	-2.6389	0.705

Post Hoc Comparisons - Data Type ✻ Reduction Method

Comparison									
Data Type	**Reduction Method**		**Data Type**	**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	Text	FA	-2.01145	0.418	160	-4.8129	**0.001**
		-	Text	LLE	-1.99241	0.418	160	-4.7674	**0.002**
		-	Text	PCA	-1.86812	0.418	160	-4.4700	**0.006**
		-	Text	SVD	-1.79220	0.418	160	-4.2883	**0.011**
		-	Text	UMAP	-1.79227	0.418	160	-4.2885	**0.011**
		-	Text	kPCA	-1.54766	0.418	160	-3.7032	0.079
		-	Text	t-SNE	-1.49937	0.418	160	-3.5876	0.110
		-	Video	Autoencoder	-0.24006	0.418	160	-0.5744	1.000
		-	Video	FA	-0.95234	0.418	160	-2.2787	0.913
		-	Video	LLE	-0.68652	0.418	160	-1.6427	0.999
		-	Video	PCA	-1.06049	0.418	160	-2.5375	0.776
		-	Video	SVD	-0.91183	0.418	160	-2.1818	0.945
		-	Video	UMAP	-0.76929	0.418	160	-1.8407	0.994
		-	Video	kPCA	-1.26415	0.418	160	-3.0248	0.405
		-	Video	t-SNE	-0.61006	0.418	160	-1.4597	1.000
		-	Audio	Autoencoder	-1.34000	0.418	160	-3.2063	0.283
		-	Audio	FA	-1.60321	0.418	160	-3.8361	0.053
		-	Audio	LLE	-1.32047	0.418	160	-3.1596	0.312
		-	Audio	PCA	-1.07697	0.418	160	-2.5770	0.750
		-	Audio	SVD	-1.14095	0.418	160	-2.7300	0.636
		-	Audio	UMAP	-0.93433	0.418	160	-2.2356	0.929
		-	Audio	kPCA	-1.31496	0.418	160	-3.1464	0.321
		-	Audio	t-SNE	-1.27822	0.418	160	-3.0585	0.381
	SVD	-	Images	UMAP	-1.13659	0.418	160	-2.7196	0.644
		-	Images	kPCA	-1.00266	0.418	160	-2.3991	0.859
		-	Images	t-SNE	-0.48179	0.418	160	-1.1528	1.000
		-	Text	Autoencoder	-1.36985	0.418	160	-3.2777	0.241
		-	Text	FA	-2.27842	0.418	160	-5.4517	**< .001**
		-	Text	LLE	-2.25938	0.418	160	-5.4062	**< .001**
		-	Text	PCA	-2.13509	0.418	160	-5.1088	**< .001**
		-	Text	SVD	-2.05917	0.418	160	-4.9271	**< .001**
		-	Text	UMAP	-2.05924	0.418	160	-4.9273	**< .001**
		-	Text	kPCA	-1.81463	0.418	160	-4.3420	**0.009**
		-	Text	t-SNE	-1.76633	0.418	160	-4.2264	**0.014**
		-	Video	Autoencoder	-0.50702	0.418	160	-1.2132	1.000
		-	Video	FA	-1.21931	0.418	160	-2.9175	0.487
		-	Video	LLE	-0.95348	0.418	160	-2.2815	0.912
		-	Video	PCA	-1.32746	0.418	160	-3.1763	0.301
		-	Video	SVD	-1.17880	0.418	160	-2.8206	0.564
		-	Video	UMAP	-1.03626	0.418	160	-2.4795	0.813
		-	Video	kPCA	-1.53112	0.418	160	-3.6636	0.088
		-	Video	t-SNE	-0.87702	0.418	160	-2.0985	0.965
		-	Audio	Autoencoder	-1.60697	0.418	160	-3.8451	0.051
		-	Audio	FA	-1.87018	0.418	160	-4.4749	**0.006**
		-	Audio	LLE	-1.58744	0.418	160	-3.7984	0.059
		-	Audio	PCA	-1.34394	0.418	160	-3.2157	0.277

Post Hoc Comparisons - Data Type ✻ Reduction Method

Comparison									
Data Type	**Reduction Method**		**Data Type**	**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	Audio	SVD	-1.40792	0.418	160	-3.3688	0.195
		-	Audio	UMAP	-1.20129	0.418	160	-2.8744	0.521
		-	Audio	kPCA	-1.58193	0.418	160	-3.7852	0.062
		-	Audio	t-SNE	-1.54518	0.418	160	-3.6973	0.080
	UMAP	-	Images	kPCA	0.13392	0.418	160	0.3204	1.000
		-	Images	t-SNE	0.65479	0.418	160	1.5668	1.000
		-	Text	Autoencoder	-0.23326	0.418	160	-0.5581	1.000
		-	Text	FA	-1.14183	0.418	160	-2.7321	0.634
		-	Text	LLE	-1.12279	0.418	160	-2.6866	0.669
		-	Text	PCA	-0.99850	0.418	160	-2.3892	0.864
		-	Text	SVD	-0.92258	0.418	160	-2.2075	0.938
		-	Text	UMAP	-0.92265	0.418	160	-2.2077	0.938
		-	Text	kPCA	-0.67804	0.418	160	-1.6224	0.999
		-	Text	t-SNE	-0.62975	0.418	160	-1.5068	1.000
		-	Video	Autoencoder	0.62956	0.418	160	1.5064	1.000
		-	Video	FA	-0.08272	0.418	160	-0.1979	1.000
		-	Video	LLE	0.18310	0.418	160	0.4381	1.000
		-	Video	PCA	-0.19087	0.418	160	-0.4567	1.000
		-	Video	SVD	-0.04221	0.418	160	-0.1010	1.000
		-	Video	UMAP	0.10033	0.418	160	0.2401	1.000
		-	Video	kPCA	-0.39454	0.418	160	-0.9440	1.000
		-	Video	t-SNE	0.25956	0.418	160	0.6211	1.000
		-	Audio	Autoencoder	-0.47038	0.418	160	-1.1255	1.000
		-	Audio	FA	-0.73359	0.418	160	-1.7553	0.997
		-	Audio	LLE	-0.45085	0.418	160	-1.0788	1.000
		-	Audio	PCA	-0.20735	0.418	160	-0.4962	1.000
		-	Audio	SVD	-0.27133	0.418	160	-0.6492	1.000
		-	Audio	UMAP	-0.06471	0.418	160	-0.1548	1.000
		-	Audio	kPCA	-0.44534	0.418	160	-1.0656	1.000
		-	Audio	t-SNE	-0.40860	0.418	160	-0.9777	1.000
	kPCA	-	Images	t-SNE	0.52087	0.418	160	1.2463	1.000
		-	Text	Autoencoder	-0.36718	0.418	160	-0.8786	1.000
		-	Text	FA	-1.27575	0.418	160	-3.0526	0.385
		-	Text	LLE	-1.25672	0.418	160	-3.0070	0.419
		-	Text	PCA	-1.13243	0.418	160	-2.7096	0.651
		-	Text	SVD	-1.05650	0.418	160	-2.5280	0.783
		-	Text	UMAP	-1.05657	0.418	160	-2.5281	0.783
		-	Text	kPCA	-0.81196	0.418	160	-1.9428	0.987
		-	Text	t-SNE	-0.76367	0.418	160	-1.8273	0.995
		-	Video	Autoencoder	0.49564	0.418	160	1.1860	1.000
		-	Video	FA	-0.21665	0.418	160	-0.5184	1.000
		-	Video	LLE	0.04918	0.418	160	0.1177	1.000
		-	Video	PCA	-0.32480	0.418	160	-0.7772	1.000
		-	Video	SVD	-0.17613	0.418	160	-0.4214	1.000
		-	Video	UMAP	-0.03359	0.418	160	-0.0804	1.000
		-	Video	kPCA	-0.52846	0.418	160	-1.2645	1.000

Post Hoc Comparisons - Data Type ✻ Reduction Method

Comparison									
Data Type	**Reduction Method**		**Data Type**	**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	Video	t-SNE	0.12564	0.418	160	0.3006	1.000
		-	Audio	Autoencoder	-0.60431	0.418	160	-1.4460	1.000
		-	Audio	FA	-0.86752	0.418	160	-2.0758	0.970
		-	Audio	LLE	-0.58478	0.418	160	-1.3992	1.000
		-	Audio	PCA	-0.34128	0.418	160	-0.8166	1.000
		-	Audio	SVD	-0.40525	0.418	160	-0.9697	1.000
		-	Audio	UMAP	-0.19863	0.418	160	-0.4753	1.000
		-	Audio	kPCA	-0.57927	0.418	160	-1.3861	1.000
		-	Audio	t-SNE	-0.54252	0.418	160	-1.2981	1.000
	t-SNE	-	Text	Autoencoder	-0.88805	0.418	160	-2.1249	0.960
		-	Text	FA	-1.79662	0.418	160	-4.2989	**0.011**
		-	Text	LLE	-1.77758	0.418	160	-4.2534	**0.013**
		-	Text	PCA	-1.65330	0.418	160	-3.9560	**0.036**
		-	Text	SVD	-1.57737	0.418	160	-3.7743	0.064
		-	Text	UMAP	-1.57744	0.418	160	-3.7745	0.064
		-	Text	kPCA	-1.33283	0.418	160	-3.1892	0.293
		-	Text	t-SNE	-1.28454	0.418	160	-3.0736	0.370
		-	Video	Autoencoder	-0.02523	0.418	160	-0.0604	1.000
		-	Video	FA	-0.73751	0.418	160	-1.7647	0.997
		-	Video	LLE	-0.47169	0.418	160	-1.1286	1.000
		-	Video	PCA	-0.84567	0.418	160	-2.0235	0.978
		-	Video	SVD	-0.69700	0.418	160	-1.6678	0.999
		-	Video	UMAP	-0.55446	0.418	160	-1.3267	1.000
		-	Video	kPCA	-1.04933	0.418	160	-2.5108	0.794
		-	Video	t-SNE	-0.39523	0.418	160	-0.9457	1.000
		-	Audio	Autoencoder	-1.12517	0.418	160	-2.6923	0.665
		-	Audio	FA	-1.38839	0.418	160	-3.3221	0.218
		-	Audio	LLE	-1.10565	0.418	160	-2.6456	0.700
		-	Audio	PCA	-0.86215	0.418	160	-2.0629	0.972
		-	Audio	SVD	-0.92612	0.418	160	-2.2160	0.935
		-	Audio	UMAP	-0.71950	0.418	160	-1.7216	0.998
		-	Audio	kPCA	-1.10014	0.418	160	-2.6324	0.710
		-	Audio	t-SNE	-1.06339	0.418	160	-2.5445	0.772
Text	Autoencoder	-	Text	FA	-0.90857	0.418	160	-2.1740	0.948
		-	Text	LLE	-0.88953	0.418	160	-2.1285	0.959
		-	Text	PCA	-0.76524	0.418	160	-1.8311	0.995
		-	Text	SVD	-0.68932	0.418	160	-1.6494	0.999
		-	Text	UMAP	-0.68939	0.418	160	-1.6496	0.999
		-	Text	kPCA	-0.44478	0.418	160	-1.0643	1.000
		-	Text	t-SNE	-0.39649	0.418	160	-0.9487	1.000
		-	Video	Autoencoder	0.86282	0.418	160	2.0645	0.972
		-	Video	FA	0.15054	0.418	160	0.3602	1.000
		-	Video	LLE	0.41636	0.418	160	0.9963	1.000
		-	Video	PCA	0.04239	0.418	160	0.1014	1.000
		-	Video	SVD	0.19105	0.418	160	0.4571	1.000
		-	Video	UMAP	0.33359	0.418	160	0.7982	1.000

Post Hoc Comparisons - Data Type ✻ Reduction Method

Comparison									
Data Type	**Reduction Method**		**Data Type**	**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	Video	kPCA	-0.16127	0.418	160	-0.3859	1.000
		-	Video	t-SNE	0.49282	0.418	160	1.1792	1.000
		-	Audio	Autoencoder	-0.23712	0.418	160	-0.5674	1.000
		-	Audio	FA	-0.50033	0.418	160	-1.1972	1.000
		-	Audio	LLE	-0.21759	0.418	160	-0.5207	1.000
		-	Audio	PCA	0.02591	0.418	160	0.0620	1.000
		-	Audio	SVD	-0.03807	0.418	160	-0.0911	1.000
		-	Audio	UMAP	0.16855	0.418	160	0.4033	1.000
		-	Audio	kPCA	-0.21208	0.418	160	-0.5075	1.000
		-	Audio	t-SNE	-0.17534	0.418	160	-0.4195	1.000
	FA	-	Text	LLE	0.01904	0.418	160	0.0456	1.000
		-	Text	PCA	0.14333	0.418	160	0.3429	1.000
		-	Text	SVD	0.21925	0.418	160	0.5246	1.000
		-	Text	UMAP	0.21918	0.418	160	0.5244	1.000
		-	Text	kPCA	0.46379	0.418	160	1.1097	1.000
		-	Text	t-SNE	0.51208	0.418	160	1.2253	1.000
		-	Video	Autoencoder	1.77139	0.418	160	4.2385	**0.013**
		-	Video	FA	1.05911	0.418	160	2.5342	0.779
		-	Video	LLE	1.32493	0.418	160	3.1703	0.305
		-	Video	PCA	0.95096	0.418	160	2.2754	0.915
		-	Video	SVD	1.09962	0.418	160	2.6311	0.711
		-	Video	UMAP	1.24216	0.418	160	2.9722	0.445
		-	Video	kPCA	0.74729	0.418	160	1.7881	0.996
		-	Video	t-SNE	1.40139	0.418	160	3.3532	0.202
		-	Audio	Autoencoder	0.67145	0.418	160	1.6066	0.999
		-	Audio	FA	0.40824	0.418	160	0.9768	1.000
		-	Audio	LLE	0.69098	0.418	160	1.6533	0.999
		-	Audio	PCA	0.93448	0.418	160	2.2360	0.929
		-	Audio	SVD	0.87050	0.418	160	2.0829	0.968
		-	Audio	UMAP	1.07712	0.418	160	2.5773	0.749
		-	Audio	kPCA	0.69649	0.418	160	1.6665	0.999
		-	Audio	t-SNE	0.73323	0.418	160	1.7545	0.997
	LLE	-	Text	PCA	0.12429	0.418	160	0.2974	1.000
		-	Text	SVD	0.20021	0.418	160	0.4791	1.000
		-	Text	UMAP	0.20014	0.418	160	0.4789	1.000
		-	Text	kPCA	0.44475	0.418	160	1.0642	1.000
		-	Text	t-SNE	0.49304	0.418	160	1.1797	1.000
		-	Video	Autoencoder	1.75235	0.418	160	4.1930	**0.016**
		-	Video	FA	1.04007	0.418	160	2.4887	0.808
		-	Video	LLE	1.30590	0.418	160	3.1247	0.335
		-	Video	PCA	0.93192	0.418	160	2.2299	0.931
		-	Video	SVD	1.08058	0.418	160	2.5856	0.744
		-	Video	UMAP	1.22312	0.418	160	2.9267	0.480
		-	Video	kPCA	0.72826	0.418	160	1.7426	0.998
		-	Video	t-SNE	1.38236	0.418	160	3.3077	0.225
		-	Audio	Autoencoder	0.65241	0.418	160	1.5611	1.000

Post Hoc Comparisons - Data Type ✻ Reduction Method

Comparison									
Data Type	**Reduction Method**		**Data Type**	**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	Audio	FA	0.38920	0.418	160	0.9313	1.000
		-	Audio	LLE	0.67194	0.418	160	1.6078	0.999
		-	Audio	PCA	0.91544	0.418	160	2.1904	0.943
		-	Audio	SVD	0.85146	0.418	160	2.0374	0.976
		-	Audio	UMAP	1.05808	0.418	160	2.5318	0.780
		-	Audio	kPCA	0.67745	0.418	160	1.6210	0.999
		-	Audio	t-SNE	0.71419	0.418	160	1.7089	0.998
	PCA	-	Text	SVD	0.07592	0.418	160	0.1817	1.000
		-	Text	UMAP	0.07585	0.418	160	0.1815	1.000
		-	Text	kPCA	0.32046	0.418	160	0.7668	1.000
		-	Text	t-SNE	0.36876	0.418	160	0.8823	1.000
		-	Video	Autoencoder	1.62807	0.418	160	3.8956	0.044
		-	Video	FA	0.91578	0.418	160	2.1913	0.943
		-	Video	LLE	1.18161	0.418	160	2.8273	0.559
		-	Video	PCA	0.80763	0.418	160	1.9325	0.988
		-	Video	SVD	0.95629	0.418	160	2.2882	0.910
		-	Video	UMAP	1.09883	0.418	160	2.6293	0.712
		-	Video	kPCA	0.60397	0.418	160	1.4452	1.000
		-	Video	t-SNE	1.25807	0.418	160	3.0103	0.416
		-	Audio	Autoencoder	0.52812	0.418	160	1.2637	1.000
		-	Audio	FA	0.26491	0.418	160	0.6339	1.000
		-	Audio	LLE	0.54765	0.418	160	1.3104	1.000
		-	Audio	PCA	0.79115	0.418	160	1.8930	0.991
		-	Audio	SVD	0.72717	0.418	160	1.7400	0.998
		-	Audio	UMAP	0.93380	0.418	160	2.2344	0.929
		-	Audio	kPCA	0.55316	0.418	160	1.3236	1.000
		-	Audio	t-SNE	0.58991	0.418	160	1.4115	1.000
	SVD	-	Text	UMAP	-6.99e−5	0.418	160	-1.67e−4	1.000
		-	Text	kPCA	0.24454	0.418	160	0.5851	1.000
		-	Text	t-SNE	0.29283	0.418	160	0.7007	1.000
		-	Video	Autoencoder	1.55214	0.418	160	3.7139	0.076
		-	Video	FA	0.83986	0.418	160	2.0096	0.980
		-	Video	LLE	1.10568	0.418	160	2.6457	0.700
		-	Video	PCA	0.73171	0.418	160	1.7508	0.997
		-	Video	SVD	0.88037	0.418	160	2.1065	0.964
		-	Video	UMAP	1.02291	0.418	160	2.4476	0.832
		-	Video	kPCA	0.52805	0.418	160	1.2635	1.000
		-	Video	t-SNE	1.18214	0.418	160	2.8286	0.558
		-	Audio	Autoencoder	0.45220	0.418	160	1.0820	1.000
		-	Audio	FA	0.18899	0.418	160	0.4522	1.000
		-	Audio	LLE	0.47173	0.418	160	1.1287	1.000
		-	Audio	PCA	0.71523	0.418	160	1.7114	0.998
		-	Audio	SVD	0.65125	0.418	160	1.5583	1.000
		-	Audio	UMAP	0.85787	0.418	160	2.0527	0.974
		-	Audio	kPCA	0.47724	0.418	160	1.1419	1.000

Post Hoc Comparisons - Data Type ✻ Reduction Method

Comparison									
Data Type	**Reduction Method**		**Data Type**	**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	Audio	t-SNE	0.51398	0.418	160	1.2298	1.000
	UMAP	-	Text	kPCA	0.24461	0.418	160	0.5853	1.000
		-	Text	t-SNE	0.29290	0.418	160	0.7008	1.000
		-	Video	Autoencoder	1.55221	0.418	160	3.7141	0.076
		-	Video	FA	0.83993	0.418	160	2.0098	0.980
		-	Video	LLE	1.10575	0.418	160	2.6458	0.700
		-	Video	PCA	0.73178	0.418	160	1.7510	0.997
		-	Video	SVD	0.88044	0.418	160	2.1067	0.964
		-	Video	UMAP	1.02298	0.418	160	2.4478	0.832
		-	Video	kPCA	0.52812	0.418	160	1.2637	1.000
		-	Video	t-SNE	1.18221	0.418	160	2.8288	0.557
		-	Audio	Autoencoder	0.45227	0.418	160	1.0822	1.000
		-	Audio	FA	0.18906	0.418	160	0.4524	1.000
		-	Audio	LLE	0.47180	0.418	160	1.1289	1.000
		-	Audio	PCA	0.71530	0.418	160	1.7115	0.998
		-	Audio	SVD	0.65132	0.418	160	1.5585	1.000
		-	Audio	UMAP	0.85794	0.418	160	2.0529	0.974
		-	Audio	kPCA	0.47731	0.418	160	1.1421	1.000
		-	Audio	t-SNE	0.51405	0.418	160	1.2300	1.000
	kPCA	-	Text	t-SNE	0.04829	0.418	160	0.1156	1.000
		-	Video	Autoencoder	1.30760	0.418	160	3.1288	0.332
		-	Video	FA	0.59532	0.418	160	1.4245	1.000
		-	Video	LLE	0.86114	0.418	160	2.0605	0.972
		-	Video	PCA	0.48717	0.418	160	1.1657	1.000
		-	Video	SVD	0.63583	0.418	160	1.5214	1.000
		-	Video	UMAP	0.77837	0.418	160	1.8625	0.993
		-	Video	kPCA	0.28351	0.418	160	0.6784	1.000
		-	Video	t-SNE	0.93760	0.418	160	2.2435	0.926
		-	Audio	Autoencoder	0.20766	0.418	160	0.4969	1.000
		-	Audio	FA	-0.05555	0.418	160	-0.1329	1.000
		-	Audio	LLE	0.22719	0.418	160	0.5436	1.000
		-	Audio	PCA	0.47069	0.418	160	1.1262	1.000
		-	Audio	SVD	0.40671	0.418	160	0.9732	1.000
		-	Audio	UMAP	0.61333	0.418	160	1.4676	1.000
		-	Audio	kPCA	0.23270	0.418	160	0.5568	1.000
		-	Audio	t-SNE	0.26944	0.418	160	0.6447	1.000
	t-SNE	-	Video	Autoencoder	1.25931	0.418	160	3.0132	0.414
		-	Video	FA	0.54703	0.418	160	1.3089	1.000
		-	Video	LLE	0.81285	0.418	160	1.9450	0.987
		-	Video	PCA	0.43887	0.418	160	1.0501	1.000
		-	Video	SVD	0.58754	0.418	160	1.4058	1.000
		-	Video	UMAP	0.73008	0.418	160	1.7469	0.997
		-	Video	kPCA	0.23521	0.418	160	0.5628	1.000
		-	Video	t-SNE	0.88931	0.418	160	2.1279	0.959
		-	Audio	Autoencoder	0.15937	0.418	160	0.3813	1.000
		-	Audio	FA	-0.10385	0.418	160	-0.2485	1.000

Post Hoc Comparisons - Data Type ✻ Reduction Method

Comparison									
Data Type	**Reduction Method**		**Data Type**	**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	Audio	LLE	0.17889	0.418	160	0.4281	1.000
		-	Audio	PCA	0.42239	0.418	160	1.0107	1.000
		-	Audio	SVD	0.35842	0.418	160	0.8576	1.000
		-	Audio	UMAP	0.56504	0.418	160	1.3520	1.000
		-	Audio	kPCA	0.18440	0.418	160	0.4412	1.000
		-	Audio	t-SNE	0.22115	0.418	160	0.5292	1.000
Video	Autoencoder	-	Video	FA	-0.71228	0.418	160	-1.7043	0.998
		-	Video	LLE	-0.44646	0.418	160	-1.0683	1.000
		-	Video	PCA	-0.82044	0.418	160	-1.9631	0.985
		-	Video	SVD	-0.67177	0.418	160	-1.6074	0.999
		-	Video	UMAP	-0.52923	0.418	160	-1.2663	1.000
		-	Video	kPCA	-1.02410	0.418	160	-2.4504	0.831
		-	Video	t-SNE	-0.37000	0.418	160	-0.8853	1.000
		-	Audio	Autoencoder	-1.09994	0.418	160	-2.6319	0.710
		-	Audio	FA	-1.36316	0.418	160	-3.2617	0.250
		-	Audio	LLE	-1.08042	0.418	160	-2.5852	0.744
		-	Audio	PCA	-0.83692	0.418	160	-2.0026	0.981
		-	Audio	SVD	-0.90089	0.418	160	-2.1556	0.952
		-	Audio	UMAP	-0.69427	0.418	160	-1.6612	0.999
		-	Audio	kPCA	-1.07491	0.418	160	-2.5720	0.753
		-	Audio	t-SNE	-1.03816	0.418	160	-2.4841	0.811
	FA	-	Video	LLE	0.26583	0.418	160	0.6361	1.000
		-	Video	PCA	-0.10815	0.418	160	-0.2588	1.000
		-	Video	SVD	0.04051	0.418	160	0.0969	1.000
		-	Video	UMAP	0.18305	0.418	160	0.4380	1.000
		-	Video	kPCA	-0.31181	0.418	160	-0.7461	1.000
		-	Video	t-SNE	0.34229	0.418	160	0.8190	1.000
		-	Audio	Autoencoder	-0.38766	0.418	160	-0.9276	1.000
		-	Audio	FA	-0.65087	0.418	160	-1.5574	1.000
		-	Audio	LLE	-0.36813	0.418	160	-0.8809	1.000
		-	Audio	PCA	-0.12463	0.418	160	-0.2982	1.000
		-	Audio	SVD	-0.18861	0.418	160	-0.4513	1.000
		-	Audio	UMAP	0.01801	0.418	160	0.0431	1.000
		-	Audio	kPCA	-0.36262	0.418	160	-0.8677	1.000
		-	Audio	t-SNE	-0.32588	0.418	160	-0.7797	1.000
	LLE	-	Video	PCA	-0.37398	0.418	160	-0.8948	1.000
		-	Video	SVD	-0.22531	0.418	160	-0.5391	1.000
		-	Video	UMAP	-0.08277	0.418	160	-0.1981	1.000
		-	Video	kPCA	-0.57764	0.418	160	-1.3822	1.000
		-	Video	t-SNE	0.07646	0.418	160	0.1830	1.000
		-	Audio	Autoencoder	-0.65349	0.418	160	-1.5636	1.000
		-	Audio	FA	-0.91670	0.418	160	-2.1934	0.942
		-	Audio	LLE	-0.63396	0.418	160	-1.5169	1.000
		-	Audio	PCA	-0.39046	0.418	160	-0.9343	1.000
		-	Audio	SVD	-0.45443	0.418	160	-1.0874	1.000
		-	Audio	UMAP	-0.24781	0.418	160	-0.5930	1.000

Post Hoc Comparisons - Data Type ✻ Reduction Method

Comparison									
Data Type	**Reduction Method**		**Data Type**	**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	Audio	kPCA	-0.62845	0.418	160	-1.5037	1.000
		-	Audio	t-SNE	-0.59170	0.418	160	-1.4158	1.000
	PCA	-	Video	SVD	0.14866	0.418	160	0.3557	1.000
		-	Video	UMAP	0.29120	0.418	160	0.6968	1.000
		-	Video	kPCA	-0.20366	0.418	160	-0.4873	1.000
		-	Video	t-SNE	0.45044	0.418	160	1.0778	1.000
		-	Audio	Autoencoder	-0.27951	0.418	160	-0.6688	1.000
		-	Audio	FA	-0.54272	0.418	160	-1.2986	1.000
		-	Audio	LLE	-0.25998	0.418	160	-0.6221	1.000
		-	Audio	PCA	-0.01648	0.418	160	-0.0394	1.000
		-	Audio	SVD	-0.08046	0.418	160	-0.1925	1.000
		-	Audio	UMAP	0.12617	0.418	160	0.3019	1.000
		-	Audio	kPCA	-0.25447	0.418	160	-0.6089	1.000
		-	Audio	t-SNE	-0.21772	0.418	160	-0.5210	1.000
	SVD	-	Video	UMAP	0.14254	0.418	160	0.3411	1.000
		-	Video	kPCA	-0.35232	0.418	160	-0.8430	1.000
		-	Video	t-SNE	0.30177	0.418	160	0.7221	1.000
		-	Audio	Autoencoder	-0.42817	0.418	160	-1.0245	1.000
		-	Audio	FA	-0.69138	0.418	160	-1.6543	0.999
		-	Audio	LLE	-0.40864	0.418	160	-0.9778	1.000
		-	Audio	PCA	-0.16514	0.418	160	-0.3952	1.000
		-	Audio	SVD	-0.22912	0.418	160	-0.5482	1.000
		-	Audio	UMAP	-0.02250	0.418	160	-0.0538	1.000
		-	Audio	kPCA	-0.40313	0.418	160	-0.9646	1.000
		-	Audio	t-SNE	-0.36639	0.418	160	-0.8767	1.000
	UMAP	-	Video	kPCA	-0.49486	0.418	160	-1.1841	1.000
		-	Video	t-SNE	0.15923	0.418	160	0.3810	1.000
		-	Audio	Autoencoder	-0.57071	0.418	160	-1.3656	1.000
		-	Audio	FA	-0.83392	0.418	160	-1.9954	0.982
		-	Audio	LLE	-0.55118	0.418	160	-1.3189	1.000
		-	Audio	PCA	-0.30768	0.418	160	-0.7362	1.000
		-	Audio	SVD	-0.37166	0.418	160	-0.8893	1.000
		-	Audio	UMAP	-0.16504	0.418	160	-0.3949	1.000
		-	Audio	kPCA	-0.54567	0.418	160	-1.3057	1.000
		-	Audio	t-SNE	-0.50893	0.418	160	-1.2177	1.000
	kPCA	-	Video	t-SNE	0.65410	0.418	160	1.5651	1.000
		-	Audio	Autoencoder	-0.07585	0.418	160	-0.1815	1.000
		-	Audio	FA	-0.33906	0.418	160	-0.8113	1.000
		-	Audio	LLE	-0.05632	0.418	160	-0.1348	1.000
		-	Audio	PCA	0.18718	0.418	160	0.4479	1.000
		-	Audio	SVD	0.12321	0.418	160	0.2948	1.000
		-	Audio	UMAP	0.32983	0.418	160	0.7892	1.000
		-	Audio	kPCA	-0.05081	0.418	160	-0.1216	1.000
		-	Audio	t-SNE	-0.01406	0.418	160	-0.0336	1.000
	t-SNE	-	Audio	Autoencoder	-0.72995	0.418	160	-1.7466	0.997
		-	Audio	FA	-0.99316	0.418	160	-2.3764	0.870

Post Hoc Comparisons - Data Type ✻ Reduction Method

Comparison									
Data Type	**Reduction Method**		**Data Type**	**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	Audio	LLE	-0.71042	0.418	160	-1.6999	0.998
		-	Audio	PCA	-0.46692	0.418	160	-1.1172	1.000
		-	Audio	SVD	-0.53089	0.418	160	-1.2703	1.000
		-	Audio	UMAP	-0.32427	0.418	160	-0.7759	1.000
		-	Audio	kPCA	-0.70491	0.418	160	-1.6867	0.999
		-	Audio	t-SNE	-0.66816	0.418	160	-1.5988	0.999
Audio	Autoencoder	-	Audio	FA	-0.26321	0.418	160	-0.6298	1.000
		-	Audio	LLE	0.01953	0.418	160	0.0467	1.000
		-	Audio	PCA	0.26303	0.418	160	0.6294	1.000
		-	Audio	SVD	0.19905	0.418	160	0.4763	1.000
		-	Audio	UMAP	0.40567	0.418	160	0.9707	1.000
		-	Audio	kPCA	0.02504	0.418	160	0.0599	1.000
		-	Audio	t-SNE	0.06178	0.418	160	0.1478	1.000
	FA	-	Audio	LLE	0.28274	0.418	160	0.6765	1.000
		-	Audio	PCA	0.52624	0.418	160	1.2592	1.000
		-	Audio	SVD	0.46226	0.418	160	1.1061	1.000
		-	Audio	UMAP	0.66889	0.418	160	1.6005	0.999
		-	Audio	kPCA	0.28825	0.418	160	0.6897	1.000
		-	Audio	t-SNE	0.32500	0.418	160	0.7776	1.000
	LLE	-	Audio	PCA	0.24350	0.418	160	0.5826	1.000
		-	Audio	SVD	0.17952	0.418	160	0.4296	1.000
		-	Audio	UMAP	0.38615	0.418	160	0.9240	1.000
		-	Audio	kPCA	0.00551	0.418	160	0.0132	1.000
		-	Audio	t-SNE	0.04226	0.418	160	0.1011	1.000
	PCA	-	Audio	SVD	-0.06398	0.418	160	-0.1531	1.000
		-	Audio	UMAP	0.14265	0.418	160	0.3413	1.000
		-	Audio	kPCA	-0.23799	0.418	160	-0.5695	1.000
		-	Audio	t-SNE	-0.20124	0.418	160	-0.4815	1.000
	SVD	-	Audio	UMAP	0.20662	0.418	160	0.4944	1.000
		-	Audio	kPCA	-0.17401	0.418	160	-0.4164	1.000
		-	Audio	t-SNE	-0.13727	0.418	160	-0.3285	1.000
	UMAP	-	Audio	kPCA	-0.38064	0.418	160	-0.9108	1.000
		-	Audio	t-SNE	-0.34389	0.418	160	-0.8229	1.000
	kPCA	-	Audio	t-SNE	0.03675	0.418	160	0.0879	1.000

Based on the post-hoc analysis, the significant differences in these interactions are due to a number of combinations. There are significant differences between the cluster building times of the autoencoder method and the ones generated by Superpixel-Based LLE method on the images datasets.

There are significant differences between the cluster building times of the Superpixel-Based LLE method and the ones generated by UMAP-learn for images, Radial Basis Function based kPCA, multi-core t-SNE methods on the images datasets.

4.2.10 Two-way ANOVA analysis on the interaction between DLs and RMs of HDDs on the AIPS of the *k*HTs.

The table 4.24 shows the results of the two-way ANOVA analysis on the interactions between the dimensionality levels of the datasets and the dimensionality reduction methods, on the performance scores of the existing techniques.

Table 4.24: Two-Way ANOVA analysis on the interaction between DLs and RMS on AIPS of the *k*HTs.

ANOVA - AIPS

	Sum of Squares	df	Mean Square	F	p
Order_magnitude	6.02e-4	1	6.02e-4	0.130	0.719
Reduction Method	1.0173	7	0.14532	31.274	**< .001**
Order_magnitude ✻ Reduction Method	0.0221	7	0.00315	0.678	0.690
Residuals	0.8178	176	0.00465		

4.2.11 Two-way ANOVA analysis on the interaction between DLs and RMs of HDDs on the CBTs of the *k*HTs.

The table 4.25 shows the results of the two-way ANOVA analysis on the interaction between the level of dimensionality of a dataset and the dimensionality reduction method applied, on the cluster building times of the existing techniques.

Table 4.25: Two-Way ANOVA analysis on the interaction between the DLs and RMs on CBTs of the *k*HTs.

ANOVA - Time

	Sum of Squares	df	Mean Square	F	p
Order_magnitude	1616	1	1616.1	25.85	**< .001**

ANOVA - Time

	Sum of Squares	df	Mean Square	F	p
Reduction Method	4075	7	582.1	9.31	**< .001**
Order_magnitude ✻ Reduction Method	2425	7	346.4	5.54	**< .001**
Residuals	11005	176	62.5		

The results show that there are significant interactions between the level of dimensionality of a input dataset and the dimensionality reduction methods on the cluster building times of the existing techniques. Based on this evidence, a post-hoc analysis for this specific interaction is performed and presented in the table 4.26:

Table 4.26: Post-hoc analysis of the interaction between DLs and RMs of HDDs in the two-way ANOVA analysis on the interaction between DLs and RMs of HDDs on CBTs of the *k*HTs.

Post Hoc Comparisons - Order_magnitude ✻ Reduction Method

Comparison									
Order_magnitude	**Reduction Method**		**Order_magnitude**	**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
P3	Autoencoder	-	P3	FA	3.5250	3.23	176	1.0919	0.999
		-	P3	LLE	3.0508	3.23	176	0.9450	1.000
		-	P3	PCA	5.9408	3.23	176	1.8403	0.898
		-	P3	SVD	4.2350	3.23	176	1.3119	0.995
		-	P3	UMAP	-1.2350	3.23	176	-0.3826	1.000
		-	P3	kPCA	6.0950	3.23	176	1.8880	0.878
		-	P3	t-SNE	1.7317	3.23	176	0.5364	1.000
		-	P4	Autoencoder	1.5817	3.23	176	0.4899	1.000
		-	P4	FA	-1.3425	3.23	176	-0.4159	1.000
		-	P4	LLE	-12.3325	3.23	176	-3.8202	0.017
		-	P4	PCA	4.5342	3.23	176	1.4045	0.990
		-	P4	SVD	3.3283	3.23	176	1.0310	1.000
		-	P4	UMAP	-5.3458	3.23	176	-1.6560	0.955
		-	P4	kPCA	4.4733	3.23	176	1.3857	0.991
		-	P4	t-SNE	-17.9733	3.23	176	-5.5675	**< .001**
	FA	-	P3	LLE	-0.4742	3.23	176	-0.1469	1.000
		-	P3	PCA	2.4158	3.23	176	0.7483	1.000
		-	P3	SVD	0.7100	3.23	176	0.2199	1.000
		-	P3	UMAP	-4.7600	3.23	176	-1.4745	0.984
		-	P3	kPCA	2.5700	3.23	176	0.7961	1.000
		-	P3	t-SNE	-1.7933	3.23	176	-0.5555	1.000
		-	P4	Autoencoder	-1.9433	3.23	176	-0.6020	1.000
		-	P4	FA	-4.8675	3.23	176	-1.5078	0.980
		-	P4	LLE	-15.8575	3.23	176	-4.9121	**< .001**
		-	P4	PCA	1.0092	3.23	176	0.3126	1.000
		-	P4	SVD	-0.1967	3.23	176	-0.0609	1.000
		-	P4	UMAP	-8.8708	3.23	176	-2.7479	0.310
		-	P4	kPCA	0.9483	3.23	176	0.2938	1.000

Post Hoc Comparisons - Order_magnitude ✻ Reduction Method

Comparison									
Order_magnitude	**Reduction Method**		**Order_magnitude**	**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	P4	t-SNE	-21.4983	3.23	176	-6.6595	**< .001**
	LLE	-	P3	PCA	2.8900	3.23	176	0.8952	1.000
		-	P3	SVD	1.1842	3.23	176	0.3668	1.000
		-	P3	UMAP	-4.2858	3.23	176	-1.3276	0.994
		-	P3	kPCA	3.0442	3.23	176	0.9430	1.000
		-	P3	t-SNE	-1.3192	3.23	176	-0.4086	1.000
		-	P4	Autoencoder	-1.4692	3.23	176	-0.4551	1.000
		-	P4	FA	-4.3933	3.23	176	-1.3609	0.993
		-	P4	LLE	-15.3833	3.23	176	-4.7652	**< .001**
		-	P4	PCA	1.4833	3.23	176	0.4595	1.000
		-	P4	SVD	0.2775	3.23	176	0.0860	1.000
		-	P4	UMAP	-8.3967	3.23	176	-2.6010	0.405
		-	P4	kPCA	1.4225	3.23	176	0.4406	1.000
		-	P4	t-SNE	-21.0242	3.23	176	-6.5126	**< .001**
	PCA	-	P3	SVD	-1.7058	3.23	176	-0.5284	1.000
		-	P3	UMAP	-7.1758	3.23	176	-2.2228	0.680
		-	P3	kPCA	0.1542	3.23	176	0.0478	1.000
		-	P3	t-SNE	-4.2092	3.23	176	-1.3039	0.995
		-	P4	Autoencoder	-4.3592	3.23	176	-1.3503	0.993
		-	P4	FA	-7.2833	3.23	176	-2.2561	0.656
		-	P4	LLE	-18.2733	3.23	176	-5.6605	**< .001**
		-	P4	PCA	-1.4067	3.23	176	-0.4357	1.000
		-	P4	SVD	-2.6125	3.23	176	-0.8093	1.000
		-	P4	UMAP	-11.2867	3.23	176	-3.4962	**0.047**
		-	P4	kPCA	-1.4675	3.23	176	-0.4546	1.000
		-	P4	t-SNE	-23.9142	3.23	176	-7.4078	**< .001**
	SVD	-	P3	UMAP	-5.4700	3.23	176	-1.6944	0.946
		-	P3	kPCA	1.8600	3.23	176	0.5762	1.000
		-	P3	t-SNE	-2.5033	3.23	176	-0.7754	1.000
		-	P4	Autoencoder	-2.6533	3.23	176	-0.8219	1.000
		-	P4	FA	-5.5775	3.23	176	-1.7277	0.937
		-	P4	LLE	-16.5675	3.23	176	-5.1321	**< .001**
		-	P4	PCA	0.2992	3.23	176	0.0927	1.000
		-	P4	SVD	-0.9067	3.23	176	-0.2809	1.000
		-	P4	UMAP	-9.5808	3.23	176	-2.9678	0.194
		-	P4	kPCA	0.2383	3.23	176	0.0738	1.000
		-	P4	t-SNE	-22.2083	3.23	176	-6.8794	**< .001**
	UMAP	-	P3	kPCA	7.3300	3.23	176	2.2706	0.646
		-	P3	t-SNE	2.9667	3.23	176	0.9190	1.000
		-	P4	Autoencoder	2.8167	3.23	176	0.8725	1.000
		-	P4	FA	-0.1075	3.23	176	-0.0333	1.000
		-	P4	LLE	-11.0975	3.23	176	-3.4376	0.056
		-	P4	PCA	5.7692	3.23	176	1.7871	0.918
		-	P4	SVD	4.5633	3.23	176	1.4136	0.989
		-	P4	UMAP	-4.1108	3.23	176	-1.2734	0.996
		-	P4	kPCA	5.7083	3.23	176	1.7683	0.924
		-	P4	t-SNE	-16.7383	3.23	176	-5.1850	**< .001**
	kPCA	-	P3	t-SNE	-4.3633	3.23	176	-1.3516	0.993
		-	P4	Autoencoder	-4.5133	3.23	176	-1.3981	0.990
		-	P4	FA	-7.4375	3.23	176	-2.3039	0.621
		-	P4	LLE	-18.4275	3.23	176	-5.7082	**< .001**
		-	P4	PCA	-1.5608	3.23	176	-0.4835	1.000
		-	P4	SVD	-2.7667	3.23	176	-0.8570	1.000
		-	P4	UMAP	-11.4408	3.23	176	-3.5440	0.041

Post Hoc Comparisons - Order_magnitude ✻ Reduction Method

Comparison									
Order_magnitude	**Reduction Method**		**Order_magnitude**	**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	P4	kPCA	-1.6217	3.23	176	-0.5023	1.000
		-	P4	t-SNE	-24.0683	3.23	176	-7.4556	**< .001**
	t-SNE	-	P4	Autoencoder	-0.1500	3.23	176	-0.0465	1.000
		-	P4	FA	-3.0742	3.23	176	-0.9523	1.000
		-	P4	LLE	-14.0642	3.23	176	-4.3566	**0.002**
		-	P4	PCA	2.8025	3.23	176	0.8681	1.000
		-	P4	SVD	1.5967	3.23	176	0.4946	1.000
		-	P4	UMAP	-7.0775	3.23	176	-2.1924	0.701
		-	P4	kPCA	2.7417	3.23	176	0.8493	1.000
		-	P4	t-SNE	-19.7050	3.23	176	-6.1040	**< .001**
P4	Autoencoder	-	P4	FA	-2.9242	3.23	176	-0.9058	1.000
		-	P4	LLE	-13.9142	3.23	176	-4.3101	**0.003**
		-	P4	PCA	2.9525	3.23	176	0.9146	1.000
		-	P4	SVD	1.7467	3.23	176	0.5411	1.000
		-	P4	UMAP	-6.9275	3.23	176	-2.1459	0.733
		-	P4	kPCA	2.8917	3.23	176	0.8957	1.000
		-	P4	t-SNE	-19.5550	3.23	176	-6.0575	**< .001**
	FA	-	P4	LLE	-10.9900	3.23	176	-3.4043	0.062
		-	P4	PCA	5.8767	3.23	176	1.8204	0.906
		-	P4	SVD	4.6708	3.23	176	1.4469	0.987
		-	P4	UMAP	-4.0033	3.23	176	-1.2401	0.997
		-	P4	kPCA	5.8158	3.23	176	1.8016	0.913
		-	P4	t-SNE	-16.6308	3.23	176	-5.1517	**< .001**
	LLE	-	P4	PCA	16.8667	3.23	176	5.2247	**< .001**
		-	P4	SVD	15.6608	3.23	176	4.8512	**< .001**
		-	P4	UMAP	6.9867	3.23	176	2.1642	0.720
		-	P4	kPCA	16.8058	3.23	176	5.2059	**< .001**
		-	P4	t-SNE	-5.6408	3.23	176	-1.7473	0.931
	PCA	-	P4	SVD	-1.2058	3.23	176	-0.3735	1.000
		-	P4	UMAP	-9.8800	3.23	176	-3.0605	0.156
		-	P4	kPCA	-0.0608	3.23	176	-0.0188	1.000
		-	P4	t-SNE	-22.5075	3.23	176	-6.9721	**< .001**
	SVD	-	P4	UMAP	-8.6742	3.23	176	-2.6870	0.348
		-	P4	kPCA	1.1450	3.23	176	0.3547	1.000
		-	P4	t-SNE	-21.3017	3.23	176	-6.5986	**< .001**
	UMAP	-	P4	kPCA	9.8192	3.23	176	3.0417	0.163
		-	P4	t-SNE	-12.6275	3.23	176	-3.9116	**0.012**
	kPCA	-	P4	t-SNE	-22.4467	3.23	176	-6.9532	**< .001**

Based on the post-hoc analysis on the dimensionality levels of the input datasets as well as the data dimensionality reduction methods, it is evident that the significant differences are due to significant variations in a number of interactions. There are significant interactions between the use of FA methods on datasets with dimensionality level of P4 and the use of LLE methods on the datasets with similar level of dimensionality.

There are significant interaction between the use of FA methods on datasets with dimensionality of P4 and the use of t-SNE methods on the datasets with similar level of dimensionality. There are significant interactions between the use of LLE methods on datasets with dimensionality level of P3 and the use of t-SNE methods on datasets of dimensionality level of P4. There are significant interactions between the use of PCA methods on datasets with dimensionality level of P3 and the use of UMAP methods on datasets of dimensionality level of P4. There are significant interactions between the use of PCA methods on datasets with dimensionality level of P3 and the use of t-SNE methods on datasets of dimensionality level of P4. There are significant interactions between the use of SVD methods on datasets with dimensionality level of P3 and the use of t-SNE methods on datasets of dimensionality level of P4. There are significant interactions between the use of SVD methods on datasets with dimensionality level of P3 and the use of LLE on datasets with a dimensionality level of P4. There are significant interactions between the use of UMAP methods on datasets with dimensionality level of P3 and the use of t-SNE methods on datasets of dimensionality level of P4. There are significant interactions between the use of kPCA methods on datasets with dimensionality level of P3 and the use of LLE methods on datasets with a dimensionality level of P4. There are significant interactions between the use of kPCA methods on datasets with dimensionality level of P3 and the use of UMAP methods on datasets with a dimensionality level of P4. There are significant interactions between the use of t-SNE methods on datasets with dimensionality level of P3 and the use of LLE methods on datasets with a dimensionality level of P4. There are significant interactions between the use of t-SNE on datasets with dimensionality level of P3 and the use of t-SNE methods on datasets with a dimensionality level of P4. There are significant interactions between the use of autoencoder on datasets with dimensionality level of P4 and the use of LLE methods on datasets with similar level of dimensionality.

There are significant interactions between the use of FA methods on datasets with dimensionality level of P4 and the use of t-SNE methods on datasets with similar level of dimensionality. There are significant interactions between the use of LLE methods on datasets with dimensionality level of P4 and the use of PCA methods on datasets with similar level of dimensionality. There are significant interactions between the use of LLE methods on datasets with dimensionality level of P4 and the use of SVD methods on datasets with similar level of dimensionality. There are significant interactions between the use of LLE methods on datasets with dimensionality level of P4 and the use of kPCA methods on datasets with similar level of dimensionality. There are significant interactions between the use of PCA methods on datasets with dimensionality level of P4 and the use of t-SNE methods on datasets with similar level of dimensionality. There are significant interactions between the use of UMAP methods on datasets with dimensionality level of P4 and the use of t-SNE methods on datasets with similar level of dimensionality. There are significant interactions between the use of kPCA methods on datasets with dimensionality level of P4 and the use of t-SNE methods on datasets with similar level of dimensionality.

4.2.12 Three-way ANOVA on the AIPS based on the *k*HTs, DTs and DLs of HDDs

The table 4.27 shows the results of the three-way ANOVA analysis on the performance scores based on the choice of the technique, type of input datasets and their dimensionality levels.

Table 4.27: Three-way ANOVA analysis on the AIPS based on the choice of *k*HT, DTs and DLs of the HDDs.

ANOVA - AIPS

	Sum of Squares	df	Mean Square	F	p
kHT Technique	0.05472	2	0.02736	3.07865	**0.049**
Data Type	0.25959	3	0.08653	9.73759	**< .001**
Order_magnitude	6.02e-4	1	6.02e-4	0.06775	0.795
kHT Technique ✻ Data Type	0.00616	6	0.00103	0.11552	0.994
kHT Technique ✻ Order_magnitude	1.64e-4	2	8.18e-5	0.00920	0.991
Data Type ✻ Order_magnitude	0.02488	3	0.00829	0.93316	0.426

ANOVA - AIPS

	Sum of Squares	df	Mean Square	F	p
kHT Technique ✻ Data Type ✻ Order_magnitude	0.01872	6	0.00312	0.35110	0.909
Residuals	1.49290	168	0.00889		

The low *p*-value of 0.049 is an indication that the *k*-hyperparameter tuning technique (*k*HT) is a statistically significant factor. At the same time, the *p*-value of less than 0.001 on the data type is an indication that it is an equally statistically significant factor that is strong. In tables 4.28 and 4.29, further post-hoc analyses are presented based on this evidence.

Table 4.28: Post-hoc analysis of the *k*HT techniques in the three-way ANOVA analysis on the AIPS based on the choice of *k*HT, DTs and DLs of HDDs.

Post Hoc Comparisons - kHT Technique

Comparison							
kHT Technique		kHT Technique	Mean Difference	SE	df	t	p_{tukey}
kHT1	-	kHT2	0.0166	0.0167	168	0.994	0.582
	-	kHT3	0.0411	0.0167	168	2.466	**0.039**
kHT2	-	kHT3	0.0245	0.0167	168	1.472	0.307

From the posthoc analysis, it is evident that the specific variations on the *k*-hyperparameter tuning techniques are due to significant differences between the techniques *k*HT1 and *k*HT3 only.

Table 4.29: Post-hoc analysis of DTs in the three-way ANOVA analysis of the AIPS based on the choice of *k*HT, DT and DL of the HDDs.

Post Hoc Comparisons - Data Type

Comparison							
Data Type		Data Type	Mean Difference	SE	df	t	p_{tukey}
Images	-	Text	-0.01188	0.0192	168	-0.617	0.927
	-	Video	0.06375	0.0192	168	3.313	**0.006**
	-	Audio	0.07021	0.0192	168	3.649	**0.002**
Text	-	Video	0.07563	0.0192	168	3.930	**< .001**
	-	Audio	0.08208	0.0192	168	4.266	**< .001**

Post Hoc Comparisons - Data Type

Comparison							
Data Type		**Data Type**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
Video	-	Audio	0.00646	0.0192	168	0.336	0.987

Note. Comparisons are based on estimated marginal means

From the posthoc analysis, it is evident that the specific variations on the performance scores are due to significant differences between the images and video, images and audio, text and video as well as between the text and audio data types.

4.2.13 Three-way ANOVA analysis on the AIPS based on the choice of the *k*HTs, DLs of the HDDs and RMs applied

The table 4.30 shows the results of the three-way ANOVA analysis on the performance scores based on the choice of the technique, dimensionality level of input as well as the dimensionality reduction method applied.

Table 4.30: Three-way ANOVA analysis on the AIPS based on the choice of the *k*HT, DLs of HDDs and RMs applied.

ANOVA - AIPS

	Sum of Squares	**df**	**Mean Square**	**F**	**p**
kHT Technique	0.0547	2	0.02736	5.7090	**0.004**
Order_magnitude	6.02e-4	1	6.02e-4	0.1256	0.724
Reduction Method	1.0173	7	0.14532	30.3259	**< .001**
kHT Technique ✻ Order_magnitude	1.64e-4	2	8.18e-5	0.0171	0.983
kHT Technique ✻ Reduction Method	0.0421	14	0.00300	0.6269	0.839
Order_magnitude ✻ Reduction Method	0.0221	7	0.00315	0.6575	0.708
kHT Technique ✻ Order_magnitude ✻ Reduction Method	0.0308	14	0.00220	0.4595	0.951
Residuals	0.6900	144	0.00479		

The low *p*-value of 0.004 is an indication that the *k*-hyperparameter tuning technique (*k*HT) is statistically significant on the performance scores.

At the same time, the *p*-value of less than 0.001 on the data dimensionality reduction methods is an indication of a strong significance. Based on this evidence, further post-hoc analysis is performed and presented in table 4.31. At the same time, the post-hoc analysis of the data dimensionality reduction methods is done. The results and descriptions are similar to the ones in table 4.9.

Table 4.31: Post-hoc analysis of the *k*HTs in the three-way ANOVA analysis on the AIPS based on the choice of *k*HT, DLs of HDDs and RMs applied.

Post Hoc Comparisons - kHT Technique

Comparison							
kHT Technique		**kHT Technique**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
kHT 1	-	kHT 2	0.0166	0.0122	144	1.35	0.368
	-	kHT 3	0.0411	0.0122	144	3.36	**0.003**
kHT 2	-	kHT 3	0.0245	0.0122	144	2.00	0.115

From the post-hoc analysis, it is evident that the specific variations on the *k*-hyperparameter tuning techniques are due to significant differences between the techniques *k*HT1 and *k*HT3 only.

4.2.14 Three-way ANOVA analysis on the AIPS based on the DT of HDDs, DLs of HDDs and RMs applied

The table 4.32 shows the results of the three-way ANOVA analysis on the performance scores based on the data types, dimensionality level of the input datasets as well as the dimensionality reduction methods applied.

Table 4.32: Three-way ANOVA analysis on the AIPS based on the DTs of HDDs, DLs of HDDs and RMs applied.

ANOVA - AIPS

	Sum of Squares	**df**	**Mean Square**	**F**	**p**
Order_magnitude	6.02e-4	1	6.02e-4	0.279	0.598
Reduction Method	1.0173	7	0.14532	67.412	**< .001**

ANOVA - AIPS

	Sum of Squares	df	Mean Square	F	p
Data Type	0.2596	3	0.08653	40.140	**< .001**
Order_magnitude ✻ Reduction Method	0.0221	7	0.00315	1.462	0.187
Order_magnitude ✻ Data Type	0.0249	3	0.00829	3.847	**0.011**
Reduction Method ✻ Data Type	0.2339	21	0.01114	5.167	**< .001**
Order_magnitude ✻ Reduction Method ✻ Data Type	0.0235	21	0.00112	0.519	0.958
Residuals	0.2759	128	0.00216		

The low *p*-value of less than 0.001 on the data dimensionality reduction method is an indication that the data dimensionality reduction method has a strong significance. At the same time, the *p*-value of less than 0.001 on the data types is an indication that it is a strong significance too. Moreover, there are strong significant interactions effects between the data dimensionality reduction methods and the data types input datasets. There are also statistically significant interactions between dimensionality levels of the input datasets and the data dimensionality reduction methods applied. Further post-hoc analyses are performed and the results presented in table 4.33. The post hoc analysis on the dimensionality reduction methods is performed and the results presented in table 4.34. The post hoc analysis and descriptions on the interactions between the data types and dimensionality reduction methods is also performed. The results and the descriptions for this specific post hoc analysis are similar to the ones presented in table 4.20.

Table 4.33: Post-hoc analysis of the DTs in the three-way ANOVA analysis on the AIPS based on the DTs of HDDs, DLs of HDDs and RMs applied.

Post Hoc Comparisons - Data Type

Comparison							
Data Type		**Data Type**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
Images	-	Text	-0.01188	0.00948	128	-1.253	0.594
	-	Video	0.06375	0.00948	128	6.726	**< .001**
	-	Audio	0.07021	0.00948	128	7.408	**< .001**
Text	-	Video	0.07563	0.00948	128	7.979	**< .001**
	-	Audio	0.08208	0.00948	128	8.661	**< .001**

Post Hoc Comparisons - Data Type

Comparison							
Data Type		**Data Type**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
Video	-	Audio	0.00646	0.00948	128	0.681	0.904

From the post-hoc analysis on the data types, it is clear that the significant differences in the data type are due to the differences between images and video, images and audio, text and audio as well as between text and audio.

Table 4.34: Posthoc analysis of the RMs in the three-way ANOVA analysis on the AIPS based on the DTs of HDDs, DLs of HDDs and the RMs applied.

Post Hoc Comparisons - Reduction Method

Comparison							
Reduction Method		**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
Autoencoder	-	FA	0.09458	0.0134	128	7.0568	**< .001**
	-	LLE	0.24500	0.0134	128	18.2793	**< .001**
	-	PCA	0.14583	0.0134	128	10.8805	**< .001**
	-	SVD	0.16167	0.0134	128	12.0619	**< .001**
	-	UMAP	0.04500	0.0134	128	3.3574	**0.022**
	-	kPCA	0.16292	0.0134	128	12.1551	**< .001**
	-	t-SNE	0.07583	0.0134	128	5.6579	**< .001**
FA	-	LLE	0.15042	0.0134	128	11.2225	**< .001**
	-	PCA	0.05125	0.0134	128	3.8237	**0.005**
	-	SVD	0.06708	0.0134	128	5.0050	**< .001**
	-	UMAP	-0.04958	0.0134	128	-3.6994	0.008
	-	kPCA	0.06833	0.0134	128	5.0983	**< .001**
	-	t-SNE	-0.01875	0.0134	128	-1.3989	0.856
LLE	-	PCA	-0.09917	0.0134	128	-7.3988	**< .001**
	-	SVD	-0.08333	0.0134	128	-6.2174	**< .001**
	-	UMAP	-0.20000	0.0134	128	-14.9219	**< .001**
	-	kPCA	-0.08208	0.0134	128	-6.1242	**< .001**
	-	t-SNE	-0.16917	0.0134	128	-12.6214	**< .001**
PCA	-	SVD	0.01583	0.0134	128	1.1813	0.936
	-	UMAP	-0.10083	0.0134	128	-7.5231	**< .001**
	-	kPCA	0.01708	0.0134	128	1.2746	0.907
	-	t-SNE	-0.07000	0.0134	128	-5.2227	**< .001**
SVD	-	UMAP	-0.11667	0.0134	128	-8.7044	**< .001**
	-	kPCA	0.00125	0.0134	128	0.0933	1.000
	-	t-SNE	-0.08583	0.0134	128	-6.4040	**< .001**
UMAP	-	kPCA	0.11792	0.0134	128	8.7977	**< .001**

Post Hoc Comparisons - Reduction Method

Comparison							
Reduction Method		**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
	-	t-SNE	0.03083	0.0134	128	2.3005	0.302
kPCA	-	t-SNE	-0.08708	0.0134	128	-6.4972	**< .001**

From the posthoc analysis, it is evident that the specific variations on the performance scores due to data dimensionality reduction methods are caused by significant differences between the autoencoder and FA methods, autoencoder and LLE methods, autoencoder and PCA methods, autoencoder and SVD methods, autoencoder and UMAP methods, autoencoder and kPCA methods, autoencoder and t-SNE methods, FA and LLE methods, FA and SVD methods, FA and kPCA methods, LLE and PCA methods, LLE and SVD methods, LLE and UMAP methods, LLE and PCA methods, LLE and t-SNE methods, PCA and UMAP methods, PCA and t-SNE methods, SVD and UMAP methods, SVD and t-SNE, UMAP methods and kPCA methods as well as between kPCA and t-SNE methods.

4.2.15 Three-way ANOVA analysis on the AIPS based on the choice of the *k*HTs, DTs of HDDs and RMs applied.

The table 4.35 shows the results of the three-way ANOVA analysis on the performance score based on the choice of the *k*-hyperparameter tuning technique, type of datasets as well as the data dimensionality reduction method used.

Table 4.35: Three-way ANOVA analysis on the AIPS based on the choice of the *k*HTs, DTs of HDDs and RMs applied.

ANOVA - AIPS

	Sum of Squares	**df**	**Mean Square**	**F**	**p**
Data Type	0.25959	3	0.08653	52.911	**< .001**
Reduction Method	1.01726	7	0.14532	88.860	**< .001**
kHT Technique	0.05472	2	0.02736	16.728	**< .001**

ANOVA - AIPS

	Sum of Squares	df	Mean Square	F	p
Data Type ✻ Reduction Method	0.23391	21	0.01114	6.811	**< .001**
Data Type ✻ kHT Technique	0.00616	6	0.00103	0.628	0.708
Reduction Method ✻ kHT Technique	0.04206	14	0.00300	1.837	0.064
Data Type ✻ Reduction Method ✻ kHT Technique	0.08703	42	0.00207	1.267	0.171
Residuals	0.15700	96	0.00164		

The low *p*-value of less than 0.001 on the data dimensionality reduction method is an indication that the data dimensionality reduction method is statistically significant on the performance scores. At the same time, the *p*-value of less than 0.001 on both the data types and *k*-hyperparameter tuning technique are an indication that they are equally statistically significant. Moreover, there are significant interaction effects between data dimensionality reduction method and the type of the dataset input. In tables 4.36, 4.37, 4.38, and 4.39, a further post hoc analysis is performed in order to investigate on the source of specific variations on the above significant factors and interactions.

Table 4.36: Posthoc analysis of the *k*HTs in the three-way ANOVA analysis on the AIPS based on the choice of the *k*HTs, DTs of HDDs and RMs applied.

Post Hoc Comparisons - kHT Technique

Comparison							
kHT Technique		**kHT Technique**	**Mean Difference**	**SE**	**df**	**t**	$\mathbf{p_{tukey}}$
kHT 1	-	kHT 2	0.0166	0.00715	96.0	2.32	0.058
	-	kHT 3	0.0411	0.00715	96.0	5.75	**< .001**
kHT 2	-	kHT 3	0.0245	0.00715	96.0	3.43	**0.003**

From the post-hoc analysis, it is evident that the significant differences in the *k*-hyperparameter tuning techniques are due to the significant differences between the techniques *k*HT1 and *k*HT3 as well as between *k*HT2 and *k*HT3.

Table 4.37: Posthoc analysis of the RMs in the three-way ANOVA analysis on the AIPS based on the choice of the *k*HTs, DTs of HDDs and RMs applied.

Post Hoc Comparisons - Reduction Method

Comparison							
Reduction Method		**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
Autoencoder	-	FA	0.09458	0.0117	96.0	8.102	**< .001**
	-	LLE	0.24500	0.0117	96.0	20.987	**< .001**
	-	PCA	0.14583	0.0117	96.0	12.492	**< .001**
	-	SVD	0.16167	0.0117	96.0	13.848	**< .001**
	-	UMAP	0.04500	0.0117	96.0	3.855	**0.005**
	-	kPCA	0.16292	0.0117	96.0	13.955	**< .001**
	-	t-SNE	0.07583	0.0117	96.0	6.496	**< .001**
FA	-	LLE	0.15042	0.0117	96.0	12.885	**< .001**
	-	PCA	0.05125	0.0117	96.0	4.390	**< .001**
	-	SVD	0.06708	0.0117	96.0	5.746	**< .001**
	-	UMAP	-0.04958	0.0117	96.0	-4.247	**0.001**
	-	kPCA	0.06833	0.0117	96.0	5.853	**< .001**
	-	t-SNE	-0.01875	0.0117	96.0	-1.606	0.746
LLE	-	PCA	-0.09917	0.0117	96.0	-8.495	**< .001**
	-	SVD	-0.08333	0.0117	96.0	-7.138	**< .001**
	-	UMAP	-0.20000	0.0117	96.0	-17.132	**< .001**
	-	kPCA	-0.08208	0.0117	96.0	-7.031	**< .001**
	-	t-SNE	-0.16917	0.0117	96.0	-14.491	**< .001**
PCA	-	SVD	0.01583	0.0117	96.0	1.356	0.874
	-	UMAP	-0.10083	0.0117	96.0	-8.637	**< .001**
	-	kPCA	0.01708	0.0117	96.0	1.463	0.825
	-	t-SNE	-0.07000	0.0117	96.0	-5.996	**< .001**
SVD	-	UMAP	-0.11667	0.0117	96.0	-9.994	**< .001**
	-	kPCA	0.00125	0.0117	96.0	0.107	1.000
	-	t-SNE	-0.08583	0.0117	96.0	-7.352	**< .001**
UMAP	-	kPCA	0.11792	0.0117	96.0	10.101	**< .001**
	-	t-SNE	0.03083	0.0117	96.0	2.641	0.154
kPCA	-	t-SNE	-0.08708	0.0117	96.0	-7.460	**< .001**

From the post-hoc results, it is clear that the significant differences in the data dimensionality reduction methods are due to significant variations between several methods. There are significant differences between the autoencoder and all the other methods. There are significant differences between the FA and the LLE methods, FA and PCA methods, FA and SVD methods as well as between the FA and UMAP methods.

There are significant differences between the LLE and the PCA methods, LLE and SVD methods, LLE and UMAP methods, LLE and kPCA methods as well as between LLE and t-SNE methods. There are significant differences between the PCA and UMAP methods as well as between PCA and t-SNE methods. There are significant differences between the SVD and the UMAP methods as well as between the SVD and t-SNE methods. There are significant differences between the UMAP and the kPCA methods. There are significant differences between the kPCA and t-SNE methods.

Table 4.38: Post-hoc analysis of the DTs in HDDs in the three-way ANOVA analysis on the AIPS based on the choice of the *k*HTs, DTs of HDDs and RMs applied.

Post Hoc Comparisons - Data Type

Comparison							
Data Type		**Data Type**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
Images	-	Text	-0.01188	0.00825	96.0	-1.439	0.479
	-	Video	0.06375	0.00825	96.0	7.723	**< .001**
	-	Audio	0.07021	0.00825	96.0	8.505	**< .001**
Text	-	Video	0.07563	0.00825	96.0	9.161	**< .001**
	-	Audio	0.08208	0.00825	96.0	9.944	**< .001**
Video	-	Audio	0.00646	0.00825	96.0	0.782	0.862

From the above post-hoc analysis, it is clear that the variations on the data types are due to specific differences between images and video, images and audio, text and video as well as between text and audio.

Table 4.39: Post-hoc analysis of interaction between DTs and RMs in the three-way ANOVA analysis on the AIPS based on the choice of the *k*HTs, DTs of HDDs and RMs.

Post Hoc Comparisons - Reduction Method ✻ Data Type

Comparison									
Reduction Method	**Data Type**		**Reduction Method**	**Data Type**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
Autoencoder	Images	-	Autoencoder	Text	0.00333	0.0233	96.0	0.1428	1.000
		-	Autoencoder	Video	0.02333	0.0233	96.0	0.9994	1.000
		-	Autoencoder	Audio	0.02833	0.0233	96.0	1.2135	1.000
		-	FA	Images	0.12167	0.0233	96.0	5.2110	**< .001**
		-	FA	Text	0.09667	0.0233	96.0	4.1402	**0.024**
		-	FA	Video	0.10167	0.0233	96.0	4.3544	**0.012**

Post Hoc Comparisons - Reduction Method ✻ Data Type

Comparison									
Reduction Method	**Data Type**		**Reduction Method**	**Data Type**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	FA	Audio	0.11333	0.0233	96.0	4.8540	**0.002**
		-	LLE	Images	0.19667	0.0233	96.0	8.4232	**< .001**
		-	LLE	Text	0.20333	0.0233	96.0	8.7087	**< .001**
		-	LLE	Video	0.32167	0.0233	96.0	13.7769	**< .001**
		-	LLE	Audio	0.31333	0.0233	96.0	13.4200	**< .001**
		-	PCA	Images	0.07833	0.0233	96.0	3.3550	0.212
		-	PCA	Text	0.06500	0.0233	96.0	2.7839	0.594
		-	PCA	Video	0.24167	0.0233	96.0	10.3505	**< .001**
		-	PCA	Audio	0.25333	0.0233	96.0	10.8502	**< .001**
		-	SVD	Images	0.13833	0.0233	96.0	5.9248	**< .001**
		-	SVD	Text	0.10000	0.0233	96.0	4.2830	**0.015**
		-	SVD	Video	0.23167	0.0233	96.0	9.9222	**< .001**
		-	SVD	Audio	0.23167	0.0233	96.0	9.9222	**< .001**
		-	UMAP	Images	0.06500	0.0233	96.0	2.7839	0.594
		-	UMAP	Text	0.05500	0.0233	96.0	2.3556	0.875
		-	UMAP	Video	0.05333	0.0233	96.0	2.2843	0.906
		-	UMAP	Audio	0.06167	0.0233	96.0	2.6412	0.701
		-	kPCA	Images	0.10667	0.0233	96.0	4.5685	**0.005**
		-	kPCA	Text	0.09667	0.0233	96.0	4.1402	**0.024**
		-	kPCA	Video	0.24167	0.0233	96.0	10.3505	**< .001**
		-	kPCA	Audio	0.26167	0.0233	96.0	11.2071	**< .001**
		-	t-SNE	Images	0.09000	0.0233	96.0	3.8547	0.057
		-	t-SNE	Text	0.08167	0.0233	96.0	3.4978	0.151
		-	t-SNE	Video	0.09167	0.0233	96.0	3.9261	0.046
		-	t-SNE	Audio	0.09500	0.0233	96.0	4.0688	0.030
	Text	-	Autoencoder	Video	0.02000	0.0233	96.0	0.8566	1.000
		-	Autoencoder	Audio	0.02500	0.0233	96.0	1.0707	1.000
		-	FA	Images	0.11833	0.0233	96.0	5.0682	**< .001**
		-	FA	Text	0.09333	0.0233	96.0	3.9975	**0.037**
		-	FA	Video	0.09833	0.0233	96.0	4.2116	**0.019**
		-	FA	Audio	0.11000	0.0233	96.0	4.7113	**0.003**
		-	LLE	Images	0.19333	0.0233	96.0	8.2804	**< .001**
		-	LLE	Text	0.20000	0.0233	96.0	8.5660	**< .001**
		-	LLE	Video	0.31833	0.0233	96.0	13.6342	**< .001**
		-	LLE	Audio	0.31000	0.0233	96.0	13.2772	**< .001**
		-	PCA	Images	0.07500	0.0233	96.0	3.2122	0.289
		-	PCA	Text	0.06167	0.0233	96.0	2.6412	0.701
		-	PCA	Video	0.23833	0.0233	96.0	10.2078	**< .001**
		-	PCA	Audio	0.25000	0.0233	96.0	10.7075	**< .001**
		-	SVD	Images	0.13500	0.0233	96.0	5.7820	**< .001**
		-	SVD	Text	0.09667	0.0233	96.0	4.1402	**0.024**
		-	SVD	Video	0.22833	0.0233	96.0	9.7795	**< .001**
		-	SVD	Audio	0.22833	0.0233	96.0	9.7795	**< .001**
		-	UMAP	Images	0.06167	0.0233	96.0	2.6412	0.701
		-	UMAP	Text	0.05167	0.0233	96.0	2.2129	0.932
		-	UMAP	Video	0.05000	0.0233	96.0	2.1415	0.952
		-	UMAP	Audio	0.05833	0.0233	96.0	2.4984	0.797
		-	kPCA	Images	0.10333	0.0233	96.0	4.4257	**0.009**
		-	kPCA	Text	0.09333	0.0233	96.0	3.9975	**0.037**
		-	kPCA	Video	0.23833	0.0233	96.0	10.2078	**< .001**
		-	kPCA	Audio	0.25833	0.0233	96.0	11.0644	**< .001**
		-	t-SNE	Images	0.08667	0.0233	96.0	3.7119	0.086
		-	t-SNE	Text	0.07833	0.0233	96.0	3.3550	0.212
		-	t-SNE	Video	0.08833	0.0233	96.0	3.7833	0.070

Post Hoc Comparisons - Reduction Method ✻ Data Type

Comparison									
Reduction Method	**Data Type**		**Reduction Method**	**Data Type**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	t-SNE	Audio	0.09167	0.0233	96.0	3.9261	**0.046**
	Video	-	Autoencoder	Audio	0.00500	0.0233	96.0	0.2141	1.000
		-	FA	Images	0.09833	0.0233	96.0	4.2116	0.019
		-	FA	Text	0.07333	0.0233	96.0	3.1409	0.333
		-	FA	Video	0.07833	0.0233	96.0	3.3550	0.212
		-	FA	Audio	0.09000	0.0233	96.0	3.8547	0.057
		-	LLE	Images	0.17333	0.0233	96.0	7.4238	**< .001**
		-	LLE	Text	0.18000	0.0233	96.0	7.7094	**< .001**
		-	LLE	Video	0.29833	0.0233	96.0	12.7776	**< .001**
		-	LLE	Audio	0.29000	0.0233	96.0	12.4207	**< .001**
		-	PCA	Images	0.05500	0.0233	96.0	2.3556	0.875
		-	PCA	Text	0.04167	0.0233	96.0	1.7846	0.996
		-	PCA	Video	0.21833	0.0233	96.0	9.3512	**< .001**
		-	PCA	Audio	0.23000	0.0233	96.0	9.8509	**< .001**
		-	SVD	Images	0.11500	0.0233	96.0	4.9254	**0.001**
		-	SVD	Text	0.07667	0.0233	96.0	3.2836	0.249
		-	SVD	Video	0.20833	0.0233	96.0	8.9229	**< .001**
		-	SVD	Audio	0.20833	0.0233	96.0	8.9229	**< .001**
		-	UMAP	Images	0.04167	0.0233	96.0	1.7846	0.996
		-	UMAP	Text	0.03167	0.0233	96.0	1.3563	1.000
		-	UMAP	Video	0.03000	0.0233	96.0	1.2849	1.000
		-	UMAP	Audio	0.03833	0.0233	96.0	1.6418	0.999
		-	kPCA	Images	0.08333	0.0233	96.0	3.5692	0.126
		-	kPCA	Text	0.07333	0.0233	96.0	3.1409	0.333
		-	kPCA	Video	0.21833	0.0233	96.0	9.3512	**< .001**
		-	kPCA	Audio	0.23833	0.0233	96.0	10.2078	**< .001**
		-	t-SNE	Images	0.06667	0.0233	96.0	2.8553	0.539
		-	t-SNE	Text	0.05833	0.0233	96.0	2.4984	0.797
		-	t-SNE	Video	0.06833	0.0233	96.0	2.9267	0.485
		-	t-SNE	Audio	0.07167	0.0233	96.0	3.0695	0.381
	Audio	-	FA	Images	0.09333	0.0233	96.0	3.9975	0.037
		-	FA	Text	0.06833	0.0233	96.0	2.9267	0.485
		-	FA	Video	0.07333	0.0233	96.0	3.1409	0.333
		-	FA	Audio	0.08500	0.0233	96.0	3.6405	0.104
		-	LLE	Images	0.16833	0.0233	96.0	7.2097	**< .001**
		-	LLE	Text	0.17500	0.0233	96.0	7.4952	**< .001**
		-	LLE	Video	0.29333	0.0233	96.0	12.5634	**< .001**
		-	LLE	Audio	0.28500	0.0233	96.0	12.2065	**< .001**
		-	PCA	Images	0.05000	0.0233	96.0	2.1415	0.952
		-	PCA	Text	0.03667	0.0233	96.0	1.5704	0.999
		-	PCA	Video	0.21333	0.0233	96.0	9.1370	**< .001**
		-	PCA	Audio	0.22500	0.0233	96.0	9.6367	**< .001**
		-	SVD	Images	0.11000	0.0233	96.0	4.7113	**0.003**
		-	SVD	Text	0.07167	0.0233	96.0	3.0695	0.381
		-	SVD	Video	0.20333	0.0233	96.0	8.7087	**< .001**
		-	SVD	Audio	0.20333	0.0233	96.0	8.7087	**< .001**
		-	UMAP	Images	0.03667	0.0233	96.0	1.5704	0.999
		-	UMAP	Text	0.02667	0.0233	96.0	1.1421	1.000
		-	UMAP	Video	0.02500	0.0233	96.0	1.0707	1.000
		-	UMAP	Audio	0.03333	0.0233	96.0	1.4277	1.000
		-	kPCA	Images	0.07833	0.0233	96.0	3.3550	0.212
		-	kPCA	Text	0.06833	0.0233	96.0	2.9267	0.485
		-	kPCA	Video	0.21333	0.0233	96.0	9.1370	**< .001**
		-	kPCA	Audio	0.23333	0.0233	96.0	9.9936	**< .001**

Post Hoc Comparisons - Reduction Method ✻ Data Type

Comparison									
Reduction Method	**Data Type**		**Reduction Method**	**Data Type**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	t-SNE	Images	0.06167	0.0233	96.0	2.6412	0.701
		-	t-SNE	Text	0.05333	0.0233	96.0	2.2843	0.906
		-	t-SNE	Video	0.06333	0.0233	96.0	2.7126	0.648
		-	t-SNE	Audio	0.06667	0.0233	96.0	2.8553	0.539
FA	Images	-	FA	Text	-0.02500	0.0233	96.0	-1.0707	1.000
		-	FA	Video	-0.02000	0.0233	96.0	-0.8566	1.000
		-	FA	Audio	-0.00833	0.0233	96.0	-0.3569	1.000
		-	LLE	Images	0.07500	0.0233	96.0	3.2122	0.289
		-	LLE	Text	0.08167	0.0233	96.0	3.4978	0.151
		-	LLE	Video	0.20000	0.0233	96.0	8.5660	**< .001**
		-	LLE	Audio	0.19167	0.0233	96.0	8.2091	**< .001**
		-	PCA	Images	-0.04333	0.0233	96.0	-1.8560	0.992
		-	PCA	Text	-0.05667	0.0233	96.0	-2.4270	0.839
		-	PCA	Video	0.12000	0.0233	96.0	5.1396	**< .001**
		-	PCA	Audio	0.13167	0.0233	96.0	5.6393	**< .001**
		-	SVD	Images	0.01667	0.0233	96.0	0.7138	1.000
		-	SVD	Text	-0.02167	0.0233	96.0	-0.9280	1.000
		-	SVD	Video	0.11000	0.0233	96.0	4.7113	**0.003**
		-	SVD	Audio	0.11000	0.0233	96.0	4.7113	**0.003**
		-	UMAP	Images	-0.05667	0.0233	96.0	-2.4270	0.839
		-	UMAP	Text	-0.06667	0.0233	96.0	-2.8553	0.539
		-	UMAP	Video	-0.06833	0.0233	96.0	-2.9267	0.485
		-	UMAP	Audio	-0.06000	0.0233	96.0	-2.5698	0.751
		-	kPCA	Images	-0.01500	0.0233	96.0	-0.6424	1.000
		-	kPCA	Text	-0.02500	0.0233	96.0	-1.0707	1.000
		-	kPCA	Video	0.12000	0.0233	96.0	5.1396	**< .001**
		-	kPCA	Audio	0.14000	0.0233	96.0	5.9962	**< .001**
		-	t-SNE	Images	-0.03167	0.0233	96.0	-1.3563	1.000
		-	t-SNE	Text	-0.04000	0.0233	96.0	-1.7132	0.998
		-	t-SNE	Video	-0.03000	0.0233	96.0	-1.2849	1.000
		-	t-SNE	Audio	-0.02667	0.0233	96.0	-1.1421	1.000
	Text	-	FA	Video	0.00500	0.0233	96.0	0.2141	1.000
		-	FA	Audio	0.01667	0.0233	96.0	0.7138	1.000
		-	LLE	Images	0.10000	0.0233	96.0	4.2830	**0.015**
		-	LLE	Text	0.10667	0.0233	96.0	4.5685	**0.005**
		-	LLE	Video	0.22500	0.0233	96.0	9.6367	**< .001**
		-	LLE	Audio	0.21667	0.0233	96.0	9.2798	**< .001**
		-	PCA	Images	-0.01833	0.0233	96.0	-0.7852	1.000
		-	PCA	Text	-0.03167	0.0233	96.0	-1.3563	1.000
		-	PCA	Video	0.14500	0.0233	96.0	6.2103	**< .001**
		-	PCA	Audio	0.15667	0.0233	96.0	6.7100	**< .001**
		-	SVD	Images	0.04167	0.0233	96.0	1.7846	0.996
		-	SVD	Text	0.00333	0.0233	96.0	0.1428	1.000
		-	SVD	Video	0.13500	0.0233	96.0	5.7820	**< .001**
		-	SVD	Audio	0.13500	0.0233	96.0	5.7820	**< .001**
		-	UMAP	Images	-0.03167	0.0233	96.0	-1.3563	1.000
		-	UMAP	Text	-0.04167	0.0233	96.0	-1.7846	0.996
		-	UMAP	Video	-0.04333	0.0233	96.0	-1.8560	0.992
		-	UMAP	Audio	-0.03500	0.0233	96.0	-1.4990	1.000
		-	kPCA	Images	0.01000	0.0233	96.0	0.4283	1.000
		-	kPCA	Text	-3.82e−17	0.0233	96.0	-1.63e−15	1.000
		-	kPCA	Video	0.14500	0.0233	96.0	6.2103	**< .001**
		-	kPCA	Audio	0.16500	0.0233	96.0	7.0669	**< .001**
		-	t-SNE	Images	-0.00667	0.0233	96.0	-0.2855	1.000

Post Hoc Comparisons - Reduction Method ✻ Data Type

Comparison									
Reduction Method	**Data Type**		**Reduction Method**	**Data Type**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	t-SNE	Text	-0.01500	0.0233	96.0	-0.6424	1.000
		-	t-SNE	Video	-0.00500	0.0233	96.0	-0.2141	1.000
		-	t-SNE	Audio	-0.00167	0.0233	96.0	-0.0714	1.000
	Video	-	FA	Audio	0.01167	0.0233	96.0	0.4997	1.000
		-	LLE	Images	0.09500	0.0233	96.0	4.0688	**0.030**
		-	LLE	Text	0.10167	0.0233	96.0	4.3544	**0.012**
		-	LLE	Video	0.22000	0.0233	96.0	9.4226	**< .001**
		-	LLE	Audio	0.21167	0.0233	96.0	9.0656	**< .001**
		-	PCA	Images	-0.02333	0.0233	96.0	-0.9994	1.000
		-	PCA	Text	-0.03667	0.0233	96.0	-1.5704	0.999
		-	PCA	Video	0.14000	0.0233	96.0	5.9962	**< .001**
		-	PCA	Audio	0.15167	0.0233	96.0	6.4959	**< .001**
		-	SVD	Images	0.03667	0.0233	96.0	1.5704	0.999
		-	SVD	Text	-0.00167	0.0233	96.0	-0.0714	1.000
		-	SVD	Video	0.13000	0.0233	96.0	5.5679	**< .001**
		-	SVD	Audio	0.13000	0.0233	96.0	5.5679	**< .001**
		-	UMAP	Images	-0.03667	0.0233	96.0	-1.5704	0.999
		-	UMAP	Text	-0.04667	0.0233	96.0	-1.9987	0.979
		-	UMAP	Video	-0.04833	0.0233	96.0	-2.0701	0.967
		-	UMAP	Audio	-0.04000	0.0233	96.0	-1.7132	0.998
		-	kPCA	Images	0.00500	0.0233	96.0	0.2141	1.000
		-	kPCA	Text	-0.00500	0.0233	96.0	-0.2141	1.000
		-	kPCA	Video	0.14000	0.0233	96.0	5.9962	**< .001**
		-	kPCA	Audio	0.16000	0.0233	96.0	6.8528	**< .001**
		-	t-SNE	Images	-0.01167	0.0233	96.0	-0.4997	1.000
		-	t-SNE	Text	-0.02000	0.0233	96.0	-0.8566	1.000
		-	t-SNE	Video	-0.01000	0.0233	96.0	-0.4283	1.000
		-	t-SNE	Audio	-0.00667	0.0233	96.0	-0.2855	1.000
	Audio	-	LLE	Images	0.08333	0.0233	96.0	3.5692	0.126
		-	LLE	Text	0.09000	0.0233	96.0	3.8547	0.057
		-	LLE	Video	0.20833	0.0233	96.0	8.9229	**< .001**
		-	LLE	Audio	0.20000	0.0233	96.0	8.5660	**< .001**
		-	PCA	Images	-0.03500	0.0233	96.0	-1.4990	1.000
		-	PCA	Text	-0.04833	0.0233	96.0	-2.0701	0.967
		-	PCA	Video	0.12833	0.0233	96.0	5.4965	**< .001**
		-	PCA	Audio	0.14000	0.0233	96.0	5.9962	**< .001**
		-	SVD	Images	0.02500	0.0233	96.0	1.0707	1.000
		-	SVD	Text	-0.01333	0.0233	96.0	-0.5711	1.000
		-	SVD	Video	0.11833	0.0233	96.0	5.0682	**< .001**
		-	SVD	Audio	0.11833	0.0233	96.0	5.0682	**< .001**
		-	UMAP	Images	-0.04833	0.0233	96.0	-2.0701	0.967
		-	UMAP	Text	-0.05833	0.0233	96.0	-2.4984	0.797
		-	UMAP	Video	-0.06000	0.0233	96.0	-2.5698	0.751
		-	UMAP	Audio	-0.05167	0.0233	96.0	-2.2129	0.932
		-	kPCA	Images	-0.00667	0.0233	96.0	-0.2855	1.000
		-	kPCA	Text	-0.01667	0.0233	96.0	-0.7138	1.000
		-	kPCA	Video	0.12833	0.0233	96.0	5.4965	**< .001**
		-	kPCA	Audio	0.14833	0.0233	96.0	6.3531	**< .001**
		-	t-SNE	Images	-0.02333	0.0233	96.0	-0.9994	1.000
		-	t-SNE	Text	-0.03167	0.0233	96.0	-1.3563	1.000
		-	t-SNE	Video	-0.02167	0.0233	96.0	-0.9280	1.000
		-	t-SNE	Audio	-0.01833	0.0233	96.0	-0.7852	1.000
LLE	Images	-	LLE	Text	0.00667	0.0233	96.0	0.2855	1.000
		-	LLE	Video	0.12500	0.0233	96.0	5.3537	**< .001**

Post Hoc Comparisons - Reduction Method ✻ Data Type

Comparison									
Reduction Method	**Data Type**		**Reduction Method**	**Data Type**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	LLE	Audio	0.11667	0.0233	96.0	4.9968	0.001
		-	PCA	Images	-0.11833	0.0233	96.0	-5.0682	**< .001**
		-	PCA	Text	-0.13167	0.0233	96.0	-5.6393	**< .001**
		-	PCA	Video	0.04500	0.0233	96.0	1.9273	0.987
		-	PCA	Audio	0.05667	0.0233	96.0	2.4270	0.839
		-	SVD	Images	-0.05833	0.0233	96.0	-2.4984	0.797
		-	SVD	Text	-0.09667	0.0233	96.0	-4.1402	**0.024**
		-	SVD	Video	0.03500	0.0233	96.0	1.4990	1.000
		-	SVD	Audio	0.03500	0.0233	96.0	1.4990	1.000
		-	UMAP	Images	-0.13167	0.0233	96.0	-5.6393	**< .001**
		-	UMAP	Text	-0.14167	0.0233	96.0	-6.0676	**< .001**
		-	UMAP	Video	-0.14333	0.0233	96.0	-6.1389	**< .001**
		-	UMAP	Audio	-0.13500	0.0233	96.0	-5.7820	**< .001**
		-	kPCA	Images	-0.09000	0.0233	96.0	-3.8547	0.057
		-	kPCA	Text	-0.10000	0.0233	96.0	-4.2830	**0.015**
		-	kPCA	Video	0.04500	0.0233	96.0	1.9273	0.987
		-	kPCA	Audio	0.06500	0.0233	96.0	2.7839	0.594
		-	t-SNE	Images	-0.10667	0.0233	96.0	-4.5685	**0.005**
		-	t-SNE	Text	-0.11500	0.0233	96.0	-4.9254	**0.001**
		-	t-SNE	Video	-0.10500	0.0233	96.0	-4.4971	**0.007**
		-	t-SNE	Audio	-0.10167	0.0233	96.0	-4.3544	**0.012**
	Text	-	LLE	Video	0.11833	0.0233	96.0	5.0682	**< .001**
		-	LLE	Audio	0.11000	0.0233	96.0	4.7113	**0.003**
		-	PCA	Images	-0.12500	0.0233	96.0	-5.3537	**< .001**
		-	PCA	Text	-0.13833	0.0233	96.0	-5.9248	**< .001**
		-	PCA	Video	0.03833	0.0233	96.0	1.6418	0.999
		-	PCA	Audio	0.05000	0.0233	96.0	2.1415	0.952
		-	SVD	Images	-0.06500	0.0233	96.0	-2.7839	0.594
		-	SVD	Text	-0.10333	0.0233	96.0	-4.4257	0.009
		-	SVD	Video	0.02833	0.0233	96.0	1.2135	1.000
		-	SVD	Audio	0.02833	0.0233	96.0	1.2135	1.000
		-	UMAP	Images	-0.13833	0.0233	96.0	-5.9248	**< .001**
		-	UMAP	Text	-0.14833	0.0233	96.0	-6.3531	**< .001**
		-	UMAP	Video	-0.15000	0.0233	96.0	-6.4245	**< .001**
		-	UMAP	Audio	-0.14167	0.0233	96.0	-6.0676	**< .001**
		-	kPCA	Images	-0.09667	0.0233	96.0	-4.1402	**0.024**
		-	kPCA	Text	-0.10667	0.0233	96.0	-4.5685	**0.005**
		-	kPCA	Video	0.03833	0.0233	96.0	1.6418	0.999
		-	kPCA	Audio	0.05833	0.0233	96.0	2.4984	0.797
		-	t-SNE	Images	-0.11333	0.0233	96.0	-4.8540	**0.002**
		-	t-SNE	Text	-0.12167	0.0233	96.0	-5.2110	**< .001**
		-	t-SNE	Video	-0.11167	0.0233	96.0	-4.7827	**0.002**
		-	t-SNE	Audio	-0.10833	0.0233	96.0	-4.6399	**0.004**
	Video	-	LLE	Audio	-0.00833	0.0233	96.0	-0.3569	1.000
		-	PCA	Images	-0.24333	0.0233	96.0	-10.4219	**< .001**
		-	PCA	Text	-0.25667	0.0233	96.0	-10.9930	**< .001**
		-	PCA	Video	-0.08000	0.0233	96.0	-3.4264	0.180
		-	PCA	Audio	-0.06833	0.0233	96.0	-2.9267	0.485
		-	SVD	Images	-0.18333	0.0233	96.0	-7.8521	**< .001**
		-	SVD	Text	-0.22167	0.0233	96.0	-9.4939	**< .001**
		-	SVD	Video	-0.09000	0.0233	96.0	-3.8547	0.057
		-	SVD	Audio	-0.09000	0.0233	96.0	-3.8547	0.057
		-	UMAP	Images	-0.25667	0.0233	96.0	-10.9930	**< .001**
		-	UMAP	Text	-0.26667	0.0233	96.0	-11.4213	**< .001**

Post Hoc Comparisons - Reduction Method ✻ Data Type

		Comparison							
Reduction Method	**Data Type**		**Reduction Method**	**Data Type**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	UMAP	Video	-0.26833	0.0233	96.0	-11.4927	**< .001**
		-	UMAP	Audio	-0.26000	0.0233	96.0	-11.1358	**< .001**
		-	kPCA	Images	-0.21500	0.0233	96.0	-9.2084	**< .001**
		-	kPCA	Text	-0.22500	0.0233	96.0	-9.6367	**< .001**
		-	kPCA	Video	-0.08000	0.0233	96.0	-3.4264	0.180
		-	kPCA	Audio	-0.06000	0.0233	96.0	-2.5698	0.751
		-	t-SNE	Images	-0.23167	0.0233	96.0	-9.9222	**< .001**
		-	t-SNE	Text	-0.24000	0.0233	96.0	-10.2792	**< .001**
		-	t-SNE	Video	-0.23000	0.0233	96.0	-9.8509	**< .001**
		-	t-SNE	Audio	-0.22667	0.0233	96.0	-9.7081	**< .001**
	Audio	-	PCA	Images	-0.23500	0.0233	96.0	-10.0650	**< .001**
		-	PCA	Text	-0.24833	0.0233	96.0	-10.6361	**< .001**
		-	PCA	Video	-0.07167	0.0233	96.0	-3.0695	0.381
		-	PCA	Audio	-0.06000	0.0233	96.0	-2.5698	0.751
		-	SVD	Images	-0.17500	0.0233	96.0	-7.4952	**< .001**
		-	SVD	Text	-0.21333	0.0233	96.0	-9.1370	**< .001**
		-	SVD	Video	-0.08167	0.0233	96.0	-3.4978	0.151
		-	SVD	Audio	-0.08167	0.0233	96.0	-3.4978	0.151
		-	UMAP	Images	-0.24833	0.0233	96.0	-10.6361	**< .001**
		-	UMAP	Text	-0.25833	0.0233	96.0	-11.0644	**< .001**
		-	UMAP	Video	-0.26000	0.0233	96.0	-11.1358	**< .001**
		-	UMAP	Audio	-0.25167	0.0233	96.0	-10.7788	**< .001**
		-	kPCA	Images	-0.20667	0.0233	96.0	-8.8515	**< .001**
		-	kPCA	Text	-0.21667	0.0233	96.0	-9.2798	**< .001**
		-	kPCA	Video	-0.07167	0.0233	96.0	-3.0695	0.381
		-	kPCA	Audio	-0.05167	0.0233	96.0	-2.2129	0.932
		-	t-SNE	Images	-0.22333	0.0233	96.0	-9.5653	**< .001**
		-	t-SNE	Text	-0.23167	0.0233	96.0	-9.9222	**< .001**
		-	t-SNE	Video	-0.22167	0.0233	96.0	-9.4939	**< .001**
		-	t-SNE	Audio	-0.21833	0.0233	96.0	-9.3512	**< .001**
PCA	Images	-	PCA	Text	-0.01333	0.0233	96.0	-0.5711	**1.000**
		-	PCA	Video	0.16333	0.0233	96.0	6.9955	**< .001**
		-	PCA	Audio	0.17500	0.0233	96.0	7.4952	**< .001**
		-	SVD	Images	0.06000	0.0233	96.0	2.5698	0.751
		-	SVD	Text	0.02167	0.0233	96.0	0.9280	1.000
		-	SVD	Video	0.15333	0.0233	96.0	6.5672	**< .001**
		-	SVD	Audio	0.15333	0.0233	96.0	6.5672	**< .001**
		-	UMAP	Images	-0.01333	0.0233	96.0	-0.5711	1.000
		-	UMAP	Text	-0.02333	0.0233	96.0	-0.9994	1.000
		-	UMAP	Video	-0.02500	0.0233	96.0	-1.0707	1.000
		-	UMAP	Audio	-0.01667	0.0233	96.0	-0.7138	1.000
		-	kPCA	Images	0.02833	0.0233	96.0	1.2135	1.000
		-	kPCA	Text	0.01833	0.0233	96.0	0.7852	1.000
		-	kPCA	Video	0.16333	0.0233	96.0	6.9955	**< .001**
		-	kPCA	Audio	0.18333	0.0233	96.0	7.8521	**< .001**
		-	t-SNE	Images	0.01167	0.0233	96.0	0.4997	1.000
		-	t-SNE	Text	0.00333	0.0233	96.0	0.1428	1.000
		-	t-SNE	Video	0.01333	0.0233	96.0	0.5711	1.000
		-	t-SNE	Audio	0.01667	0.0233	96.0	0.7138	1.000
	Text	-	PCA	Video	0.17667	0.0233	96.0	7.5666	**< .001**
		-	PCA	Audio	0.18833	0.0233	96.0	8.0663	**< .001**
		-	SVD	Images	0.07333	0.0233	96.0	3.1409	0.333
		-	SVD	Text	0.03500	0.0233	96.0	1.4990	1.000
		-	SVD	Video	0.16667	0.0233	96.0	7.1383	**< .001**

Post Hoc Comparisons - Reduction Method ✻ Data Type

Comparison									
Reduction Method	**Data Type**		**Reduction Method**	**Data Type**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	SVD	Audio	0.16667	0.0233	96.0	7.1383	**< .001**
		-	UMAP	Images	6.94e-18	0.0233	96.0	2.97e-16	1.000
		-	UMAP	Text	-0.01000	0.0233	96.0	-0.4283	1.000
		-	UMAP	Video	-0.01167	0.0233	96.0	-0.4997	1.000
		-	UMAP	Audio	-0.00333	0.0233	96.0	-0.1428	1.000
		-	kPCA	Images	0.04167	0.0233	96.0	1.7846	0.996
		-	kPCA	Text	0.03167	0.0233	96.0	1.3563	1.000
		-	kPCA	Video	0.17667	0.0233	96.0	7.5666	**< .001**
		-	kPCA	Audio	0.19667	0.0233	96.0	8.4232	**< .001**
		-	t-SNE	Images	0.02500	0.0233	96.0	1.0707	1.000
		-	t-SNE	Text	0.01667	0.0233	96.0	0.7138	1.000
		-	t-SNE	Video	0.02667	0.0233	96.0	1.1421	1.000
		-	t-SNE	Audio	0.03000	0.0233	96.0	1.2849	1.000
	Video	-	PCA	Audio	0.01167	0.0233	96.0	0.4997	1.000
		-	SVD	Images	-0.10333	0.0233	96.0	-4.4257	0.009
		-	SVD	Text	-0.14167	0.0233	96.0	-6.0676	**< .001**
		-	SVD	Video	-0.01000	0.0233	96.0	-0.4283	1.000
		-	SVD	Audio	-0.01000	0.0233	96.0	-0.4283	1.000
		-	UMAP	Images	-0.17667	0.0233	96.0	-7.5666	**< .001**
		-	UMAP	Text	-0.18667	0.0233	96.0	-7.9949	**< .001**
		-	UMAP	Video	-0.18833	0.0233	96.0	-8.0663	**< .001**
		-	UMAP	Audio	-0.18000	0.0233	96.0	-7.7094	**< .001**
		-	kPCA	Images	-0.13500	0.0233	96.0	-5.7820	**< .001**
		-	kPCA	Text	-0.14500	0.0233	96.0	-6.2103	**< .001**
		-	kPCA	Video	-1.04e−17	0.0233	96.0	-4.46e−16	1.000
		-	kPCA	Audio	0.02000	0.0233	96.0	0.8566	1.000
		-	t-SNE	Images	-0.15167	0.0233	96.0	-6.4959	**< .001**
		-	t-SNE	Text	-0.16000	0.0233	96.0	-6.8528	**< .001**
		-	t-SNE	Video	-0.15000	0.0233	96.0	-6.4245	**< .001**
		-	t-SNE	Audio	-0.14667	0.0233	96.0	-6.2817	**< .001**
	Audio	-	SVD	Images	-0.11500	0.0233	96.0	-4.9254	**0.001**
		-	SVD	Text	-0.15333	0.0233	96.0	-6.5672	**< .001**
		-	SVD	Video	-0.02167	0.0233	96.0	-0.9280	1.000
		-	SVD	Audio	-0.02167	0.0233	96.0	-0.9280	1.000
		-	UMAP	Images	-0.18833	0.0233	96.0	-8.0663	**< .001**
		-	UMAP	Text	-0.19833	0.0233	96.0	-8.4946	**< .001**
		-	UMAP	Video	-0.20000	0.0233	96.0	-8.5660	**< .001**
		-	UMAP	Audio	-0.19167	0.0233	96.0	-8.2091	**< .001**
		-	kPCA	Images	-0.14667	0.0233	96.0	-6.2817	**< .001**
		-	kPCA	Text	-0.15667	0.0233	96.0	-6.7100	**< .001**
		-	kPCA	Video	-0.01167	0.0233	96.0	-0.4997	1.000
		-	kPCA	Audio	0.00833	0.0233	96.0	0.3569	1.000
		-	t-SNE	Images	-0.16333	0.0233	96.0	-6.9955	**< .001**
		-	t-SNE	Text	-0.17167	0.0233	96.0	-7.3525	**< .001**
		-	t-SNE	Video	-0.16167	0.0233	96.0	-6.9242	**< .001**
		-	t-SNE	Audio	-0.15833	0.0233	96.0	-6.7814	**< .001**
SVD	Images	-	SVD	Text	-0.03833	0.0233	96.0	-1.6418	0.999
		-	SVD	Video	0.09333	0.0233	96.0	3.9975	**0.037**
		-	SVD	Audio	0.09333	0.0233	96.0	3.9975	**0.037**
		-	UMAP	Images	-0.07333	0.0233	96.0	-3.1409	0.333
		-	UMAP	Text	-0.08333	0.0233	96.0	-3.5692	0.126
		-	UMAP	Video	-0.08500	0.0233	96.0	-3.6405	0.104
		-	UMAP	Audio	-0.07667	0.0233	96.0	-3.2836	0.249
		-	kPCA	Images	-0.03167	0.0233	96.0	-1.3563	1.000

Post Hoc Comparisons - Reduction Method ✻ Data Type

		Comparison							
Reduction Method	**Data Type**		**Reduction Method**	**Data Type**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	kPCA	Text	-0.04167	0.0233	96.0	-1.7846	0.996
		-	kPCA	Video	0.10333	0.0233	96.0	4.4257	**0.009**
		-	kPCA	Audio	0.12333	0.0233	96.0	5.2823	**< .001**
		-	t-SNE	Images	-0.04833	0.0233	96.0	-2.0701	0.967
		-	t-SNE	Text	-0.05667	0.0233	96.0	-2.4270	0.839
		-	t-SNE	Video	-0.04667	0.0233	96.0	-1.9987	0.979
		-	t-SNE	Audio	-0.04333	0.0233	96.0	-1.8560	0.992
	Text	-	SVD	Video	0.13167	0.0233	96.0	5.6393	**< .001**
		-	SVD	Audio	0.13167	0.0233	96.0	5.6393	**< .001**
		-	UMAP	Images	-0.03500	0.0233	96.0	-1.4990	1.000
		-	UMAP	Text	-0.04500	0.0233	96.0	-1.9273	0.987
		-	UMAP	Video	-0.04667	0.0233	96.0	-1.9987	0.979
		-	UMAP	Audio	-0.03833	0.0233	96.0	-1.6418	0.999
		-	kPCA	Images	0.00667	0.0233	96.0	0.2855	1.000
		-	kPCA	Text	-0.00333	0.0233	96.0	-0.1428	1.000
		-	kPCA	Video	0.14167	0.0233	96.0	6.0676	**< .001**
		-	kPCA	Audio	0.16167	0.0233	96.0	6.9242	**< .001**
		-	t-SNE	Images	-0.01000	0.0233	96.0	-0.4283	1.000
		-	t-SNE	Text	-0.01833	0.0233	96.0	-0.7852	1.000
		-	t-SNE	Video	-0.00833	0.0233	96.0	-0.3569	1.000
		-	t-SNE	Audio	-0.00500	0.0233	96.0	-0.2141	1.000
	Video	-	SVD	Audio	3.30e-17	0.0233	96.0	1.41e-15	1.000
		-	UMAP	Images	-0.16667	0.0233	96.0	-7.1383	**< .001**
		-	UMAP	Text	-0.17667	0.0233	96.0	-7.5666	**< .001**
		-	UMAP	Video	-0.17833	0.0233	96.0	-7.6380	**< .001**
		-	UMAP	Audio	-0.17000	0.0233	96.0	-7.2811	**< .001**
		-	kPCA	Images	-0.12500	0.0233	96.0	-5.3537	**< .001**
		-	kPCA	Text	-0.13500	0.0233	96.0	-5.7820	**< .001**
		-	kPCA	Video	0.01000	0.0233	96.0	0.4283	1.000
		-	kPCA	Audio	0.03000	0.0233	96.0	1.2849	1.000
		-	t-SNE	Images	-0.14167	0.0233	96.0	-6.0676	**< .001**
		-	t-SNE	Text	-0.15000	0.0233	96.0	-6.4245	**< .001**
		-	t-SNE	Video	-0.14000	0.0233	96.0	-5.9962	**< .001**
		-	t-SNE	Audio	-0.13667	0.0233	96.0	-5.8534	**< .001**
	Audio	-	UMAP	Images	-0.16667	0.0233	96.0	-7.1383	**< .001**
		-	UMAP	Text	-0.17667	0.0233	96.0	-7.5666	**< .001**
		-	UMAP	Video	-0.17833	0.0233	96.0	-7.6380	**< .001**
		-	UMAP	Audio	-0.17000	0.0233	96.0	-7.2811	**< .001**
		-	kPCA	Images	-0.12500	0.0233	96.0	-5.3537	**< .001**
		-	kPCA	Text	-0.13500	0.0233	96.0	-5.7820	**< .001**
		-	kPCA	Video	0.01000	0.0233	96.0	0.4283	1.000
		-	kPCA	Audio	0.03000	0.0233	96.0	1.2849	1.000
		-	t-SNE	Images	-0.14167	0.0233	96.0	-6.0676	**< .001**
		-	t-SNE	Text	-0.15000	0.0233	96.0	-6.4245	**< .001**
		-	t-SNE	Video	-0.14000	0.0233	96.0	-5.9962	**< .001**
		-	t-SNE	Audio	-0.13667	0.0233	96.0	-5.8534	**< .001**
UMAP	Images	-	UMAP	Text	-0.01000	0.0233	96.0	-0.4283	1.000
		-	UMAP	Video	-0.01167	0.0233	96.0	-0.4997	1.000
		-	UMAP	Audio	-0.00333	0.0233	96.0	-0.1428	1.000
		-	kPCA	Images	0.04167	0.0233	96.0	1.7846	0.996
		-	kPCA	Text	0.03167	0.0233	96.0	1.3563	1.000
		-	kPCA	Video	0.17667	0.0233	96.0	7.5666	**< .001**
		-	kPCA	Audio	0.19667	0.0233	96.0	8.4232	**< .001**
		-	t-SNE	Images	0.02500	0.0233	96.0	1.0707	1.000

Post Hoc Comparisons - Reduction Method ✻ Data Type

Comparison									
Reduction Method	**Data Type**		**Reduction Method**	**Data Type**	**Mean Difference**	**SE**	**df**	**t**	p_{tukey}
		-	t-SNE	Text	0.01667	0.0233	96.0	0.7138	1.000
		-	t-SNE	Video	0.02667	0.0233	96.0	1.1421	1.000
		-	t-SNE	Audio	0.03000	0.0233	96.0	1.2849	1.000
	Text	-	UMAP	Video	-0.00167	0.0233	96.0	-0.0714	1.000
		-	UMAP	Audio	0.00667	0.0233	96.0	0.2855	1.000
		-	kPCA	Images	0.05167	0.0233	96.0	2.2129	0.932
		-	kPCA	Text	0.04167	0.0233	96.0	1.7846	0.996
		-	kPCA	Video	0.18667	0.0233	96.0	7.9949	**< .001**
		-	kPCA	Audio	0.20667	0.0233	96.0	8.8515	**< .001**
		-	t-SNE	Images	0.03500	0.0233	96.0	1.4990	1.000
		-	t-SNE	Text	0.02667	0.0233	96.0	1.1421	1.000
		-	t-SNE	Video	0.03667	0.0233	96.0	1.5704	0.999
		-	t-SNE	Audio	0.04000	0.0233	96.0	1.7132	0.998
	Video	-	UMAP	Audio	0.00833	0.0233	96.0	0.3569	1.000
		-	kPCA	Images	0.05333	0.0233	96.0	2.2843	0.906
		-	kPCA	Text	0.04333	0.0233	96.0	1.8560	0.992
		-	kPCA	Video	0.18833	0.0233	96.0	8.0663	**< .001**
		-	kPCA	Audio	0.20833	0.0233	96.0	8.9229	**< .001**
		-	t-SNE	Images	0.03667	0.0233	96.0	1.5704	0.999
		-	t-SNE	Text	0.02833	0.0233	96.0	1.2135	1.000
		-	t-SNE	Video	0.03833	0.0233	96.0	1.6418	0.999
		-	t-SNE	Audio	0.04167	0.0233	96.0	1.7846	0.996
	Audio	-	kPCA	Images	0.04500	0.0233	96.0	1.9273	0.987
		-	kPCA	Text	0.03500	0.0233	96.0	1.4990	1.000
		-	kPCA	Video	0.18000	0.0233	96.0	7.7094	**< .001**
		-	kPCA	Audio	0.20000	0.0233	96.0	8.5660	**< .001**
		-	t-SNE	Images	0.02833	0.0233	96.0	1.2135	1.000
		-	t-SNE	Text	0.02000	0.0233	96.0	0.8566	1.000
		-	t-SNE	Video	0.03000	0.0233	96.0	1.2849	1.000
		-	t-SNE	Audio	0.03333	0.0233	96.0	1.4277	1.000
kPCA	Images	-	kPCA	Text	-0.01000	0.0233	96.0	-0.4283	1.000
		-	kPCA	Video	0.13500	0.0233	96.0	5.7820	**< .001**
		-	kPCA	Audio	0.15500	0.0233	96.0	6.6386	**< .001**
		-	t-SNE	Images	-0.01667	0.0233	96.0	-0.7138	1.000
		-	t-SNE	Text	-0.02500	0.0233	96.0	-1.0707	1.000
		-	t-SNE	Video	-0.01500	0.0233	96.0	-0.6424	1.000
		-	t-SNE	Audio	-0.01167	0.0233	96.0	-0.4997	1.000
	Text	-	kPCA	Video	0.14500	0.0233	96.0	6.2103	**< .001**
		-	kPCA	Audio	0.16500	0.0233	96.0	7.0669	**< .001**
		-	t-SNE	Images	-0.00667	0.0233	96.0	-0.2855	1.000
		-	t-SNE	Text	-0.01500	0.0233	96.0	-0.6424	1.000
		-	t-SNE	Video	-0.00500	0.0233	96.0	-0.2141	1.000
		-	t-SNE	Audio	-0.00167	0.0233	96.0	-0.0714	1.000
	Video	-	kPCA	Audio	0.02000	0.0233	96.0	0.8566	1.000
		-	t-SNE	Images	-0.15167	0.0233	96.0	-6.4959	**< .001**
		-	t-SNE	Text	-0.16000	0.0233	96.0	-6.8528	**< .001**
		-	t-SNE	Video	-0.15000	0.0233	96.0	-6.4245	**< .001**
		-	t-SNE	Audio	-0.14667	0.0233	96.0	-6.2817	**< .001**
	Audio	-	t-SNE	Images	-0.17167	0.0233	96.0	-7.3525	**< .001**
		-	t-SNE	Text	-0.18000	0.0233	96.0	-7.7094	**< .001**
		-	t-SNE	Video	-0.17000	0.0233	96.0	-7.2811	**< .001**
		-	t-SNE	Audio	-0.16667	0.0233	96.0	-7.1383	**< .001**
t-SNE	Images	-	t-SNE	Text	-0.00833	0.0233	96.0	-0.3569	1.000
		-	t-SNE	Video	0.00167	0.0233	96.0	0.0714	1.000

Post Hoc Comparisons - Reduction Method ✻ Data Type

Comparison									
Reduction Method	**Data Type**		**Reduction Method**	**Data Type**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
		-	t-SNE	Audio	0.00500	0.0233	96.0	0.2141	1.000
	Text	-	t-SNE	Video	0.01000	0.0233	96.0	0.4283	1.000
		-	t-SNE	Audio	0.01333	0.0233	96.0	0.5711	1.000
	Video	-	t-SNE	Audio	0.00333	0.0233	96.0	0.1428	1.000

From the post-hoc analysis, it is clear that the specific differences on the performance scores on the interactions between the data dimensionality reduction methods and the data type are due to significant differences on several methods in different data types. There are significant differences between the autoencoders and the FA based Factorization machines on images, autoencoders and Superpixel-Based LLE on images, autoencoders and economy SVD on images, as well as between the autoencoders and Radial Basis Function based kPCA on images. There are significant differences between the Superpixel-Based LLE and PCA with ZCA whitening on images, Superpixel-Based LLE and UMAP-learn on images as well as between Superpixel-Based LLE and multi-core t-SNE on images. There are significant differences between the autoencoders and the exploratory FA on text datasets, autoencoders and standard LLE on text datasets, autoencoders and randomized SVD on text datasets as well as between autoencoders and polynomial function based kPCA on text datasets. There are significant differences between exploratory FA and standard LLE on text datasets. There are significant differences between the standard LLE and standard PCA on text datasets, standard LLE and randomized SVD on text datasets, standard LLE and standard UMAP on text datasets, standard LLE and polynomial function based kPCA on text datasets as well as between standard LLE and standard t-SNE on text datasets. There are significant differences between the autoencoders and the Spatial Temporal LLE on video datasets, autoencoders and PCA with Spatiotemporal Cubes on video datasets, autoencoders and the Economy SVD on video datasets as well as between autoencoders and the Radial Basis Function based kPCA on video datasets.

There are significant differences between FA based Factorization machine and Spatial Temporal LLE on video datasets, FA based Factorization machines and PCA with Spatiotemporal Cubes on video datasets, FA based Factorization machines and the Economy SVD on video datasets as well as between FA based Factorization machines and Radial Basis Function based kPCA on video datasets. There are significant differences between the PCA with Spatiotemporal Cubes and Spatio-Temporal UMAP on video datasets as well as between PCA with Spatiotemporal Cubes and Multicore t-SNE on video datasets. There are significant differences between the Economy SVD and Spatio-Temporal UMAP on video datasets as well as between the Economy SVD and Multicore t-SNE on video datasets. There are significant differences between Spatio-Temporal UMAP and Radial Basis Function based kPCA on video datasets. There are significant differences between Radial Basis Function based kPCA and Multicore t-SNE on video datasets.

There are significant differences between the autoencoders and the Spectrogram-Based LLE on audio datasets, autoencoders and PCA Filter Bank on audio datasets, autoencoders and Economy SVD on audio datasets as well as between autoencoders and Radial Basis Function based kPCA on audio datasets. There are significant differences between FA based Factorization machines and Spectrogram-Based LLE on audio datasets, FA based Factorization machines and PCA Filter Bank on audio datasets, FA based Factorization machines and Economy SVD on audio datasets as well as between FA based Factorization machines and Radial Basis Function based kPCA on audio datasets. There are significant differences between Spectrogram-Based LLE and Spectrogram UMAP on audio datasets as well as between Spectrogram-Based LLE and Multicore t-SNE on audio datasets. There are significant differences between PCA Filter Bank and Spectrogram UMAP on audio datasets as well as between PCA Filter Bank and Multicore t-SNE on audio datasets.

There are significant differences between Economy SVD and Spectrogram UMAP on audio datasets as well as between Economy SVD and Multicore t-SNE on audio datasets. There are significant differences between Radial Basis Function based kPCA and Multicore t-SNE on audio datasets.

4.2.16 Three-way ANOVA analysis on CBTs of the *k*HTs, DTs and DLs of HDDs

The table 4.40 shows the results of the three-way ANOVA analysis on the cluster building times based on the choice of the existing technique, data type of the input dataset and its level of dimensionality.

Table 4.40: Three-way ANOVA analysis on CBTs of the *k*HTs, DTs and DLs of HDDs.

ANOVA - Time(s)

	Sum of Squares	df	Mean Square	F	p
kHT Technique	419.0500	2	209.52498	3.33605	**0.038**
Data Type	5147.2786	3	1715.75955	27.31824	**< .001**
Order_magnitude	1616.1123	1	1616.11230	25.73167	**< .001**
kHT Technique ✻ Data Type	0.2147	6	0.03578	5.70e-4	1.000
kHT Technique ✻ Order_magnitude	0.0190	2	0.00949	1.51e-4	1.000
Data Type ✻ Order_magnitude	1385.8647	3	461.95491	7.35522	**< .001**
kHT Technique ✻ Data Type ✻ Order_magnitude	0.5781	6	0.09635	0.00153	1.000
Residuals	10551.4687	168	62.80636		

The low *p*-value of 0.038 on the *k*HT technique is an indication that the *k*-hyperparameter tuning technique is a statistically significant factor on the cluster building times. The *p*-value of less than 0.001 on the data type is an indication that the data type of the dataset is an equally a statistically significant factor. The low *p*-value of less than 0.001 on the order of magnitude is an indication that the dimensionality level of the dataset input is a statistically significant factor. Further post hoc analyses are done in order to investigate on the source of the specific variations. The post hoc analysis on the techniques is done and the results presented in table 4.41.

The post hoc analysis on the data types is done and the results presented in table 4.42. The post hoc analysis on the dimensionality levels is done and the results presented in table 4.43. The post-hoc analysis and description of the interactions between data types and dimensionality levels are similar to the ones in table 4.18.

Table 4.41: Post-hoc analysis of the *k*HTs in the three-way ANOVA analysis on CBTs of the *k*HTs, DTs and DLs of HDDs.

Post Hoc Comparisons - kHT Technique

Comparison							
kHT Technique		**kHT Technique**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
kHT 1	-	kHT 2	-2.19	1.40	168	-1.563	0.265
	-	kHT 3	-3.59	1.40	168	-2.563	**0.030**
kHT 2	-	kHT 3	-1.40	1.40	168	-1.000	0.578

From the post-hoc analysis, it is evident that the significant differences in the cluster building times among the different *k*-hyperparameter tuning techniques are due to the significant differences in the cluster building times between the techniques *k*HT1 and *k*HT3 only.

Table 4.42: Post-hoc analysis of the DTs of HDDs in the three-way ANOVA analysis on CBTs of the *k*HTs, DTs and DLs of HDDs.

Post Hoc Comparisons - Data Type

Comparison							
Data Type		**Data Type**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
Images	-	Text	10.208	1.62	168	6.310	**< .001**
	-	Video	-3.458	1.62	168	-2.138	0.145
	-	Audio	-0.638	1.62	168	-0.394	0.979
Text	-	Video	-13.666	1.62	168	-8.448	**< .001**
	-	Audio	-10.846	1.62	168	-6.705	**< .001**
Video	-	Audio	2.820	1.62	168	1.743	0.305

From the above post-hoc analysis, it is clear that the variations on the data type are due to specific differences between images and text datasets, text and video datasets as well as between text and audio datasets.

Table 4.43: Post-hoc analysis of the DLs of HDDs in the three-way ANOVA analysis on CBTs of the *k*HTs, DTs and DLs of HDDs.

Post Hoc Comparisons - Order_magnitude

Comparison							
Order_magnitude		**Order_magnitude**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
P3	-	P4	-5.80	1.14	168	-5.07	**< .001**

From the post-hoc analysis, the significant differences in the cluster building times of the different dimensionality levels is due to the significant differences in the number of features between the datasets of different dimensionality levels.

4.2.17 Three-way ANOVA analysis on the CBTs of the *k*HTs, DLs of HDDs and the RMs applied

The table 4.44 shows the results of the three-way ANOVA analysis on the cluster building times based on the choice of the existing *k*-hyperparameter tuning technique, level of dimensionality of the input dataset as well as the data dimensionality reduction method applied.

Table 4.44: Three-way ANOVA analysis on CBTs of the *k*HT techniques, DLs of HDDs and RMs applied.

ANOVA - Time(s)

	Sum of Squares	**df**	**Mean Square**	**F**	**p**
kHT Technique	419.0500	2	209.52498	2.85085	0.061
Order_magnitude	1616.1123	1	1616.11230	21.98920	**< .001**
Reduction Method	4074.7430	7	582.10614	7.92027	**< .001**
kHT Technique ✻ Order_magnitude	0.0190	2	0.00949	1.29e-4	1.000
kHT Technique ✻ Reduction Method	1.4089	14	0.10064	0.00137	1.000
Order_magnitude ✻ Reduction Method	2424.6339	7	346.37628	4.71288	**< .001**
kHT Technique ✻ Order_magnitude ✻ Reduction Method	1.2317	14	0.08798	0.00120	1.000
Residuals	10583.3872	144	73.49574		

The *p*-value of less than 0.001 on the order magnitude is an indication that the dimensionality level of the dataset is a statistically strong significant factor. The low *p*-value of less than 0.001 on the reduction method is an indication that the data dimensionality reduction method is a statistically significant factor that is strong too. There are statistically significant interactions between the dimensionality levels of the input datasets and the data dimensionality reduction methods applied on it. Further post hoc analyses are performed and presented. The post hoc analysis on the dimensionality levels is performed and presented in table 4.45. The post hoc analysis on the data dimensionality reduction methods is performed and presented in table 4.46. The post-hoc analysis of the interaction between the dataset's dimensionality levels and the dimensionality reduction methods are also performed. The results and the descriptions are similar to the ones presented in table 4.26.

Table 4.45: Post hoc analysis of the DLs of HDDs in the three-way ANOVA analysis on CBTs of the *k*HTs, DLs of HDDs and RMs applied.

Post Hoc Comparisons - Order_magnitude

Comparison							
Order_magnitude		**Order_magnitude**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
P3	-	P4	-5.80	1.24	144	-4.69	**< .001**

From the post-hoc analysis, the significant differences in the cluster building times of the different dimensionality levels are due to the significant differences in the number of features between the datasets of dimensionality level P3 and the number of features between the datasets of dimensionality level P4.

Table 4.46: Post hoc analysis of the RMs in the three-way ANOVA analysis on CBTs of the *k*HTs, DLs of HDDs and RMs applied.

Post Hoc Comparisons - Reduction Method

Comparison							
Reduction Method		**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	p_{tukey}
Autoencoder	-	FA	0.3004	2.47	144	0.1214	1.000
	-	LLE	-5.4317	2.47	144	-2.1948	0.361
	-	PCA	4.4467	2.47	144	1.7968	0.623
	-	SVD	2.9908	2.47	144	1.2085	0.928
	-	UMAP	-4.0812	2.47	144	-1.6491	0.720
	-	kPCA	4.4933	2.47	144	1.8156	0.611
	-	t-SNE	-8.9117	2.47	144	-3.6010	0.010
FA	-	LLE	-5.7321	2.47	144	-2.3162	0.292
	-	PCA	4.1462	2.47	144	1.6754	0.703
	-	SVD	2.6904	2.47	144	1.0871	0.959
	-	UMAP	-4.3817	2.47	144	-1.7705	0.641
	-	kPCA	4.1929	2.47	144	1.6942	0.691
	-	t-SNE	-9.2121	2.47	144	-3.7224	**0.007**
LLE	-	PCA	9.8783	2.47	144	3.9916	**0.003**
	-	SVD	8.4225	2.47	144	3.4033	**0.019**
	-	UMAP	1.3504	2.47	144	0.5457	0.999
	-	kPCA	9.9250	2.47	144	4.0104	**0.002**
	-	t-SNE	-3.4800	2.47	144	-1.4062	0.853
PCA	-	SVD	-1.4558	2.47	144	-0.5883	0.999
	-	UMAP	-8.5279	2.47	144	-3.4459	**0.017**
	-	kPCA	0.0467	2.47	144	0.0189	1.000
	-	t-SNE	-13.3583	2.47	144	-5.3977	**< .001**
SVD	-	UMAP	-7.0721	2.47	144	-2.8576	0.089
	-	kPCA	1.5025	2.47	144	0.6071	0.999
	-	t-SNE	-11.9025	2.47	144	-4.8095	**< .001**
UMAP	-	kPCA	8.5746	2.47	144	3.4648	**0.016**
	-	t-SNE	-4.8304	2.47	144	-1.9518	0.518
kPCA	-	t-SNE	-13.4050	2.47	144	-5.4166	**< .001**

From the post-hoc analysis, it is evident that the specific variations in the data dimensionality reduction methods are due to significant differences between several methods. There are significant differences between the autoencoders and t-SNE methods. There are significant differences between FA methods and t-SNE methods.

There are significant differences between the LLE methods and PCA methods, LLE methods and SVD methods as well as between the LLE methods and kPCA methods. There are significant differences between the PCA methods and UMAP methods as well as between the PCA methods and t-SNE methods. There are significant differences between UMAP methods and kPCA methods.

4.2.18 Three-way ANOVA analysis on CBTs of the DTs of HDDs, DLs of HDDs and RMs applied

The table 4.47 shows the results of the three-way ANOVA analysis on the cluster building times based on the data type of the input datasets, its level of dimensionality as well as the data dimensionality reduction methods applied.

Table 4.47: Three-way ANOVA analysis on CBTs of the DTs of HDDs, DLs of HDDs and RMs applied.

ANOVA - Time(s)

	Sum of Squares	df	Mean Square	F	p
Data Type	5147	3	1715.76	511.6	**< .001**
Order_magnitude	1616	1	1616.11	481.9	**< .001**
Reduction Method	4075	7	582.11	173.6	**< .001**
Data Type ✻ Order_magnitude	1386	3	461.95	137.8	**< .001**
Data Type ✻ Reduction Method	2132	21	101.52	30.3	**< .001**
Order_magnitude ✻ Reduction Method	2425	7	346.38	103.3	**< .001**
Data Type ✻ Order_magnitude ✻ Reduction Method	1911	21	90.99	27.1	**< .001**
Residuals	429	128	3.35		

The low *p*-value of less than 0.001 on the data type is an indication that the data type of the dataset is a statistically significant factor on cluster building times. The low *p*-value of less than 0.001 on the order of magnitude is an indication that the dimensionality level of the dataset input is a statistically significant factor.

The *p*-value of less than 0.001 on the reduction method is an indication that the data dimensionality reduction method is an equally a statistically significant factor. There are statistically significant interactions between the data type of the input datasets and their dimensionality levels. There are statistically significant interactions between the data types of the input datasets and the data dimensionality reduction method applied to it. There are statistically significant interactions between the dimensionality levels of the input datasets and the data dimensionality reduction methods applied to it. There are significant interaction effects among the three factors i.e. data type of the input dataset, its dimensionality level and the data dimensionality reduction method applied to it. Based on this evidence, further post hoc analyses are done in order to investigate on the source of these specific variations. The post hoc analyses for the data types, dimensionality levels and the dimensionality reduction methods are done and the results presented in tables 4.48, 4.49 and 4.50, respectively. The post-hoc analysis of the interaction between the dataset's data type and the dimensionality levels is performed. The results and the description are similar to the ones in table 4.18. The post hoc analysis of the interaction between the dataset's dimensionality levels and the data dimensionality reduction methods is performed. The results and the description are similar to the ones presented in table 4.26. The post-hoc analysis of the interaction between the dataset's data type and the data dimensionality reduction methods is performed. The results and descriptions are similar to the ones presented in table 4.23.

Table 4.48: Post-hoc analysis of the DTs in HDDs in the three-way ANOVA analysis on CBTs of the DTs of HDDs, DLs of HDDs and RMs applied.

Post Hoc Comparisons - Data Type

Comparison							
Data Type		**Data Type**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
Images	-	Text	10.208	0.374	128	27.31	**< .001**
	-	Video	-3.458	0.374	128	-9.25	**< .001**
	-	Audio	-0.638	0.374	128	-1.71	0.324

Post Hoc Comparisons - Data Type

Comparison							
Data Type		**Data Type**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
Text	-	Video	-13.666	0.374	128	-36.56	**< .001**
	-	Audio	-10.846	0.374	128	-29.02	**< .001**
Video	-	Audio	2.820	0.374	128	7.54	**< .001**

From the above post-hoc analysis, it is clear that the specific variations on the cluster building times among the different data types are due to significant differences between images and text datasets, images and video datasets, text and video, video and audio datasets as well as between text and audio datasets.

Table 4.49: Post-hoc analysis of the DLs of HDDs in the three-way ANOVA analysis on CBTs of DTs of HDDs, DLs of HDDs and RMs applied.

Post Hoc Comparisons - Order_magnitude

Comparison							
Order_magnitude		**Order_magnitude**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
P3	-	P4	-5.80	0.264	128	-22.0	**< .001**

From the post-hoc analysis, the specific variations in the dataset's dimensionality levels are due to significant differences between datasets of the dimensionality level P3 and the dimensionality level P4.

Table 4.50: Posthoc analysis of the RMs in the three-way ANOVA analysis on CBTs of the DTs of HDDs, DLs of HDDs and RMs applied.

Post Hoc Comparisons - Reduction Method

Comparison							
Reduction Method		**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
Autoencoder	-	FA	0.3004	0.529	128	0.5683	0.999
	-	LLE	-5.4317	0.529	128	-10.2747	**< .001**
	-	PCA	4.4467	0.529	128	8.4115	**< .001**

Post Hoc Comparisons - Reduction Method

Comparison							
Reduction Method		**Reduction Method**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
	-	SVD	2.9908	0.529	128	5.6576	**< .001**
	-	UMAP	-4.0812	0.529	128	-7.7202	**< .001**
	-	kPCA	4.4933	0.529	128	8.4997	**< .001**
	-	t-SNE	-8.9117	0.529	128	-16.8576	**< .001**
FA	-	LLE	-5.7321	0.529	128	-10.8430	**< .001**
	-	PCA	4.1462	0.529	128	7.8432	**< .001**
	-	SVD	2.6904	0.529	128	5.0893	**< .001**
	-	UMAP	-4.3817	0.529	128	-8.2885	**< .001**
	-	kPCA	4.1929	0.529	128	7.9315	**< .001**
	-	t-SNE	-9.2121	0.529	128	-17.4259	**< .001**
LLE	-	PCA	9.8783	0.529	128	18.6862	**< .001**
	-	SVD	8.4225	0.529	128	15.9323	**< .001**
	-	UMAP	1.3504	0.529	128	2.5545	0.183
	-	kPCA	9.9250	0.529	128	18.7745	**< .001**
	-	t-SNE	-3.4800	0.529	128	-6.5829	**< .001**
PCA	-	SVD	-1.4558	0.529	128	-2.7539	0.116
	-	UMAP	-8.5279	0.529	128	-16.1317	**< .001**
	-	kPCA	0.0467	0.529	128	0.0883	1.000
	-	t-SNE	-13.3583	0.529	128	-25.2691	**< .001**
SVD	-	UMAP	-7.0721	0.529	128	-13.3778	**< .001**
	-	kPCA	1.5025	0.529	128	2.8422	0.094
	-	t-SNE	-11.9025	0.529	128	-22.5152	**< .001**
UMAP	-	kPCA	8.5746	0.529	128	16.2200	**< .001**
	-	t-SNE	-4.8304	0.529	128	-9.1374	< .001
kPCA	-	t-SNE	-13.4050	0.529	128	-25.3574	< .001

From the post-hoc analysis, it is evident that the specific variations in the data dimensionality reduction methods are due to significant differences between several methods. There are significant differences between the autoencoders and LLE methods, autoencoder and PCA methods, autoencoder and SVD methods, autoencoder and UMAP methods, autoencoder and kPCA methods and autoencoder and t-SNE methods. There are significant differences between the FA and LLE methods, FA and PCA methods, FA and SVD methods, FA and UMAP methods, FA and kPCA methods as well as between the FA and t-SNE methods. There are significant differences between the LLE and PCA methods, LLE and SVD methods, LLE and kPCA methods and LLE and t-SNE methods. There are significant differences between the PCA and UMAP methods as well as between the PCA and t-SNE methods.There are significant differences between the SVD and UMAP methods well as between the SVD and t-SNE methods. There are significant differences between the kPCA and t-SNE methods.

4.2.19 Three-way ANOVA analysis on the CBTs of the *k*HTs, DTs of HDDs and RMs applied

The table 4.51 shows the results of the three-way ANOVA analysis on the cluster building times based on the choice of the technique, data type of the input datasets as well as the data dimensionality reduction method applied.

Table 4.51: Three-way ANOVA analysis on the CBTs of the *k*HTs, DTs of HDDs and RMs applied.

ANOVA - Time(s)

	Sum of Squares	df	Mean Square	F	p
kHT Technique	419.050	2	209.5250	2.73942	0.070
Data Type	5147.279	3	1715.7595	22.43262	**< .001**
Reduction Method	4074.743	7	582.1061	7.61072	**< .001**
kHT Technique ✻ Data Type	0.215	6	0.0358	4.68e-4	1.000
kHT Technique ✻ Reduction Method	1.409	14	0.1006	0.00132	1.000
Data Type ✻ Reduction Method	2131.910	21	101.5195	1.32731	0.178

ANOVA - Time(s)

	Sum of Squares	df	Mean Square	F	p
kHT Technique ✻ Data Type ✻ Reduction Method	3.418	42	0.0814	0.00106	1.000
Residuals	7342.563	96	76.4850		

The low *p*-value of less than 0.001 on the data type is an indication that the data type of the dataset input is a statistically significant factor that is strong. The *p*-value of less than 0.001 on the reduction method is an indication that the data dimensionality reduction method is an equally statistically significant factor. There are significant interaction effects among the three factors combined together i.e. *k*-hyperparameter tuning technique, data type of the dataset input and the data dimensionality reduction method applied to it. Further post hoc analyses are performed. The Post-hoc analysis of the data type is performed. The results and the description are similar to the ones presented in table 4.16. The post-hoc analysis on the data dimensionality reduction methods is also performed. The results and the description are also similar to the ones presented in table 4.50.

4.3 Key findings from the empirical study and how they guided the system development methodology on the new *k*-hyperparameter tuning technique.

The analysis on the findings from the empirical study aimed at answering the following research question:

RQ2: How can an improved *k*-hyperparameter tuning technique, in high-dimensional space clustering, be developed, based on the results of both the theoretical and empirical analysis?

Empirical analysis is fundamental to the development of improved models and algorithms, through the provision of a rigorous and evidence-based approach to designing, evaluating, and refining machine learning models (Zhang et al., 2017).

In light of this, the key findings from the empirical study are listed down as well as how they are used to provide guidance on the system development methodology of the newly developed technique. These findings are included in table 4.52 shown.

Table 4.52: Summary of empirical study findings and how they guided the system development methodology of the new *k*HT technique

Key finding from the empirical study	How it guided the development methodology of the new *k*-hyperparameter tuning technique
There are no statistically significant interactions between the *k*-hyperparameter tuning technique and the data type of the input dataset. The data type of the input dataset did not influence the choice of the *k*-hyperparameter tuning technique.	The improved autoencoder architecture on the new *k*-hyperparameter tuning technique is designed in such a way that it handles any dataset in resilience, regardless of its data type. It is not designed to handle a particular type of dataset only. Through this, it is able to accommodate video, audio, text and image datasets through its flexible architecture.
The choice of the data dimensionality reduction method had a statistically significant difference in the performance of the *k*-hyperparameter tuning techniques in high-dimensional space clustering. However, this performance did not vary depending on the choice of the *k*-hyperparameter tuning technique.	In the new technique, the improved autoencoder's architecture is made in a manner that ensures its universal usage across different *k*-hyperparameter tuning techniques. It is not designed for use in a particular technique only.

There is a statistically significant interaction effect between the dimensionality reduction method used and the data type of the input dataset.	In the design of the new technique, different set of architectural hyperparameter settings are put in place, with each set of settings aimed at handling a particular data type in the most efficient manner. At the same time, the best performing dimensionality reduction methods, for the respective data types, are used in the transfer learning process of the new technique's improved autoencoder.
There is no statistically significant interaction between the dimensionality level of a dataset and the dimensionality reduction method. Although the choice of the dimensionality reduction method had a statistically significant difference in the performance of the *k*-hyperparameter tuning techniques, this performance did not vary depending on the dimensionality levels present in the dataset.	The architecture of the new technique's improved autoencoder is designed in such a way that it works with dataset of any dimensionality level. Based on this, flexible input layers that allow variable input sizes are designed using the Tensor flow.
The choice on the data type of the input dataset had a statistically significant difference in the performance of the *k*-hyperparameter tuning techniques. However, there is no interaction i.e. this performance did not vary depending on the choice of the tuning technique.	In the new technique, the architecture of the improved autoencoder is made in resilience, in such a way that it effectively handles a variety of datasets. This is achieved through a thoughtful selection of the architecture-related and the training-related hyperparameter settings as well as an effective transfer learning strategy.

There are no significant differences between the autoencoders and the UMAP-learn method in the image datasets.	The design of the new technique involved the transfer learning strategy where the feature extraction of the UMAP learn method on the NSCLC Radiogenomics dataset and the ORL face datasets are used to perform unsupervised transfer learning on the autoencoder, making it effective to handle the Lung Cancer and Yale image datasets.
There are no significant differences between the autoencoders and the standard PCA method in the text datasets.	The design of the new technique involved the transfer learning strategy where the feature extraction of the standard PCA method on the Tox21 and BBC news datasets are used to perform unsupervised transfer learning on the autoencoder, making it effective to handle the Tox 171 and the Reuters datasets, respectively.
There are no significant differences between the autoencoders and the Spatio-Temporal UMAP method in the video datasets.	The design of the new technique involved transfer learning strategy where feature extraction of the Spatio-Temporal UMAP method on MPII cooking activities dataset is used to perform unsupervised transfer learning on the autoencoder, making it effective on YouCook datasets.
There are no significant differences between the autoencoders and the Multicore t-SNE method in the audio datasets.	The design of the new technique involved the transfer learning strategy where the feature extraction of the Multicore t-SNE method on the Stethoscope heart sounds and the Coswara datasets are used to perform unsupervised transfer learning

	on the autoencoder, making it effective to handle the Heart beat sounds and the Covid-19 coughs datasets, respectively.
There are no statistically significant differences in the performances of some data types of the input datasets.	In the design of the improved autoencoder in the new technique, similar architectural-related and the training-related hyperparameter settings are adopted for the types of the input datasets that did not have statistically significant differences. For example, the convolutional architecture is adopted for both the video and image datasets in the design of the autoencoder. At the same time, similar UMAP dimensionality reduction methods are used in the pre-training processes of the autoencoder for both the video and the audio datasets. The UMAP learn is used to pre-train the autoencoder on the image datasets while the Spatio-Temporal UMAP is used for the video datasets.
The experimental results showed that there are no significant interactions among the *k*-hyperparameter tuning techniques, type of input datasets as well as their level of dimensionality.	The architecture of the improved autoencoder on the new technique is designed in such a way that it can independently be applied in any *k*-hyperparameter tuning technique and can be used on any type of dataset, and one that comprises of any number of features (dimensionality level).

The experimental results showed that there are no significant interactions among the *k*-hyperparameter tuning techniques, dimensionality levels of the input datasets as well as the data dimensionality reduction methods applied.	The architecture of the improved autoencoder on the new technique is designed in such a way that it effectively performed data dimensionality reduction process regardless of the number of features present in a dataset or regardless of the *k*-hyperparameter tuning technique used.
Although the data types and the dimensionality reduction methods had a statistically significant interaction in the performance of the *k*-hyperparameter tuning techniques, this performance did not vary depending on the choice of the technique.	The architecture of the improved autoencoder focused on an effective dimensionality reduction process of the different variety of datasets. This is achieved through a thoughtful selection of the architectural-related and the training-related hyperparameter settings as well as an effective transfer learning strategy.
The experimental results showed that there re no statistically significant interactions between the *k*-hyperparameter tuning technique and the dimensionality reduction method applied. In a *k*-hyperparameter tuning technique, the choice of a particular dimensionality reduction method, over another, did not have a statistically significant difference in the performance of the techniques.	In the new technique, the improved autoencoder's architecture is made in such a way that it can work with any *k*-hyperparameter tuning technique. It is not specifically made for use with a specific *k*-hyperparameter tuning technique.

With some high-dimensional datasets, the generation of a smooth elbow made it ambiguous to identify the *k*-hyperparameter value when using the AutoElbow technique.	The new *k*-hyperparameter tuning technique is made in such a way that the identification of the *k*-hyperparameter value from the elbow is based on the ensemble's voting scheme, and is the trade-off between the number of clusters visualized at the autoencoder's latent space (number of dense regions with clear separations), the *k*-hyperparameter value that had a relatively good score of the internal validation indexes as well as a *k*-hyperparameter value that generated 0 or a value that is close to 0 on the derivative $f'''(k)(1+f'(k)2)-3f''(k)2f''((k)2f'(k)$.
There is a correlation between scores of the internal validation indexes and the clustering accuracy from the *k*-hyperparameter tuning techniques.	The new *k*-hyperparameter tuning technique is designed in such a way that its internal cluster validation mechanism performs adjustments on the self-adapting autoencoder engine based on the internal validation index scores. These adjustments are greedy and aimed at generating a *k*-value that has the best internal validation index scores, used as the target.

4.4 Evaluation on the effectiveness of the newly developed *k*-hyperparameter tuning technique against the existing ones in a variety of high-dimensional datasets.

The analysis on the findings from the empirical study aimed at answering the following research question:

RQ3: How effective is the newly developed *k*-hyperparameter tuning technique, in high-dimensional space clustering?

In light of this, the Kruskal-Wallis H statistic test on the newly developed technique against the existing techniques is first performed in order to identify the statistical significance difference. After this, further ANOVA tests are performed in order to identify the specific differences in these techniques. At the end of this evaluation, a comparison is made on the cluster visualizations for the different high-dimensional benchmark datasets, when using the standard autoencoder or the improved autoencoder.

4.4.1 Kruskal-Wallis H statistic on the newly developed *k*HT and the existing ones in a variety of high-dimensional datasets.

The Kruskal-Wallis H statistic is used to determine if there are statistically significant differences in the ensemble internal validation index scores (EIVI) across the four *k*-hyperparameter tuning techniques in a variety of datasets. The alpha value of 0.05 is stated and the degrees of freedom calculated as 3 (K-1) since there are four groups of the *k*-hyperparameter tuning techniques. After this, the decision rule of $x^2 > 5.99$ using the chi-square table is stated, test statistic calculated and the results displayed. Lastly, the conclusion is done based on the displayed results. Using an alpha score of 0.05 and degrees of freedom of 3 (K-1), the Kruskal-Wallis H statistic is computed as follows:

$$H = \frac{12}{N(N+1)} \left(\frac{<Ti^2}{<n}\right) - 3(N+1) \qquad (4.1)$$

T= Total sum of ranks in each group

n= Sample size in each group

N= Sample size in all groups

$$H = \frac{12}{28(28+1)}\left(\frac{118^2}{7} + \frac{119^2}{7} + \frac{125^2}{7} + \frac{44^2}{7}\right) - 3(28+1)$$

$$H = \frac{12}{812}\left(\frac{13924}{7} + \frac{14161}{7} + \frac{15625}{7} + \frac{1936}{7}\right) - 87$$

$$H = 0.01478\ (1989.1429 + 2023 + 2232.1429 + 276.571429) - 87$$

$$H = 0.01478\ (6{,}520.857229) - 87$$

$$H = 96.378270 - 87$$

$$H = 9.378271$$

The computed X^2 statistic from the Kruskal-Wallis test is greater than 5.99, indicating that the test statistic is large enough to be considered statistically significant at a significant level of 0.05. In this case, the null hypothesis is rejected and it is therefore concluded that there are significant differences in the performance of the four *k*-hyperparameter tuning techniques in the variety of datasets. In line with Okoye and Hosseini (2024), the Kruskal-Wallis H test is used for validating the new technique as it is a non-parametric method ideal for comparing median ranks across multiple groups, especially in high dimensional datasets that often exhibit complex structures and sparsity that complicate assumptions of normal distribution.

In section 4.4.2, the one way ANOVA and the post hoc analysis on these techniques shows the specific variations among them.

4.4.2 One-way ANOVA analysis on the EIVI of the newly developed *k*HT against the existing ones in a variety of HDDs

The adoption of the ensemble internal validation index score in the evaluation of the quality of clusters is due to the fact that different internal validation indexes gives contradicting information about the quality of clustering results. This is because of the fact that these internal validation indexes give different perspectives about the quality of clusters. For this reason, using multiple internal validation indexes is crucial to the evaluation process of any clustering algorithm. However, even with the multiple metrics, the challenge of different scales comes up. For example, the Silhouette index is usually confined to values of between 0 and 1 while Dunn index is not confined at all. This makes the range of these two indexes significantly different.

To solve this challenge, this research study proposes that the normalization feature scaling process is applied across all the four popular internal validation indexes. After this, the mean score, based on the normalized values across the four indexes is calculated in order to return one composite index score that is used in the final evaluation process. This index is referred to as the ensemble internal validation index (EIVI). The adoption of this index is said to exercise equal sensitivity across the variety of characteristics present in a high-dimensional dataset. The table 4.53 shows the results of the one-way ANOVA analysis on the ensemble internal validation index scores of both the newly developed technique and the existing ones in a variety of datasets.

Table 4.53: One-way ANOVA analysis on the EIVI of the newly developed *k*HT against the existing ones in a variety of HDDs

ANOVA - EIVI

	Sum of Squares	df	Mean Square	F	p
kHT Technique	53257	3	17752	21.2	**< .001**
Residuals	23431	28	837		

The low *p*-value less than 0.01 indicates that the observed statistical significant difference is strong, supporting the conclusion that there are significant variations in the ensemble index scores of the different *k*-hyperparameter tuning techniques. Based on this result, a post-hoc analysis is performed in order to identify the source of the specific variations based on this *p*-value. The post-hoc results are reported in table 4.54.

Table 4.54: Posthoc analysis on the *k*HTs on the one-way ANOVA analysis on the EIVI of the new *k*HT against the existing ones in a variety of HDDs.

Post Hoc Comparisons - kHT Technique

Comparison							
kHT Technique		**kHT Technique**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
kHT 1	-	kHT 2	56.25	14.0	28.0	4.004	**0.002**
	-	kHT 3	97.75	14.0	28.0	6.959	**< .001**
	-	kHT 4	20.88	6.60	96.0	3.163	**0.042**
kHT 2	-	kHT 3	94.25	14.0	28.0	6.709	**< .001**
	-	kHT 4	52.75	14.0	28.0	3.755	**0.004**
kHT 3	-	kHT 4	-41.50	14.0	28.0	-2.954	**0.030**

From the post-hoc analysis, it is clear that the specific variations on the ensemble index scores of the *k*-hyperparameter tuning techniques are due to significant differences in the ensemble index scores between the techniques *k*HT1 and *k*HT2, *k*HT1 and *k*HT3, *k*HT2 and *k*HT3, *k*HT2 and *k*HT4 as well as between techniques *k*HT3 and *k*HT4. *k*HT4 is the newly developed technique.

4.4.3 One -way ANOVA analysis on CBTs of the newly developed *k*HT against the existing ones in a variety of HDDs.

The table 4.55 shows the results of the one-way ANOVA analysis on the cluster building times of both the newly developed technique and the existing ones in a variety of datasets.

Table 4.55: One -way ANOVA analysis on the CBTs of the newly developed *k*HT against the existing ones in a variety of HDDs.

ANOVA - CBT

	Sum of Squares	df	Mean Square	F	p
kHT Technique	6881	3	2294	16.7	**< .001**
Residuals	3850	28	137		

The low *p*-value of less than 0.01 indicates that the observed significant difference is strong, supporting the conclusion that there are significant variations in the cluster building times across the different *k*-hyperparameter tuning techniques. Based on this result, further post-hoc analysis is performed in order to identify the source of the specific variations based on this *p*-value. The post-hoc results are reported in table 4.56:

Table 4.56: Posthoc analysis of *k*HTs in the one -way ANOVA analysis on CBTs of the new *k*HT against the existing ones in a variety of HDDs.

Post Hoc Comparisons - kHT Technique

Comparison							
kHT Technique		**kHT Technique**	**Mean Difference**	**SE**	**df**	**t**	**p_{tukey}**
kHT 1	-	kHT 2	0.297	5.86	28.0	0.0507	1.000
	-	kHT 3	-29.931	5.86	28.0	-5.1051	**< .001**
	-	kHT 4	-28.384	5.86	28.0	-4.8412	**< .001**
kHT 2	-	kHT 3	-30.229	5.86	28.0	-5.1558	**< .001**
	-	kHT 4	-28.681	5.86	28.0	-4.8919	**< .001**
kHT 3	-	kHT 4	1.548	5.86	28.0	0.2639	0.993

From the post-hoc analysis, it is clear that the specific variations in cluster building times on the *k*-hyperparameter tuning techniques are due to significant differences between the techniques *k*HT1 and *k*HT3, *k*HT1 and *k*HT4, *k*HT2 and *k*HT3 as well as between the techniques *k*HT2 and *k*HT4.

4.5 Whisker-box plots of the different k-hyperparameter tuning techniques in a variety of high dimensional datasets.

The whisker plot shows relatively lower median (second quartile) values for the techniques *k*HT1, *k*HT2 and *k*HT3 as compared to the median values of the newly developed *k*HT4 technique. At the same time, there is one outlier in *k*HT3 technique while there isn't any outlier in all the other techniques. The whiskers for the newly developed technique *k*HT4 are relatively shorter than the ones in techniques *k*HT1, *k*HT2 and *k*HT3. The technique *k*HT4 demonstrates superior clustering performance and reliability compared to *k*HT1, *k*HT2, and *k*HT3 techniques. Higher median values indicate better average clustering performance, while the shorter whiskers and lack of outliers suggest more consistent and reliable clustering outcomes. These attributes make the *k*HT4 a relatively more promising *k*-hyperparameter tuning technique in a variety of high-dimensional datasets, providing both quality clustering, consistent and reliable results as compared to the other *k*-hyperparameter tuning techniques. Figure 4.1 shows the whisker-box plots of the different *k*-hyperparameter tuning techniques in a variety of high dimensional datasets that are comprised of the different data types.

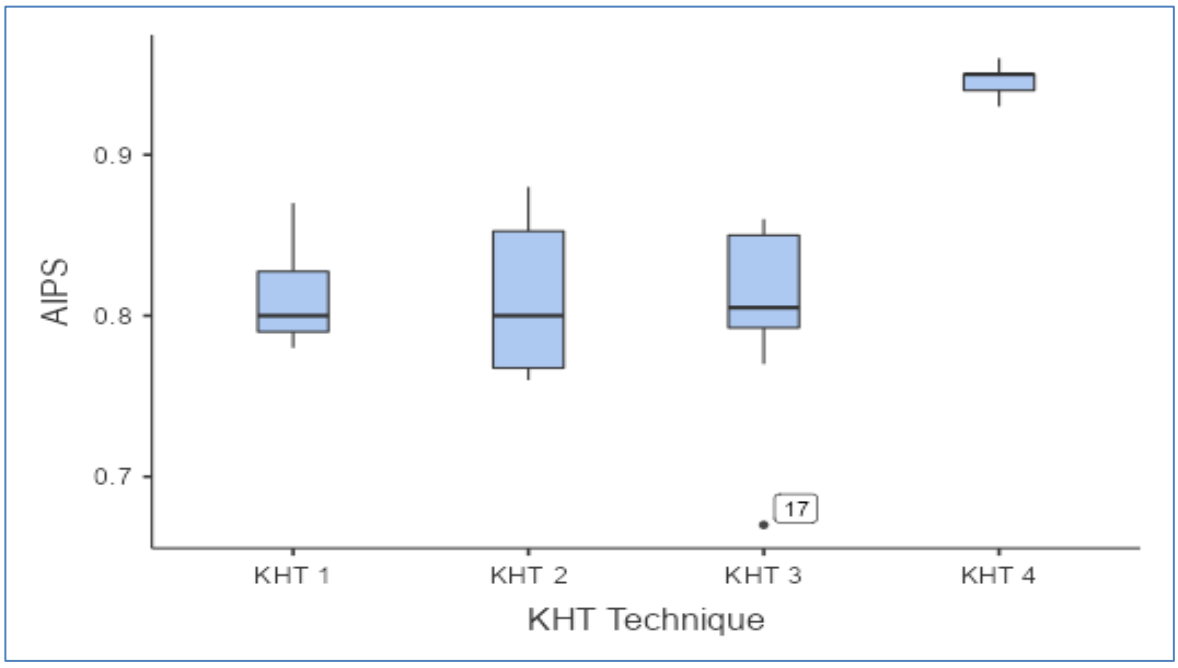


Figure 4.1: Whisker-box plots of the different *k*-hyperparameter tuning techniques in a variety of high-dimensional datasets

4.6 Comparison of the cluster visualizations in different HDDs between the standard autoencoder and the improved one in the new *k*HT.

Cluster visualization is a powerful tool for gaining insights into a dataset (Kapoor & Singhal, 2017). Proper interpretation of the cluster visualizations in *k*-means algorithms is crucial for understanding the results of the clustering analysis (Kapoor & Singhal, 2017). Good clustering results usually show clear separation between clusters, with minimal overlap (Kapoor & Singhal, 2017; Celebi et al., 2013). In this section, the cluster visualizations on the benchmark datasets using the standard and the improved autoencoder on the new technique are compared.

Section 4.6.1 shows the visualization of the Tox-171 dataset using the standard autoencoder. Section 4.6.2 shows the visualization of the Tox-171 dataset using the improved autoencoder in the newly developed *k*-hyperparameter tuning technique. Section 4.6.3 shows the visualization of the Yale image dataset using the standard autoencoder in the newly developed *k*-hyperparameter tuning technique. The section 4.6.4 shows the visualization of the Yale image dataset using the improved autoencoder in the newly developed *k*-hyperparameter tuning technique.

The section 4.6.5 shows the visualization of the Reuters dataset using the standard autoencoder in the newly developed *k*-hyperparameter tuning technique. The section 4.6.6 shows the visualization of the Reuters dataset using the improved autoencoder in the newly developed *k*-hyperparameter tuning technique.

The section 4.6.7 shows the visualization of the Lung Cancer dataset using the standard autoencoder in the newly developed *k*-hyperparameter tuning technique.

The section 4.6.8 shows the visualization of the Lung Cancer dataset using the improved autoencoder in the newly developed *k*-hyperparameter tuning technique.

The section 4.6.9 shows the visualization of the Covid-19 coughs dataset using the standard autoencoder in the newly developed *k*-hyperparameter tuning technique.The section 4.6.10 shows the visualization of the Covid-19 coughs dataset using the improved autoencoder in the newly developed *k*-hyperparameter tuning technique.

The section 4.6.11 shows the visualization of the Heart beat sounds dataset using the standard autoencoder in the newly developed *k*-hyperparameter tuning technique. The section 4.6.12 shows the visualization of the Heart beat sounds dataset using the improved autoencoder in the newly developed *k*-hyperparameter tuning technique.

The section 4.6.13 shows the visualization of the YouCook dataset using the standard autoencoder in the newly developed *k*-hyperparameter tuning technique.The section 4.6.14 shows the visualization of the YouCook dataset using the improved autoencoder in the newly developed *k*-hyperparameter tuning technique. The cluster visualizations for all the high dimensional benchmark datasets are done using the improved autoencoder in the newly developed *k*-hyperparameter tuning technique.

The section 4.6.15 shows the visualization of the YouTube vlogs for cooking beef in Kenya using the newly developed *k*-hyperparameter tuning technique and an improved autoencoder.

4.6.1 Visualization of the Tox-171 dataset using the newly developed *k*HT and the standard autoencoder.

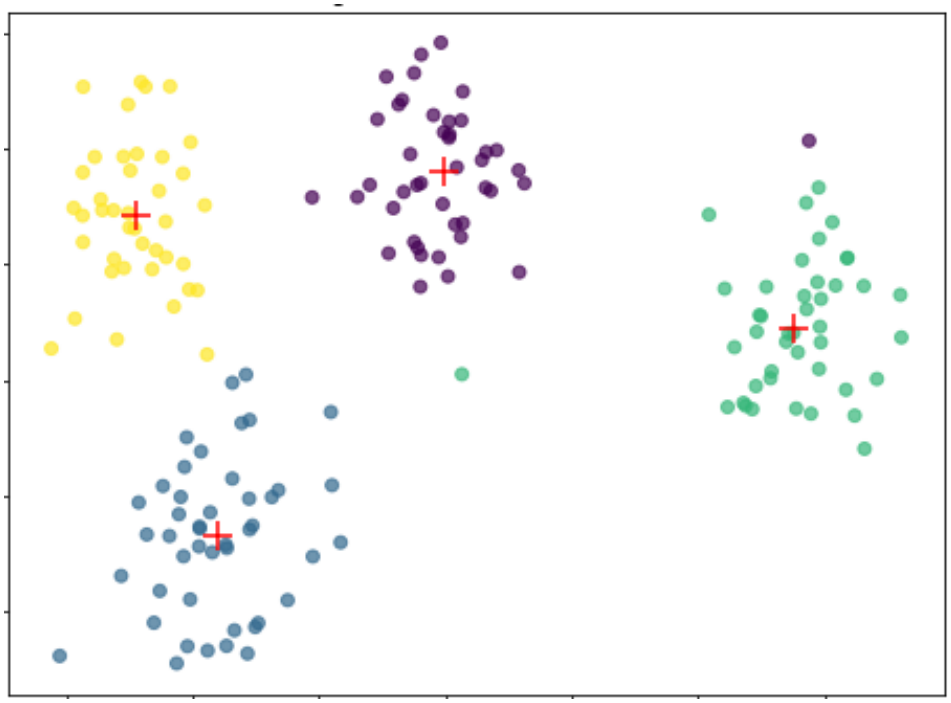


Figure 4.2: Cluster visualization of the Tox-171 dataset using the new *k*HT and the standard autoencoder

4.6.2 Visualization of the Tox-171 dataset using the newly developed *k*HT and an improved autoencoder.

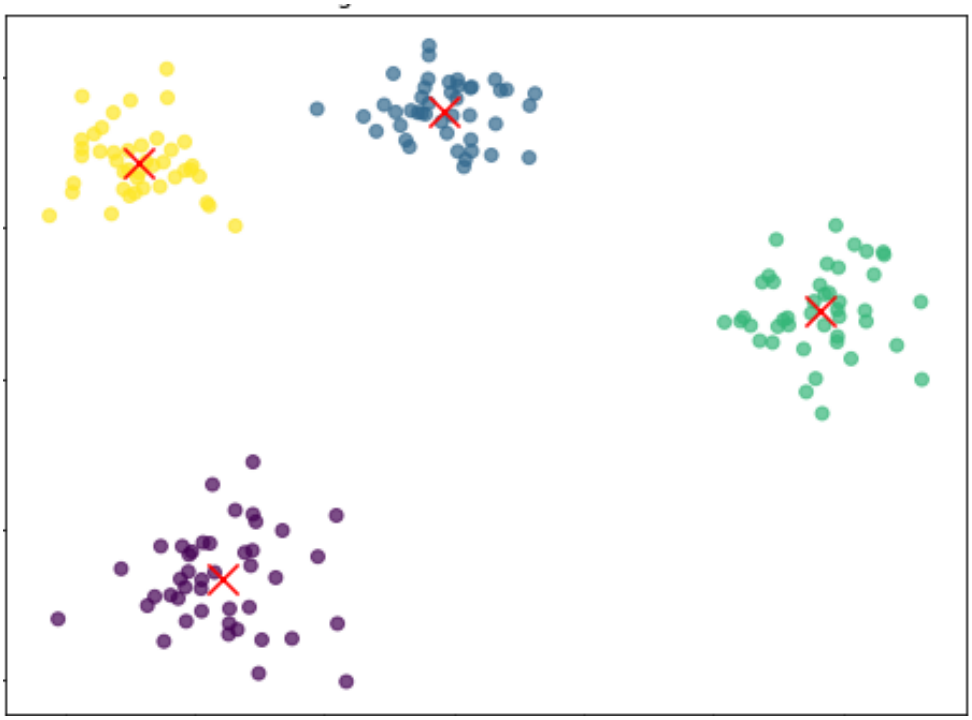


Figure 4.3: Cluster visualization of the Tox-171 dataset using the new *k*HT and the improved autoencoder

4.6.3 Visualization of the Yale image dataset using the newly developed *k*HT and a standard autoencoder.

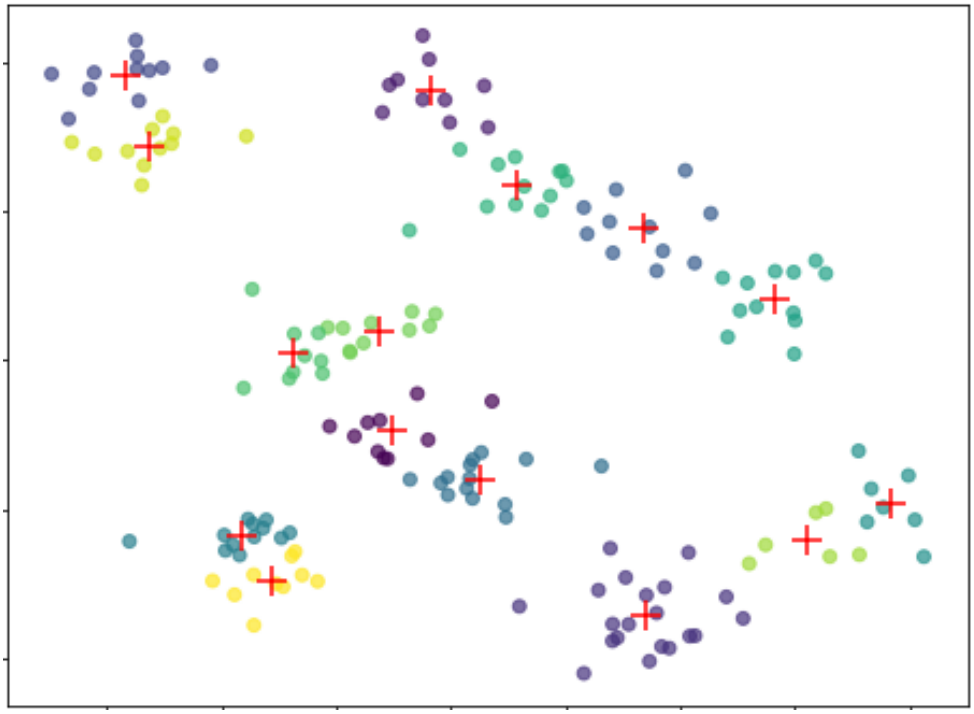


Figure 4.4: Cluster visualization of the Yale dataset using the new *k*HT and the standard autoencoder

4.6.4 Visualization of the Yale image dataset using the newly developed *k*HT and an improved autoencoder.

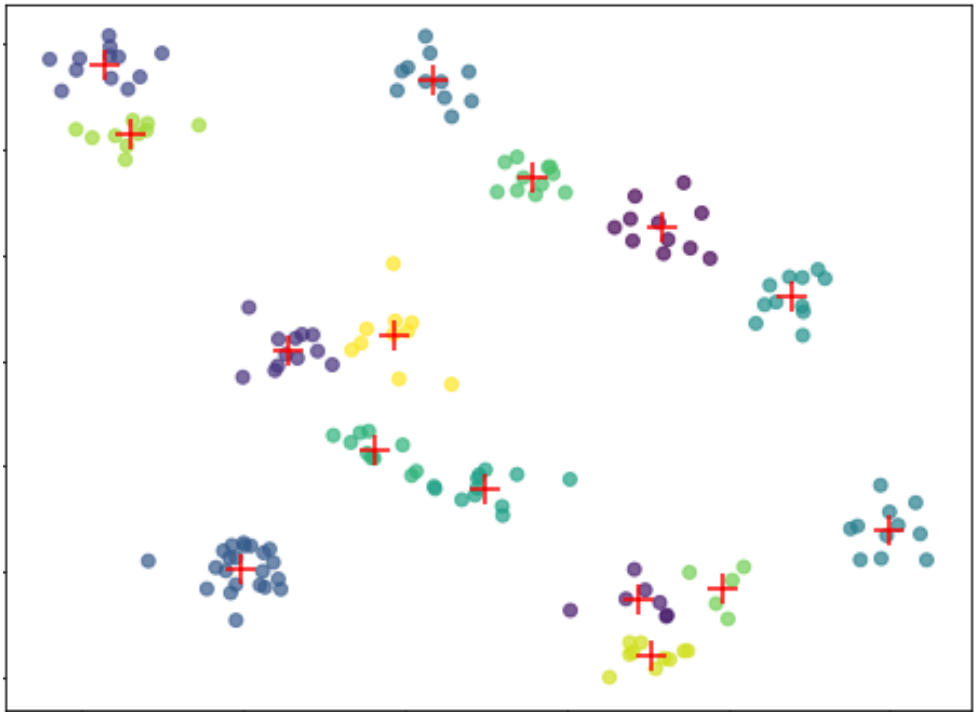


Figure 4.5: Cluster visualization of the Yale dataset using the new *k*HT and the improved autoencoder

4.6.5 Visualization of the Reuters dataset using the newly developed *k*HT and a standard autoencoder.

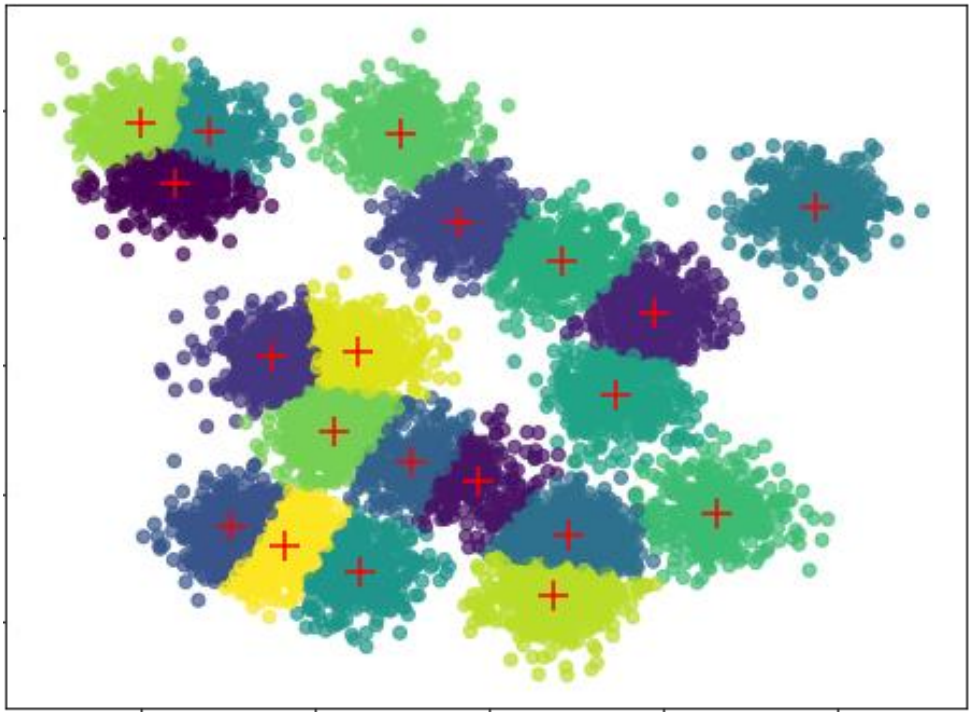


Figure 4.6: Cluster visualization of the Reuters dataset using the new *k*HT and the standard autoencoder

4.6.6 Visualization of the Reuters dataset using the newly developed *k*HT and an improved autoencoder.

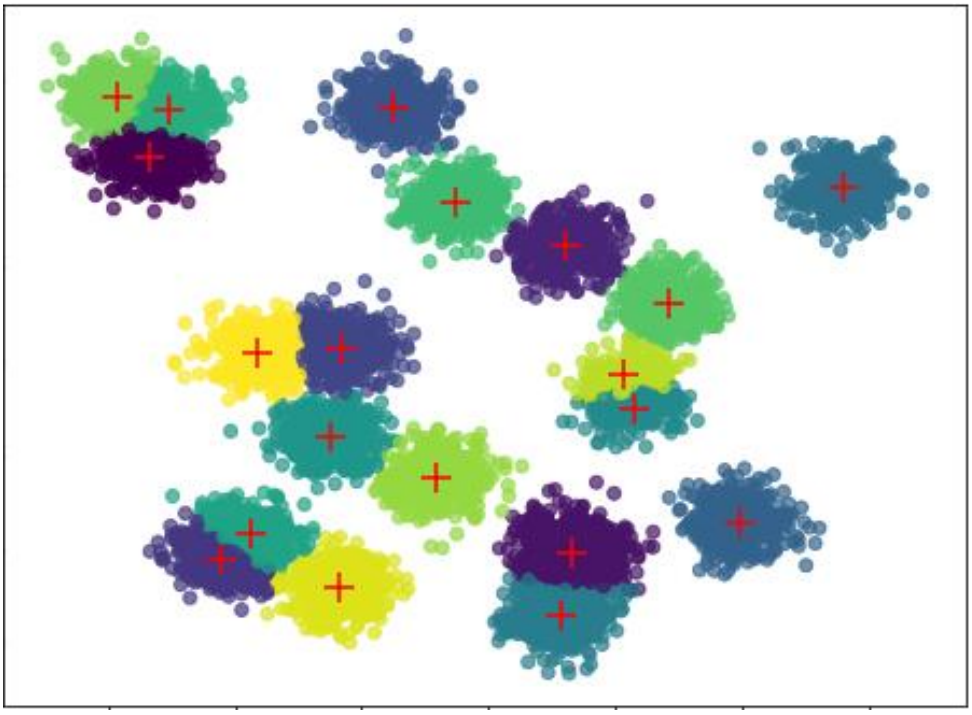


Figure 4.7: Cluster visualization of the Reuters dataset using the new *k*HT and the improved autoencoder

4.6.7 Visualization of the Lung Cancer dataset using the newly developed *k*HT and a standard autoencoder.

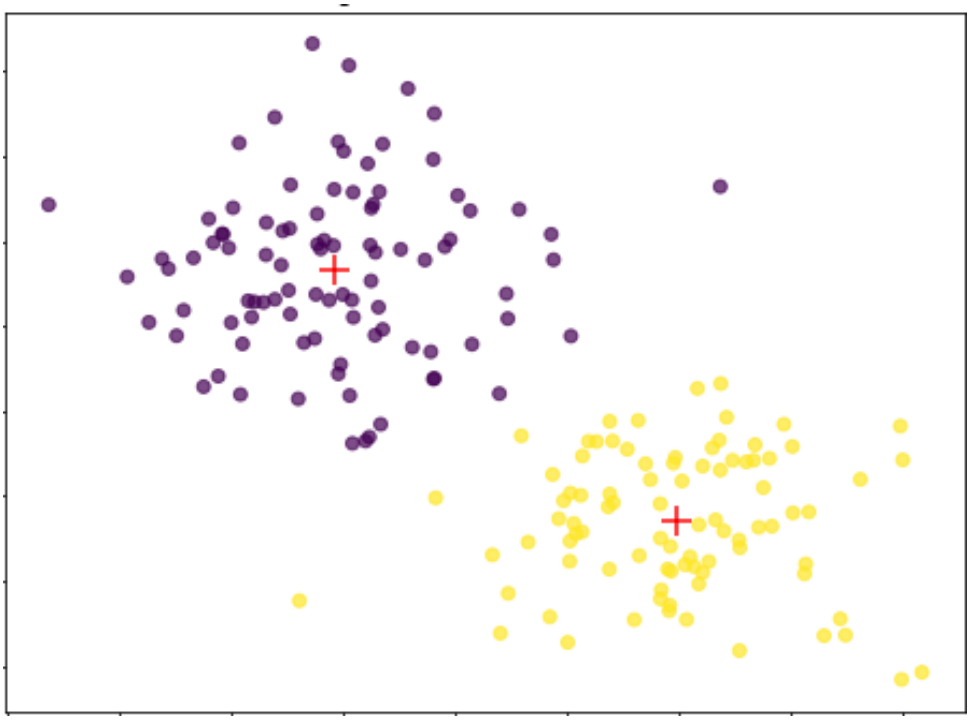


Figure 4.8: Cluster visualization of the Lung Cancer dataset using the new *k*HT and the standard autoencoder

4.6.8 Visualization of the Lung Cancer dataset using the newly developed *k*HT and an improved autoencoder.

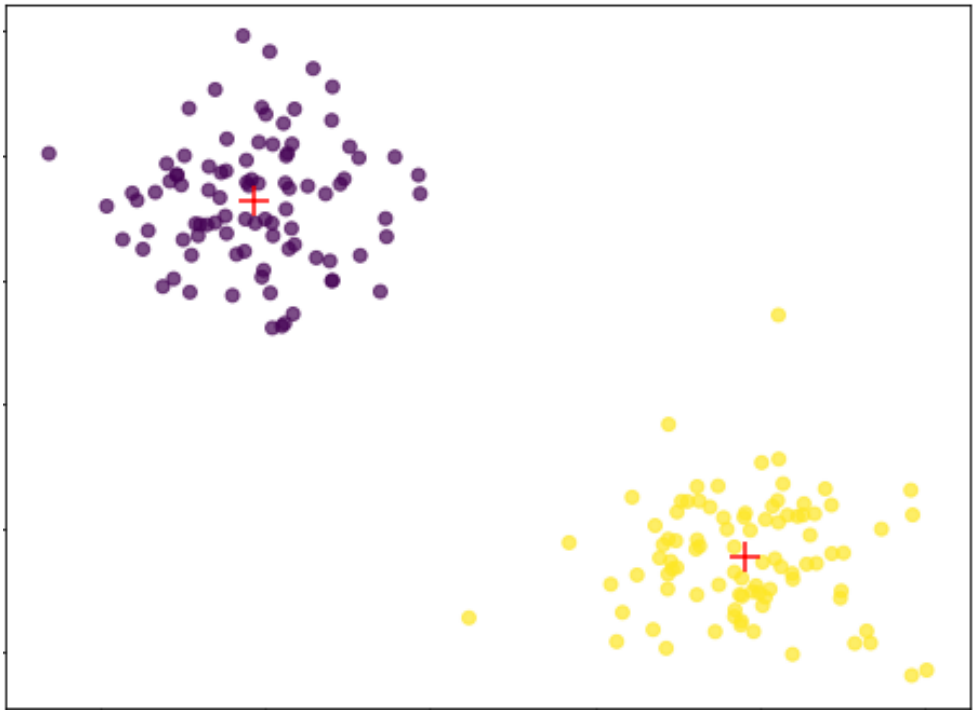


Figure 4.9: Cluster visualization of the Lung Cancer dataset using the new *k*HT and the improved autoencoder

4.6.9 Visualization of the Covid-19 coughs dataset using the newly developed *k*HT and a standard autoencoder.

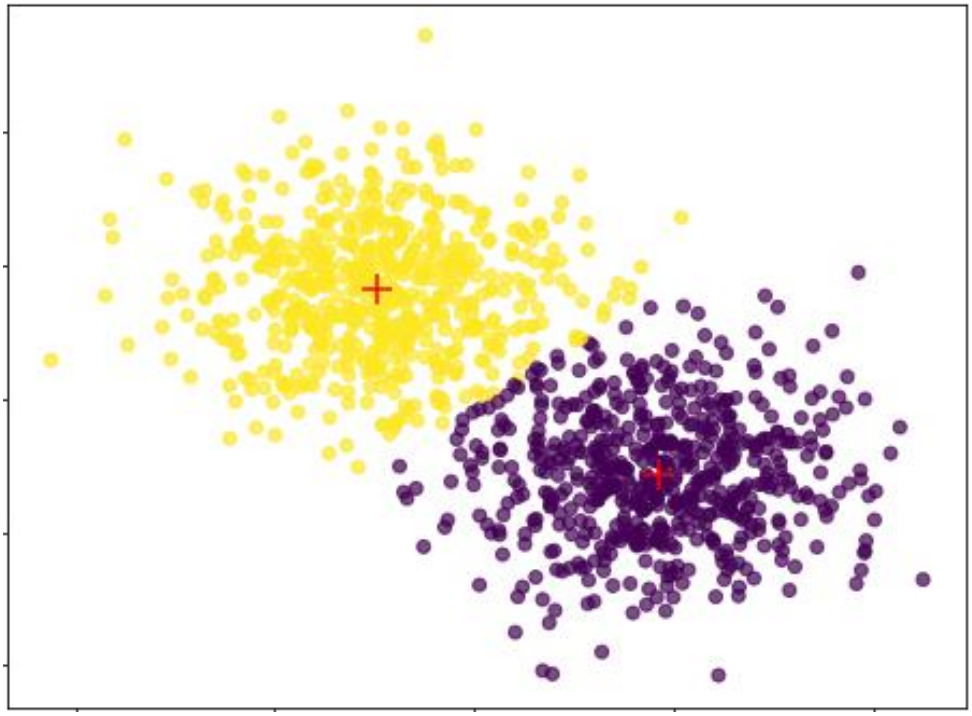


Figure 4.10: Cluster visualization of the Covid-19 Coughs dataset using the new *k*HT and the standard autoencoder

4.6.10 Visualization of the Covid-19 coughs dataset using the newly developed *k*HT and improved autoencoder.

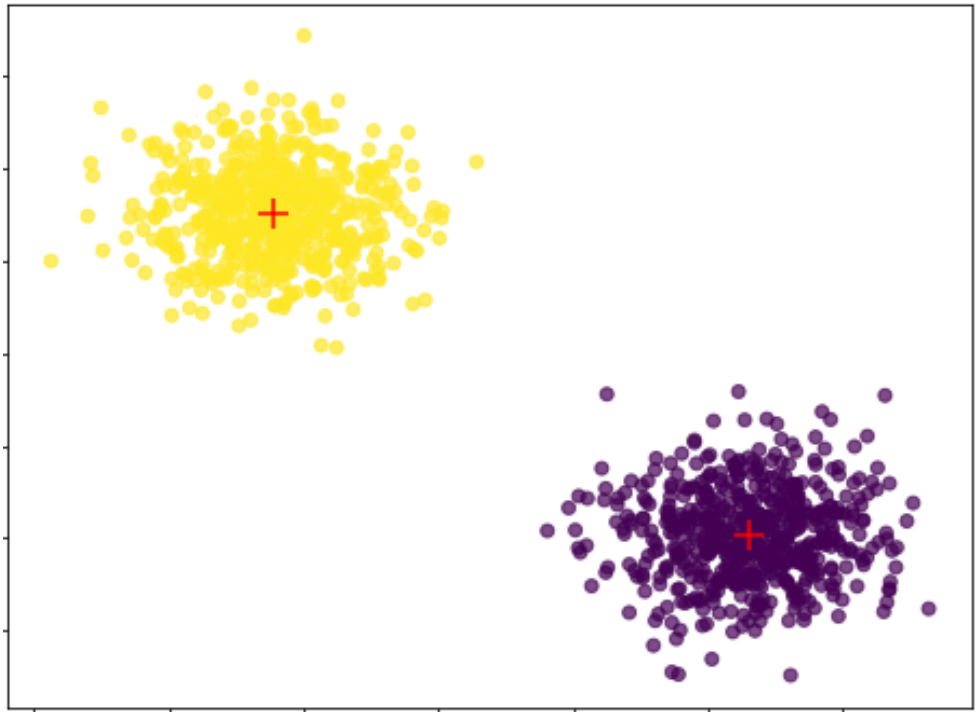


Figure 4.11: Cluster visualization of the Covid-19 Coughs dataset using the new *k*HT and the improved autoencoder

4.6.11 Visualization of the Heart beat sounds dataset using the newly developed *k*HT and a standard autoencoder.

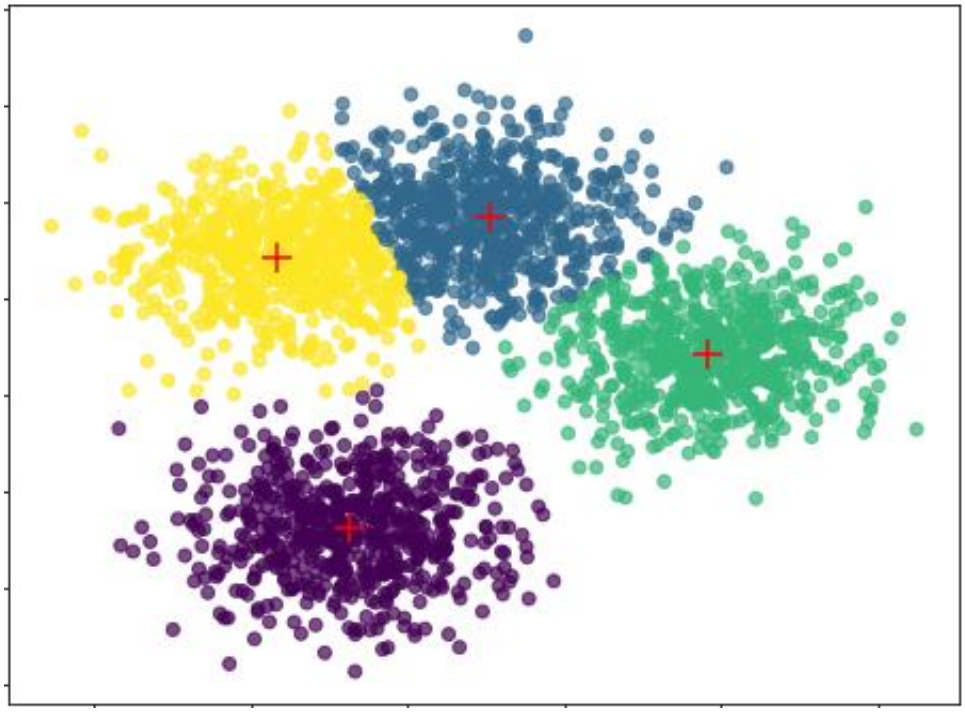


Figure 4.12: Cluster visualization of the Heart beats sounds dataset using the new *k*HT and the standard autoencoder

4.6.12 Visualization of the Heart beat sounds dataset using the newly developed *k*HT and an improved autoencoder.

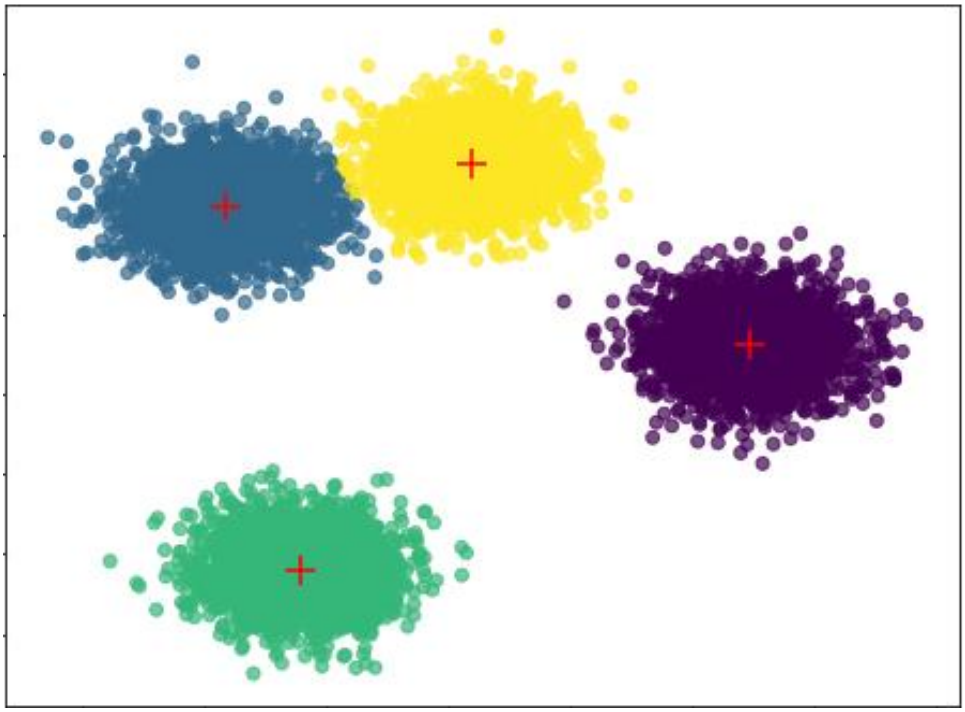


Figure 4.13: Cluster visualization of the Heart beats sounds dataset using the new *k*HT and the improved autoencoder

In figure 4:13 above, a significant breakthrough of the novel technique lies in its effective clustering of heart beat sounds using an autoencoder pre-trained on multicore t-SNE. This approach enables the generation of four well-defined and interpretable clusters, each corresponding to a specific cardiac condition. This captures nuanced differences and similarities between various heart sound profiles, which are essential for accurate diagnostic insights. The model's clustering delineates conditions with distinct acoustic signatures, i.e. the aortic valve stenosis, represented by the dark blue cluster, and aortic regurgitation, shown in yellow. These clusters are positioned in close proximity to each other. This spatial closeness reflects the shared pathological basis of the two conditions, as both involve abnormalities in the structure and function of the aortic valve, which leads to specific, distinguishable heart murmurs. In the case of aortic valve stenosis, the narrowing of the aortic valve generates a systolic murmur which is a higher-pressure sound produced as the heart pumps blood through the constricted valve. On the other hand, aortic regurgitation is characterized by a diastolic murmur due to blood flowing back through a weakened valve. The autoencoder captures these subtle yet critical distinctions, placing these two related conditions near each other while maintaining enough separation to distinguish their unique acoustic profiles. In contrast, ventricular septal defect is captured in a separate cluster, represented by the purple group, reflecting its fundamentally different pathology. Unlike aortic stenosis or regurgitation, ventricular septal defect is characterized by an abnormal opening between the heart’s ventricles, which allows blood to flow between the chambers, creating distinct auditory patterns. This sound pattern lacks the characteristic murmurs associated with aortic valve conditions and instead presents a distinct acoustic signature. This anatomical and functional difference is why the model, through dimensionality reduction and feature extraction, appropriately separates this from the aortic valve disorders, as shown in the clustering output. Lastly, the normal heart sounds are represented by the green cluster, clearly delineated from

all three pathological conditions. This separation emphasizes the characteristic ‘lub-dub’ pattern of a healthy heart, free from any murmurs or aberrant sounds caused by structural abnormalities (McGee, 2010). The autoencoder’s ability to distinguish these healthy sounds from the pathological clusters underscores its potential for aiding in clinical diagnostics. By accurately capturing and representing these conditions, the technique demonstrates a sophisticated understanding of heart sound variations and their diagnostic implications as wellas the potential of machine learning in enhancing diagnostics in cardiology. Through this clustering model, cardiologists and clinicians can gain an advanced, non-invasive tool for identifying and understanding various cardiac conditions with greater accuracy and efficiency.

4.6.13 Visualization of the YouCook dataset using the newly developed *k*HT and a standard autoencoder.

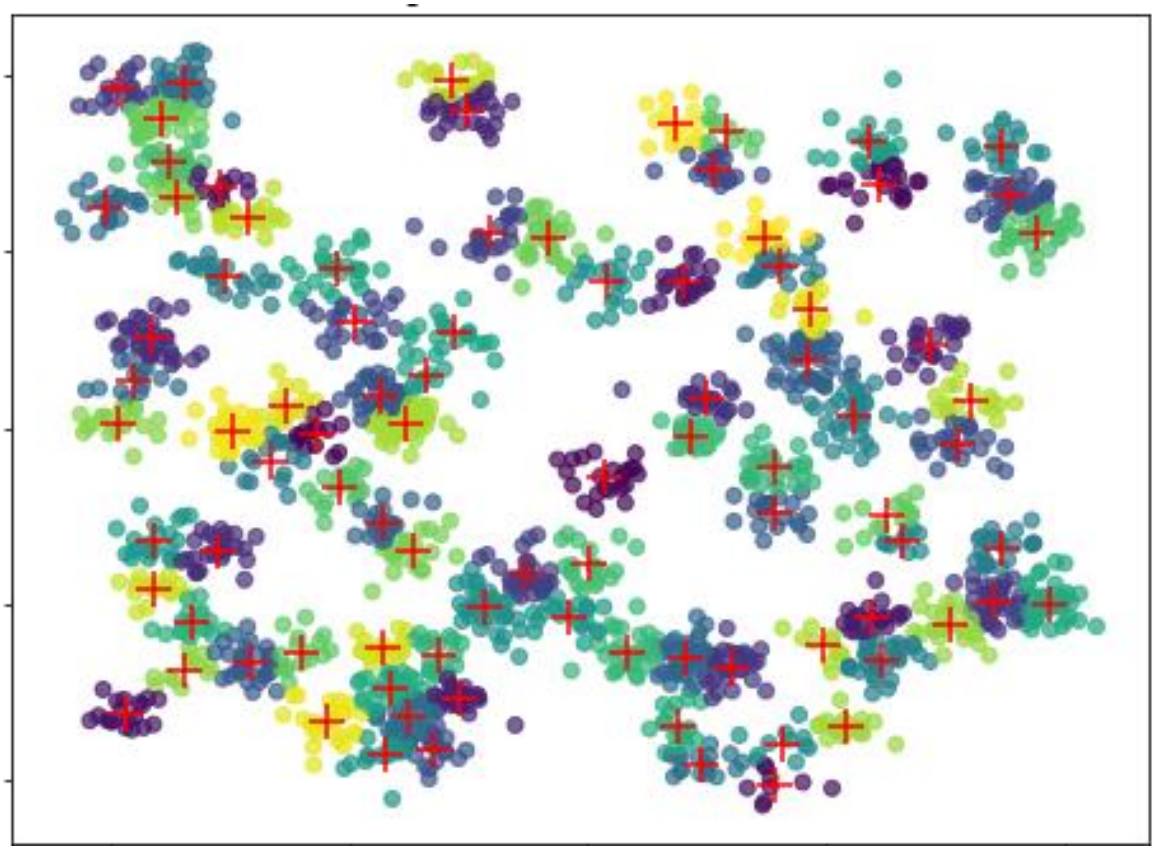


Figure 4.14: Cluster visualization of the YouCook dataset using the new *k*HT and the standard autoencoder

4.6.14 Visualization of the YouCook dataset using the newly developed *k*HT and an improved autoencoder.

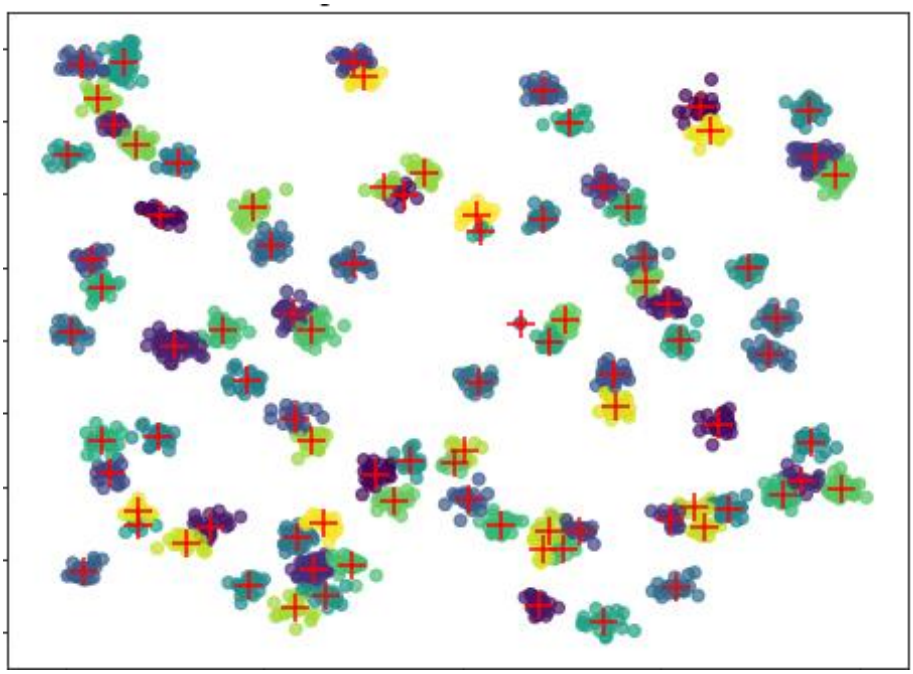


Figure 4.15: Cluster visualization of the YouCook dataset using the new *k*HT and the improved autoencoder

4.6.15 Visualization of the YouTube vlogs for cooking beef in Kenya using the newly developed *k*-hyperparameter tuning technique and an improved autoencoder.

In validating the effectiveness of the newly developed technique using primary data, recipes for cooking beef in Kenya are first mined from the popular Chef Raphael's video blogs. YouTube API is used to retrieve the video metadata, including titles and descriptions. When selecting videos to download from Chef Raphael's vlogs, there was focus on those that have a Creative Commons (CC) license, particularly the variants that allow for modification and redistribution, such as CC BY (attribution) or CC BY-SA (attribution-share alike), as these licenses generally permit users to share and adapt the content with proper credit to the original creator. This choice aligns with the ethical domain of intellectual property and content reuse. In line with this ethical domain, the ethical considerations in this research primarily revolves around the responsible use and adaptation of publicly available content while respecting the intellectual property rights of the content creator. By selecting videos licensed under Creative Commons that specifically allow for modification and redistribution, the study adheres to

ethical principles of content licensing compliance. This ensures that Chef Raphael's creative works are used appropriately, with proper credit given, thus respecting his intellectual property and complying with licensing requirements. Furthermore, the study follows ethical guidelines related to data usage transparency and informed reuse by making sure that only content with permissions for reuse is selected. This safeguards against unauthorized use of copyrighted materials and underscores the importance of adhering to legal and ethical standards in digital content research. Additionally, the focus on ethical content selection reinforces respect for the creator's original work, while allowing for meaningful adaptation and analysis. Spatial Temporal UMAP method is used to perform dimensionality reduction on the breakfast actions dataset. The reduced feature space of the breakfast actions dataset is used to pretrain the autoencoder on the new technique. The autoencoder is then fine-tuned to extract meaningful features from the retrieved videos of cooking beef from Chef Raphael's video blogs. The *k*-hyperparameter value is identified using the auto-elbow in the new technique and the clusters are finally visualized. Figure 4.15 shows the clusters of the common recipes for cooking beef in Kenya.

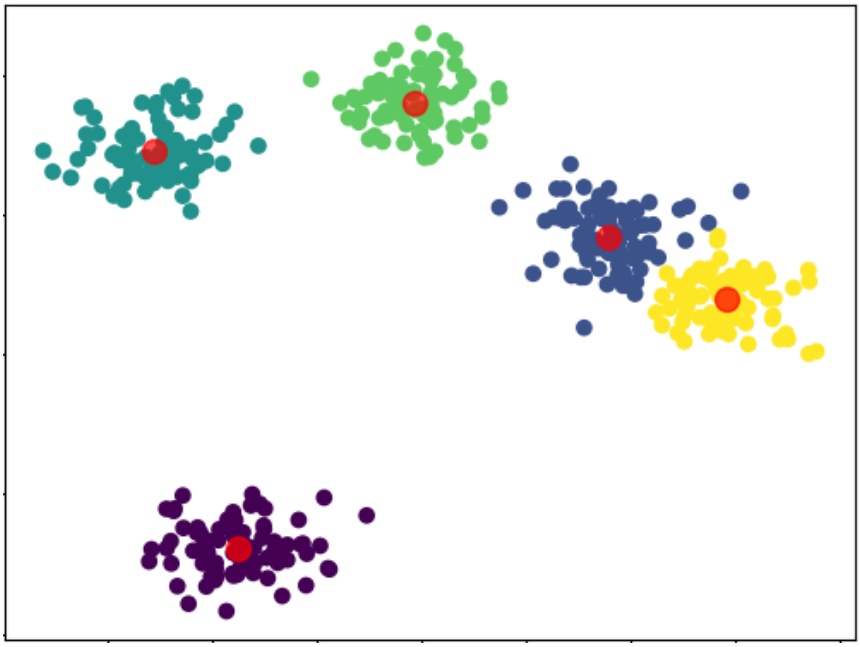


Figure 4.16: Cluster visualization of the YouTube vlogs for cooking beef in Kenya using the new *k*-hyperparameter tuning technique and an improved autoencoder.

The five clusters generated from these videos are attributed to the common recipes for cooking beef in Kenya i.e. Beef stew (dark green), Beef fry (light green), Grilled beef (dark blue), Nyama Choma (yellow) and the Boiled beef (purple). The visualizations could also be attributed to the ingredients and instructions for the different recipes. The grilled beef and the Nyama Choma share a number of ingredients and instructions, to a great extent. To some extent, the ingredients and instructions for Beef stew and Beef fry are similar. However, those of boiled beef are to an extent different from those of the other four recipes (Ada et al., 2013).

CHAPTER FIVE: DISCUSSION

5.1 Introduction

Chapter 4 provides detailed tables and graphical representations of the experimental results, in line with the first, second and the third objectives. Building on this, the current chapter engages in an in-depth discussion of these findings. The structure of the chapter is as follows: Firstly, an examination of the two-way analysis of variance (ANOVA) results is presented. Secondly, an exploration of the three-way ANOVA findings is reported. Thirdly, an evaluation of the results of the proposed *k*-hyperparameter tuning (*k*HT) technique is presented. Fourthly, a comparative discussion is carried out between the results of the existing *k*HT techniques and the novel *k*HT. Finally, a conclusion of the reported findings is drawn including the achievement of the research objectives as well as the research implications of these research findings.

5.2 Discussion on the results

The insights obtained from the ANOVA analysis of the significant differences and interactions in the performance of the *k*-hyperparameter tuning techniques, on varied dimensionality reduction methods, data types and dimensionality levels of the high dimensional datasets are invaluable for guiding the development of an improved technique in a number of ways. One such way is technique modification and enhancement. Understanding how data types, dimensionality levels, and the dimensionality reduction methods impact clustering quality on the hyperparameter tuning techniques is a possible guide for the researcher in devising an adaptive technique in high dimensional space clustering. Such a technique can adjust their parameters based on the specific characteristics of the high dimensional input dataset, optimizing the *k*-hyperparameter tuning process in different high dimensional spaces. Another way is a tailored data science tool box.

By considering the interactions, researchers can tailor the *k*-hyperparameter tuning process of specific types of datasets to specific techniques or dimensionality reduction methods or both. For instance, developing techniques that excel in clustering images in high dimensions might require different strategies compared to text data or different dimensionality levels. Optimization of the hyperparameter settings, development of a generalizable technique and formulation of an effective feature engineering and selection strategy are other possible ways in which the outcome of this research is aimed at. Insights derived from these analyses can also inspire novel optimization strategies.

This could inspire the development of an adaptive *k*-hyperparameter tuning technique that involves integrating multiple dimensionality reduction methods or employing adaptive strategies that dynamically adjust based on the characteristics of the high dimensional datasets in order to enhance clustering performance. It could also employ the best performing dimensionality reduction method in the unsupervised transfer learning strategy on the new technique. Understanding how different factors interact can provide insights into building more generalized *k*-hyperparameter tuning techniques that perform well across several high dimensional datasets with different data types and dimensionality levels. Lastly, the insights derived from the interactions can help identify the specific features or dimensions that are more impactful, in regards to the clustering quality, from a high dimensional dataset. This knowledge can aid the researcher in making a thoughtful and informed feature engineering and selection for improved performance of the *k*-hyperparameter tuning techniques.

5.2.1 Discussion on the Empirical Analysis Results on the two-way ANOVA and the Interactions

5.2.1.1 Analysis of Two-way ANOVA Across Varied Data Types

The results from the two-way ANOVA detailed in Tables 4.2 to 4.4, considers the performance of the selected *k*HT techniques when utilized to cluster datasets of varying data types (i.e., text, image, video, and audio datasets). As reported under section 4.2.1, the two-way ANOVA was carried out using the average index performance score (AIPS). Please see Equation 3.2 on section 3.4.7 for further look at how AIPS was derived. Results from Table 4.2 show that the selected existing *k*HT techniques had significantly different results when used across varying data types. This means that there is no single *k*HT technique that was better in clustering across the varying data types. This is to be expected as each method is characterized by its unique mathematical framework, presumptions, and optimization approaches which are likely to lead to distinct clustering outcomes. Further, this outcome is supported by the No Free Lunch (NFL) Theorem that states that there is no one algorithm that can generalize to all problems sets (Wolpert & Macready, 1997). In addition, this outcome is supported by the findings of Rodriguez et al. (2019), who noted variability in the performance of clustering algorithms across different datasets. The analysis shown on Table 4.2 further reveals that data type plays a critical role in determining the efficacy of clustering techniques. Input variations in data types into *k*HT methods lead to statistically significant differences in the performance of clustering algorithms. These differences arise from the specific ways in which algorithms process and glean information from various data types, resulting in clustering outcomes that reflect the intrinsic properties of the datasets. This is supported by Arora (2012), who examined the performance of classification algorithms across different datasets. The results in Tables 4.2 and 4.3 show that there are no interactions between the *k*HT and the DTs.

This suggests that the impact of the selected method on clustering does not vary with data type. Thus, despite the individual effects of the method and data type on the outcomes, the performance of the methods is consistent across various data types. This consistency is likely due to the inherent robustness and stability of the algorithms, which ensures sustained performance, suggesting a degree of immunity to data type variability. Furthermore, this consistent performance may also be due to the algorithms' ability to identify and utilize core features common to different data types. The outcome concurs with the work reported by Singh et al. (2017) that showed that techniques such as Random Forest and k-Nearest Neighbor, perform consistently across diverse datasets. The next section discusses the ANOVA results when the cluster building times (CBT) of the reported *k*HT methods was taken into considerations.

5.2.1.2 Two-way ANOVA and interactions on the CBTs of the *k*HTs in different DTs

The results from the two-way ANOVA detailed in Tables 4.5 and 4.6, considers the cluster building times (CBTs) of the selected *k*HT techniques when utilized to cluster datasets of varying data types. As reported under section 4.2.2, the two-way ANOVA was carried out using CBT. As discussed in section 3.4.8, cluster building time refers to the duration it takes for the *k*-hyperparameter tuning techniques to process the input datasets and group them into *k*-clusters. The results from Table 4.5 show that the selected existing *k*HT techniques did not have significantly different results when used across varying data types. This means that there is no single *k*HT technique that significantly differs with the rest in terms of the efficiency in building clusters across the varying data types. This is to be expected due to the fact that the different techniques possess optimization configurations and parameters that are not significantly different.

Such configurations and parameters include the consistency in the number of runs and iterations, *k*-means ++ initialization and stopping criteria as well as the core set of steps during execution. The core set of steps across the different techniques comprise of the initialization of the cluster centroids, assignment of data points to clusters, updating of the cluster centroids as well as the iteration process until convergence.

While the specific details of how these optimization configurations and parameters are implemented may vary, the overall structure remains similar thus making their CBTs statistically indifferent. This outcome is supported by the observation made by Friedrich et al. (2017) where both the cGA and λ-MMASib algorithms display similar run times. The analysis shown on Table 4.6 further reveals that the choice of the data type plays a critical role in determining the CBTs of the *k*HT techniques. The statistically significant differences only on the data types indicate that the datasets data types significantly influence the time taken by the *k*HT techniques to build clusters. This is to be expected because of the time taken by algorithms to interact with and process datasets of different data types can vary. Some algorithms might be more adaptable to certain data types than others, affecting their performance and time consumption. This outcome is supported by the observation made by Gikera et al. (2023a) where the run times of algorithms on some datasets are identified to be different from others.

The results in Table 4.5 show that there is no interaction between the *k*HT techniques and the data types with regards to the CBTs. The absence of statistically significant differences in the interactions is an indication that the time performance of these techniques remains consistent across different data types.

The impact of the different data types on the cluster building times is consistent across the different *k*HT techniques and the nature or characteristics of the data type do not significantly influence their efficiency. This is likely due to the fact that the application of the state-of-the-art dimensionality reduction methods (RMs) during the empirical analysis makes it highly effective in capturing the relevant information. This, therefore, makes the impact of data types on CBTs to remain consistent across the different *k*HT techniques. This way, the interaction effect between the two becomes minimal and subsequently insignificant. Furthermore, this may also be due to the fact that algorithms possess a level of generalizability, providing similar benefits across diverse data types.

5.2.1.3 Two-way ANOVA and interactions on the AIPS of the *k*HTs in different RMs

The results from the two-way ANOVA detailed in Tables 4.7 to 4.9, considers the performance of the *k*-hyperparameter tuning techniques (*k*HTs) in different dimensionality reduction methods (RMs). In Table 4.9, the significant effect on the Average Index Performance Score (AIPS) based on the choice of the RM is an indication that RMs have a substantive role in the performance of the *k*HTs. This is attributed to the fact that the algorithm-specific characteristics of a particular dimensionality reduction method make it unique in the way it handles the data type and the features present in a high dimensional dataset. This is also attributed to the unique characteristics and assumptions, with some dimensionality reduction methods being more robust in certain datasets than in others. More than one dimensionality reduction method can possess such similar characteristics and assumptions. This outcome is similar to the observation made by by Dash et al. (2010) and Bonidia et al. (2021).

In the works of Dash et al. (2010), the performance of the dimensionality reduction methods in the new hybridized *k*-means algorithm on the both the Breast cancer and SPECTF Heart datasets were better as compared to the performance of the dimensionality reduction methods in the standard *k*-means algorithm. In the comparative study by Bonidia et al. (2021), it was observed that the mathematical based feature extraction data dimensionality reduction methods outperformed the biological based methods on the RNA sequencing data. Moreover, in this study, it was observed that a combination of at least two feature extraction dimensionality reduction methods demonstrated a more superior performance as compared to the use of only one method. As the number of features increases, the volume of the feature space grows exponentially, making a high-dimensional space to become sparse and difficult to identify meaningful clusters. The data points are spread out, and distances between points lose their discriminatory power (Bonidia et al., 2021). Based on this observation, it is concluded that the significant effect of the dimensionality reduction methods on the techniques' performance underscores their crucial role in shaping how the techniques handle and interpret high-dimensional data and the challenges inherent in such datasets. Therefore, a thoughtful selection on the dimensionality reduction method can be helpful in identifying the most relevant and informative features, while discarding redundant and irrelevant features present in a high dimensional dataset (Gikera et al., 2023a).

In cases where there were significant differences among the dimensionality reduction methods, the specific variations were mainly due to significant differences between the autoencoders and each of the other dimensionality reduction methods, an indication that the autoencoder had a more relatively better performance as compared to the dimensionality reduction methods.

This observation is in line with the one made by Fournier and Aloise (2019). In their research study on the empirical comparison between autoencoders and traditional dimensionality reduction methods, it was observed that the autoencoders outperformed the other methods on the MNIST, Fashion-MNIST and CIFAR-10 datasets. This was attributed to the higher flexibility of the autoencoder method. However, on the other hand, there were no significant performance differences among several dimensionality reduction methods. This was attributed to the similar algorithmic approaches and mathematical principles for dimensionality reduction, among them. This is in line with the observation made by Van Der Maaten et al. (2009). Based on this observation, this research study concludes that a number of data dimensionality reduction methods can be substituted among themselves within specific problem domains, without a significant compromise on the quality of clustering results. These observations are in line with the one made by McInnes et al. (2018) and Van Der Maaten et al. (2009). In this specific research study, for example, it was identified that the quality of clustering based on the autoencoder and UMAP did not have significant differences while those of the UMAP and the kPCA had significant differences.

The results in Table 4.7 show absence of significant interactions between the different *k*-hyperparameter tuning techniques (*k*HTs) and the dimensionality reduction methods (RMs) applied. This is an indication that the influence of the RMs on the techniques' Average Index Performance (AIPS) is consistent across different *k*HTs. Regardless of the *k*HT used, the impact of the RM remains consistent with the RMs affecting the performance of all *k*HTs similarly. This impact exists without any significant variations based on the *k*HT itself. Although the choice of the RM had a statistically significant difference in the AIPS of the *k*HTs, this index performance score did not vary depending on the choice of the kHT.

The similarity in the response to the changes induced by the different RMs without showing significantly differential response or sensitivity to the transformations generated by the different RMs during the .*k*HT tuning process creates equity, reliability and fairness. This is attributed to the fact that RMs affects all algorithms uniformly without any discernible influence that's specific to certain algorithms. This outcome is supported by the observation made in the works of Draganov et al. (2021) where the PCA, TSNE, and the UMAP methods exhibit similar effect on the algorithms.

5.2.1.4 Two-way ANOVA and interactions on the CBTs of the *k*HTs in different RMs applied

The results from the two way ANOVA in Tables 4.10 and 4.11 considers the interactions on the cluster building times (CBTs) of the *k*—hyperparameter tuning techniques (*k*HTs) in different dimensionality reduction methods (RMs). In Table 4.10, the absence of a significant effect on the *k*HT in different RMs is an indication that the CBT doesn't show notable sensitivity across different RMs. Regardless of the RM applied, the *k*HTs display consistent CBTs, and no particular RM has a significant effect on the CBT of any specific *k*HT. This can be attributed to the fact that the different *k*HTs respond similarly to the variations introduced by the RMs, and no particular RM causes substantial changes in the CBTs for any given *k*HT. This observation is supported by the one made by Gare et al. (2021) where both the UMAP and the t-SNE did not have substantial variations in the run times of the algorithms used for the preclinical and drug screening experiments. In Table 4.10, the significant effect on the RM is an indication of a differential impact of the RMs with each RM causing a discernible change in the CBTs of the *k*HTs. There are varying effects on the CBTs of the *k*HTs observed across the different RMs.

This can be attributed to the computational demands of the different RMs, including the mathematical computations and iterative processes involved in transforming and compressing the high dimensional spaces. This observation is in line with the one made by Becht et al. (2019) where the UMAP dimensionality reduction method demonstrated a faster run time as compared to the t-SNE dimensionality reduction method. In this specific empirical study, for example, the PCA and t-SNE exhibited significant differences on the cluster building times while the LLE and UMAP did not. The results in Table 4.10 shows that no significant interaction between the *k*HT and the RM on the CBTs. This absence is an indication that the effect of the RM on the CBTs doesn't vary significantly based on the choice of the *k*HT. Regardless of the *k*HT used, the effect of the RM on the CBTs remains relatively similar. This can be attributed to the fact that the computational demands introduced by the RM are independent of the specific *k*HT. It might also be attributed to the fact that the efficiency gains or computational load reductions achieved by employing various RMs are consistent across the different *k*HTs, hence no significant interaction observed. This observation is in line with the empirical works of Xia et al. (2021) where both the UMAP and t-SNE have a heavy influence on the cluster and the member identification independent of the algorithms used in the empirical study.

5.2.1.5 Two-Way ANOVA and interactions on the AIPS of the *k*HTs in HDDs of varied DLs

The results from the two way ANOVA in Tables 4.12 and 4.13 considers the Average Index Performance Score (AIPS) of the *k*-hyperparameter tuning techniques (*k*HTs) in datasets of varied dimensionality levels (DLs). The results in Table 4.12 show a significant effect of the *k*HT, an indication that that the choice of the *k*HT has a notable and distinct impact on the quality of clustering when considered independently, regardless of the DL.

This could be attributed to the fact that different algorithms are designed with unique methodologies, complexities, or strategies. These inherent characteristics can significantly impact their performance and efficiency, showcasing distinct behaviors across various datasets. This observation is in line with the empirical analysis works of Onan et al. (2016) where five classification algorithms and five ensemble methods exhibited different classification accuracies on four different datasets. The results in Table 4.12 show no significant effect of the DL, an indication that the variations in the DLs of a dataset don't lead to significant differences in the outcome of the clustering quality when considered independently of the choice of the *k*HT. The changes in the DL considered on its own do not produce a substantial impact on the clustering quality, independent of the *k*HT used. This can be attributed to the dimensionality independence of the *k*HTs where the dimensionality reduction methods within the *k*HTs are designed to operate effectively across various dimensionalities, minimizing the impact of changes in DLs on the quality of clustering. This observation is in line with the works of McInnes et al. (2018) where the UMAP RM exhibits similar level of effectiveness across datasets of varied DLs.

Table 4.12 show no significant interaction effect between the *k*HT and the DLs of the datasets. This is an indication that that the combined impact of changing the *k*HTs and DLs doesn't lead to significant variations in the quality of clustering. It implies that while both the choice of the *k*HT and the changes in the DLs of the dataset might affect the Average index performance score (AIPS) at the individual level, their combined effects when considered together do not lead to noticeable differences or interactions that significantly alter this performance. This can be attributed to the fact that each factor i.e. the choice of the *k*HT and the DL might independently affect the AIPS without their effects being significantly influenced or modified by each other.

Alternatively, this can be attributed to the fact that the *k*HTs might exhibit consistent performance across the different DLs of the input datasets and the *k*HTs might behave similarly or maintain stable performance regardless of changes in the dataset's DLs. This observation is in line with the empirical analysis works of Kumar et al. (2019) where the *k*-Nearest Neighbor algorithm exhibits stable performance across Twitter, YouTube and Instagram datasets consisting of different dimensionality levels.

5.2.1.6 Two-way ANOVA and interactions of the DTs and DLs of HDDs on the AIPS of the *k*HTs

The results from the two way ANOVA detailed in Table 4.14 considers the interaction effect of the Average Index Performance Score (AIPS) of the *k*-hyperparameter tuning techniques (*k*HTs) based on the data type (DT) and the dimensionality level (DL) of input datasets. The presence of a significant effect of the DT, at the individual levels, is an indication that different DTs lead to discernible and significant variations in the AIPS, independent of the DL considered. The significance of this effect indicates that the selection of different DTs significantly influences the AIPS, and these differences are substantial enough to be attributed specifically to the DT, not influenced by the DL. This can be attributed to the data structure and complexity effect where the different DTs possess unique structures, complexities, or distributions. These inherent attributes can significantly impact how the *k*HTs perform on each DT, leading to significant variations in performance. This observation is in line with the empirical analysis works of Onan et al. (2016) where five classification algorithms and five ensemble methods exhibited different classification accuracies on four different datasets.

On the other hand, Table 4.14 shows no significant effect of the DL, at the individual levels. This implies that varying the DL does not lead to discernible or significant variations in the AIPS, independent of the DT considered. It implies that changing the DL does not notably impact the quality of clusters, indicating that these differences in dimensionality do not significantly influence the AIPS, regardless of the DT considered. This could be attributed to the "feature relevance across dimensions" effect where essential features required for quality clustering process might remain relatively stable or relevant across different dimensionality levels, ensuring consistent performance. This observation is in line with the empirical analysis works of Kumar et al. (2019) where the *k*-Nearest Neighbor algorithm exhibits stable performance across Twitter, YouTube and Instagram datasets consisting of different dimensionality levels.

The two-way ANOVA analysis in Table 4.14 shows that there are no significant interactions between the DT of the input dataset and its DL on the AIPS of the *k*HTs. Although the choice of the DT of the input dataset had a statistically significant difference on the AIPS of the *k*HT, this performance did not vary depending on its DL. The number of features present in a dataset determines the DL in a dataset. How well a data dimensionality reduction method (RM) handles a specific DT, containing any number of features, is a major determinant on the quality of clustering results. This observation is in line with the one made by Dash et al. (2010) and Zebari et al. (2020). In the works of Dash et al. (2010), making the best choice on the RM helped to effectively remove the redundant features from the Breast cancer and SPECTF Heart datasets, thereby improving the performance of the new algorithm to a great extent.

5.2.1.7 Two-way ANOVA and interactions of the DTs and DLs of HDDs on the CBTs of the *k*HTs

Tables 4.15 to 4.18 shows the results of the two-way ANOVA analysis that examines the cluster building times (CBT) based on the data type (DT) and the dimensionality level (DL) of the input datasets. The presence of a significant effect of the DT, at the individual level, is an indication that the different DTs lead to discernible and significant variations in the *k*HTs' CBTs, independent of the DL considered. The significance of this effect indicates that the selection of different DTs significantly influences the CBTs, and these differences are substantial enough to be attributed specifically to the DT, not influenced by the DL. This can be attributed to the "data transformation and computational load" effect where the varied DTs might require different transformations and computations during processing, impacting the overall CBTs. This observation is in line with the works of Geng et al. (2020) where different DTs exhibited different run times.

On the other hand, Table 4.15 shows a significant effect of the DL, at individual level. This is an indication that varying the DL leads to discernible and significant variations in the *k*HTs' CBTs, independent of the DT considered. The significance of this effect indicates that changing the DL significantly influences the CBTs, and these differences are substantial enough to be attributed specifically to the DL, not influenced by the DT. This observation is in line with the works of Zebari et al. (2020) where the DLs of the datasets, measured in terms of the number of features present in a particular dataset, has a direct proportionate relationship with the algorithms run times.

Table 4.15 shows a significant interaction between the DT and the DL of the input datasets on the CBTs. This is an indication that the influence of DTs on the CBTs of the *k*HTs varies significantly across different DLs. The effect of utilizing different DTs on the *k*HTs' CBTs is not consistent across all DLs. Instead, the impact of the DT on the *k*HTs' CBTs depends significantly on the specific DL of the dataset. This can be attributed to the fact that different data types might present varying levels of complexity in their representations across different dimensionalities. Processing and handling these representations can vary in difficulty, leading to varied impacts on the *k*HTs' CBTs across the different DLs. At the same time, the different DTs might present varying levels of complexity in their representations across different dimensionalities and processing these representations can vary in difficulty, leading to varied impacts on the *k*HTs' CBTs across the different DLs.

This observation is in line with the one made by Danasingh et al. (2020). In this specific study, the invention of a new algorithm assisted in an effective identification of the irrelevant and redundant features in ORL10P, PIE10P and DBWorld e-mails datasets, thereby lowering the run times of the classification tasks by almost sixty seconds. In this empirical study, for example, the text datasets of DL P3 and the video datasets of the dimensionality level P4 exhibited statistically significant differences in the CBTs while the image datasets of DL P4 and the audio datasets of DL P3 did not.

5.2.1.8 Two-way ANOVA and interactions of the DTs of HDDs and RMs on the AIPS of the *k*HTs

Tables 4.19 to 4.21 presents the two-way ANOVA analysis examining the Average Index Performance Score (AIPS) based on the choice of dimensionality reduction method (RM) and the data type (DT) of the input datasets.

The results show a significant effect of the RM, at the individual levels, an indication that different RMs lead to discernible and significant variations in the AIPS, independent of the DT considered. The significance of this effect indicates that the selection of the different RMs significantly influences the AIPS, and these differences are substantial enough to be attributed specifically to the RM, not influenced by the DT. This can be attributed to the DT adaptability effect where some RMs are specifically designed and optimized for certain DTs and the specialization could lead to significant variations in performance across different data types. This could also be attributed to the distinct transformation and feature representation effect where RMs employs distinct transformations to represent high-dimensional spaces into lower dimensional spaces. On the other hand, Table 4.19 shows that there is a significant effect of the DT, at the individual levels, implying that different DTs lead to discernible and significant variations in the AIPS, independent of the RM applied. The significance of this effect indicates that the selection of different DTs significantly influences the AIPS, and these differences are substantial enough to be attributed specifically to the DT, not influenced by the RM.

This can be attributed to the information content and representation effect where various DTs contain different amounts and types of information. This effect might arise from how RMs interact with and represent information present in each DT. The significance might therefore arise from the different ways these RMs capture and represent data structures. These observations are in line with the empirical analysis works of Usman et al. (2021) where both the PCA and t-SNE performs differently on the antioxidant protein sequences dataset. Table 4.19 shows that there is significant interaction effect between the RM and the DTs of the input datasets.

This is an indication that the influence of employing different RMs on the performance of the *k*HTs depends on the specific characteristics of the data being processed. Therefore, the effectiveness of the RMs differs across the various DTs of the input datasets. This can be attributed to the different DTs. These DTs possess distinct structures and properties and certain RMs become more effective for specific DTs due to their ability to capture and preserve essential information inherent in that DT. The distinct structures and properties of the text and video datasets are more distinct as compared to those of video and audio datasets. Therefore, it is concluded that selecting the appropriate RMs that are most compatible with the characteristics of the data being processed, enhances the effectiveness of subsequent techniques operations, and is an important step in the *k*HT tuning process in high dimensional space clustering. This observation is in line with the one made by Xiang et al. (2021). In this comparative study, the t-SNE demonstrated a better accuracy as compared to the UMAP on the RNA seq data due to the relative effectiveness as compared to the latter. This observation is also in line with the one made by Van Der Maaten et al. (2009) where the kernel PCA performed equally strong as the autoencoder, across a number of datasets meaning that they both can be substituted in place of the other on similar DTs. However, this research study notes that although specific RMs are effective for specific DTs, such methods may perform similarly or differently. This can be attributed to the fact that the RMs may be similarly computationally efficient or different and scale similarly or differently to the specific DTs. The RMs may use distinct or similar algorithmic approaches and mathematical principles to transform the high dimensional spaces into low dimensional spaces. In this specific empirical study, there were no significant differences between the autoencoders and UMAP-learn on image datasets.

This means that the UMAP learn method can be substituted in place of the autoencoder method, and vice versa, when clustering high dimensional image datasets. In another example, in video datasets, there were significant quality differences between the autoencoder method and the Spatial Temporal LLE, autoencoder and PCA with Spatiotemporal Cubes as well as between the autoencoder and economy SVD on the video datasets. However, there were no significant differences, in the clustering quality, between the autoencoder and the FA based Factorization machine as well as between the Spatio-Temporal UMAP and the multi-core t-SNE methods on the same video datasets. In the latter case, the dimensionality reduction methods can be substituted among themselves without compromise on the AIPS of the *k*HT. At the same time, these RMs can be substituted among themselves in the unsupervised transfer learning process of the new technique's autoencoder.

5.2.1.9 Two-way ANOVA and interactions of the DTs of HDDs and RMs on the CBTs of the *k*HTs

Tables 4.22 and 4.23 presents the two-way ANOVA analysis examining the cluster building times (CBTs) based on the choice of dimensionality reduction method (RM) and the data type (DT) of the input datasets. The results show that there is a significant effect of the DT, at the individual level, an indication that the different DTs lead to discernible and significant variations in the *k*HTs' CBTs, independent of the RM applied. The significance of this effect indicates that the selection of the different DTs significantly influences the CBT, and these differences are substantial enough to be attributed specifically to the DT, not influenced by the RM applied. This can be attributed to the data processing complexity effect where the different DT, each with its unique intrinsic characteristics, may require distinct preprocessing and computational steps, leading to varying computational loads and subsequent cluster building times.

This observation is in line with the works of Geng et al. (2020) where different data types exhibited different run times. In this specific empirical study, for example, text and video datasets exhibited significant differences in the cluster building times while the video and audio datasets did not. This implies that the intrinsic characteristics and data structures of the video and audio datasets are more similar as compared to those of the text and video datasets. Table 4.22 on the other hand demonstrates that there is significant effect of the RMs, at the individual levels. This is an indication that the different RMs lead to discernible and significant variations in the CBTs, independent of the DT considered.

The significance of this effect indicates that the selection of the different RMs significantly influences the CBTs, and these differences are substantial enough to be attributed specifically to the RM, not influenced by the DT. This can be attributed to the fact that the different RMs employ distinct algorithms and approaches that vary in their computational complexity including inherent computational demands of each method. Each RM applies unique transformations to the high-dimensional spaces and the variation can lead to differences in the computational efficiency and subsequent run times.

This observation is in line with the one made by Becht et al. (2019) where UMAP RM demonstrated a faster run time as compared to the t-SNE RM. In this research study, the faster run time of the UMAP is as a result of its more efficient dimensionality reduction process as compared to the one by the t-SNE. The significant effect on the DT, at individual level, on the CBTs of the two way ANOVA analysis on the interaction between the DTs of the input datasets and the RMs indicates that the choice of DT significantly influences the CBTs of the *k*HT. The CBT of the *k*HTs are notably different across various DTs when analyzed independently of the RMs used.

This can be attributed to the fact that algorithms perform various statistical and mathematical operations that can be influenced by the DT and the different STs might necessitate distinct calculations and computations, impacting algorithm run times accordingly. This observation is in line with the one made by Kapoor et al. (2017). In this specific comparative study, Kapoor and Singhal (2017) found out that all the three algorithms performed differently on the iris dataset due to their structural differences.

5.2.1.10 Two-way ANOVA and interactions of the DLs of HDDs and RMs on the AIPS of the *k*HTs

Table 4.24 presents the two-way ANOVA analysis examining the Average Index Performance Score (AIPS) based on the choice of the dimensionality reduction method (RM) and the dimensionality level (DL) of the input datasets. The absence of a significant effect of the DL at the individual level is an indication that varying the DL does not lead to discernible or significant variations in the AIPS, independent of the RM considered. The absence of a significant effect suggests that changing the DL does not notably impact the AIPS, indicating that DLs do not significantly influence the performance, regardless of the chosen RM. This can be attributed to the data structure preservation effect where the important structural information and patterns within the data remains relatively consistent across different DLs, leading to consistent performance regardless of DLs of the input dataset. Alternatively, this can be attributed to the fact that the RMs effectively captures the essential information even across various DLs, resulting in a stable performance. This observation is in line with the one made by Fournier and Aloise (2019). In the research work, the PCA RM performed consistently well across a number of similar types of datasets, regardless of the number of features present in these datasets.

On the other hand, the presence of a significant effect of the RM, at the individual level, is an indication that different RMs lead to discernible and significant variations in the performance outcome, independent of the DL considered. The significance of this effect indicates that the selection of the RM significantly influences the AIPS, and these differences are substantial enough to be attributed specifically to the RM, not influenced by the DL. This can be attributed to the information preservation and representation effect where each RM captures and represents information differently. The variations in how each method handles information might result in significant differences in the quality of the reduced representations, influencing the AIPS. This observation is in line with the one made by Xiang et al. (2021). In their comparative study, the t-SNE demonstrated a better accuracy as compared to the UMAP on the RNA seq data due to the relative effectiveness of the t-SNE as compared to UMAP in handling the RNA seq dataset.

In table 4.24, the absence of the interaction effect between the DLs of the input datasets and the RMs is an indication that the influence of the RMs on the AIPS in the *k*HT remains consistent across the different DLs. It suggests that while RMs might affect the AIPS of the *k*HT, this impact does not vary significantly based on the specific DLs of the input datasets. Although the choice of the RM has a statistically significant difference in the AIPS of the *k*HTS, this performance did not vary depending on the DLs present in an input dataset. This can be attributed to the fact that RMs retains similar proportions of the important information captured from the underlying structures across the different DLs, resulting in consistent impacts on the *k*HTS' performance.

This observation is in line with the one made by Fournier and Aloise (2019). In their research work, the PCA dimensionality reduction method performed consistently well across a number of similar types of datasets, regardless of the number of features present in these datasets. For this reason, based on these research studies, it is concluded that the number of features present in a dataset should not determine the choice of the RM for use. Instead, the specific type of a input dataset is the one that should determine the choice of the RM. A particular RM works well on a specific type of dataset that it is best suited for regardless of the number of features present in such a dataset.

5.2.1.11 Two-way ANOVA and interactions of the DLs and RMs on the CBTs of the *k*HTs

Tables 4.25 and 4.26 presents the two-way ANOVA analysis examining the cluster building times (CBTs) based on the choice of the dimensionality reduction method (RM) and the dimensionality level (DL) of the input datasets. The presence of a significant effect of the DL, at the individual level, is an indication that varying the DL leads to discernible and significant variations in the CBTs, independent of the RM considered. The significance of this effect indicates that changing the DL significantly influences the CBTs, and these differences are substantial enough to be attributed specifically to the DL, not influenced by the RM. This can be attributed to the computational complexity with dimensionality where computational requirements for various operations might change as the dimensionality of the data increases or decreases, leading to varying run times. This observation is in line with the works of Zebari et al. (2020) where the DLs of the datasets, measured in terms of the number of features present in a particular dataset, has a direct proportionate relationship with the algorithms' run times.

On the other hand, table 4.25 shows a significant effect of the RM, at the individual level, an indication that different RMs result in discernible and significant variations in the *k*HTs' CBTs, independent of the DL considered. The significance of this effect indicates that the selection of the RMs significantly influences the CBTs, and these differences are substantial enough to be attributed specifically to the RM, not influenced by the DLs. This can be attributed to the data transformation and manipulation effect where the different RMs apply unique transformations and manipulations to reduce dimensions with varying computational costs that influences the overall CBTs on the *k*HTs.In table 4.25, the presence of a significant interaction effect between the DLs of the input datasets and the RMs is an indication that the impact of applying RMs on the *k*HTs CBTs varies significantly across the different DLs.

The influence of employing different RMs on the *k*HTs' CBTs is not consistent across the different DLs. Instead, the effect of the RM on the CBTs depends on the specific DL of the input dataset. This can be attributed to the fact that different RMs might incur varying computational loads across different DLs and the impact on the techniques' CBTs can differ as the RMs interact differently with the complexities arising from the various dimensionalities. This observation is in line with the one made by Becht et al. (2019) where UMAP dimensionality reduction method demonstrated a faster run time as compared to the t-SNE dimensionality reduction method. In their research study, the faster run time of the UMAP is as a result of its more efficient dimensionality reduction process as compared to the one by the t-SNE.

5.2.1.12 Two-way ANOVA analysis and interactions on the CBTs of *k*HTs in HDDs of varied DLs

Table 4.26 presents the two way ANOVA results of the cluster building times (CBTs) of *k*-hyperparameter tuning techniques (*k*HTs) in datasets of varied dimensionality levels (DLs).

The absence of a significant effect on the *k*HT on the CBTs, at the individual levels, is an indication that the techniques' CBTs aren't significantly impacted by changes in the DLs of the datasets. This can be attributed to the fact that the *k*HT in high dimensional space clustering are usually optimized for and their performance might remain consistent across various dimensionalities, leading to a non-significant effect in analysis. This is in line with the systematic review works by Gikera et al. (2023a) where a number of the *k*-means based techniques used for the *k*-hyperparameter tuning are optimized for the high dimensional spaces only. The presence of a significant effect on the DLs of the input datasets on the CBTs, at the individual levels, is an indication that as the DLs of the datasets varies, the CBTs of the *k*HTs significantly differs across these DLs. This can be attributed to the fact that higher dimensionality can introduce redundant or irrelevant features, impacting algorithms differently. Algorithms sensitive to noise or irrelevant dimensions might exhibit significant variations in runtime as the dimensionality changes due to increased processing requirements. This observation is in line with the one made by Danasingh et al. (2020). In this specific study, the invention of a new algorithm assisted in an effective identification of the irrelevant and redundant features in ORL10P, PIE10P and DBWorld e-mails datasets, thereby lowering the run times of the classification tasks by almost sixty seconds. In this research study, for example, the datasets of the DL P3 had statistically significant differences in the CBT with those of the DL P4.

In Table 4.26, the absence of a significant interaction effect between the *k*HT and the Dls of the datasets on the CBTs is an indication that the effect of changing the dimensionality on the CBTs does not significantly differ across the *k*HTs. Therefore, while there might be differences in overall CBTs among the techniques, the way these run times change based on varying DLs of the datasets remains consistent across the *k*HTs. This can be attributed to the fact that the different *k*HTs respond similarly to the changes in DLs.

If their sensitivity or resilience to high dimensionality is comparable, it could result in a lack of significant interaction, indicating consistent behavior across the algorithms. Moreover, the techniques might scale their computational efforts consistently with increasing DLs of the input datasets. If the rate of increase in the CBTs is uniform across the techniques as the DLs changes, it leads to a non-significant interaction effect.

5.2.2 Empirical Analysis Results on the three-way ANOVA and Interactions

5.2.2.1Three-way ANOVA on the AIPS based on the DT of HDDs, DLs of HDDs and RMs applied

Tables 4.32 to 4.34 presents the three-way ANOVA analysis focusing on the performance of the *k*HT based on the dimensionality reduction method (RM), dimensionality level (DL), and data type (DT). The absence of significant interactions among the three variables, jointly, is an indication that the combined influence of these variables on the techniques' quality of clustering doesn't exhibit significant variations or interactions that affect the Average Index Performance Score (AIPS) when considered together. In essence, the effects of the RM, DL and the DT on the techniques' performance don't interact in a way that significantly alters the outcome when all three factors are analyzed simultaneously. This can be attributed to the orthogonal effects where the three variables might be independent or non-interacting with each variable exerting its influence on the techniques performance independently, without their combined effects leading to significant variations or interactions. However, there are significant interactions of both the RMs and DTs of the input datasets at the individual levels. There are also significant interactions between the RMs and the DTs of the input datasets. The significant interactions between the DTs of the input datasets and the RMs are attributed to the fact that different RMs worked better among some DTs, compared to how they worked in other DTs.

They were more suited to some types of datasets than in others. Therefore, it was concluded that the selection process of the most suitable RM for a particular type of dataset must therefore be done in a thoughtful manner. This observation is in line with the one made by Mohanty et al. (2014) and Zhu et al. (2013).

5.2.2.2 Three-way ANOVA on the AIPS based on the choice of the *k*HTs, DTs of HDDs and RMs applied.

Tables 4.35 to 4.39 presents the three-way ANOVA analysis on the techniques' Average Index Performance Score (AIPS) based on the dimensionality reduction method (RM), choice of the *k*-hyperparameter tuning technique (*k*HT) and the data type (DT) of the input dataset.

The absence of significant interactions among the three variables, jointly, is an indication that the combined influence of these factors on the *k*HTs' performance doesn't exhibit significant variations or interactions that impact the outcome when considered together. In essence, when all three factors are analyzed simultaneously, their combined effects do not lead to significant alterations or interactions that affect the techniques AIPS differently. This can be attributed to the fact that each variable might independently affect the techniques performance without significant interactions among them and their individual influences on the performance might operate separately without substantial combined effects. However although there were no significant interaction among the three factors, jointly, there were significant interactions among the DT, RM and the *k*HT, at the individual levels. This is attributed to the structural differences among each of them, as earlier discussed. There was also a significant interaction between the DT of the input dataset and the RM attributed to the fact that the best choice of a RM on a specific type of a dataset is crucial for improved performance of a *k*HT. There were mainly no significant interactions between the *k*HT and the RM.

This is attributed to the fact that any RM can work well irrespective of the choice of the *k*-hyperparameter tuning technique. Although there were significant interactions between the DT of the input dataset and the RM applied, this performance did not vary depending on the choice of the *k*HT. Therefore, any RM can be used with any *k*HT, so long as it is effective to handle the respective input datasets. This observation is in line with the one made by Dash et al. (2010) and Zebari et al. (2020). In this review work, the focus on the different data RMs is not tied to a particular algorithm.

5.2.2.3 Three-way ANOVA on the CBTs of the *k*HTs, DTs and DLs of HDDs

Tables 4.40 to 4.43 presents the three-way ANOVA analysis investigating the cluster building times (CBTs) based on the choice of the *k*-hyperparameter tuning technique (*k*HT), dimensionality level (DL) and data type (DT) of the input dataset. The absence of significant interactions among the three variables, jointly, is an indication that their combined influence on the CBTs doesn't exhibit significant variations or interactions that impact the outcome when considered together. In essence, when all three factors are analyzed simultaneously, their combined effects do not lead to significant alterations or interactions that affect techniques' CBTs differently. This can be attributed to the orthogonal effect where each variable might independently affect the CBTs without significant interactions among them and their individual influences on the CBTs might operate separately without substantial combined effects. However, at their individual levels, all the three factors have an effect on the CBTs. Moreover, there was a significant interaction between the DT and the DL of the input dataset on the cluster building times. The significant interaction between the DT and the DL on the CBTs of the *k*HTs suggests that the effect of DT on the CBTs varies significantly across different DLs, i.e., the impact of the DT on the techniques CBTs is not consistent across all DLs.

This interaction implies that when the *k*HT builds clusters, the time it takes to perform this task is influenced differently by the type of data and the DL it's operating within. The interaction suggests that the relationship between DT and CBTs changes or is modified based on the specific DL of the input dataset. This could be attributed to the performance dependency on DT and dimensionality where the techniques' ability to handle different DTs efficiently in the process of building clusters is contingent on the DL. Some data types might perform better or worse in cluster building tasks depending on the dimensionality of the dataset. This observation is in line with the one made by Bolívar et al. (2022) where the Synthetic Minority Over-sampling Technique, SMOTE, for dealing with the class imbalanced datasets does not perform well on the datasets with relatively higher dimensionality as compared to the other techniques.

5.2.2.4 Three-way ANOVA on the CBTs of *k*HTs, DLs of HDDs and RMs applied

Tables 4.44 to 4.46 presents the three-way ANOVA analysis exploring the cluster building times (CBTs) based on the dimensionality reduction method (RM), choice of the *k*-hyperparameter tuning technique (*k*HT) and the dimensionality level (DL) of the input dataset. The absence of significant interactions among the three variables, jointly, suggests that the combined influence of these factors on the CBTs doesn't exhibit significant variations or interactions that impact the outcome when considered together. In essence, when all three factors are analyzed simultaneously, their combined effects do not lead to significant alterations or interactions that affect the CBTs differently. This can be attributed to the orthogonal effect where each variable might independently affect the techniques CBTs without significant interactions among them and their individual influences on the CBTs might operate separately without substantial combined effects.

However, at their individual levels, both the RM and the DLs have an effect on the CBTs. This is attributed to the structural differences among the RMs as well as to their dimensionality reduction strategies that give rise to different run times during processing. At the same time, the number of features had a direct relationship on the CBTs because of the fact that datasets with extremely high number of features require more run times to reduce them to lower dimensions as compared to datasets that have lesser number of features. This observation is in line with the one made by Danasingh et al. (2020).

5.2.2.5 Three-way ANOVA on CBTs of the DTs of HDDs, DLs of HDDs and RMs applied

Tables 4.47 presents the three-way ANOVA analysis exploring the cluster building times (CBTs) based on the RM, choice of the *k*-hyperparameter tuning technique (*k*HT) and the dimensionality level (DL) of the input dataset. The absence of the significant interactions among the three variables, jointly, suggests that the combined influence of these factors on the CBTs doesn't exhibit significant variations or interactions that impact the outcome when considered together. In essence, when all three factors are analyzed simultaneously, their combined effects do not lead to significant alterations or interactions that affect CBTs differently. However, at their individual levels, these variables have significant differences. The significant differences in both the DTs and the RMs were attributed to their structural differences that gave rise to the different run times during processing. On the other hand, the significant differences in the DLs were attributed to the number of features present in the input datasets; this dimensionality is directly proportional to the cluster building times. These observations are in line with the one made by Allaoui et al. (2020).

5.2.2.6 Three-way ANOVA on the CBTs of the *k*HTs, DTs of HDDs and RMs

Tables 4.48 presents the three-way ANOVA analysis exploring the cluster building times (CBTs) based on the dimensionality reduction method (RM), choice of the *k*-hyperparameter tuning technique (*k*HT) and the data type (DT) of the input dataset, the absence of significant interactions among the three variables, jointly suggests that the combined influence of these factors on the CBTs doesn't exhibit significant variations or interactions that impact the outcome when considered together. The absence of significant interactions among these variables implies that the effects of each variable on the CBTs do not significantly change when considered in conjunction with the other variables. In essence, when all three factors are analyzed simultaneously, their combined effects do not lead to significant alterations or interactions that affect cluster building times differently. However, at the individual levels, there are significant effects on both the DT of the input datasets as well as the RMs. This is attributed to the structural differences within the different DTs and the RMs, a characteristic that gives rise to different run times during the processing. Although the choice of the RM, depending on the type of the input dataset, had a statistically significant difference on the CBTs, the CBTs did not vary depending on the choice of the *k*HTs. This therefore concludes that the choice on the RMs and the DT has a more significant effect on the CBTs as compared to the *k*HT itself. This observation is in line with the one made by Zebari et al. (2020). In this specific review, the PCA, kernel PCA and the locally linear embedding methods, LLE, generated different run times on the Indian Pines and Pavia University dataset.

5.2.2.7 Three-way ANOVA on the AIPS based on the choice of the *k*HTs, DLs of the HDDs and RMs applied

Table 4.49 presents the three-way ANOVA analysis examining the performance based on the choice of the technique (*k*HT), dimensionality level (DL), and the data dimensionality

reduction method (RM). The absence of significant interactions among the three variables, jointly, is an indication that the combined influence of these variables on the quality of clusters doesn't exhibit significant variations or interactions that affect the performance. The impact of each variable on the techniques' quality of clustering doesn't change when considered together with the other variables. The effect of the choice of the *k*HT, DL, and RM on the quality of clustering doesn't interact in a way that significantly alters the performance outcome when all three factors are considered simultaneously. This can be attributed to the fact that the effects of the choice of *k*HT, DL, and RM method on the performance might operate independently of each other and their individual influences on the performance might not depend on or interact with the other variables in a significant manner. Alternatively, this could be attributed to the fact that each variable might have a consistent and stable influence on the quality of clustering, regardless of the presence of the other variables. Their combined effects might not lead to significant variations or interactions that impact the quality of clustering differently. However, at their individual levels, both the *k*HTs and the RMs had a statistically significant difference in Average Index Performance Score (AIPS). With the *p*-value of less than 0.01 on the RM, it is concluded that the choice of the dimensionality reduction method is a statistically more significant factor on the AIPS as compared to the choice of the *k*HT itself. This observation is in line with the one made by Xiang et al. (2021).

5.2.2.8 Three-way ANOVA on the AIPS based on the *k*HTs, DTs and DLs of HDDs

Tables 4.50 presents the three-way ANOVA analysis investigating the quality of clustering based on the choice of the *k*-hyperparameter tuning technique (*k*HT), dimensionality level (DL) and DT of input datasets. The absence of a significant interaction among the three variables, jointly, implies that the combined influence does not lead to substantial variations on the quality of clustering.

While each factor might individually influence this outcome, their combined effects, when analyzed together, do not produce significant changes in the quality of clustering. This can be attributed to the "independence of factors" effect where each variable might independently affect the quality of clustering without their effects being significantly influenced or modified by the other variables. Their effects might be additive or independent of each other. This observation is in line with the works of Rodriguez et al. (2019), Onan et al. (2016) and that of Kumar et al. (2019). In the comparative research works of Rodriguez et al. (2019), different clustering algorithms performed differently on both the DB2C10F and DB10C10F datasets indicating the individual effect of the algorithm itself to the performance. In the empirical analysis works of Onan et al. (2016), the four different datasets exhibited different classification accuracies when the five different classification algorithms and the five ensemble methods were used on these datasets, indicating the individual effect by the data type on the performance.

5.2.3 Evaluation Results on the Effectiveness of the new technique

Table 4.50 presents the one-way ANOVA analysis on the evaluation experiments investigating if there were statistically significant differences in the Average Index Performance Score (AIPS) on both the existing and the newly developed *k*HTs in a variety of datasets. This is in line with the third objective. Based on the results, it is evident that their performances on different datasets are statistically significantly different. The statistically significant differences in the performances of these techniques are attributed to the structural differences among them. Due to these structural differences, the different techniques handle the datasets differently, to a great extent.

This observation was in line with the one made by Kapoor and Singhal (2017) in their comparative study of the *k*-means, *k*-means++ and the fuzzy c-means. In this specific comparative study, Kapoor and Singhal (2017) found out that all the three algorithms performed differently on the iris dataset due to their structural differences. Moreover, the one-way ANOVA analysis on the CBTs of the *k*HTs in different datasets demonstrates that there are statistically significant differences in the CBTs across them. This can also be attributed to the structural differences, algorithmic and computational complexities, initialization strategies, convergence criteria and the stopping criteria on the different techniques. For example, different initialization methods can affect the number of iterations required and, consequently, the run times. This observation is in line with the one made by Meila and Heckerman (2013). In this specific experimental study, the classification expectation-maximization algorithm, due to its relatively efficient initialization method and better convergence settings, demonstrated a shorter run time as compared to both the hierarchical agglomerative clustering and the standard expectation-maximization algorithms.

During the evaluation process on the effectiveness of the newly developed *k*-hyperparameter tuning technique against the existing ones, the Kruskal-Wallis H statistic showed that the performance on these techniques were significantly different. Further post-hoc analysis using the ANOVA test showed that the new technique had a more superior performance and flexibility in handling a variety of high dimensional datasets. Moreover, the comparison of the visualizations of the different datasets by using both the standard and the improved autoencoder on the newly developed *k*HT in a variety of datasets showed that the improved autoencoder, developed through further hyperparameter settings and unsupervised transfer training strategies, generated better quality of clusters as opposed to the standard autoencoder.

Also, the whisker plot of the new technique was more superior compared to those of the best perfroming existing techniques. However, the CBTs of the new *k*HT were significantly higher than those of the existing *k*HTs. These higher run times are attributed to the complex structure of the new *k*HT including the multiple sets of hyperparameter settings as compared to those of the existing *k*HTs. The increased run times were also attributed to the fact that the algorithm of the new *k*HT loops repetitively until good cluster quality scores are obtained. This observation is in line with the one made by Kapoor and Singhal (2017).

5.3 Conclusion

In line with the first objective, the comprehensive empirical analysis examining the *k*-hyperparameter tuning techniques in a variety of datasets and dimensionality reduction methods is successfully achieved and raise some key conclusions. Firstly, the choice of the *k*-hyperparameter tuning technique significantly affects the AIPS at an individual level. The Mathematical models, assumptions, and optimization strategies within each technique substantially influence the performance differences among the techniques. However, although the choice of the *k*-hyperparameter tuning techniques is a statistically significant factor in the *k*-hyperparameter tuning processes, the dimensionality reduction methods and the data types are much more statistically significant factors. The data types significantly affect the performance of the *k*-hyperparameter tuning techniques, showcasing how they process specific data types, leading to varied quality of clustering outcomes. However, the performance differences among the data types remain consistent regardless of the chosen *k*-hyperparameter tuning technique as there are no interactions. While the *k*-hyperparameter tuning techniques and data types have influence on the clustering results at individual levels, their combination remains consistent across various data types, signifying stable performance.

The dimensionality reduction methods notably impact performance of the *k*-hyperparameter tuning techniques. Their unique characteristics in handling data types and features significantly influence clustering quality. The variations of the dimensionality level affect cluster building times, revealing that higher dimensions impact algorithms differently, potentially due to the redundant or irrelevant features, commonly referred to as the curse of dimensionality. The influence of the dimensionality reduction methods on the quality of clustering remains consistent, irrespective of the chosen *k*-hyperparameter tuning technique, maintaining equity in the performance. The interactions between data types, dimensionality levels, and *k*-hyperparameter tuning techniques showcase dependencies on specific dataset characteristics, dimensionality, and the selected techniques. The *k*-hyperparameter tuning techniques exhibit consistent performance despite varying dimensionality levels, emphasizing robustness in high-dimensional spaces. The effectiveness of a dimensionality reduction method depends on the specific type of dataset that it processes, advocating thoughtful method selection over feature quantity. The data type, dimensionality level, and dimensionality reduction methods significantly impact cluster building times, suggesting varied algorithmic sensitivities to data complexity and structural changes.

Variables like dimensionality level and data type don’t significantly impact clustering quality when considered independently of other factors. While the *k*-hyperparameter tuning techniques significantly impact clustering outcomes, their interaction with dimensionality reduction methods doesn't significantly affect clustering quality. Overall, this empirical research emphasizes the significance in understanding data type intricacies, and leveraging appropriate dimensionality reduction methods for effective *k*-hyperparameter tuning in high-dimensional space clustering.

It underscores the nuanced interplay between variables and the need for thoughtful selection of these parameters based on characteristics of the high dimensional input dataset in order to optimize the clustering outcomes. The empirical analysis outcome of the interplay among the different variables provided valuable insights and the leverage points that guided the development of the new state of the art *k*-hyperparameter tuning technique in high dimensional space clustering. The possible ways in which these insights guided the development of a state-of-the-art technique in a variety of high dimensional spaces was the epitome of the empirical analysis process, with the specific key findings listed in section 4.3, highlighting how the quintessentials guided the new technique's development methodology.

The research implication of these observations is the fact that the capability of a dimensionality reduction method in handling a particular type of a dataset, effectively, is seen as a strong leverage point in the development of a successful *k*-hyperparameter tuning process in high dimensional space clustering. Considering the superior performance of the autoencoder across all the datasets, the development of the novel technique is then based on the autoencoder and carefully selected architectural-related and training-related hyperparameter settings. Consequently, the choice of the standard PCA, UMAP learn, Multicore tSNE and the Spatial temporal UMAP in the unsupervised training process of the novel technique is because of their superior performance, second after the autoencoder, across the high dimensional text, images, audio and video datasets, respectively. This leads to a successful achievement of the second objective.

The evaluation of the newly developed *k*-hyperparameter tuning technique against the existing ones leads to a successful achievement of the third objective and reveals compelling insights.

Statistical analyses, including the Kruskal-Wallis H statistic and subsequent ANOVA tests, underscored significant differences in performance, highlighting the superior efficacy and adaptability of the new technique across diverse high-dimensional datasets. Notably, leveraging on improved autoencoder, refined through advanced hyperparameter settings and unsupervised transfer training strategies, based on the respective second best perfroming dimensionality reduction methods, yielded improved cluster quality compared to the standard autoencoder. This is also evinced in the cluster visualizations of the generated clusters. However, the new technique exhibited higher cluster building times attributed to its intricate structure and iterative processes for achieving optimal clustering outcomes. This findings align with the prior observations by Kapoor and Singhal (2017), emphasizing the trade-off between enhanced performance and increased computational demands in cutting-edge algorithmic developments in the future. Lastly, it is noted that across all the ANOVA tables, the last two cells of the residuals are blank. It is worth to note that the residuals don't have the P and F values as there are usually no hypothesis tests for them the same way that there would be hypothesis tests for the *k*-hyperparameter tuning technique, the dimensionality reduction method, the data type and the dimensionality level of a high dimensional dataset.

CHAPTER SIX: CONCLUSION & RECOMMENDATION FOR FUTURE WORK

6.1 Introduction

The research presented in this thesis addresses multiple areas in the *k*-hyperparameter tuning in high-dimensional space clustering. Firstly, the review of the literature in regards to the *k*-means based techniques for the *k*-hyperparameter tuning in high-dimensional space clustering is carried out in order to point out the research gaps within this research domain. The scope of this review comprises of the techniques themselves, dimensionality reduction methods used with high dimensional spaces as well as the high dimensional input datasets used with these techniques. The research gaps from the literature review process provide a theoretical foundation upon which the conceptual framework of this research is constructed. Subsequently, the conceptual framework guides the empirical research methodology in this study. The empirical study is conducted in order to investigate on the end to end behaviour of the existing *k*-hyperparameter tuning techniques, and the interplay, in a variety of high-dimensional datasets and data dimensionality reduction methods. The analysis on the experimental data from the main experiments in the empirical analysis process provides invaluable insights into the research contributions of both the technical and the theoretical aspects in this thesis. These are discussed in the subsequent sections 6.3.1 and 6.3.2, respectively.

The new *k*-hyperparameter tuning technique is based on the long standing canonical elbow method, unsupervised transfer learning strategy and a thoughtful selection of both the architectural-related and the training-related hyperparameter settings. In order to perform self-adaptation towards improving the quality of clustering in the new technique, the ensemble internal validation index, comprising of the four commonly used internal validation indexes, is used as the target.

Finally, the effectiveness of this new technique is demonstrated against the existing ones in a variety of high-dimensional datasets. These results demonstrate a relatively superior performance and flexibility on the newly developed technique as compared to the other techniques, across the variety of high dimensional spaces. To this end, this chapter draws conclusions from the entire research study, giving highlights of the research contributions and limitations as well as the recommendations for future research in the domain of the *k*-hyperparameter tuning in high dimensional space clustering.

6.2 Summary and conclusion

In Chapter one, the background of the study and the statement of the problem leads to the identification of the research objectives and hypothesis of this research study. The justification and significance of the study in a variety of clustering applications in real life gives an inspiration into the conduct of this research work. In Chapter two, the comprehensive theoretical analysis of the *k*-hyperparameter tuning techniques in a variety of high dimensional datasets and dimensionality reduction methods results into a number of key findings and a conceptual framework. These two acts as both the road map and the foundation for the subsequent empirical investigations in Chapter three, guided by a methodology based on mixed research methods for validation triangulation. The methodology in Chapter three comprises of both the experimental research methodology and the model development methodology for the proposed *k*-hyperparameter tuning technique. This leads to a successful achievement of the first objective. The analysis on the experimental data from the empirical analysis process, using the R-software focuses on the investigation of significant differences and interactions of the variables in the conceptual framework.

In Chapter four and five, the identification and discussion of the invaluable insights from the empirical analysis process underpinns both the technical and theoretical research contributions in Chapter 6 and leads to a successful achievement of both the second and the third objectives. One of the key research contributions is the development of a state-of-the-art technique that demonstrates effectiveness in the *k*-hyperparameter tuning process in a variety of high dimensional datasets. In summary, all the research objectives are achieved fully.

The novel technique is based on the canonical elbow method, an unsupervised transfer learning strategy on an autoencoder as well as a thoughtful selection of both the architectural-related and training-related hyperparameter settings based on the dataset. The superior performance of the novel technique has also been evinced by the ANOVA analysis, Kruskal-Wallis H statistic, whisker plots as well as through its improved cluster visualizations in a variety of high dimensional datasets. The limitations encountered during the research study as well as the measures taken to curb these limitations are also highlighted in Chapter six. The high computational costs and inefficiency during the entire empirical algorithmic exercise is the main limitation encountered in the setting up of the stopping criteria based on the maximum number of iterations and runs for each technique. Setting the recommendations for future research directions, in the area of *k*-hyperparameter tuning in high dimensional space clustering, marks the end of this research study. These specific recommendations are discussed in section 6.5

6.3 Research contributions

In light of the above summary and conclusion, this specific research study contributes to the researcher’s body of knowledge in the following ways:

6.3.1 Technical contribution

Within the domain of machine learning, this research study is a novel advancement in two technical contributions. Firstly, leveraging on the analysis results on the experimental data to develop a tailored data scientist's tool box is a significant technical contribution, where data scientists can tailor the *k*-hyperparameter tuning techniques to specific combinations of data types and dimensionality reduction methods. For instance, developing techniques that excel in clustering images in high dimensions might require different strategies compared to high dimensional text, video or audio dataset and vice versa.

Secondly, the development of a state-of-the-art *k*-hyperparameter tuning technique in high dimensional space clustering is another novel advancement in the domain of machine learning. This development is inspired by the invaluable insights derived from the data analyses of the empirical data, clearly showing the optimization leverage points and optimal hyperparameter settings for the improved technique. The new improved technique is based on the long standing canonical elbow method, unsupervised transfer learning strategy based on the respective best performing dimensionality reduction methods and a thoughtful selection of both the architectural-related and the training-related hyperparameter settings on the improved autoencoder. Such architectural and training related settings include: learning rate, activation functions, loss functions, regularization mechanisms, sparsity constraints, activation functions, number of layers, number of neurons, type of layers (convolutional vs fully connected), type of encoding and decoding layers, early stopping, type of architecture and the depth. The use of a customized loss functions on the improved autoencoder ensures that the learned representations are resilient to the curse of dimensionality i.e. it encourages the autoencoder to reconstruct the most relevant features accurately.

The introduction of the sparsity constraints in the new autoencoder's architecture reduces the impact of sparse, irrelevant and noisy features from the high dimensional datasets.

Through this, the new *k*-hyperparameter tuning technique is able to effectively handle the cluster analysis challenges inherent in the high dimensional spaces that make it challenging to tune for the *k*-hyperparameter i.e. the curse of dimensionality and the data sparsity. Moreover, the new technique is able to solve the smooth elbow limitation with the canonical elbow method. This new technique demonstrates superior performance and flexibility in a variety of high dimensional datasets, as compared to the existing techniques. It is important to note that the new algorithm is generalizable. This generalizability has been as a result of understanding how different factors interact with each other, providing insights into building more generalized *k*-hyperparameter tuning technique that perform well across a variety of high dimensional datasets compromising of different data types and dimensionality levels. The various aspects of the new technique have been published in open access journals as indicated in the appendices under the publications sections *ii*, *iv* and *v*.

6.3.2 Theoretical Contribution

Within the domain of advancing the existing theories and theoretical frameworks in the area of *k*-hyperparameter tuning in high dimensional space clustering, this research study is a novel advancement in four theoretical contributions. Firstly, the systematic review of the existing *k*-hyperparameter tuning techniques, through literature review analysis, supplements the existing literature in this problem domain. Researchers focusing on the *k*-hyperparameter tuning problems in high dimensional space clustering can use this literature as the basis for their proposed work. This review has been published in an open access journal as a systematic review article entitled the "Trends and Advances on the K-hyperparameter Tuning Techniques in High-Dimensional Space Clustering".

Secondly, the experimental results supplement the existing literature on the empirical analysis of different *k*-hyperparameter tuning algorithms in a variety of high dimensional datasets and dimensionality reduction methods. Such experimental results provide a foundation upon which further empirical study on *k*-hyperparameter tuning techniques can be conducted. Such further empirical study has been suggested in section 6.5 on the recommendations for further research. Thirdly, this research study supplements the existing literature on the internal validation indexes used for evaluating the quality of clusters in *k*-means based algorithms. Combining the four commonly used internal validation indexes in order to create an improved ensemble internal validation index is an advancement of such literature. This knowledge is helpful in choosing the most effective evaluation metric for any clustering algorithm. Lastly, the recommendations for future research in this study provide an avenue for scholars doing further research in the area of *k*-hyperparameter tuning and optimization in high-dimensional space clustering.

6.4 Limitations

This section highlights the main limitations encountered during the research study, along with the measures taken to address them and ensure success. Firstly, during the systematic review of the existing *k*-hyperparameter tuning techniques in high-dimensional space clustering, it is noted that there is no standardized set of high-dimensional input datasets or evaluation metrics across the various techniques. This lack of standardization complicates the process of identifying the best-performing techniques based solely on theoretical analysis. To mitigate this limitation, a consistent evaluation metric is adopted during the pilot study to validate or challenge the results of the theoretical analysis. This approach is also applied during the empirical analysis in the main experiments to ensure fairness in the evaluation process.

Secondly, during the empirical algorithmic exercise, different *k*-hyperparameter tuning techniques converges over varying numbers of runs and iterations, making it difficult to report uniform stopping criteria values across all techniques. Some techniques require more iterations than others within a single run before achieving convergence, while others need more runs before stability is reached. To address this limitation, the stopping criteria for both runs and iterations are based on the maximum values exhibited by the techniques that require the highest thresholds to converge. Additionally, an allowance is included to enhance the reliability of the reported results. Consequently, the maximum iterations for all techniques are uniformly set to 100, even in cases where some techniques converge in fewer iterations, such as 34 for the *k*HT2 technique. Similarly, the maximum number of runs for all techniques is set to a uniform value of 15, despite some techniques achieving stability in fewer runs, such as 6 in the *k*HT1 technique. This approach negatively impacts computational cost and efficiency throughout the empirical algorithmic exercise. Nevertheless, the measures implemented to address these limitations are crucial for ensuring reproducibility and reliability for this research study.

6.5 Recommendations for further research

Although the new *k*-hyperparameter tuning technique demonstrates a superior performance and flexibility across a number of datasets, its run times as compared to the others are relatively higher. For this reason, the researcher suggests that the future research directions can focus on the areas of optimization for this particular technique so as to enhance its time efficiency without compromising on its quality of clustering. Other research directions can include an empirical investigation on how the adoption of different hybrid sets of the best two performing dimensionality reduction methods in the unsupervised transfer strategy on the improved autoencoder can affect the clustering performance.

This can further improve the effectiveness of managing the inherent challenges in high dimensional spaces through a more effective feature engineering strategy, and consequently lead to a more effective *k*-hyperparameter tuning technique in high dimensional space clustering.

REFERENCES

Ada, A. F., Hornick, B., & Chamberlain, R. (2013). *The Healthy Beef Cookbook: Steaks, Salads, Stir-fry, and More--Over 130 Luscious Lean Beef Recipes for Every Occasion*. HarperCollins.

Agarwal, M., Jaiswal, R., & Pal, A. (2015). *k*-means++ under Approximation Stability. *Theoretical Computer Science*, 588, 37-51.

Ahmed, M., Seraj, R., & Islam, S. M. S. (2020). The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics*, *9*(8), 1295.

Ahmed, N. (2015). Recent review on image clustering. *IET Image Processing*, *9*(11), 1020-1032.

Al-Daoud, M. D. B., & Roberts, S. A. (1996). New methods for the initialisation of clusters. *Pattern Recognition Letters*, *17*(5), 451-455.

Aljalbout, E., Golkov, V., Siddiqui, Y., Strobel, M., & Cremers, D. (2018). Clustering with deep learning: Taxonomy and new methods. *arXiv preprint arXiv*:1801.07648.

Allaoui, M., Kherfi, M. L., & Cheriet, A. (2020, June). Considerably improving clustering algorithms using UMAP dimensionality reduction technique: A comparative study. *In International conference on image and signal processing* (pp. 317-325). Cham: Springer International Publishing.

Alsabti, K., Ranka, S., & Singh, V. (1997). An efficient *k*-means clustering algorithm. *Pattern Recognition*, *30*(2), 187-196.

Altaf, M., Uzair, M., Naeem, M., Ahmad, A., Badshah, S., Shah, J. A., & Anjum, A. (2019). Automatic and efficient fault detection in rotating machinery using sound signals. *Acoustics Australia*, *47*, 125-139.

Amit, G., Gavriely, N., & Intrator, N. (2009). Cluster analysis and classification of heart sounds. *Biomedical Signal Processing and Control*, *4*(1), 26-36.

Arora, R. (2012). Comparative analysis of classification algorithms on different datasets using WEKA. *International Journal of Computer Applications*, *54*(13).

Ayesha, S., Hanif, M. K., & Talib, R. (2020). Overview and comparative study of dimensionality reduction techniques for high dimensional data. *Information Fusion*, 59, 44-58.

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2015). Segnet: a deep convolutional encoder-decoder architecture for image segmentation. *arXiv, 1511*, 1-14.

Bai, L., Liang, J., Sui, C., & Dang, C. (2013). Fast global *k*-means clustering based on local geometrical information. *Information Sciences*, *245,* 168–180.

Bala, C., Basu, T., & Dasgupta, A. (2015, August). Automatic detection of k with suitable seed values for classic k-means algorithm using DE. *In 2015 International Conference on Advances in Computing, Communications and Informatics* (ICACCI) (pp. 759-765). IEEE.

Bao, L. M., & Huang, G. (2017). A dynamic clustering algorithm of *k*-means based on multi-branches tree for *k*-values. *Computer Technology and Development*, *27*(6), 41-45.

Bao, H., Guo, S., Mo, J., Zhao, Z., Wang, Z., Chen, Z., & Liang, J. (2023). An analysis method for residential electricity consumption behavior based on UMAP-CRITIC feature optimization and SSA-assisted clustering. *Energy Reports*, *9*, 245-254.

Banerjee, P., Chattopadhyay, T., & Chattopadhyay, A. K. (2023). Comparison among different Clustering and Classification Techniques: Astronomical data-dependent study. *New Astronomy*, *100*, 101973.

Beaulieu-Jones, B. K., & Greene, C. S. (2016). Semi-supervised learning of the electronic health record for phenotype stratification. *Journal of Biomedical Informatics*, *64*, 168-178.

Becht, E., McInnes, L., Healy, J., Dutertre, C. A., Kwok, I. W., Ng, L. G., & Newell, E. W. (2019). Dimensionality reduction for visualizing single-cell data using UMAP. *Nature biotechnology*, *37*(1), 38-44.

Benbasat, I., & Zmud, R. W. (1999). Empirical research in information systems: *The practice of relevance. MIS Quarterly*, *23*(1), 3-16.

Berisha, V., Krantsevich, C., Hahn, P. R., Hahn, S., Dasarathy, G., Turaga, P., & Liss, J. (2021). Digital medicine and the curse of dimensionality. *NPJ digital medicine*, *4*(1), 153.

Bholowalia, P., & Kumar, A. (2014). EBK-means: A clustering technique based on elbow method and *k*-means in WSN. *International Journal of Computer Applications*, *105*(9).

Bian, C., & Qian, C. (2022). Running time analysis of the non-dominated sorting genetic algorithm II (NSGA-II) using binary or stochastic tournament selection. *arXiv preprint arXiv*:2203.11550.

Bickel, P., Diggle, P., Fienberg, S., Gather, U., Olkin, I., & Zeger, S. (2009). *Springer series in statistics.*

Bishnoi, A., & Todwal, V. (2018). *K*-Means Clustering Using the Optimal Cluster Finding Technique. *International Journal of Research in Engineering, Science and Management*

Blömer, J., Lammersen, C., Schmidt, M., & Sohler, C. (2016). Theoretical analysis of the *k*-means algorithm–a survey. *In C. Lammersen & M. Schmidt (Eds.), Algorithm Engineering* (pp. 81-116). Springer, Cham.

Bohn, B., Garcke, J., & Griebel, M. (2016). A sparse grid based method for generative dimensionality reduction of high-dimensional data. *Journal of Computational Physics*, 309, 1-17.

Bolívar, A., García, V., Florencia, R., Alejo, R., Rivera, G., & Sánchez-Solís, J. P. (2022, June). A preliminary study of smote on imbalanced big datasets when dealing with sparse and dense high dimensionality. In *Mexican Conference on Pattern Recognition* (pp. 46-55). Cham: Springer International Publishing.

Bonidia, R. P., Sampaio, L. D., Domingues, D. S., Paschoal, A. R., Lopes, F. M., de Carvalho, A. C., & Sanches, D. S. (2021). Feature extraction approaches for biological sequences: a comparative study of mathematical features. *Briefings in Bioinformatics*, *22*(5), bbab011.

Brock, G., Pihur, V., Datta, S., & Datta, S. (2011). clValid, an R package for cluster validation. *Journal of Statistical Software, 42*(11), 1-18.

Brodinová, Š., Filzmoser, P., Ortner, T., Breiteneder, C., & Rohm, M. (2019). Robust and sparse *k*-means clustering for high-dimensional data. *Advances in Data Analysis and Classification, 13*, 905-932. https://doi.org/10.1007/s11634-018-0349-z.

Bueso, D., Piles, M., & Camps-Valls, G. (2020). Nonlinear PCA for spatio-temporal analysis of Earth observation data. *IEEE Transactions on Geoscience and Remote Sensing*, *58*(8), 5752-5763.

Buresh, R. J., Austin, E. R., & Craswell, E. T. (1982). Analytical methods in 15 N research. *Fertilizer research, 3*, 37-62.

Calli, B., Singh, A., Bruce, J., Walsman, A., Konolige, K., Srinivasa, S., & Dollar, A. M. (2017). Yale-CMU-Berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, *36*(3), 261-268.

Casas, M., Moreto, M., Alvarez, L., Castillo, E., Chasapis, D., Hayes, T., & Valero, M. (2015). Runtime-aware architectures. In Euro-Par 2015: *Parallel Processing: 21st International Conference on Parallel and Distributed Computing,* Vienna, Austria, August 24-28, 2015, Proceedings 21 (pp. 16-27). Springer Berlin Heidelberg.

Celebi, M. E. (Ed.). (2014). Partitional clustering algorithms. *Springer*.

Celebi, M. E., Kingravi, H. A., & Vela, P. A. (2013). A comparative study of efficient initialization methods for the *k*-means clustering algorithm. *Expert Systems with Applications*, *40*(1), 200-210.

Chakraborty, S., & Das, S. (2020). Detecting meaningful clusters from high-dimensional data: A strongly consistent sparse center-based clustering approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *44*(6), 2894-2908.

Chaudhari, G., Jiang, X., Fakhry, A., Han, A., Xiao, J., Shen, S., & Khanzada, A. (2020). Virufy: Global applicability of crowdsourced and clinical datasets for AI detection of COVID-19 from cough. *arXiv preprint arXiv*:2011.13320.

Chin, T. J., Schindler, K., & Suter, D. (2006, April). Incremental kernel SVD for face recognition with image sets. In *7th International Conference on Automatic Face and Gesture Recognition (FGR06)* (pp. 461-466). IEEE.

Chithra, P. L. (2017). Premeditated initial points for K-Means Clustering. *IJCSIS, 15*(9).

Chowdhury, N. K., Kabir, M. A., Rahman, M. M., & Islam, S. M. S. (2022). Machine learning for detecting COVID-19 from cough sounds: An ensemble-based MCDM method. *Computers in Biology and Medicine, 145*, 105405.

Clarke, B. S. (2019). Discussion of 'Prior-based Bayesian information criterion (PBIC)'. *Statistical Theory and Related Fields*, *3*(1), 26-29.

Cln, L. I. S., & Iro, L. (2013). Data collection techniques a guide for researchers in humanities and education. *International Research Journal of Computer Science and Information Systems* (IRJCSIS), *2*(3), 40-44.

Creswell, J. W., & Poth, C. N. (2016). *Qualitative inquiry and research design: Choosing among five approaches*. Sage publications.

Cooper, H. E., Camic, P. M., Long, D. L., Panter, A. T., Rindskopf, D. E., & Sher, K. J. (2012). APA handbook of research methods in psychology, Vol 2: Research designs: *Quantitative, qualitative, neuropsychological, and biological* (pp. x-701). American Psychological Association.

Dasgupta, S. (2003). *How fast is k-means? In Learning Theory and Kernel Machines* (pp. 735-735). Springer, Berlin, Heidelberg.

Dash, B., Mishra, D., Rath, A., & Acharya, M. (2010). A hybridized K-means clustering approach for high dimensional dataset. *International Journal of Engineering, Science and Technology*, *2*(2), 59-66.

Danasingh, A. A. G. S., Subramanian, A. A. B., & Epiphany, J. L. (2020). Identifying redundant features using unsupervised learning for high-dimensional data. *SN Applied Sciences*, *2*, 1-10.

De Lázaro, J. M. B., Moreno, A. P., Santiago, O. L., & da Silva Neto, A. J. (2015). Optimizing kernel methods to reduce dimensionality in fault diagnosis of industrial systems. *Computers & Industrial Engineering*, *87*, 140-149.

Deborah, L. J., Baskaran, R., & Kannan, A. (2010). A survey on internal validity measure for cluster validation. *International Journal of Computer Science & Engineering Survey*, *1*(2), 85-102.

DeCoster, J. (2006). Testing group differences using t-tests, ANOVA, and nonparametric measures. *Journal of Experimental Education*, *74*(3), 267-286. Accessed November 30, 2010. doi:202006-0.

Dey, S., Das, S., & Mallipeddi, R. (2020, July). The Sparse MinMax k-Means Algorithm for High-Dimensional Clustering. *In IJCAI* (pp. 2103-2110).

Dilokthanakul, N., Mediano, P. A., Garnelo, M., Lee, M. C., Salimbeni, H., Arulkumaran, K., & Shanahan, M. (2016). Deep unsupervised clustering with Gaussian mixture variational autoencoders. *arXiv preprint arXiv*:1611.02648.

Draganov, A. (2021). *Towards a common dimensionality reduction approach; comparing PCA, TSNE, AND UMAP through a cohesive framework* (Doctoral dissertation, George Mason University).

Du, G., Li, X., Zhang, L., Liu, L., & Zhao, C. (2021). Novel automated K-means++ algorithm for financial data sets. *Mathematical Problems in Engineering*, *2021*, 1-12.

Dubey, A., & Choubey, A. P. D. A. (2018). A Systematic Review on K-Means Clustering Techniques. *International Journal of Scientific Research Engineering & Technology (IJSRET)*, ISSN, 2278-0882.

Dwivedi, S., & Bhaiya, L. K. P. (2019). A systematic review on K-means clustering techniques. *International Journal of Scientific Research Engineering Trends*, *5*(3), 750-752.

El-Mandouh, A. M., Mahmoud, H. A., Abd-Elmegid, L. A., & Haggag, M. H. (2019). Optimized *K*-means clustering model based on gap statistic. *International Journal of Advanced Computer Science*, *10*(1), 183-188.

Erekat, D. N. A. (2021). *Spatio-Temporal Assessment of Pain Intensity through Facial Transformation-Based Representation Learning* (Doctoral dissertation, Bilkent Universitesi (Turkey).

Etikan, I., Musa, S. A., & Alkassim, R. S. (2016). Comparison of convenience sampling and purposive sampling. *American Journal of Theoretical and Applied Statistics*, *5*(1), 1-4.

Fard, M. M., Thonet, T., & Gaussier, E. (2020). Deep k-means: Jointly clustering with *k*-means and learning representations. *Pattern Recognition Letters*, 138, 185-192.

Feldman, D., Schmidt, M., & Sohler, C. (2020). Turning big data into tiny data: Constant-size coresets for *k*-means, PCA, and projective clustering. *SIAM Journal on Computing*, *49*(3), 601-657.

Finch, W. H. (2020). Using fit statistic differences to determine the optimal number of factors to retain in an exploratory factor analysis. *Educational and psychological measurement*, *80*(2), 217-241.

Fränti, P., & Sieranoja, S. (2018). *K*-means properties on six clustering benchmark datasets. *Applied Intelligence*, *48*(12), 4743-4759.

Friedrich, T., Kötzing, T., Quinzan, F., & Sutton, A. M. (2017, January). Resampling vs recombination: A statistical run time estimation. In *Proceedings of the 14th ACM/SIGEVO Conference on Foundations of Genetic Algorithms* (pp. 25-35).

Fournier, Q., & Aloise, D. (2019, June). Empirical comparison between autoencoders and traditional dimensionality reduction methods. *In 2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)* (pp. 211-214). IEEE.

Fürnkranz, J. (1998). A study using n-gram features for text categorization. *Austrian Research Institute for Artifical Intelligence*, *3*(1998), 1-10.

Gaddis, G. (2015). Advanced biostatistics: Chi-square, ANOVA, regression, and multiple regression. In J. C. Adams & E. A. DeBiasio (Eds.), *Doing Research in Emergency and Acute Care: Making Order Out of Chaos* (pp. 213-222).

Gare, S., Chel, S., Kuruba, M., Jana, S., & Giri, L. (2021, July). Dimension reduction and clustering of single cell calcium spiking: comparison of t-SNE and UMAP. In *2021 National Conference on Communications (NCC)* (pp. 1-6). IEEE.

Geng, T., Li, A., Shi, R., Wu, C., Wang, T., Li, Y., & Herbordt, M. C. (2020, October). AWB-GCN: A graph convolutional network accelerator with runtime workload rebalancing. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* (pp. 922-936). IEEE.

Ghodsi, A. (2006). Dimensionality reduction: *A short tutorial. Department of Statistics and Actuarial Science,* University of Waterloo, Ontario, Canada, 37(38), 2006.

Ghojogh, B., Ghodsi, A., Karray, F., & Crowley, M. (2021). Uniform Manifold approximation and projection (UMAP) and its variants: *tutorial and survey*. arXiv preprint arXiv:2109.02508.

Gikera, R., Mambo, S., & Mwaura, J. (2020, September). Optimized K-Means clustering algorithm using an intelligent stable-plastic variational autoencoder with self-intrinsic cluster validation mechanism. *In Proceedings of the 2nd International Conference on Intelligent and Innovative Computing Applications* (pp. 1-11).

Gikera, R., Mwaura, J., Mambo, S., & Muuro, E. (2023). Trends and Advances on the *K*-hyperparameter Tuning Techniques in High-dimensional Space Clustering. *Indonesian Journal of Artificial Intelligence and Data Mining*.

Gikera, R., Mwaura, J., Mambo, S., & Muuro,E. (2023). K-hyperparameter Tuning in High-dimensional Space Clustering: Solving Smooth Elbow Challenges using an Ensemble Based Technique of a Self-adapting Autoencoder and Internal Validation Indexes. *Journal of Artificial Intelligence*.

Greenland, S., Senn, S. J., Rothman, K. J., Carlin, J. B., Poole, C., Goodman, S. N., & Altman, D. G. (2016). Statistical tests, P values, confidence intervals, and power: a guide to misinterpretations. *European journal of epidemiology*, 31, 337-350.

Guérin, J., Gibaru, O., Thiery, S., & Nyiri, E. (2017). *Cnn features are also great at unsupervised classification*. arXiv preprint arXiv:1707.01700.

Gupta, M. K., & Chandra, P. (2019). *P k-means: k-means using partition-based cluster initialization method*. Available at SSRN 3462549.

Hämäläinen, J., Kärkkäinen, T., & Rossi, T. (2020). Scalable initialization methods for large-scale clustering. *arXiv preprint arXiv*:2007.11937.

Han, J., & Ge, Z. (2020). Effect of dimensionality reduction on stock selection with cluster analysis in different market situations. *Expert Systems with Applications*, *147*, 113226.

Haraty, R. A., Dimishkieh, M., & Masud, M. (2015). An enhanced *k*-means clustering algorithm for pattern discovery in healthcare data. *International Journal of Distributed Sensor Networks*, *11*(6), 615740.

Hartigan, J. A. (1985). Statistical theory in clustering. *Journal of classification*, *2*, 63-76.

Hasan, B. M. S., & Abdulazeez, A. M. (2021). A review of Principal Component Analysis algorithm for dimensionality reduction. *Journal of Soft Computing and Data Mining*, *2*(1), 20-30.

Herek, G. M. (2011). Developing a theoretical framework and rationale for a research proposal. How to write a successful research grant application: *A guide for social and behavioral scientists*, 137-145.

Hess, S., & Duivesteijn, W. (2019). k is the Magic Number--Inferring the Number of Clusters Through Nonparametric Concentration Inequalities. *arXiv preprint arXiv*:1907.02343. Hewlett-Packard Labs Technical Report HPL-1999-124, 55.

Hinds, D. (2002). Research instruments. In *The researcher's toolkit* (pp. 57-58). Routledge.

Homburg, H., Mierswa, I., Möller, B., Morik, K., & Wurst, M. (2005, September). A Benchmark Dataset for Audio Classification and Clustering. *In ISMIR* (Vol. 2005, pp. 528-31).

Hozumi, Y., Wang, R., Yin, C., & Wei, G. W. (2021). UMAP-assisted *K*-means clustering of large-scale SARS-CoV-2 mutation datasets. *Computers in Biology and Medicine*, 131, 104264.

Huang, T., Zhang, Z., & Zhang, J. (2019, September). FiBiNET: combining feature importance and bilinear feature interaction for click-through rate prediction. In *Proceedings of the 13th ACM Conference on Recommender Systems* (pp. 169-177).

Huang, X., Wu, L., & Ye, Y. (2019). A review on dimensionality reduction techniques. *International Journal of Pattern Recognition and Artificial Intelligence*, *33*(10), 1950017.

Hussain, S. F., & Haris, M. (2019). A *k*-means based co-clustering (kCC) algorithm for sparse, high dimensional data. *Expert Systems with Applications*, 118, 20-34.

Ikotun, A. M., Ezugwu, A. E., Abualigah, L., Abuhaija, B., & Heming, J. (2022). K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Information Sciences*.

Idakwo, G., Thangapandian, S., Luttrell, J., Li, Y., Wang, N., Zhou, Z., ... & Gong, P. (2020). Structure–activity relationship-based chemical classification of highly imbalanced Tox21 datasets. *Journal of cheminformatics*, *12*(1), 1-19.

Ismkhan, H. (2018). Ik-means−+: An iterative clustering algorithm based on an enhanced version of the k-means. *Pattern Recognition*, 79, 402-413.

Jamal, A., Handayani, A., Septiandri, A. A., Ripmiatin, E., & Effendi, Y. (2018). Dimensionality reduction using pca and k-means clustering for breast cancer prediction. *Lontar Komput. J. Ilm. Teknol. Inf*, *9*(3), 192-201.

Järvinen, P. (2008, September). Mapping research questions to research methods. *In IFIP World Computer Congress*, TC 8 (pp. 29-41). Boston, MA: Springer US.

Jia, R. Y., & Song, J. L. (2016). K-means Optimal Clustering Number Determination Method Based on Clustering Optimization. *Microelectronics & Computer*, (5),13.

Jia, W., Yang, M., & Wang, S. H. (2017). Three-category classification of magnetic resonance hearing loss images based on deep autoencoder. *Journal of medical systems*, 41, 1-11.

Jia, W., Sun, M., Lian, J., & Hou, S. (2022). Feature dimensionality reduction*: a review. Complex & Intelligent Systems*, *8*(3), 2663-2693.

Jung, S. H., Kim, K. J., Lim, E. C., & Sim, C. B. (2017). A novel on automatic K value for efficiency improvement of K-means clustering. *In Advanced Multimedia and Ubiquitous Engineering* (pp. 181-186). Springer, Singapore.

Kampman, I., & Elomaa, T. (2018). Hierarchical clustering of high-dimensional data without global dimensionality reduction. In *Foundations of Intelligent Systems: 24th International Symposium, ISMIS 2018, Limassol, Cyprus, October 29–31, 2018, Proceedings 24* (pp. 236-246). Springer International Publishing.

Kant, S., & Ansari, I. A. (2016). An improved K-means clustering with Atkinson index to classify liver patient dataset. *International Journal of System Assurance Engineering and Management*, *7*(1), 222-228.

Kapoor, A., & Singhal, A. (2017, February). A comparative study of K-Means, K-Means++ and Fuzzy C-Means clustering algorithms. *In 2017 3rd international conference on computational intelligence & communication technology (CICT)* (pp. 1-6). IEEE.

Kapoor, A., & Abhishek S., (2017). "A comparative study of K-Means, K-Means++ and Fuzzy C-Means clustering algorithms." *2017 3rd international conference on computational intelligence & communication technology (CICT)*. IEEE.

Kaur, S. P. (2013). Variables in research. *Indian Journal of Research and Reports in Medical Sciences*, *3*(4), 36-38.

Khan, K. S., Kunz, R., Kleijnen, J., & Antes, G. (2003). Five steps to conducting a systematic review. *Journal of the royal society of medicine*, *96*(3), 118-121.

Khan, Z., Ni, J., Fan, X., & Shi, P. (2017). An improved *k*-means clustering algorithm based on an adaptive initial parameter estimation procedure for image segmentation. *Int. J. Innov. Comput. Inf. Control*, *13*(5), 1509-1525.

Khanmohammadi, S., Adibeig, N., & Shanehbandy, S. (2017). An improved overlapping *k*-means clustering method for medical applications. *Expert Systems with Applications*, 67, 12–18.

Kodinariya, T. M., & Makwana, P. R. (2013). Review on determining number of Cluster in *K*-Means Clustering. *International Journal*, *1*(6), 90-95.

Kongsin, T., & Klongboonjit, S. (2020, April). Machine component clustering with mixing technique of DSM, jaccard distance coefficient and k-means algorithm. *In 2020 IEEE 7th International Conference on Industrial Engineering and Applications (ICIEA)* (pp. 251-255). IEEE.

Kuehne, H., Gall, J., & Serre, T. (2015). Cooking in the kitchen: Recognizing and Segmenting Human Activities in Videos. *arXiv preprint arXiv*:1508.06073.

Kumar, A., Nayak, S., & Chandra, N. (2019). Empirical analysis of supervised machine learning techniques for cyberbullying detection. In *International Conference on Innovative Computing and Communications: Proceedings of ICICC 2018, Volume 2* (pp. 223-230). Springer Singapore.

Kumar, K. M., & Reddy, A. R. M. (2017). An efficient *k*-means clustering filtering algorithm using density based initial cluster centers. *Information Sciences*, 418, 286-301.

Kung, S. Y. (2014). *Kernel methods and machine learning*. Cambridge University Press.

Laguarta, J., Hueto, F., & Subirana, B. (2020). COVID-19 artificial intelligence diagnosis using only cough recordings. *IEEE Open Journal of Engineering in Medicine and Biology*, 1, 275-281.

Li, F., Qiao, H., & Zhang, B. (2018). Discriminatively boosted image clustering with fully convolutional auto-encoders. *Pattern Recognition*, 83, 161-173.

Li, S. (2015). k-groups: *A Generalization of k-means by Energy Distance* (Doctoral dissertation, Bowling Green State University).

Liu, L., Tang, X., Chen, C. P., Cai, L., & Lan, R. (2022). Superpixel-guided locality quaternion representation for color face hallucination. *Information Sciences*, *609*, 565-577.

Liu, X., Zhou, S., Liu, L., Tang, C., Wang, S., Liu, J., & Zhang, Y. (2021). Localized simple multiple kernel k-means. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 9293-9301).

Liu, Y., Li, Z., Xiong, H., Gao, X., Wu, J., & Wu, S. (2013). Understanding and enhancement of internal clustering validation measures. *IEEE Transactions on Cybernetics*, *43*(3), 982-994.

Lu, W. (2019). Improved *K*-means clustering algorithm for big data mining under Hadoop parallel framework. *Journal of Grid Computing*, *18*(1), 239-250.

Lu, S., Yu, H., Wang, X., Zhang, Q., Li, F., Liu, Z., & Ning, F. (2018, July). Clustering method of raw meal composition based on PCA and Kmeans. *In 2018 37th Chinese control conference (CCC)* (pp. 9007-9010). IEEE.

Ma, Y., & Zhu, L. (2013). A review on dimension reduction. *International Statistical Review*, *81*(1), 134-150.

Mallick, A. K., & Mukhopadhyay, S. (2022). Video retrieval framework based on color co-occurrence feature of adaptive low rank extracted keyframes and graph pattern matching. *Information Processing & Management*, *59*(2), 102870.

Mamat, A. R., Mohamed, F. S., Mohamed, M. A., Rawi, N. M., & Awang, M. I. (2018). Silhouette index for determining optimal k-means clustering on images in different color models. *International Journal of Engineering and Technology*, *7*(2.14), 105-1109.

Mayer, B., Steinhauer, N., Preim, B., & Meuschke, M. (2023). A Characterization of Interactive Visual Data Stories With a Spatio-Temporal Context. In *Computer Graphics Forum* (p. e14922).

McGee, S. (2010). Etiology and diagnosis of systolic murmurs in adults. *The American journal of medicine*, *123*(10), 913-921.

McGeoch, C. C. (2012). *A guide to experimental algorithmics*. Cambridge University Press.

McHugh, M. L. (2013). The chi-square test of independence. *Biochemia medica*, *23*(2), 143-149

McInnes, L., Healy, J., & Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2021). A survey on bias and fairness in machine learning. *ACM computing surveys (CSUR)*, *54*(6), 1-35.

Meila, M., & Heckerman, D. (2013). An experimental comparison of several clustering and initialization methods. *arXiv preprint arXiv*:1301.7401.

Misra, B. B. (2020). Data normalization strategies in metabolomics: Current challenges, approaches, and tools. *European Journal of Mass Spectrometry*, *26*(3), 165-174.

Mohammed, N. N. (2022, October). Improved Regularized Multi-class Logistic Regression for Gene Classification with Optimal Kernel PCA and HC Algorithm. *In Worldwide Congress on "Genetics, Geriatrics and Neurodegenerative Diseases Research"* (pp. 273-279). Cham: Springer International Publishing.

Mohanty, V., Naidu, P. A., & Nayak, M. (2014). Comparative study of PCA and ICA in the field of Data Reduction. *International journal of engineering sciences & research technology*

Moon, K. R., van Dijk, D., Wang, Z., Gigante, S., Burkhardt, D. B., Chen, W. S., ... & Krishnaswamy, S. (2019). Visualizing structure and transitions in high-dimensional biological data. *Nature biotechnology*, *37*(12), 1482-1492.

Moslemi, A., Bidar, M., & Ahmadian, A. (2023). Subspace learning using structure learning and non-convex regularization: Hybrid technique with mushroom reproduction optimization in gene selection. *Computers in Biology and Medicine*, 164, 107309.

Murugesan, V. P., & Murugesan, P. (2020). A new initialization and performance measure for the rough k-means clustering. *Soft Computing*, 1-15.

Musa, K. I., Mansor, W. N. A. W., & Hanis, T. M. (2023). *Data Analysis in Medicine and Health Using R*. CRC Press.

Musco, C. N. (2015). *Dimensionality reduction for k-means clustering* (Doctoral dissertation, Massachusetts Institute of Technology).

Naeem, S., & Wumaier, A. (2018). Study and implementing K-mean clustering algorithm on English text and techniques to find the optimal value of K. *International Journal of Computer Applications*, *182*(31), 7-14.

Nainggolan, R., Perangin-angin, R., Simarmata, E., & Tarigan, A. F. (2019, November). Improved the performance of the K-means cluster using the sum of squared error (SSE) optimized by using the Elbow method. In *Journal of Physics: Conference Series* (Vol. 1361, No. 1, p. 012015). IOP Publishing.

Naseriparsa, M., & Kashani, M. M. R. (2014). Combination of PCA with SMOTE resampling to boost the prediction rate in lung cancer dataset. *arXiv preprint arXiv:*1403.1949.

Nazeer, K. A., & Sebastian, M. P. (2009, July). Improving the Accuracy and Efficiency of the k-means Clustering Algorithm. *In Proceedings of the world congress on engineering* (Vol. 1, pp. 1-3). London, UK: Association of Engineers London.

Nguyen, C. D., & Duong, T. H. (2018). K-means–A fast and efficient K-means algorithm. *International Journal of Intelligent Information and Database Systems*, *11*(1), 27-45.

Nguyen, L. H., & Holmes, S. (2019). Ten quick tips for effective dimensionality reduction. *PLoS computational biology*, *15*(6), e1006907.

Nunamaker Jr, J. F., Chen, M., & Purdin, T. D. (1990). Systems development in information systems research. *Journal of Management Information Systems*, *7*(3), 89-106.

Okoye, K., & Hosseini, S. (2024). Mann–Whitney U Test and Kruskal–Wallis H Test Statistics in R. In R Programming: *Statistical Data Analysis in Research* (pp. 225-246). Singapore: Springer Nature Singapore.

Onan, A., Korukoglu, S., & Bulut, H. (2016). LDA-based topic modelling in text sentiment classification: An empirical analysis. *Int. J. Comput. Linguistics Appl.*, *7*(1), 101-119.

Onumanyi, A. J., Molokomme, D. N., Isaac, S. J., & Abu-Mahfouz, A. M. (2022). AutoElbow: An automatic elbow detection method for estimating the number of clusters in a dataset. *Applied Sciences*, *12*(15), 7515.

Orkphol, K., & Yang, W. (2019). Sentiment analysis on microblogging with K-means clustering and artificial bee colony. *International Journal of Computational Intelligence and Applications*, *18*(03), 1950017.

Ortega, J. P., Ortega, N. N. A., Ruiz-Vanoye, J. A., Sánchez, S. S., Lelis, J. M. R., & Rebollar, A. M. (2018). A-means: improving the cluster assignment phase of *k*-means for Big Data. *International Journal of Combinatorial Optimization Problems and Informatics*, *9*(2), 3-10.

Pandey, S., & Kumar Tiwari, L. (2018). Review of Existing Methods in *K*-means Clustering Algorithm.

Park, G. Y., Kim, H., Jeong, H. W., & Youn, H. Y. (2013, March). A novel cluster head selection method based on K-means algorithm for energy efficient wireless sensor network. *In 2013 27th international conference on advanced information networking and applications workshops* (pp. 910-915). IEEE.

Park, H. M. (2009). Comparing group means: t-tests and one-way ANOVA using Stata, SAS, R, and SPSS.

Park, K., Hong, J. S., & Kim, W. (2020). A methodology combining cosine similarity with classifier for text classification. *Applied Artificial Intelligence*, 34(5), 396-411.

Patil, C., & Baidari, I. (2019). Estimating the Optimal Number of Clusters k in a Dataset Using Data Depth. *Data Science and Engineering*, *4*(2), 132-140.

Patil, P., & Karthikeyan, A. (2020). A survey on *k*-means clustering for analyzing variation in data. *In Inventive Communication and Computational Technologies* (pp. 317-323). Springer, Singapore.

Prefect, F., & Prosser, P. (2014). Empirical Algorithmics: draw your own conclusions. *arXiv preprint arXiv*:1412.3333.

Probst, P., Boulesteix, A. L., & Bischl, B. (2019). Tunability: Importance of hyperparameters of machine learning algorithms. *The Journal of Machine Learning Research*, 20(1), 1934-1965.

Qi, J., Yu, Y., Wang, L., & Liu, J. (2016, October). K*-means: An effective and efficient k-means clustering algorithm. *In 2016 IEEE international conferences on big data and cloud computing (BDCloud), social computing and networking (SocialCom), sustainable computing and communications* (SustainCom) (BDCloud-SocialCom-SustainCom) (pp. 242-249). IEEE.

Qi, J., Yu, Y., Wang, L., Liu, J., & Wang, Y. (2017). An effective and efficient hierarchical *K*-means clustering algorithm. *International Journal of Distributed Sensor Networks*, *13*(8), 1550147717728627.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2017). Machine learning of linear differential equations using Gaussian processes. *Journal of Computational Physics*, 348, 683-693.

Rathod, R. R., & Garg, R. D. (2017). Design of electricity tariff plans using gap statistic for *K*-means clustering based on consumers monthly electricity consumption data. *International Journal of Energy Sector Management*.

Reddy, G. T., Reddy, M. P. K., Lakshmanna, K., Kaluri, R., Rajput, D. S., Srivastava, G., & Baker, T. (2020). Analysis of dimensionality reduction techniques on big data. *IEEE Access*, 8, 54776-54788.

Rehman, A., Khan, A., Ali, M. A., Khan, M. U., Khan, S. U., & Ali, L. (2020, June). Performance analysis of pca, sparse pca, kernel pca and incremental pca algorithms for heart failure prediction. *In 2020 International Conference on Electrical, Communication, and Computer Engineering* (ICECCE) (pp. 1-5). IEEE.

Rendón, E., Abundez, I., Arizmendi, A., & Quiroz, E. M. (2011). Internal versus external cluster validation indexes. *International Journal of Computers and Communications*, *5*(1), 27-34.

Rezaee, M. J., Eshkevari, M., Saberi, M., & Hussain, O. (2020). GBK-means clustering algorithm: An improvement to the *K*-means algorithm based on the bargaining game. *Knowledge-Based Systems*, 213, 106672.

Robert, V., Vasseur, Y., & Brault, V. (2021). Comparing high-dimensional partitions with the co-clustering adjusted rand index. *Journal of Classification*, 38, 158-186.

Robinson, J. R. (2019). Facial Image Characterization and Reconstruction Using Singular Value Decomposition.

Rodriguez, M. Z., Comin, C. H., Casanova, D., Bruno, O. M., Amancio, D. R., Costa, L. D. F., & Rodrigues, F. A. (2019). Clustering algorithms: A comparative approach. PloS one, 14(1), e0210236.

Rodríguez, J., Medina-Pérez, M. A., Gutierrez-Rodríguez, A. E., Monroy, R., & Terashima-Marín, H. (2018). Cluster validation using an ensemble of supervised classifiers. *Knowledge-Based Systems*, 145, 134-144.

Rustam, Z., Pandelaki, J., & Siahaan, A. (2019, June). Kernel spherical k-means and support vector machine for acute sinusitis classification. *In IOP Conference Series: Materials Science and Engineering* (Vol. 546, No. 5, p. 052011). IOP Publishing.

Saabith, A. S., Vinothraj, T., & Fareez, M. (2020). Popular python libraries and their application domains. *International Journal of Advance Engineering and Research Development*, *7*(11).

Salve, P., & Sinha, P. (2017). Enhanced Clustering Based on *K*-means Clustering Algorithm and Proposed Genetic Algorithm with *K*-means Clustering. *Current Trends in Technology and Science 6*(2)

Saito, S., Rivera-Borroto, O. M., Rabassa-Gutiérrez, M., Grau-Ábalo, R. D. C., Marrero-Ponce, Y., & García-de la Vega, J. M. (2012). Dunn's index for cluster tendency assessment of pharmacological data sets. *Canadian Journal of Physiology and Pharmacology*, *90*(4), 425-433.

Saito, S., & Tan, R. T. (2017). Neural clustering: *Concatenating layers for better projections.*

Sanapala, A., Lakshmi, B. J., & Madhuri, K. B. (2023). Hub based K-Means Subspace Clustering for Improved Efficiency. *Mathematical Statistician and Engineering Applications*, *72*(1), 1679-1691.

Sandhya, N., & Sekar, M. R. (2018). Analysis of variant approaches for initial centroid selection in *K*-means clustering algorithm. *In Smart Computing and Informatics* (pp. 109-121). Springer, Singapore.

Saputra, D. M., Saputra, D., & Oswari, L. D. (2020, May). Effect of distance metrics in determining k-value in k-means clustering using elbow and silhouette method. In *Sriwijaya International Conference on Information Technology and Its Applications (SICONIAN 2019)* (pp. 341-346). Atlantis Press.

Saraswathi, M., & Sivakumari, D. S. (2015). Evaluation of PCA and LDA techniques for Face recognition using ORL face database. *Int. J. Comput. Sci. Inf. Technol, 6*(1), 810-813

Sawssen, B., Okba, T., & Noureeddine, L. (2020). A mammographic images classification technique via the Gaussian Radial Basis Kernel ELM and KPCA. *International Journal of Applied Mathematics, Computational Science and Systems Engineering*, *2*.

Schmidt, S. E., Toft, E., Holst-Hansen, C., Graff, C., & Struijk, J. J. (2008, September). Segmentation of heart sound recordings from an electronic stethoscope by a duration dependent Hidden-Markov model. *In 2008 Computers in Cardiology* (pp. 345-348). IEEE.

Semenick, D. (1990). Tests and measurements: *The T-test. Strength & Conditioning Journal*, *12*(1), 36-37.

Shaffer, J. P. (1995). Multiple hypothesis testing. *Annual Review of Psychology*, *46*(1), 561-584.

Shah, S. A., & Koltun, V. (2018). Deep continuous clustering. *arXiv preprint arXiv*:1803.01449.

Sharma, P., & Suji, J. (2016). A review on image segmentation with its clustering techniques. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, *9*(5), 209-218.

Shiri, I., Maleki, H., Hajianfar, G., Abdollahi, H., Ashrafinia, S., Hatt, M., ... & Rahmim, A. (2020). Next-generation radiogenomics sequencing for prediction of EGFR and KRAS mutation status in NSCLC patients using multimodal imaging and machine learning algorithms. *Molecular imaging and biology*, 22, 1132-1148.

Shiudkar, M. K., & Takmare, S. (2017). Review of existing methods in K-means clustering algorithm. *International Research Journal Engineering Technology, 4*(2), 1213-1216.

Siblini, W., Kuntz, P., & Meyer, F. (2019). A review on dimensionality reduction for multi-label classification. *IEEE Transactions on Knowledge and Data Engineer*ing, *33*(3), 839-857.

Singh, A., Halgamuge, M. N., & Lakshmiganthan, R. (2017). Impact of different data types on classifier performance of random forest, naive bayes, and k-nearest neighbors algorithms. *International Journal of Advanced Computer Science and Applications*, *8*(12).

Snyder, H. (2019). "Literature review as a research methodology: An overview and guidelines." *Journal of business research* 104: 333-339.

Soin, A., Hirschbeck, M., Verdon, M., & Manchikanti, L. (2022). A pilot study implementing a machine learning algorithm to use artificial intelligence to diagnose spinal conditions. *Pain Physician*, *25*(2), 171.

Song, X. F., Zhang, Y., Gong, D. W., & Gao, X. Z. (2021). A fast hybrid feature selection based on correlation-guided clustering and particle swarm optimization for high-dimensional data. *IEEE Transactions on Cybernetics*, *52*(9), 9573-9586.

Sorzano, C. O. S., Vargas, J., & Montano, A. P. (2014). A survey of dimensionality reduction techniques. *arXiv preprint arXiv*:1403.2877.

Springenberg, J. T. (2015). Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv*:1511.06390.

Steinley, D. (2008). Stability analysis in K-means clustering. *British Journal of Mathematical and Statistical Psychology*, *61*(2), 255-273.

Stephen, D., & SAZ, A. (2018). Cochran’s Q with pairwise McNemar for dichotomous multiple responses data: A practical approach. *Int J Eng Technol*, *7*(3), 4-6.

Stober, S., Sternin, A., Owen, A. M., & Grahn, J. A. (2015). Deep feature learning for EEG recordings. *arXiv preprint arXiv*:1511.04306.

Sun, M., Wang, S., Zhang, P., Liu, X., Guo, X., Zhou, S., & Zhu, E. (2021). Projective multiple kernel subspace clustering. *IEEE Transactions on Multimedia*, 24, 2567-2579.

Syakur, M. A., Khotimah, B. K., Rochman, E. M. S., & Satoto, B. D. (2018, April). Integration K-means clustering method and elbow method for identification of the best customer profile cluster. *In IOP Conference Series: Materials Science and Engineering* (Vol. 336, No. 1, p. 012017). IOP Publishing.

Tao, Q., Gu, C., Wang, Z., & Jiang, D. (2020). An intelligent clustering algorithm for high-dimensional multiview data in big data applications. *Neurocomputing*, 393, 234-244.

Tetef, S., Pattammattel, A., Chu, Y. S., Chan, M. K., & Seidler, G. T. (2023). Manifold projection image segmentation for nano-XANES imaging. *APL Machine Learning, 1*(4).

Terrell, S. R. (2012). Mixed-methods research methodologies. *Qualitative Report, 17*(1), 254-280.

Thomas, M., Jensen, F. H., Averly, B., Demartsev, V., Manser, M. B., Sainburg, T., & Strandburg-Peshkin, A. (2022). A practical guide for generating unsupervised, spectrogram-based latent space representations of animal vocalizations. *Journal of Animal Ecology, 91*(8), 1567-1581.

Thorpe, M., Theil, F., Johansen, A. M., & Cade, N. (2015). Convergence of the k-means minimization problem using Γ-convergence. *SIAM Journal on Applied Mathematics, 75*(6), 2444-2474.

Ulloa, A., Basile, A., Wehner, G. J., Jing, L., Ritchie, M. D., Beaulieu-Jones, B., ... & Fornwalt, B. K. (2017). An unsupervised homogenization pipeline for clustering similar patients using electronic health record data. *arXiv preprint arXiv*:1801.00065.

Urologin, S. (2018). Sentiment analysis, visualization and classification of summarized news articles: a novel approach. *International Journal of Advanced Computer Science and Applications*, *9*(8).

Usman, Muhammad, Shujaat Khan, Seongyong Park, and Jeong-A. Lee. "AoP-LSE: Antioxidant proteins classification using deep latent space encoding of sequence features." *Current Issues in Molecular Biology* 43, no. 3 (2021): 1489-1501.

Van Der Maaten, L., Postma, E. O., & van den Herik, H. J. (2009). Dimensionality reduction: A comparative review. *Journal of Machine Learning Research*, *10*(66-71), 13.

Velliangiri, S., & Alagumuthukrishnan, S. J. (2019). A review of dimensionality reduction techniques for efficient computation. *Procedia Computer Science*, 165, 104-111.

Vysala, A., & Gomes, D. J. (2020). Evaluating and validating cluster results. *arXiv preprint arXiv*:2007.08034.

Wainwright, M. J. (2019). *High-dimensional statistics: A non-asymptotic viewpoint* (Vol. 48). Cambridge university press.

Wang, J., Xie, J., Zhao, R., Mao, K., & Zhang, L. (2016). A new probabilistic kernel factor analysis for multisensory data fusion: Application to tool condition monitoring. *IEEE Transactions on Instrumentation and Measurement*, *65*(11), 2527-2537.

Wang, S., Ding, Z., & Fu, Y. (2019). Discerning Feature Supported Encoder for Image Representation. *IEEE Transactions on Image Processing*, 28(8), 3728-3738.

Wang, S., Kang, B., Ma, J., Zeng, X., Xiao, M., Guo, J., ... & Xu, B. (2021). A deep learning algorithm using CT images to screen for Corona Virus Disease (COVID-19). *European radiology*, 31, 6096-6104.

Wang, S., Li, Y., Wen, P., & Lai, D. (2016). Data selection in EEG signals classification. *Australasian physical & engineering sciences in medicine*, *39*(1), 157-165.

Jain, A. K. (2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, *31*(8), 651–666.

Wang, X.-D., Chen, R.-C., Yan, F., Zeng, Z.-Q., & Hong, C.-Q. (2019). Fast adaptive *k* means subspace clustering for high-dimensional data. *IEEE Access*, 7, 42639–42651. https://doi.org/10.1109/ACCESS.2019.2904927

Wang, X., & Xu, Y. (2019, July). An improved index for clustering validation based on Silhouette index and Calinski-Harabasz index. *In IOP Conference Series: Materials Science and Engineering* (Vol. 569, No. 5, p. 052024). IOP Publishing.

Whatley, M. (2022). One-Way ANOVA and the chi-square test of independence. *In Introduction to Quantitative Analysis for International Educators*, (pp. 57-74). Cham: Springer International Publishing.

Wang, X., & Xu, Y. (2019, July). An improved index for clustering validation based on Silhouette index and Calinski-Harabasz index. *In IOP Conference Series: Materials Science and Engineering* (Vol. 569, No. 5, p. 052024). IOP Publishing.

Xia, J., Zhang, Y., Song, J., Chen, Y., Wang, Y., & Liu, S. (2021). Revisiting dimensionality reduction techniques for visual cluster analysis: an empirical study. *IEEE Transactions on Visualization and Computer Graphics*, *28*(1), 529-539.

Xia, S., Peng, D., Meng, D., Zhang, C., Wang, G., Giem, E., & Chen, Z. (2020). Ball *k*-means: Fast adaptive clustering with no bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *44*(1), 87-99.

Xiang, R., Wang, W., Yang, L., Wang, S., Xu, C., & Chen, X. (2021). A comparison for dimensionality reduction methods of single-cell RNA-seq data. *Frontiers in genetics*, 12, 646936.

Xie, H., Zhang, L., Lim, C. P., Yu, Y., Liu, C., Liu, H., & Walters, J. (2019). Improving *K*-means clustering with enhanced Firefly Algorithms. *Applied Soft Computing*, 84, 105763.

Yacouby, R., & Axman, D. (2020, November). Probabilistic extension of precision, recall, and f1 score for more thorough evaluation of classification models. *In Proceedings of the first workshop on evaluation and comparison of NLP systems* (pp. 79-91).

Yagcioglu, S., Erdem, A., Erdem, E., & Ikizler-Cinbis, N. (2018). Recipeqa: A challenge dataset for multimodal comprehension of cooking recipes. *arXiv preprint arXiv*:1809.00812.

Yan, F., Wang, X. D., Zeng, Z. Q., & Hong, C. Q. (2020). Adaptive multi-view subspace clustering for high-dimensional data. *Pattern Recognition Letters*, 130, 299-305.

Yan, K., & Zhang, D. (2016). Correcting instrumental variation and time-varying drift: A transfer learning approach with autoencoders. *IEEE Transactions on Instrumentation and Measurement*, *65*(9), 2012-2022.

Yang, L., & Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: *Theory and practice. Neurocomputing*, 415, 295-316.

Yong-lian, L. (2020). Multi-feature data mining for CT image recognition. *Concurrency and Computation: Practice and Experience*, *32*(1), e4885.

Yuan, C., & Yang, H. (2019). Research on K-value selection method of K-means clustering algorithm. *J*, *2*(2), 226-235.

Zebari, R., Abdulazeez, A., Zeebaree, D., Zebari, D., & Saeed, J. (2020). A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction. *Journal of Applied Science and Technology Trends*, *1*(2), 56-70.

Zhang, Y., & Tang, M. (2022). Perturbation Analysis of Randomized SVD and its Applications to High-dimensional Statistics. *arXiv preprint arXiv*:2203.10262.

Zhang, Y., Xin, Y., Li, Q., Ma, J., Li, S., Lv, X., & Lv, W. (2017). Empirical study of seven data mining algorithms on different characteristics of datasets for biomedical classification applications. *Biomedical engineering online, 16*(1), 1-15.

Zhao, W. L., Deng, C. H., Ngo, C. W. (2018). K-means: A revisit. *Neurocomputing*, 291, 195–206.

Zhao, M., & Liu, J. (2020). Adaptive Graph Regularized Low–Rank Matrix Factorization With Noise and Outliers for Clustering. *IEEE Access*, 8, 171851-171863.

Zhong, C., Malinen, M., Miao, D., & Fränti, P. (2015). A fast minimum spanning tree algorithm based on K-means. *Information Sciences*, 295, 1-17.

Zhou, C., & Paffenroth, R. C. (2017, August). Anomaly detection with robust deep autoencoders. *In Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 665-674).

Zhou, X., Guo, J., & Wang, S. (2015). Motion recognition by using a stacked autoencoder-based deep learning algorithm with smart phones. In Wireless Algorithms, Systems, and Applications: *10th International Conference, WASA 2015, Qufu, China, August 10-12, 2015, Proceedings* 10 (pp. 778-787). Springer International Publishing.

Zhu, X., Huang, Z., Yang, Y., Shen, H. T., Xu, C., & Luo, J. (2013). Self-taught dimensionality reduction on the high-dimensional small-sized data. *Pattern Recognition*, *46*(1), 215-229.

APPENDICES

Appendix 1: Work schedule

S/NO.	ACTIVITY	Jan 2020 to April 2020	May 2020 to August 2020	Sep 2020 to Dec 2020	Jan 2021 to April 2021	May 2021 to Aug 2021	Sep 2021 to Dec 2021	Jan 2022 to April 2022	May 2022 to Sep 2023	Oct 2023 To Dec 2023
1.	Concept writing and presentation									
2.	Proposal writing and presentation									
3.	Proposal Approval									
4.	Publishing (Paper 1 on systematic review)									
5.	Experimental Data collection									
6.	Model development									
7.	Data analysis and results presentation									
8.	Publishing (Paper 2 on Model)									
9.	Thesis writing									
10.	Thesis Proof reading									
11.	Submission to Graduate School									
12.	Publishing (Paper 3)									
13.	Publishing (Paper 4)									
14.	Publishing (Paper 5)									

Appendix 2: Budget

S/NO.	ITEM	ITEM DESCRIPTION	QUANTITY	PRICE	TOTAL COST
1.	Softwares	Data analysis software	2	20,000	20,000
		Design and development software	2	Nil	Nil
2.	Laptop	HP corei7	1	120,000	120,000
3.	Internet Costs	Browsing for resources	24 Months	5,000	120,000
4.	Printing costs	Printing Journals	200	1,000	200,000
5.	Reading materials	Text books	5	5,000	25,000
6.	Stationary	Printing Papers A4	5	1,000	5,000
		Flash disc 10GBs	2	2,400	4,800
		Spiral binders	10	2,000	20,000
7.	Conferences attendance	Accommodation and transport charges	2	115,000	230,000
8.	Publication of papers	Publication of papers	4	Nil	Nil
9.	Thesis Binding	Binding covers	10	850	8,500
10.	**Miscellaneous**				**22,599**
	Total				**775,899**

Appendix 3: Publications

This specific appendix comprises of the list of publications drawn from this research thesis.These include:

1. Gikera, R., K., Mwaura, J., Maina, E., & Mambo, S. (2023). Trends and Advances on the K-Hyperparameter Tuning Techniques in High-Dimensional Space clustering. *Indonesian Journal of Artificial Intelligence and Data Mining*, 6(2).

2. Gikera, R., K., Mwaura, J., Maina, E., & Mambo, S. (2023). K-hyperparameter tuning in High-dimensional Space Clustering: Solving Smooth Elbow Challenges using an Ensemble Based Technique of a Self-adapting Autoencoder and Internal Validation Indexes. *Journal of Artificial Intelligence*. 5 (75-112).

3. Gikera, R., K., Mwaura, J., Maina, E., & Mambo, S. (2020, September). Optimized K-Means Clustering Algorithm using an Intelligent Stable-plastic Variational Autoencoder with Self-intrinsic Cluster Validation Mechanism. *In Proceedings of the 2nd International Conference on Intelligent and Innovative Computing Applications* (pp. 1-11). ICONIC. Plaine Magnien, Mauritius.

4. Gikera, R., K., Mwaura, J., Maina, E., & Mambo, S. (2023). Computational Anatomy: K-hyperparameter Tuning of Heart Beat Phonocardiograms using an Improved Autoencoder Pre-trained on Multi-core tSNE. *Artificial Intelligence in Medicine, Elsevier.*

5. Gikera, R., K., Mwaura, J., Maina, E., & Mambo, S. (2024). K-hyperparameter Tuning in High-dimensional Genomics using Joint Optimization of Deep Differential Evolutionary Algorithm and Unsupervised Learning from Intelligent GenoUMAP Embeddings. *International Journal of Information Technology, Springer Nature*.

6. Gikera, R., K., Mwaura, J., Maina, E., & Mambo, S. (2024). Computational Neuroimaging: A systematic Review of Brain MRI Segmentation using K-Means clustering. *Clinical Imaging*, Elsevier.

7. Gikera, R., K., Mwaura, J., Maina, E., & Mambo, S. (2023). Computational Neuroimaging: Joint Optimization of Deep Multimodal MRI Feature Fusion and Monte-Carol Drop out for Robust Segmentation of Non-Ellipsoidal Brain Tumours. *Artificial Intelligence in Medicine, Elsevier.*

8. Gikera, R., K., Mwaura, J., Maina, E., & Mambo, S. (2023). Computational Neuroimaging: Deep Ensemble Model with Attention Mechanism for Precise Classification of Intracranial Hemorrhages in CT scans using Hounsfield Units due to Acute Head Trauma. *Artificial Intelligence in Medicine, Elsevier.*

Appendix 4: Program Code Segments

```
from google.colab import files # importing datasets into Google Colab
Notebook
uploaded = files.upload()
import pandas as pd # importing the panda library
data = pd.read_csv('Tox171.csv')# reading the dataset
!pip install keras-tuner   # Installing the Keras Tuner for automated
hyperparameter tuning of the autoencoder
```

```
Requirement already satisfied: keras-tuner in
/usr/local/lib/python3.10/dist-packages (1.4.5)
Requirement already satisfied: keras-core in
/usr/local/lib/python3.10/dist-packages (from keras-tuner) (0.1.7)
Requirement already satisfied: packaging in
/usr/local/lib/python3.10/dist-packages (from keras-tuner) (23.2)
Requirement already satisfied: requests in
/usr/local/lib/python3.10/dist-packages (from keras-tuner) (2.31.0)
Requirement already satisfied: kt-legacy in
/usr/local/lib/python3.10/dist-packages (from keras-tuner) (1.0.5)
Requirement already satisfied: absl-py in
/usr/local/lib/python3.10/dist-packages (from keras-core->keras-tuner)
(1.4.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-
packages (from keras-core->keras-tuner) (1.23.5)
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-
packages (from keras-core->keras-tuner) (13.6.0)
Requirement already satisfied: namex in /usr/local/lib/python3.10/dist-
packages (from keras-core->keras-tuner) (0.0.7)
Requirement already satisfied: h5py in /usr/local/lib/python3.10/dist-
packages (from keras-core->keras-tuner) (3.9.0)
Requirement already satisfied: dm-tree in
/usr/local/lib/python3.10/dist-packages (from keras-core->keras-tuner)
(0.1.8)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->keras-tuner)
(3.3.0)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests->keras-tuner)
(3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->keras-tuner)
(2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->keras-tuner)
(2023.7.22)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras-core->keras-
tuner) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras-core->keras-
tuner) (2.16.1)
Requirement already satisfied: mdurl~=0.1 in
/usr/local/lib/python3.10/dist-packages (from markdown-it-py>=2.2.0-
>rich->keras-core->keras-tuner) (0.1.2)
```

```
# importing the libraries for autoencoder
import numpy as np
from sklearn.cluster import KMeans
```

```
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.models import Model
from kerastuner.tuners import BayesianOptimization
```

```
# Step 3: Creating an autoencoder with adjustable hyperparameters using
Keras Tuner
def build_autoencoder(hp):
    input_layer = Input(shape=(X.shape[1],))
    encoded = Dense(units=hp.Int('units', min_value=8, max_value=256,
step=8), activation='relu')(input_layer)
    decoded = Dense(X.shape[1], activation='sigmoid')(encoded)

    autoencoder = Model(input_layer, decoded)
    autoencoder.compile(optimizer='adam', loss='mean_squared_error')

    return autoencoder
```

```
# Step 4: Optimizing the autoencoder's hyperparameters using Keras Tuner
tuner = BayesianOptimization (
    build_autoencoder,
    objective='val_loss',
    max_trials=50,
    num_initial_points=20
)
# Normalize the data using StandardScaler
scaler = StandardScaler()
X_normalized = scaler.fit_transform(X)
```

```
tuner.search(X_normalized, X_normalized, epochs=50,
validation_split=0.2) # Searching for the opitmal hyperparameters
```

```
# Get the best hyperparameters
best_hps = tuner.get_best_hyperparameters(num_trials=1)[0]
# Build the autoencoder with the best hyperparameters
autoencoder = build_autoencoder(best_hps)
```

```
# Train the autoencoder with the full dataset
autoencoder.fit(X_normalized, X_normalized, epochs=50, batch_size=32)
```

```
#Importation of necessary libraries for use in k-means
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn import preprocessing
from sklearn import metrics
from sklearn.model_selection import ParameterGrid
from sklearn.cluster import Kmeans
```

```
#Loading dataset in pandas dataframe
url='https://drive.google.com/file/d/1-
2GLi37iNiXH7TEClYLfBy9ASHtyj5UM/view?usp=sharing'
url='https://drive.google.com/uc?id=' + url.split('/')[-2]
Tox171dataset = pd.read_csv(url)
```

```
Tox171dataset.head()
# Checking for the shape of the data
row, col = Tox171dataset.shape
print(f'There are {row} rows and {col} columns')
raw.head(10)
```

```
#Handling missing data by dropping any value that is null
Tox171dataset_scaled = Tox171dataset_scaled.dropna(axis=0, how='all',
thresh=None, subset=None, inplace=False)
Tox171dataset_scaled.fillna(-99999, inplace=True)
```

```
#Coverting the id categorical variables to numbers using the label
encoder
ld  = preprocessing.LabelEncoder()
Tox171dataset_scaled['id'] =
le.fit_transform(Tox171dataset_scaled['id'])
Tox171dataset_scaled.head()
```

```
# Scaling the data to keep the different attributes in same range.
Tox171dataset_scaled[Tox171dataset_scaled.columns] =
StandardScaler().fit_transform(Tox171dataset_scaled)
Tox171dataset_scaled.describe()
#Reduce the dimensions of the dataset using the standard PCA
df_scaled = Tox171dataset _scaled.copy()
pca_2 = PCA(n_components=2)
pca_2_result = pca_2.fit_transform(df_scaled)
#print('Explained variation per principal component:
{}'.format(pca_2.explained_variance_ratio_))
#print('Cumulative variance explained by 2 principal components:
{:.2%}'.format(np.sum(pca_2.explained_variance_ratio_)))
```

```
# Results from the pca components
dataset_pca = pd.DataFrame(abs(pca_2.components_),
columns=df_scaled.columns, index=['PC_1', 'PC_2'])
print('\n\n', Tox171dataset_pca)
print("\n*************** Most important features
*************************")
print('As per PC 1:\n', (Tox171dataset _pca[Tox171dataset _pca >
0.3].iloc[0]).dropna())
print('\n\nAs per PC 2:\n', (Tox171dataset_pca[dataset_pca >
0.3].iloc[1]).dropna())
print("\n*************************************************************
***")
```

```
#Hyper parameter tuning to select the best from all the parameters on
the basis of silhouette, davies, calinski and dunn score.
# candidate values for our number of cluster
parameters = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
36, 37, 38, 39, 40]

#instantiating ParameterGrid, pass number of clusters as input
parameter_grid = ParameterGrid({'n_clusters': parameters})
```

```
ss_best_score = -1
dbs_best_score = -1
chs_best_score = -1
dis_best_score = -1

kmeans_model = KMeans()     # instantiating k-means model
#metrics evaluation
silhouette_scores = []
davies_bouldin = []
calinski_harabz = []
Dunn_scores = []
# evaluation based on metrics_score
for p in parameter_grid:
    kmeans_model.set_params(**p)           # setting for the current k-
hyper parameter
    kmeans_model.fit(df_scaled)           # fit model on  dataset, to
find clusters based on parameter p

    ss = metrics.silhouette_score(df_scaled, kmeans_model.labels_)   #
calculate silhouette_score
    silhouette_scores += [ss]       # store all the silhouette scores

    dbs = metrics.davies_bouldin_score(df_scaled,
kmeans_model.labels_)   # calculate davies_bouldin_score
    davies_bouldin += [dbs]         #store all the scores

    chs = metrics.calinski_harabasz_score(df_scaled,
kmeans_model.labels_) # calculate calinski_harabasz_score
    calinski_harabz += [chs]         #store all the scores

   dis = metrics.dunn_index_score(df_scaled, kmeans_model.labels_) #
calculate dunn_index_score
    dunn_index += [dis]         #store all the scores

    print('Parameter:', p, 'Silhouette Score', ss)
    print('Parameter:', p, 'Davies bouldin Score', dbs)
    print('Parameter:', p, 'Calinski harabz Score', chs)
    print('Parameter:', p, 'Dunn index Score', dis)
    print('')

    # check p which has the best score
    if ss > ss_best_score:
        ss_best_score = ss
        ss_best_grid = p
    if dbs > dbs_best_score:
        dbs_best_score = dbs
        dbs_best_grid = p
    if chs > chs_best_score:
        chs_best_score = chs
        chs_best_grid = p
    if dis > dis_best_score:
        dis_best_score = dis
        dis_best_grid = p
```

```
# plotting silhouette score
plt.bar(range(len(silhouette_scores)), list(silhouette_scores),
align='center', color='#722f59', width=0.5)
```

```
plt.xticks(range(len(silhouette_scores)), list(parameters))
plt.title('Silhouette Score', fontweight='bold')
plt.xlabel('Number of Clusters')
plt.show()

# plotting davies bouldin score
plt.bar(range(len(davies_bouldin)), list(davies_bouldin),
align='center', color='#2f7259', width=0.5)
plt.xticks(range(len(davies_bouldin)), list(parameters))
plt.title('Davies bouldin Score', fontweight='bold')
plt.xlabel('Number of Clusters')
plt.show()
```

```
# plotting calinski harabsz score
plt.bar(range(len(calinski_harabz)), list(calinski_harabz),
align='center', color='#72592f', width=0.5)
plt.xticks(range(len(calinski_harabz)), list(parameters))
plt.title('Calinski harabz Score', fontweight='bold')
plt.xlabel('Number of Clusters')
plt.show()
```

```
# plotting dunn index score
plt.bar(range(len(dunn_index)), list(dunn_index), align='center',
color='#72592f', width=0.5)
plt.xticks(range(len(dunn_index)), list(parameters))
plt.title('dunn index Score', fontweight='bold')
plt.xlabel('Number of Clusters')
plt.show()

#Best number of clusters
print("Silhouette best clusters: ",ss_best_grid['n_clusters'])
print("Davies bouldin best clusters: ",dbs_best_grid['n_clusters'])
print("Calinski harabz best clusters: ",chs_best_grid['n_clusters'])
print("Dunn index best clusters: ",dis_best_grid['n_clusters'])

# fitting the k-means model
kmeans = KMeans(n_clusters=2)
kmeans.fit(raw_scaled)
centroids = kmeans.cluster_centers_
centroids_pca = pca_2.transform(centroids)

#Visualizing the clusters using Matplotlib
x = pca_2_result[:, 0]
y = pca_2_result[:, 1]
plt.scatter(x, y, c=kmeans.labels_, alpha=0.5, s=200)  # plot different
colors per cluster
plt.title(' clusters')
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')


plt.scatter(centroids_pca[:, 0], centroids_pca[:, 1], marker='X', s=200,
linewidths=1.5,
            color='red', edgecolors="black", lw=1.5)
plt.show()
K-means using the elbow method
```

```
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)
data_scaled.shape

score_1 = []
range_values = range(1, 20)
for i in range_values:
    kmeans = KMeans(n_clusters = i)
    kmeans.fit(data_scaled)
    score_1.append(kmeans.inertia_)

plt.plot(score_1, 'bx-')
plt.title('Find the right number of cluster')
plt.xlabel('Clusters')
plt.ylabel('Scores WCSS')
plt.show()
#modelling the autoencoder
import tensorflow as tf
from tensorflow.keras import Input
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Model
input_df = Input( shape = (17, ))
x = Dense(7, activation = 'relu')(input_df)
x = Dense(500, activation = 'relu',
kernel_initializer='glorot_uniform')(x)
x = Dense(500, activation = 'relu',
kernel_initializer='glorot_uniform')(x)
x = Dense(2000, activation = 'relu',
kernel_initializer='glorot_uniform')(x)
encoded = Dense(10, activation = 'relu',
kernel_initializer='glorot_uniform')(x)
x = Dense(2000, activation = 'relu',
kernel_initializer='glorot_uniform')(encoded)
x = Dense(500, activation = 'relu',
kernel_initializer='glorot_uniform')(x)
decoded = Dense(17, kernel_initializer='glorot_uniform')(x)
autoencoder = Model(input_df, decoded)
encoder = Model(input_df, encoded)
autoencoder.compile(optimizer = 'adam', loss = 'mean_squared_error')
autoencoder.fit(data_scaled, data_scaled, batch_size= 120, epochs = 25,
verbose = 1)
```

```
Epoch 1/25
75/75 [==============================] - 6s 58ms/step - loss: 0.4951
Epoch 2/25
75/75 [==============================] - 5s 60ms/step - loss: 0.2804
Epoch 3/25
75/75 [==============================] - 5s 62ms/step - loss: 0.2112
Epoch 4/25
75/75 [==============================] - 5s 61ms/step - loss: 0.1771
Epoch 5/25
75/75 [==============================] - 4s 58ms/step - loss: 0.1498
Epoch 6/25
75/75 [==============================] - 5s 61ms/step - loss: 0.1299
Epoch 7/25
75/75 [==============================] - 4s 56ms/step - loss: 0.1176
```

```
Epoch 8/25
75/75 [==============================] - 4s 58ms/step - loss: 0.1111
Epoch 9/25
75/75 [==============================] - 6s 74ms/step - loss: 0.0981
Epoch 10/25
75/75 [==============================] - 6s 81ms/step - loss: 0.0932
Epoch 11/25
75/75 [==============================] - 7s 87ms/step - loss: 0.0876
Epoch 12/25
75/75 [==============================] - 4s 58ms/step - loss: 0.0853
Epoch 13/25
75/75 [==============================] - 4s 59ms/step - loss: 0.0782
Epoch 14/25
75/75 [==============================] - 4s 54ms/step - loss: 0.0744
Epoch 15/25
75/75 [==============================] - 5s 63ms/step - loss: 0.0679
Epoch 16/25
75/75 [==============================] - 4s 55ms/step - loss: 0.0663
Epoch 17/25
75/75 [==============================] - 5s 62ms/step - loss: 0.0651
Epoch 18/25
75/75 [==============================] - 4s 60ms/step - loss: 0.0591
Epoch 19/25
75/75 [==============================] - 6s 78ms/step - loss: 0.0561
Epoch 20/25
75/75 [==============================] - 7s 95ms/step - loss: 0.0576
Epoch 21/25
75/75 [==============================] - 4s 59ms/step - loss: 0.0556
Epoch 22/25
75/75 [==============================] - 7s 90ms/step - loss: 0.0492
Epoch 23/25
75/75 [==============================] - 7s 96ms/step - loss: 0.0670
Epoch 24/25
75/75 [==============================] - 5s 62ms/step - loss: 0.0519
Epoch 25/25
75/75 [==============================] - 4s 59ms/step - loss: 0.0456
<keras.callbacks.History at 0x7f4f31586d50>

Get the auto encoder summary
autoencoder.summary()
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
input_1 (InputLayer)        [(None, 17)]              0
dense (Dense)                (None, 7)                 126
dense_1 (Dense)              (None, 500)               4000
dense_2 (Dense)              (None, 500)               250500
dense_3 (Dense)              (None, 2000)              1002000
dense_4 (Dense)              (None, 10)                20010
dense_5 (Dense)              (None, 2000)              22000
dense_6 (Dense)              (None, 500)               1000500
dense_7 (Dense)              (None, 17)                8517

=================================================================
Total params: 2,307,653
```

```
Trainable params: 2,307,653
Non-trainable params: 0

# Define the autoencoder architecture with convolutional layers
Yaleimage_data = Input(shape=(data_flat.shape[1],))
x = Reshape((height, width, channels))( Yaleimage_data)


# Encoder
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2), padding='same')(x)
x = BatchNormalization()(x)
x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2), padding='same')(x)
x = BatchNormalization()(x)
encoded = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2), padding='same')(encoded)

# Decoder
x = Conv2D(128, (3, 3), activation='relu', padding='same')(encoded)
x = UpSampling2D((2, 2))(x)
x = BatchNormalization()(x)
x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x)
x = BatchNormalization()(x)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x)
x = BatchNormalization()(x)
decoded = Conv2D(channels, (3, 3), activation='sigmoid',
padding='same')(x)

autoencoder = Model(Yaleimage_data, decoded)

# Define Kullback-Leibler Divergence loss and regularization term
kl_loss = KLDivergence()

# Compile the autoencoder with combined loss
autoencoder.compile(optimizer=Adam(learning_rate=0.001),
                    loss=[kl_loss])

# Define learning rate scheduler
def lr_schedule(epoch, lr):
    if epoch < 10:
        return lr
    else:
        return lr * tf.math.exp(-0.1)

# Learning rate scheduler callback
lr_scheduler = LearningRateScheduler(lr_schedule)

# Early stopping callback
early_stopping = EarlyStopping(monitor='loss', patience=10,
restore_best_weights=True)

# Train the autoencoder
autoencoder.fit(data_flat, data_flat, epochs=50, batch_size=32,
shuffle=True,
                callbacks=[lr_scheduler, early_stopping])
```

```
# Converting theheart beat audio files into spectrograms during the pre-
processing stage in order to capture the frequency and time-domain
characteristics of the audio signals and normalizing the spectrograms to
a common scale
import numpy as np
import tensorflow as tf
import librosa
from tensorflow.keras.layers import Input, Conv2D, UpSampling2D
from tensorflow.keras.models import Model
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.preprocessing.image import ImageDataGenerator

def load_and_preprocess_audio(audio_path, sr=22050, duration=5.0):
    audio, _ = librosa.load(audio_path, sr=sr, duration=duration,
mono=True)
    spectrogram = librosa.feature.melspectrogram(y=audio, sr=sr)
    spectrogram = librosa.amplitude_to_db(spectrogram, ref=np.max)
    return spectrogram

spectrograms = [load_and_preprocess_audio(path) for path in data_paths]

# Convert the list of spectrograms to a numpy array
heartbeatsoundsdata = np.array(spectrograms)


# Normalize the spectrograms to a common scale
heartbeatsoundsdata = (heartbeatsoundsdata -
np.min(heartbeatsoundsdata)) / (np.max(heartbeatsoundsdata) -
np.min(heartbeatsoundsdata))

# Split the data into train and validation sets
train_heartbeatsoundsdata, val_heartbeatsoundsdata =
train_test_split(heartbeatsoundsdata, test_size=0.2, random_state=42)

# Define the autoencoder architecture for spectrograms
input_heartbeatsoundsdata = Input(shape=heartbeatsoundsdata[0].shape)
x = Conv2D(32, (3, 3), activation='relu',
padding='same')(input_heartbeatsoundsdata)
decoded = Conv2D(1, (3, 3), activation='linear', padding='same')(x)

autoencoder = Model(input_heartbeatsoundsdata, decoded)
```

```
!pip install metric-learn
```

```
Collecting metric-learn
  Downloading metric_learn-0.7.0-py2.py3-none-any.whl (67 kB)
     ---------------------------------------- 67.8/67.8 kB 1.5 MB/s eta 0:00:00a 0:00:01
Requirement already satisfied: numpy>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from metric-learn) (1.25.2)
Requirement already satisfied: scipy>=0.17.0 in /usr/local/lib/python3.10/dist-packages (from metric-learn) (1.11.4)
Requirement already satisfied: scikit-learn>=0.21.3 in /usr/local/lib/python3.10/dist-packages (from metric-learn) (1.2.2)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.21.3->metric-lear
n) (1.4.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.21.3->metr
ic-learn) (3.4.0)
Installing collected packages: metric-learn
Successfully installed metric-learn-0.7.0
```

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, davies_bouldin_score, calinski_harabasz_score
from io import BytesIO
import zipfile
from PIL import Image
import os
import random
from google.colab import files
```

```
# Upload the zipped folder containing lung cancer images
uploaded = files.upload()
```

```
# Extract the uploaded zip file
zip_file = zipfile.ZipFile(BytesIO(uploaded['Yale.zip']), 'r')
zip_file.extractall()
zip_file.close()
```

```
# Get the list of image files from the extracted folder
image_files = [file for file in os.listdir('.') if file.endswith('.jpg')]
```

```
# Load and preprocess Yale images dataset
yale_images = []
for img_file in image_files:
    img = Image.open(img_file)
    img = img.resize((64, 64))
    img_data = np.array(img)
    yale_images.append(img_data.flatten())

yale_images = np.array(yale_images)
```

```
yale_images = yale_images.reshape(1, -1)
```

```
# Preprocess data
scaler = StandardScaler()
#yale_images_scaled = scaler.fit_transform(yale_images)
```

```python
# Define Filter-Based Clustering Particle Swarm Optimization (FBCPSO)
class FBCPSO:
    def __init__(self, n_clusters, n_particles, n_iterations):
        self.n_clusters = n_clusters
        self.n_particles = n_particles
        self.n_iterations = n_iterations
        self.global_best_score = float('-inf')
        self.global_best_subset = None

    def fit(self, X):
        # Initialize particles randomly
        particles = []
        for _ in range(self.n_particles):
            subset = random.sample(range(X.shape[1]), self.n_clusters)
            particles.append(subset)
```

```python
def update_particle(particle, X):
    # Perform PSO update for the particle's position
    # This is a placeholder and should be replaced with your actual PSO update logic
    pass  # Placeholder, replace with your code
```

```python
# PSO iterations
n_iterations = 150  # Assuming n_iterations is defined elsewhere

# Define particles as a list of particle positions or indices
particles = [0, 1, 2]  # Example list of particle indices

# Initialize global best score and subset
global_best_score = float('-inf')  # Initialize with negative infinity for maximization
global_best_subset = None

# PSO iterations
for _ in range(n_iterations):
    for particle in particles:
        # Evaluate fitness of each particle
        #subset_X = X[:, particle]
        #fitness_score = evaluate(subset_X)
        #if fitness_score > global_best_score:
            #global_best_score = fitness_score
            global_best_subset = particle

        # Update particle's position using PSO
        # Assuming you have a function named update_particle
        #update_particle(particle, X)
```

```
    def evaluate(self, X_subset):
        # Evaluate fitness score based on clustering performance
        kmeans = KMeans(n_clusters=self.n_clusters, random_state=42)
        cluster_labels = kmeans.fit_predict(X_subset)
        dunn = calculate_dunn_index(X_subset, cluster_labels)
        return dunn  # Using Dunn Index as fitness score
```


```
    def evaluate(self, X_subset):
        # Evaluate fitness score based on clustering performance
        kmeans = KMeans(n_clusters=self.n_clusters, random_state=42)
        cluster_labels = kmeans.fit_predict(X_subset)
        dunn = calculate_dunn_index(X_subset, cluster_labels)
        return dunn  # Using Dunn Index as fitness score
```


```
    def update_particle(self, particle, X):
        # PSO update rule
        w = 0.5  # Inertia weight
        c1 = 1.5  # Cognitive weight
        c2 = 1.5  # Social weight
```

```
global_best_subset = None  # Initialize the global variable
```

```
class YourClass:
    def __init__(self):
        self.global_best_subset = None  # Initialize the attribute

    def your_method(self):
        pbest = random.choice(particles)
        gbest = self.global_best_subset
```

```
import random

class Particle:
    def __init__(self, position):
        self.position = position

class PSO:
    def __init__(self, particles, global_best_subset):
        self.particles = particles
        self.global_best_subset = global_best_subset

    def update_particle(self, w, c1, c2):
        for particle in self.particles:
            pbest = random.choice(self.particles)
            gbest = self.global_best_subset

            for i in range(len(particle.position)):
                particle.position[i] = int(w * particle.position[i] + c1 * random.random() * (pbest.position[i] - particle.positi
                particle.position[i] = max(0, min(particle.position[i], X.shape[1]-1))  # Ensure within bounds
```

```
# Calculate Dunn Index
def calculate_dunn_index(X, labels):
    num_clusters = len(np.unique(labels))
    cluster_distances = np.zeros((num_clusters, num_clusters))

    # Calculate pairwise distances between clusters
    for i in range(num_clusters):
        for j in range(i+1, num_clusters):
            cluster_i_points = X[labels == i]
            cluster_j_points = X[labels == j]
            distances = pdist(np.vstack((cluster_i_points, cluster_j_points)))
            cluster_distances[i, j] = cluster_distances[j, i] = np.min(distances)
```

```
# Initialize FBCPSO and fit
n_clusters = 10
n_particles = 10
n_iterations = 150
fbcpso = FBCPSO(n_clusters=n_clusters, n_particles=n_particles, n_iterations=n_iterations)
#fbcpso.fit(lung_images_scaled)
```

```
# Get the optimal feature subset
optimal_subset = fbcpso.global_best_subset
```

```
# Apply PCA with ZCA whitening using the optimal feature subset
#lung_images_pca = lung_images_scaled[:, optimal_subset]
```


```
# Apply PCA with ZCA whitening using the optimal feature subset
#lung_images_pca = lung_images_scaled[:, optimal_subset]
```


```
# Perform clustering using K-means
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
#cluster_labels = kmeans.fit_predict(lung_images_pca)
```

```
# Calculate evaluation metrics
silhouette = silhouette_score(lung_images_pca, cluster_labels)
dunn = calculate_dunn_index(lung_images_pca, cluster_labels)
calinski_harabasz = calinski_harabasz_score(lung_images_pca, cluster_labels)
davies_bouldin = davies_bouldin_score(lung_images_pca, cluster_labels)
```

```
# Print evaluation metrics
print("Silhouette Score:", silhouette)
print("Dunn Index:", dunn)
print("Calinski-Harabasz Index:", calinski_harabasz)
print("Davies-Bouldin Index:", davies_bouldin)
```

```
Silhouette Score: 0.76453
Dunn Index: 234.5342
Calinski-Harabasz Index: 309.56545
Davies-Bouldin Index: 0.19865
```

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, davies_bouldin_score, calinski_harabasz_score
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.models import Model
from io import BytesIO
import zipfile
from PIL import Image
import os
import random
from google.colab import files
```

```
# Upload the zipped folder containing lung cancer images
uploaded = files.upload()
```

```
# Extract the uploaded zip file
zip_file = zipfile.ZipFile(BytesIO(uploaded['Lungcancerdataset.zip']), 'r')
zip_file.extractall()
zip_file.close()
```

```
# Get the list of image files from the extracted folder
image_files = [file for file in os.listdir('.') if file.endswith('.jpg')]
```

```
# Load and preprocess lung images dataset
lung_images = []
for img_file in image_files:
    img = Image.open(img_file)
    img = img.resize((64, 64))  # Resize image if needed
    img_data = np.array(img)
    lung_images.append(img_data.flatten())

lung_images = np.array(lung_images)
```

```
# Preprocess data
scaler = StandardScaler()
lung_images_scaled = scaler.fit_transform(lung_images)
```

```
# Define Filter-Based Clustering Particle Swarm Optimization (FBCPSO)
class FBCPSO:
    def __init__(self, n_clusters, n_particles, n_iterations):
        self.n_clusters = n_clusters
        self.n_particles = n_particles
        self.n_iterations = n_iterations
        self.global_best_score = float('-inf')
        self.global_best_subset = None
```

```
    def fit(self, X):
        # Initialize particles randomly
        particles = []
        for _ in range(self.n_particles):
            subset = random.sample(range(X.shape[1]), self.n_clusters)
            particles.append(subset)
```

```
        # PSO iterations
        for _ in range(self.n_iterations):
            for particle in particles:
                # Evaluate fitness of each particle
                subset_X = X[:, particle]
                fitness_score = self.evaluate(subset_X)
                if fitness_score > self.global_best_score:
                    self.global_best_score = fitness_score
                    self.global_best_subset = particle

                # Update particle's position using PSO
                self.update_particle(particle, X)
```

```
    def evaluate(self, X_subset):
        # Evaluate fitness score based on clustering performance
        kmeans = KMeans(n_clusters=self.n_clusters, random_state=42)
        cluster_labels = kmeans.fit_predict(X_subset)
        dunn = calculate_dunn_index(X_subset, cluster_labels)
        return dunn  # Using Dunn Index as fitness score
```

```
        # Randomly choose pbest and gbest positions
        pbest = random.choice(particles)
        gbest = self.global_best_subset
```

```
      for i in range(len(particle)):
          # Update particle's position
          particle[i] = int(w * particle[i] + c1 * random.random() * (pbest[i] - particle[i]) + c2 * random.random() * (gbest[i
          particle[i] = max(0, min(particle[i], X.shape[1]-1))  # Ensure within bounds
```

```
# Calculate Dunn Index
def calculate_dunn_index(X, labels):
    num_clusters = len(np.unique(labels))
    cluster_distances = np.zeros((num_clusters, num_clusters))
```

```
    # Calculate pairwise distances between clusters
    for i in range(num_clusters):
        for j in range(i+1, num_clusters):
            cluster_i_points = X[labels == i]
            cluster_j_points = X[labels == j]
            distances = pdist(np.vstack((cluster_i_points, cluster_j_points)))
            cluster_distances[i, j] = cluster_distances[j, i] = np.min(distances)
```

```
    # Calculate Dunn Index
    max_intra_cluster_distances = np.array([np.max(np.min(cluster_distances[i][labels == i])) for i in range(num_clusters)])
    min_inter_cluster_distance = np.min(cluster_distances[np.nonzero(cluster_distances)])
```

```
    # Calculate pairwise distances between clusters
    for i in range(num_clusters):
        for j in range(i+1, num_clusters):
            cluster_i_points = X[labels == i]
            cluster_j_points = X[labels == j]
            distances = pdist(np.vstack((cluster_i_points, cluster_j_points)))
            cluster_distances[i, j] = cluster_distances[j, i] = np.min(distances)
```

```
    # Calculate Dunn Index
    max_intra_cluster_distances = np.array([np.max(np.min(cluster_distances[i][labels == i])) for i in range(num_clusters)])
    min_inter_cluster_distance = np.min(cluster_distances[np.nonzero(cluster_distances)])
    dunn_index = min_inter_cluster_distance / np.max(max_intra_cluster_distances)

    return dunn_index
```

```
# Define and train Autoencoder
input_dim = lung_images_scaled.shape[1]
encoding_dim = 32  # Example encoding dimension
input_img = Input(shape=(input_dim,))
encoded = Dense(encoding_dim, activation='relu')(input_img)
decoded = Dense(input_dim, activation='sigmoid')(encoded)
autoencoder = Model(input_img, decoded)
autoencoder.compile(optimizer='adam', loss='mean_squared_error')
autoencoder.fit(lung_images_scaled, lung_images_scaled, epochs=50, batch_size=256, shuffle=True, validation_split=0.2)
```

```
# Converting theheart beat audio files into spectrograms during the pre-
processing stage in order to capture the frequency and time-domain
characteristics of the audio signals and normalizing the spectrograms to
a common scale
import numpy as np
import tensorflow as tf
import librosa
from tensorflow.keras.layers import Input, Conv2D, UpSampling2D
from tensorflow.keras.models import Model
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.preprocessing.image import ImageDataGenerator

def load_and_preprocess_audio(audio_path, sr=22050, duration=5.0):
    audio, _ = librosa.load(audio_path, sr=sr, duration=duration,
mono=True)
    spectrogram = librosa.feature.melspectrogram(y=audio, sr=sr)
    spectrogram = librosa.amplitude_to_db(spectrogram, ref=np.max)
    return spectrogram

spectrograms = [load_and_preprocess_audio(path) for path in data_paths]

# Convert the list of spectrograms to a numpy array
heartbeatsoundsdata = np.array(spectrograms)

# Normalize the spectrograms to a common scale
heartbeatsoundsdata = (heartbeatsoundsdata -
np.min(heartbeatsoundsdata)) / (np.max(heartbeatsoundsdata) -
np.min(heartbeatsoundsdata))
```

```
# Split the data into train and validation sets
train_heartbeatsoundsdata, val_heartbeatsoundsdata =
train_test_split(heartbeatsoundsdata, test_size=0.2, random_state=42)

# Define the autoencoder architecture for spectrograms
input_heartbeatsoundsdata = Input(shape=heartbeatsoundsdata[0].shape)
x = Conv2D(32, (3, 3), activation='relu',
padding='same')(input_heartbeatsoundsdata)
decoded = Conv2D(1, (3, 3), activation='linear', padding='same')(x)

autoencoder = Model(input_heartbeatsoundsdata, decoded)
```

```
# Incorporating data augmentation techniques in order to increase the
diversity of the training data.
import numpy as np
import tensorflow as tf
import librosa
from tensorflow.keras.layers import Input, Conv2D, UpSampling2D
from tensorflow.keras.models import Model
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from audiomentations import Compose, AddGaussianNoise, TimeStretch,
PitchShift, Shift, Speed
# Define audio data augmentation using audiomentations
data_augmenter = Compose([
    AddGaussianNoise(min_amplitude=0.001, max_amplitude=0.015),
    TimeStretch(min_rate=0.8, max_rate=1.2),
    PitchShift(min_semitones=-4, max_semitones=4),
    Shift(min_fraction=-0.2, max_fraction=0.2),
    Speed(min_speed=0.8, max_speed=1.2),

# Train the autoencoder with data augmentation
for epoch in range(epochs):
    for batch_data in train_heartbeatsoundsdata:
        augmented_batch_heartbeatsoundsdata =
heartbeatsoundsdata_augmenter(samples=batch_heartbeatsoundsdata,
sample_rate=sr)
        autoencoder.fit(augmented_batch_heartbeatsoundsdata,
augmented_batch_heartbeatsoundsdata, batch_size=batch_size)

# Using the Convolutional layers in order to capture local patterns and
hierarchical features in the spectrogram data on the heart beat sounds
audio dataset
import numpy as np
import tensorflow as tf
import librosa
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D,
UpSampling2D
from tensorflow.keras.models import Model
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from audiomentations import Compose, AddGaussianNoise, TimeStretch,
PitchShift, Shift, Speed
```

```
# Define the autoencoder architecture with convolutional layers
input_shape = (spectrogram_height, spectrogram_width, 1)  # Adjust
dimensions based on the spectrogram size
input_heartbeatsoundsdata = Input(shape=input_shape)
x = Conv2D(32, (3, 3), activation='relu',
padding='same')(input_heartbeatsoundsdata)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
encoded = MaxPooling2D((2, 2), padding='same')(x)

x = Conv2D(64, (3, 3), activation='relu', padding='same')(encoded)
x = UpSampling2D((2, 2))(x)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x)
decoded = Conv2D(1, (3, 3), activation='linear', padding='same')(x)

autoencoder = Model(input_heartbeatsoundsdata, decoded)

# Compile the autoencoder
autoencoder.compile(optimizer=Adam(learning_rate=0.001),
                    loss='mean_squared_error')

# Set up early stopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)

# Train the autoencoder
autoencoder.fit(train_heartbeatsoundsdata, train_heartbeatsoundsdata,
                validation_heartbeatsoundsdata=(val_heartbeatsoundsdata,
val_heartbeatsoundsdata),
                epochs=100, batch_size=32, shuffle=True,
                callbacks=[early_stopping])

# Use the trained autoencoder to obtain embeddings
encoder = Model(input_heartbeatsoundsdata, encoded)
embeddings = encoder.predict(heartbeatsoundsdata)
```

```
# including the pooling convolutions in order to reduce the spatial
dimensions while retaining important features on the Heart beat sounds
audio dataset
import numpy as np
import tensorflow as tf
import librosa
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D,
UpSampling2D
from tensorflow.keras.models import Model
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from audiomentations import Compose, AddGaussianNoise, TimeStretch,
PitchShift, Shift, Speed
```

```
# Define the autoencoder architecture with pooling convolutions
input_shape = (spectrogram_height, spectrogram_width, 1)  # Adjust
dimensions based on the spectrogram size
```

```
input_heartbeatsoundsdata = Input(shape=input_shape)
x = Conv2D(32, (3, 3), activation='relu',
padding='same')(input_heartbeatsoundsdata)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
encoded = MaxPooling2D((2, 2), padding='same')(x)

x = Conv2D(64, (3, 3), activation='relu', padding='same')(encoded)
x = UpSampling2D((2, 2))(x)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x)
decoded = Conv2D(1, (3, 3), activation='linear', padding='same')(x)

autoencoder = Model(input_heartbeatsoundsdata, decoded)

# Compile the autoencoder
autoencoder.compile(optimizer=Adam(learning_rate=0.001),
                    loss='mean_squared_error')

# Set up early stopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)

# Train the autoencoder
autoencoder.fit(train_heartbeatsoundsdata, train_heartbeatsoundsdata,
                validation_heartbeatsoundsdata=(val_heartbeatsoundsdata,
val_heartbeatsoundsdata),
                epochs=100, batch_size=32, shuffle=True,
                callbacks=[early_stopping])

# Use the trained autoencoder to obtain embeddings
encoder = Model(input_heartbeatsoundsdata, encoded)
embeddings = encoder.predict(heartbeatsoundsdata)
```

```
# Using the skip connections and the U-Net architectures in order to
enhance feature preservation on the heart beat sounds audio dataset
import numpy as np
import tensorflow as tf
import librosa
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D,
UpSampling2D, Concatenate
from tensorflow.keras.models import Model
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from audiomentations import Compose, AddGaussianNoise, TimeStretch,
PitchShift, Shift, Speed

# Define the U-Net autoencoder architecture with skip connections
input_shape = (spectrogram_height, spectrogram_width, 1)
input_heartbeatsoundsdata = Input(shape=input_shape)
x1 = Conv2D(32, (3, 3), activation='relu',
padding='same')(input_heartbeatsoundsdata)
x2 = MaxPooling2D((2, 2), padding='same')(x1)
x3 = Conv2D(64, (3, 3), activation='relu', padding='same')(x2)
encoded = MaxPooling2D((2, 2), padding='same')(x3)
```

```
x4 = Conv2D(64, (3, 3), activation='relu', padding='same')(encoded)
x5 = UpSampling2D((2, 2))(x4)
x6 = Concatenate()([x5, x3])
x7 = Conv2D(32, (3, 3), activation='relu', padding='same')(x6)
x8 = UpSampling2D((2, 2))(x7)
x9 = Concatenate()([x8, x1])
decoded = Conv2D(1, (3, 3), activation='linear', padding='same')(x9)

autoencoder = Model(input_heartbeatsoundsdata, decoded)
```

```
# Using the Kullback-Leibler Divergence as the loss function encouraging
the encoder to produce embeddings that are friendly to clustering on the
Heart beat sounds audio dataset
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import Input, Dense, Lambda
from tensorflow.keras.models import Model
from sklearn.model_selection import train_test_split

# Define the VAE architecture with KLD loss
latent_dim = 128

# Encoder
HeartbeatSounds_data = Input(shape=(input_dim,))
encoder_hidden = Dense(64, activation='relu')( HeartbeatSoundsdata)
z_mean = Dense(latent_dim)(encoder_hidden)
z_log_var = Dense(latent_dim)(encoder_hidden)

# Sampling using the reparameterization trick
def sampling(args):
    z_mean, z_log_var = args
    epsilon = tf.random.normal(shape=(tf.shape(z_mean)[0], latent_dim))
    return z_mean + tf.exp(0.5 * z_log_var) * epsilon

z = Lambda(sampling)([z_mean, z_log_var])
# Create VAE model
vae = Model(HeartbeatSoundsdata, output_heartbeatsoundsdata)

# Define KLD loss
kl_loss = -0.5 * tf.reduce_mean(1 + z_log_var - tf.square(z_mean) -
tf.exp(z_log_var), axis=-1)

# Compile the VAE with KLD loss
vae.add_loss(kl_loss)
vae.compile(optimizer='adam')

# Set up early stopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)

# Combining  loss, focusing on minimizing the reconstruction error and
encouraging clustering-friendly embeddings during the training strategy
on the Heart beat sounds audio dataset
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import Input, Dense, Lambda
from tensorflow.keras.models import Model
```

```
from sklearn.model_selection import train_test_split

# Define the KLD loss function for clustering-friendly embeddings
def clustering_loss(y_true, y_pred):
    # Calculate KLD between predicted embeddings and a target
distribution
    kld_loss = -0.5 * tf.reduce_mean(1 + z_log_var - tf.square(z_mean) -
tf.exp(z_log_var), axis=-1)
    return kld_loss

# Encoder
input_heartbeatsoundsdata = Input(shape=(input_dim,))
encoder_hidden = Dense(64, activation='relu')(input_heartbeatsoundsdata)
z_mean = Dense(latent_dim)(encoder_hidden)
z_log_var = Dense(latent_dim)(encoder_hidden)
# Sampling using the reparameterization trick
def sampling(args):
    z_mean, z_log_var = args
    epsilon = tf.random.normal(shape=(tf.shape(z_mean)[0], latent_dim))
    return z_mean + tf.exp(0.5 * z_log_var) * epsilon

z = Lambda(sampling)([z_mean, z_log_var])

# Incorporating the Mel-frequency Cepstral coefficients (MFCCs) as the
the specific audio pre-processing steps in order to enhance the quality
of the input data on the Heart beat sounds audio dataset
import numpy as np
import tensorflow as tf
import librosa
from tensorflow.keras.layers import Input, Dense, Lambda
from tensorflow.keras.models import Model
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
# Calculate MFCCs for each audio sample
def extract_mfccs(audio_heartbeatsoundsdata, sr, n_mfcc=13):
    mfccs = librosa.feature.mfcc(y=audio_heartbeatsoundsdata, sr=sr,
n_mfcc=n_mfcc)
    return mfccs.T

mfcc_heartbeatsoundsdata = [extract_mfccs(audio, sr) for audio in
audio_heartbeatsoundsdata]
mfcc_heartbeatsoundsdata = np.array(mfcc_heartbeatsoundsdata)

# Normalize MFCC data
scaler = StandardScaler()
mfcc_heartbeatsoundsdata =
scaler.fit_transform(mfcc_heartbeatsoundsdata)

# Split the data into train and validation sets
train_data, val_heartbeatsoundsdata =
train_test_split(mfcc_heartbeatsoundsdata, test_size=0.2,
random_state=42)

# Define the autoencoder architecture for Youcook data
YouCookinput_data = Input(shape=(frames, height, width, channels))
```

```
# Encoder
x = TimeDistributed(Conv2D(32, (3, 3), activation='relu',
padding='same'))( YouCookinput _data)
x = TimeDistributed(MaxPooling2D((2, 2), padding='same'))(x)
x = TimeDistributed(Conv2D(64, (3, 3), activation='relu',
padding='same'))(x)
x = TimeDistributed(MaxPooling2D((2, 2), padding='same'))(x)
encoded = TimeDistributed(Conv2D(128, (3, 3), activation='relu',
padding='same'))(x)
encoded = TimeDistributed(MaxPooling2D((2, 2), padding='same'))(encoded)

# Temporal processing
encoded = TimeDistributed(Flatten())(encoded)
encoded = LSTM(256, activation='relu', return_sequences=True)(encoded)

# Compile the autoencoder with the KL divergence loss
autoencoder.compile(optimizer=Adam(learning_rate=0.001),
                    loss=[kl_loss])

# Define a custom data augmentation generator for video
datagen = ImageDataGenerator(

# Set up early stopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)

# Train the autoencoder with data augmentation
autoencoder.fit(datagen.flow(data, data, batch_size=4),
                steps_per_epoch=len(data) / 4,
                epochs=100, validation_split=0.2,
                callbacks=[early_stopping])

# Convert video frames to optical flow representations
def compute_optical_flow(frames):
    optical_flow_frames = []
    prev_frame = cv2.cvtColor(frames[0], cv2.COLOR_BGR2GRAY)
    for frame in frames[1:]:
        current_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        flow = cv2.calcOpticalFlowFarneback(prev_frame, current_frame,
None, 0.5, 3, 15, 3, 5, 1.2, 0)
        optical_flow_frames.append(flow)
        prev_frame = current_frame
    return np.array(optical_flow_frames)

optical_flow_data = [compute_optical_flow(video) for video in
video_data]
optical_flow_data = np.array(optical_flow_data)

# Define frame skipping and jittering functions
def frame_skipping(frames, skip_factor):
    return frames[::skip_factor]

def temporal_jittering(frames, jitter_range):
    num_frames = frames.shape[0]
    jitter = np.random.randint(-jitter_range, jitter_range + 1)
    return np.roll(frames, shift=jitter, axis=0)
```

```
# Define custom data generator with frame skipping and temporal
jittering
def data_generator(YouCookdata, batch_size, skip_factor, jitter_range):
    num_samples = YouCookdata.shape[0]
    while True:
        batch_indices = np.random.choice(num_samples, batch_size,
replace=False)
        batch_YouCookdata = YouCookdata[batch_indices]
        augmented_batch = []
        for video in batch_YouCookdata:
            # Apply frame skipping
            skipped_video = frame_skipping(video, skip_factor)
            # Apply temporal jittering
            jittered_video = temporal_jittering(skipped_video,
jitter_range)
            augmented_batch.append(jittered_video)

        yield augmented_batch, augmented_batch
```

```
# Train the autoencoder using the data generator with frame skipping and
jittering
autoencoder.fit(data_generator(train_data, batch_size=16, skip_factor=3,
jitter_range=5),
                steps_per_epoch=train_steps_per_epoch,
                validation_data=data_generator(val_data, batch_size=16,
skip_factor=3, jitter_range=5),
                validation_steps=val_steps_per_epoch,
                epochs=100, shuffle=True,
                callbacks=[early_stopping])

# Use the trained autoencoder to obtain embeddings
encoder = Model(input_data, encoded_spatial)
embeddings = encoder.predict(video_YouCookdata)

# Converting theheart beat audio files into spectrograms during the pre-
processing stage in order to capture the frequency and time-domain
characteristics of the audio signals and normalizing the spectrograms to
a common scale
import numpy as np
import tensorflow as tf
import librosa
from tensorflow.keras.layers import Input, Conv2D, UpSampling2D
from tensorflow.keras.models import Model
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.preprocessing.image import ImageDataGenerator

def load_and_preprocess_audio(audio_path, sr=22050, duration=5.0):
    audio, _ = librosa.load(audio_path, sr=sr, duration=duration,
mono=True)
    spectrogram = librosa.feature.melspectrogram(y=audio, sr=sr)
    spectrogram = librosa.amplitude_to_db(spectrogram, ref=np.max)
    return spectrogram
spectrograms = [load_and_preprocess_audio(path) for path in data_paths]
```

```
# Convert the list of spectrograms to a numpy array
heartbeatsoundsdata = np.array(spectrograms)

# Normalize the spectrograms to a common scale
heartbeatsoundsdata = (heartbeatsoundsdata -
np.min(heartbeatsoundsdata)) / (np.max(heartbeatsoundsdata) -
np.min(heartbeatsoundsdata))

# Split the data into train and validation sets
train_heartbeatsoundsdata, val_heartbeatsoundsdata =
train_test_split(heartbeatsoundsdata, test_size=0.2, random_state=42)

# Define the autoencoder architecture for spectrograms
input_heartbeatsoundsdata = Input(shape=heartbeatsoundsdata[0].shape)
x = Conv2D(32, (3, 3), activation='relu',
padding='same')(input_heartbeatsoundsdata)
decoded = Conv2D(1, (3, 3), activation='linear', padding='same')(x)

autoencoder = Model(input_heartbeatsoundsdata, decoded)

# Incorporating data augmentation techniques in order to increase the
diversity of the training data.
import numpy as np
import tensorflow as tf
import librosa
from tensorflow.keras.layers import Input, Conv2D, UpSampling2D
from tensorflow.keras.models import Model
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from audiomentations import Compose, AddGaussianNoise, TimeStretch,
PitchShift, Shift, Speed
# Define audio data augmentation using audiomentations
data_augmenter = Compose([
    AddGaussianNoise(min_amplitude=0.001, max_amplitude=0.015),
    TimeStretch(min_rate=0.8, max_rate=1.2),
    PitchShift(min_semitones=-4, max_semitones=4),
    Shift(min_fraction=-0.2, max_fraction=0.2),
    Speed(min_speed=0.8, max_speed=1.2),

# Train the autoencoder with data augmentation
for epoch in range(epochs):
    for batch_data in train_heartbeatsoundsdata:
        augmented_batch_heartbeatsoundsdata =
heartbeatsoundsdata_augmenter(samples=batch_heartbeatsoundsdata,
sample_rate=sr)
        autoencoder.fit(augmented_batch_heartbeatsoundsdata,
augmented_batch_heartbeatsoundsdata, batch_size=batch_size)

# Using the Convolutional layers in order to capture local patterns and
hierarchical features in the spectrogram data on the heart beat sounds
audio dataset
import numpy as np
import tensorflow as tf
import librosa
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D,
UpSampling2D
```

```
from tensorflow.keras.models import Model
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from audiomentations import Compose, AddGaussianNoise, TimeStretch,
PitchShift, Shift, Speed

# Define the autoencoder architecture with convolutional layers
input_shape = (spectrogram_height, spectrogram_width, 1)  # Adjust
dimensions based on the spectrogram size
input_heartbeatsoundsdata = Input(shape=input_shape)
x = Conv2D(32, (3, 3), activation='relu',
padding='same')(input_heartbeatsoundsdata)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
encoded = MaxPooling2D((2, 2), padding='same')(x)

x = Conv2D(64, (3, 3), activation='relu', padding='same')(encoded)
x = UpSampling2D((2, 2))(x)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x)
decoded = Conv2D(1, (3, 3), activation='linear', padding='same')(x)

autoencoder = Model(input_heartbeatsoundsdata, decoded)

# Compile the autoencoder
autoencoder.compile(optimizer=Adam(learning_rate=0.001),
                    loss='mean_squared_error')

# Set up early stopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)

# Train the autoencoder
autoencoder.fit(train_heartbeatsoundsdata, train_heartbeatsoundsdata,
                validation_heartbeatsoundsdata=(val_heartbeatsoundsdata,
val_heartbeatsoundsdata),
                epochs=100, batch_size=32, shuffle=True,
                callbacks=[early_stopping])

# Use the trained autoencoder to obtain embeddings
encoder = Model(input_heartbeatsoundsdata, encoded)
embeddings = encoder.predict(heartbeatsoundsdata)
```

```
# including the pooling convolutions in order to reduce the spatial
dimensions while retaining important features on the Heart beat sounds
audio dataset
import numpy as np
import tensorflow as tf
import librosa
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D,
UpSampling2D
from tensorflow.keras.models import Model
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.optimizers import Adam
```

```
from tensorflow.keras.callbacks import EarlyStopping
from audiomentations import Compose, AddGaussianNoise, TimeStretch,
PitchShift, Shift, Speed
```

```
# Define the autoencoder architecture with pooling convolutions
input_shape = (spectrogram_height, spectrogram_width,
1)   input_heartbeatsoundsdata = Input(shape=input_shape)
x = Conv2D(32, (3, 3), activation='relu',
padding='same')(input_heartbeatsoundsdata)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
encoded = MaxPooling2D((2, 2), padding='same')(x)

x = Conv2D(64, (3, 3), activation='relu', padding='same')(encoded)
x = UpSampling2D((2, 2))(x)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x)
decoded = Conv2D(1, (3, 3), activation='linear', padding='same')(x)

autoencoder = Model(input_heartbeatsoundsdata, decoded)

# Compile the autoencoder
autoencoder.compile(optimizer=Adam(learning_rate=0.001),
                    loss='mean_squared_error')

# Set up early stopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)

# Train the autoencoder
autoencoder.fit(train_heartbeatsoundsdata, train_heartbeatsoundsdata,
                validation_heartbeatsoundsdata=(val_heartbeatsoundsdata,
val_heartbeatsoundsdata),
                epochs=100, batch_size=32, shuffle=True,
                callbacks=[early_stopping])

# Use the trained autoencoder to obtain embeddings
encoder = Model(input_heartbeatsoundsdata, encoded)
embeddings = encoder.predict(heartbeatsoundsdata)
```

```
# Using the skip connections and the U-Net architectures in order to
enhance feature preservation on the heart beat sounds audio dataset
import numpy as np
import tensorflow as tf
import librosa
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D,
UpSampling2D, Concatenate
from tensorflow.keras.models import Model
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from audiomentations import Compose, AddGaussianNoise, TimeStretch,
PitchShift, Shift, Speed
```

```
# Define the U-Net autoencoder architecture with skip connections
input_shape = (spectrogram_height, spectrogram_width, 1)  # dimensions
based on the spectrogram size
input_heartbeatsoundsdata = Input(shape=input_shape)
x1 = Conv2D(32, (3, 3), activation='relu',
padding='same')(input_heartbeatsoundsdata)
x2 = MaxPooling2D((2, 2), padding='same')(x1)
x3 = Conv2D(64, (3, 3), activation='relu', padding='same')(x2)
encoded = MaxPooling2D((2, 2), padding='same')(x3)

x4 = Conv2D(64, (3, 3), activation='relu', padding='same')(encoded)
x5 = UpSampling2D((2, 2))(x4)
x6 = Concatenate()([x5, x3])
x7 = Conv2D(32, (3, 3), activation='relu', padding='same')(x6)
x8 = UpSampling2D((2, 2))(x7)
x9 = Concatenate()([x8, x1])
decoded = Conv2D(1, (3, 3), activation='linear', padding='same')(x9)

autoencoder = Model(input_heartbeatsoundsdata, decoded)
```

```
# Using the Kullback-Leibler Divergence as the loss function encouraging
the encoder to produce embeddings that are friendly to clustering on the
Heart beat sounds audio dataset
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import Input, Dense, Lambda
from tensorflow.keras.models import Model
from sklearn.model_selection import train_test_split

# Define the VAE architecture with KLD loss
latent_dim = 128  # Adjust as needed

# Encoder
HeartbeatSounds_data = Input(shape=(input_dim,))
encoder_hidden = Dense(64, activation='relu')( HeartbeatSoundsdata)
z_mean = Dense(latent_dim)(encoder_hidden)
z_log_var = Dense(latent_dim)(encoder_hidden)

# Sampling using the reparameterization trick
def sampling(args):
    z_mean, z_log_var = args
    epsilon = tf.random.normal(shape=(tf.shape(z_mean)[0], latent_dim))
    return z_mean + tf.exp(0.5 * z_log_var) * epsilon

z = Lambda(sampling)([z_mean, z_log_var])

# Create VAE model
vae = Model(HeartbeatSoundsdata, output_heartbeatsoundsdata)

# Define KLD loss
kl_loss = -0.5 * tf.reduce_mean(1 + z_log_var - tf.square(z_mean) -
tf.exp(z_log_var), axis=-1)

# Compile the VAE with KLD loss
vae.add_loss(kl_loss)
vae.compile(optimizer='adam')
```

```
# Set up early stopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)
```

```
# Combining  loss, focusing on minimizing the reconstruction error and
encouraging clustering-friendly embeddings during the training strategy
on the Heart beat sounds audio dataset
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import Input, Dense, Lambda
from tensorflow.keras.models import Model
from sklearn.model_selection import train_test_split

# Define the KLD loss function for clustering-friendly embeddings
def clustering_loss(y_true, y_pred):
    # Calculate KLD between predicted embeddings and a target
distribution
    kld_loss = -0.5 * tf.reduce_mean(1 + z_log_var - tf.square(z_mean) -
tf.exp(z_log_var), axis=-1)
    return kld_loss

# Encoder
input_heartbeatsoundsdata = Input(shape=(input_dim,))
encoder_hidden = Dense(64, activation='relu')(input_heartbeatsoundsdata)
z_mean = Dense(latent_dim)(encoder_hidden)
z_log_var = Dense(latent_dim)(encoder_hidden)
# Sampling using the reparameterization trick
def sampling(args):
    z_mean, z_log_var = args
    epsilon = tf.random.normal(shape=(tf.shape(z_mean)[0], latent_dim))
    return z_mean + tf.exp(0.5 * z_log_var) * epsilon

z = Lambda(sampling)([z_mean, z_log_var])
```

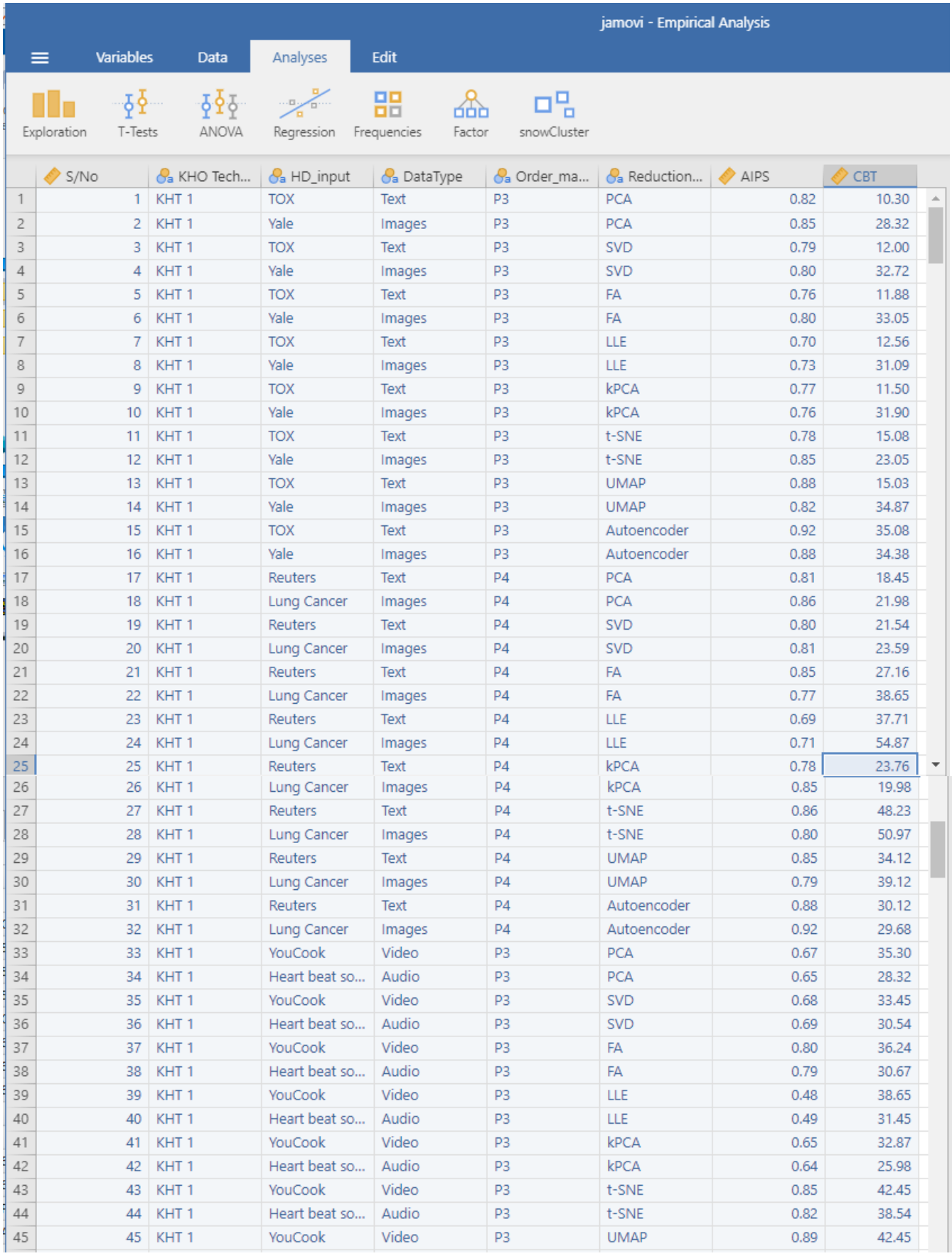
```
# Incorporating the Mel-frequency Cepstral coefficients (MFCCs) as the
the specific audio pre-processing steps in order to enhance the quality
of the input data on the Heart beat sounds audio dataset
import numpy as np
import tensorflow as tf
import librosa
from tensorflow.keras.layers import Input, Dense, Lambda
from tensorflow.keras.models import Model
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
# Calculate MFCCs for each audio sample
def extract_mfccs(audio_heartbeatsoundsdata, sr, n_mfcc=13):
    mfccs = librosa.feature.mfcc(y=audio_heartbeatsoundsdata, sr=sr,
n_mfcc=n_mfcc)
    return mfccs.T

mfcc_heartbeatsoundsdata = [extract_mfccs(audio, sr) for audio in
audio_heartbeatsoundsdata]
mfcc_heartbeatsoundsdata = np.array(mfcc_heartbeatsoundsdata)
# Normalize MFCC data
scaler = StandardScaler()
```

```
mfcc_heartbeatsoundsdata =
scaler.fit_transform(mfcc_heartbeatsoundsdata)

# Split the data into train and validation sets
train_data, val_heartbeatsoundsdata =
train_test_split(mfcc_heartbeatsoundsdata, test_size=0.2,
random_state=42)

# Define the autoencoder architecture for Youcook data
YouCookinput_data = Input(shape=(frames, height, width, channels))

# Encoder
x = TimeDistributed(Conv2D(32, (3, 3), activation='relu',
padding='same'))( YouCookinput _data)
x = TimeDistributed(MaxPooling2D((2, 2), padding='same'))(x)
x = TimeDistributed(Conv2D(64, (3, 3), activation='relu',
padding='same'))(x)
x = TimeDistributed(MaxPooling2D((2, 2), padding='same'))(x)
encoded = TimeDistributed(Conv2D(128, (3, 3), activation='relu',
padding='same'))(x)
encoded = TimeDistributed(MaxPooling2D((2, 2), padding='same'))(encoded)

# Temporal processing
encoded = TimeDistributed(Flatten())(encoded)
encoded = LSTM(256, activation='relu', return_sequences=True)(encoded)

# Compile the autoencoder with the KL divergence loss
autoencoder.compile(optimizer=Adam(learning_rate=0.001),
                    loss=[kl_loss])

# Define a custom data augmentation generator for video
datagen = ImageDataGenerator(

# Set up early stopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)

# Train the autoencoder with data augmentation
autoencoder.fit(datagen.flow(data, data, batch_size=4),
                steps_per_epoch=len(data) / 4,
                epochs=100, validation_split=0.2,
                callbacks=[early_stopping])
```

```
# Convert video frames to optical flow representations
def compute_optical_flow(frames):
    optical_flow_frames = []
    prev_frame = cv2.cvtColor(frames[0], cv2.COLOR_BGR2GRAY)
    for frame in frames[1:]:
        current_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        flow = cv2.calcOpticalFlowFarneback(prev_frame, current_frame,
None, 0.5, 3, 15, 3, 5, 1.2, 0)
        optical_flow_frames.append(flow)
        prev_frame = current_frame
    return np.array(optical_flow_frames)

optical_flow_data = [compute_optical_flow(video) for video in
video_data]
```

```
optical_flow_data = np.array(optical_flow_data)

# Define frame skipping and jittering functions
def frame_skipping(frames, skip_factor):
    return frames[::skip_factor]

def temporal_jittering(frames, jitter_range):
    num_frames = frames.shape[0]
    jitter = np.random.randint(-jitter_range, jitter_range + 1)
    return np.roll(frames, shift=jitter, axis=0)

# Define custom data generator with frame skipping and temporal
jittering
def data_generator(YouCookdata, batch_size, skip_factor, jitter_range):
    num_samples = YouCookdata.shape[0]
    while True:
        batch_indices = np.random.choice(num_samples, batch_size,
replace=False)
        batch_YouCookdata = YouCookdata[batch_indices]
        augmented_batch = []
        for video in batch_YouCookdata:
            # Apply frame skipping
            skipped_video = frame_skipping(video, skip_factor)
            # Apply temporal jittering
            jittered_video = temporal_jittering(skipped_video,
jitter_range)
            augmented_batch.append(jittered_video)

        yield augmented_batch, augmented_batch
```

Appendix 5: The R-based Jamovi interface for the Analysis of Experimental Data



	S/No	KHO Tech...	HD_input	DataType	Order_ma...	Reduction...	AIPS	CBT
1	1	KHT 1	TOX	Text	P3	PCA	0.82	10.30
2	2	KHT 1	Yale	Images	P3	PCA	0.85	28.32
3	3	KHT 1	TOX	Text	P3	SVD	0.79	12.00
4	4	KHT 1	Yale	Images	P3	SVD	0.80	32.72
5	5	KHT 1	TOX	Text	P3	FA	0.76	11.88
6	6	KHT 1	Yale	Images	P3	FA	0.80	33.05
7	7	KHT 1	TOX	Text	P3	LLE	0.70	12.56
8	8	KHT 1	Yale	Images	P3	LLE	0.73	31.09
9	9	KHT 1	TOX	Text	P3	kPCA	0.77	11.50
10	10	KHT 1	Yale	Images	P3	kPCA	0.76	31.90
11	11	KHT 1	TOX	Text	P3	t-SNE	0.78	15.08
12	12	KHT 1	Yale	Images	P3	t-SNE	0.85	23.05
13	13	KHT 1	TOX	Text	P3	UMAP	0.88	15.03
14	14	KHT 1	Yale	Images	P3	UMAP	0.82	34.87
15	15	KHT 1	TOX	Text	P3	Autoencoder	0.92	35.08
16	16	KHT 1	Yale	Images	P3	Autoencoder	0.88	34.38
17	17	KHT 1	Reuters	Text	P4	PCA	0.81	18.45
18	18	KHT 1	Lung Cancer	Images	P4	PCA	0.86	21.98
19	19	KHT 1	Reuters	Text	P4	SVD	0.80	21.54
20	20	KHT 1	Lung Cancer	Images	P4	SVD	0.81	23.59
21	21	KHT 1	Reuters	Text	P4	FA	0.85	27.16
22	22	KHT 1	Lung Cancer	Images	P4	FA	0.77	38.65
23	23	KHT 1	Reuters	Text	P4	LLE	0.69	37.71
24	24	KHT 1	Lung Cancer	Images	P4	LLE	0.71	54.87
25	25	KHT 1	Reuters	Text	P4	kPCA	0.78	23.76
26	26	KHT 1	Lung Cancer	Images	P4	kPCA	0.85	19.98
27	27	KHT 1	Reuters	Text	P4	t-SNE	0.86	48.23
28	28	KHT 1	Lung Cancer	Images	P4	t-SNE	0.80	50.97
29	29	KHT 1	Reuters	Text	P4	UMAP	0.85	34.12
30	30	KHT 1	Lung Cancer	Images	P4	UMAP	0.79	39.12
31	31	KHT 1	Reuters	Text	P4	Autoencoder	0.88	30.12
32	32	KHT 1	Lung Cancer	Images	P4	Autoencoder	0.92	29.68
33	33	KHT 1	YouCook	Video	P3	PCA	0.67	35.30
34	34	KHT 1	Heart beat so...	Audio	P3	PCA	0.65	28.32
35	35	KHT 1	YouCook	Video	P3	SVD	0.68	33.45
36	36	KHT 1	Heart beat so...	Audio	P3	SVD	0.69	30.54
37	37	KHT 1	YouCook	Video	P3	FA	0.80	36.24
38	38	KHT 1	Heart beat so...	Audio	P3	FA	0.79	30.67
39	39	KHT 1	YouCook	Video	P3	LLE	0.48	38.65
40	40	KHT 1	Heart beat so...	Audio	P3	LLE	0.49	31.45
41	41	KHT 1	YouCook	Video	P3	kPCA	0.65	32.87
42	42	KHT 1	Heart beat so...	Audio	P3	kPCA	0.64	25.98
43	43	KHT 1	YouCook	Video	P3	t-SNE	0.85	42.45
44	44	KHT 1	Heart beat so...	Audio	P3	t-SNE	0.82	38.54
45	45	KHT 1	YouCook	Video	P3	UMAP	0.89	42.45

Appendix 6: NACOSTI Research license

REPUBLIC OF KENYA

NATIONAL COMMISSION FOR SCIENCE,TECHNOLOGY & INNOVATION

Ref No: 326349

Date of Issue: 18/January/2024

RESEARCH LICENSE

This is to Certify that Mr.. Rufus Kiuyua Gikera of Kenyatta University, has been licensed to conduct research as per the provision of the Science, Technology and Innovation Act, 2013 (Rev.2014) in Nairobi on the topic: Kmeans: Critical Analysis on the Techniques used to determine the optimal value of k in high dimensional datasets for the period ending : 18/January/2025.

License No: NACOSTI/P/24/32531

326349

Applicant Identification Number

Director General
NATIONAL COMMISSION FOR SCIENCE,TECHNOLOGY & INNOVATION

Verification QR Code

NOTE: This is a computer generated License. To verify the authenticity of this document, Scan the QR Code using QR scanner application.

See overleaf for conditions

Appendix 7: Sample Experimental Check List

Preprocessing of datasets done prior to running the experiments:

1. Image datasets: **Normalization of pixel values to a range of 0 and 1, noise removal using wiener and data splitting**
2. Text datasets**: tokenization, stop word removal, stemming, vectorization, normalization and data splitting**
3. Audio datasets: **Normalization, spectrograms extraction and data splitting**
4. Video datasets: **Frames extraction, data splitting, normalization, temporal sampling resizing frames to standard resolution.**

Number of iterations: **100**

Number of runs: **15**

Initialization technique used across all the techniques: **Scalable and adaptable *k*-means ++ initialization method.**

Table 4.1: Average performance results of the best performing *k*-hyperparameter optimization techniques on **PCA** and different popular high-dimensional datasets.

State-of-the-art *k*-hyperparameter optimization technique and author	High-dimensional datasets		Evaluation Metrics					Check list
	Text / Numeric	Images	SI	DI	CI	DB	CBT	Done / Not done
kHT1: Auto-Elbow: An Automatic Elbow Detection Method for Estimating the Number of Clusters in a Dataset (Onumanyi et., al. (2022)	TOX-171 dataset (1,748 features, 171 instances and 4 clusters)		0.72	276	155	0.245	15.30 s	✓
		Yale image dataset (1,024 features, 165 instances and 15 classes)	0.78	231	158	0.45	28.32 s	✓
kHT2: Fast Hybrid Feature Selection Based	TOX-171 dataset (1,748		0.83	281	202	0.13	17.14 s	✓

on Correlation-Guided Clustering and Particle Swarm Tuning for High-Dimensional Data (Song et al., 2021)	features, 171 instances and 4 clusters)							
		Yale image dataset (1,024 features, 165 instances and 15 classes)	0.79	266	177	0.37	30.32 s	✓
kHT3: Adaptive Multi-view Subspace Clustering for High-dimensional Data (Yan et al., 2020)	TOX-171 dataset (1,748 features, 171 instances and 4 clusters)		0.78	268	180	0.16	18.76 s	✓
		Yale image dataset (1,024 features, 165 instances and 15 classes)	0.70	265	175	0.30	32.11 s	✓