Unsupervised learning

María Sofía Uribe
Universidad EAFIT
Medellin, Colombia
msuribec@eafit.edu.co

Olga Lucía Quintero Montoya
Universidad EAFIT
Medellin, Colombia
oquinte1@eafit.edu.co

***Abstract*—This paper presents the implementations of five clustering algorithms. We use density-based clustering algorithms such as mountain clustering and subtractive clustering as exploratory algorithms, which we use to select the number of clusters used in algorithms like k means and fuzzy cmeans. We test our workflow with the well-known iris dataset and then assess the performance of the algorithms on real-world data. We perform internal and external validation for each algorithm and present comparisons regarding the performance of each algorithm. We compare the results obtained for the original data and those of lower and higher dimensions obtained through autoencoders.**

***Index Terms*—Nonsupervised, clustering, kmeans, density-based clustering**

I. Introduction

In machine learning, unsupervised learning is a unique approach that unveils hidden patterns and insights from unlabeled data. Unlike its supervised counterpart, unsupervised learning algorithms do not use predefined labels or desired outputs. Instead, they independently identify underlying structures and relationships within the data. This self-guided exploration allows unsupervised learning algorithms to uncover meaningful groupings, extract inherent features, and reveal previously unknown patterns. Unsupervised learning techniques, such as clustering and dimensionality reduction, are powerful tools for exploratory data analysis, enabling researchers to understand complex datasets better. These techniques are precious in scenarios where data labeling is impractical or impossible, such as analyzing customer behavior, market segmentation, or anomaly detection.

Unsupervised clustering, a cornerstone of unsupervised learning, delves into the task of partitioning unlabeled data into distinct groups based on inherent similarities. This technique empowers researchers to uncover hidden structures and patterns within datasets, revealing meaningful groupings without predefined labels. Unsupervised clustering algorithms, such as k-means, hierarchical, and density-based clustering, facilitate data partitioning into clusters with shared properties. These algorithms employ various strategies to assess the proximity of data points, iteratively grouping similar instances while separating dissimilar ones. The resulting clusters represent distinct subgroups within the dataset, each characterized by a shared set of attributes.

The applications of unsupervised clustering span a wide range of domains, including customer segmentation, market analysis, and anomaly detection. In customer segmentation, clustering can identify distinct customer profiles based on purchasing patterns or demographic data. Clustering can reveal market segments with shared preferences or behaviors in market analysis. In anomaly detection, clustering can identify outliers that deviate from expected patterns, potentially signaling fraudulent activity or system malfunctions.

Density-based clustering algorithms represent a significant subset of unsupervised machine-learning techniques designed to identify clusters in complex, high-dimensional datasets based on the concentration of data points within a given space. Unlike centroid-based methods, which rely on distance measures to form clusters, density-based algorithms focus on regions of high data density, distinguishing them from areas of sparsity or noise. Density-based clustering algorithms perform well in datasets with non-uniform distributions; this flexibility allows density-based algorithms to uncover clusters that might be missed by traditional methods, particularly in the presence of outliers or irregularly shaped data.

Centroid-based clustering algorithms are another prominent family of unsupervised machine-learning methods for partitioning data. The central idea is to separate the data into clusters based on the notion of a central representative point, or centroid, for each cluster. K-means clustering, the most well-known algorithm in this category, iteratively assigns data points to clusters by minimizing the data variance within each cluster, thus determining the optimal cluster centroids. These algorithms have a broad range of applications, including data mining, image processing, and document categorization, due to their ability to efficiently handle large datasets. However, they assume that clusters are convex, isotropic, and of equal size, which can limit their effectiveness in scenarios where clusters have non-uniform shapes and sizes. The refinement of centroid-based clustering algorithms is a subject of continuous academic research to improve their adaptability to complex real-world data structures.

Dimensionality reduction techniques play a prominent role in enhancing the effectiveness of clustering algorithms in data analysis. By reducing the dimensionality of data prior to clustering, the surplus of redundant, irrelevant, or noisy features can be alleviated, thus enhancing the purity of the data representation—this refinement in data quality results in more well-defined, representative, and cohesive clusters. Moreover, employing dimensionality reduction mitigates computational complexity, rendering clustering algorithms more computationally tractable while alleviating the challenges posed by

the curse of dimensionality. We can create meaningful visuals and improve interpretability by projecting the data to lower-dimensional spaces. In this paper, we implement our workflow with the original data and data of high and lower dimensions obtained through autoencoders.

In previous papers, we explored autoencoders as a tool for dimensionality reduction, but other techniques for non-linear projections can achieve similar results. UMAP, or Uniform Manifold Approximation and Projection, is a powerful dimensionality reduction technique that has gained popularity in various machine learning and data analysis applications, particularly in clustering tasks. UMAP excels in preserving global and local data structures, making it particularly useful for capturing complex relationships within high-dimensional data. It combines topology and manifold learning concepts to provide a robust and flexible framework for data representation. UMAP's utility in clustering applications lies in its ability to reveal intricate patterns in data, enabling more accurate and meaningful cluster formations. By reducing the dimensionality of the data while preserving its intrinsic structure, UMAP helps uncover hidden clusters and relationships, making it an invaluable tool for unsupervised learning tasks, such as clustering, that rely on understanding the underlying data distribution.

In this paper, we show the implementation of two density-based algorithms, mountain and subtractive clustering, two centroid-based algorithms, K-means and fuzzy cmeans and a novel clustering algorithm based on the universal gravity rule. We compare the results in the original space, lower and higher dimensional spaces obtained through autoencoders and a lower-dimensional space obtained through UMAP. We obtain the number of clusters for the centroid-based methods from the results of the density-based algorithms. We rank the different algorithms according to internal and external validation indexes.

The remainder of this paper is structured as follows: Section 2 describes the methodology; this section presents the data and the algorithms implemented; Section 3 presents the results; and Section 4 presents the analysis. Finally, section 5 presents the conclusions.

II. Methods

A. Density-based algorithms

1) Mountain algorithm: The Mountain Clustering Algorithm, proposed by [1] in 1994, is a data clustering method that operates on the principles of density-based clustering, specifically designed to identify clusters in complex, non-convex, and irregularly shaped data distributions.

The algorithm begins by estimating the data point density throughout the dataset. In this context, density refers to the number of data points in the vicinity of a given point. High-density regions are indicative of potential cluster centers. The clustering algorithm starts by building a grid V; the vertices of the grid are the potential cluster centers, so it is possible to obtain better estimations by building a grid with the appropriate granularity. Now, we compute the value of the mountain function m_i at each vertex $v \in V$ as follows:

$$m_i(v) = \sum_{x \in X} \exp\Big(- \frac{||v - x||^2}{2\sigma^2} \Big) \tag{1}$$

Where x is a feature vector of the space X, and σ is an application-specific constant that affects the height and smoothing of the mountain function. The algorithm identifies local maxima in the estimated density function. We assign the first center to the local maxima. Local maxima represent areas where the data density is at its highest, making them excellent candidates for cluster centers. These local maxima are often termed "peaks."

The algorithm applies a threshold or filtering mechanism to select only the most significant peaks to avoid over-segmentation. The algorithm focuses on the most prominent clusters and ignores noise or minor density variations. The value of the new mountain function is computed as follows.

$$m_{i+1}(v) = m_i(v) - m_i(c_i) \exp\Big(- \frac{||v - c_i||^2}{2\beta^2} \Big) \tag{2}$$

We repeat these steps until the stopping criteria are met; in our case, we stop if the maximum density reaches a tolerance value set by the designer. According to [2] it is appropriate to take $\beta = 1.5\sigma$ or $\beta = 2\sigma$.

2) Subtractive algorithm: Subtractive clustering, proposed by [3] in 1994, is a data clustering algorithm used in unsupervised machine learning to automatically identify and partition data into clusters based on their underlying density distribution. It operates by iteratively selecting cluster prototypes, starting with the data point exhibiting the highest density in the dataset. There is a region of influence (ROI) around each selected prototype, and data points falling within this region have their densities subtracted, gradually decreasing the likelihood of selecting additional prototypes within that area. This process continues until no data point with a density above a predefined threshold remains. Data points are then assigned to the nearest prototype, forming clusters. Subtractive clustering effectively identifies irregularly shaped and varying-sized clusters, making it a valuable tool in data analysis and pattern recognition tasks.

We first calculate a density measure D_i at each object x_j as follows:

$$D_i(x_j) = \sum_{l=1}^{N} \exp\left(- \frac{||x_j - x_l||^2}{(\frac{r_a}{2})^2} \right) \tag{3}$$

We now assign the first center to the object with the maximum density.

$$D_{i+1}(x_j) = D_i(x_j) - D_i(c_i) \exp\left(- \frac{||x_j - c_i||^2}{(\frac{r_b}{2})^2} \right) \tag{4}$$

We repeat these steps until the stopping criteria are met; in our case, we stop if the maximum density reaches a tolerance value set by the designer. We take $r_b = 2r_a$ or $r_b = 1.5r_a$

B. Centroid-based algorithms

K-means is a clustering algorithm that partitions a dataset into K distinct clusters by iteratively refining the positions of cluster centroids and assigning each data point belonging to the cluster with the nearest mean. In mathematical terms, given a set of $x_1, x_2, \ldots, x_N$ data points, K clusters with $c_1, c_2, \ldots, c_K$ centers and a membership matrix U, where uij specifies the assignment of the object x_j to the cluster i. The algorithm aims to minimize a cost function $\jmath$

$$\jmath = \sum_{i=1}^{K} \sum_{j=1}^{N} u_{ij} ||x_j - c_i||^2 \tag{5}$$

where

$$u_{ij} = \begin{cases} 1, & \text{if } ||x_j - c_i||^2 \leq ||x_j - c_l||^2 \text{ for } \mathrm{l} \neq i \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

In this respect, the cost function $\jmath$ is equal to the sum of the squared distances between each object and its nearest cluster center. The optimal center c_i that minimizes the cost function $\jmath$ is the sample mean for the cluster C_i and can be determined based on the entries of membership matrix U as follows:

$$c_i = \frac{1}{N_i} \sum_{j=1}^{N} u_{ij} x_j \tag{7}$$

where $N_i = \sum_{j=1}^{n} u_{ij}$ is the size of the cluster C_i

We initialize the centers randomly and then iteratively follow a three-step process until the cost function converges.

1) Determine the membership matrix U according to 6
2) Assign each data point to the nearest cluster centroid based on a distance measure, according to 7
3) Recalculate the centroids by averaging the positions of all data points assigned to each cluster using equation 5

1) Fuzzy cmeans: The Fuzzy C-Means (FCM) algorithm is a clustering technique used in data analysis and pattern recognition. It extends the traditional K-Means clustering algorithm by allowing data points to belong to multiple clusters with varying degrees of membership, as opposed to a hard assignment in K-Means.

Initially, we randomly assign cluster centers and membership values for each data point. The algorithm proceeds with iterative steps where it updates the cluster centers and membership values. The membership value for each data point is computed as a measure of how close the point is to each cluster center, and the degree of fuzziness is controlled by a parameter called "fuzziness exponent" (usually set to 2).

The centers are calculated as follows:

$$c_i = \frac{\sum_{j=1}^{N} (u_{ij})^m x_j}{\sum_{j=1}^{N} (u_{ij})^m} \tag{8}$$

The entries of the membership matrix U are given by

$$u_{ij} = \frac{1}{\sum_{l=1}^{K} \left(\frac{||c_i - x_j||}{||c_l - x_j||} \right)^{\frac{2}{m-1}}} \tag{9}$$

Where K is the number of clusters.

The cost function that the algorithm tries to minimize is given by

$$\jmath = \sum_{i=1}^{K} \sum_{j=1}^{N} (u_{ij})^m ||x_j - c_i|| \tag{10}$$

We initialize the centers randomly and then iteratively follow a three-step process until the cost function converges.

1) Calculate the cluster centers according to Equation 8
2) Determine the membership matrix U according to 9
3) Compute the cost function J according to Equation 10

FCM is a valuable tool in cases where data points may have ambiguous membership in multiple clusters, and it is particularly useful in applications like image segmentation, where a single pixel can belong to multiple object classes simultaneously.

C. Clustering based on universal gravity rule

Universal gravity rule clustering is an algorithm introduced by [4] in 2015. In their paper, the authors introduce a clustering algorithm inspired by the principles of Newtonian gravity, which exhibits impressive resilience to noise and outliers while also displaying a notable insensitivity to initial centroid placements. This approach treats data points as fixed celestial objects and cluster centroids as movable entities. The celestial objects exert gravitational forces on the movable objects, resulting in the dynamic repositioning of centroids within the feature space. Consequently, the algorithm leverages the law of gravity to identify optimal cluster centroid positions.

Let K be the number of clusters, T the number of iterations, G0 and p the parameters used in the law of gravity, X_j the data point j and Zj be the centroid of the cluster C_j where $j = 1, 2 \ldots, K$. The total gravity force applied to the cluster centroid Zj from all fixed objects within cluster Cj is given by:

$$F_j(t) = \frac{G(t)}{|C_j|} \sum_{X_i \in C_j} r_i \frac{m_i m_j}{R_{ij}^p + \epsilon} \big(X_i(t) - Z_j(t) \big) \tag{11}$$

Where:

- m_i and mj represent the mass values of the fixed object i and agents j, respectively. We take $m_i = mj = 1$
- R_{ij} is the distance between data point i and cluster centroid j
- p tunes the effect of distance on the calculation of the force
- r_i is a randomly generated vector from a uniform distribution in [0,1]
- ϵ is a small positive number to avoid dividing by zero

- The gravitational constant G is a decreasing function of time given by $G(t) = G_0\big(1 - \frac{1}{T}\big)$

For object j, the acceleration is given by:

$$a_j(t+1) = \frac{F_j(t)}{m_j} \tag{12}$$

For object j, the velocity is given by:

$$V_j(t+1) = V_j(t) + a_j(t+1) \tag{13}$$

The algorithm is as follows:

1) Set $t = 0$
2) Randomly initialize the centroids
3) Assign the data to the nearest cluster
4) Compute the total gravity force applied to each cluster centroid from all fixed objects within cluster with equation 11
5) Update the acceleration for each agent with equation 12
6) Update the vvelocity for each agent with equation 13
7) Update cluster centroids, Z_j with equation $Z_j(t+1) = Z_j(t) + V_j(t+1)$

D. Validation indexes

Clustering validation is a crucial aspect of the data analysis process in unsupervised machine learning. It refers to the set of techniques and metrics used to assess the quality and reliability of clustering results. In essence, clustering validation helps determine how well a given clustering algorithm has grouped data points into meaningful and distinct clusters. We achieve this by comparing the intrinsic properties of the clusters, such as cohesion and separation. Various metrics and methods, such as the silhouette score or the Davies-Bouldin index, are employed to measure the effectiveness of clustering algorithms in creating well-defined and internally homogeneous clusters while maintaining sufficient dissimilarity between them. Clustering validation is essential for ensuring that the outcomes of clustering algorithms align with the underlying structure of the data and that the chosen clustering approach is indeed a valuable tool for data exploration and pattern recognition.

Most validation indexes measure separation and cohesion to evaluate the quality of clusters generated by clustering algorithms. These two measures provide insights into how well the clusters are defined and how distinct they are from each other. Cohesion measures how closely related data points within the same cluster are. In other words, it quantifies the compactness or tightness of a cluster. A cluster with high cohesion has data points that are close to each other, indicating that the members of the cluster are more similar to one another in terms of their features or attributes. To calculate cohesion, we typically find the average distance or similarity between all data points within the same cluster. High cohesion suggests that the cluster captures a meaningful and internally homogeneous group of data points, which is desirable in clustering as it indicates that the algorithm has successfully grouped similar data.

Separation, also known as inter-cluster separation, quantifies how distinct or well-separated different clusters are from each other. It measures the dissimilarity between clusters and evaluates whether there is sufficient space or gap between clusters to distinguish them effectively. Separation is typically calculated as the average distance or dissimilarity between data points from different clusters. The idea is to find the average "distance" between the centroids or representative points of each cluster or the minimum distance between any two points from different clusters. High separation indicates that the clustering algorithm has successfully created distinct clusters with well-defined boundaries, ensuring that the data points in one cluster are significantly different from those in other clusters.

In practice, a good clustering algorithm should maximize cohesion within clusters while simultaneously maximizing separation between clusters. The balance between cohesion and separation is critical in achieving meaningful and effective clustering results. High cohesion and high separation indicate that the clusters are well-formed and accurately capture the underlying structure of the data. Various clustering validation metrics, such as the Silhouette Score and Davies-Bouldin index, take both cohesion and separation into account to assess the overall quality of the clustering solution.

External and internal clustering validation are two distinct approaches to evaluating the quality of clustering results in unsupervised machine learning, each serving a unique purpose and using different methods. The primary goal of external validation is to assess the clustering results by comparing them to a ground truth or reference set. This reference set typically contains known cluster assignments, making it suitable for evaluating how well the clustering algorithm has captured the true underlying structure of the data. External validation is particularly useful when the true clusters are known, such as in classification tasks, and it helps measure the clustering algorithm's accuracy in reproducing these known groupings. Metrics commonly used for external validation include the Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), and Fowlkes-Mallows index, among others. These metrics compare the algorithm's cluster assignments to the known, true clusters.

In contrast, internal validation does not rely on any external reference or ground truth. Instead, it evaluates the quality of clustering solely based on the intrinsic characteristics of the clusters formed by the algorithm. Internal validation metrics assess properties like compactness, separation, and cohesion within the clusters to determine the algorithm's ability to create meaningful and well-separated clusters without external guidance. Examples of internal validation metrics are the Silhouette Score, Davies-Bouldin index, and Dunn index, which consider aspects like cluster compactness, separation, and overall coherence. Internal validation is more broadly applicable and does not require knowledge of the true clusters. It can be used when exploring data without labeled ground truth, making it versatile for various unsupervised learning scenarios. The choice between these approaches depends on

the availability of ground truth and the specific goals of the clustering analysis.

E. Internal indexes

To define the different validation indexes, we first present two important concepts

- Sum of Squared Within (SSW): Internal measure used to evaluate the Cohesion of the clusters.

$$SSW = \sum_{i=1}^{K} \sum_{x \in C_i} ||c_i - x|| \tag{14}$$

- Sum of Squared Between (SSB): It is a separation measure used to evaluate intercluster distance (Separation)

$$SSB = \sum_{i=1}^{K} N_i ||c_i - \bar{x}|| \tag{15}$$

Where K is the number of clusters, C_i is the ith cluster, c_i is the centroid of cluster C_i, N_i is the number of elements in cluster C_i and $\bar{x}$ is the average of the data set.

1) Calinski and Harabasz:

$$CH = \frac{SSB/(K-1)}{SSW/(N-K)} = \frac{SSB(N-K)}{SSW(K-1)} \tag{16}$$

Where N is the number of data points, and K is the number of clusters.

The Calinski-Harabasz Index, also known as the Variance Ratio Criterion (VRC) or the C-H index, is a statistical measure used to evaluate the quality of clustering in cluster analysis. It helps determine the optimal number of clusters in a dataset by quantifying the ratio of between-cluster variance to within-cluster variance. The goal is to find the number of clusters that maximizes this ratio, as a higher ratio indicates more distinct and well-separated clusters.

2) Hartigan:

$$H = \log\left(\frac{SSB}{SSW}\right) \tag{17}$$

A higher Hartigan index indicates that the clusters are well-separated and compact. A lower Hartigan index indicates that the clusters are not well-separated or that they are not compact.

3) Hu index:

$$XU = M \log\left(\frac{SSW}{MN^2}\right) + \log(K) \tag{18}$$

Where N is the number of data points, M is the number of features and K is the number of clusters

4) Davies-Bouldin Index: The Davies-Bouldin index (DBI) is a metric used to evaluate the quality of clustering algorithms. It measures the average similarity between each cluster and its most similar cluster, where similarity is defined as the ratio of within-cluster distances to between-cluster distances.

$$DB = \frac{1}{K} \sum_{i=1, i \neq j} max\left(\frac{\sigma_i + \sigma_j}{||c_i - c_j||}\right) \tag{19}$$

Where K is the number of clusters, c_i is the centroid of cluster C_i, and σ_i is the average distance between each point of cluster i and its centroid.

A lower DBI score indicates that the clusters are well-separated and compact. A higher DBI score indicates that the clusters are not well-separated or that they are not compact. Consequently, we aim to minimize the Davies-Bouldin index.

5) Silhouette Index:

$$silhouette(x) = \frac{b(x) - a(x)}{max\{b(x), a(x)\}} \tag{20}$$

Where $a(x)$ is the average distance between x and the rest of the points in the same cluster, and $b(x)$ is the average distance between x and the points of the nearest cluster.

The silhouette coefficient for the clustering is

$$SC = \frac{1}{N} \sum_{i=1}^{N} silhouette(x) \tag{21}$$

A higher silhouette score indicates a better clustering result, where clusters are more distinct and well-separated. A silhouette score close to 0 suggests that clusters may overlap or that the data points are not clearly assigned to one cluster. A negative silhouette score indicates that the data points are likely assigned to the wrong clusters.

F. External indexes

Given a set of N elements S and two partitions of S to compare: X and Y

- TP: The number of pairs of elements that are in the same subset in X and the same subset Y
- FN: The number of pairs of elements that are in different subsets in X and in the same subset in Y
- FP: the number of pairs of elements that are in the same subset in X and different subsets in Y
- TN: the number of pairs of elements that are in different subsets in X and different subsets in Y

Note that the total of pairs is $\frac{N(N-1)}{2}$

1) Rand index:

$$R = \frac{TP + TN}{\frac{N(N-1)}{2}} \tag{22}$$

The Rand Index value ranges from 0 to 1, where 0 indicates no agreement between the two clusterings, meaning that they are completely dissimilar, and 1 indicates perfect agreement between the two clusterings, meaning that they are identical.

2) Jaccard index:

$$R = \frac{TP}{TP + FP + FN} \tag{23}$$

The Jaccard Index values range from 0 to 1: 0 indicates no overlap between the clusters of the two different clusterings, meaning they are completely dissimilar, and 1 indicates perfect overlap, where the two clusterings are identical.

3) Fowlkes-Mallows index:

$$FM = \sqrt{\frac{TP}{TP+F} \cdot \frac{TP}{TP+FN}} \tag{24}$$

The minimum possible value of the Fowlkes–Mallows index is 0, which corresponds to the worst binary classification possible, where all the elements have been misclassified. The maximum possible value of the Fowlkes–Mallows index is 1, which corresponds to the best binary classification possible, where all the elements have been perfectly classified.

G. Autoencoders

An autoencoder is a neural network composed of an encoder and a decoder. The encoder maps the input into a code, and a decoder maps the code to a reconstruction of the original input. The autoencoder described in this section is a multilayer perceptron with one input layer, one hidden layer, and one output layer. The defining aspect of an autoencoder is that the input layer contains as much information as the output layer [5].

Kramer [6] first proposed the autoencoder as a nonlinear generalization of principal components analysis (PCA). The autoencoder has also been called the autoassociator or Diabolo network. Its first applications date to the early 1990s. Autoencoders are helpful because they can significantly reduce the noise of input data, making the creation of deep learning models much more efficient. They can detect anomalies, tackle unsupervised learning problems, and perform feature extraction and dimensionality reduction. Although autoencoders are valuable for dimensionality reduction, they can also increase dimensionality.

In the case of dimensionality reduction, the hidden layer of the autoencoder should specialize the critical features of the training sample. Thus, the outputs of the hidden layer should summarize the data set. In this case, the autoencoder acts as a feature extractor; if we successfully reduce the data into a lower dimensional space, we can train a less complex model to predict the desired output. In mathematical terms, an autoencoder consists of an encoding function ε and a decoding function D, defined as follows.

$$\varepsilon(x) = \phi(w'x + b) \tag{25}$$

$$D(h_x) = \phi(wh_x + b) \tag{26}$$

Where x is the input, h_x is the output of the hidden layer, w and w' are linear transformations, b and b' are the biases vectors and $\phi(.)$ is a nonlinear function

An autoencoder is denoising if the encoding function satisfies with high probability:

$$\varepsilon(D(h_x) + \xi) = h_x \tag{27}$$

Where ξ is drawn from a noise distribution

Figure 1 shows the architecture for the autoencoder with low dimension.

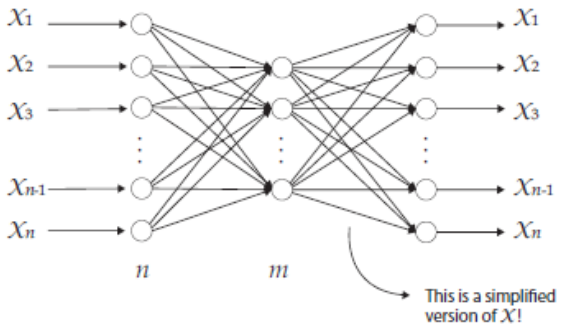

Fig. 1. Architecture for the autoencoder with low dimension

H. UMAP

Uniform Manifold Approximation and Projection (UMAP) is a dimensionality reduction technique proposed by [7] in 2020. This technique is widely used for visualizing high-dimensional data in a lower-dimensional space while preserving its inherent structure. UMAP operates on the principle of manifold learning, seeking to identify the underlying topological structure of the data. It combines elements of both graph-based and algebraic approaches. The key idea behind UMAP is to model the data as a fuzzy topological graph where data points are connected with varying degrees of similarity. UMAP aims to find an embedding in a lower-dimensional space that minimizes the discrepancies between the topological relationships in the original data and the embedded data.

Mathematically, UMAP minimizes a loss function that balances the preservation of local and global structures in the data. UMAP focuses on preserving local neighborhood information. UMAP employs stochastic gradient descent to find the optimal embedding; this technique can reveal complex data structures and relationships in a visually intuitive manner.

I. Data

1) Iris dataset: The Iris dataset is a well-known and frequently used dataset in the field of machine learning and statistics. It was introduced by the British biologist and statistician Ronald A. Fisher in 1936 and has become a popular dataset for practicing classification and clustering techniques. The dataset is named after the iris flower, as it consists of measurements taken from three different species of iris flowers. The Iris dataset contains the following information for 150 iris flowers, with 50 samples from each of the three species:

- Sepal Length: This is the measurement of the length of the iris flower's sepal (the green leaf-like structure that protects the flower). It is typically measured in centimeters.
- Sepal Width: This is the measurement of the width of the iris flower's sepal, also typically in centimeters.
- Petal Length: This is the measurement of the length of the iris flower's petal (the colorful part of the flower). It is typically measured in centimeters.

- Petal Width: This is the measurement of the width of the iris flower's petal, also typically in centimeters.
- Species: This is the target variable and represents the species of the iris flower. There are three species in the dataset: Iris Setosa, Iris Versicolor, and Iris Virginica

J. Customer data

The mall customer segmentation dataset comprises information on a diverse set of shoppers, with key variables that help classify and understand their behavior. This dataset includes demographic details such as age and gender, allowing for the exploration of age-related shopping preferences and gender-based shopping trends. Furthermore, it incorporates financial information, specifically annual income in thousands (USD), which can be instrumental in segmenting customers based on their economic capacity. Lastly, the spending score, a valuable indicator of consumer behavior, is included to gauge each shopper's propensity to spend. By analyzing this dataset, businesses can uncover insights into various customer segments, tailoring their marketing and product offerings to cater to the unique needs and preferences of different groups, thus enhancing the overall shopping experience.

K. Experiments

For each dataset, we apply the following workflow:

1) Perform descriptive analysis of the data to gain insights into its characteristics and ensure that clustering is a suitable approach, which includes understanding the central tendencies and variability in the data. Analyze histograms, box plots, or scatter plots to explore the data's distribution skewness and assess the correlation between variables.
2) Process the data, which includes handling missing values and inconsistencies in the data, handling categorical variables by encoding them (e.g., one-hot encoding) if needed and normalizing the data to prevent certain variables from dominating the clustering process due to their larger scales.
3) Transform the data through techniques to increase or decrease dimensionality, such as autoencoders or UMAP. In the case of autoencoders, we search for the best autoencoder in low dimension; that is, if the data set has M features, we train autoencoders with $1, \ldots, M-1$ neurons in the hidden layer and choose the best model. For the autoencoder in high dimension, we train autoencoders with $M+1, M+2 \ldots$ neurons in the hidden layer and stop when the validation error does not decrease significantly (according to a selected tolerance)
4) Choose a norm; this norm will induce a distance to compute the distance matrix
5) Run multiple instances of the Mountain and Subtractive algorithms with different parameters
6) Select a small set of plausible numbers of clusters
7) Run K-Means, Fuzzy C-Means with each number of clusters in the set previously defined
8) Compute the internal and external cluster validation indexes mentioned in this section

The distances considered for the experiments are:

- Euclidean
- Manhattan
- Mahalanobis
- Minkowski $p = 3$
- Cosine (pseudo-distance)

III. Results

A. Iris (original)

Tables 1 and 2 show the best clustering results for the original iris data set with the mountain and subtractive algorithms. The results suggest that the number of appropriate clusters is between $k = 2$ and $k = 3$. Tables 3 and 4 show the best clustering results for the original iris data set with the kmeans and fuzzy cmeans algorithms. Figures 2, 3, 4 and 5 show examples of clustering results for the original iris dataset with the mountain, subtractive, kmeans and fuzzy cmeans algorithms and various configurations. Although the mountain and subtractive algorithms are exploratory algorithms that we use to obtain the number of clusters, in some configurations, they achieved a similar result to kmeans and fuzzy cmeans; Figure 2 shows that the clusters obtained with the mountain algorithm are similar to the clusters obtained with the k means algorithm in Figure 4. Figure 6 shows the clustering results for one configuration of the universal gravity rule clustering algorithm; this algorithm was sensitive to the gravitational parameters but often achieved similar clusters to those found by the density-based algorithms.

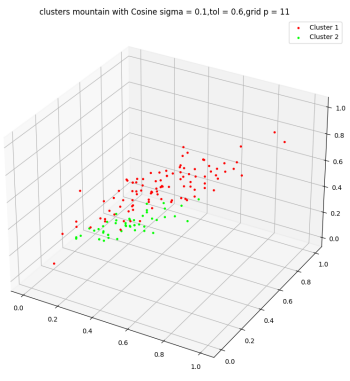

Fig. 2. Original iris data clustered with mountain algorithm

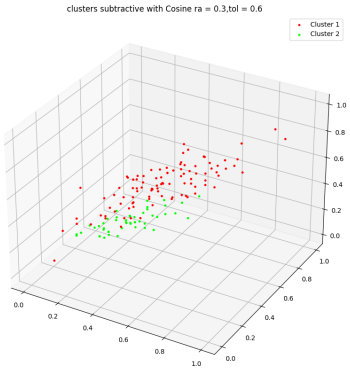

Fig. 3. Original iris data clustered with subtractive algorithm

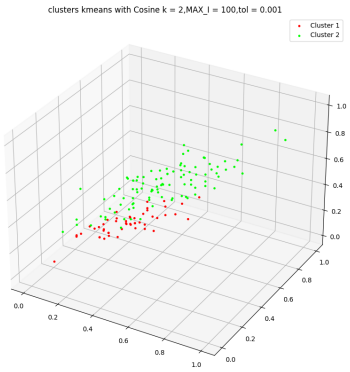

Fig. 4. Original iris data clustered with kmeans

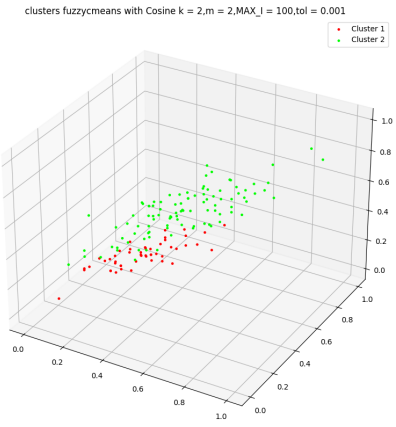

Fig. 5. Original iris data clustered with fuzzy cmeans

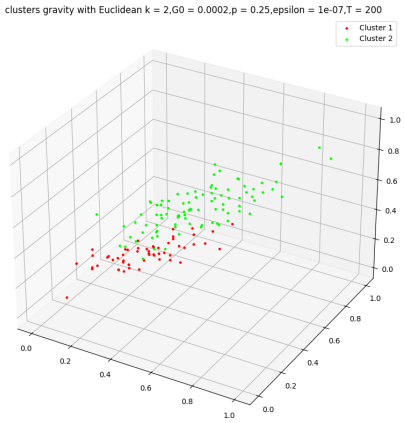

Fig. 6. Original iris data clustered with universal gravity clustering

B. *Iris in low dimension (autoencoder)*

Tables 5 and 6 show the best clustering results for the iris dataset in low dimension (from an autoencoder) with the mountain and subtractive algorithms. The results suggest that the number of appropriate clusters is between $k = 2$ and $k = 3$. Tables7 and 8 show the best clustering results for the iris data set in low dimension (from an autoencoder) with the kmeans and fuzzy cmeans algorithms. Figures 7, 8, 9, 10 and 11 show examples of clustering results for the iris dataset in low dimension (from an autoencoder) with the mountain, subtractive, kmeans, fuzzy cmeans and universal gravity rule clustering algorithms and various configurations. In this case, all algorithms with the shown configurations achieve the same results.

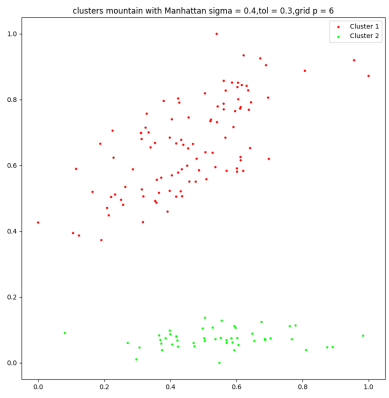

Fig. 7. Iris data in low dimension (from autoencoder) clustered with mountain algorithm

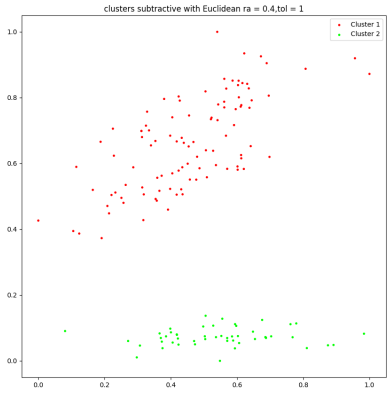

Fig. 8. Iris data in low dimension (from autoencoder) clustered with subtractive algorithm

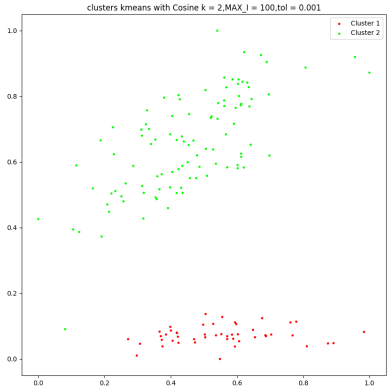

Fig. 9. Iris data in low dimension (from autoencoder) clustered with kmeans

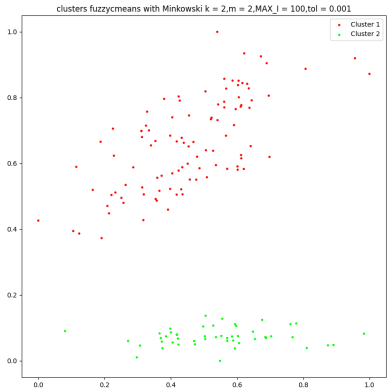

Fig. 10. Iris data in low dimension (from autoencoder) clustered with fuzzy cmeans

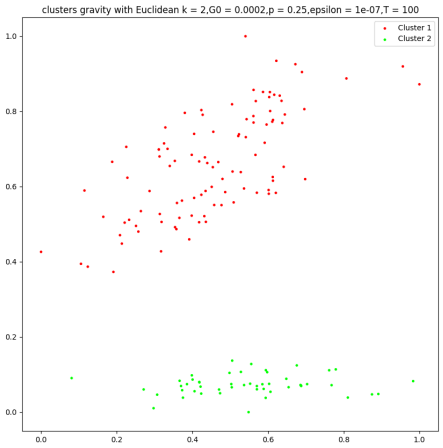

Fig. 11. Iris data in low dimension (from autoencoder) clustered with universal gravity clustering

C. *Iris in high dimension (autoencoder)*

Tables 9 and 10 show the best clustering results for the iris dataset in high dimension (from an autoencoder) with the mountain and subtractive algorithms. The results suggest that the number of appropriate clusters is $k = 2$. Tables11 and 12 show the best clustering results for the iris data set in high dimension (from an autoencoder) with the kmeans and fuzzy cmeans algorithms. Figures 12, 13, 14, 15 and 16 show examples of clustering results for the iris dataset in high dimension (from an autoencoder) with the mountain, subtractive, kmeans, fuzzy cmeans and universal gravity rule clustering algorithms in various configurations. Similarly to the low-dimension case, all algorithms with the shown configurations achieve the same results.

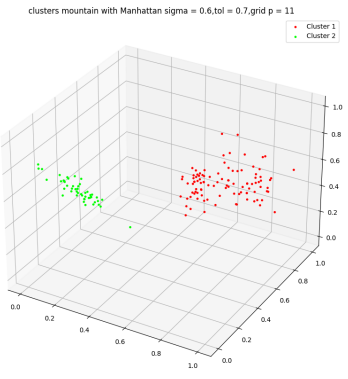

Fig. 12. Iris data in high dimension (from autoencoder) clustered with mountain algorithm

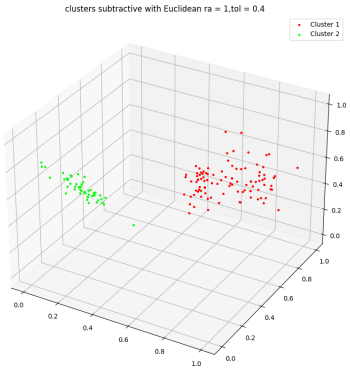

Fig. 13. Iris data in high dimension (from autoencoder) clustered with subtractive algorithm

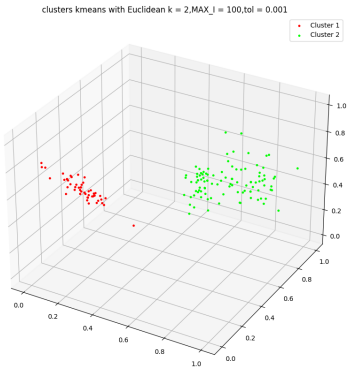

Fig. 14. Iris data in high dimension (from autoencoder) clustered with kmeans

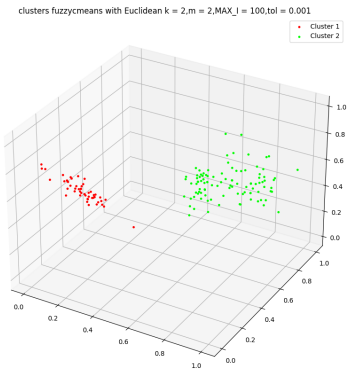

Fig. 15. Iris data in high dimension (from autoencoder) clustered with fuzzy cmeans

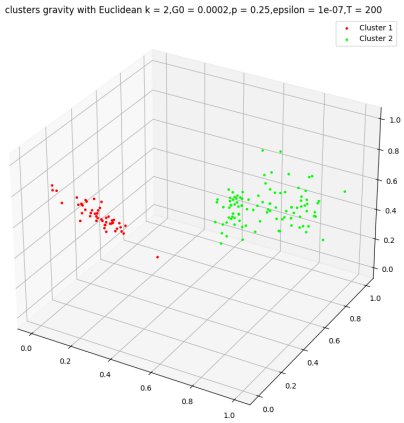

Fig. 16. Iris data in high dimension (from autoencoder) clustered with universal gravity clustering

D. Iris in low dimension (UMAP)

Tables 13 and 14 show the best clustering results for the iris dataset in low dimension (from UMAP) with the mountain and subtractive algorithms. The results suggest that the number of appropriate clusters is between $k = 2$ and $k = 3$. Tables15 and 16 show the best clustering results for the iris data set in low dimension (from UMAP) with the kmeans and fuzzy cmeans algorithms. Figures 17, 18 and 20 and 21 show examples of clustering results for the iris dataset in low dimension (from UMAP) with the subtractive, kmeans, fuzzy cmeans and universal gravity rule clustering algorithms in various configurations. In this case, all algorithms with the shown configurations achieve the same results. Figure 19 shows the clustering results for the iris dataset in low dimension (from UMAP) kmeans set to $k = 3$

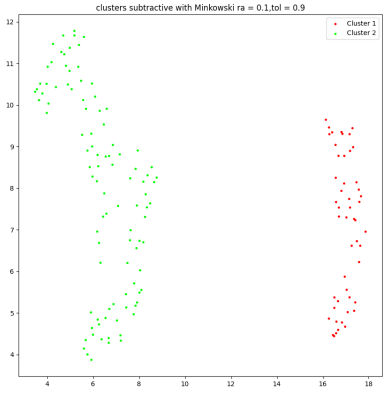

Fig. 17. Iris data in low dimension (from UMAP) clustered with subtractive algorithm

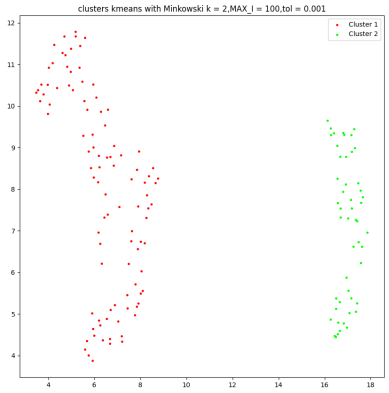

Fig. 18. Iris data in low dimension (from UMAP) clustered with kmeans $k = 2$ algorithm

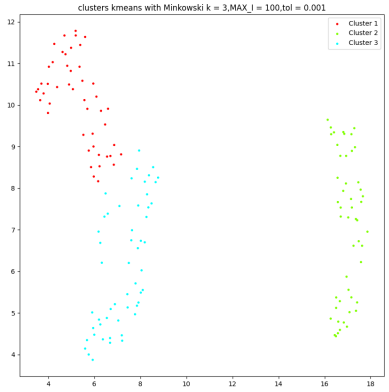

Fig. 19. Iris data in low dimension (from UMAP) clustered with kmeans $k = 3$

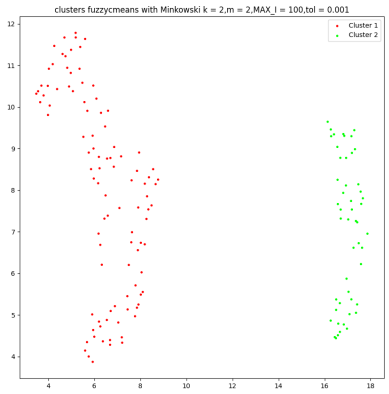

Fig. 20. Iris data in low dimension (from UMAP) clustered with fuzzy cmeans

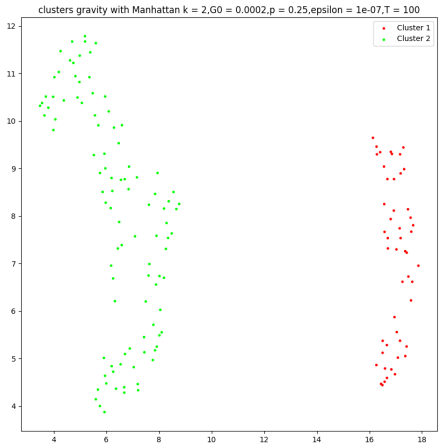

Fig. 21. Iris data in low dimension (from UMAP) clustered with universal gravity clustering

E. Customer segmentation (original data)

Tables 17 and 18 show the best clustering results for the customer's dataset with the mountain and subtractive algorithms. The results suggest that the number of appropriate clusters $k = 2$. Tables 19 and20 show the best clustering results for the customer data set with the kmeans and fuzzy cmeans algorithms. Figures 22, 23, 24, 25 and 26 show examples of clustering results for the customer's data set with the mountain, subtractive, kmeans, fuzzy cmeans and universal gravity rule clustering algorithms in various configurations. In this case, the algorithms achieve similar results with different distances, as can be seen in Figure 22; the mountain algorithm can identify 3 clusters in some configurations.

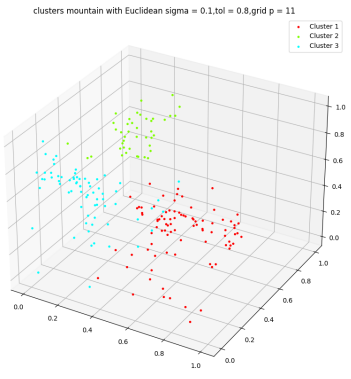

Fig. 22. Original customer segmentation data clustered with mountain algorithms

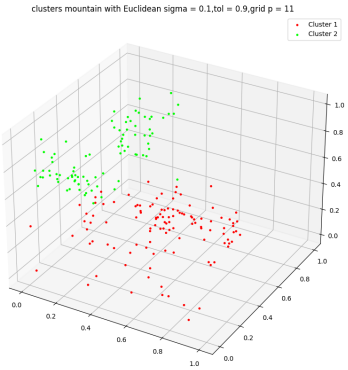

Fig. 23. Original customer segmentation data clustered with mountain algorithm

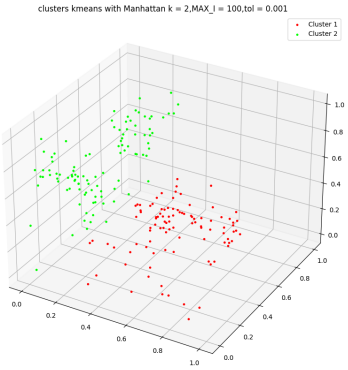

Fig. 24. Original customer segmentation data clustered with kmeans

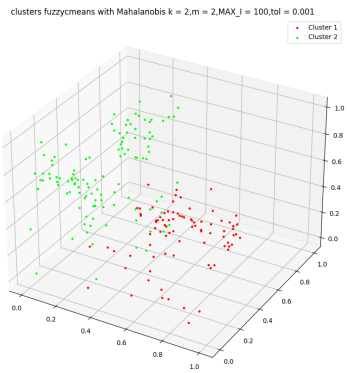

Fig. 25. Original customer segmentation data clustered with fuzzy cmeans

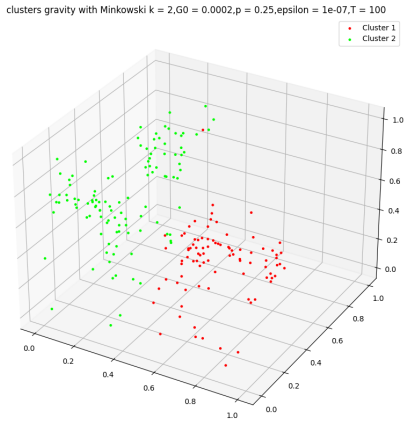

Fig. 26. Original customer segmentation data clustered with universal gravity clustering

F. Customer segmentation in low dimension (autoencoder)

Tables 21 and 22 show the best clustering results for the customer segmentation dataset in low dimensions (from an autoencoder) with the mountain and subtractive algorithms. The results suggest that the number of appropriate clusters is $k = 2$. Tables 23 and24 show the best clustering results for the customer segmentation dataset in low dimensions (from an autoencoder) set with the kmeans and fuzzy cmeans algorithms. Figures 27, 28, 29, 30 and 31 show examples of clustering results for the customers data set with the mountain, subtractive, kmeans, fuzzy cmeans and universal gravity rule clustering algorithms in various configurations. In this case, all the algorithms achieve the same results.

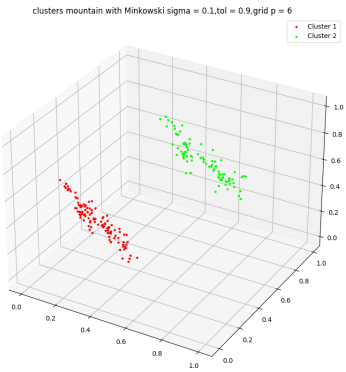

Fig. 27. Customer segmentation data in low dimension (autoencoder) clustered with mountain algorithm

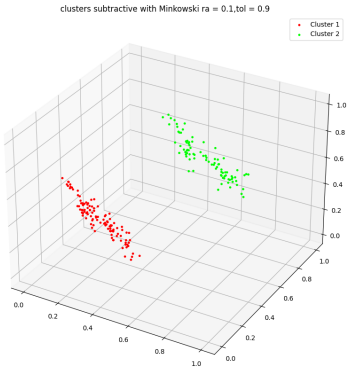

Fig. 28. Customer segmentation data in low dimension (autoencoder) clustered with subtractive algorithm

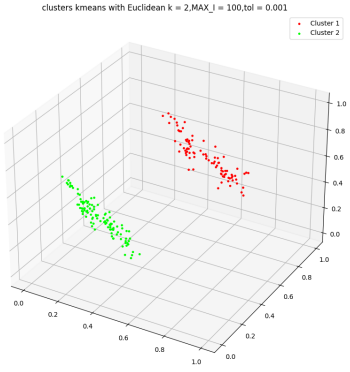

Fig. 29. Customer segmentation data in low dimension (autoencoder) clustered with kmeans

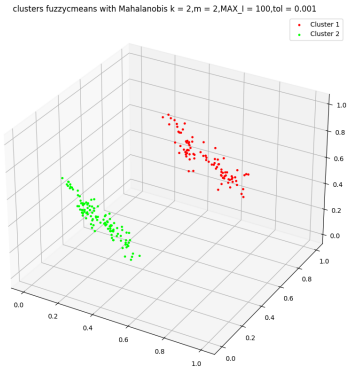

Fig. 30. Customer segmentation data in low dimension (autoencoder) clustered with fuzzy cmeans

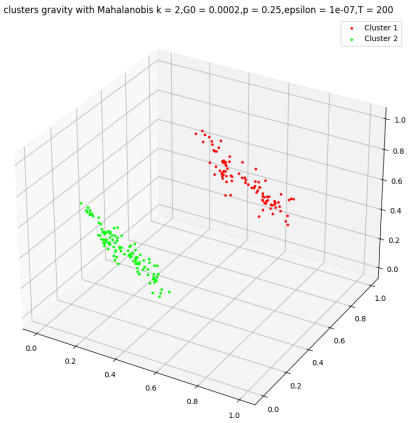

Fig. 31. Customer segmentation data in low dimension (autoencoder) clustered with universal gravity clustering

G. Customer segmentation in high dimension (autoencoder)

Tables 25 and 26 show the best clustering results for the customer segmentation dataset in high dimensions (from an autoencoder) with the mountain and subtractive algorithms. The results suggest that the number of appropriate clusters is $k = 2$. Tables 27 and28 show the best clustering results for the customer segmentation dataset in high dimensions (from an autoencoder) set with the kmeans and fuzzy cmeans algorithms. Figures 32, 33, 34, 35 and 36 show examples of clustering results for the customer's data set with the mountain, subtractive, kmeans, fuzzy cmeans and universal gravity rule clustering algorithms in various configurations. Similarly to the low-dimensional case, all the algorithms achieve the same results.

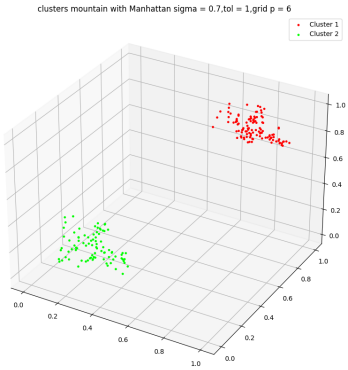

Fig. 32. Customer segmentation data in high dimension (from autoencoder) clustered with mountain algorithm

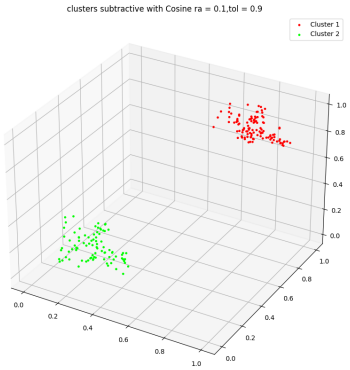

Fig. 33. Customer segmentation data in high dimension (from autoencoder) clustered with subtractive algorithm

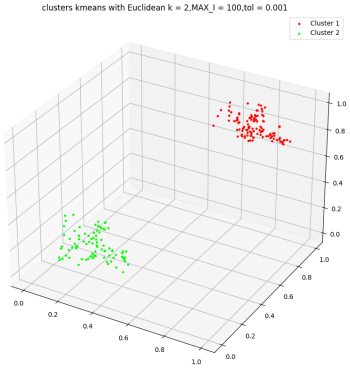

Fig. 34. Customer segmentation data in high dimension (from autoencoder) clustered with kmeans

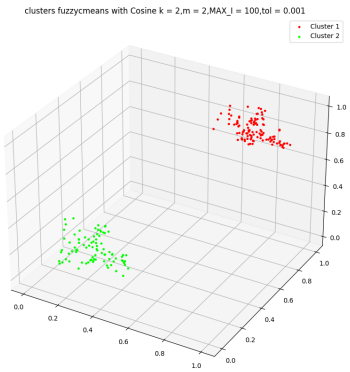

Fig. 35. Customer segmentation data in high dimension (from autoencoder) clustered with fuzzy cmeans

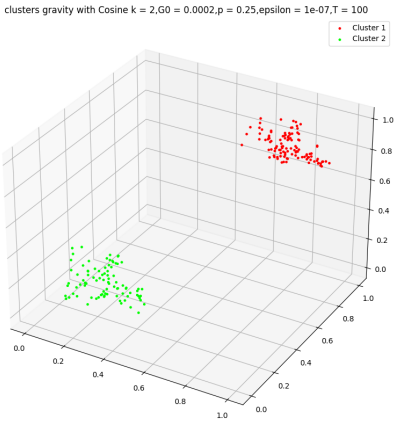

Fig. 36. Customer segmentation data in high dimension (from autoencoder) clustered with universal gravity clustering

H. Customer segmentation in low dimension (UMAP)

Tables 29 and 30 show the best clustering results for the customer segmentation dataset in low dimensions (from UMAP) with the mountain and subtractive algorithms. The results suggest that the appropriate number of clusters is between $k = 2$ and $k = 3$. Tables 31 and32 show the best clustering results for the customer segmentation dataset in low dimensions (from UMAP) set with the kmeans and fuzzy cmeans algorithms. Figures 37, 38, 39, 40 and 41 show examples of clustering results for the customers data set with the mountain, subtractive, kmeans, fuzzy cmeans and universal gravity rule clustering algorithms in various configurations. In this case, the mountain algorithm identifies 2 clusters while the subtractive algorithm identifies 3.

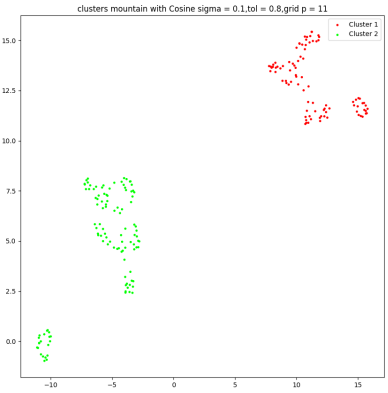

Fig. 37. Customer segmentation data in low dimension (from UMAP) clustered with mountain algorithm

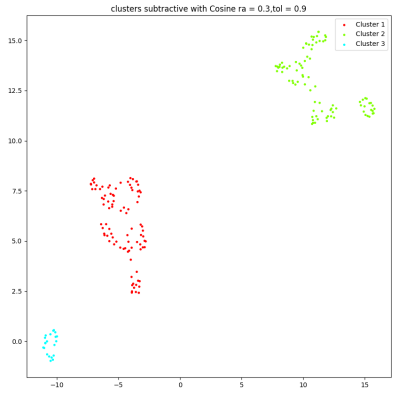

Fig. 38. Customer segmentation data in low dimension (from UMAP) clustered with subtractive algorithm

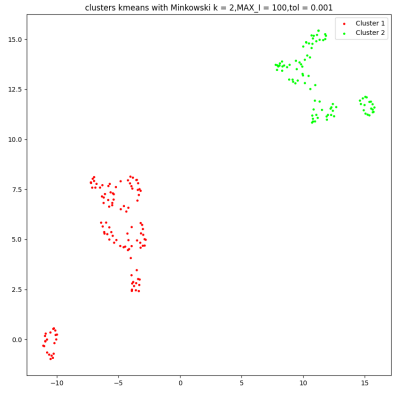

Fig. 39. Customer segmentation data in low dimension (from UMAP) clustered with kmeans

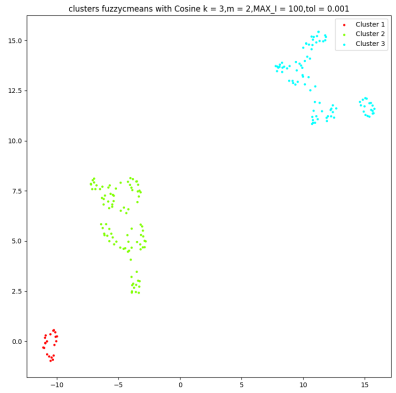

Fig. 40. Customer segmentation data in low dimension (from UMAP) clustered with fuzzy cmeans

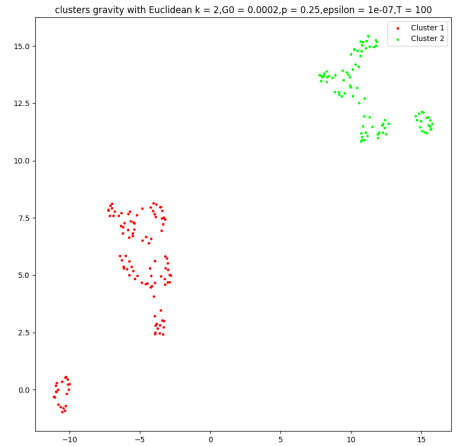

Fig. 41. Customer segmentation data in low dimension (from UMAP) clustered with universal gravity clustering

IV. Discussion

A. Iris dataset

In the results obtained, we see that some configurations of the exploratory algorithms with the Euclidean distance were able to identify 3 clusters; nevertheless, in the majority of configurations, the number of clusters is two rather than 3, as originally indicated in the Iris data set. This behavior occurs in both the original data and when transformed into higher or lower-dimensional spaces with autoencoders and UMAP.

The majority of configurations of the exploratory clustering algorithms suggest the presence of only two clusters in the Iris dataset, which inherently contains three distinct clusters. This discrepancy often arises when the data points from different clusters exhibit substantial overlap or are not well-separated in the feature space. When the Iris dataset, which consists of sepal and petal measurements from three different species of iris flowers, exhibits overlapping feature values between two species, these algorithms may incorrectly group them into a single cluster. Nevertheless, the algorithms do identify 3 clusters in the lower dimensional space obtained with UMAP.

The best overall results, in terms of external validation, were obtained with the centroid-based algorithms, namely, the kmeans and fuzzy cmeans algorithms. The best overall results, in terms of internal validation, were obtained with the density-based algorithms, namely, the mountain and subtractive algorithms.

B. Real customer segmentation data

In the results for the customer segmentation data set, we see that the exploratory algorithms with the Euclidean distance identified 2 clusters in most configurations. This behavior occurs for both the original data and when transformed into higher or lower-dimensional spaces with autoencoders. Nevertheless, when transforming the original space into a lower

dimensional space with UMAP, the subtractive algorithm was able to identify 3 clusters.

Finally, in both cases, it is intriguing to observe that the number of clusters remains consistent across different configurations and distances. The results suggest the potential utility of autoencoders for discerning patterns within high-dimensional data while preserving inter-cluster relationships. Similarly, UMAP offers the capacity to maintain the local structural integrity of clusters when projecting data into a lower-dimensional space.

V. Conclusions

In summary, this paper has explored a diverse array of clustering algorithms, namely Mountain, Subtractive, K-means, Fuzzy C-means and Universal gravity rule, shedding light on their distinct advantages and applications. Mountain clustering focuses on identifying dense regions, which makes it invaluable for tasks like detecting anomalies or locating critical areas within data. Subtractive clustering offers an efficient approach to reducing computational complexity, making it well-suited for larger datasets and scenarios where speed is paramount. K-means clustering remains a workhorse in the field, providing a reliable method for partitioning data into well-defined clusters, thereby aiding in segmentation and pattern recognition. Meanwhile, Fuzzy C-means introduces a level of uncertainty that mirrors real-world ambiguity, making it an ideal choice for applications where objects exhibit multiple characteristics or belong to multiple clusters. The Universal gravity rule algorithm was robust against noise and displayed a notable insensitivity to initial centroid placements.

The selection of a clustering algorithm is not a one-size-fits-all decision but rather a task-specific choice that depends on the nature of the data and the goals of the analysis. Many clustering algorithms rely on distance measures to determine the similarity between data points. In high-dimensional spaces, these distance measures can become less meaningful as all points become approximately equidistant from each other. High-dimensional data also requires more computational resources to perform clustering, which can lead to longer processing times and increased memory requirements. Some clustering algorithms may become computationally infeasible as dimensionality increases. Thus, it is important to explore dimensionality reduction techniques such as TSNE, UMAP, and autoencoders that preserve global and local structures. These techniques allowed us to effectively cluster the well-known iris dataset as well as real-world data efficiently. Although dimensionality reduction methods such as UMAP aim to retain as much vital information as possible while reducing the data's dimensionality, the interpretability of the features will be lost.

In summary, the workflow applied was successful in solving both problems proposed initially. It is appropriate to obtain the number of clusters with exploratory algorithms such as the mountain and subtractive algorithms. These candidate parameters can then be used to run centroid-based methods such as kmeans and fuzzy cmeans. Furthermore, investigating both lower and higher-dimensional data spaces and different clustering algorithms has the potential to yield valuable information about the data's global and local structures. The results presented here were consistent through different algorithms, distances and parameter configurations, thus underscoring the stability and robustness of the employed techniques.

Acknowledgments

I want to acknowledge Professor Olga Lucia Quintero's guidance and insights that aided the refinement and development of this work's content and conceptual framework.

References

[1] R. Yager and D. Filev, "Approximate clustering via the mountain method," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 24, no. 8, pp. 1279–1284, 1994.

[2] H. Mishra, Shuchi, and S. Tripathi, "A comparative study of data clustering techniques," 05 2017.

[3] S. Chiu, "Fuzzy model identification based on cluster estimation," *Journal of the Intelligent and Fuzzy Systems*, vol. 2, pp. 267–278, 01 1994.

[4] A. Bahrololoum, H. Nezamabadi-pour, and S. Saryazdi, "A data clustering approach based on universal gravity rule," *Engineering Applications of Artificial Intelligence*, vol. 45, pp. 415–428, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197615001736

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. The MIT Press, 2017.

[6] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE Journal*, vol. 37, no. 2, p. 233–243, 1991.

[7] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," 2020.

APPENDIX

TABLE 1
BEST RESULTS FOR IRIS DATASET IN MOUNTAIN ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters		
σ = 0.1	Cosine	14292.890	4.570	12.373	0.053	0.763	0.758	0.578	2
σ = 0.6	Manhattan	353.829	1.572	25.050	0.931	0.906	0.859	0.753	3

TABLE 2
BEST RESULTS FOR IRIS DATASET IN SUBTRACTIVE ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters		
$r_a = 0.3$	Cosine	14289.682	4.570	12.529	0.056	0.763	0.758	0.578	2
$r_a = 0.1$	Euclidean	188.835	0.944	23.273	1.362	0.957	0.936	0.879	3

TABLE 3
BEST RESULTS FOR IRIS DATASET IN K MEANS ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters		
$k = 3$	Euclidean	359.845	1.588	22.255	0.760	0.874	0.811	0.682	3
$k = 2$	Cosine	12032.981	4.398	11.901	0.053	0.776	0.771	0.595	2

TABLE 4
BEST RESULTS FOR IRIS DATASET IN FUZZY CMEANS ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters		
$k = 2$, $m = 2$	Mahalanobis	11.154	-2.585	30.727	1.702	0.772	0.764	0.587	2
$k = 3$, $m = 2$	Euclidean	159.319	0.774	23.872	0.333	0.874	0.809	0.680	3

TABLE 5
BEST RESULTS FOR IRIS DATASET LOW DIMENSION (AUTOENCODER) IN MOUNTAIN ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters		
$\sigma = 0.5$	Euclidean	213.457	0.366	19.097	0.683	0.776	0.771	0.595	2
$\sigma = 0.6$	Mahalanobis	61.505	-0.878	23.032	1.527	0.763	0.758	0.578	2

TABLE 6
BEST RESULTS FOR IRIS DATASET LOW DIMENSION (AUTOENCODER) IN SUBTRACTIVE ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters		
ra = 0.2	Euclidean	246.615	1.211	18.192	1.024	0.880	0.820	0.694	3
ra = 0.1	Minkowski	2096.190	3.351	15.066	0.645	0.910	0.864	0.760	3

TABLE 7
BEST RESULTS FOR IRIS DATASET IN LOW DIMENSION (AUTOENCODER) K MEANS ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters		
$k = 2$	Cosine	12032.981	4.398	11.901	0.053	0.776	0.771	0.595	2
$k = 2$	Cosine	66.003	-0.808	16.008	0.344	0.776	0.771	0.595	2

TABLE 8
BEST RESULTS FOR IRIS DATASET IN LOW DIMENSION (AUTOENCODER) IN FUZZY CMEANS ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters		
$k = 2$, $m = 2$	Mahalanobis	11.154	-2.585	30.727	1.702	0.772	0.764	0.587	2
$k = 2$, $m = 2$	Mahalanobis	30.524	-1.579	23.150	0.977	0.776	0.771	0.595	2

TABLE 9
BEST RESULTS FOR IRIS DATASET HIGH DIMENSION (AUTOENCODER) IN MOUNTAIN ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters		
$\sigma = 0.7$	Euclidean	104.001	-0.353	40.251	10.800	0.549	0.515	0.330	2
$\sigma = 0.5$	Manhattan	64.545	-0.830	44.103	24.966	0.445	0.520	0.316	2

TABLE 10
BEST RESULTS FOR IRIS DATASET HIGH DIMENSION (AUTOENCODER) IN SUBTRACTIVE ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters		
$r_a = 0.1$	Cosine	5165.423	3.553	14.703	0.089	0.776	0.771	0.595	2
$r_a = 0.5$	Manhattan	125.329	0.534	38.291	1.457	0.846	0.777	0.635	3

TABLE 11
BEST RESULTS FOR IRIS DATASET IN HIGH DIMENSION (AUTOENCODER) K MEANS ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters		
$k = 3$	Euclidean	456.775	1.827	27.174	0.790	0.892	0.837	0.720	3
$k = 3$	Euclidean	456.775	1.827	27.174	0.790	0.892	0.837	0.720	3

TABLE 12
BEST RESULTS FOR IRIS DATASET IN HIGH DIMENSION (AUTOENCODER) IN FUZZY CMEANS ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters		
k = 3, m = 2	Minkowski	1441.034	2.976	23.634	0.141	0.906	0.858	0.751	3
k = 3, m = 2	Minkowski	1441.034	2.976	23.634	0.141	0.906	0.858	0.751	3

TABLE 13
BEST RESULTS FOR IRIS DATASET LOW DIMENSION (UMAP) IN MOUNTAIN ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters		
$\sigma = 0.1$	Cosine	1101.508	2.007	9.052	0.309	0.446	0.546	0.333	2
$\sigma = 0.1$	Cosine	1101.508	2.007	9.052	0.309	0.446	0.546	0.333	2

TABLE 14
BEST RESULTS FOR IRIS DATASET LOW DIMENSION (UMAP) IN SUBTRACTIVE ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters		
$r_a = 0.3$	Minkowski	29.199	-0.923	19.911	0.237	0.950	0.923	0.858	3
$r_a = 0.3$	Minkowski	29.199	-0.923	19.911	0.237	0.950	0.923	0.858	3

TABLE 15
BEST RESULTS FOR IRIS DATASET IN LOW DIMENSION (UMAP) K MEANS ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters		
$k = 3$	Cosine	5885.695	4.383	5.255	0.160	0.957	0.935	0.878	3
$k = 3$	Manhattan	106.312	0.369	16.894	0.552	0.957	0.936	0.879	3

TABLE 16
BEST RESULTS FOR IRIS DATASET IN LOW DIMENSION (UMAP) IN FUZZY CMEANS ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters		
$k = 3$, $m = 2$	Minkowski	62.444	-0.163	18.906	0.069	0.957	0.935	0.879	3
$k = 3$, $m = 2$	Minkowski	62.444	-0.163	18.906	0.069	0.957	0.935	0.879	3

TABLE 17
BEST RESULTS FOR CUSTOMERS DATASET IN MOUNTAIN ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters
sigma = 0.4,tol = 0.9,grid p = 11	Minkowski	1545.31328	2.05471491	28.9235235	0.19075068	0.84390081	2
sigma = 0.5,tol = 0.2,grid p = 11	Minkowski	1545.31328	2.05471491	28.9235235	0.19075068	0.84390081	2

TABLE 18
BEST RESULTS FOR CUSTOMERS DATASET IN SUBTRACTIVE ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters
ra = 0.6,tol = 0.4	Minkowski	1370.20738	1.93445035	29.2323353	0.197804	0.84390081	2
ra = 0.3,tol = 0.6	Cosine	1793.12215	2.20344657	22.5536442	0.15994382	0.85729402	2

TABLE 19
BEST RESULTS FOR CUSTOMERS DATASET IN KMEANS ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters
k = 2,MAX_I = 100,tol = 0.001	Minkowski	1570.55065	2.07091454	28.8829599	0.18920884	0.84390081	2
k = 2,MAX_I = 100,tol = 0.001	Cosine	1745.96721	2.17679693	22.3852856	0.15876442	0.85729402	2

TABLE 20
BEST RESULTS FOR CUSTOMERS DATASET IN FUZZY CMEANS ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters
k = 2,m = 2,MAX_I = 100,tol = 0.001	Cosine	884.347627	1.4965832	24.078691	0.08517941	0.85004539	2
k = 2,m = 2,MAX_I = 100,tol = 0.001	Minkowski	799.752196	1.39603489	30.5675128	0.10218456	0.83491483	2

TABLE 21
BEST RESULTS FOR CUSTOMERS DATASET LOW DIMENSION (AUTOENCODER) IN MOUNTAIN ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters
ra = 0.7,tol = 0.9	Cosine	2557.29476	2.55843822	20.160283	0.13923932	0.87329488	2
ra = 0.9,tol = 0.7	Minkowski	1399.5749	1.9556568	24.6879432	0.15137852	0.86646382	2

TABLE 22
BEST RESULTS FOR CUSTOMERS DATASET LOW DIMENSION (AUTOENCODER) IN SUBTRACTIVE ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters
sigma = 0.8,tol = 0.6,grid p = 11	Minkowski	75.3850452	-0.96565811	31.7565568	1.51755199	0.86646382	2
sigma = 0.4,tol = 0.6,grid p = 11	Manhattan	131.89783	-0.40623942	31.6964711	2.12391932	0.68068272	2

TABLE 23
BEST RESULTS FOR CUSTOMERS DATASET LOW DIMENSION (AUTOENCODER) IN KMEANS ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters
k = 2,MAX_I = 100,tol = 0.001	Cosine	5522.22975	3.32826997	19.925845	0.10294201	0.89802661	2
k = 2,MAX_I = 100,tol = 0.001	Minkowski	1817.4071	2.21689906	23.8116748	0.12843721	0.8865126	2

TABLE 24
BEST RESULTS FOR CUSTOMERS DATASET LOW DIMENSION (AUTOENCODER) IN FUZZY CMEANS ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters
k = 2,m = 2,MAX_I = 100,tol = 0.001	Minkowski	928.958275	1.54579679	25.1576812	0.06829099	0.88137544	2
k = 2,m = 2,MAX_I = 100,tol = 0.001	Cosine	2808.30584	2.65206965	21.2796365	0.0541524	0.89395462	2

TABLE 25
BEST RESULTS FOR CUSTOMERS DATASET HIGH DIMENSION (AUTOENCODER) IN MOUNTAIN ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters
sigma = 0.1,tol = 0.6,grid p = 11	Cosine	297.838922	0.40828578	35.7016006	1.26104696	0.7209408	2
sigma = 0.1,tol = 0.9,grid p = 11	Minkowski	71.4340639	-1.01949219	46.008647	1.92955269	0.83100677	2

TABLE 26
BEST RESULTS FOR CUSTOMERS DATASET HIGH DIMENSION (AUTOENCODER) IN SUBTRACTIVE ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters
ra = 0.4,tol = 0.8	Cosine	1709.75596	2.15583889	28.2651873	0.14078754	0.86566462	2
ra = 0.6,tol = 1	Minkowski	650.871652	1.19004544	37.0778438	0.21623743	0.83100677	2

TABLE 27
BEST RESULTS FOR CUSTOMERS DATASET IN HIGH DIMENSION (AUTOENCODER) K MEANS ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters
k = 2,MAX_I = 100,tol = 0.001	Cosine	487.763837	0.90156432	30.8717354	0.29198101	0.7751745	2
k = 2,MAX_I = 100,tol = 0.001	Minkowski	395.651769	0.69226742	42.6636119	0.30866184	0.76507094	2

TABLE 28
BEST RESULTS FOR IRIS DATASET IN HIGH DIMENSION (AUTOENCODER) IN FUZZY CMEANS ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters
k = 2,m = 2,MAX_I = 100,tol = 0.001	Cosine	262.087031	0.2804096	33.0090694	0.15824674	0.75497332	2
k = 2,m = 2,MAX_I = 100,tol = 0.001	Minkowski	213.898281	0.07723355	44.8415027	0.17170415	0.74356283	2

TABLE 29
BEST RESULTS FOR IRIS DATASET LOW DIMENSION (UMAP) (AUTOENCODER) IN MOUNTAIN ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters
sigma = 0.1,tol = 0.8,grid p = 11	Cosine	11.6504958	-2.83291829	13.9106917	1.51571808	0.91697677	2
sigma = 0.2,tol = 1,grid p = 11	Cosine	43.2013513	-1.52239526	14.2835342	0.84922148	0.3639787	2

TABLE 30
BEST RESULTS FOR CUSTOMERS DATASET LOW DIMENSION (UMAP) IN SUBTRACTIVE ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters
ra = 0.3,tol = 0.9	Cosine	118744.841	7.09467573	7.81998834	0.03340505	0.96090263	3
ra = 0.6,tol = 1	Minkowski	22.2191147	-2.18731409	22.7840578	0.11363857	0.94095767	2

TABLE 31
BEST RESULTS FOR CUSTOMERS DATASET IN LOW DIMENSION (UMAP) K MEANS ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters
k = 2,MAX_I = 100,tol = 0.001	Minkowski	50.3337746	-1.36959072	21.6174925	0.06074438	0.94095767	2
k = 2,MAX_I = 100,tol = 0.001	Cosine	4775.15902	3.18291552	10.9408696	0.07628135	0.91697677	2

TABLE 32
BEST RESULTS FOR CUSTOMERS DATASET IN LOW DIMENSION (UMAP) IN FUZZY CMEANS ALGORITHM

Parameters	Distance	CH	Hartigan	xu	DB	S	Clusters
k = 2,m = 2,MAX_I = 100,tol = 0.001	Minkowski	25.1728813	-2.06249975	22.3097511	0.03307313	0.93947432	2
k = 3,m = 2,MAX_I = 100,tol = 0.001	Cosine	37176.3423	5.93337133	8.88677766	0.0145295	0.95840582	3