UNIVERSITÉ DU QUÉBEC À MONTRÉAL

COMPARAISON DE DIFFÉRENTES APPROCHES DE RÉDUCTION DE LA DIMENSIONNALITÉ EN VUE DE L'OBTENTION D'UN MEILLEUR CLUSTERING

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

OUSMANE ASSANI AMATE

MAI 2026

ACKNOWLEDGMENTS

I would like to thank my entire family and my research supervisor, Vladimir Makarenkov, for guiding and supporting me during this period of my life. I would also like to thank Amaratou Mahamadou Saley and Mohammadreza for their time and valuable advice, which helped raise the quality of this work to a higher level.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

UQAM Université du Québec à Montréal.

DR Dimensionality Reduction.

PCA Principal Component Analysis.

KERNEL PCA Kernel Principal Component Analysis.

AE Autoencoders.

DAE Denoising Autoencoders.

VAE Variational Autoencoders.

LDR Linear Dimensionality Reduction.

NLDR Non-linear Dimensionality Reduction.

ICA Independent Component Analysis.

MDS Multidimensional Scaling.

ISOMAP Isometric Mapping.

LDA Linear Discriminant Analysis.

SVD Singular Value Decomposition.

t-SNE t-Stochastic Neighbor Embedding.

UMAP Uniform Manifold Approximation and Projection.

KL Kullback–Leibler.

GAN Generative Adversarial Network.

AHC Agglomerative Hierarchical Clustering.

CF Clustering Features.

DBSCAN Density-Based Spatial Clustering of Applications with Noise.

OPTICS Ordering Points To Identify the Clustering Structure.

GMM Gaussian Mixture Model.

FCM Fuzzy C-Means clustering.

ARI Adjusted Rand Index.

DEC Deep Embedded Clustering.

IDEC Improved Deep Embedded Clustering.

VaDE Variational Deep Embedding.

ABSTRACT

Dimensionality reduction is a critical preprocessing step for clustering high-dimensional data, yet comprehensive evaluations of its impact across diverse clustering algorithms and data types remain limited. In this study, we systematically assess the influence of five dimensionality reduction techniques such as Principal Component Analysis (PCA), Kernel Principal Component Analysis (Kernel PCA), Variational Autoencoders (VAEs), Isometric Mapping (Isomap), and Multidimensional Scaling (MDS) on the performance of four clustering algorithms: K-means, Agglomerative Hierarchical Clustering (AHC), Gaussian Mixture Models (GMM), and OPTICS. Experiments are conducted on 1,165 synthetic datasets with controlled geometric structures, as well as on 20 heterogeneous real-world benchmarks. Clustering performance is evaluated using the Adjusted Rand Index (ARI) under multiple reduction levels, including $k-1$ (where k is the number of clusters), and 25%, and 50% of the original dimensionality.

Our results show that the gains from dimensionality reduction are consistent but strongly dependent on the interaction between the embedding geometry and the downstream clustering algorithm. Three robust patterns emerge. First, method algorithm fit dominates; for example, Kernel PCA yields the most reliable improvements for centroid and distance-based methods, Isomap benefits GMM on nonlinear manifolds, while VAE embeddings tend to destabilize hierarchical and density-based clustering. Second, moderate dimensionality reduction strategies (25–50%) consistently outperform more aggressive settings such as reduction to $k-1$ dimensions, as, in most cases, they manage to reduce noise and redundancy while preserving the cluster-discriminative structure of the data. Third, domain characteristics matter: improvements are attenuated on real data, yet targeted pairings, such as Kernel PCA with OPTICS, provide measurable gains, whereas indiscriminate use of deep latent spaces can distort neighborhood topology. These findings position dimensionality reduction not as an optional preprocessing step, but as a first-order design variable in unsupervised learning pipelines. Overall, this study provides actionable guidelines and a reproducible benchmarking framework to help practitioners select dimensionality reduction techniques that align with their data geometry and clustering objectives.

RÉSUMÉ

La réduction de la dimension constitue une étape de prétraitement essentielle pour le clustering de données à haute dimension. Cependant, les évaluations exhaustives de son impact à travers des algorithmes de clustering variés et des types de données hétérogènes restent encore limitées. Dans ce travail, nous analysons de manière systématique l'influence de cinq techniques de réduction de dimension telles que l'Analyse en Composantes Principales (PCA), l'Analyse en Composantes Principales à Noyau (Kernel PCA), les Autoencodeurs Variationnels (VAEs), l'Isometric Mapping (Isomap) et le Multidimensional Scaling (MDS) sur les performances de quatre algorithmes de clustering : k-means, le clustering hiérarchique agglomératif (AHC), les mélanges gaussiens (GMM) et OPTICS. Les expériences sont menées sur 1 165 jeux de données synthétiques aux structures connues, ainsi que sur 20 jeux de données réels présentant une forte hétérogénéité. Les résultats sont évalués à l'aide de l'Adjusted Rand Index (ARI) sous différents niveaux de réduction de la dimension recommandés dans la littérature, y compris la réduction à $k-1$ dimensions (où k est le nombre de clusters ou classes), de même que 25 % et 50 % de la taille de la dimension initiale.

Nos résultats montrent que les gains apportés par la réduction de la dimension sont réels et cohérents, mais dépendent fortement de l'interaction entre la géométrie de l'espace réduit et l'algorithme de clustering utilisé. Trois observations principales se dégagent. Premièrement, l'adéquation du couple - méthode de réduction de la dimension + algorithme de clustering - est déterminante; par exemple, Kernel PCA offre les améliorations les plus fiables pour les algorithmes de clustering basés sur les distances ou les centroïdes, Isomap apporte un avantage notable pour le GMM sur des structures non linéaires, tandis que les représentations issues des VAEs tendent à déstabiliser le clustering hiérarchique et les algorithmes basés sur la densité. Deuxièmement, les stratégies de réduction de dimensionnalité modérées (25 à 50%) surpassent systématiquement les stratégies plus agressives, telles que la réduction à $k-1$ dimensions, car, dans la plupart des cas, elles parviennent à réduire le bruit et la redondance tout en préservant la structure discriminante des clusters présents dans les données. Troisièmement, les caractéristiques du domaine influencent fortement les performances : les gains diminuent sur les données réelles, bien que certains couplages ciblés, comme Kernel PCA avec OPTICS, restent bénéfiques, alors qu'un usage non maîtrisé de représentations profondes peut détériorer la topologie des voisinages. Ces résultats montrent que la réduction de la dimension ne doit pas être considérée comme une simple étape optionnelle, mais bien comme un paramètre clé dans la conception des pipelines non supervisés. Dans l'ensemble, cette étude fournit des recommandations pratiques et un cadre de comparaison reproductible pour aider les praticiens à choisir

des stratégies de réduction adaptées à la géométrie de leurs données et aux objectifs de clustering.

INTRODUCTION

Clustering is a fundamental technique in unsupervised machine learning that groups similar data points into clusters based on inherent patterns or similarities. Unlike supervised learning, it operates without labels. Hence, it has been applied to many different problems in areas such as cybersecurity, streaming, Internet-of-Things, anomaly detection, and others (de Amorim et Makarenkov, 2023). Machine learning is a technique that uses algorithms to analyze data, detect trends, and make predictions. Learning can be supervised, unsupervised, or reinforcement-based, depending on the presence or absence of labels in the data.

Although clustering algorithms are widely used, they have limitations when the data is highly noisy or has a large number of dimensions. In fact, in a high-dimensional space, the distance between points becomes less meaningful. This degrades the performance of clustering algorithms (Assent, 2012). This situation also makes data visualization and interpretation more complex. To solve these problems, dimensionality reduction (DR) is often used as a pre-processing step. Traditional techniques such as Principal Component Analysis (PCA) and several others can transform the initial data into a low-dimensional space while preserving the overall structure as much as possible. However, these techniques reach their limits when it comes to capturing complex nonlinear relationships.

Thus, with the development of deep learning, new and more powerful DR methods have emerged, such as autoencoders. These are neural networks that learn to compress data into a latent representation (encoding) and then reconstruct it. They differ from traditional methods in that they are capable of modeling complex nonlinear relationships and extracting more representative features for clustering. Much work has focused either on comparing clustering algorithms on raw data or on evaluating different DR methods, whether linear or nonlinear. Sometimes these studies have focused on analyzing limited combinations of clustering algorithms and RD methods or on fixing a single DR method and a single clustering algorithm. However, few studies have looked at a joint and systematic analysis of these techniques combined with various clustering algorithms.

It is therefore important to conduct an in-depth study that assesses the impact of these DR methods, whether traditional or not, on the quality of clusters and compares these results with those of clusters obtained from the original data without DR. The objective of this project is to evaluate the impact of dif-

ferent DR approaches on clustering quality and determine the most effective combinations. To achieve this objective, we must go through sub-objectives that will enable us to conduct this research properly. The specific objectives are as follows:

1. Generate different synthetic datasets for experimentation.
2. Implement and compare several linear and nonlinear dimensionality reduction techniques.
3. Apply various clustering algorithms to the initial and reduced data.
4. Evaluate the quality of the groupings obtained using metrics that measure the similarity between clusters.
5. Identify the most effective combinations of dimension reduction and clustering techniques for different data sets.

Research question

How does the use of different dimensionality reduction methods influence the quality of clustering, and which clustering algorithms produce the most relevant clusters based on the representations generated?

To meet the challenges presented by this project, we are adapting a methodology that consists of several successive steps. First, the data is preprocessed and normalized to ensure its quality and consistency for subsequent analysis. Next, several clustering algorithms are applied directly to the original data to obtain an initial assessment of clustering performance. The results obtained at this stage are evaluated using metrics that measure the similarity between clusters. In a second step, DR is first performed on the original data using classical techniques and autoencoder-based approaches. The same clustering algorithms are then reapplied to this reduced data set. The performance obtained is evaluated and compared using the same similarity metrics as those used in the analysis of the original data.

Finally, an in-depth comparative analysis is conducted to identify the most effective combinations of DR methods and clustering algorithms, with the aim of achieving more accurate and relevant data clustering. The major contributions of our work consist firstly of a systematic comparison of five DR methods from

different families across several levels of reduction and clustering algorithms in an unsupervised setting within a statistically controlled experimental design. Second, we propose practical recommendations for selecting RD methods based on data characteristics (size, dimensionality, intrinsic geometry) and clustering objectives, supported by open source code and reproducible experiments.

We will begin by presenting, in Chapter 1, a state of the art on the topics that will be addressed in this document. Chapter 2 describes the methodology used to try to provide solutions to the issues raised in this project and the datasets that were generated and used in our work. Finally, Chapter 3 shows the results of the experiments and analyzes their performance.

CHAPTER 1
LITERATURE REVIEW

This chapter provides a literature review overview of the principal dimensionality reduction approaches and clustering methods, as well as previous and current work combining these two fields. We begin by examining traditional dimensionality reduction methods, such as PCA, Kernel Principal Component Analysis (Kernel PCA) and Isometric Mapping (Isomap), presenting their principles, advantages, and limitations. We also explore deep learning-based methods, including autoencoders and their variants, such as Simple Autoencoders (AEs), Denoising Autoencoders (DAEs), and Variational Autoencoders (VAEs), and compare these approaches in terms of their ability to capture nonlinear structures. The chapter also discusses the various clustering algorithms, their classification, and clustering evaluation measures. Finally, this chapter will conclude by combining dimensionality reduction techniques and clustering algorithms, presenting studies found in the literature as well as a comparative analysis.

1.1 Dimensionality reduction

Dimensionality reduction (DR) refers to the process of representing a dataset using a reduced number of features (i.e., dimensions) while preserving the most informative and meaningful properties of the original data. This process involves the elimination of irrelevant, redundant, or noisy features in order to construct a more compact and efficient representation with fewer variables. DR encompasses a broad range of feature selection and data compression techniques commonly applied during the preprocessing stage. Although dimensionality reduction methods differ in their underlying mechanisms, they all aim to transform data from a high-dimensional space to a low-dimensional one through variable extraction, selection, or combination. Ideally, the dimensionality of the reduced representation should approximate the intrinsic dimensionality of the data. The intrinsic dimensionality is defined as the minimum number of parameters required to account for the observed properties of a dataset (Fukunaga, 2013).

In machine learning, dimensions (or features) correspond to the predictor variables that determine a model's output and are also referred to as input variables. High-dimensional data generally describe datasets characterized by a large number of predictor variables. Such datasets frequently arise in fields such as biostatistics and social science observational studies, where the number of features may be large relative to the number of observations. High-dimensional datasets introduce several practical challenges for machine learning al-

gorithms, including increased computational cost and substantial storage requirements. However, the most critical issue is often the degradation of predictive performance. Statistical and machine learning models trained on high-dimensional data are prone to overfitting and frequently exhibit poor generalization to unseen data.

DR is important in many domains, since it mitigates the curse of dimensionality and other undesired properties of high-dimensional spaces (Jimenez et Landgrebe, 2002). The curse of dimensionality refers to the inverse relationship between increasing model dimensions and decreasing generalizability. As the number of model input variables increases, the model's space increases. If the number of data points remains the same, however, the data becomes sparse. DR can do in two different methods. In the first method, feature selection techniques are used to select essential features from the input dataset. The second method is feature extraction, which creates new features from the existing features in the input dataset.

DR techniques are generally classified into two main categories: linear and nonlinear methods. Linear dimensionality reduction (LDR) methods transform data using linear operations. Conceptually, these methods project the original high-dimensional data into a lower-dimensional subspace, where the new features are linear combinations of the original ones. Such transformations can be interpreted as geometric operations including rotations, scalings, and shearings of the data cloud, however, they remain fundamentally constrained to flat, linear subspaces.

Many datasets, especially those involving images, text, or complex biological systems, do not follow a simple linear structure. Data points may be distributed across a variety of lower dimensions, forming a kind of curved surface or complex entity embedded in a higher-dimensional space. For example, images of the handwritten numeral "3" may vary in slant, thickness, and style, but they all fundamentally belong to a variety of "three-ness" that is much lower in dimension than raw pixel space. Non-linear dimensionality reduction (NLDR) techniques are specifically designed to uncover and unfold such manifolds, producing a lower-dimensional representation that more faithfully reflects the intrinsic geometry of the data. These methods prioritize the preservation of local neighborhood relationships, ensuring that points close in the original high-dimensional space remain proximate in the reduced embedding.

1.1.1 Traditional methods

Traditional dimensionality reduction techniques are essential for managing high-dimensional datasets in machine learning and data analysis. These methods help in reducing the number of features while retaining the most important information, improving model performance, and making complex data easier to visualize and interpret.

1.1.1.1 Principal Component Analysis

Principal Component Analysis (PCA) was first introduced by Pearson (1901) and later independently developed and formalized by Hotelling (1933). As an unsupervised linear dimensionality reduction technique, PCA identifies a set of orthogonal directions, known as principal components, that capture the maximum variance in the data through the eigenvalue decomposition of the covariance matrix. Although PCA is particularly effective when the data approximately follow a Gaussian distribution, its applicability extends broadly to diverse datasets owing to its clear geometric interpretation and high computational efficiency.

According to Pearson (1901), PCA is the problem of finding the line or plane of closest fit to a system of points in space, where "closest fit" is defined by minimizing the sum of squared perpendicular distances from the points to that line or plane. In practice, it is mostly implemented via eigenvalue decomposition of the covariance matrix or by means of Singular Value Decomposition (SVD). Alternative formulations include latent variable models and probabilistic interpretations. However, PCA is not typically derived from linear regression or factor analysis in its standard formulation (Meng *et al.*, 2016; Jolliffe et Cadima, 2016; Erichson *et al.*, 2020). Over the decades, PCA has become a cornerstone of multivariate statistical analysis, with wide-ranging applications in image and speech processing, data visualization, exploratory data analysis, pattern recognition, and robotic sensor data interpretation (Jolliffe et Cadima, 2016).

The primary objective of PCA is to extract a set of mutually uncorrelated features, known as Principal Components (PCs). The first principal component captures the largest proportion of the total variance in the data, the second captures the next largest proportion subject to being orthogonal to the first, and so on. The first k PCs typically retain the majority of the variability present in the original dataset, thereby reducing its dimensionality from p to k, with $k \ll p$. Only the most significant principal components, which preserve the maximum amount of information, are usually selected for low-dimensional data visualization and analysis. Let $\mathbf{Y} \in \mathbb{R}^{\boldsymbol{m}\times\boldsymbol{p}}$ be an $m \times p$ matrix having m observations and p features. The PCs $z_i \in \mathbb{R}^m$

can be computed as a linear weighted combination of features (Erichson *et al.*, 2020).

$$\mathbf{Z} = \mathbf{YW}. \tag{1.1}$$

Here **Z** $= [z_1, z_2, \ldots, z_m] \in \mathbb{R}^{m \times p}$ and **W** $= [w_1, w_2, \ldots, w_p \in \mathbb{R}^{p \times p}]$. The transformed data minimize reconstruction error while maximizing retained variance across all possible projections (Wiskott, 2013).

1.1.1.2 Independent Component Analysis

Independent Component Analysis (ICA), introduced by (Comon, 1994) is a statistical technique that seeks a linear transformation of the observed data such that the resulting components are as statistically independent as possible. The fundamental idea behind ICA is to assume an implicit statistical model for the observed variables:

$$\boldsymbol{x} = \mathbf{A}\boldsymbol{s}. \tag{1.2}$$

The framework described is known as ICA, which models observed data as a linear combination of statistically independent latent components. These independent components represent latent (unobserved) variables, while the mixing matrix **A** is considered unknown. Only the random vector x is directly observable, and the goal of ICA is to estimate both A and s based on a limited set of assumptions. The ICA assumes that the components are statistically independent, non-Gaussian distributed, and the unknown mixed matrix is a square matrix. If the inverse **W** of **A** can be calculated, the independent components can be calculated from:

$$\boldsymbol{s} = \mathbf{W}\boldsymbol{x}. \tag{1.3}$$

Consequently, the ICA model cannot identify either the variances or the ordering of the independent components. To address this limitation, ICA employs an objective function that quantifies the residual statistical dependence among the estimated components. A variety of algorithms have thus been developed to estimate these independent components, each based on a distinct choice of objective function (Rahmanishamsi *et al.*, 2018). ICA has found applications in blind source separation, Bayesian detection, data analysis, compression, and source localization (Comon, 1994). Compared to PCA, ICA often yields components that are more interpretable, as they are derived through an independence-based optimization criterion rather than by maximizing variance, as in PCA (Scholz *et al.*, 2004). Moreover, ICA is capable of extracting a greater

amount of meaningful information embedded in the observed data (Pochet *et al.*, 2004). In addition to reducing the risk of overfitting, ICA also permits reconstruction of the data in the original feature space (Pochet *et al.*, 2004). A major challenge with ICA algorithms, however, lies in their inherent stochasticity. Most ICA methods rely on gradient-based optimization to solve problems such as maximizing the non-Gaussianity of the sources (Hyvärinen et Oja, 1997), minimizing mutual information (Cichocki *et al.*, 1996), or performing maximum likelihood estimation (Hyvärinen *et al.*, 1998).

1.1.1.3 Kernel Principal Component Analysis

Kernel Principal Component Analysis (Kernel PCA) introduced by Schölkopf *et al.* (1997), extends conventional PCA to high-dimensional feature spaces through the use of kernel methods. Unlike standard PCA, which computes the eigenvectors of the covariance matrix (Van Der Maaten *et al.*, 2009), Kernel PCA performs an eigen-decomposition of the kernel matrix. The application of PCA in the kernel space provides Kernel PCA the property of constructing nonlinear mappings. Kernel PCA calculates the kernel matrix $\boldsymbol{K}$ corresponding to the data points x_i. Each element of this matrix is defined as follows:

$$\boldsymbol{k}_{ij} = \boldsymbol{\kappa}(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{1.4}$$

where $\boldsymbol{k}$ is a kernel function. Subsequently, the kernel matrix **K** is centered using the following modification of the entries:

$$\boldsymbol{k}_{ij} = \boldsymbol{k}_{ij} - \frac{1}{n}\sum_{l}\boldsymbol{k}_{il} - \frac{1}{n}\sum_{l}\boldsymbol{k}_{jl} + \frac{1}{n^2}\sum_{l}\sum_{m}\boldsymbol{k}_{lm}. \tag{1.5}$$

The centering process is equivalent to subtracting the mean of the features, as done in conventional PCA. This ensures that the mapped features in the high-dimensional space induced by the kernel function have a zero mean. Subsequently, the principal d eigenvectors v_i of the centered kernel matrix are computed. The eigenvectors of the covariance matrix α_i can now be computed, since they are related to the eigenvectors of the kernel matrix v_i through (Chatfield, 2018):

$$\boldsymbol{\alpha}_i = \frac{1}{\sqrt{\boldsymbol{\lambda}_i}}\boldsymbol{X}\boldsymbol{v}_i. \tag{1.6}$$

To derive the low-dimensional representation of the data, the samples are projected onto the eigenvectors of the covariance matrix α_i. The resulting projection, denoted as **Y**, corresponds to the low-dimensional

embedding of the data:

$$\boldsymbol{y}_i = \left\{ \sum_{j=1}^{n} \boldsymbol{\alpha}_1^j \, \boldsymbol{\kappa}(\boldsymbol{x}_j, \boldsymbol{x}_i), \; \ldots, \; \sum_{j=1}^{n} \boldsymbol{\alpha}_d^j \, \boldsymbol{\kappa}(\boldsymbol{x}_j, \boldsymbol{x}_i) \right\}. \tag{1.7}$$

Here α_1^j denotes the jth entry of the vector α_1 and k refers to the kernel function used in the construction of the kernel matrix. A well-known limitation of Kernel PCA is its potentially prohibitive computational cost, which can introduce numerical difficulties when diagonalizing large matrices (Rosipal et Girolami, 2001). To address these limitations, Rosipal et Girolami (2001) introduced an EM-based algorithm for Kernel PCA . This expectation–maximization approach enables the execution of Kernel PCA efficiently, with experimental results demonstrating its computational advantages, particularly when dealing with large datasets. One drawback of this approach however is that it needs to still store the $\boldsymbol{N} \times \boldsymbol{N}$ kernel matrix, which limits its applicability in many large dataset problems.

1.1.1.4 Multidimensional Scaling

Multidimensional Scaling (MDS) introduced by Kruskal (1978) is an unsupervised NLDR technique. Its primary objective is to construct a low-dimensional embedding that preserves the pairwise (dis)similarities among data points. MDS has been used for exploratory analysis, multivariate analysis, and visualization. A key advantage of MDS is its ability to produce graphical representations of complex datasets, thereby enhancing interpretability and facilitating intuitive understanding of underlying structural relationships. MDS performs effectively when the input (dis)similarity matrix corresponds to a configuration of points embeddable in a d-dimensional space such that the original pairwise distances are faithfully maintained in the embedding.

In multidimensional scaling, the configuration is obtained by minimizing a least-squares type criterion that measures the discrepancy between the observed dissimilarities and the interpoint distances in the low-dimensional embedding. This idea is classically formalized through Kruskal's stress function. Let **D** denote the distance matrix containing the pairwise dissimilarities between the observations in the input dataset **Y**. Multidimensional Scaling (MDS) seeks a low dimensional embedding of these points such that the distances d'_{ij} in the reduced space approximate the original distances d_{ij} as closely as possible. This procedure can

be expressed as an optimization problem:

$$\boldsymbol{\delta}(\boldsymbol{Z}) = \min \left(\sum_{i=1}^{m} \sum_{j=1}^{m} (\boldsymbol{d}_{ij} - \boldsymbol{d}'_{ij})^2 \right). \tag{1.8}$$

The closeness is measured by stress function which is calculated by sum-of-squares error. One of the most frequently used forms of this stress metric is:

$$\boldsymbol{Stress} = \frac{\sum_{i<j} (\boldsymbol{d}_{ij} - \boldsymbol{d}'_{ij})^2}{\sum_{i<j} (\boldsymbol{d}'_{ij})^2}. \tag{1.9}$$

The popularity of MDS has led to the proposal of variants such as SPE (Agrafiotis, 2003), CCA (Demartines et Hérault, 1997), SNE (Hinton, 2003), and FastMap (Faloutsos et Lin, 1995).

1.1.1.5 Isometric Mapping

Isometric Mapping (Isomap) assumes that the data lie on a low-dimensional manifold embedded in a high-dimensional ambient space (Balasubramanian et Schwartz, 2002; Zhang *et al.*, 2012). It seeks to recover a low-dimensional representation by preserving pairwise geodesic distances,i.e., the shortest path lengths between points along the manifold, thereby respecting the intrinsic geometry of the underlying structure. To achieve this, Isomap first constructs a neighborhood graph (typically using k-nearest neighbors or an ϵ- ball criterion), computes approximate geodesic distances between all pairs of points via shortest-path algorithms (e.g., Dijkstra's or Floyd–Warshall), and then applies classical MDS to these geodesic distances to obtain the final embedding. Consequently, Isomap can be viewed as a nonlinear extension of classical MDS that incorporates manifold geometry. The main steps of Isomap are as follows:

Step 1: Build the local neighborhoods around each data point. First for the dataset $X = \{x_1, x_2, \ldots, x_n\}$, calculating the Euclidean distance $d_x(x_i, x_j)$ of any two sample vectors x_i and x_j. Comparing each point with all of the others. They are adjacent when the distance between two points is less than the fixed radius ϵ, then connect them. The length of the side is $d_x(x_i, x_j)$, and the neighborhood graph G is obtained.

Step 2: Compute the shortest distance. In graph G, the shortest distance between any two sample vectors x_i and x_j is $d_G(x_i, x_j)$. If there is a connection between x_i and x_j, then the initial value of $d_G(x_i, x_j)$ is $d_x(x_i, x_j)$, otherwise $d_G(x_i, x_j) = \infty$. For $k = 1, 2, \ldots, n$, there is:

$$d_G(x_i, x_j) = \min \left\{ d_G(x_i, x_j),\ d_G(x_i, x_k) + d_G(x_k, x_j) \right\}, \tag{1.10}$$

so we can get the matrix $D_G = \{d_G(x_i, x_j)\}$. It is composed of the shortest path of all pairs of points in graph G.

Step 3: Construct a d- dimensional embedding. Constructing a d- dimensional which maintains the feature geometry embedded in the space Y with the MDS method:

$$\boldsymbol{\tau}(\mathbf{D}_G) = -\frac{\mathbf{H} \times (\mathbf{D}_G)^2 \times \mathbf{H}}{2}. \tag{1.11}$$

H is is the unit matrix and is in the same order with $\mathrm{D_G}$. Perform eigen decomposition on $\tau(\mathrm{D_G})$, taking the largest d eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_d$ and corresponding eigenvectors $\mathrm{V}_1, \mathrm{V}_2, \ldots, \mathrm{V}_d$. Let V_p^i be the ith component of the pth feature vector, and the corresponding low- dimensional data represents as $y_i = \lambda_p^{1/2} V_p^i$.

Isomap has demonstrated effectiveness in detecting irregularities in real-time video analytics applications (Yang *et al.*, 2016). The method inherits key advantages from both PCA and MDS such as computational efficiency, asymptotic convergence guarantees, and global optimality while extending these benefits to a broad range of nonlinear manifolds (Nanga *et al.*, 2021). An important weakness of the Isomap algorithm is its topological instability (Balasubramanian et Schwartz, 2002).

1.1.1.6 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a well-established supervised linear dimensionality reduction (LDR) technique, originally introduced by Ronald A. Fisher in his seminal 1936 paper "The Use of Multiple Measurements in Taxonomic Problems", where it was successfully applied to flower classification (McLachlan, 2005). As a representative example of linear methods, the main objective of LDA is to project the original data samples onto an optimal discriminant subspace that maximizes class separability. This projection aims to extract relevant classification information while reducing dimensionality, ensuring that the projected samples exhibit the greatest possible distance between classes and the smallest possible distance within each class that is, maximizing the between-class scatter matrix while minimizing the within-class scatter matrix. However, LDA captures only the global geometric structure of the data and does not account for local geometric variations among samples within the same class. LDA has been widely applied in diverse domains, including facial recognition (Park et Park, 2008), text recognition (Yu et Yang, 2001; Park et Park, 2008), and the automatic diagnosis of mechanical systems (Kedadouche *et al.*, 2016). The aim of LDA is to find a linear transformation matrix $\mathrm{W} \in \mathbb{R}^{m \times k}$ $(k \ll p)$ to map high dimensional data $\mathrm{Y} \in \mathbb{R}^{m \times k}$ into

lower dimensional data $\mathrm{Z} \in \mathbb{R}^{m \times k}$:

$$\mathbf{Z} = \mathbf{W}^T \mathbf{Y}. \tag{1.12}$$

Optimal projection matrix can be computed using this equation :

$$\min \mathrm{Tr}\left(\frac{\mathbf{W}^T \mathbf{S}_w \mathbf{W}}{\mathbf{W}^T \mathbf{S}_t \mathbf{W}}\right), \tag{1.13}$$

where $Tr(.)$ is the trace of the matrix. $\mathbf{S}_w$ and $\mathbf{S}_t$ can be computed as:

$$\mathbf{S}_w = \frac{1}{m} \sum_{i=1}^{c} m_i \left(\bar{\mathbf{y}}_i - \bar{\mathbf{y}}\right) \left(\bar{\mathbf{y}}_i - \bar{\mathbf{y}}\right)^T. \tag{1.14}$$

$$\mathbf{S}_t = \frac{1}{m} \sum_{i=1}^{c} \sum_{j=1}^{m_i} \left(\mathbf{y}_{ij} - \bar{\mathbf{y}}_i\right) \left(\mathbf{y}_{ij} - \bar{\mathbf{y}}_i\right)^T, \tag{1.15}$$

where $\bar{\mathrm{y}}$ is the mean of all the samples.

1.1.1.7 Singular Value Decomposition

Singular Value Decomposition (SVD) is an unsupervised LDR that is closely related to PCA. SVD have been used in different areas by researchers. These include the area of digital image processing (Cao, 2006), taxonomic classification of biological sequences, Natural Language Processing and bio-informatics(Santos *et al.*, 2011). SVD is developed for matrix decomposition. It factorizes matrix $\mathbf{Y} \in \mathbb{R}^{m \times p}$ into $\mathbf{U}\mathbf{S}\mathbf{V}^T$. Matrix $\mathbf{U}$ and $\mathbf{V}$ are two orthogonal matrices having dimensions $m \times k$ and $p \times k$, respectively. Matrix $\mathbf{S}$ is a $k \times k$ diagonal matrix containing singular values of $\mathbf{Y}$. Number of singular values obtained is k (Strang, 2022). One important feature of SVD is to reconstruct original matrix Y using matrices $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{S}$. SVD is suitable for a wide range of eigenvector analysis problems. Chen *et al.* (2018) proposed a clustering method based on dimensionality reduction using the daily load curve and SVD . This approach effectively addressed the challenges associated with the large volume of historical load curve data in terms of storage and computation, resulting in reduced processing time and improved clustering accuracy. One drawback of SVD is that it is expensive computationally. It can however be improved when random sampling is applied.

1.1.1.8 t-Stochastic Neighbor Embedding

t-Stochastic Neighbor Embedding(t-SNE) introduced by Maaten et Hinton (2008) is a powerful unsupervised NLDR technique. It is a non-parametric method that is particularly well suited for the visualization of

high-dimensional datasets exhibiting complex nonlinear structures. By preserving local neighborhood relationships, t-SNE reveals meaningful low-dimensional patterns while also capturing certain aspects of the global organization of the data, making it highly effective for manifold learning tasks.

Owing to its strong ability to uncover clusters and latent structure in complex data, t-SNE has become one of the most widely used visualization techniques in single-cell genomics and other high-throughput biological analyses. To extend its applicability, Gisbrecht *et al.* (2015) proposed a kernel-based variant of t-SNE that preserves the core strengths of the original method while enabling parametric modeling. Furthermore, Xie *et al.* (2011) introduced a multi-view Stochastic Neighbor Embedding (m-SNE) framework to systematically integrate structural information from multiple data sources into a unified representation for subsequent processing. Experimental results demonstrated the effectiveness of m-SNE for scene recognition and data visualization. Despite its advantages, t-SNE also presents notable limitations, including high computational cost, limited scalability to very large datasets, and a tendency to distort large-scale (global) data structures (Xie *et al.*, 2011).

1.1.1.9 Uniform Manifold Approximation and Projection

Uniform manifold approximation and projection (UMAP) is an unsupervised NLDR proposed by McInnes *et al.* (2018). Rather than prioritizing the preservation of large-scale global structure, UMAP primarily focuses on preserving the local neighborhood relationships within a dataset. The core principle of UMAP is the construction of fuzzy topological representations of both the high-dimensional data and their low-dimensional embeddings, followed by the optimization of the embedding such that its fuzzy topological structure closely matches that of the original high-dimensional space (Ghojogh *et al.*, 2021). UMAP has been widely adopted for the visualization and feature extraction of DNA and single-cell datasets (Becht *et al.*, 2019). In practice, UMAP has been shown to compete favorably with t-SNE, which is widely regarded as a robust reference method for dimensionality reduction–based visualization.

Unlike t-SNE, UMAP does not impose strict computational limitations on the embedding dimensionality, making it suitable for both visualization and general-purpose dimensionality reduction. Although UMAP is conceptually similar to t-SNE, it typically offers faster computation and produces clearer global structure in the resulting embeddings. One limitation of UMAP is that, as a relatively recent method, it still lacks the same level of theoretical and empirical maturity as more established techniques. Nevertheless, UMAP has demonstrated strong performance in practical applications.

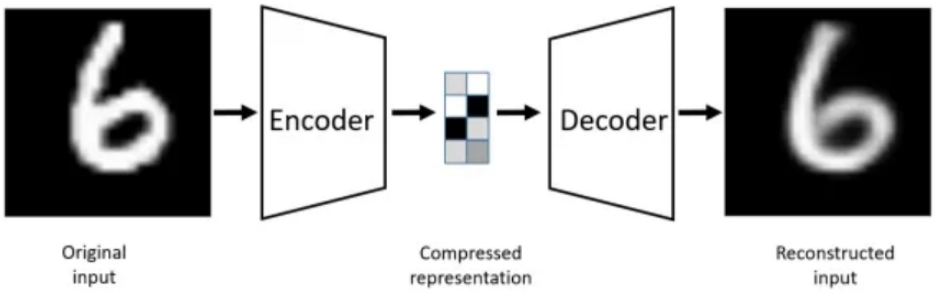

Fig. 1.1 General structure of an autoencoder (Bank *et al.*, 2023).

1.1.2 Deep learning-based methods:

Autoencoders (AEs) were first introduced by (Rumelhart *et al.*, 1986) as neural networks trained to reconstruct their input data. Their primary objective is to learn, in an unsupervised manner, an informative latent representation that can be exploited for various downstream tasks, including clustering (Bank *et al.*, 2023). To better understand AEs, it is helpful to examine their typical architecture, illustrated in Figure 1.1. An AE consists of two successive stages: first, the data are compressed and encoded into a lower-dimensional latent representation, and then they are reconstructed from this compressed form. The goal is to ensure that the reconstructed output is as close as possible to the original input. This process is achieved through two main components: the encoder, which maps the input data to a reduced latent space, and the decoder, which reconstructs the original data from this latent representation.

By design, an AE inherently performs dimensionality reduction while simultaneously learning to filter out noise and preserve the most relevant information. This dual capability makes AEs particularly well suited for applications such as data denoising, dimensionality reduction, and feature extraction, where it is essential to retain the most informative features while suppressing irrelevant or noisy components. The problem, as formally defined in (Baldi, 2012) is to learn the functions:

$$\mathbf{A} : \mathbb{R}^n \to \mathbb{R}^p \quad \text{(encoder)},$$

and

$$\mathbf{B} : \mathbb{R}^p \to \mathbb{R}^n \quad \text{(decoder)},$$

that satisfy:

$$\arg\min_{\mathbf{A},\,\mathbf{B}} \mathbb{E}[\boldsymbol{\Delta}(\boldsymbol{x},\, \mathbf{B} \circ \mathbf{A}(\boldsymbol{x}))]\,, \tag{1.16}$$

where **E** is the expectation over the the distribution of x, and Δ is the reconstruction loss function, which measures the distance between the output of the decoder and the input. In the most popular form of AEs, **A** and **B** are neural networks (Ranzato *et al.*, 2007).

AEs excel at dimensionality reduction by learning compact and efficient representations of data. They compress high-dimensional data into lower-dimensional forms, effectively reducing the number of features. This simplification makes the data easier to visualize, analyze, and interpret. However the performance of AEs is highly sensitive to the selection of hyperparameters, such as the number and size of layers, learning rate, activation functions, and the type of loss function used. Finding the optimal combination of these hyperparameters often requires extensive experimentation and fine-tuning, which can be time-consuming and requires expertise in machine learning. Figure 1.1 (taken from Bank *et al.* (2023)) provides an illustration of the autoencoder model.

Regularized autoencoders overcome the limitations of undercomplete autoencoders by incorporating regularization techniques. These techniques constrain or modify the model's calculation of reconstruction error, effectively reducing overfitting and enabling the autoencoder to learn useful features or functions. Regularization not only prevents the model from simply copying the input but also ensures that it captures the most essential and informative aspects of the data.

1.1.2.1 Denoising Autoencoders

Denoising autoencoders (DAEs) (Vincent *et al.*, 2008) are designed to handle partially corrupted input data and are trained to restore the original input by removing extraneous noise through dimensionality reduction. In these architectures, the input data is intentionally corrupted by noise (such as additive white Gaussian noise or using techniques like Dropout), and the autoencoder is trained to reconstruct the original, clean version of the input, as shown in in Figure 1.2.

Note that $\tilde{x}$ is a random variable, whose distribution is given by $\mathbf{C}(\tilde{\mathbf{x}} \,|\, \mathbf{x})$. Two common options for **C** are:

$$\mathbf{C}_\sigma(\tilde{\boldsymbol{x}} \,|\, \boldsymbol{x}) = \boldsymbol{\mathcal{N}}\big(\boldsymbol{x},\, \sigma^2 \boldsymbol{\mathcal{I}}\big)\,. \tag{1.17}$$

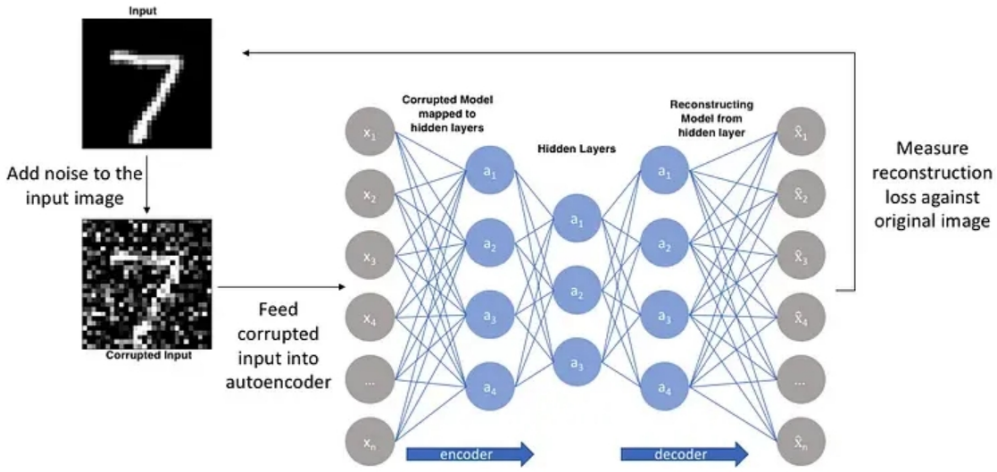

Fig. 1.2 A denoising autoencoder example (source : `https://trendspider.com/learning-center/autoencoders/`).

and

$$\mathbf{C}_p(\tilde{\boldsymbol{x}} \,|\, \boldsymbol{x}) = \boldsymbol{\beta} \odot \boldsymbol{x}, \quad \boldsymbol{\beta} \sim \mathrm{Ber}(p), \tag{1.18}$$

where $\odot$ denotes an element-wise product. The variance parameter σ sets the impact of the noise. After, the parameter p sets the probability of a value in x not being nullified. Denoising autoencoders are particularly useful for cleaning up noisy or corrupted image and audio files. They serve as foundational training paradigms for state-of-the-art image generation models, such as those used in Stable Diffusion, where they contribute to the ability to generate high-quality, noise-free images.

Wang *et al*. (2020) apply DAEs to gene expression data to identify molecular biomarkers of lung adenocarcinoma. The DAE effectively removes noise and extracts latent features from high-dimensional biological data, improving classification accuracy. It demonstrates the power of DAEs to reveal hidden patterns in noisy genomic datasets. In Wang *et al*. (2023), the authors design a dual DAE architecture to process single-cell RNA sequencing (scRNA-seq) data, reducing dimensionality while preserving biological structure.

1.1.2.2 Variational Autoencoders

The Variational Autoencoder (VAE) was initially proposed by Kingma et Welling (2013a) and Rezende *et al*. (2014). The inputs get mapped onto a distribution in latent space, rather than a single point. Kingma et Welling (2013a) map each input onto a multivariate Gaussian distribution with no correlations. The latent

layer is replaced with two sets of neurons: one representing the means in each of the dimensions of latent space, the other representing the log variances (log variances are used to guarantee that the variances are positive). We will refer to these values as the latent means and latent log variances. Figure 1.3 shows a VAE architecture. The latent means and latent variances each have their own weights and biases (Portillo *et al.*, 2020) :

$$\boldsymbol{\mu} = \mathbf{W}^{(\mu)}\mathbf{e} + \mathbf{b}^{(\mu)}, \tag{1.19}$$

$$\ln \boldsymbol{\sigma}^2 = \mathbf{W}^{(\ln \sigma^2)}\mathbf{e} + \mathbf{b}^{(\ln \sigma^2)}, \tag{1.20}$$

and a point in latent space is sampled from $q(\mathbf{z}|\mathbf{x})$, the Gaussian with these latent means and variances:

$$\mathbf{z} \sim q(\mathbf{z}|\mathbf{x}) \equiv \mathcal{N}\big(\boldsymbol{\mu}, \operatorname{diag}\left(\exp(\ln \boldsymbol{\sigma}^2)\right)\big)\,. \tag{1.21}$$

This Gaussian can be thought of as a distribution of latent representations that are consistent with the input. Instead of propagating this distribution forward, the decoder samples a point from it and propagates it:

$$\mathbf{d} = f(W^{(d)}\mathbf{z} + \mathbf{b}^{(d)}). \tag{1.22}$$

The objective function used is the evidence lower bound (ELBO),which is the sum of the reconstruction loss and the Kullback–Leibler (KL) divergence (Kullback et Leibler, 1951) between the latent distribution for the input $q(\mathbf{z}|\mathbf{x})$ and the prior $p(\mathbf{z})$:

$$\mathrm{ELBO} = \mathcal{L}(\mathbf{x}, \mathbf{x}') + D_{\mathrm{KL}}\big(q(\mathbf{z}|\mathbf{x}) \,\|\, p(\mathbf{z})\big). \tag{1.23}$$

The KL divergence between two probability distributions p and q is a measure of the difference between the two distributions and is defined as:

$$D_{\mathrm{KL}}(q\|p) = \int q(\mathbf{z}) \log\left(\frac{q(\mathbf{z})}{p(\mathbf{z})}\right) d\mathbf{z}. \tag{1.24}$$

In Gomari *et al.* (2022), the authors apply Variational Autoencoders (VAEs) to large-scale metabolomics datasets comprising more than 4,500 individuals to extract biologically meaningful and transferable features. The learned latent representations capture complex nonlinear relationships and outperform PCA and Kernel PCA in identifying disease-related patterns. A key advantage of VAEs lies in their ability to learn continuous and structured latent spaces in which similar data points are encoded in close proximity (Kingma

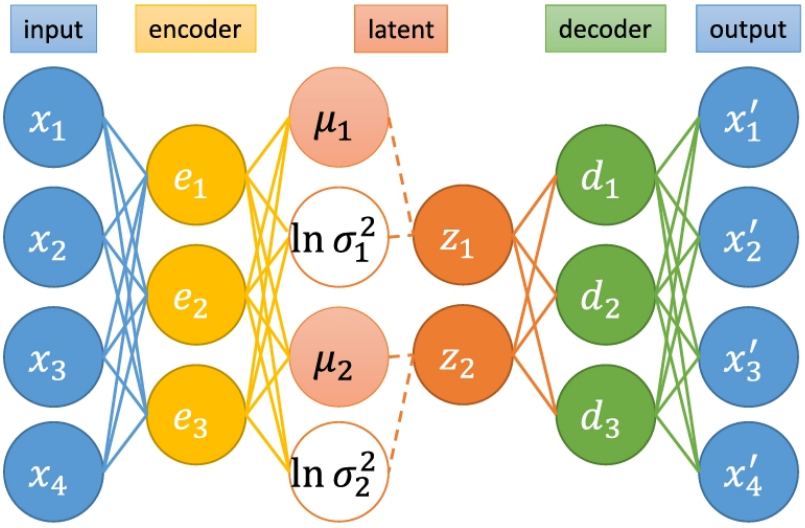

Fig. 1.3 Structure of a variational autoencodernwith a 4-3-2-3-4 architecture. The dashed lines indicate that each latent value $z_{1,2}$is drawn from a Gaussian distribution defined by a latent mean $\mu_{1,2}$ and log variance $\ln \sigma_{1,2}^2$ (Portillo *et al.*, 2020).

et Welling, 2013a). This property makes VAEs particularly effective for nonlinear dimensionality reduction and downstream tasks such as clustering, as the latent space tends to preserve meaningful underlying data structure. Despite these strengths, VAEs also exhibit several limitations. First, they often produce blurrier or less detailed reconstructions than Generative Adversarial Networks (GANs), largely because the VAE objective maximizes a variational lower bound on the data likelihood rather than directly optimizing pixel-level fidelity (Dosovitskiy et Brox, 2016). Second, VAEs are subject to an inherent trade-off between reconstruction accuracy and latent space regularization: when the KL-divergence term dominates the objective, the encoder may disregard the latent variables, leading to a failure mode known as posterior collapse (Lucas *et al.*, 2019).

1.2 Clustering methods

Clustering is a common technique for statistical data analysis, which is used in many fields, including machine learning, data mining, pattern recognition, image analysis and bioinformatics (Makarenkov et Leclerc, 1996, 2000; Tahiri *et al.*, 2018, 2022; Ghosal *et al.*, 2020; Oyewole et Thopil, 2023). Clustering is the process of grouping similar objects into different groups, or more precisely, the partitioning of a data set into subsets, so that the data in each subset according to some defined distance measure (Madhulatha, 2012). The main clustering approaches include centroid-based, probabilistic, hierarchical, graph-based, and density-based clustering (Wani, 2024).

1.2.1 Partitioning (Centroid-based) Clustering

This technique involves dividing a dataset into several disjoint partitions or clusters. Each data point is assigned to a specific cluster based on certain criteria

1.2.1.1 K-means

Centroid-based clustering, a prominent class of partitioning methods, organizes data points into clusters based on their proximity to representative centroids (Wani, 2024). K-means clustering is one of the most widely used techniques in multivariate data analysis, particularly within machine learning, data mining, and quantitative psychology (Rykov *et al.*, 2024). K-means algorithm is used as an iterative clustering algorithm. It takes the distance as the measurement standard, gives the K clusters in the data set, calculates the average value of the distance, and then gives the initial centroid. Each cluster is described by the centroid (Yuan et Yang, 2019). For a given data set X containing n multidimensional data points and the category K to be divided, the Euclidean distance is selected as the similarity index and the clustering targets minimize the sum of the squares of the various types, that is, it minimizes (Wang *et al.*, 2012) :

$$d = \sum_{k=1}^{K} \sum_{i=1}^{n} \|(x_i - u_k)\|^2, \tag{1.25}$$

where K represents K cluster centers, u_k represents the kth center, and x_i represents the ith point in the data set. The solution to the centroid u_k is as follows:

$$\begin{aligned} \frac{\partial}{\partial u_k} &= \frac{\partial}{\partial u_k} \sum_{k=1}^{K} \sum_{i=1}^{n} (x_i - u_k)^2 \\ &= \sum_{k=1}^{K} \sum_{i=1}^{n} \frac{\partial}{\partial u_k} (x_i - u_k)^2 \\ &= \sum_{i=1}^{n} 2(x_i - u_k). \end{aligned} \tag{1.26}$$

If the value given by Equation (1.26) is zero, then:

$$u_k = \frac{1}{n} \sum_{i=1}^{n} x_i.$$

The central idea of algorithm implementation is to randomly extract K sample points from the sample set as the center of the initial cluster: Divide each sample point into the cluster represented by the nearest center point, then the center point of all sample points in each cluster is the center point of the cluster. Repeat the above steps until the center point of the cluster is unchanged or reaches the set number of

iterations. Despite k-means clustering's guaranteed convergence, it often falls into local minima due to its reliance on random centroid initialization, classifying it as a greedy algorithm. This can result in suboptimal clustering solutions or increased convergence times. A common strategy to address this issue is to run k-means multiple times with different initializations and select the solution with the lowest within-cluster sum of squares (WCSS), which helps in finding a better global optimum. Several advanced techniques have been developed to mitigate the local minima problem in k-means clustering. These include:

- **Repeated random initializations:** Running k-means multiple times with different random starting points and choosing the best result (Fränti et Sieranoja, 2019).
- **k-means++:** Strategically initializes centroids to ensure better initial separation. This approach improves convergence speed and reduces the likelihood of poor local optima (Jain, 2010; Steinley, 2006).

1.2.2 Hierarchical Clustering

Hierarchical clustering is a popular algorithm used in unsupervised machine learning for dividing a dataset into a hierarchy of clusters. It aims to organize the data points in a tree-like structure called a dendrogram, which represents the relationships between the points based on their similarities or dissimilarities.

1.2.2.1 Agglomerative Hierarchical Clustering

The Agglomerative Hierarchical Clustering (AHC) is the most common type of hierarchical clustering method used in grouping objects in clusters based on their similarity (Everitt *et al.*, 2011). Agglomerative clustering algorithms operate in a bottom-up manner. They begin by treating each data point $y_i \in Y$ as an individual cluster (singleton). The algorithm then iteratively merges the closest pairs of clusters, step by step, until all points are grouped into a single cluster or until a predefined number of clusters is reached, while the divisive approach takes the opposite strategy by following a top-down approach (De Amorim *et al.*, 2016). The similarity between data points is quantified using distance measures such as Euclidean, Manhattan similarity etc (Jain et Dubes, 1988; Friedman, 2009). The merging process employs various linkage criteria to recalculate distances between clusters. If x and y are two data points in an n-dimensional space. Mathematically:

$$\text{Distance} = \begin{cases} \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}, & \text{(Euclidean Distance)} \\ \sum_{i=1}^{n}|x_i - y_i|, & \text{(Manhattan Distance)}. \end{cases} \tag{1.27}$$

In AHC, the linkage criterion is crucial as it determines how distances between clusters are calculated, which in turn affects cluster assignments and the overall outcome of the clustering process. There are primarily five types of linkage criteria: single, complete, average, centroid linkage, and Ward's method (Schubert *et al.*, 2017).

- **Single Linkage:** The distance between two clusters is defined as the shortest distance between any two points (one in each cluster). Tends to form long "chain-like" clusters and is sensitive to noise and outliers.
- **Complete Linkage:** Uses the largest distance between any two points in the two clusters. Produces more compact, spherical clusters but can break large natural clusters.
- **Average Linkage:** Computes the average distance between all pairs of points in the two clusters. Offers a balance between single and complete linkage and is less sensitive to noise.
- **Centroid Linkage:** Defines the distance between clusters as the distance between their centroids (mean vectors). Can lead to cluster inversions (i.e., non-monotonic dendrograms).
- **Ward's Linkage:** Merges clusters that result in the smallest increase in total within-cluster variance (minimizes the error sum of squares). Generally produces compact, well-separated, and balanced clusters.

1.2.2.2 Birch

Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) algorithm is an integrated hierarchical clustering algorithm. It uses the clustering features (CF) and cluster feature tree (CF Tree), two concepts for the general cluster description (Ramadhani *et al.*, 2020). Clustering feature tree outlines the clustering of useful information, and space is much smaller than the meta-data collection can be stored in memory, which can improve the algorithm in clustering large data sets on the speed and scalability.

Before implementing Birch, we must understand two important terms: CF and CF Tree : BIRCH summarizes large datasets into smaller, dense regions called CF entries. Formally, a CF entry is defined as an ordered triple - (N, LS, SS), where N is the number of data points in the cluster, LS is the linear sum of the data points, and SS is the squared sum of the data points in the cluster. It is possible for a CF entry to be composed of other CF entries. CF Tree: The CF tree is the actual compact representation that we have been speaking of so far. A CF tree is a tree where each leaf node contains a sub-cluster. Every entry in a CF tree contains a pointer to a child node and a CF entry made up of the sum of CF entries in the child nodes. There is a maximum number of entries in each leaf node. This maximum number is called the threshold. We will learn more about what this threshold value is. Parameters of BIRCH algorithm :

- **threshold:** threshold is the maximum number of data points a sub-cluster in the leaf node of the CF tree can hold.
- **branching_factor:** This parameter specifies the maximum number of CF sub-clusters in each node (internal node).
- **n_clusters:** The number of clusters to be returned after the entire BIRCH algorithm is complete i.e., number of clusters after the final clustering step. If set to None, the final clustering step is not performed and intermediate clusters are returned.

Birch algorithm is a clustering algorithm suitable for very large data sets. Zheng *et al.* (2008) used BIRCH to cluster GPS trajectories and study user mobility behaviors. BIRCH efficiently groups large-scale geospatial data and helps extract meaningful urban mobility patterns.

1.2.3 Density-based Clustering

Density-based clustering is a technique used to discover clusters in a data collection based on the density of data points in their vicinity. It identifies areas with a higher density of data points compared to the surrounding areas. The clusters are defined as regions of high-density separated by regions of low-density. The points within a cluster are uniformly dense, and regions of low-density separate points in different clusters.

1.2.3.1 Dbscan

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is the first density based clustering algorithm. It was proposed by Ester *et al*. (1996). DBSCAN was designed to cluster data of arbitrary shapes in the presence of noise in spatial and non-spatial high dimensional databases. The key idea of DBSCAN is that for each object of a cluster the neighborhood of a given radius $\boldsymbol{Eps}$ has to contain at least a minimum number of objects $\boldsymbol{MinPts}$, which means that the cardinality of the neighborhood has to exceed some threshold (Khan *et al.*, 2014). The ϵ-neighborhood of an arbitrary point p is defined as:

$$\mathbf{N}_{\varepsilon}(\boldsymbol{p}) = \{\, \boldsymbol{q} \in \mathbf{D} \mid \operatorname{dist}(\boldsymbol{p}, \boldsymbol{q}) < \operatorname{Eps} \,\}. \tag{1.28}$$

Here, **D** is the database of objects. If the ϵ-neighborhoods of a point P at least contain a minimal number of points, and then this point is called core point. The core point is defined as:

$$\left| \mathbf{N}_{\varepsilon}(\boldsymbol{P}) \right| > \operatorname{MinPts}. \tag{1.29}$$

Here $\boldsymbol{Eps}$ and $\boldsymbol{MinPts}$ are the user's specified parameters which mean the radius of the neighborhood and minimum number of points in the ϵ-neighborhood of a core point respectively. If this condition is not satisfied then this point is considered as non-core point. DBSACN searches for the clusters by checking the ϵ-neighborhood of each object in the dataset. If the ϵ-neighborhood of an object p contains more than $\boldsymbol{MinPts}$, a new cluster with p as a core object is created. It then iteratively collects directly density-reachable objects from these core objects, which may involve the merge of a new density-reachable cluster. The process terminates when no new object can be added to any cluster (Agrawal *et al.*, 1998).

1.2.3.2 OPTICS

Ordering Points To Identify the Clustering Structure (OPTICS) is the Density Based clustering by creating an ordering of the points that allows the extraction of clusters with arbitrary values for ϵ. The parameter ϵ is a distance, it is the neighborhood radius (Kanagala et Krishnaiah, 2016). Unlike DBSCAN, which struggles with varying densities, OPTICS does not directly assign clusters but instead creates a reachability plot which visually represents clusters. The key concepts in OPTICS are:

- **Core Distance:** The minimum distance needed for a point to be classified as a core point. If a point does not have enough nearby neighbours, its core distance is undefined.

- **Reachability Distance:** It is a measure of how difficult it is to reach from one point to another. It is calculated as the larger core distance of the starting point and the actual point.

1.2.4 Probabilistic (Model-based) Clustering

Model-based clustering takes a statistical approach. It assumes that the data points are drawn from a finite combination of component models. Each of these component models are probability distribution. It then finds the specific mixture and its parameters that best fit the data.

1.2.4.1 Gaussian Mixture Model

Gaussian Mixture Models (GMMs) are probabilistic models that assume all the data points are generated from a mixture of several Gaussian distributions with unknown parameters (Wolfe, 1970). In a GMM, each Gaussian component represents a cluster within the data. The overall density of the data is modeled as a weighted sum of these Gaussian densities. Each cluster corresponds to a Gaussian distribution. For a given data point x_n of belonging to a cluster. GMM computes the probability it belongs to each cluster k :

$$\boldsymbol{P}(z_n = k \mid \boldsymbol{x}_n) = \frac{\boldsymbol{\pi}_k \cdot \boldsymbol{\mathcal{N}}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k=1}^{K} \boldsymbol{\pi}_k \cdot \boldsymbol{\mathcal{N}}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}, \tag{1.30}$$

where:

- $z_n = k$ is a latent variable indicating which Gaussian the point belongs to.
- π_k is the mixing probability of the k-th Gaussian.
- $\mathcal{N}(x_n \mid \mu_k, \Sigma_k)$ is the Gaussian distribution with mean μ_k and covariance Σ_k.

Next we need to calculate the overall likelihood of observing a data point x_n under all Gaussians. This is achieved by summing over all possible clusters (Gaussians) for each point :

$$P(x_n) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n \mid \mu_k, \Sigma_k), \tag{1.31}$$

where:

- $P(x_n)$ is the overall likelihood of observing the data point x_n.

- The sum accounts for all possible Gaussians k.

To fit a Gaussian Mixture Model to the data, we use the Expectation-Maximization (EM) algorithm that is an iterative method that optimizes the parameters of the Gaussian distributions such as mean, covariance and mixing coefficients. It works in two main steps:

- **Expectation Step (E-step):** In this step the algorithm calculates the probability that each data point belongs to each cluster based on the current parameter estimates (mean, covariance, mixing coefficients).
- **Maximization Step (M-step):** After estimating the probabilities the algorithm updates the parameters (mean, covariance and mixing coefficients) to better fit the data.

These two steps are repeated until the model converges meaning the parameters no longer change significantly between iterations. Here's a simple breakdown of the GMM process:

1. **Initialization:** Start with initial guesses for the means, covariances and mixing coefficients of each Gaussian distribution.
2. **E-step:** For each data point, calculate the probability of it belonging to each Gaussian distribution (cluster).
3. **M-step:** Update the parameters (means, covariances, mixing coefficients) using the probabilities calculated in the E-step.
4. **Repeat:** Continue alternating between the E-step and M-step until the log-likelihood of the data (a measure of how well the model fits the data) converges.

Formula:

$$L(\mu_k, \Sigma_k, \pi_k) = \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n \mid \mu_k, \Sigma_k). \tag{1.32}$$

The E-step computes the probabilities that each data point belongs to each Gaussian while the M-step updates the parameters μ_k, Σ_k, and π_k based on these probabilities. GMMs are widely applied across domains such as speech recognition and speaker identification (Reynolds et Rose, 2002), image segmentation

in computer vision (Permuter *et al.*, 2006), anomaly detection in cybersecurity and industrial monitoring (Ahmed *et al.*, 2016; Zong *et al.*, 2018). Their ability to model multimodal and complex distributions makes GMM a versatile tool in both supervised and unsupervised learning tasks.

1.2.5 Fuzzy Clustering

Fuzzy clustering is a technique used in data analysis and data mining to classify data points into multiple clusters based on their similarity. Unlike traditional clustering algorithms, fuzzy clustering assigns a degree of membership to each data point, indicating the likelihood of that point belonging to each cluster.

1.2.5.1 Fuzzy C-Means

Bezdek (2013) introduced Fuzzy C-Means clustering (FCM) method in 1981, extend from Hard C-Mean clustering method. FCM is an unsupervised clustering algorithm that is applied to wide range of problems connected with feature analysis, clustering and classifier design. FCM is widely applied in agricultural engineering, astronomy, chemistry, geology, image analysis, medical diagnosis, shape analysis and target recognition (Yong *et al.*, 2004). Unlike traditional clustering (hard clustering), where a point belongs to exactly one cluster, fuzzy clustering assigns a membership value between 0 and 1 for each point's participation in each cluster as shown in the figure 1.4. This allows more flexible grouping, especially when clusters overlap or data points are ambiguous.

Fuzzy clustering follows an iterative optimization process where data points are assigned membership values instead of hard cluster labels. Here's a step-by-step breakdown of how it works:

- **Step 1: Initialize Membership Values Randomly:** Every data point begins with a set of initial membership values, one for each cluster, indicating the degree or probability of its association with that cluster. In contrast to hard clustering where an item must belong entirely to just one group, this approach allows a point to share partial membership across multiple clusters at the same time.
- **Step 2: Compute Cluster Centroids:** Cluster centroids are computed as a weighted mean of all data points, with the weights determined by raising each point's membership value to the power of the fuzziness parameter m. Data points having higher membership degrees contribute more significantly to the position of the centroid.

The centroid coordinate v_{ij} for cluster i and feature j is:

$$v_{ij} = \frac{\sum_{k=1}^{n} \gamma_{ik}^{m} x_{kj}}{\sum_{k=1}^{n} \gamma_{ik}^{m}}, \tag{1.33}$$

where:

- γ_{ik} = membership of point k in cluster i.
- m = fuzziness parameter (usually $m = 2$).
- x_{kj} = value of feature j for point k.

- **Step 3: Calculate Distance Between Data Points and Centroids:** Calculate the Euclidean distance from each data point to all cluster centroids to assess their relative closeness; these distances will inform the subsequent update of membership degrees.

- **Step 4: Update Membership Values:** Membership values are adjusted based on the inverse of the distances points nearer to a centroid receive higher membership scores, while those farther away are assigned lower values.

  Updated membership γ_{ik} for point k in cluster i is:

$$\gamma_{ik} = \left[\sum_{j=1}^{n} \left(\frac{d_{ki}^2}{d_{kj}^2} \right)^{\frac{1}{m-1}} \right]^{-1}. \tag{1.34}$$

- **Step 5: Repeat Until Convergence:** Steps 2–4 are repeated until the membership values stabilize meaning there are no significant changes from one iteration to the next. This indicates that the clustering has reached an optimal state.

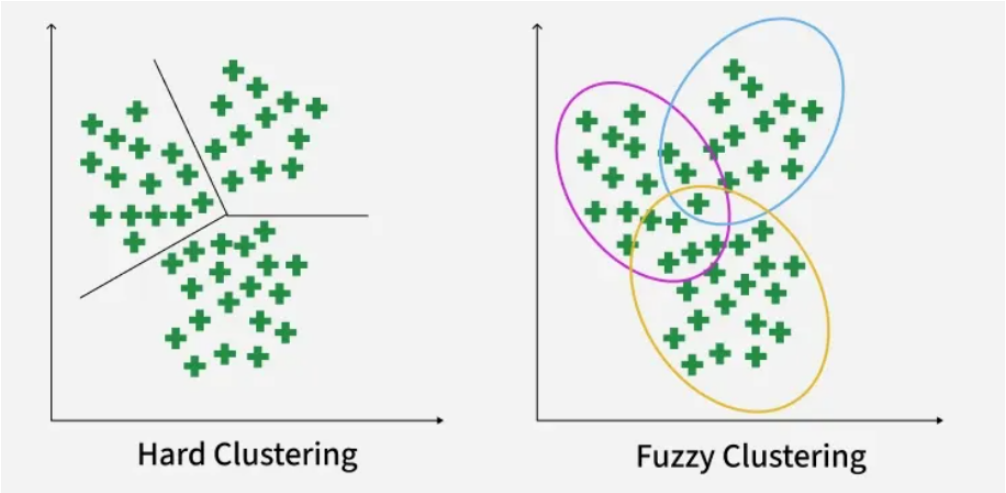

Fig. 1.4 Hard Clustering vs Fuzzy Clustering (source: `https://www.geeksforgeeks.org/machine-learning/ml-fuzzy-clustering/`).

1.3 Combination of dimensionality reduction and clustering

In high-dimensional settings, distance measures tend to lose contrast and become less informative, which reduces the effectiveness of similarity-based methods such as clustering and motivates applying dimensionality reduction before grouping the data (Herrmann *et al.*, 2024; Aggarwal *et al.*, 2001; Beyer *et al.*, 1999). Prior research has broadly explored DR$\rightarrow$clustering strategies across three directions: (i) using a fixed dimensionality-reduction method with a fixed clustering algorithm, (ii) holding the clustering method constant while varying the reduction technique, and (iii) fixing the reduction step and comparing different clustering algorithms. A widely studied pairing is PCA$\rightarrow k$-means, supported by theory showing that principal components align with relaxed cluster indicator vectors (Ding et He, 2004). Related probabilistic approaches incorporate a PCA-like latent space inside Gaussian mixture models, effectively performing joint dimensionality reduction and clustering (Tipping et Bishop, 1999). Within manifold learning, empirical papers commonly apply Isomap or MDS followed by k-means or GMM, often reporting performance improvements while still analyzing method pairs rather than systematic combinations (Tseng *et al.*, 2023).

When either the clustering method or the dimensionality-reduction (DR) step is held constant, numerous studies show that the DR choice substantially impacts downstream clustering quality. For instance, in the context of k-means, Alkhayrat *et al.* (2020) observed that latent representations learned by deep autoencoders consistently produced stronger internal validity metrics than PCA or raw data, with the retained latent dimensionality further influencing segmentation performance. Similarly, Hozumi *et al.* (2021) ex-

amined PCA, t-SNE, and UMAP across multiple datasets and reduction ratios, reporting large performance variations and noting that UMAP tended to provide more stable results as dimensionality changed. Going beyond k-means, Roh *et al.* (2025) fixed GMM while systematically varying DR methods (PCA, ICA, Isomap, LLE, t-SNE, UMAP), using external evaluation and generalization tests; their findings again emphasize that DR constitutes a critical design component in the clustering pipeline. Conversely, works that hold DR fixed and vary the clustering algorithm highlight heterogeneous algorithmic responses: using UMAP as preprocessing, Allaoui *et al.* (2020) documented improvements across k-means, agglomerative, GMM, and HDBSCAN, albeit with method-specific sensitivities. Complementing this evidence, Dalmaijer *et al.* (2022) conducted simulation experiments and found that k-means and agglomerative clustering perform similarly with or without DR, whereas HDBSCAN benefits substantially from UMAP embeddings, with detectability (via ARI and Silhouette) peaking in the low-dimensional UMAP space (Dalmaijer *et al.*, 2022).

Moreover generic DR$\rightarrow$clustering pipelines, applied studies frequently evaluate dimensionality reduction and clustering combinations within domain-specific workflows. For example, Lötsch et Ultsch (2024) systematically paired several DR techniques (PCA, ICA, Isomap, MDS, t-SNE and derivatives) with multiple clustering algorithms (k-means, k-medoids, and hierarchical variants including single, average, and Ward linkage) across synthetic and biomedical datasets, showing that the embedding step often plays a dominant role in determining clustering quality and that certain DR–clustering pairings yield consistently strong performance. Likewise, in reactive-flow physics, Rovira *et al.* (2022) adopted a structured three-stage pipeline nonlinear dimensionality reduction followed by unsupervised clustering and feature correlation to identify flow regimes and their governing mechanisms. In the single-cell and spatial transcriptomics domain, Sun *et al.* (2024) surveyed prevailing analytical workflows, noting widespread use of PCA-based graph embeddings with Louvain/Leiden clustering and manifold learning for visualization, shaped by modality-specific constraints. Overall, while these studies provide valuable empirical insight, they remain either pairwise or tightly domain-adapted, and exhibit substantial variation in datasets, reduction depths, and evaluation metrics.

Among these lines, three empirical regularities recur: (1) nonlinear/deep embeddings (e.g., UMAP, AE) often improve clustering stability and accuracy relative to linear PCA when intrinsic structure is nonlinear (Alkhayrat *et al.*, 2020; Hozumi *et al.*, 2021; Allaoui *et al.*, 2020; Dalmaijer *et al.*, 2022); (2) density-based methods (e.g., HDBSCAN) show the largest relative gains after manifold DR, while partition/hierarchical methods benefit more modestly (Dalmaijer *et al.*, 2022; Allaoui *et al.*, 2020); and (3) the reduction level

itself is consequential, with performance varying as the target dimension changes (Hozumi *et al.*, 2021). However, most studies evaluate only a limited number of DR $\rightarrow$ clustering algorithm pairs, focus on specific domains, or employ inconsistent settings that complicate cross-study comparisons.

1.4 Critical Synthesis of Dimensionality Reduction Methods

This section provides a structured critical synthesis of the dimensionality reduction methods reviewed in this chapter, namely PCA, Kernel PCA, Variational Autoencoders (VAEs), Isomap, and MDS. The comparison is conducted according to the three key criteria: (i) the ability to preserve global structure, (ii) sensitivity to noise, and (iii) computational complexity.

1.4.1 Preservation of Global Structure

PCA is well known for its strong ability to preserve the global variance of the data, making it effective for capturing large scale linear structures. However, it fails to model nonlinear relationships. Kernel PCA extends PCA to nonlinear settings by implicitly mapping data into a higher dimensional feature space, allowing it to capture more complex structures, although the preservation of global geometry depends heavily on kernel choice and parameter tuning.

Manifold learning methods such as Isomap and MDS aim to preserve the intrinsic geometry of the data. Isomap is particularly effective at preserving global geodesic distances, making it suitable for data lying on nonlinear manifolds. MDS, on the other hand, attempts to preserve pairwise distances and can capture global relationships but may struggle when the data is highly nonlinear or noisy. VAEs provide a flexible nonlinear representation through deep neural networks. While they can capture complex structures, their ability to preserve global relationships is not guaranteed and depends on the balance between reconstruction and regularization in the latent space.

1.4.2 Sensitivity to Noise

PCA is relatively robust to small amounts of noise due to its variance-maximizing nature, but it can be affected by outliers. Kernel PCA is more sensitive to noise, especially when inappropriate kernel parameters are used. Isomap is highly sensitive to noise, as it relies on nearest-neighbor graphs, which can be significantly distorted in noisy environments. Similarly, MDS can be affected by noise in distance measurements,

leading to distortions in the embedding. VAEs can be more robust to noise due to their probabilistic formulation, especially when regularization is properly applied. However, their performance depends strongly on training stability and hyperparameter tuning.

1.4.3 Computational Complexity

PCA is computationally efficient and scales well with large datasets, especially when using optimized singular value decomposition (SVD) algorithms. Kernel PCA has significantly higher computational complexity due to the computation and storage of the kernel matrix, typically scaling quadratically or cubically with the number of samples. Isomap and MDS are also computationally expensive, as they require the computation of pairwise distances and, in the case of Isomap, shortest paths on a graph. These methods are therefore less scalable to large datasets. VAEs require training deep neural networks, which can be computationally intensive and time consuming, particularly for high dimensional data and large datasets.

1.4.4 Comparative Summary

Table 1.1 Comparison of dimensionality reduction methods according to global structure preservation, noise sensitivity, and computational complexity

Method	**Global Structure**	**Noise Sensitivity**	**Computational Cost**
PCA	High (linear)	Low to moderate	Low
Kernel PCA	Moderate to high	High	High
Isomap	High (nonlinear)	High	High
MDS	Moderate	Moderate to high	High
VAE	Variable	Moderate	Very high

1.4.5 Research Gaps and Motivation

Despite their widespread use, these dimensionality reduction methods exhibit several limitations. First, their effectiveness is highly dependent on data characteristics such as dimensionality, noise level, and underlying structure. Second, there is no consensus on which method consistently improves clustering performance across different datasets. Third, nonlinear and deep methods often introduce significant computational overhead, which is rarely evaluated in comparative studies.

Furthermore, the interaction between dimensionality reduction and clustering algorithms remains insuf-

ficiently understood, particularly in terms of how different reduction techniques affect clustering quality under varying conditions. This work aims to address these gaps by systematically evaluating the impact of multiple dimensionality reduction techniques on clustering performance across a wide range of synthetic and real-world datasets, using consistent evaluation metrics and experimental settings.

1.5 Conclusion

In summary, this chapter reviewed the dimensionality reduction methods and clustering algorithms, as well as prior studies that combine these two approaches. The analysis highlighted that, although traditional linear techniques remain effective in many contexts, nonlinear and deep learning based methods increasingly enhance cluster separability. Nevertheless, the existing literature still lacks systematic and comprehensive comparisons of multiple dimensionality reduction clustering combinations across diverse application domains. These gaps directly motivate the experimental framework proposed in the next chapter, which aims to evaluate different analytical pipelines and to identify the most effective method combinations.

CHAPTER 2
METHODOLOGY

Chapter 2 of this thesis begins with the presentation of the data. Since the quality of model clustering depends on the data used to train them, particular attention must be paid to the creation of the datasets used for clustering. We then present the experimentation allowing us to evaluate the applicability of the different clustering algorithms as well as the dimensionality reduction methods in real and synthetic scenarios.

2.1 Data presentation

2.1.1 Generation of Synthetic datasets

To systematically evaluate the impact of dimensionality reduction on clustering performance, we constructed a diverse suite of synthetic datasets designed to capture a broad range of geometric structures, noise levels, and dimensionalities. These datasets support controlled benchmarking in scenarios that challenge clustering algorithms due to high-dimensional complexity, nonlinearity, and varying degrees of cluster overlap. We first employed the data generation method introduced by Rodriguez *et al.* (2019) to produce normally distributed synthetic data with systematic control over cluster separability and feature covariance. Hereafter, we refer to this as the Rodriguez Structured Gaussian (RSG) method. Each dataset was parameterized by the number of clusters $k \in \{2, 10, 50\}$, number of features $F \in \{10, 50, 200\}$, and number of samples per cluster $N_e \in \{5, 50, 100, 500, 5000\}$. The generator samples cluster-specific covariance matrices that are symmetric and positive semi-definite, ensuring realistic correlation structures between features. A mixing parameter α was tuned to ensure a range of cluster overlaps and complexity levels.

To simulate high-dimensional, anisotropic clusters with substantial inter-class separation, we employed the Repliclust method proposed by Zellinger et Bühlmann (2025). Two datasets were generated, one with two clusters and one with five clusters each in dimensions of 10, 50, and 200, containing 100 samples per cluster. In both cases, clusters were elongated along specific feature dimensions while maintaining centroid separation. This configuration provides a challenging benchmark for evaluating clustering robustness in high-dimensional settings. To assess performance on nonlinearly separable structures, we generated geometric datasets using the *make_moons* and *make_circles* functions from *scikit-learn*, supplemented by custom transformations to introduce structural diversity. The Moons dataset comprised a two-cluster crescent-

shaped structure with 20,000 samples. A five-cluster variant was produced by applying controlled stretching (scaling factors in [1.0,1.5]), rotations ($\pm$160°, $\pm$10°, and 180°), and translations (x-shifts of $\pm$2 to $\pm$4; y-shifts of 1.0, 1.2, and 1.5) to create crescent-like arrangements, each containing 2,000 samples per cluster. The Circles dataset consisted of a two-cluster nested circular structure with 20,000 samples, where the *factor* parameter was set to 0.5, ensuring the inner circle radius was half that of the outer circle. The five-cluster variant comprised concentric rings with 800 samples per cluster and radial scaling factors of 1.0, 2.0, 3.5, 5.0, and 7.0, ensuring distinct separation between rings. Unlike standard *make_circles*, only the circumference of each circle was retained to create ring-like structures. To enable high-dimensional evaluation, all 2D datasets were embedded into 10, 50, and 200 dimensions via Gaussian Random Projection, approximately preserving pairwise distances as per Frankl et Maehara (1988).

To assess performance on nonlinearly separable structures, we generated geometric datasets using the *make_moons* and *make_circles* functions from *scikit-learn*, supplemented by custom transformations to introduce structural diversity. The Moons dataset comprised a two-cluster crescent-shaped structure with 20,000 samples. A five-cluster variant was produced by applying controlled stretching (scaling factors in [1.0,1.5]), rotations ($\pm$160°, $\pm$10°, and 180°), and translations (x-shifts of $\pm$2 to $\pm$4; y-shifts of 1.0, 1.2, and 1.5) to create crescent-like arrangements, each containing 2,000 samples per cluster. The Circles dataset consisted of a two-cluster nested circular structure with 20,000 samples, where the *factor* parameter was set to 0.5, ensuring the inner circle radius was half that of the outer circle. The five-cluster variant comprised concentric rings with 800 samples per cluster and radial scaling factors of 1.0, 2.0, 3.5, 5.0, and 7.0, ensuring distinct separation between rings. Unlike standard *make_circles*, only the circumference of each circle was retained to create ring-like structures. To enable high-dimensional evaluation, all 2D datasets were embedded into 10, 50, and 200 dimensions via Gaussian Random Projection, approximately preserving pairwise distances as per Frankl et Maehara (1988).

To ensure realism in feature variability and robustness of clustering algorithms under noise, we applied a structured noise injection scheme. All features were first standardized using z-score normalization. We then added Gaussian noise to 75% of the features. One-third of the noisy features received standard Gaussian noise ($\mu = 0, \sigma = 1$), one-third received noise with $\sigma = 0.5$, and the final third with $\sigma = 0.25$. The remaining 25% of the features were left unperturbed. This design introduces heterogeneous noise distributions, enabling rigorous stress-testing of both clustering algorithms and dimensionality reduction techniques.

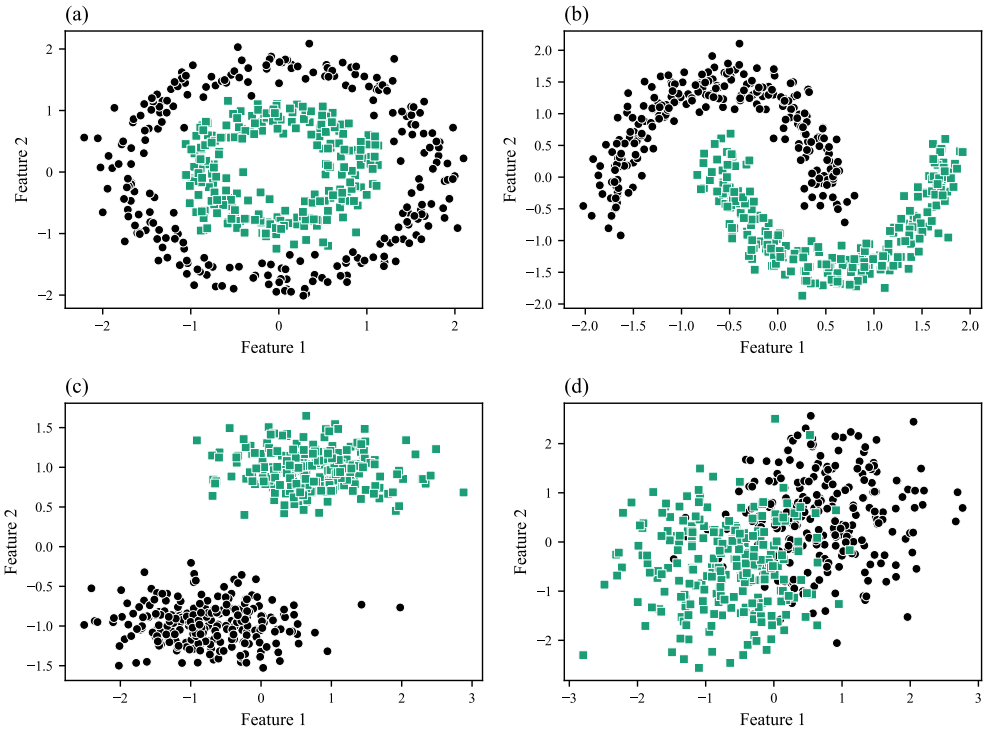

Fig. 2.1 Examples of synthetic datasets with 50 features and 2 clusters, visualized by projecting onto the first two principal components. The datasets correspond to the following generation methods: (a) Circles, (b) Moons, (c) Rodriguez Structured Gaussian (RSG), and (d) Repliclust.

For the Moons, Circles, and Repliclust datasets, we considered six configurations for each type, determined by the combination of two cluster counts ($k \in \{2, 5\}$) and three dimensionalities ($D \in \{10, 50, 200\}$), resulting in a total of 18 configurations. For each configuration, 50 independent datasets were generated. For the RSG data, we directly used the publicly released synthetic datasets provided by Rodriguez et al. (Rodriguez *et al.*, 2019), which were generated using their structured Gaussian method. These datasets span three cluster counts ($k \in \{2, 10, 50\}$) and three dimensionalities ($D \in \{10, 50, 200\}$), with various numbers of samples per cluster ($N_e \in \{100, 50, 5\}$) as defined in their benchmark. We did not regenerate these datasets; instead, we adopted their exact configurations, totaling 265 datasets, to ensure comparability with prior work. In total, our experimental setup comprised 27 unique configurations, six for each of the Moons, Circles, and Repliclust datasets, and nine for the RSG datasets yielding 1,165 synthetic datasets for analysis. Examples of the generated datasets under the configuration of 50 features and two clusters are shown in Figure 2.1, where the data are visualized using their first and second principal components.

2.1.2 Real-world datasets

To complement the synthetic benchmarks, we evaluated clustering performance on 20 real-world datasets obtained from the UCI repository[1]. These datasets, originally compiled as a standard benchmark set in the work of Arbelaitz *et al*. (2013), span a diverse range of domains, sizes, feature dimensionalities, and class distributions. This diversity ensures a comprehensive evaluation of clustering algorithms under realistic conditions, including varying levels of noise, nonlinearity, and class imbalance. The key characteristics of these datasets including the number of objects, features, and classes are summarized in Table 2.1. They encompass application areas such as biology, medicine, chemistry, speech recognition, and image analysis. Each dataset is fully labeled, enabling the computation of external clustering validity indices by providing the ground-truth class memberships for all observations.

2.2 Dimensionality reduction methods used

We employed five dimensionality reduction techniques to project high-dimensional data into lower-dimensional spaces while preserving structural properties relevant for clustering: Principal Component Analysis (PCA), Kernel Principal Component Analysis (Kernel PCA), Variational Autoencoder (VAE), Isometric Mapping (Isomap), and Multidimensional Scaling (MDS). These methods encompass both linear and nonlinear approaches, providing a comprehensive framework for evaluating how geometry-preserving transformations influence clustering performance.

2.2.1 Principal Component Analysis (PCA)

PCA is a widely used linear dimensionality reduction technique that projects data onto a lower-dimensional orthogonal subspace by identifying directions known as principal components that maximize variance (Pearson, 1901). Given a dataset with n-dimensional features, PCA computes the covariance matrix of standardized data and derives its eigenvectors. The top m eigenvectors, corresponding to the largest eigenvalues, form the new basis. The data is then projected onto this m-dimensional subspace. PCA is effective when the data lies approximately in a linear manifold and provides an efficient baseline for comparison.

[1] https://archive.ics.uci.edu

2.2.2 Kernel Principal Component Analysis (Kernel PCA)

Kernel PCA extends PCA to nonlinear settings by implicitly mapping the data into a high-dimensional feature space via kernel functions, and then performing PCA in this transformed space (Schölkopf *et al.*, 1998). A kernel matrix **K** is constructed using pairwise similarities, commonly with a radial basis function:

$$K(y_i, y_j) = \exp\left(-\frac{\|y_i - y_j\|^2}{2\lambda^2}\right), \tag{2.1}$$

where λ denotes the kernel bandwidth. The centered kernel matrix is then decomposed to extract principal components in the kernel-induced space. This approach enables Kernel PCA to uncover nonlinear structures while retaining computational tractability.

2.2.3 Variational Autoencoder (VAE)

VAEs are generative models that learn probabilistic latent representations through neural networks (Kingma et Welling, 2013b). Unlike deterministic autoencoders, VAEs approximate the posterior distribution over latent variables using a variational inference framework. Given an observed variable x and a latent variable z, the objective of the VAE is to maximize the log-likelihood of the data:

$$\log p(x) = \log \int p(x|z)p(z)dz, \tag{2.2}$$

which is approximated using the evidence lower bound (ELBO):

$$\mathcal{L} = \mathbb{E}_{q(z|x)}[\log p(x|z)] - D_{KL}(q(z|x)||p(z)), \tag{2.3}$$

where $q(z|x)$ is the encoder, $p(x|z)$ the decoder, and $D_{KL}(\cdot||\cdot)$ is the Kullback-Leibler divergence, which regularizes the latent space by enforcing it to follow a prior distribution $p(z)$, typically a standard normal distribution $\mathcal{N}(0, I)$. The reparameterization trick enables differentiable sampling:

$$z = \mu + \sigma \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \tag{2.4}$$

where μ and σ are the model parameters representing the mean and variance of the latent distribution. VAEs are well-suited for capturing complex, nonlinear manifolds in high-dimensional data.

We employ a symmetric encoder–decoder design, following established principles in the literature to ensure stability, regularization, and representational power. The encoder has two fully connected layers (64 and

32 neurons, ReLU), followed by Batch Normalization and Dropout (0.4). The latent space is parameterized by z_{mean} and $z_{\log \text{var}}$ of size $n_{\text{components}}$, with sampling:

$$z = z_{\text{mean}} + \exp\big(0.5\, z_{\log \text{var}}\big) \odot \epsilon.$$

The decoder mirrors the encoder (32 and 64 neurons, ReLU, BatchNorm, Dropout = 0.4) and outputs dimension d with sigmoid activation. Batch Normalization and Dropout accelerate convergence, improve generalization, and mitigate posterior collapse. The VAE is trained using the Adam optimizer, MSE reconstruction loss, a batch size of 64, and 100 epochs, with a 70/30 train–test split. After training, the latent mean z_{mean} is retained as a deterministic embedding for clustering.

2.2.4 Isometric Mapping (Isomap)

Isomap is a nonlinear dimensionality reduction technique based on manifold learning that preserves geodesic distances rather than Euclidean distances (Tenenbaum *et al.*, 2000). It first constructs a neighborhood graph using the k-nearest neighbors method and stores the Euclidean distances between connected points in a weighted adjacency matrix $\mathbf{B}_G$. Pairwise geodesic distances are then estimated via shortest path algorithms, producing the geodesic distance matrix B_M. Finally, classical multidimensional scaling is applied to $\mathbf{B}_M$ to embed the data into a lower-dimensional space to produce a low-dimensional representation $\boldsymbol{Z}$ of the original high-dimensional data $\boldsymbol{X}$. The algorithm minimizes the following cost function:

$$\mathbf{E} = \| \gamma(\mathbf{B}_G - \mathbf{B}_Z) \|_F, \tag{2.5}$$

where $\mathbf{B}_Z$ is the distance matrix computed from the low-dimensional embedding $\boldsymbol{Z}$, and $\|.\|_F$ denotes the Frobenius norm, which measures the overall discrepancy between two matrices. The function $\gamma(.)$ converts a distance matrix into an inner product matrix using double centering:

$$\gamma(\mathbf{B}) = -\frac{\mathbf{H}\,\mathbf{S}\,\mathbf{H}}{2}, \tag{2.6}$$

with **S** being the matrix of squared distances and $\mathbf{H} = \mathbf{I} - \frac{1}{n}\,\mathbf{1}\,\mathbf{1}^\top$ is the centering matrix. Isomap excels at uncovering global nonlinear structures when the data lie on a low-dimensional manifold embedded in a higher-dimensional space.

2.2.5 Multidimensional Scaling (MDS)

MDS aims to place data points in a low-dimensional space such that pairwise distances approximate a given dissimilarity matrix (Kruskal, 1978). MDS is particularly well-suited for datasets where the notion of similarity

is not directly tied to Euclidean geometry for example, when dissimilarities are derived from correlation distances, semantic similarity measures, or domain-specific metrics. Given a set of dissimilarities, d_{ij} between all pairs of points, MDS finds a configuration $\boldsymbol{Z}$ in a low-dimensional space that minimizes the discrepancy between d_{ij} and the Euclidean distances d'_{ij} computed from $\boldsymbol{Z}$:

$$\delta(Z) = \min \sum_{i<j} (d_{ij} - d'_{ij})^2. \tag{2.7}$$

The stress function quantifies the embedding quality:

$$Stress = \frac{\sum_{i<j} (d_{ij} - d'_{ij})^2}{\sum_{i<j} (d'_{ij})^2}, \tag{2.8}$$

where lower stress values indicate a better preservation of the original dissimilarities.

2.3 Clustering algorithms used

We evaluated four widely used clustering algorithms representing distinct paradigms: centroid-based, probabilistic, hierarchical, and density-based clustering.

2.3.1 K-means

K-means is a centroid-based clustering algorithm that aim to find natural group in a given dataset so that each group is composed of similar entities (i.e., objects), whereas entities between groups are dissimilar. It iteratively assigns points to the nearest cluster centroid and updates centroids as the mean of assigned points until convergence (MacQueen, 1967; De Amorim et Makarenkov, 2016). While efficient, k-means assumes spherical clusters of equal variance and is sensitive to initialization. To improve robustness and reduce variance across runs, we adopt the k-*means++* initialization strategy and repeat clustering with $n\_init = 100$.

2.3.2 Agglomerative Hierarchical Clustering (AHC)

AHC is a bottom-up clustering algorithm that initially treats each data point as an individual cluster and successively merges the closest pairs until a stopping condition (e.g., number of clusters k) is reached (Sneath et Sokal, 1973). In our experiments, we consider different affinity measures (*euclidean*, $l1$, $l2$, *manhattan*, and *cosine*) together with multiple linkage strategies (*complete*, *ward*, *average*, and *single*), which allows us to systematically assess the influence of distance and merging methods on clustering outcomes.

2.3.3 Gaussian Mixture Model (GMM)

GMM is a probabilistic clustering model that assumes data are generated from a mixture of Gaussian distributions with unknown parameters (Wolfe, 1970). Each data point is associated with a probability of belonging to each component. Parameters (means, covariances, mixing weights) are estimated using the Expectation-Maximization (EM) algorithm, enabling soft clustering and modeling of elliptical cluster shapes.

GMM is a probabilistic clustering model that assumes data are generated from a mixture of Gaussian distributions with unknown parameters (Wolfe, 1970). Each point is assigned a probability of belonging to each component, and parameters (means, covariances, mixing weights) are estimated via the Expectation–Maximization (EM) algorithm, enabling soft assignments and flexible modeling of elliptical cluster shapes. In our experiments, we evaluate GMM under different covariance parameterizations (*spherical*, *tied*, *diag*, and *full*).

2.3.4 OPTICS

Ordering Points To Identify the Clustering Structure (OPTICS) is a density-based clustering algorithm that constructs an augmented ordering of the database to reveal density-based clusters of varying densities (Ankerst *et al.*, 1999). Unlike DBSCAN, it does not require a single global density threshold. OPTICS identifies reachability distances and core distances to generate a reachability plot, from which clusters can be extracted post hoc without strict parameterization. In our experiments, we vary key parameters to examine their impact on cluster detection: `min_samples` is swept from 5 to 10 (step = 1), the cluster extraction method is set to `xi`, and `min_cluster_size` ranges from 0 to 1 in increments of 0.05.

2.4 Evaluation Metric

We use the Adjusted Rand Index (ARI) to quantify the agreement between a clustering result and ground-truth labels when available (in our synthetic and real-world datasets). ARI is a chance-adjusted variant of the Rand Index with range $[-1, 1]$; random, independent partitions have an expected ARI close to 0, perfect agreement yields 1, and negative values indicate worse than random agreement Hubert et Arabie (1985).

Consider a dataset of n items with two partitions: $X = \{X_1, \ldots, X_r\}$ and $Y = \{Y_1, \ldots, Y_s\}$. Let $n_{ij} = |X_i \cap Y_j|$ denote the entries of the contingency table for $i = 1, \ldots, r$ and $j = 1, \ldots, s$, and let $a_i =$

$\sum_{j=1}^{s} n_{ij}$ and $b_j = \sum_{i=1}^{r} n_{ij}$ be the corresponding row and column sums. ARI is defined as follows:

$$\mathrm{ARI} = \frac{\sum_{i=1}^{r}\sum_{j=1}^{s}\binom{n_{ij}}{2} - \left[\sum_{i=1}^{r}\binom{a_i}{2}\sum_{j=1}^{s}\binom{b_j}{2}\right]/\binom{n}{2}}{\frac{1}{2}\left[\sum_{i=1}^{r}\binom{a_i}{2} + \sum_{j=1}^{s}\binom{b_j}{2}\right] - \left[\sum_{i=1}^{r}\binom{a_i}{2}\sum_{j=1}^{s}\binom{b_j}{2}\right]/\binom{n}{2}}. \tag{2.9}$$

Here, $\binom{m}{2}$ denotes the number of unordered pairs from m items.

2.5 Experimental Pipeline

The experimental pipeline consists of four main stages. First, all datasets were preprocessed using z-score normalization via the **StandardScaler** method, ensuring that each feature has zero mean and unit variance. This step mitigates the influence of scale differences across dimensions, which is crucial for distance-based clustering and projection techniques. Next, clustering was performed directly on the normalized datasets using four representative algorithms: k-means, AHC, GMM, and OPTICS. The quality of the resulting cluster partitions was evaluated using the ARI.

In the third stage, five dimensionality reduction techniques PCA, Kernel PCA, VAE, Isomap, and MDS were applied to each normalized dataset to project it into a lower-dimensional subspace. We considered three levels of dimensionality reduction: reducing to $\boldsymbol{k-1}$ dimensions, where $\boldsymbol{k}$ denotes the number of clusters and is motivated by the subspace bound in Ding *et al.* (2002), a more aggressive compression to **25%** of the original dimensionality; and a moderate retention of **50%** of the original dimensionality. These settings enable evaluation of clustering performance under both theoretically motivated and practically relevant dimensionality constraints. Clustering algorithms were then re-applied to the transformed data, and the ARI scores were computed to assess performance changes.

Finally, clustering outcomes with and without dimensionality reduction were compared to analyze the influence of each technique on clustering performance across various dataset geometries and dimensionalities. This comparison highlights the strengths and limitations of linear and nonlinear reduction methods under different datasets. Figure 2.2 presents a schematic overview of the experimental pipeline, illustrating the flow of data through preprocessing, dimensionality reduction, clustering, and evaluation stages.

All hyperparameters used in the experiments are summarized in Tables 2.2 and 2.3 to ensure full reproducibility. *Note:* k in Table 2.2 denotes the number of clusters in the dataset.

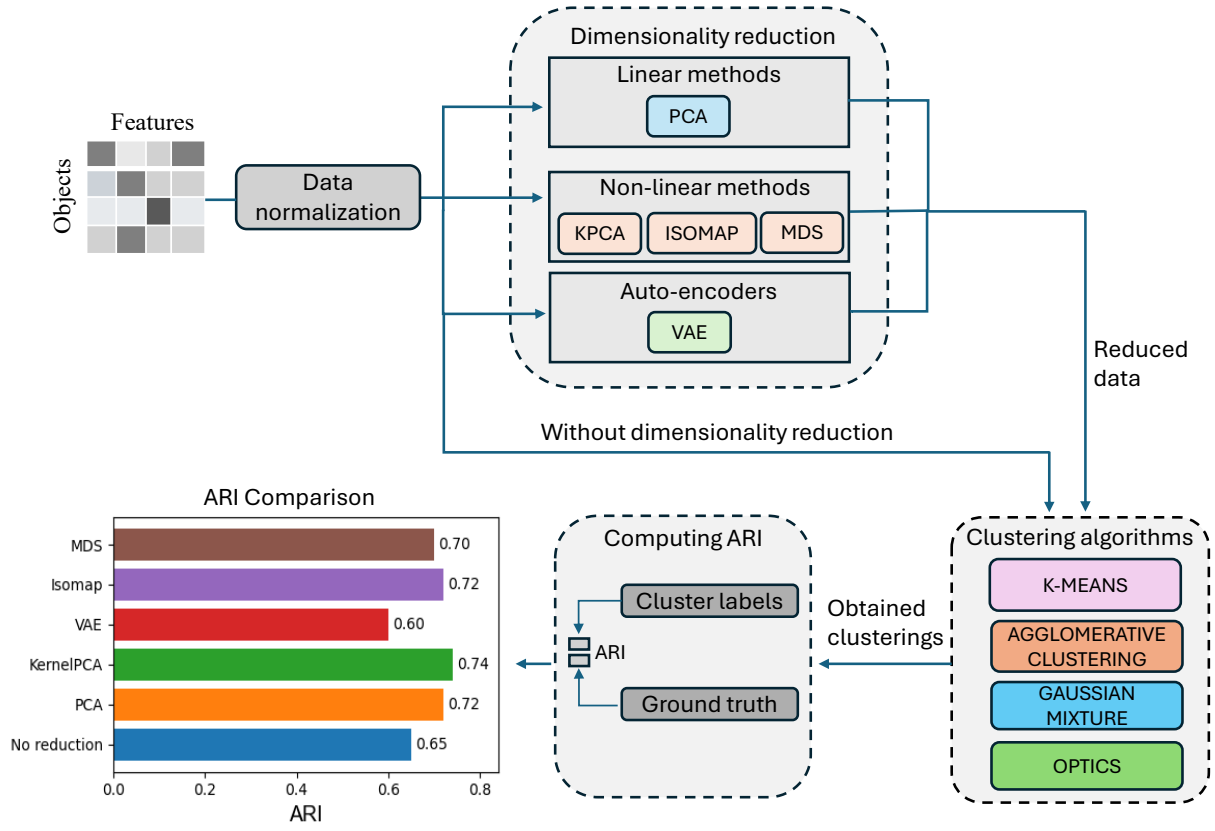

Fig. 2.2 Overview of the evaluation pipeline assessing the effect of dimensionality reduction on clustering.

2.6 Conclusion

In this chapter, we presented datasets that provide a diverse and complementary benchmark for evaluating clustering under a wide range of conditions. Synthetic datasets offer controlled variation in terms of geometry, dimensionality, and noise, while real-world datasets introduce realistic challenges arising from domain complexity and the inherent variability of the data. We then demonstrated how experiments can be conducted by describing the clustering algorithms and dimensionality reduction methods used, as well as the metric for evaluating model performance.

Table 2.1 Real-world datasets from the UCI Machine Learning Repository used in our evaluation.

Dataset	No. of objects	Features	Classes
Breast tissue	106	9	6
Breast Wisconsin	569	30	2
Ecoli	336	7	8
Glass	214	9	7
Haberman	306	3	2
Ionosphere	351	34	2
Iris	150	4	3
Movement libras	360	90	15
Musk	476	166	2
Parkinsons	195	22	2
Segmentation	2310	19	7
Sonar all	208	60	2
Spectf	267	44	2
Transfusion	748	4	2
Vehicle	846	18	4
Vertebral column	310	6	3
Vowel context	990	10	11
Wine	178	13	3
Winequality red	1599	11	6
Yeast	1484	8	10

Table 2.2 Hyperparameters used for dimensionality reduction methods

Method	**Hyperparameter**	**Value**
PCA	Number of components	$k-1$, 25%, 50% of original dimension
Kernel PCA	Kernel	RBF
	Components	$k-1$, 25%, 50% of original dimension
	Solver	Auto
VAE	Encoder	2 Dense layers (128 and 64)
	Decoder	2 Dense layers (64 and 128)
	Latent dimension	$k-1$, 25%, 50% of original dimension
	Activation	ReLU (hidden), sigmoid (output)
	Loss	MSE + KL divergence
	Optimizer	Adam
	Epochs	100 (early stopping)
	Batch size	64
	Learning rate	0.001
Isomap	Neighbors	5
	Components	$k-1$, 25%, 50% of original dimension
MDS	Components	$k-1$, 25%, 50% of original dimension
	random_state	10
	n_init	50
	Max iterations	300

Table 2.3 Hyperparameters used for clustering algorithms

Algorithm	**Hyperparameter**	**Value**
K-means	Number of clusters	Number of classes
	Initialization	k-means++
	n_init	100
AHC	Linkage	{ward, complete, average} (best selected per dataset based on ARI)
	Number of clusters	Number of classes
	Metric	{euclidean, manhattan} (best selected per dataset based on ARI)
GMM	Components	Number of classes
	Covariance type	full
OPTICS	Min samples	5
	min_cluster_size	0.05
	xi	0.02

CHAPTER 3
RESULTS AND DISCUSSION

The work presented in the previous chapters enabled us to conduct two experiments. This chapter will present the results and the conclusions that can be drawn from them. The first experiment concerns the experiments performed on the various synthetic data we generated. The second experiment focuses on real data, allowing us to gain insight into our framework applied to data under realistic conditions.

3.1 Results

3.1.1 Synthetic Data

We report results on synthetic datasets under three target dimensionalities: $k-1$, 25% reduction, and 50% reduction relative to the original dimensionality. Figures 3.1 to 3.4 summarize performance per algorithm (box plots), while section A.1 (Tables A.1 to A.4) provides per-dataset ARI values. Across synthetic datasets, the box plots show that applying a non-linear reduction typically shifts the ARI distribution upward relative to the baseline (no reduction). Kernel PCA delivers the most consistent gains for k-means and AHC, Isomap is particularly strong for GMM on nonlinear geometries, and MDS shows dataset-dependent improvements, notably on *Repliclust* (see Table A.4). VAE is generally unstable. Detailed win rates and average ARI changes relative to the baseline are reported in Section 3.1.3 (see Tables 3.1 to 3.4).

Figure 3.1 reports ARI distributions for k-means clustering. Kernel PCA emerged as the top performer, especially at 50% reduction, yielding high medians with low interquartile spread across datasets. It reached top ARI values in Circles, Moons, and Repliclust datasets (Tables A.1, A.2, and A.4). Isomap also provided strong results, often outperforming the baseline and PCA, particularly for manifold-shaped data such as Moons. VAE delivered moderate performance, generally better at higher dimensionality reduction. MDS exhibited mixed behavior, with some outlier high scores but overall less stability. Notably, the no-reduction baseline was rarely optimal, indicating that a moderate reduction via a suitable non-linear technique can improve separability for centroid-based clustering.

Figure 3.2 shows the ARI distribution for AHC across dimensionality reduction methods. Overall, non-linear manifold-based methods, particularly Isomap at 25% dimensionality, achieved the highest median

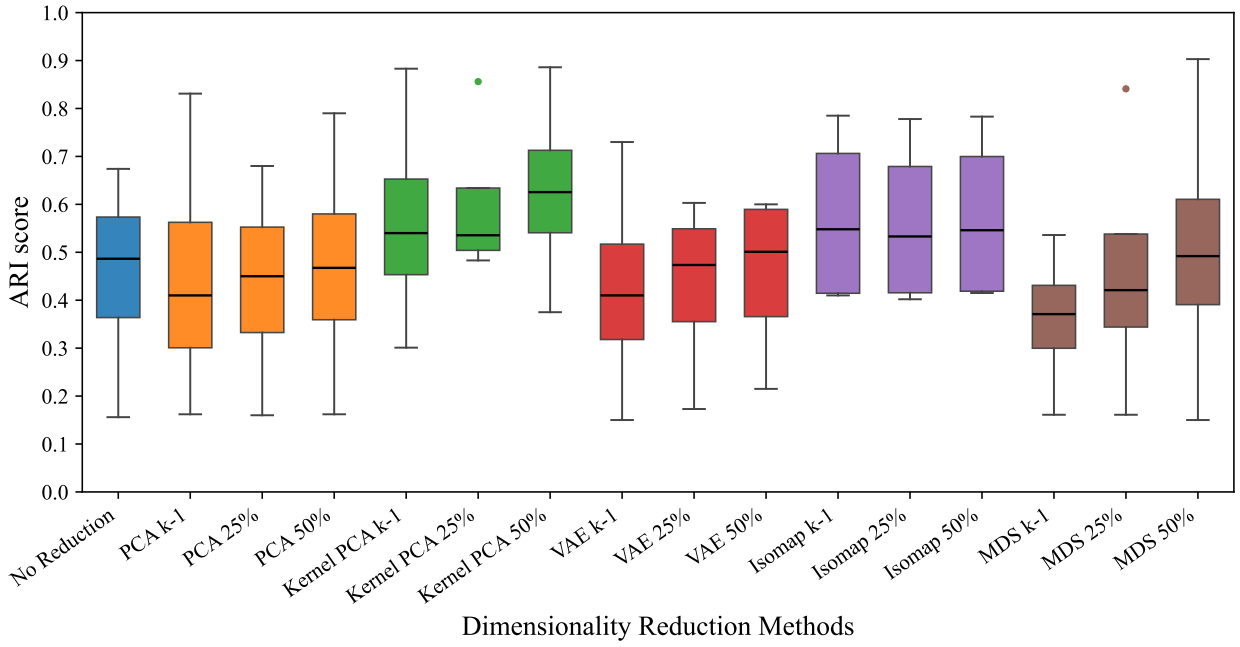

Fig. 3.1 Boxplots summarizing ARI scores for different dimensionality reduction methods applied to synthetic datasets, followed by clustering with k-means.

ARI scores across datasets, outperforming the no-reduction baseline in several cases. This trend is clearly visible in the Rodriguez and Moons datasets (Tables A.3 and A.2), where Isomap consistently surpasses both PCA and Kernel PCA. Kernel PCA also consistently maintained high performance. PCA yielded competitive results with slightly higher variance, while MDS demonstrated dataset-dependent variability, performing well in specific cases such as synthetic manifold data but poorly in others. VAE generally underperformed for AHC, with median scores lower than most traditional methods, particularly when reduced to very low dimensions, $k - 1$.

Figure 3.3 presents the ARI results for GMM. Here, the advantage of non-linear methods is even more evident: Isomap again dominated at 25%–50% dimensionality, delivering the highest median ARIs across most datasets, with the Moons dataset reaching above 0.88 ARI at 50% reduction (Table A.2). Kernel PCA maintained strong and stable performance, while PCA showed competitive yet slightly less stable behavior. VAE performance improved compared to AHC, especially at 50% reduction, suggesting better compatibility with probabilistic mixture models (e.g., Rodriguez dataset, Table A.3). MDS remained dataset-sensitive, with strong results in Repliclust (Table A.4) but weaker in Circles (Table A.1).

Figure 3.4 illustrates ARI performance for OPTICS. Results were more dataset-dependent, with MDS and

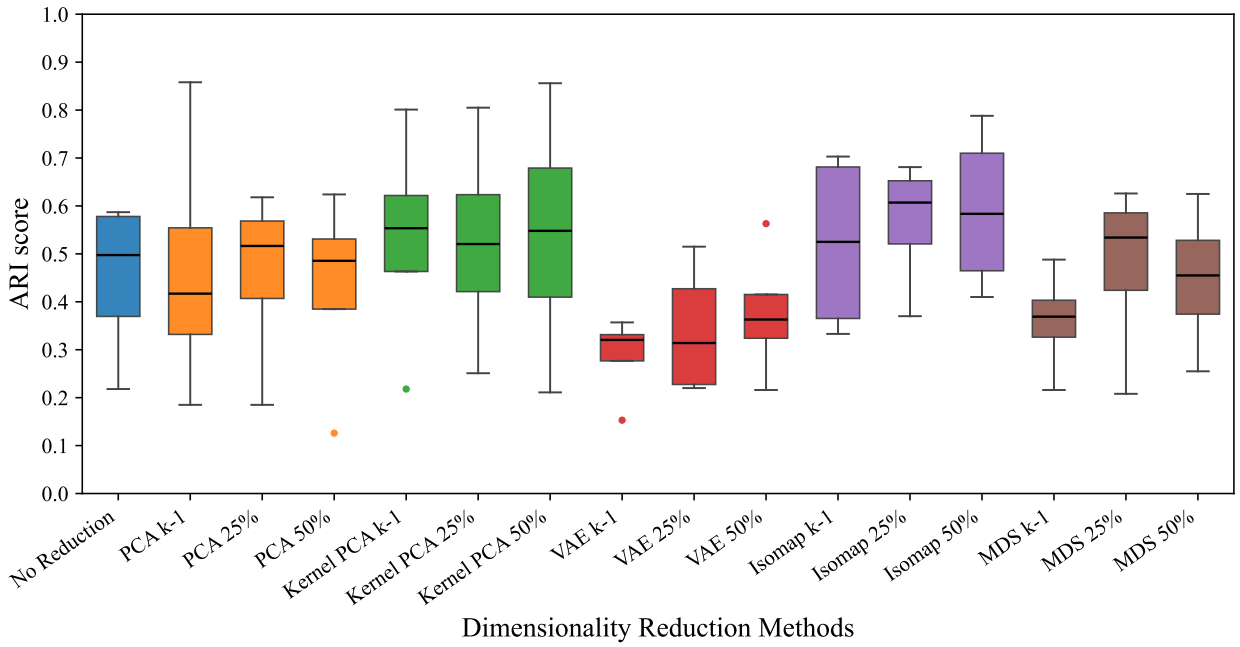

Fig. 3.2 Boxplots summarizing ARI scores for different dimensionality reduction methods applied to synthetic datasets, followed by clustering with AHC.

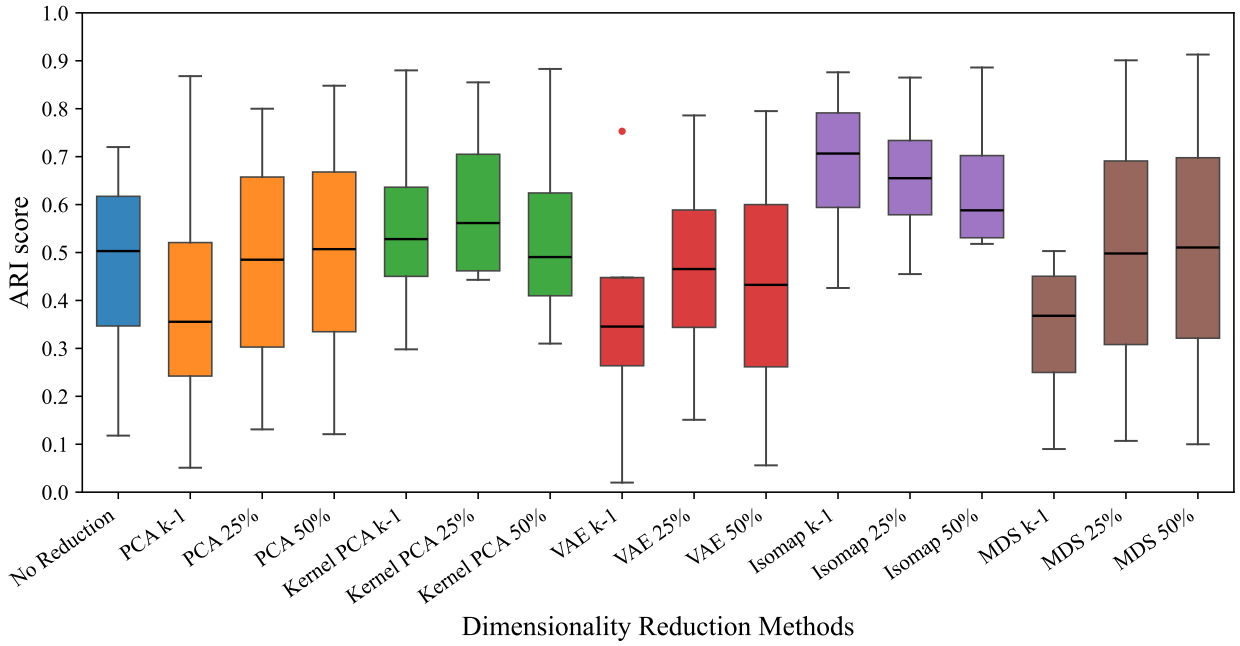

Fig. 3.3 Boxplots summarizing ARI scores for different dimensionality reduction methods applied to synthetic datasets, followed by clustering with GMM.

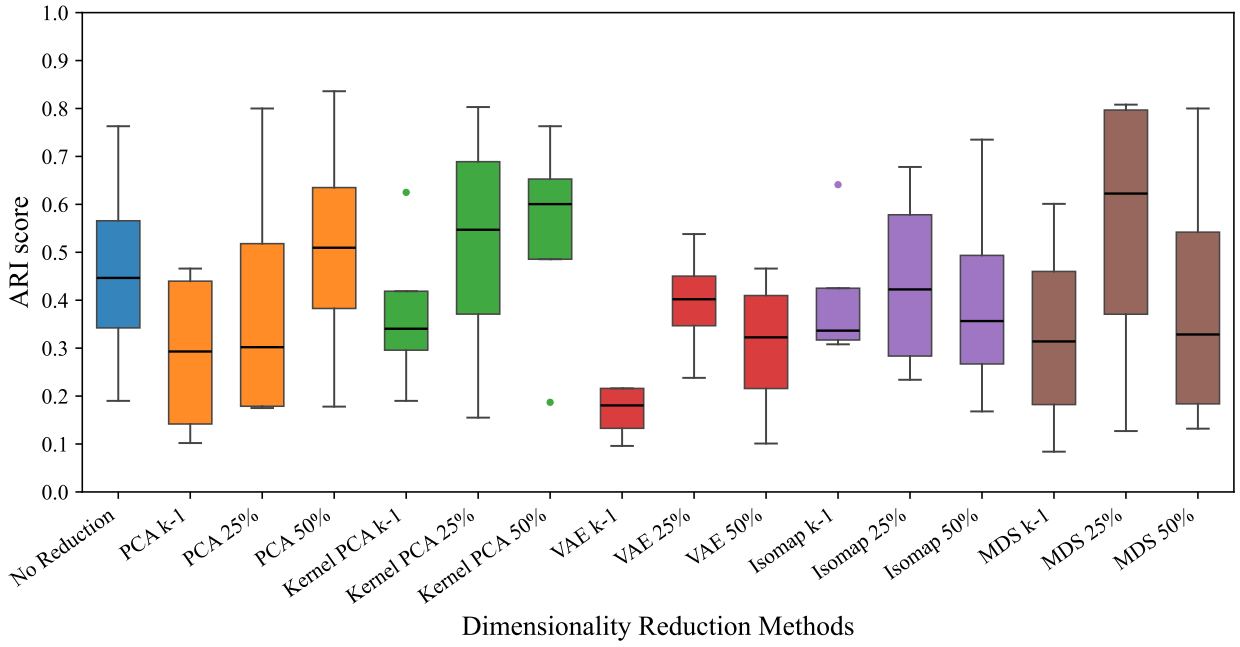

Fig. 3.4 Boxplots summarizing ARI scores for different dimensionality reduction methods applied to synthetic datasets, followed by clustering with OPTICS.

Kernel PCA showing competitive medians at specific reduction levels, such as MDS at 25% in the Circles dataset (Table A.1) and Kernel PCA at 50% in Repliclust (Table A.4). Isomap, despite its success in other algorithms, showed weaker performance here, likely due to the density-based nature of OPTICS being sensitive to distortions from manifold learning. PCA results were variable, with high ARIs on simple manifolds but notable drops in noisy high-dimensional data. VAE consistently underperformed, particularly at $k-1$, reflecting instability in preserving density-based neighborhood structures.

3.1.2 Experiments with Real-world Data

Across the 20 real-world datasets, dimensionality reduction rarely hurts k-means and occasionally yields sizeable gains, but no single technique dominates. The boxplots show medians clustered near the no-reduction baseline, with longer upper whiskers for PCA (25–50% reduction) and Isomap, indicating sporadic but meaningful improvements (see Fig. 3.5). Table A.5 from Appendix confirms these cases: on Iris, PCA (25% reduction) raises ARI from 0.62 to 0.80, and on Wine, the manifold methods remain competitive—Isomap (50% reduction) at 0.85 and MDS (50% reduction) at 0.88 without degrading the top baseline of 0.90. Conversely, Kernel PCA can be fragile on some real-world data (e.g., Breast Wisconsin = 0.00 when reduced), underscoring a strong interaction between the chosen kernel, retained dimensionality, and clus-

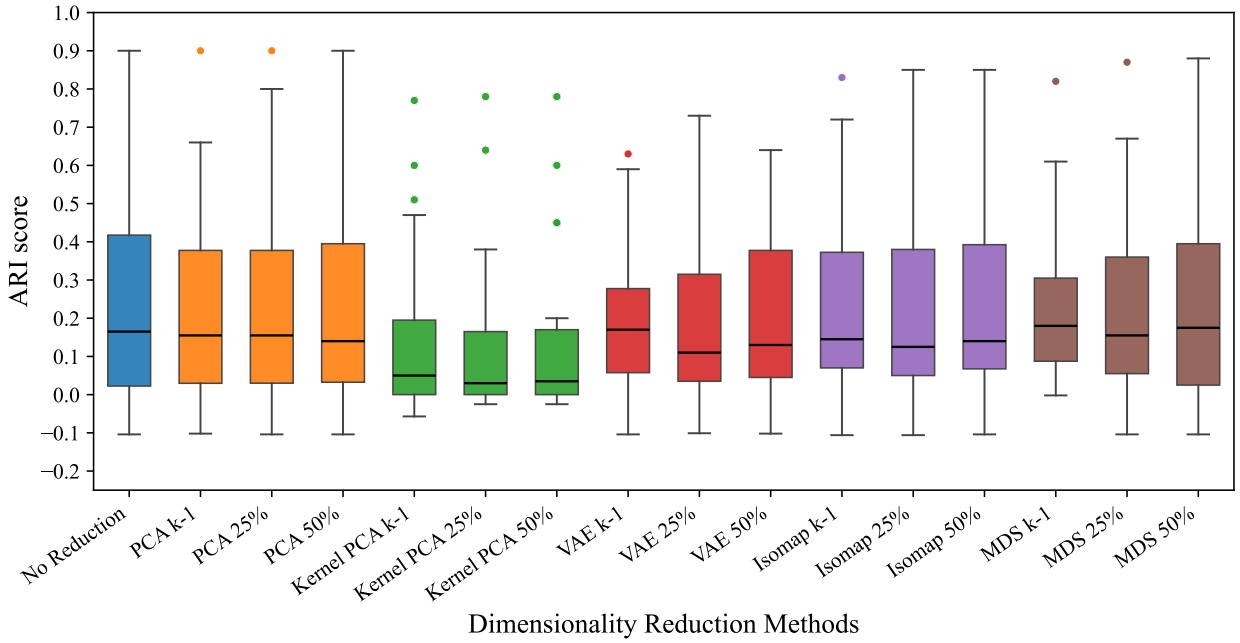

Fig. 3.5 Boxplots summarizing ARI scores for different dimensionality reduction methods applied to real-world datasets, followed by clustering with k-means.

ter geometry.

For hierarchical clustering, moderate reduction tends to help more consistently, with the distributions for PCA(25–50% reduction), Kernel PCA, and Isomap generally shifted upward relative to no reduction (see Fig. 3.6). Several datasets illustrate this pattern in Table A.6: Breast Wisconsin improves from 0.58 (no reduction) to 0.71 with PCA (50% reduction), Wine reaches its best score with Kernel PCA ($k-1$ reduction) at 0.93, and Ecoli benefits from Isomap (50% reduction), 0.55 vs. 0.52 baseline. Still, variance is non-negligible, and VAE can introduce instability (e.g., negative or near-zero ARIs on Spectf), which aligns with the lower medians and wider spreads visible in the figure.

GMM profits the most from geometry-preserving reductions. The highest whiskers are associated with Kernel PCA and Isomap (see Fig. 3.7), and the appendix shows repeated wins on structured data: Wine peaks at 0.95 with Kernel PCA (50% reduction) and remains strong with Isomap (25–50% reduction) at 0.82–0.90 (Table A.7). PCA frequently matches or slightly exceeds the baseline on Segmentation datasets (0.47 at PCA (50% reduction) vs. 0.43 with no reduction). In contrast, VAE exhibits dataset-dependent behavior—helpful on Winequality red (0.19 at 25% reduction) but detrimental on several small, noisy sets mirroring the wider dispersion seen in the boxplots.

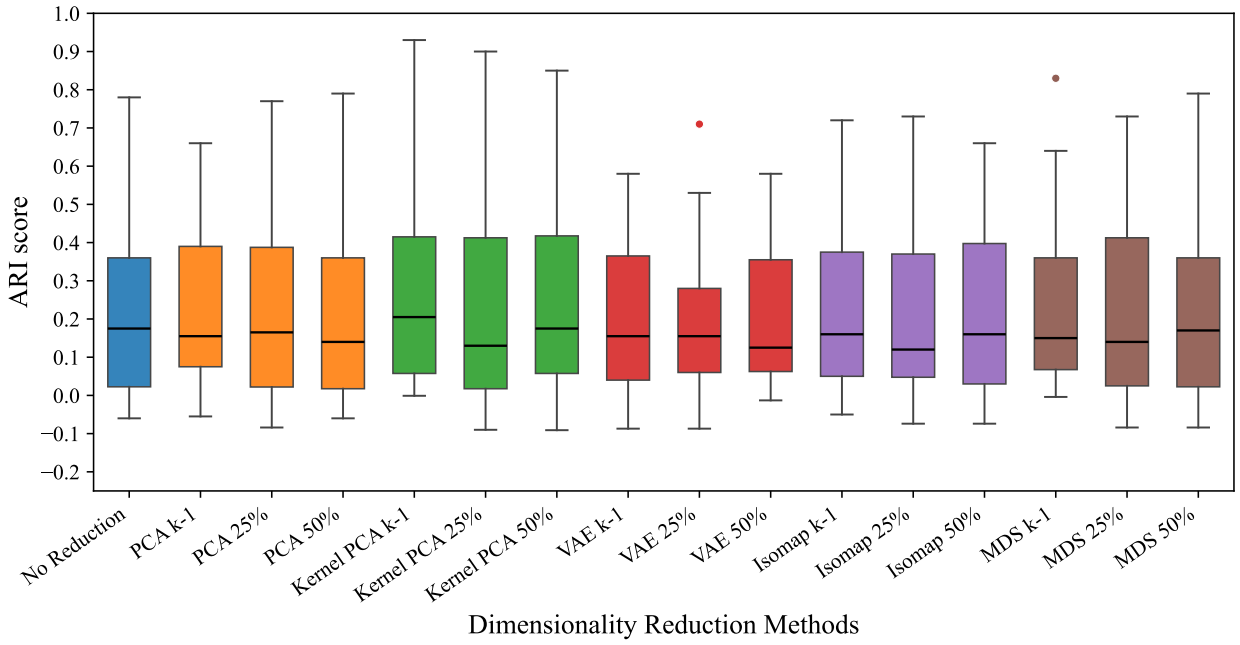

Fig. 3.6 Boxplots summarizing ARI scores for different dimensionality reduction methods applied to real-world datasets, followed by clustering with AHC.

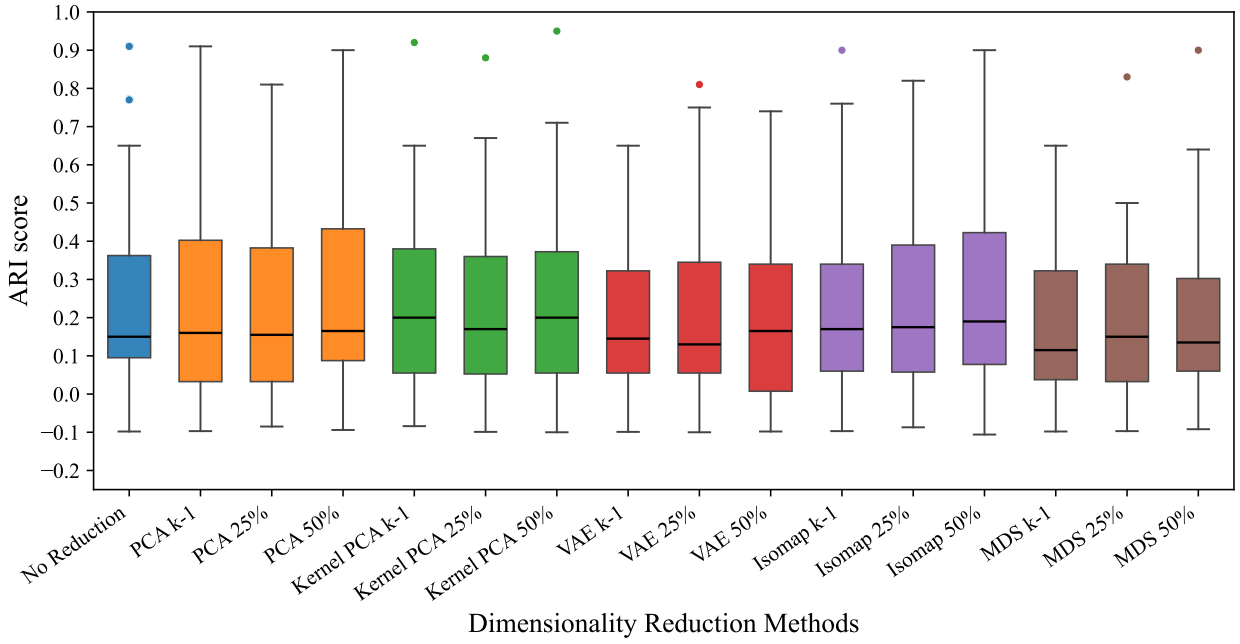

Fig. 3.7 Boxplots summarizing ARI scores for different dimensionality reduction methods applied to real-world datasets, followed by clustering with GMM.

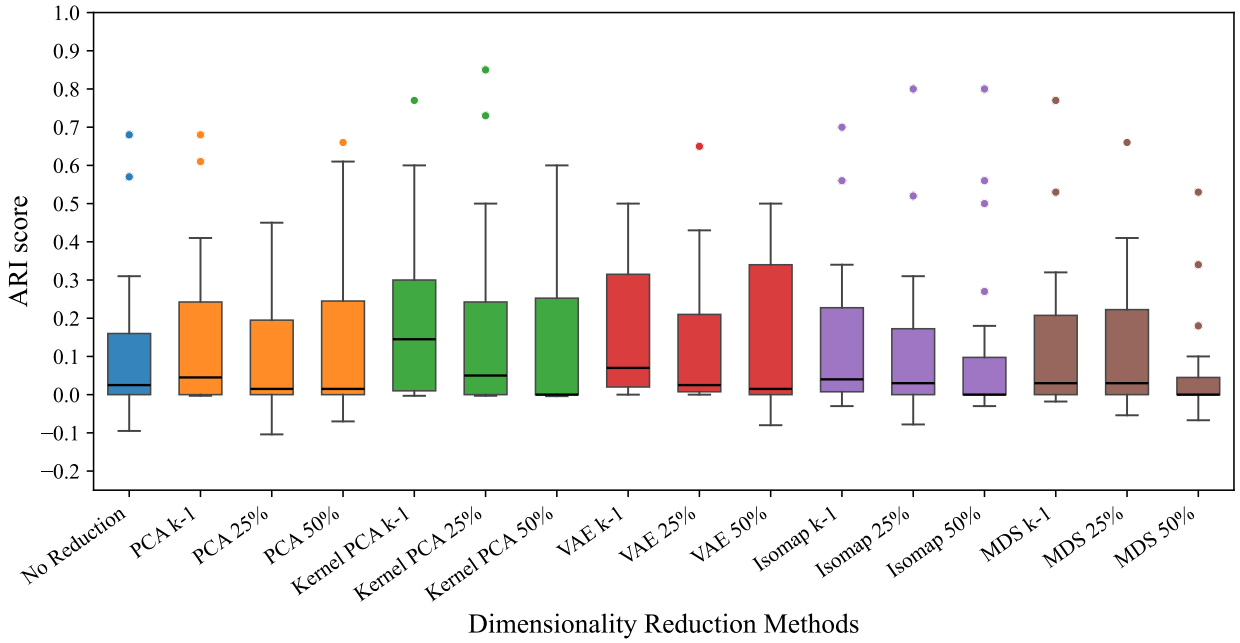

Fig. 3.8 Boxplots summarizing ARI scores for different dimensionality reduction methods applied to real-world datasets, followed by clustering with OPTICS.

Density-based clustering is the most sensitive to the choice of reduction: distributions are broad with many low medians, yet select method–dataset pairs deliver clear gains (see Fig. 3.8. The appendix highlights notable outliers: Ionosphere improves with Kernel PCA (25% D) to 0.73 from a 0.68 baseline, while Wine benefits markedly from Kernel PCA (25% reduction) and Isomap (25–50% reduction), reaching 0.85–0.80 versus 0.00 baseline (Table A.8). At the same time, some VAE settings collapse neighborhood structure (e.g., Spectf -0.87 at 25% reduction), explaining the low medians and long lower whiskers for several VAE configurations.

3.1.3 Aggregate analysis

We summarize the effect of dimensionality reduction across algorithms using two complementary statistics: (i) percentage of wins, how often a reduced representation beats the No Reduction baseline and (ii) average win/loss percentage, the mean relative ARI change when it wins or loses. Tables 3.1 to 3.4 report these metrics for synthetic vs. real-world data and for three retention levels ($k-1$, 25% reduction, and 50% reduction). This aggregate view controls for dataset idiosyncrasies and exposes algorithm–reduction interactions that are not visible from per-dataset plots alone.

Table 3.1 Comparison of performance improvements for k-means across different dimensionality reduction methods on synthetic and real-world datasets.

Method	Reduction	Percentage of Wins		Average win/loss (-) percentage	
		Synthetic Data	Real-world Data	Synthetic Data	Real-world Data
PCA	$k-1$	26.48	55.55	-3.34	-0.15
	25%	28.07	70.00	-1.99	0.70
	50%	25.02	45.45	0.21	0.09
Kernel PCA	$k-1$	75.18	20.00	13.66	-7.70
	25%	61.27	23.52	12.64	-9.80
	50%	73.30	16.66	18.19	-7.90
VAE	$k-1$	27.26	25.00	-15.31	-4.75
	25%	27.30	25.00	-9.03	-3.75
	50%	31.33	29.41	-7.08	-2.50
Isomap	$k-1$	55.25	53.84	5.11	-0.40
	25%	47.34	40.00	-1.40	-1.30
	50%	55.96	50.00	4.58	0.59
MDS	$k-1$	51.76	54.54	-16.41	-1.50
	25%	30.63	53.84	-3.08	-0.30
	50%	29.18	63.63	-0.27	0.29

The results reported in Table 3.1 indicate that for the k-means, Kernel PCA is clearly dominant (wins up to 75.18% with 18.19% average gains at 50% reduction), indicating that non-linear geometry can substantially improve centroid separability. However, this advantage collapses on real-world data (wins $\leq$23.52%, average changes -7% to -10%), suggesting sensitivity to kernel/model mismatch and noise. In contrast, PCA and Isomap are steadier across domains (e.g., PCA at 25% reduction: 70.00% wins, 0.70% average change on real-world data; Isomap at 50% reduction: 50.00%, 0.59%), delivering small but reliable benefits. VAE and MDS are consistently unreliable for k-means on real-world data (negative average changes at all levels), and should generally be avoided.

Table 3.2 Comparison of performance improvements for AHC across different dimensionality reduction methods on synthetic and real-world datasets.

Method	Reduction	Percentage of Wins		Average win/loss (-) percentage	
		Synthetic Data	Real-world Data	Synthetic Data	Real-world Data
PCA	$k-1$	41.35	50.00	-0.38	0.25
	25%	43.85	60.00	0.10	0.80
	50%	46.16	46.15	1.45	-0.24
Kernel PCA	$k-1$	68.21	66.66	23.09	2.55
	25%	60.76	52.94	18.34	-0.59
	50%	64.80	57.89	25.67	0.40
VAE	$k-1$	26.35	29.41	-16.58	-3.35
	25%	27.50	37.50	-13.56	-3.89
	50%	29.95	23.52	-12.30	-4.85
Isomap	$k-1$	58.89	69.23	12.78	2.05
	25%	50.99	50.00	6.59	-0.29
	50%	57.57	52.63	13.40	0.55
MDS	$k-1$	51.11	60.00	-7.09	-0.34
	25%	40.14	55.55	0.89	-0.39
	50%	41.63	41.17	1.02	-0.49

For AHC, Kernel PCA is the most robust overall (see Table 3.2), winning frequently on both synthetic (68.21% at $k-1$ reduction) and real-world datasets (66.66% at $k-1$ reduction), with sizeable mean improvements on synthetic (+25.67%) at 50% reduction and non-negative changes on real-world data. Isomap is a strong second (e.g., 57.57% wins, and +13.40% average win/loss at 50% reduction for synthetic; 52.63% wins,+0.55% average win/loss for real-world data), while PCA yields moderate, often positive, changes (best at 25% reduction: 60.00% wins, and +0.80% average win/loss on real-world data). VAE underperforms systematically (average win/loss changes -12% to -17% for synthetic; -3% to -5% for real-world data), indicating that its latent geometry does not preserve the linkages needed by hierarchical clustering.

Table 3.3 Comparison of performance improvements for the GMM across different dimensionality reduction methods on synthetic and real-world datasets.

Method	Reduction	Percentage of Wins		Average win/loss (-) percentage	
		Synthetic Data	Real-world Data	Synthetic Data	Real-world Data
PCA	$k-1$	32.43	38.46	-2.05	-1.85
	25%	47.57	53.33	0.86	-3.89
	50%	43.96	62.50	0.10	-1.60
Kernel PCA	$k-1$	60.13	52.94	15.22	-0.05
	25%	58.34	33.33	11.11	-2.65
	50%	50.98	43.75	10.89	-0.45
VAE	$k-1$	30.71	23.52	-13.01	-6.49
	25%	36.29	31.57	-3.49	-3.19
	50%	36.73	27.77	-3.06	-4.35
Isomap	$k-1$	62.42	20.00	15.91	-0.70
	25%	59.54	52.94	11.16	-0.95
	50%	59.82	44.44	16.11	0.19
MDS	$k-1$	32.80	43.75	-17.09	-6.30
	25%	45.39	44.44	1.53	-4.85
	50%	45.92	25.00	0.38	-4.65

In the context of the GMM (see Table 3.3), for synthetic data, Kernel PCA and Isomap again help (wins 50–62%, average gains +10–16%), consistent with mixtures benefiting from curvature-aware embeddings. On real-world data, however, all reductions are at best neutral and often harmful: win rates fall, and average changes are negative or near zero for PCA, Kernel PCA, VAE, Isomap, and MDS. Practically, dimensionality reduction should be applied cautiously or not at all before GMMs on real-world data.

In contrast to GMM, density-based clustering (OPTICS) on real-world data benefits from Kernel PCA and, notably, VAE at modest retention: at $k-1$ and 25% reduction, win rates are 73.33%/62.50% with +7.10%/+6.99% average win/loss (Kernel PCA) and 66.66%/56.25% wins with +4.60%/-3.24% average win/loss (VAE), re-

Table 3.4 Comparison of performance improvements for OPTICS across different dimensionality reduction methods on synthetic and real-world datasets.

Method	Reduction	Percentage of Wins		Average win/loss (-) percentage	
		Synthetic Data	Real-world Data	Synthetic Data	Real-world Data
PCA	$k-1$	43.12	46.15	-18.33	3.2
	25%	32.61	46.15	2.46	-0.94
	50%	25.02	41.66	-0.73	1.00
Kernel PCA	$k-1$	46.09	73.33	-15.44	7.10
	25%	41.15	62.50	1.38	6.99
	50%	43.60	56.25	4.67	0.65
VAE	$k-1$	36.75	66.66	-25.39	4.60
	25%	29.72	56.25	-4.34	-3.24
	50%	24.46	53.33	-7.92	0.65
Isomap	$k-1$	43.21	46.66	-24.81	2.40
	25%	37.43	41.17	-25.13	0.15
	50%	39.46	43.75	-19.28	0.30
MDS	$k-1$	41.54	53.33	-17.83	2.05
	25%	33.67	45.45	1.65	-0.40
	50%	26.95	23.07	-1.49	-5.80

spectively (see Table 3.4). While VAE's mean change at 25% reduction is negative, its high win rate and positive result at $k-1$ indicate promise when compression is aggressive. MDS shows occasional utility (e.g., at $k-1$: 53.33% win rate, and +2.05% gain for average win/loss on real-world data) but deteriorates at stronger retention. PCA and Isomap are unreliable for OPTICS on real-world data (frequent losses, small or negative averages). The results of all our experiments, including the source code, are available at: `https://github.com/OusmaneAmate/Dimensionality_reduction-for-clustering`.

3.2 Discussion

The results from synthetic datasets underscore the strong influence of the interaction between clustering algorithms and dimensionality reduction techniques. Non-linear manifold learning methods, particularly Isomap and Kernel PCA, consistently enhanced performance for algorithms relying on distance or centroid structures (k-means, AHC, GMM), confirming earlier findings that such methods preserve global and local geometry better than purely linear projections when the intrinsic data structure is non-linear. Moderate dimensionality reductions (25%–50% of original features) often outperformed both extreme reductions ($k-1$) and no reduction, suggesting a balance between noise removal and information preservation. The VAE-based approach, while theoretically appealing for complex manifolds, showed instability, especially for hierarchical and density-based clustering, likely due to its probabilistic latent representation, introducing distortions in inter-point distances. MDS displayed high variance, excelling on some datasets with well-separated clusters but failing on others with overlapping manifolds, reflecting its sensitivity to noise and scaling. Importantly, the consistent outperformance of certain reduction methods over the no-reduction baseline challenges the assumption that clustering should always be performed in the original feature space, and supports the integration of dimensionality reduction particularly non-linear approaches as a standard preprocessing step in complex high-dimensional clustering pipelines.

Taken together, the results for real-world data argue for method-algorithm fit and moderate retention (25-50% reduction) over both aggressive compression to $k-1$ and naive no reduction. For distance or model-based clustering algorithms (AHC, GMM), geometry-aware maps Kernel PCA with a well-tuned kernel and Isomap most reliably improve separability, often converting middling baselines into state-of-the-art ARIs (e.g., Wine in Tables A.6 to A.7). k-means benefits more opportunistically: linear PCA or manifold learners can deliver large jumps on intrinsically low-dimensional structure (Iris, Wine), but can also be neutral when clusters are already roughly spherical in the input space. OPTICS is uniquely fragile, preserving fine-grained density requires embeddings that maintain local neighborhoods with minimal distortion, Kernel PCA and Isomap sometimes succeed (e.g., Ionosphere, Wine), whereas VAE's stochastic latent geometry can be harmful (Table A.8). Practically, we recommend (i) screening reductions at 25-50% reduction as a default, (ii) preferring Kernel PCA/Isomap for hierarchical or mixture models, (iii) using PCA as a strong, low-risk baseline for k-means, and (iv) validating any VAE embedding with neighborhood and density diagnostics before applying density-based clustering. These guidelines, grounded in 20 heterogeneous real-world datasets, support dimensionality reduction as a principled, performance-enhancing step rather than

a cosmetic preprocessing choice.

Finally, from the aggregate analysis, three patterns emerge. First, method–algorithm fit dominates: Kernel PCA/Isomap pair naturally with AHC and (on synthetic) GMM, while Kernel PCA/VAE are preferable for OPTICS in real-world data. Second, moderate retention (25–50% reduction) is generally safer than aggressive $k-1$ compression, except for OPTICS, where $k-1$ can be competitive. Third, domain matters: gains observed on synthetic manifolds do not automatically transfer to heterogeneous real-world datasets. Collectively, the aggregates justify a results-driven protocol: screen a small slate of reductions tailored to the clustering algorithm, prefer moderate compression, and avoid VAE/MDS for k-means and GMM on real-world data unless validated by per-dataset diagnostics.

The obtained results show that nonlinear dimensionality reduction techniques such as Kernel PCA and Isomap can be effective in several clustering scenarios. However, their performance remains strongly dependent on the choice of hyperparameters. In the case of Kernel PCA, the selected kernel and its associated parameters directly influence the structure of the reduced representation. Similarly, Isomap is highly sensitive to the neighborhood graph construction, particularly to the number of neighbors used to capture the local geometry of the data. As a result, the performance reported in this study should be interpreted in light of the experimental parameter settings adopted rather than as an absolute indication of the superiority of these methods.

CONCLUSION

In recent years, the growth of high-dimensional data in fields like computer vision, biomedical analysis, and industrial monitoring has made it harder to group observations effectively. In these high-dimensional settings, traditional distance measures often lose their ability to distinguish well clustering structures. Data sparsity increases, and clustering algorithms often struggle to reveal meaningful patterns.

At the same time, DR techniques have emerged. These range from classical linear methods like PCA to nonlinear approaches such as manifold learning and deep representation learning. They provide powerful tools to simplify complex data into lower-dimensional and more informative forms. Despite this variety of DR methods, there is still a gap in the literature as to there use prior to clustering experiments. The choice of the DR method to use with a given clustering algorithm is often made in a casual way, influenced by the specific features of the dataset. It rarely follows a careful, systematic comparison. This highlights the need for a detailed investigation into how different DR strategies impact clustering results, and which DR and clustering pairings offer consistently strong performance across various data types and conditions.

The main goal of this work is to carry out a thorough comparison of how dimensionality reduction techniques interact with clustering algorithms. We want to see how these combinations influence the overall quality of clustering. To meet this goal, we created an experimental framework that includes several DR methods, such as PCA, Kernel PCA, Isomap, MDS, and VAEs. We paired them with four popular clustering algorithms: k-means, AHC, GMM, and OPTICS. A significant part of our work consists in the detailed assessment of these combinations on both synthetic and real-world datasets. This evaluation is backed by a quantitative analysis using the Adjusted Rand Index (ARI).

Our study provides a large-scale, statistically grounded answer to a simple but consequential question: *how much can clustering performance be improved by reducing data dimensionality?* Pairing five reduction techniques with four clustering algorithms over 1,165 synthetic datasets and 20 real benchmarks, we find that dimensionality reduction yields consistent, and sometimes substantial gains. Yet, its benefit hinges on the fit between the embedding geometry and the downstream clustering algorithm. Three conclusions emerge. (i) DR method - Clustering algorithm fit dominates. On synthetic data, Kernel PCA is the most reliable enhancer for distance and centroid-based methods, e.g., 25.67% average ARI improvement with AHC and 18.19% with k-means at 50% feature dimensionality reduction, while Isomap is particularly effective with GMM on non-

linear manifolds; conversely, VAE embeddings are unstable for hierarchical and density-based clustering. (ii) Moderate DR beats extreme cases. Keeping 25 to 50% of features typically balances denoising and structure preservation better than either no reduction or aggressive compression to $k-1$ dimensions. (iii) Domain matters. Improvements attenuate on heterogeneous real-world datasets, but targeted pairings still help; for example, OPTICS benefits from Kernel PCA (up to 6.99% at 25% retention), whereas indiscriminate use of deep embeddings can degrade neighborhood structure. These patterns recast dimensionality reduction as a first-order design factor in unsupervised pipelines, not an afterthought.

Practically, we recommend: (a) using reduction rate of 25–50% as a default, (b) prioritizing Kernel PCA or Isomap with AHC and, on synthetic-like geometry, with GMM, (c) using PCA as a strong, low-risk baseline for k-means, and (d) validating VAE embeddings with local-neighborhood diagnostics before applying density-based methods.

Beyond these contributions, several promising directions for future investigation emerge. Automated model selection for kernels, similarity graphs, and neighborhood parameters could substantially improve DR robustness. Joint learning frameworks that integrate dimensionality reduction and clustering such as DEC (Deep Embedded Clustering), IDEC (Improved Deep Embedded Clustering), or VaDE (Variational Deep Embedding) represent a natural extension of this work, enabling embeddings explicitly optimized for cluster separation. Further studies should also examine robustness to noise, class imbalance, and varying degrees of heterogeneity, as well as scalability to extremely high-dimensional or streaming datasets. Incorporating additional modalities, such as time series (Durbin, 1960; Esling et Agon, 2012), images (Wäldchen et Mäder, 2018; Willemink *et al.*, 2020), or graph-structured data (Makarenkov et Legendre, 2000; Makarenkov *et al.*, 2004), may reveal new interactions between data properties or geometry and clustering performance. Finally, integrating visualization oriented methods (e.g., UMAP, t-SNE) within reproducible benchmarking protocols would help clarify their role beyond exploratory analysis.

One limitation of this study is that it does not explicitly evaluate the computational cost associated with the dimensionality reduction methods under consideration. While the analysis focused on the impact of these methods on clustering quality, some techniques, particularly nonlinear and deep learning-based approaches such as MDS and VAEs, may involve significantly higher computational demands than linear methods such as PCA. These costs may appear in the form of longer execution times, increased memory consumption, and reduced scalability for high-dimensional or large-scale datasets. As a result, the conclusions

drawn in this work should be interpreted primarily from a performance perspective rather than from a computational efficiency standpoint.

By releasing a reproducible experimental framework, this thesis aims to support future research and to guide practitioners in selecting dimensionality reduction strategies that are well aligned with data geometry and clustering objectives.

BIBLIOGRAPHY

Aggarwal, C. C., Hinneburg, A. et Keim, D. A. (2001). On the surprising behavior of distance metrics in high dimensional space. Dans *International conference on database theory*, 420–434. Springer.

Agrafiotis, D. K. (2003). Stochastic proximity embedding. *Journal of computational chemistry*, *24*(10), 1215–1221.

Agrawal, R., Gehrke, J., Gunopulos, D. et Raghavan, P. (1998). Automatic subspace clustering of high dimensional data for data mining applications. Dans *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, 94–105.

Ahmed, M., Mahmood, A. N. et Hu, J. (2016). A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, *60*, 19–31.

Alkhayrat, M., Aljnidi, M. et Aljoumaa, K. (2020). A comparative dimensionality reduction study in telecom customer segmentation using deep learning and pca. *Journal of Big Data*, *7*(1), 9.

Allaoui, M., Kherfi, M. L. et Cheriet, A. (2020). Considerably improving clustering algorithms using umap dimensionality reduction technique: a comparative study. Dans *International conference on image and signal processing*, 317–325. Springer.

Ankerst, M., Breunig, M. M., Kriegel, H.-P. et Sander, J. (1999). Optics: ordering points to identify the clustering structure. Dans *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, SIGMOD '99, p. 49–60., New York, NY, USA. Association for Computing Machinery. `http://dx.doi.org/10.1145/304182.304187`

Arbelaitz, O., Gurrutxaga, I., Muguerza, J., Pérez, J. M. et Perona, I. (2013). An extensive comparative study of cluster validity indices. *Pattern Recognition*, *46*(1), 243–256. `http://dx.doi.org/10.1016/j.patcog.2012.07.021`

Assent, I. (2012). Clustering high dimensional data. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *2*(4), 340–350.

Balasubramanian, M. et Schwartz, E. L. (2002). The isomap algorithm and topological stability. *Science*, *295*(5552), 7–7.

Baldi, P. (2012). Autoencoders, unsupervised learning, and deep architectures. Dans *Proceedings of ICML workshop on unsupervised and transfer learning*, 37–49. JMLR Workshop and Conference Proceedings.

Bank, D., Koenigstein, N. et Giryes, R. (2023). Autoencoders. *Machine learning for data science handbook: data mining and knowledge discovery handbook*, 353–374.

Becht, E., McInnes, L., Healy, J., Dutertre, C.-A., Kwok, I. W., Ng, L. G., Ginhoux, F. et Newell, E. W. (2019). Dimensionality reduction for visualizing single-cell data using umap. *Nature biotechnology*, *37*(1), 38–44.

Beyer, K., Goldstein, J., Ramakrishnan, R. et Shaft, U. (1999). When is "nearest neighbor" meaningful? Dans *International conference on database theory*, 217–235. Springer.

Bezdek, J. C. (2013). *Pattern recognition with fuzzy objective function algorithms*. Springer Science & Business Media.

Cao, L. (2006). Singular value decomposition applied to digital image processing. *Division of Computing Studies, Arizona State University Polytechnic Campus, Mesa, Arizona State University polytechnic Campus*, 1–15.

Chatfield, C. (2018). *Introduction to multivariate analysis*. Routledge.

Chen, Y., Wu, H., Shi, J., Shang, J. et Sun, W. (2018). Application of singular value decomposition algorithm to dimension-reduced clustering analysis of daily load profiles. *Automation of Electric Power Systems*, *42*(3), 105–111.

Cichocki, A., Yang, H. *et al.* (1996). A new learning algorithm for blind signal separation. *Advances in neural information processing systems*, *8*, 757–763.

Comon, P. (1994). Independent component analysis, a new concept? *Signal processing*, *36*(3), 287–314.

Dalmaijer, E. S., Nord, C. L. et Astle, D. E. (2022). Statistical power for cluster analysis. *BMC bioinformatics*, *23*(1), 205.

De Amorim, R. C. et Makarenkov, V. (2016). Applying subclustering and lp distance in weighted k-means with distributed centroids. *Neurocomputing*, *173*, 700–707.

de Amorim, R. C. et Makarenkov, V. (2023). On k-means iterations and gaussian clusters. *Neurocomputing*, *553*, 126547.

De Amorim, R. C., Makarenkov, V. et Mirkin, B. (2016). A-wardpβ: Effective hierarchical clustering using the minkowski metric and a fast k-means initialisation. *Information Sciences*, *370*, 343–354.

Demartines, P. et Hérault, J. (1997). Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets. *IEEE Transactions on neural networks*, *8*(1), 148–154.

Ding, C. et He, X. (2004). K-means clustering via principal component analysis. Dans *Proceedings of the twenty-first international conference on Machine learning*, p. 29.

Ding, C., He, X., Zha, H. et Simon, H. (2002). Adaptive dimension reduction for clustering high dimensional data. Dans *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, 147–154. `http://dx.doi.org/10.1109/ICDM.2002.1183897`

Dosovitskiy, A. et Brox, T. (2016). Generating images with perceptual similarity metrics based on deep networks. *Advances in neural information processing systems*, *29*.

Durbin, J. (1960). The fitting of time-series models. *Revue de l'Institut International de Statistique*, 233–244.

Erichson, N. B., Zheng, P., Manohar, K., Brunton, S. L., Kutz, J. N. et Aravkin, A. Y. (2020). Sparse principal component analysis via variable projection. *SIAM Journal on Applied Mathematics*, *80*(2), 977–1002.

Esling, P. et Agon, C. (2012). Time-series data mining. *ACM Computing Surveys (CSUR)*, *45*(1), 1–34.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X. *et al.* (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. Dans *kdd*, volume 96, 226–231.

Everitt, B. S., Landau, S., Leese, M. et Stahl, D. (2011). Cluster analysis.

Faloutsos, C. et Lin, K.-I. (1995). Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. Dans *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, 163–174.

Frankl, P. et Maehara, H. (1988). The johnson-lindenstrauss lemma and the sphericity of some graphs. *Journal of Combinatorial Theory, Series B*, *44*(3), 355–362. `http://dx.doi.org/10.1016/0095-8956(88)90043-3`

Fränti, P. et Sieranoja, S. (2019). How much can k-means be improved by using better initialization and repeats? *Pattern Recognition*, *93*, 95–112.

Friedman, J. (2009). The elements of statistical learning: Data mining, inference, and prediction. *(No Title)*.

Fukunaga, K. (2013). *Introduction to statistical pattern recognition*. Elsevier.

Ghojogh, B., Ghodsi, A., Karray, F. et Crowley, M. (2021). Uniform manifold approximation and projection (umap) and its variants: tutorial and survey. *arXiv preprint arXiv:2109.02508*.

Ghosal, A., Nandy, A., Das, A. K., Goswami, S. et Panday, M. (2020). A short review on different clustering techniques and their applications. *Emerging technology in modelling and graphics*, 69–83.

Gisbrecht, A., Schulz, A. et Hammer, B. (2015). Parametric nonlinear dimensionality reduction using kernel t-sne. *Neurocomputing*, *147*, 71–82.

Gomari, D. P., Schweickart, A., Cerchietti, L., Paietta, E., Fernandez, H., Al-Amin, H., Suhre, K. et Krumsiek, J. (2022). Variational autoencoders learn transferrable representations of metabolomics data. *Communications Biology*, *5*(1), 645.

Herrmann, M., Kazempour, D., Scheipl, F. et Kröger, P. (2024). Enhancing cluster analysis via topological manifold learning. *Data Mining and Knowledge Discovery*, *38*(3), 840–887.

Hinton, G. (2003). Stochastic neighbor embedding. *Advances in neural information processing systems*, *15*, 857–864.

Hotelling, H. (1933). Analysis of a complex of statistical variables with principal components. *J. Educ. Psy.*, *24*, 498–520.

Hozumi, Y., Wang, R., Yin, C. et Wei, G.-W. (2021). Umap-assisted k-means clustering of large-scale sars-cov-2 mutation datasets. *Computers in biology and medicine*, *131*, 104264.

Hubert, L. et Arabie, P. (1985). Comparing partitions. *Journal of classification*, *2*, 193–218. `http://dx.doi.org/10.1007/BF01908075`

Hyvärinen, A., Hoyer, P. et Oja, E. (1998). Sparse code shrinkage: Denoising by nonlinear maximum likelihood estimation. *Advances in Neural Information Processing Systems*, *11*.

Hyvärinen, A. et Oja, E. (1997). A fast fixed-point algorithm for independent component analysis. *Neural computation*, *9*(7), 1483–1492.

Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern recognition letters*, *31*(8), 651–666.

Jain, A. K. et Dubes, R. C. (1988). *Algorithms for clustering data*. Prentice-Hall, Inc.

Jimenez, L. O. et Landgrebe, D. A. (2002). Supervised classification in high-dimensional space: geometrical, statistical, and asymptotical properties of multivariate data. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *28*(1), 39–54.

Jolliffe, I. T. et Cadima, J. (2016). Principal component analysis: a review and recent developments. *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences*, *374*(2065), 20150202.

Kanagala, H. K. et Krishnaiah, V. J. R. (2016). A comparative study of k-means, dbscan and optics. Dans *2016 International Conference on Computer Communication and Informatics (ICCCI)*, 1–6. IEEE.

Kedadouche, M., Liu, Z. et Thomas, M. (2016). Bearing fault feature extraction using autoregressive coefficients, linear discriminant analysis and support vector machine under variable operating conditions. Dans *International Conference on Condition Monitoring of Machinery in Non-Stationary Operation*, 339–352. Springer.

Khan, K., Rehman, S. U., Aziz, K., Fong, S. et Sarasvady, S. (2014). Dbscan: Past, present and future. Dans *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*, 232–238. IEEE.

Kingma, D. P. et Welling, M. (2013a). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kingma, D. P. et Welling, M. (2013b). Auto-encoding variational bayes. Récupéré de `https://arxiv.org/abs/1312.6114`

Kruskal, J. B. (1978). Multidimensional scaling. *Murry Hill*.

Kullback, S. et Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, *22*(1), 79–86.

Lötsch, J. et Ultsch, A. (2024). Comparative assessment of projection and clustering method combinations in the analysis of biomedical data. *Informatics in medicine unlocked*, *50*, 101573.

Lucas, J., Tucker, G., Grosse, R. et Norouzi, M. (2019). Understanding posterior collapse in generative latent variable models.

Maaten, L. v. d. et Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, *9*(Nov), 2579–2605.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. Dans *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, volume 5, 281–298. University of California press. Récupéré de `http://projecteuclid.org/euclid.bsmsp/1200512992`

Madhulatha, T. S. (2012). An overview on clustering methods. *arXiv preprint arXiv:1205.1117*.

Makarenkov, V. et Leclerc, B. (1996). Circular orders of tree metrics, and their uses for the reconstruction and fitting of phylogenetic trees. Dans *Mathematical hierarchies and Biology*, 183–208.

Makarenkov, V. et Leclerc, B. (2000). Comparison of additive trees using circular orders. *Journal of Computational Biology*, *7*(5), 731–744.

Makarenkov, V. et Legendre, P. (2000). Improving the additive tree representation of a dissimilarity matrix using reticulations. In *Data analysis, classification, and related methods* 35–40. Springer.

Makarenkov, V., Legendre, P. et Desdevises, Y. (2004). Modelling phylogenetic relationships using reticulated networks. *Zoologica scripta*, *33*(1), 89–96.

McInnes, L., Healy, J. et Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

McLachlan, G. J. (2005). *Discriminant analysis and statistical pattern recognition*. John Wiley & Sons.

Meng, C., Zeleznik, O. A., Thallinger, G. G., Kuster, B., Gholami, A. M. et Culhane, A. C. (2016). Dimension reduction techniques for the integrative analysis of multi-omics data. *Briefings in bioinformatics*, *17*(4), 628–641.

Nanga, S., Bawah, A. T., Acquaye, B. A., Billa, M.-I., Baeta, F. D., Odai, N. A., Obeng, S. K. et Nsiah, A. D. (2021). Review of dimension reduction methods. *Journal of Data Analysis and Information Processing*, *9*(3), 189–231.

Oyewole, G. J. et Thopil, G. A. (2023). Data clustering: application and trends. *Artificial intelligence review*, *56*(7), 6439–6475.

Park, C. H. et Park, H. (2008). A comparison of generalized linear discriminant analysis algorithms. *Pattern Recognition*, *41*(3), 1083–1097.

Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, *2*(11), 559–572.

Permuter, H., Francos, J. et Jermyn, I. (2006). A study of gaussian mixture models of color and texture features for image classification and segmentation. *Pattern recognition*, *39*(4), 695–706.

Pochet, N., De Smet, F., Suykens, J. A. et De Moor, B. L. (2004). Systematic benchmarking of microarray data classification: assessing the role of non-linearity and dimensionality reduction. *Bioinformatics*, *20*(17), 3185–3195.

Portillo, S. K., Parejko, J. K., Vergara, J. R. et Connolly, A. J. (2020). Dimensionality reduction of sdss spectra with variational autoencoders. *The Astronomical Journal*, *160*(1), 45.

Rahmanishamsi, J., Dolati, A. et Aghabozorgi, M. R. (2018). A copula based ica algorithm and its application to time series clustering. *Journal of Classification*, *35*(2), 230–249.

Ramadhani, F., Zarlis, M. et Suwilo, S. (2020). Improve birch algorithm for big data clustering. Dans *IOP conference series: materials science and engineering*, volume 725, p. 012090. IOP Publishing.

Ranzato, M., Huang, F. J., Boureau, Y.-L. et LeCun, Y. (2007). Unsupervised learning of invariant feature hierarchies with applications to object recognition. Dans *2007 IEEE conference on computer vision and pattern recognition*, 1–8. IEEE.

Reynolds, D. A. et Rose, R. C. (2002). Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE transactions on speech and audio processing*, *3*(1), 72–83.

Rezende, D. J., Mohamed, S. et Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. Dans *International conference on machine learning*, 1278–1286. PMLR.

Rodriguez, M. Z., Comin, C. H., Casanova, D., Bruno, O. M., Amancio, D. R., Costa, L. d. F. et Rodrigues, F. A. (2019). Clustering algorithms: A comparative approach. *PloS one*, *14*(1), e0210236. `http://dx.doi.org/10.1371/journal.pone.0210236`

Roh, H. R., Kim, C. S., Lee, Y. et Lee, J. M. (2025). Dimensionality reduction for clustering of nonlinear industrial data: A tutorial. *Korean Journal of Chemical Engineering*, 1–15.

Rosipal, R. et Girolami, M. (2001). An expectation-maximization approach to nonlinear component analysis. *Neural Computation*, *13*(3), 505–510.

Rovira, M., Engvall, K. et Duwig, C. (2022). Identifying key features in reactive flows: A tutorial on combining dimensionality reduction, unsupervised clustering, and feature correlation. *Chemical Engineering Journal*, *438*, 135250.

Rumelhart, D. E., McClelland, J. L., Group, P. R. *et al.* (1986). *Parallel distributed processing, volume 1: Explorations in the microstructure of cognition: Foundations*. The MIT press.

Rykov, A., De Amorim, R., Makarenkov, V. et Mirkin, B. (2024). Inertia-based indices to determine the number of clusters in k-means: An experimental evaluation. ieee access, 12 (december 2023), 11761–11773.

Santos, A. R., Santos, M. A., Baumbach, J., McCulloch, J. A., Oliveira, G. C., Silva, A., Miyoshi, A. et Azevedo, V. (2011). A singular value decomposition approach for improved taxonomic classification of biological sequences. *BMC genomics*, *12*(Suppl 4), S11.

Schölkopf, B., Smola, A. et Müller, K.-R. (1997). Kernel principal component analysis. Dans *International conference on artificial neural networks*, 583–588. Springer.

Scholz, M., Gatzek, S., Sterling, A., Fiehn, O. et Selbig, J. (2004). Metabolite fingerprinting: detecting biological features by independent component analysis. *Bioinformatics*, *20*(15), 2447–2454.

Schubert, E., Sander, J., Ester, M., Kriegel, H. P. et Xu, X. (2017). Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, *42*(3), 1–21.

Schölkopf, B., Smola, A. et Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*(5), 1299–1319. `http://dx.doi.org/10.1162/089976698300017467`

Sneath, P. H. A. et Sokal, R. R. (1973). *Numerical taxonomy. The principles and practice of numerical classification*. W. H. Freeman and Company.

Steinley, D. (2006). K-means clustering: a half-century synthesis. *British Journal of Mathematical and Statistical Psychology*, *59*(1), 1–34.

Strang, G. (2022). *Introduction to linear algebra*. SIAM.

Sun, Y., Kong, L., Huang, J., Deng, H., Bian, X., Li, X., Cui, F., Dou, L., Cao, C., Zou, Q. *et al.* (2024). A comprehensive survey of dimensionality reduction and clustering methods for single-cell and spatial transcriptomics data. *Briefings in Functional Genomics*, *23*(6), 733–744.

Tahiri, N., Fichet, B. et Makarenkov, V. (2022). Building alternative consensus trees and supertrees using k-means and robinson and foulds distance. *Bioinformatics*, *38*(13), 3367–3376.

Tahiri, N., Willems, M. et Makarenkov, V. (2018). A new fast method for inferring multiple consensus trees using k-medoids. *BMC evolutionary biology*, *18*(1), 48.

Tenenbaum, J. B., de Silva, V. et Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*(5500), 2319–2323. `http://dx.doi.org/10.1126/science.290.5500.2319`

Tipping, M. E. et Bishop, C. M. (1999). Mixtures of probabilistic principal component analyzers. *Neural computation*, *11*(2), 443–482.

Tseng, J. C.-H., Tsai, B.-A. et Chung, K. (2023). Sea surface temperature clustering and prediction in the pacific ocean based on isometric feature mapping analysis. *Geoscience Letters*, *10*(1), 42.

Van Der Maaten, L., Postma, E. O., Van Den Herik, H. J. *et al.* (2009). Dimensionality reduction: A comparative review. *Journal of machine learning research*, *10*(66-71), 13.

Vincent, P., Larochelle, H., Bengio, Y. et Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. Dans *Proceedings of the 25th international conference on Machine learning*, 1096–1103.

Wäldchen, J. et Mäder, P. (2018). Machine learning for image based species identification. *Methods in Ecology and Evolution*, *9*(11), 2216–2225.

Wang, J., Xie, X., Shi, J., He, W., Chen, Q., Chen, L., Gu, W. et Zhou, T. (2020). Denoising autoencoder, a deep learning algorithm, aids the identification of a novel molecular signature of lung adenocarcinoma. *Genomics, proteomics & bioinformatics*, *18*(4), 468–480.

Wang, Q., Wang, C., Feng, Z. et Ye, J.-f. (2012). Review of k-means clustering algorithm. *Electronic design engineering*, *20*(7), 21–24.

Wang, Y., Yu, Z., Li, S., Bian, C., Liang, Y., Wong, K.-C. et Li, X. (2023). scbgeda: deep single-cell clustering analysis via a dual denoising autoencoder with bipartite graph ensemble clustering. *Bioinformatics*, *39*(2), btad075.

Wani, A. A. (2024). Comprehensive analysis of clustering algorithms: exploring limitations and innovative solutions. *PeerJ Computer Science*, *10*, e2286.

Willemink, M. J., Koszek, W. A., Hardell, C., Wu, J., Fleischmann, D., Harvey, H., Folio, L. R., Summers, R. M., Rubin, D. L. et Lungren, M. P. (2020). Preparing medical imaging data for machine learning. *Radiology*, *295*(1), 4–15.

Wiskott, L. (2013). Lecture notes on principal component analysis.

Wolfe, J. H. (1970). Pattern clustering by multivariate mixture analysis. *Multivariate behavioral research*, *5*(3), 329–350.

Xie, B., Mu, Y., Tao, D. et Huang, K. (2011). m-sne: Multiview stochastic neighbor embedding. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *41*(4), 1088–1096.

Yang, M., Rajasegarar, S., Rao, A. S., Leckie, C. et Palaniswami, M. (2016). Anomalous behavior detection in crowded scenes using clustering and spatio-temporal features. Dans *International Conference on Intelligent Information Processing*, 132–141. Springer.

Yong, Y., Chongxun, Z. et Pan, L. (2004). A novel fuzzy c-means clustering algorithm for image thresholding. *Measurement science review*, *4*(1), 11–19.

Yu, H. et Yang, J. (2001). A direct lda algorithm for high-dimensional data—with application to face recognition. *Pattern recognition*, *34*(10), 2067–2070.

Yuan, C. et Yang, H. (2019). Research on k-value selection method of k-means clustering algorithm. *J*, *2*(2), 226–235.

Zellinger, M. J. et Bühlmann, P. (2025). Natural language-based synthetic data generation for cluster analysis. *Journal of Classification*, 1–27. http://dx.doi.org/10.1007/s00357-025-09501-w

Zhang, Z., Chow, T. W. et Zhao, M. (2012). M-isomap: Orthogonal constrained marginal isomap for nonlinear dimensionality reduction. *IEEE transactions on cybernetics*, *43*(1), 180–191.

Zheng, Y., Li, Q., Chen, Y., Xie, X. et Ma, W.-Y. (2008). Understanding mobility based on gps data. Dans *Proceedings of the 10th international conference on Ubiquitous computing*, 312–321.

Zong, B., Song, Q., Min, M. R., Cheng, W., Lumezanu, C., Cho, D. et Chen, H. (2018). Deep autoencoding gaussian mixture model for unsupervised anomaly detection. Dans *International conference on learning representations*.

APPENDIX A

A.1 Detailed ARI Scores for Synthetic and Real-World Datasets

This section presents complete ARI results for all clustering algorithms (k-means, AHC, GMM, and OPTICS) applied to synthetic and real-world datasets, with and without dimensionality reduction. The initial dimensionality D was reduced to $k-1$ (where k is the known number of clusters), 25% of D, or 50% of D.

Table A.1 ARI scores for k-means, AHC, GMM, and OPTICS on the Circles synthetic dataset under different dimensionality reduction methods.

	No Reduction	**PCA**			**Kernel PCA**			**VAE**			**Isomap**			**MDS**		
Algorithms		$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**
k-means	0.156	0.162	0.160	0.162	0.301	0.483	0.655	0.150	0.173	0.215	0.410	0.420	0.415	0.161	0.161	0.150
AHC	0.218	0.185	0.185	0.126	0.218	0.251	0.211	0.153	0.220	0.216	0.376	0.571	0.483	0.216	0.208	0.255
GMM	0.118	0.051	0.131	0.121	0.298	0.468	0.310	0.020	0.151	0.056	0.763	0.690	0.518	0.090	0.107	0.100
OPTICS	0.763	0.431	0.800	0.836	0.350	0.803	0.763	0.096	0.538	0.466	0.353	0.545	0.413	0.601	0.808	0.800

Table A.2 ARI scores for k-means, AHC, GMM, and OPTICS on the Moons synthetic dataset under different dimensionality reduction methods.

	No Reduction	**PCA**			**Kernel PCA**			**VAE**			**Isomap**			**MDS**		
Algorithms		$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**
k-means	0.540	0.473	0.510	0.510	0.576	0.560	0.596	0.446	0.416	0.416	0.785	0.778	0.783	0.536	0.405	0.513
AHC	0.575	0.453	0.618	0.471	0.545	0.478	0.476	0.323	0.398	0.366	0.703	0.681	0.788	0.488	0.626	0.496
GMM	0.583	0.405	0.610	0.608	0.555	0.655	0.538	0.346	0.408	0.535	0.876	0.865	0.886	0.503	0.621	0.626
OPTICS	0.500	0.155	0.575	0.568	0.331	0.651	0.616	0.216	0.421	0.101	0.641	0.678	0.735	0.215	0.793	0.201

Table A.3 ARI scores for k-means, AHC, GMM, and OPTICS on the Rodriguez Structured Gaussian synthetic dataset under different dimensionality reduction methods.

	No Reduction	**PCA**			**Kernel PCA**			**VAE**			**Isomap**			**MDS**		
Algorithms		$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**
k-means	0.433	0.347	0.390	0.425	0.504	0.511	0.375	0.374	0.531	0.586	0.680	0.646	0.672	0.346	0.437	0.471
AHC	0.587	0.381	0.552	0.624	0.562	0.563	0.620	0.357	0.515	0.563	0.674	0.643	0.684	0.363	0.572	0.625
GMM	0.423	0.306	0.360	0.406	0.501	0.443	0.443	0.345	0.523	0.330	0.650	0.620	0.641	0.303	0.375	0.395
OPTICS	0.190	0.102	0.180	0.178	0.190	0.155	0.187	0.145	0.238	0.254	0.308	0.234	0.168	0.084	0.127	0.132

Table A.4 ARI scores for k-means, AHC, GMM, and OPTICS on the Repliclust synthetic dataset under different dimensionality reduction methods.

	No Reduction	**PCA**			**Kernel PCA**			**VAE**			**Isomap**			**MDS**		
Algorithms		$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**
k-means	0.674	0.831	0.680	0.790	0.883	0.856	0.886	0.730	0.603	0.600	0.416	0.402	0.420	0.396	0.841	0.903
AHC	0.420	0.858	0.481	0.500	0.801	0.805	0.856	0.318	0.230	0.360	0.333	0.370	0.410	0.375	0.496	0.414
GMM	0.720	0.868	0.800	0.848	0.880	0.855	0.883	0.753	0.786	0.795	0.426	0.455	0.535	0.433	0.901	0.913
OPTICS	0.393	0.466	0.424	0.451	0.625	0.443	0.585	0.216	0.383	0.391	0.320	0.300	0.300	0.413	0.452	0.456

Table A.5 ARI scores for k-means on 20 real-world datasets from the UCI repository, with results reported for each dimensionality reduction method and reduction level.

	No Reduction	PCA			Kernel PCA			VAE			Isomap			MDS		
Dataset		$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**
Breast tissue	0.27	0.29	0.28	0.26	0.19	0.06	0.16	0.27	0.28	0.29	0.30	0.24	0.26	0.27	0.30	0.26
Breast Wisconsin	0.67	0.66	0.67	0.67	0.00	0.00	0.00	0.23	0.58	0.64	0.72	0.70	0.71	0.28	0.67	0.67
Ecoli	0.51	0.51	0.46	0.48	0.51	0.38	0.45	0.51	0.36	0.46	0.51	0.44	0.71	0.51	0.35	0.53
Glass	0.18	0.19	0.24	0.27	0.16	0.04	0.16	0.19	0.15	0.16	0.16	0.17	0.16	0.20	0.21	0.19
Haberman	0.00	0.00	0.00	0.01	0.01	0.01	0.01	0.01	0.01	0.03	0.00	0.00	0.01	0.01	0.01	0.01
Ionosphere	0.17	0.15	0.17	0.17	0.06	0.14	0.14	0.18	0.13	0.14	0.10	0.10	0.10	0.10	0.17	0.17
Iris	0.62	0.62	0.80	0.62	0.60	0.64	0.60	0.59	0.62	0.59	0.65	0.64	0.65	0.61	0.66	0.61
Movement libras	0.40	0.36	0.35	0.37	0.21	0.25	0.20	0.30	0.30	0.35	0.35	0.36	0.37	0.38	0.39	0.37
Musk	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.01	0.00	0.00	0.00	0.00	0.00	0.00
Parkinsons	-0.10	-0.10	-0.10	-0.10	-0.06	-0.01	-0.01	-0.10	-0.10	-0.10	-0.10	-0.10	-0.10	0.00	-0.10	-0.10
Segmentation	0.47	0.43	0.48	0.47	0.47	0.24	0.45	0.47	0.44	0.50	0.44	0.45	0.46	0.47	0.46	0.47
Sonar all	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Spectf	-0.10	-0.10	-0.10	-0.10	-0.06	-0.01	-0.01	-0.10	-0.10	-0.10	-0.10	-0.10	-0.10	0.00	-0.10	-0.10
Transfusion	0.03	0.04	0.04	0.04	0.02	0.02	0.02	-0.01	0.04	0.07	0.07	0.05	0.06	0.06	0.08	0.08
Vehicle	0.08	0.08	0.08	0.07	0.04	0.05	0.05	0.06	0.07	0.05	0.09	0.10	0.10	0.08	0.08	0.08
Vertebral column	0.21	0.24	0.24	0.25	0.01	0.01	0.00	0.19	0.19	0.19	0.23	0.23	0.22	0.22	0.22	0.22
Vowel context	0.12	0.12	0.14	0.11	0.06	0.12	0.10	0.07	0.06	0.10	0.07	0.05	0.07	0.12	0.11	0.12
Wine	0.90	0.90	0.90	0.90	0.77	0.78	0.78	0.63	0.73	0.61	0.83	0.85	0.85	0.82	0.87	0.88
Winequality red	0.10	0.11	0.10	0.11	0.00	0.00	0.00	0.09	0.07	0.08	0.13	0.14	0.15	0.10	0.11	0.11
Yeast	0.16	0.16	0.08	0.11	0.16	0.01	0.01	0.16	0.09	0.12	0.16	0.11	0.13	0.16	0.14	0.18

Table A.6 ARI scores for AHC on 20 real-world datasets from the UCI repository, with results reported for each dimensionality reduction method and reduction level.

	No Reduction	PCA			Kernel PCA			VAE			Isomap			MDS		
Dataset		$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**
Breast tissue	0.39	0.31	0.32	0.26	0.31	0.47	0.44	0.35	0.23	0.14	0.39	0.40	0.42	0.39	0.40	0.36
Breast Wisconsin	0.58	0.59	0.67	0.71	0.65	0.68	0.52	0.49	0.53	0.37	0.72	0.72	0.66	0.25	0.64	0.62
Ecoli	0.52	0.52	0.44	0.46	0.52	0.41	0.35	0.52	0.26	0.38	0.52	0.53	0.55	0.52	0.56	0.45
Glass	0.21	0.19	0.24	0.26	0.18	0.16	0.17	0.17	0.21	0.12	0.18	0.11	0.14	0.20	0.17	0.18
Haberman	0.00	0.00	0.00	0.01	0.00	0.00	0.03	0.00	0.00	0.08	0.00	0.02	0.01	0.01	0.01	0.00
Ionosphere	0.18	0.13	0.21	0.18	0.26	0.20	0.19	0.13	0.19	0.14	0.13	0.08	0.14	0.13	0.19	0.21
Iris	0.62	0.59	0.76	0.59	0.64	0.52	0.64	0.58	0.71	0.58	0.63	0.73	0.66	0.64	0.64	0.64
Movement libras	0.35	0.37	0.37	0.35	0.41	0.42	0.41	0.30	0.34	0.34	0.35	0.32	0.29	0.35	0.37	0.34
Musk	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.02	0.00	0.00	0.00	0.00	0.12	0.00	0.00	0.00
Parkinsons	-0.06	0.14	-0.08	-0.06	0.23	-0.06	0.18	-0.09	-0.09	0.07	0.15	0.13	0.18	0.12	-0.08	-0.08
Segmentation	0.35	0.45	0.45	0.35	0.43	0.40	0.44	0.44	0.47	0.47	0.37	0.34	0.39	0.43	0.45	0.36
Sonar all	0.00	0.00	0.00	0.00	0.01	0.02	0.03	0.07	0.08	0.00	0.00	0.00	-0.07	0.03	-0.05	-0.04
Spectf	0.00	-0.06	-0.08	-0.01	0.02	-0.09	-0.09	0.21	0.17	-0.01	0.12	-0.07	-0.07	0.03	-0.05	-0.04
Transfusion	0.03	0.03	0.03	0.02	-0.00	-0.00	0.00	0.04	0.04	0.10	0.05	0.04	0.03	0.08	0.09	0.09
Vehicle	0.09	0.09	0.09	0.09	0.07	0.09	0.08	0.05	0.07	0.04	0.12	0.04	0.03	0.08	0.09	0.09
Vertebral column	0.35	0.37	0.37	0.39	0.23	0.23	0.25	0.14	0.14	0.35	0.36	0.36	0.36	0.20	0.20	0.31
Vowel context	0.10	0.10	0.12	0.09	0.10	0.08	0.12	0.04	0.07	0.04	0.05	0.04	0.03	0.09	0.09	0.09
Wine	0.78	0.66	0.77	0.79	0.93	0.90	0.85	0.41	0.40	0.39	0.72	0.58	0.66	0.83	0.73	0.79
Winequality red	0.06	0.12	0.12	0.09	0.07	0.10	0.06	0.01	0.06	0.02	0.10	0.18	0.19	0.10	0.08	0.09
Yeast	0.17	0.17	0.08	0.10	0.17	0.06	0.12	0.17	0.06	0.13	0.17	0.11	0.11	0.17	0.11	0.16

Table A.7 ARI scores for GMM on 20 real-world datasets from the UCI repository, with results reported for each dimensionality reduction method and reduction level.

	No Reduction	PCA			Kernel PCA			VAE			Isomap			MDS		
Dataset		$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**
Breast tissue	0.34	0.31	0.29	0.36	0.36	0.36	0.31	0.32	0.32	0.27	0.29	0.23	0.28	0.31	0.27	0.30
Breast Wisconsin	0.77	0.69	0.47	0.48	0.65	0.67	0.71	0.33	0.75	0.64	0.76	0.81	0.59	0.16	0.33	0.13
Ecoli	0.65	0.65	0.39	0.47	0.65	0.33	0.44	0.65	0.47	0.43	0.65	0.45	0.52	0.65	0.49	0.64
Glass	0.18	0.19	0.26	0.24	0.21	0.18	0.17	0.16	0.20	0.18	0.17	0.17	0.16	0.22	0.20	0.24
Haberman	0.10	0.00	0.00	0.13	0.00	0.00	0.00	0.00	0.00	-0.01	0.03	0.03	0.12	0.01	0.01	0.09
Ionosphere	0.18	0.14	0.18	0.17	0.21	0.25	0.25	0.17	0.04	0.14	0.11	0.07	0.08	0.07	0.17	0.17
Iris	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.56	0.46	0.56	0.57	0.57	0.57	0.57	0.57	0.57
Movement libras	0.34	0.41	0.38	0.42	0.37	0.36	0.35	0.25	0.31	0.31	0.32	0.37	0.30	0.36	0.37	0.31
Musk	0.01	0.01	0.01	0.01	-0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.01	0.01	0.00	0.01	0.00
Parkinsons	0.12	-0.09	-0.05	-0.04	0.16	0.16	0.23	-0.08	0.19	-0.09	0.06	0.18	0.22	-0.10	-0.10	-0.09
Segmentation	0.43	0.40	0.45	0.47	0.41	0.38	0.49	0.45	0.42	0.46	0.40	0.53	0.53	0.51	0.44	0.40
Sonar all	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.08	0.01	0.01	0.00	0.00	0.01	0.03	0.01	0.00
Spectf	-0.10	-0.10	-0.09	-0.09	-0.08	-0.10	-0.10	-0.10	-0.10	-0.10	-0.10	-0.09	-0.11	-0.10	-0.08	-0.09
Transfusion	0.03	0.04	0.04	0.05	0.00	0.00	0.01	0.06	0.06	0.00	0.06	0.06	0.07	0.04	0.04	0.03
Vehicle	0.11	0.08	0.11	0.12	0.08	0.07	0.08	0.07	0.10	0.14	0.09	0.12	0.15	0.07	0.09	0.09
Vertebral column	0.11	0.18	0.18	0.16	0.24	0.24	0.24	0.13	0.13	0.17	0.25	0.25	0.22	0.14	0.14	0.14
Vowel context	0.11	0.09	0.13	0.10	0.10	0.10	0.14	0.10	0.07	0.08	0.07	0.05	0.05	0.09	0.10	0.07
Wine	0.91	0.91	0.81	0.90	0.92	0.88	0.95	0.45	0.81	0.74	0.90	0.82	0.90	0.56	0.83	0.90
Winequality red	0.08	0.09	0.10	0.10	0.07	0.08	0.07	0.04	0.12	0.17	0.17	0.19	0.39	0.09	0.16	0.11
Yeast	0.19	0.19	0.12	0.19	0.19	0.07	0.12	0.19	0.13	0.16	0.19	0.12	0.11	0.19	0.11	0.19

Table A.8 ARI scores for OPTICS on 20 real-world datasets from the UCI repository, with results reported for each dimensionality reduction method and reduction level.

	No Reduction	PCA			Kernel PCA			VAE			Isomap			MDS		
Dataset		$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**
Breast tissue	0.25	0.18	0.17	0.23	0.25	0.22	0.20	0.14	0.33	0.33	0.07	0.16	0.07	0.18	0.23	0.18
Breast Wisconsin	0.02	0.41	0.02	0.00	0.37	0.00	0.00	0.45	0.02	0.00	0.34	0.00	0.00	0.04	0.02	0.00
Ecoli	0.31	0.31	0.42	0.29	0.31	0.49	0.34	0.31	0.34	0.38	0.31	0.25	0.50	0.31	0.41	0.00
Glass	0.12	0.07	0.13	0.12	0.24	0.20	0.23	0.14	0.15	0.02	-0.03	-0.03	0.00	0.09	0.12	0.10
Haberman	0.13	0.01	0.01	0.02	0.06	0.06	0.00	0.02	0.02	-0.01	0.01	0.01	0.00	-0.02	-0.02	0.09
Ionosphere	0.68	0.08	0.14	0.66	0.30	0.73	0.00	0.07	0.03	0.03	0.14	0.10	0.18	0.03	0.66	0.00
Iris	0.57	0.61	0.31	0.61	0.60	0.50	0.60	0.50	0.18	0.50	0.56	0.52	0.56	0.53	0.22	0.53
Movement libras	0.03	0.04	0.03	0.03	0.14	0.04	0.04	0.06	0.02	0.08	0.04	0.04	0.04	0.03	0.03	0.03
Musk	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.02	0.00	0.00	0.01	0.03	0.00	0.02	0.00	0.00
Parkinsons	0.04	0.02	0.00	0.03	0.01	0.17	0.00	0.07	0.08	-0.02	0.03	0.03	0.00	-0.02	0.03	0.03
Segmentation	0.25	0.22	0.27	0.33	0.30	0.31	0.32	0.33	0.30	0.46	0.20	0.21	0.27	0.32	0.32	0.34
Sonar all	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.06	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Spectf	-0.10	0.05	-0.10	-0.07	0.15	0.00	0.00	0.27	-0.87	-0.08	0.12	-0.08	0.00	0.17	-0.05	-0.07
Transfusion	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.02	0.06	0.04	0.04	0.03	0.03	0.03	0.00
Vehicle	0.00	0.00	0.00	0.00	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Vertebral column	0.00	0.34	0.34	0.00	0.19	0.19	0.34	0.43	0.43	0.37	0.31	0.31	-0.03	0.29	0.29	-0.02
Vowel context	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.01	0.01
Wine	0.00	0.68	0.45	0.32	0.77	0.85	0.44	0.41	0.65	0.38	0.70	0.80	0.80	0.77	0.00	0.00
Winequality red	0.00	0.00	0.01	0.00	0.00	0.03	0.00	0.00	0.03	0.01	0.01	0.01	0.01	0.01	0.00	0.00
Yeast	0.01	0.01	0.00	0.01	0.01	0.00	0.00	0.01	0.00	0.01	0.01	0.01	0.01	0.01	0.01	0.01

A.2 Statistical analysis of the results using the Wilcoxon test

Table A.9 Wilcoxon signed-rank test results on 20 real-world benchmarks. One-sided test: $H_1 : ARI_{method} > ARI_{baseline}, \alpha = 0.05, n = 20$. The hypothesis H_1 is that a given dimensionality reduction method significantly improves the ARI (after clustering) over no-reduction baseline. P-values shown correspond to the ARI improvements over no-reduction baseline. Significant p-values (i.e., < 0.05) are highlighted in bold.

	PCA			**Kernel PCA**			**VAE**			**Isomap**			**MDS**		
Dataset	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**	$k-1$	**25%**	**50%**
k-means	0.682	0.287	0.571	0.995	0.997	0.997	0.995	0.997	0.974	0.688	0.930	0.500	0.500	0.325	0.312
AHC	0.547	0.213	0.583	0.128	0.621	0.420	0.932	0.872	0.938	0.182	0.561	0.556	0.197	0.517	0.739
GMM	0.924	0.884	0.219	0.352	0.884	0.611	0.977	0.968	0.971	0.877	0.630	0.626	0.914	0.885	0.943
OPTICS	0.431	0.663	0.531	**0.047**	**0.046**	0.330	0.222	0.459	0.377	0.489	0.882	0.749	0.511	0.553	0.973