Unsupervised learning: clustering methods on multidimensional spaces

Salomón Cardeño Luján
Mathematical Engineering
School of Applied Sciences and Engineering
EAFIT University
scardenol@eafit.edu.co

***Abstract*—In this project, we implemented five clustering methods: density-based (mountain and subtractive) and distance-based (k-means, fuzzy c-means, and k-medoids). We evaluated the performance of these methods on two different datasets. The first dataset was the well-known iris dataset, which was used in a synthetic experiment to assess the quality of the clustering results with six validation indexes: three internal validation indexes (Davies-Bouldin, Dunn, and Calinski-Harabasz) and three external validation indexes (Rand, Jaccard, and Fowlkes-Mallows). For the second experiment, we used real data in three different spaces (original data, lower-dimensional data space obtained with UMAP, and higher-dimensional data space obtained with an autoencoder) to explore the performance of the density-based methods and use them as a heuristic to find the optimal number of clusters k for each of the three distance-based methods. To search for the optimal number of clusters k, we performed a grid-like search for the hyperparameters that resulted in multiple candidates for each resulting configuration. The optimal number of clusters k was selected based on the heuristic selection of the most repeated instance that resulted as the best on all the possible combinations of sorting or ordering based on the ideal index values for each of the four distances (Manhattan, Euclidean, Mahalanobis, and Cosine). Finally, the results of the distance-based methods were assessed by comparing the obtained centers and clusters for each of the three distance-based methods using visualizations.**

***Keywords*—Clustering, Density-based methods, Distance-based methods, Data space, Validation indexes**

I. Introduction

Unsupervised learning is a machine learning approach that involves identifying patterns and structures in data without any explicit labels or guidance from a pre-existing set of features. Clustering is one of the most commonly used techniques in unsupervised learning, where the goal is to partition data points into groups or clusters based on their similarity. Clustering has a long history in data analysis, dating back to the early days of statistics and exploratory data analysis [1]. In recent years, with the rapid growth of data-driven technologies, clustering has become even more relevant and essential in a variety of fields, from biology to finance to social network analysis. With the increasing availability of large and complex datasets, clustering algorithms are becoming more sophisticated and scalable, enabling us to analyze and extract valuable insights from the vast amount of data that is being generated.

Clustering algorithms have found applications in a wide range of fields. In biology, clustering techniques have been used to group genes with similar expression patterns [2]. In computer science, clustering has been used for document retrieval, image processing, and anomaly detection [3]. In social sciences, clustering has been used for identifying different groups of individuals based on their demographic characteristics or behaviors [4]. In marketing, clustering has been used for customer segmentation [5]. In addition to these applications, clustering is also widely used in data mining, machine learning, and pattern recognition [6].

Density-based and distance-based methods are two fundamental families of clustering algorithms that have been widely studied and applied [7], [8]. Density-based methods, such as DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [9], are based on the idea that clusters are dense regions of data points separated by regions of lower density. These methods define clusters by finding high-density areas of data while considering low-density areas as noise. On the other hand, distance-based methods, such as k-means [10] and fuzzy c-means [11], group together the data points that are closer to each other in space. These methods are based on the idea that points within a cluster are closer to each other than to points outside of the cluster. In contrast to density-based methods, distance-based methods do not have a clear way of handling noise points and may produce clusters of arbitrary shape. Despite their fundamental differences, both families of methods have been successfully applied in a variety of fields and contexts, making them important tools in unsupervised machine learning.

Recent advancements in machine learning have shown that the clustering of data is not only limited to the original high-dimensional space in which it was collected. Rather, it is possible to use a lower-dimensional space obtained by a non-linear projection method or a higher-dimensional space obtained by an autoencoder to find more meaningful structures in the data [12]. One of the most popular non-linear projection methods used for clustering is UMAP (Uniform Manifold Approximation and Projection) [13]. UMAP has been shown to be useful for data exploration, visualization, and clustering in lower-dimensional spaces [14]. On the other hand, autoencoders have been used to learn a higher-dimensional representation of the original data space [15]. Autoencoders have been used for various applications in machine learning, such as feature extraction and clustering. In this project, we will explore the use of UMAP and autoencoders to find more meaningful structures in the data and improve the performance of density-based and distance-based clustering methods.

In this study, we implemented and evaluated five clustering methods, including three distance-based methods (k-means, fuzzy c-means, and k-medoids) and two density-based methods (mountain and subtractive). The performance of these methods was assessed using synthetic and real datasets. For the synthetic dataset, we used the well-known iris dataset and six validation indexes to evaluate the quality of the clustering results. For the real dataset, we used data in three different spaces: the original data space, a lower-dimensional space obtained with UMAP, and a higher-dimensional space obtained with an autoencoder. The density-based methods were used as a heuristic to find the optimal number of clusters k for each of the three distance-based methods. The results of the clustering methods were compared and evaluated using visualizations. The goal of this study is to provide a comprehensive evaluation of these clustering methods and to compare their performance on different types of datasets and in different data spaces.

Moving forward, we will present the methodology used in our study, including the clustering methods, data sets, and experiments. We will then present our results and analyses, comparing the performance of different methods for each experiment and discussing their implications. Finally, we will provide our conclusions in contrast to our findings.

II. Methodology

A. Density-based methods

1) Mountain clustering:

The main idea of this approach is to find cluster centers based on a density measure. For this particular method, the density measure is often called the *mountain function* and represents the height of the function at a certain point from a constructed grid space. Before even considering the density function, we need to build a grid space that is determined by the number of dimensions of the data space and the granularity or number of points per dimension. We denote that grid space as V. The density measure m measures the influence of the data points x_i at a point v from the grid space and is defined as

$$m(v) = \sum_{i=1}^{N} \exp\left(-\frac{\|v - x_i\|^2}{2\sigma^2}\right)$$

where v is a point from the grid space (set of points) V, x_i is the ith feature vector with $i = 1, 2, ..., N$ and σ represents the height and the smoothing of the mountain function; is an application specific parameter, that acts as the kernel influence or bandwidth for the mountain function construction and update. The first center is selected as the maximum height $\max\{m(v)\}$. Afterward, obtaining the next center cluster requires eliminating the effect of the last center. So for every iteration $j \geq 2$, the mountain function is ”updated” or recalculated as

$$m_j(v) = m_{j-1}(v) - m(c_{j-1}) \exp\left(-\frac{\|v - x_i\|^2}{2\beta^2}\right)$$

and every new center is the maximum height of the actual iteration, i.e., $c_j = \max\{m_j(v)\}$. Note that $m(v)_1$ and c_1 are calculated with the first mountain density function, that c_{j-1} and $m_{j-1}(v)$ are the center and mountain function from the last iteration, respectively, and that the purpose of β is similar to σ but in the sense of the region of influence to be removed from the new density measure. Usually $\beta = 2\sigma$ or in some cases $\beta = 1.5\sigma$ as stated by [16]. As a final note, observe that the density depends on the selected norm, which would have different results depending on the norm choice or definition.

2) Subtractive clustering:

The idea of the subtractive method is the same as the mountain method. The only differences are that instead of requiring the construction of a grid space it measures the density at every data point in relation to every other data point, and also that the density measure does not have any special name as the *mountain function* for the previous method. The density measure D and the density measure D_k at any iteration k are defined as follows

$$D(x_i) = \sum_{j=1}^{N} \exp\left(-\frac{\|x_i - x_j\|^2}{2(r_a/2)^2}\right), \quad \forall i = 1, ..., N$$

$$D_k(x_i) = D_k - 1(x_i) - D(c_{k-1}) \exp\left(-\frac{\|x_i - x_j\|^2}{2(r_b/2)^2}\right)$$

where the first equation is used to compute the density at the first iteration $k = 1$, i.e., $D_1(x_i) = D(x_i)$ and, thus, the first center as $c_1 = \max\{D_1(x_i)\}$; on the other hand, the second equation is used calculate the new density for any iteration $k \geq 2$ where the effect of the last cluster center is removed from the density, and therefore, the new center becomes $c_k = \max\{D_k(x_i)\}$. Note that the meaning of r_a and r_b is the same as σ and β for the mountain method, and also notice that $r_a/2 = \sigma$ and $r_b/2 = \beta$. This relationship also holds $r_b = 2r_a$ or $r_b = 1.5r_a$. As a final note, observe that the density depends on the selected norm, which would have different results depending on the norm choice or definition.

B. Distance-based methods

1) k-means clustering:

The k-means algorithm is an iterative clustering method that partitions the data into k clusters by minimizing the sum of squared distances (defined in terms of an arbitrary norm) between each point x_i and its assigned centroid c_j. The algorithm starts by randomly selecting k points from the dataset as the initial centroids. Then, it assigns each data point x_i to the nearest centroid c_j based on the distance defined in terms of a specific norm. After each assignment, the algorithm recalculates the centroids by taking the mean of all data points assigned to each cluster. This process continues until the centroids no longer change significantly or until a maximum number of iterations is reached. The algorithm can be represented mathematically as follows:

Given a data set X with n data points and k initial centroids $c_1, ..., c_k$, the algorithm iteratively performs the following steps:

1) Assign each data point x_i to its closest centroid c_j, where $j = \arg\min_j(\|x_i - c_j\|^2)$.
2) Recalculate the centroids by taking the mean of all data points assigned to each cluster: $c_j = 1/|S_j| * \sum_{x_i \in S_j} x_i$, where S_j is the set of data points assigned to the centroid c_j.
3) Repeat steps 1 and 2 until convergence, i.e., the centroids no longer change significantly or a maximum number of iterations is reached.

One important characteristic of the k-means algorithm is its hard clustering nature, which means that each data point is assigned to one and only one cluster, making it a strict partitioning method. In other words, the algorithm does not allow for overlap between clusters or ambiguity in the assignment of points to clusters. This property makes k-means suitable for datasets with well-defined, non-overlapping clusters. However, it may not perform well on datasets with complex, overlapping structures or when the number of clusters is not clearly defined.

2) Fuzzy c-means clustering:

The fuzzy c-means (FCM) algorithm is a variation of the k-means algorithm that allows for soft clustering, meaning that each data point can belong to multiple clusters to varying degrees. The algorithm starts with randomly selecting a set of cluster centers, and then iteratively updates the membership weights for each data point based on their distance from each cluster center. The membership weight for each data point i and cluster j is given by the equation:

$$w_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d_{ij}}{d_{ik}}\right)^{\frac{2}{m-1}}}$$

where c is the number of clusters, $d_{ij} = \|x_i - x_j\|$ is the distance between data point i and cluster center j, and m is a fuzziness parameter that determines the degree of fuzziness in the clustering. After updating the membership weights, the cluster centers are recalculated as the weighted mean of the data points, with each point weighted by its membership weight. This process of updating the membership weights and cluster centers is repeated until convergence. The final output of the FCM algorithm is a membership matrix W where each entry w_{ij} represents the degree to which data point i belongs to cluster j.

3) k-medoids clustering:

The k-medoids algorithm is a variation of the k-means algorithm, which instead of assigning each point to the closest cluster center, it selects one of the points as the center of the cluster, and updates the membership of the points accordingly. Let $x_1, x_2, ..., x_N$ be the data points, and let $c_1, c_2, ..., c_k$ be the medoids, where k is the number of clusters. Let r_{ij} denote the membership of point i in cluster j, with $r_{ij} = 1$ if point i belongs to cluster j, and $r_{ij} = 0$ otherwise. The objective function of the k-medoids algorithm is given by:

$$J = \sum_{i=1}^{N} \sum_{j=1}^{k} r_{ij} d(x_i, c_j)$$

where $d(x_i, c_j) = \|x_i - c_j\|$ is a distance measure between point x_i and medoid c_j. The goal is to minimize this objective function by finding the optimal medoids and membership matrix. The algorithm starts by randomly selecting k medoids from the data points. Then, for each data point x_i, the distance to each medoid c_j is calculated, and the medoid with the smallest distance is selected as the new cluster center for the corresponding cluster. The membership matrix is then updated based on the new medoids, and the process is repeated until convergence.

C. Validation indexes

Clustering validation indexes play a crucial role in unsupervised learning as they provide a quantitative measure of how well a clustering solution fits the data. In general, validation indexes are used to assess the quality of a clustering algorithm by measuring how similar the clustering results are to some known ground truth or by evaluating the clustering results based on some internal criteria. External validation indexes are used when there is some known ground truth, such as in the case of simulated data or when clustering results are compared to expert-labeled data. Internal validation indexes, on the other hand, are used to evaluate the quality of a clustering solution based on some internal criteria such as the compactness and separation of clusters. The use of clustering validation indexes can help researchers and practitioners compare different clustering algorithms and configurations, and ultimately select the best approach for a particular data set and task. The following indexes are selected from the work of [17].

1) Internal validation indexes:

Internal validation indexes are used to evaluate the quality of a clustering solution without relying on external information, such as known class labels. There are various internal validation indexes available, and we used three of them in our experiments: Davies-Bouldin (DB), Dunn (DI), and Calinski-Harabasz (CH). Each of these indexes is used to evaluate the quality of a clustering solution by measuring how compact and well-separated the resulting clusters are.

a) Davies-Bouldin index: It measures the average similarity between each cluster and its most similar cluster, taking into account the size or scatter of each cluster. The lower the Davies-Bouldin index, the better the clustering solution. It is defined as

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{j \neq i} \left\{ \frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right\}$$

where k is the number of clusters, σ_i is the average distance between all points within cluster i, c_i is the centroid of cluster i, $d(c_i, c_j)$ is the distance between centroids c_i and c_j, and

$\max_{j \neq i}\{\frac{\sigma_i+\sigma_j}{d(c_i,c_j)}\}$ is the maximum value of the ratio between the average distances and the centroid distances for all pairs of clusters except for i and j.

b) Dunn index: It measures the ratio between the smallest distance between clusters and the largest diameter of any cluster. The higher the Dunn index, the better the clustering solution. It is defined as

$$DI = \frac{\min_{i \neq j d(c_i,c_j)}}{\max_i\{\text{diam}(C_i)\}}$$

where $d(c_i, c_j)$ is the distance between centroids c_i and c_j, $\min_{i \neq j} d(c_i, c_j)$ is the smallest centroid distance among all pairs of clusters, and $diam(C_i)$ is the diameter of cluster i, defined as the maximum distance between any two points in the cluster.

c) Calinski-Harabasz: It measures the ratio between the between-cluster variance and the within-cluster variance. The higher the Calinski-Harabasz index, the better the clustering solution. It is defined as

$$CH = \frac{(n-k)B}{(k-1)W}$$

where n is the total number of points, k is the number of clusters, B is the between-cluster sum of squares, defined as the sum of squared distances between each cluster's centroid and the overall centroid, and W is the within-cluster sum of squares, defined as the sum of squared distances between each point and its cluster centroid.

2) *External validation indexes:*

External validation indexes are used to compare the results of a clustering algorithm with a known set of true labels or ground truth. In this experiment, I used three external validation indexes: the Rand index (RI), the Jaccard index (JI), and the Fowlkes-Mallows index (FM). Each of these indexes is used to evaluate the similarity between a clustering solution and a known ground truth partition, by measuring the agreement between the two partitions.

a) Rand index: It measures the percentage of pairs of data points that are either in the same cluster in both the predicted and true labels or in different clusters in both. Its value range from 0 to 1, with higher values indicating better clustering results. It is defined as

$$RI = \frac{TP+TN}{TP+TN+FP+FN}$$

where TP is the number of true positive pairs, TN is the number of true negative pairs, FP is the number of false positive pairs, and FN is the number of false negative pairs.

b) Jaccard index: It measures the similarity between the predicted and true labels. Its value range from 0 to 1, with higher values indicating better clustering results. It is defined as

$$JI = \frac{TP}{TP+FP+FN}$$

where TP, FP, and FN are the same as in the Rand index formula.

c) Fowlkes-Mallows index: It measures the geometric mean between precision and recall of the predicted clusters compared to the true clusters. Its value range from 0 to 1, with higher values indicating better clustering results. It is defined as

$$\sqrt{\frac{TP}{TP+FP} \cdot \frac{TP}{TP+FN}}$$

where TP, FP, and FN are the same as in the Rand index formula.

D. *Iris data set*

The Iris flower data set [18] is a classic data set in the field of machine learning and is often used as a benchmark for testing various clustering algorithms. The data set was introduced by British statistician and biologist Ronald Fisher in 1936[1]. The data set consists of 50 samples from each of three species of Iris flowers (Iris setosa, Iris virginica, and Iris versicolor), making a total of 150 samples. For each sample, four features were measured: the length and width of the sepals and petals, in centimeters. The data set is widely used for pattern recognition, clustering, and visualization techniques, as well as for educational purposes. The Iris data set is also considered a milestone in the field of machine learning, as it demonstrated the usefulness of multivariate analysis and statistical techniques in classifying plants into different species based on their characteristics.

E. *Experiment I: clustering iris*

In the first experiment, we implemented all the clustering methods with the iris data set, which is a 4D data set with 3 clusters: 1 isolated cluster (setosa flowers) and 2 overlapping clusters (versicolor and virginica flowers). We implemented the density-based methods with parameters such that the resulting clusters would match the known number of clusters k=3, and the distance-based methods with the hyperparameter k=3. We also implemented 6 clustering validation indexes: 3 internal validation indexes (Davies-Bouldin, Dunn, Calinski-Harabsz) and 3 external validation indexes (Rand, Jaccard, Fowlkes-Mallows) as the true labels are known in the iris data set. We then computed all the indexes for every one of the 5 clustering methods. This experiment aimed to evaluate the performance of each method in the iris data set and compare their validation indexes to determine which method resulted in the best clustering. Additionally, we embedded the data into a 2D space using UMAP, but solely for visualization purposes.

F. *Real data: Mall Customer Segmentation*

The Mall Customer Segmentation data set[2] contains information about the customers of a mall. The dataset includes 200 customers and 5 features: Customer ID, Gender, Age, Annual Income (in thousands of dollars), and Spending Score (which

[1]R. A. Fisher, "*The Iris Data Set*," UCI Machine Learning Repository, Irvine, CA, [Online]. Available: https://archive.ics.uci.edu/ml/datasets/iris.

[2]Choudhary, V. (2018). Customer Segmentation Tutorial in Python. Kaggle. Retrieved from https://www.kaggle.com/vjchoudhary7/customer-segmentation-tutorial-in-python.

is a score given by the mall based on the customer's behavior and spending nature). The main objective of this dataset is to segment the customers into different groups based on their spending habits and annual income. The dataset is commonly used for clustering and visualization purposes. The customer ID and gender features are not relevant to the clustering task, so they can be ignored. The age, annual income, and spending score features can be used to create a 3-dimensional space in which the clustering methods can be applied. The dataset provides an opportunity to evaluate clustering methods in a real-world scenario where there are no true labels available.

G. UMAP

UMAP (Uniform Manifold Approximation and Projection) is a non-linear dimensionality reduction algorithm that can be used for visualization and clustering tasks [13]. UMAP is a powerful alternative to t-SNE and PCA, two popular linear dimensionality reduction techniques. The main idea behind UMAP is to construct a low-dimensional representation of the data points that preserves both the global and local structure of the data. UMAP uses a graph-based approach to identify the manifold structure of the data and uses this structure to preserve the topological relationships between the data points in the reduced-dimensional space.

To create the graph structure, UMAP uses a k-nearest neighbor algorithm to identify the local neighborhood around each data point. A weighted graph is then constructed, with each data point as a node in the graph and edges connecting nodes in their respective neighborhoods. UMAP then optimizes the layout of this graph in the lower-dimensional space to preserve the neighborhood structure of the data.

One of the key advantages of UMAP over other dimensionality reduction techniques is its ability to handle non-linear and complex data structures. Additionally, UMAP is computationally efficient and can scale to large datasets. UMAP has been successfully applied to a variety of domains, including bioinformatics, image analysis, and natural language processing.

H. Autoencoder

Autoencoders are neural networks used for unsupervised learning and have become popular in recent years for their ability to perform feature extraction and dimensionality reduction. They consist of an encoder and a decoder, where the encoder takes the input data and maps it to a lower-dimensional space, while the decoder maps it back to the original space. The encoder and decoder are trained together, with the aim of minimizing the reconstruction error between the original data and the reconstructed data.

Autoencoders can also be used for transforming the original space into a high-dimensional space, by adding additional layers in the middle of the network. This can be useful for tasks such as anomaly detection, where a high-dimensional representation of the data can make it easier to detect outliers. Autoencoders can also be used for generative tasks, such as image generation, where the decoder can be used to generate new images based on the learned representation. The performance of an autoencoder depends on the architecture of the network, the number of hidden layers, and the choice of activation functions.

I. Experiment II: clustering real data

For the second experiment, we aimed to apply the same 5 clustering methods to real data. The data was preprocessed and then density-based methods were applied to find the optimal number of clusters k. To find the optimal k, we tested the density-based methods with different parameter values in a grid-like search for each of 4 different distances: Manhattan, Euclidean, Mahalanobis, and Cosine. The selection method for the optimal number of clusters k was based on a heuristic selection where we calculated the values of the internal indexes and kept a score of the most repeated instance that resulted in the best of all the possible combinations of sorting or ordering based on the ideal index values. After finding the optimal number of clusters k, we ran the 3 distance-based methods with that value as the hyperparameter k and visualized the results. This experiment was done 3 times: one for the original data, another with a lower-dimensional data space with the use of UMAP, and a last one with a higher-dimensional data space with the use of an autoencoder.

As real data was used, there were no true labels to validate with. So the only way to validate was by using the internal validation indexes which were used to find the optimal number of clusters k with density-based methods. The validation indexes were only used on density-based methods. There was no selection of one distance-based method as the idea was to obtain results for every one of them. Therefore, there was no need to validate them. The main idea of this real data scenario was to use density-based methods as an exploratory way to determine the optimal number of clusters and then use distance-based methods to compute the centers of the data, their membership, and therefore, the resulting different clusters for each of the 3 distance-based methods, just like in a real scenario.

J. Code implementation and GitHub repository

All the code was structured and managed in a public GitHub repository created for the course (link to the repository). We implemented every algorithm in Google Colab which currently runs in `Python 3.9` with the use of either GPU or TPU acceleration. The computational resources of Google Colab are listed in table I below.

The available resources for each Colab session depend on whether the user's plan is free or paid. The paid version can be customizable, which means that every user can have particular sessions. A direct link for the notebook implementation is provided in the table II below.

III. Results and discussion

A. UMAP visualization with iris

We first normalized the data to the n-dimensional hypercube [0,1]. Next, we visualized the a priori clustering of the data

Resource	**CPU**	**GPU**	**TPU**
Processing power	12 GB RAM, 2 vCPUs	12 GB RAM, 1 GPU	16 GB HBM, 8 cores
Model	Intel Xeon or AMD EPYC processor	NVIDIA Tesla K80	Google TPU v2
Cores	Varies by model and configuration	2496 CUDA cores	8 cores
Frequency	Varies by model and configuration	Varies by model and configuration	700 MHz
Memory	Varies by model and configuration	12 GB GDDR5 VRAM	16 GB HBM
Memory Bandwidth	Varies by model and configuration	Varies by model and configuration	900 GB/s
Version	N/A	N/A	v2

TABLE I: Computational resources of Google Colab. Link to source.

Method	**Direct link**
Unsupervised learning workshop	link to source

TABLE II: Direct link to the Google Colab notebook for the unsupervised learning workshop in the GitHub repository.

in every possible 2D plane using a pairplot, which can be observed in the figure 1 below.

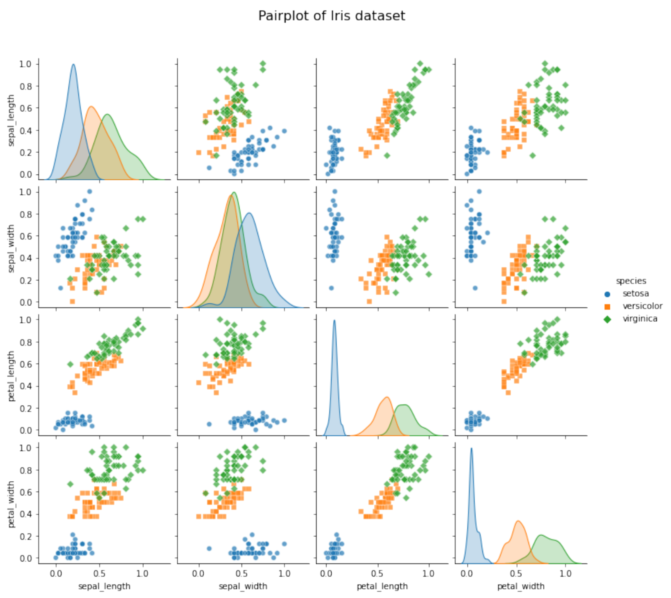

Fig. 1: Pairplot of the iris data set.

The resulting 4x4 matrix-like plot shows the smoothed histogram or density of each data cluster on the diagonal, revealing the a priori grouping of the data into 3 clusters: one isolated cluster (setosa) and two overlapping clusters (versicolor and virginica). The resulting pairplot after applying the UMAP algorithm with 2 components (the desired dimensions for the resulting space), using Euclidean distance, a minimum distance of 0.3, and five neighbors, can be seen in the figure 2 below.

The resulting 2D space preserved the clustering behavior of the original data with the same three clusters: one isolated cluster (setosa) and two overlapping clusters (versicolor and virginica). Nonetheless, the embedding implies the loss of interpretability of the axis or features.

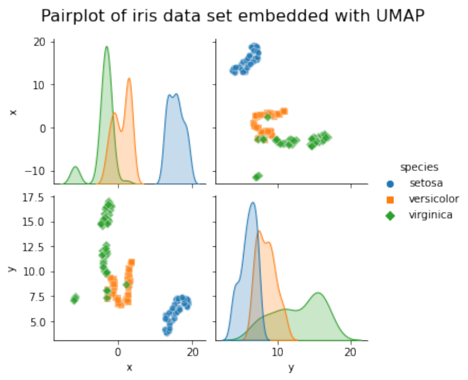

Fig. 2: Pairplot of the resulting 2D embedding of the iris data set.

B. *Experiment I: clustering iris*

1) *Mountain clustering:*

As the implementation of this method requires the construction of a grid space, it is a good idea to visualize an example to clarify the ideas exposed in the methodology. A 3D plot with the grid space and iris data set with its respective clusters can be seen in figure 3 below.

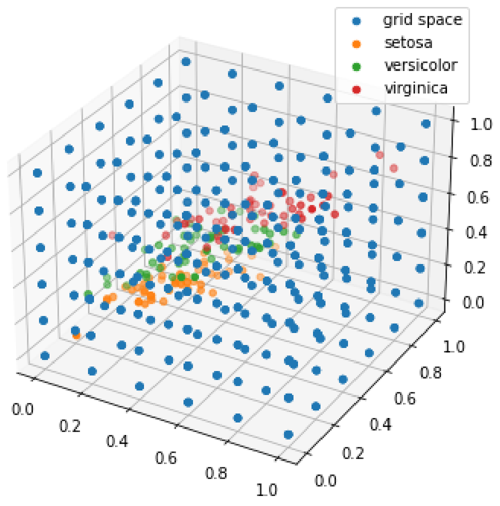

Fig. 3: Visualization of the iris data set and an example 3D grid space of granularity of 6 points per dimension.

After implementing the mountain clustering with a grid space of granularity of 2, $\sigma = 0.2$, $\beta = 2\sigma$, and Euclidean distance, the resulting centers can be seen in figure 4 below.

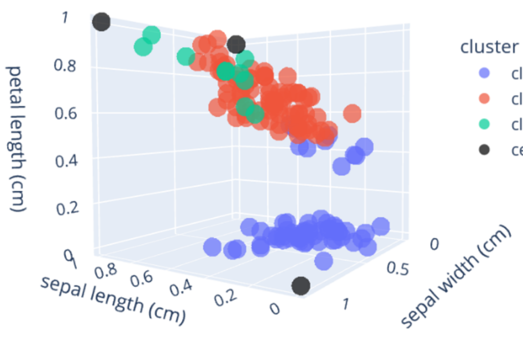

Fig. 4: Resulting clusters for the mountain algorithm.

Note that because this is a density-based clustering method, the results suggest the optimal number of clusters $k = 3$ as the number of obtained centers. Therefore, even if the focus is more on the number of centers than a membership per se, we wanted to show what the clusters of density-based methods look like when assigning a membership to each data point. In the case of the mountain algorithm, the algorithm effectively found the optimal number of clusters as 3, but the resulting centers were practically at the corners of the grid space. This suggests that if the mountain algorithm were to be used to determine the membership, i.e., the actual clusters of data, this instance or configuration would not be the best suited for this purpose.

2) Subtractive clustering:
After implementing the mountain clustering with $r_a = 0.2$, $r_b = 2r_a$, Euclidean distance, and tolerance of $1e^{-}3$, the resulting centers can be seen in figure 5 below.

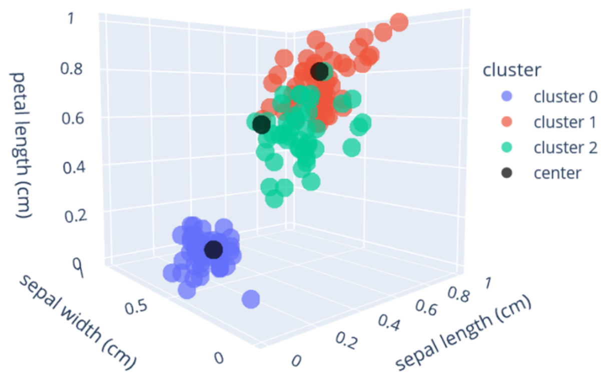

Fig. 5: Resulting clusters for the subtractive algorithm.

Note that because this is a density-based clustering method, the results suggest the optimal number of clusters $k = 3$ as the number of obtained centers. Therefore, even if the focus is more on the number of centers than a membership per se, we wanted to show what the clusters of density-based methods look like when assigning a membership to each data point. In the case of the subtractive algorithm, the algorithm effectively found the optimal number of clusters as 3, and in comparison with the mountain algorithm, this method resulted in more promising results as the centers were actually inside the clusters, thus, providing more reliable memberships if the subtractive algorithm were to be used as a method to find the actual membership, and therefore, the actual data clusters.

3) k-means clustering:
After implementing the k-means clustering with $k = 3$, a maximum number of iterations of 100, Euclidean distance, and tolerance of $1e^{-}3$, the resulting centers can be seen in figure 6 below.

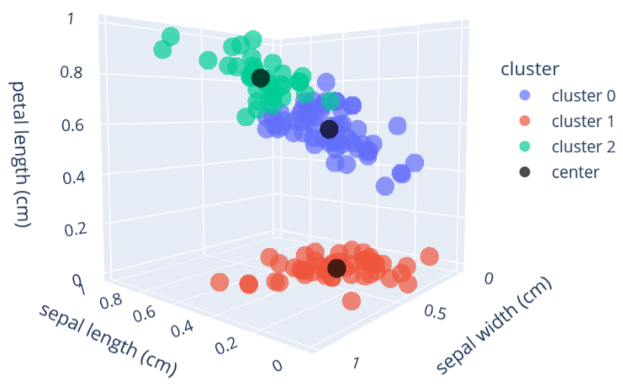

Fig. 6: Resulting clusters for k-means algorithm.

The plot showed that the algorithm successfully clustered the setosa flowers in an isolated cluster and the versicolor and virginica flowers in two separate overlapping clusters. However, it is worth noting that k-means is a distance-based clustering algorithm, which assumes that the clusters are spherical and equally sized. In the case of the iris dataset, the versicolor and virginica clusters are not clearly separated, and their shapes are not spherical, which can lead to misclassifications. Additionally, k-means is sensitive to the initial position of the centroids, which can lead to different results with different initializations. Overall, while k-means was able to cluster the iris dataset fairly well, it has its limitations and may not be the best choice for all datasets or scenarios.

4) Fuzzy c-means clustering:
After implementing the fuzzy c-means clustering with $k = 3$, $m = 2$, a maximum number of iterations of 100, Euclidean distance, and tolerance of $1e^{-}3$, the resulting centers can be seen in figure 7 below.

The fuzzy c-means clustering algorithm, similar to k-means, was able to identify the three clusters present in the

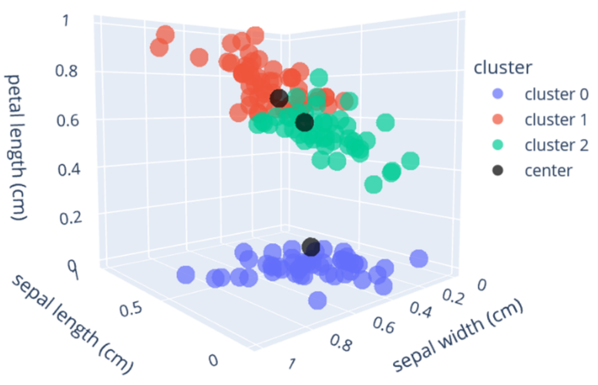

Fig. 7: Resulting cluster for the fuzzy c-means algorithm.

iris dataset. The resulting membership values obtained from fuzzy c-means, however, showed a much smoother transition between clusters than the hard cluster assignment obtained from k-means. This is due to the probabilistic nature of fuzzy c-means, which assigns membership values to each data point indicating the degree to which it belongs to each cluster. This implies that the overlapping cluster behavior present in the iris data set was better captured with this algorithm. Overall, fuzzy c-means was able to effectively identify the underlying clusters in the iris dataset with a probabilistic approach, providing additional insight into the data compared to the hard cluster assignment obtained from k-means.

5) k-medoids clustering:
After implementing the k-medoids clustering with $k = 3$, a maximum number of iterations of 100, Euclidean distance, and tolerance of $1e^{-}3$, the resulting centers can be seen in figure 8 below.

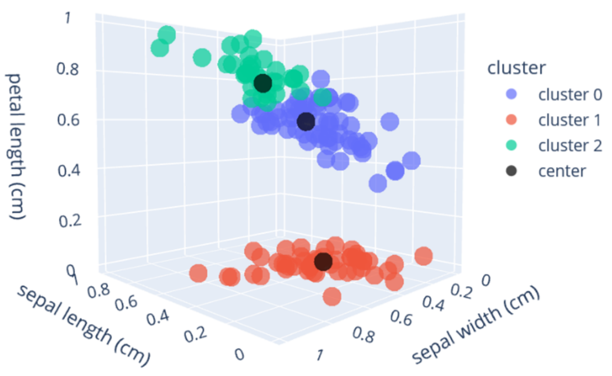

Fig. 8: Resulting cluster centers for the k-medoids algorithm.

Similar to k-means, k-medoids also found the correct number of clusters for the iris dataset, with the setosa flowers forming a separate cluster, and the versicolor and virginica flowers forming the other two clusters. The clustering of the versicolor and virginica flowers is slightly different from k-means and fuzzy c-means due to the fundamental differences of the algorithms and the factor of randomness, even if a random seed was used for reproducibility. Just as k-means, the hard clustering nature is observed in contrast with the soft clusters obtained with fuzzy c-means. One thing to note is that because the algorithm is based on medoids with actual data points in comparison with the nature of the centers that k-means uses, the resulting clusters should be more robust as the algorithm is less sensitive to outliers.

6) Validation indexes:
In clustering analysis, validation indexes are used to evaluate the quality of the resulting clustering. These indexes measure how well the clustering algorithm has grouped the data points and provide insight into which clustering algorithm and parameters may work best for a given data set. There are two types of validation indexes: internal validation indexes and external validation indexes. Internal validation indexes measure the quality of the clustering based on the data itself, without any prior knowledge of the true cluster labels. External validation indexes, on the other hand, require prior knowledge of the true cluster labels and measure how well the resulting clustering matches the true labels. In this experiment, both internal and external validation indexes were used to evaluate the performance of the five clustering algorithms applied to the Iris data set. For internal validation, three indexes were computed: Davies-Bouldin, Dunn, and Calinski-Harabsz. For external validation, three indexes were computed as well: Rand, Jaccard, and Fowlkes-Mallows. As the true cluster labels are known in this experiment, the external validation indexes could be computed, which allows for a more controlled scenario to compare the performance of the clustering algorithms. The resulting validation indexes for every one of the 5 clustering algorithms for the iris data set are shown in table III below.

Looking at the results of the validation indexes for the five clustering methods on the iris data set, we can observe some particularities. The best-performing method in terms of internal validation indexes (Davies-Bouldin, Dunn, Calinski-Harabasz) is the subtractive clustering method, followed by the mountain method. However, when looking at the external validation indexes (Rand, Jaccard, Fowlkes-Mallows), the fuzzy c-means method is the one that performs the best, followed by the k-medoids method. This means that density-based methods resulted in better cluster quality based on the internal validation indexes where subtractive was the best method overall. On the other hand, distance-based methods resulted in better cluster accuracy based on the external validation indexes whereas fuzzy c-means resulted in the method with the most accuracy overall.

C. Experiment II: clustering real data

1) Original data space:
In this experiment, we applied clustering methods to the Mall Customer Segmentation dataset, which includes information about the customers of a mall. The dataset includes 200

Clustering method	Validation indexes					
	Internal			External		
	Davies-Bouldin (↓)	Dunn (↑)	Calinski-Harabasz (↑)	Rand (↑)	Jaccard (↑)	Fowlkes-Mallows (↑)
Mountain	10.011	1.764	11.111	0.740	0.261	0.675
Subtractive	10.365	1.112	5.484	0.853	0.261	0.780
k-means	15.862	1.680	15.363	0.875	0.276	0.815
Fuzzy c-means	20.164	1.700	1.142	0.913	0.290	0.870
k-medoids	18.531	1.650	16.623	0.887	0.284	0.834

TABLE III: Validations indexes for each of the five clustering methods on the iris data set. Indexes with (↓) mean lower values are better, while indexes with (↑) mean higher values are better.

customers and 5 features, including customer ID and gender, which are categorical features. To create a numerical data space for clustering, we ignored these two features and used the age, annual income, and spending score features. The resulting 3D numerical data space was then transformed into an n-dimensional hypercube [0,1] by applying min-max normalization. The visualization of the original data space is shown in figure 9 below, where each point represents a customer with particular feature values or characteristics.

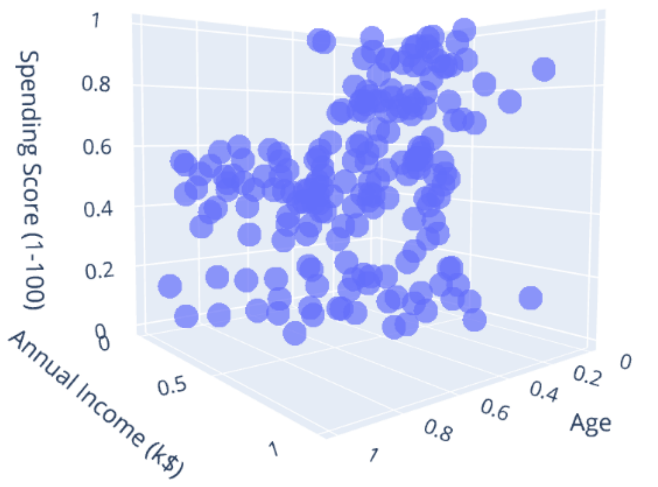

Fig. 9: Visualization of the original 3D data space of the Mall Customer Segmentation data set.

a) Heuristic for optimal number of clusters:
To choose the optimal number of clusters k we explore by experimenting with the density-based clustering methods: mountain and subtractive. The idea is to consider different hyperparameter configurations for each of our distances: Manhattan, Euclidean, Mahalanobis, and Cosine. This approach is similar to a hyperparameter grid search in a heuristic context. We perform the search by having a default set of values for the parameters and running the algorithm with the variation of one parameter at a time.

For each distance, we select a candidate number of clusters by a scoring system based on all possible combinations of sorting with the internal validation indexes, so the candidate wins a point if is selected as the configuration that yielded the lowest Davies-Bouldin index, highest Dunn index, and highest Calinski-Harabasz index, and so on with every possible combination. The candidate with the most points and which is the most parsimonious (lowest number of clusters) is the winner. Afterward, we join the results for each distance and select the best candidate for each method, and finally, the best of both methods by following the same selection criteria.

b) Mountain clustering:
A hyperparameter grid was set for the mountain clustering algorithm to perform a grid-like search. The granularity hyperparameter varied from 3 to 5 with a step size of 1, while the σ parameter varied from 0.2 to 0.4 with a step size of 0.1. While every parameter was varied, the value of the remaining parameters where set to use the default values. The default values for the hyperparameters of the grid are a granularity of 2 and $\sigma = 0.2$. Because of this, stating the configuration of a candidate would reduce to show only the particular hyperparameter value when showing the configuration in the results. The grid-like search was performed with 4 types of distances: Manhattan, Euclidean, Mahalanobis, and Cosine. The best candidates for each distance and the best overall (selected with the heuristic method previously explained) are shown in table IV below.

After comparing the best configuration candidates for every distance type to determine the number of clusters k, we find interesting results. First, both the Mahalanobis and Cosine candidates were disqualified from the competition as the mountain method did not work for the grid-like search of one of the iterations of the search. This left us with only 2 candidates where the best overall was a configuration with Euclidean distance, $\sigma = 0.4$, and default values for every other hyperparameter.

c) Subtractive clustering:
A hyperparameter grid was set for the mountain clustering algorithm to perform a grid-like search. The r_a hyperparameter varied from 0.2 to 0.4 with a step size of 0.1. While every parameter was varied, the value of the remaining parameters where set to use the default values. The default values for the hyperparameters of the grid $r_a = 0.2$. Because of this, stating

Best candidates for Mountain

Distance	Configuration	Davies-Bouldin index (↓)	Dunn index (↑)	Calinski-Harabasz index (↑)	Num. of clusters k
Manhattan	granularity $= 5$	12.226	0.142	3.893	14
Euclidean	$\sigma = 0.4$	13.815	0	4.424	15
Mahalanobis	-	-	-	-	-
Cosine	-	-	-	-	-
Best overall: Euclidean					

TABLE IV: Results for the best configuration candidates for the Mountain method for all distance types: Manhattan, Euclidean, Mahalanobis, and Cosine. The candidates with no values (-) mean that one of the iterations of the method did not work for the grid-like search for that specific distance type, disqualifying them for the selection round. (↑) means the higher the better, and (↓) the lower the better.

the configuration of a candidate would reduce to show only the particular hyperparameter value when showing the configuration in the results. The grid-like search was performed with 4 types of distances: Manhattan, Euclidean, Mahalanobis, and Cosine. The best candidates for each distance and the best overall (selected with the heuristic method previously explained) are shown in table V below.

After comparing the best configuration candidates for every distance type to determine the number of clusters k, we find interesting results. First, the Mahalanobis candidate was disqualified from the competition as the subtractive method did not work for the grid-like search of one of the iterations of the search. This left us with 3 candidates where the best overall was a configuration with Cosine distance, $r_a = 0.4$, and default values for every other hyperparameter.

d) Optimal number of clusters:
Having the best candidates from both density-based methods, we implemented the same heuristic selection method one last time to determine the best candidate overall and, therefore, the optimal number of clusters k as the number of clusters for said candidate. The results are shown in table VI below where the index values are omitted as they were already shown previously in each method's results.

Best candidate overall

Method	Distance	Configuration	Number of clusters k
Mountain	Euclidean	$\sigma = 0.4$	15
Subtractive	Cosine	$r_a = 0.4$	2
Best overall: Subtractive			

TABLE VI: Results for the best method with the best configuration to determine the optimal number of clusters k.

The best method with the best configuration overall resulted in the Subtractive method with Cosine distance, $r_a = 0.4$, and every other hyperparameter with default values. Therefore, the optimal number of clusters k is 2.

e) Data clusters:
After determining the optimal number of clusters with density-based clustering methods, we use distance-based clustering methods to visualize the results and compare the clusters. The distance-based methods inherit the 2 main hyperparameters of the best configuration found for density-based methods: distance type and the number of clusters. This means that every distance-base clustering method is implemented with $k = 2$ and Cosine distance.

f) k-means clustering:
The resulting clusters for the k-means clustering method with hyperparameters $k = 2$ and Cosine distance are shown in figure 10 below.

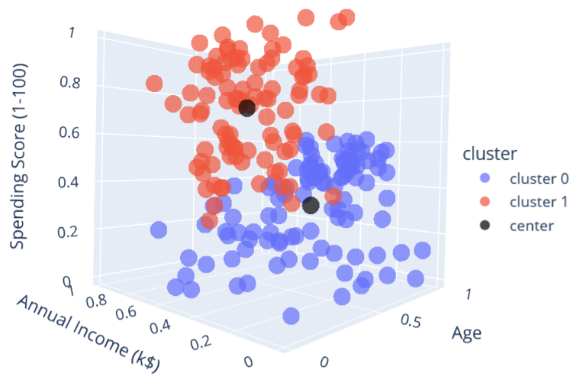

Fig. 10: Visualization of the clustering results for k-means on the original 3D data space of the Mall Customer Segmentation data set.

The results showed that k-means was able to cluster the data into visually distinct groups, suggesting that there were underlying patterns in the data. However, as there were no true labels to evaluate the performance of k-means, we relied on internal validation indexes to determine the optimal number of clusters. This raises the question of how reliable the clustering results for distance-based methods are without ground truth labels. It is important to note that distance-based clustering algorithms can still provide valuable insights into the structure of the data, even in the absence of true labels.

g) Fuzzy c-means clustering:
The resulting clusters for the fuzzy c-means clustering method with hyperparameters $k = 2$ and Cosine distance are shown in figure 11 below.

The results showed that fuzzy c-means was able to cluster the data into visually distinct groups, suggesting that there

Best candidates for Subtractive

Distance	Configuration	Davies-Bouldin index (↓)	Dunn index (↑)	Calinski-Harabasz index (↑)	Num. of clusters k
Manhattan	$r_a = 0.4$	6.615	0.602	1.406	6
Euclidean	$r_a = 0.4$	9.377	2.541	9.909	2
Mahalanobis	-	-	-	-	-
Cosine	$r_a = 0.4$	8.347	3.244	4.878	2
Best overall: Cosine					

TABLE V: Results for the best configuration candidates for the Subtractive method for all distance types: Manhattan, Euclidean, Mahalanobis, and Cosine. The candidates with no values (-) mean that one of the iterations of the method did not work for the grid-like search for that specific distance type, disqualifying them for the selection round. (↑) means the higher the better, and (↓) the lower the better.

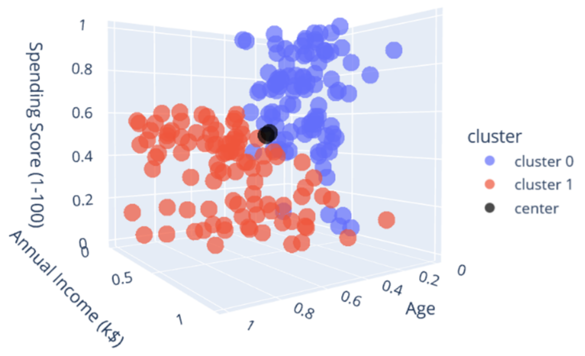

Fig. 11: Visualization of the clustering results for fuzzy c-means on the original 3D data space of the Mall Customer Segmentation data set.

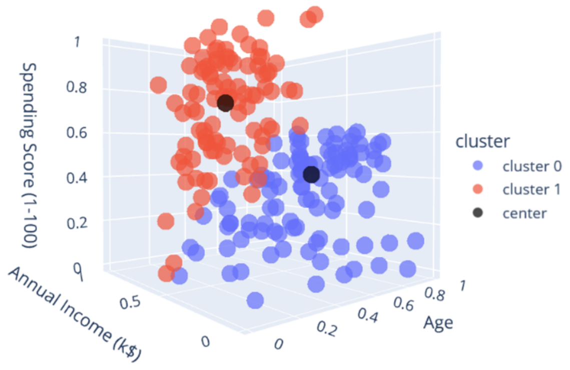

Fig. 12: Visualization of the clustering results for k-medoids on the original 3D data space of the Mall Customer Segmentation data set.

were underlying patterns in the data. However, the resulting centers were significantly close to one another. This suggests that, perhaps, the data does not present significant overlapping behavior given that other methods such as k-means show separated centers. Nonetheless, this behavior can appear on overlapping data as well, which means further validation would be necessary.

h) k-medoids clustering:
The resulting clusters for the k-medoids clustering method with hyperparameters $k = 2$ and Cosine distance are shown in figure 12 below.

The results showed that k-medoids was able to cluster the data into visually distinct groups, suggesting that there were underlying patterns in the data. The results were very similar to that of k-means, however, because of the robust nature of the method it should be more reliable than k-means in the presence of outliers.

2) Embedded data with UMAP:
In order to better understand the distribution of the data in a lower-dimensional space, we performed a Uniform Manifold Approximation and Projection (UMAP) transformation on the original data. UMAP is a non-linear dimensionality reduction technique that can uncover the underlying structure of high-dimensional data. We set the number of components to 2, which allowed us to visualize the data in a 2D embedded space. We used a Euclidean metric and set the minimum distance to 0.3 with a number of neighbors of 5. The resulting 2D embedded data space allowed us to visualize the distribution of the data and to identify potential patterns or clusters. The 2D plot is shown in figure 13 below.

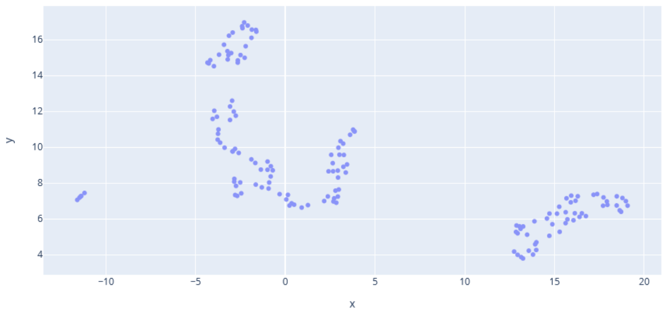

Fig. 13: Visualization of the embedded 2D data space of the Mall Customer Segmentation data set.

a) Mountain clustering: Both the same grid and methodology were performed for the embedded 2D data space. The results are shown in table VII below. After comparing the best configuration candidates for every distance type to determine the number of clusters k, we find

Best candidates for Mountain

Distance	**Configuration**	**Davies-Bouldin index (↓)**	**Dunn index (↑)**	**Calinski-Harabasz index (↑)**	**Num. of clusters k**
Manhattan	granularity $= 4$	12.376	0.775	10.631	5
Euclidean	granularity $= 3$	11.070	2.458	10.148	3
Mahalanobis	-	-	-	-	-
Cosine	-	-	-	-	-
Best overall: Manhattan					

TABLE VII: Results for the best configuration candidates for the Mountain method for all distance types: Manhattan, Euclidean, Mahalanobis, and Cosine. The candidates with no values (-) mean that one of the iterations of the method did not work for the grid-like search for that specific distance type, disqualifying them for the selection round. (↑) means the higher the better, and (↓) the lower the better.

interesting results. First, both the Mahalanobis and Cosine candidates were disqualified from the competition as the mountain method did not work for the grid-like search of one of the iterations of the search. This left us with only 2 candidates where the best overall was a configuration with Manhattan distance, granularity $= 4$, and default values for every other hyperparameter.

b) Subtractive clustering:
Both the same grid and methodology were performed for the embedded 2D data space. The results are shown in table VIII below.

After comparing the best configuration candidates for every distance type to determine the number of clusters k, we find interesting results. First, the Mahalanobis candidate was disqualified from the competition as the subtractive method did not work for the grid-like search of one of the iterations of the search. This left us with 3 candidates where the best overall was a configuration with Cosine distance, $r_a = 0.4$, and default values for every other hyperparameter.

c) Optimal number of clusters:
Having the best candidates from both density-based methods, we implemented the same heuristic selection method one last time to determine the best candidate overall and, therefore, the optimal number of clusters k as the number of clusters for said candidate. The results are shown in table IX below where the index values are omitted as they were already shown previously in each method's results.

Best candidate overall

Method	**Distance**	**Configuration**	**Number of clusters k**
Mountain	Manhattan	granularity $= 4$	5
Subtractive	Cosine	$r_a = 0.4$	2
Best overall: Subtractive			

TABLE IX: Results for the best method with the best configuration to determine the optimal number of clusters k.

The best method with the best configuration overall resulted in the Subtractive method with Cosine distance, $r_a = 0.4$, and every other hyperparameter with default values. Therefore, the optimal number of clusters k is 2. Notice this is the same result as the optimal number of clusters found for the original data space.

d) Data clusters:
After determining the optimal number of clusters with density-based clustering methods, we use distance-based clustering methods to visualize the results and compare the clusters. The distance-based methods inherit the 2 main hyperparameters of the best configuration found for density-based methods: distance type and the number of clusters. This means that every distance-base clustering method is implemented with $k = 2$ and Cosine distance.

e) k-means clustering:
The resulting clusters for the k-means clustering method with hyperparameters $k = 2$ and Cosine distance are shown in figure 14 below.

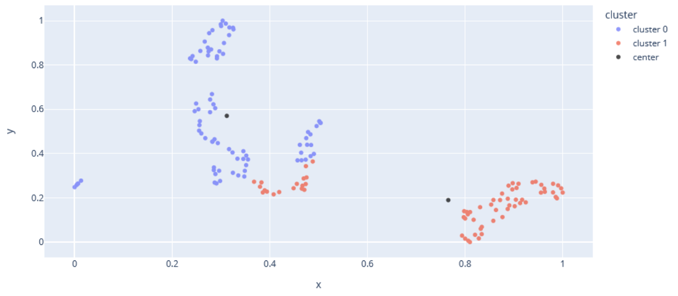

Fig. 14: Visualization of the clustering results for k-means on the embedded 2D data space of the Mall Customer Segmentation data set.

The results showed that k-means was able to cluster the data into visually distinct groups, suggesting that there were underlying patterns in the data. However, as there were no true labels to evaluate the performance of k-means, we relied on internal validation indexes to determine the optimal number of clusters. This raises the question of how reliable the clustering results for distance-based methods are without ground truth labels. One thing to note is the fact that even if visually there are 2 natural clusters that we could potentially assign intuitively, the orange cluster seems to also contain a portion of the other cluster even if the data is widely apart.

f) Fuzzy c-means clustering:
The resulting clusters for the fuzzy c-means clustering method

Best candidates for Subtractive

Distance	Configuration	Davies-Bouldin index (↓)	Dunn index (↑)	Calinski-Harabasz index (↑)	Num. of clusters k
Manhattan	$r_a = 0.4$	5.366	1.778	0.285	3
Euclidean	$r_a = 0.4$	4.978	1.594	0.448	3
Mahalanobis	-	-	-	-	-
Cosine	$r_a = 0.4$	7.728	2.751	4.600	2
Best overall: Cosine					

TABLE VIII: Results for the best configuration candidates for the Subtractive method for all distance types: Manhattan, Euclidean, Mahalanobis, and Cosine. The candidates with no values (-) mean that one of the iterations of the method did not work for the grid-like search for that specific distance type, disqualifying them for the selection round. (↑) means the higher the better, and (↓) the lower the better.

with hyperparameters $k = 2$ and Cosine distance are shown in figure 15 below.

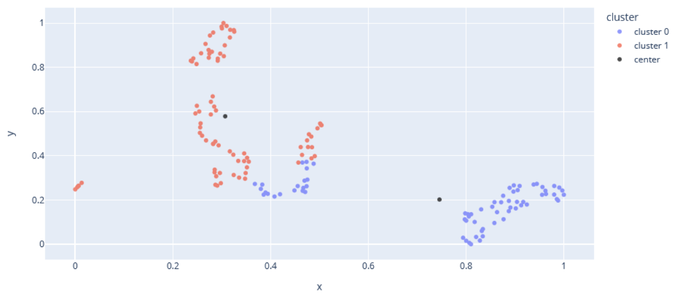

Fig. 15: Visualization of the clustering results for fuzzy c-means on the embedded 2D data space of the Mall Customer Segmentation data set.

The results showed that fuzzy c-means was able to cluster the data into visually distinct groups, suggesting that there were underlying patterns in the data. The results are noticeably similar to the results from k-means, both in the clusters and resulting centers. The only difference is the coloring, which is irrelevant as the task is to identify distinct groups. The fact that the results were practically the same and that this method's nature is soft-clustering indicates that the data is clearly separable and that the clusters are well-defined.

g) k-medoids clustering:
The resulting clusters for the k-medoids clustering method with hyperparameters $k = 2$ and Cosine distance are shown in figure 16 below.

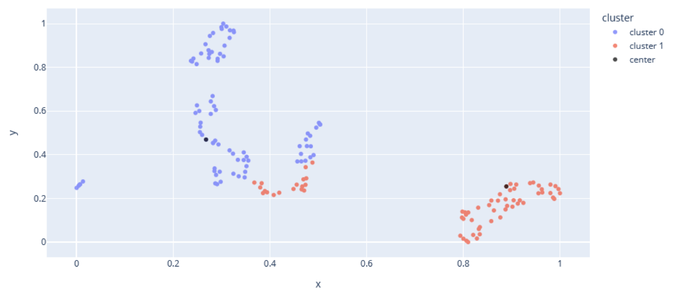

Fig. 16: Visualization of the clustering results for k-medoids on the embedded 2D data space of the Mall Customer Segmentation data set.

The results showed that k-medoids was able to cluster the data into visually distinct groups, suggesting that there were underlying patterns in the data. The results were very similar to that of previous methods. The main difference is the center locations, nonetheless, this suggests with a higher degree of confidence that the data is clearly separable and that the clusters are well-defined.

3) Autoencoded data:
In this experiment, we used an autoencoder to perform feature extraction on the Mall Customer Segmentation dataset. The autoencoder consisted of an input layer, a dense encoder layer of size 5 with a sigmoid activation function, and a dense decoder layer of size 3 with a sigmoid activation function. The network was trained with an Adam optimizer using binary cross-entropy as the loss function. After training, we obtained a 5D numerical data space which allowed us to explore the data in a more detailed way and gain insights into the underlying patterns and structures of the dataset. A pairplot of the resulting autoencoded space is shown in figure 17 below.

a) Mountain clustering: Both the same grid and methodology were performed for the autoencoded 5D data space. The results are shown in table X below. After comparing the best configuration candidates for every distance type to determine the number of clusters k, we find interesting results. First, both the Mahalanobis and Cosine candidates were disqualified from the competition as the mountain method did not work for the grid-like search of one of the iterations of the search. This left us with only 2 candidates where the best overall was a configuration with Euclidean distance, granularity $= 3$, and default values for every other hyperparameter.

b) Subtractive clustering:
Both the same grid and methodology were performed for the autoencoded 5D data space. The results are shown in table XI below.

After comparing the best configuration candidates for every distance type to determine the number of clusters k, we find interesting results. First, the Mahalanobis candidate was disqualified from the competition as the subtractive method

Best candidates for Mountain

Distance	**Configuration**	**Davies-Bouldin index (↓)**	**Dunn index (↑)**	**Calinski-Harabasz index (↑)**	**Num. of clusters k**
Manhattan	$\sigma = 0.4$	11.361	0	3.363	17
Euclidean	granularity $= 3$	8.678	0.564	3.763	7
Mahalanobis	-	-	-	-	-
Cosine	-	-	-	-	-
Best overall: Euclidean					

TABLE X: Results for the best configuration candidates for the Mountain method for all distance types: Manhattan, Euclidean, Mahalanobis, and Cosine. The candidates with no values (-) mean that one of the iterations of the method did not work for the grid-like search for that specific distance type, disqualifying them for the selection round. (↑) means the higher the better, and (↓) the lower the better.

Best candidates for Subtractive

Distance	**Configuration**	**Davies-Bouldin index (↓)**	**Dunn index (↑)**	**Calinski-Harabasz index (↑)**	**Num. of clusters k**
Manhattan	$r_a = 0.4$	4.860	0.140	0.650	7
Euclidean	$r_a = 0.4$	7.014	2.074	2.134	3
Mahalanobis	-	-	-	-	-
Cosine	$r_a = 0.4$	6.742	1.987	6.114	2
Best overall: Cosine					

TABLE XI: Results for the best configuration candidates for the Subtractive method for all distance types: Manhattan, Euclidean, Mahalanobis, and Cosine. The candidates with no values (-) mean that one of the iterations of the method did not work for the grid-like search for that specific distance type, disqualifying them for the selection round. (↑) means the higher the better, and (↓) the lower the better.

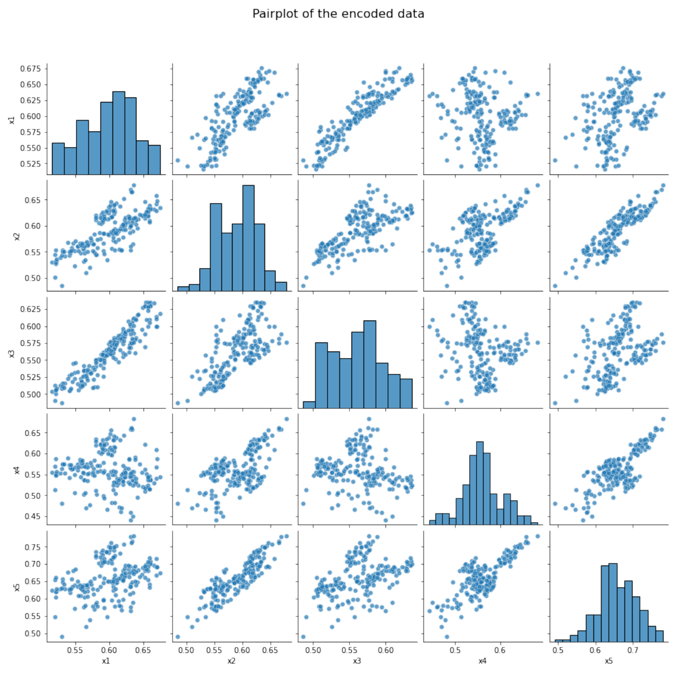

Fig. 17: Visualization of the autoencoded 5D data space of the Mall Customer Segmentation data set.

did not work for the grid-like search of one of the iterations of the search. This left us with 3 candidates where the best overall was a configuration with Cosine distance, $r_a = 0.4$, and default values for every other hyperparameter.

c) Optimal number of clusters:
Having the best candidates from both density-based methods, we implemented the same heuristic selection method one last time to determine the best candidate overall and, therefore, the optimal number of clusters k as the number of clusters for said candidate. The results are shown in table XII below where the index values are omitted as they were already shown previously in each method's results.

Best candidate overall

Method	**Distance**	**Configuration**	**Number of clusters k**
Mountain	Euclidean	granularity $= 3$	7
Subtractive	Cosine	$r_a = 0.4$	2
Best overall: Subtractive			

TABLE XII: Results for the best method with the best configuration to determine the optimal number of clusters k.

The best method with the best configuration overall resulted in the Subtractive method with Cosine distance, $r_a = 0.4$, and every other hyperparameter with default values. Therefore, the optimal number of clusters k is 2. Notice this is the same result as the optimal number of clusters found for the original data space.

d) Data clusters:
After determining the optimal number of clusters with density-based clustering methods, we use distance-based clustering methods to visualize the results and compare the clusters. The distance-based methods inherit the 2 main hyperparameters of the best configuration found for density-based methods: distance type and the number of clusters. This means that every distance-base clustering method is implemented with $k = 2$ and Cosine distance.

e) k-means clustering:
The resulting clusters for the k-means clustering method with hyperparameters $k = 2$ and Cosine distance are shown in figure 18 below.

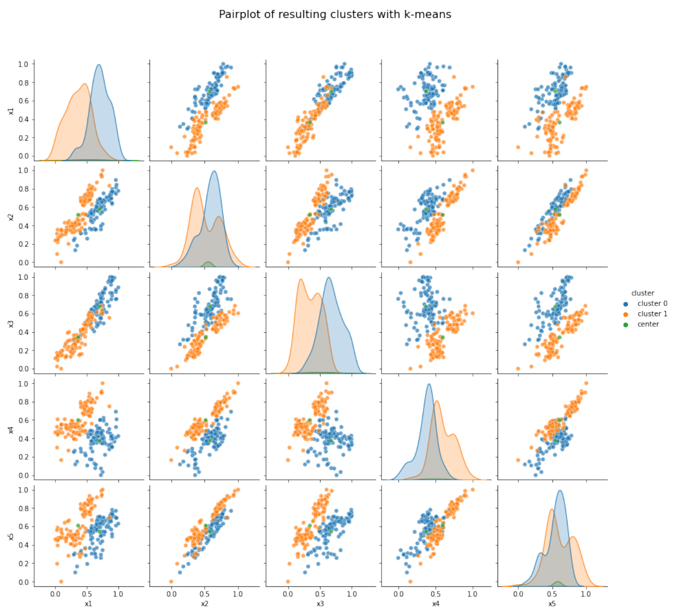

Fig. 18: Visualization of the clustering results for k-means on the autoencoded 5D data space of the Mall Customer Segmentation data set.

The results showed that k-means was able to cluster the data into visually distinct groups, suggesting that there were underlying patterns in the data. However, as there were no true labels to evaluate the performance of k-means, we relied on internal validation indexes to determine the optimal number of clusters. This raises the question of how reliable the clustering results for distance-based methods are without ground truth labels. One thing to note is the fact that every 2D feature space showed overlapping, which means that there is a degree of similarity between the two groups, which could potentially make it more difficult to accurately classify or cluster the data. This could be due to a variety of factors, such as the features being highly correlated due to the feature extraction or the existence of subgroups within the larger clusters.

f) Fuzzy c-means clustering:
The resulting clusters for the fuzzy c-means clustering method with hyperparameters $k = 2$ and Cosine distance are shown in figure 19 below.

The results showed that fuzzy c-means was able to cluster the data into visually distinct groups, suggesting that there were underlying patterns in the data. The results are noticeably similar to the results from k-means, both in the clusters and resulting centers. The fact that the results were practically the same and that this method's nature is soft-clustering indicates that the data is clearly separable and that the clusters are well-defined. Note that the same behavior

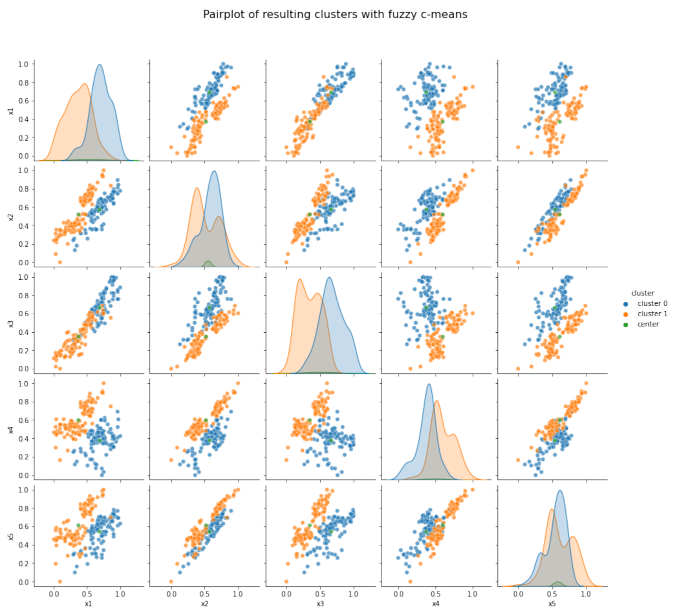

Fig. 19: Visualization of the clustering results for fuzzy c-means on the autoencoded 5D data space of the Mall Customer Segmentation data set.

occurred in the embedded 2D space.

g) k-medoids clustering:
The resulting clusters for the k-medoids clustering method with hyperparameters $k = 2$ and Cosine distance are shown in figure 20 below.

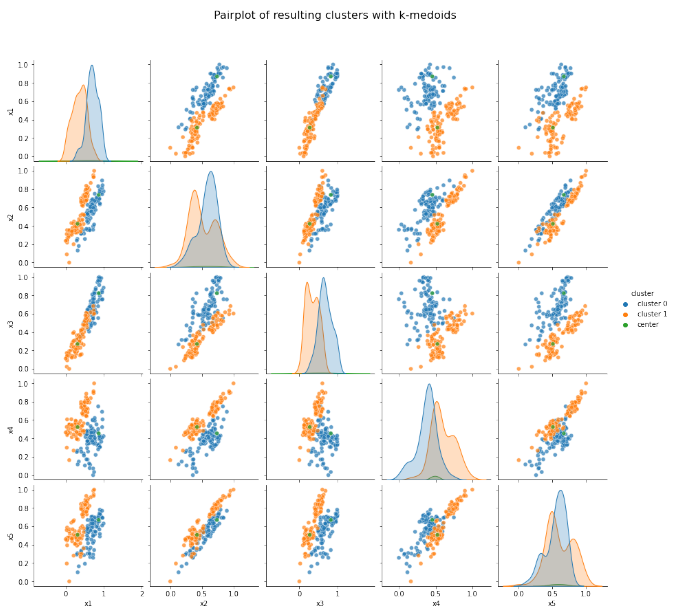

Fig. 20: Visualization of the clustering results for k-medoids on the autoencoded 5D data space of the Mall Customer Segmentation data set.

The results showed that k-medoids was able to cluster the data into visually distinct groups, suggesting that there were underlying patterns in the data. The results were very similar to that of previous methods. The main difference is the center locations, nonetheless, this suggests with a higher degree of confidence that the data is clearly separable and that the clusters are well-defined. Note that the same behavior occurred in the embedded 2D space.

IV. Conclusions

The results of the first experiment show that the choice of clustering method can have a significant impact on the quality of the resulting clusters. Having one method with the best internal validation indexes results and another with the best external validation indexes results indicates that there is a trade-off between the quality of the clustering and the accuracy of the clustering concerning the true labels. Density-based clustering methods were better in terms of cluster quality than distance-based clustering methods, nonetheless, distance-based methods were better in terms of clustering accuracy than density-based methods. In particular, the fuzzy c-means resulted in the method with the highest accuracy. This is not surprising as the presence of overlapping clusters on the iris data set was a good scenario for this method to perform as the soft nature of the resulting clusters allows for better accuracy when overlapping occurs.

When reducing the dimensionality of a high-dimensional space, some information is inevitably lost. The goal of dimensionality reduction techniques like UMAP is to preserve as much of the important information as possible while reducing the dimensionality of the data. In the case of the iris data set, the original 4D feature space contains information about the physical measurements of the flowers, including the length and width of the petals and sepals. However, when this high-dimensional space is reduced to a 2D space using UMAP, the resulting features are no longer directly interpretable as physical measurements of the flowers. Instead, they are a projection of the original features onto a lower-dimensional space that preserves the underlying structure of the data. Therefore, while the features themselves are not directly interpretable in terms of physical measurements, they can still be used to visualize and analyze the structure of the data in a lower-dimensional space.

After performing clustering analysis on three different data spaces, the Subtractive method with Cosine distance, $r_a = 0.4$, and the optimal number of clusters $k = 2$ using every other parameter with default values resulted in the best overall method to determine the optimal number of clusters. However, the behavior of the resulting clusters varied between the data spaces. In the original 3D data space, the distance-based clustering methods resulted in similar behavior with fuzzy c-means resulting in cluster centers close to one another. In the embedded 2D data space, the distance-based clustering methods resulted in two clusters with similar centers, with one covering a small region of the other, but no overlapping. In contrast, in the autoencoded 5D data space, overlapping clusters were observed in every 2D feature space. Interestingly, the Mahalanobis distance did not work in any of the data spaces, and further investigation may be needed to understand this behavior. The Mahalanobis distance requires the computation of the covariance matrix of the data. However, in some cases, the covariance matrix may be singular or ill-conditioned, which means that it cannot be inverted. This can happen when there are linearly dependent variables in the dataset or when the number of dimensions is greater than the number of observations. In such cases, the Mahalanobis distance cannot be computed. Additionally, the Mahalanobis distance assumes that the data is normally distributed, and if this assumption is not met, the distance metric may not be appropriate. Finally, the best density-based configuration to determine the optimal number of clusters was consistent across all three data spaces, highlighting the effectiveness of the Subtractive method with Cosine distance, $r_a = 0.4$, and the optimal number of clusters $k = 2$.

Exploring lower and higher dimensional data spaces for clustering has several advantages. In lower dimensional spaces, the clustering results can be easily visualized, and in some cases, a clear separation between the clusters can be observed. However, as the dimensionality of the data increases, visualizing the clusters becomes harder, and it is necessary to rely on other methods to determine the optimal number of clusters and to evaluate the performance of the clustering algorithms. On the other hand, working in higher-dimensional spaces can uncover patterns and relationships that would not be apparent in lower-dimensional spaces. Overall, exploring lower and higher dimensional data spaces for clustering can provide valuable insights into the structure and relationships within the data. The consistency of the clustering results across different data spaces in this experiment demonstrates the robustness and reliability of the clustering methods used.

References

[1] J. A. Hartigan, *Clustering algorithms*. John Wiley & Sons, Inc., 1975.

[2] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," *Proceedings of the National Academy of Sciences*, vol. 95, no. 25, pp. 14863–14868, 1998.

[3] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[4] T. Leonard, *A course in categorical data analysis*, vol. 45. CRC Press, 1999.

[5] P. Kotler, "Atmospherics as a marketing tool," *Journal of retailing*, vol. 49, no. 4, pp. 48–64, 1973.

[6] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," *ACM sigmod record*, vol. 25, no. 2, pp. 103–114, 1996.

[7] P. Dönmez, "Introduction to machine learning, 2nd ed., by ethem alpaydın. cambridge, ma: The mit press 2010. isbn: 978-0-262-01243-0," *Natural Language Engineering*, vol. 19, no. 2, p. 285–288, 2013.

[8] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[9] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *kdd*, vol. 96, num. 34, pp. 226–231, 1996.

[10] J. MacQueen, "Classification and analysis of multivariate observations," in *5th Berkeley Symp. Math. Statist. Probability*, pp. 281–297, University of California Los Angeles LA USA, 1967.

[11] W. Peizhuang, "Pattern recognition with fuzzy objective function algorithms (james c. bezdek)," *Siam Review*, vol. 25, no. 3, p. 442, 1983.

[12] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[13] L. McInnes, J. Healy, and J. Melville, “Umap: Uniform manifold approximation and projection for dimension reduction,” *arXiv preprint arXiv:1802.03426*, 2018.

[14] E. Becht, L. McInnes, J. Healy, C.-A. Dutertre, I. W. Kwok, L. G. Ng, F. Ginhoux, and E. W. Newell, “Dimensionality reduction for visualizing single-cell data using umap,” *Nature biotechnology*, vol. 37, no. 1, pp. 38–44, 2019.

[15] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, “Greedy layer-wise training of deep networks,” *Advances in neural information processing systems*, vol. 19, 2006.

[16] K. Hammouda and F. Karray, “A comparative study of data clustering techniques,” *University of Waterloo, Ontario, Canada*, vol. 1, 2000.

[17] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Pérez, and I. Perona, “An extensive comparative study of cluster validity indices,” *Pattern recognition*, vol. 46, no. 1, pp. 243–256, 2013.

[18] R. A. Fisher, “The use of multiple measurements in taxonomic problems,” *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.