Unsupervised Learning Methods

Luisa Toro Villegas
Universidad EAFIT
Medellin, Colombia
ltorov@eafit.edu.co

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

***Abstract*—This paper delves into the realm of unsupervised methods, with a focus on dimensionality reduction, and clustering. Clustering, the process of grouping similar data points into meaningful clusters or categories, unveils hidden structures and relationships within the data. Simultaneously, dimensionality reduction techniques, such as Autoencoders and UMAP Embedding, simplify the data by reducing its feature dimensions, facilitating enhanced data analysis, visualization, and model training. This research explores a spectrum of clustering algorithms, including Mountain Clustering, Subtractive Clustering, K-Means Clustering, Fuzzy C-Means Clustering, and Spectral Clustering, with the goal of identifying data point groups that share commonalities, thus enabling the discovery of pertinent data patterns and structures. The performance of these clustering algorithms is rigorously assessed using evaluation indices like the Silhouette Score, Davies-Bouldin Index, and Rand Index, providing a quantitative means of gauging their effectiveness and applicability in various data analysis scenarios.**

***Index Terms*—dimensionality reduction, clustering algorithms, autoencoders, UMAP Embedding, Silhouette Score, Davies-Bouldin Index, Rand Index**

I. Introduction

Unsupervised learning plays a pivotal role in extracting valuable insights from large and complex datasets. Unsupervised methods encompass a wide range of tasks, including clustering and dimensionality reduction, which are essential for uncovering hidden structures within data.

Clustering involves grouping similar data points into clusters or categories. These clusters help reveal underlying patterns and relationships within the data. Dimensionality reduction, on the other hand, focuses on reducing the number of features or dimensions in a dataset while preserving its essential information. This reduction simplifies data analysis, visualization, and model training.

This paper delves into the world of unsupervised methods, exploring data preprocessing techniques, dimensionality reduction algorithms, and clustering methods. Our aim is to provide a comprehensive overview of these techniques and evaluate their performance in various scenarios.

In this research, we investigate the application of Autoencoders, a powerful tool for dimensionality reduction, which can be applied to both lower and higher dimensional spaces. Additionally, we explore UMAP Embedding, a dimensionality reduction technique that leverages the intrinsic structure of data.

We explore the following clustering algorithms: Mountain Clustering, Subtractive Clustering, K-Means Clustering, Fuzzy C-Means Clustering, and Spectral Clustering. These methods aim to identify groups of data points that share similarities, aiding in the discovery of meaningful patterns and structures within the data. By comparing and contrasting these clustering techniques, we seek to shed light on their effectiveness and applicability.

Furthermore, we introduce a set of evaluation indexes to assess the quality of the clusters produced by these algorithms, facilitating a comprehensive analysis of their performance. These indexes are the Silhouette Score, Davies-Bouldin Index, and Rand Index. These indexes offer a quantitative means of assessing the quality of clusters generated by different algorithms.

II. Methodology

A. Data Preprocessing

1) Range Normalization: Range normalization is used to transform any dataset into a common scale. The goal is to map the values of the dataset to the hypercube $[0, 1] \times n$, where n is the dimensions of the data. This ensures that all features have the same scale and lie within a consistent range.

2) Autoencoder: An autoencoder is a type of neural network architecture used for unsupervised learning and dimensionality reduction. It consists of an encoder and a decoder. The encoder encodes the input data into a lower-dimensional representation, often referred to as the latent space or bottleneck layer. Subsequently, the decoder reconstructs the data from this lower-dimensional representation. Autoencoders are useful for tasks like feature learning, denoising, and anomaly detection.

By controlling the dimensions of the latent space, this paper will use autoencoders to perform both embedding (encoding to a lower-dimensional space) and dimensionality expansion (encoding to a higher-dimensional space).

3) UMAP Embedding: UMAP, or Uniform Manifold Approximation and Projection, is a dimensionality reduction technique used for visualizing and embedding high-dimensional data into lower-dimensional spaces. UMAP is based on the manifold assumption, which posits that the data lies on a low-dimensional manifold within the high-dimensional space. UMAP leverages a non-linear approach to capture the intrinsic structure of the data, preserving local and global relationships.

UMAP is particularly effective for visualizing complex datasets and is commonly used in exploratory data analysis and

clustering. It provides a powerful alternative to methods like t-SNE (t-Distributed Stochastic Neighbor Embedding) and PCA (Principal Component Analysis) for dimensionality reduction and visualization tasks.

This paper will explore UMAP to reduce the normalized data into 2 dimensions and three dimensions.

B. Clustering Algorithms

1) Mountain Clustering: Consider a collection of n data points $\{x_1, \ldots, x_n\}$ in an M-dimensional space. The data points are assumed to have been normalized within a hypercube. Then consider a grid formed within that hypercube, where the vertices V are candidates for cluster centers. The number of vertices should be lesser or equal to n. A density measure at data point v is defined as the *Mountain Function* (1):

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

where σ is a constant. Each data point x_i contributes to the height of the mountain function at v, and the contribution is inversely proportional to the distance between x_i and v. Therefore, a data point will have a high density value if it has many neighboring data points. σ defines the height and smoothness of the mountain; data points outside this radius contribute only slightly to the density measure.

After the density measure of each data point has been calculated, the data point with the highest density measure is selected as the first cluster center. Let c_1 be the point selected, and m_{c_1} its density measure.

The density measure for each grid vertex v is revised by the formula (2):

$$m_{new} = m(v) - m(c_1) \exp\left(-\frac{||v - c_1||^2}{2 \cdot \beta^2}\right) \tag{2}$$

where β is another constant. Generally, β is set to be 1.5 times σ to prevent closely spaced cluster centers.

After the density measure for each vertex is revised, the next cluster center c_2 is selected, and all of the density measures for vertices are revised again. This process is repeated until a sufficient number of cluster centers are generated or until a stop criterion is met.

2) Subtractive Clustering: Consider a collection of n data points $\{x_1, \ldots, x_n\}$ in an M-dimensional space. The data points are assumed to have been normalized within a hypercube. Each data point is a candidate for cluster centers. A density measure at data point x is defined as (3):

$$D_i = \sum_{j=1}^{n} \exp\left(-\frac{||x_i - x_j||^2}{(\frac{ra}{2})^2}\right) \tag{3}$$

where ra is a positive constant. Therefore, a data point will have a high density value if it has many neighboring data points. The radius ra defines a neighborhood; data points outside this radius contribute only slightly to the density measure.

After the density measure of each data point has been calculated, the data point with the highest density measure is selected as the first cluster center. Let x_{c1} be the point selected, and D_{c1} its density measure.

The density measure for each data point x_i is revised by the formula (4):

$$D_i = D_i - D_{c1} \exp\left(-\frac{||x_i - x_{c1}||^2}{(\frac{rb}{2})^2}\right) \tag{4}$$

where rb is a positive constant. Generally, rb is set to be 1.5 times ra to prevent closely spaced cluster centers.

After the density measure for each data point is revised, the next cluster center x_{c2} is selected, and all of the density measures for data points are revised again. This process is repeated until a sufficient number of cluster centers are generated.

3) K-Means Clustering: K-means partitions a collection of n vectors $x_j, j = 1, \ldots, n$ into c groups $G_i, i = 1, \ldots, c$, and finds a cluster center in each group such that a cost function (or an objective function) of dissimilarity (or distance) measure is minimized.

With a generic distance function $d(x_k, c_i)$ as the dissimilarity measure between a vector x_k in group j and the corresponding cluster center c_i, the cost function can be defined by Equation (5):

$$J = \sum_{i=1}^{c} \sum_{k, x \in G_i} d(x_k, c_i) \tag{5}$$

Thus, the value of J_i depends on the geometrical properties of G_i and the location of c_i.

The partitioned groups are typically defined by a $c \times n$ binary membership matrix U, where element u_{ij} is 1 if the j-th data point x_j belongs to group i, and 0 otherwise. Once the cluster centers c_i are fixed, u_{ij} can be found as Equation (6):

$$u_{ij} = \begin{cases} 1, & \text{if } \|x_j - c_i\|^2 \leq \|x_j - c_k\|^2, \text{ for each } k \neq i \end{cases} \tag{6}$$

Equation (6) states that x_j belongs to group i if c_i is the closest center among all centers. Since a data point can only be in one group, the membership matrix U has the properties as follows:

$$\sum_{i=1}^{c} u_{ij} = 1, \quad \text{for all } j = 1, \ldots, n$$

and

$$\sum_{j=1}^{n} \sum_{i=1}^{c} u_{ij} = n$$

If u_{ij} is fixed, then the optimal center c_i that minimizes Equation (3.1) is the mean of all vectors in group i:

$$c_i = \frac{1}{|G_i|} \sum_{k, x_k \in G_i} x_k$$

where $|G_i|$ is the size of G_i.

The algorithm is repeated until you reach a certain stop criteria.

4) Fuzzy C-Means Clustering: Fuzzy C-Means clustering is a data clustering algorithm that introduces the concept of fuzzy partitioning. Unlike traditional clustering methods where a data point belongs to a single cluster, it assigns each data point to multiple clusters with degrees of belongingness specified by membership grades.

The primary objective is to partition a collection of n data vectors, denoted as $x_i, i = 1, \ldots, n$, into c fuzzy groups and determine the cluster center for each group. The core idea is to minimize a cost function based on a dissimilarity measure. The membership matrix U contains elements with values between 0 and 1, representing the degree of belongingness of each data point to each cluster. Importantly, these membership grades are subject to normalization, ensuring that the sum of the degrees of belongingness for a data point always equals unity.

The cost function is represented as:

$$J = \sum_{i=1}^{c} \sum_{j=1}^{n} (u_{ij})^m \cdot d(x_j, c_i) \tag{7}$$

Where:

- u_{ij} is the membership grade of data point j in cluster i, with values between 0 and 1.
- m is a fuzziness parameter that determines the degree of fuzziness in the partition.
- c_i is the center of cluster i.
- $d(x_j, c_i)$ represents the dissimilarity between data point j and cluster center c_i.

To find the minimum of this cost function, necessary conditions for Equation (7) to reach a minimum are:

$$c_i = \frac{\sum_{j=1}^{n} u_{ij}^m x_j}{\sum_{j=1}^{n} u_{ij}^m}$$

and

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

These conditions and the FCM algorithm enable data clustering with soft boundaries, making it a valuable tool for scenarios where data points may exhibit varying degrees of similarity to multiple clusters.

5) Spectral Clustering: Spectral Clustering is a powerful data clustering technique that leverages the spectral properties of the data to group similar data points together. Unlike traditional clustering methods that operate in the original feature space, Spectral Clustering transforms the data into a new representation based on the eigenvalues and eigenvectors of a similarity or affinity matrix.

The key steps involved in Spectral Clustering can be summarized as follows:

- **Affinity or Similarity Matrix Construction** The first step in Spectral Clustering is to construct an affinity or similarity matrix, denoted as W. This matrix quantifies the pairwise relationships between data points. Common choices for constructing W include nearest neighbor graphs or Gaussian affinity functions. The choice of the affinity matrix construction method depends on the characteristics of the data and the desired clustering results.
- **Laplacian Matrix Computation** From the affinity matrix W, the Laplacian matrix L is computed. The Laplacian matrix captures the graph structure of the data and is used to uncover clusters.
- **Eigenvalue Decomposition** Spectral Clustering proceeds by performing an eigenvalue decomposition of the Laplacian matrix L. This decomposition results in a set of eigenvalues and their corresponding eigenvectors. The eigenvectors are associated with specific eigenvalues and reveal the underlying structure of the data.
- **Cluster Assignment** Once the eigenvectors are obtained, the data points are projected into a lower-dimensional space defined by these eigenvectors. The reduced-dimensional data is then subjected to K-Means, to assign data points to clusters.

Spectral Clustering offers several advantages, including its ability to identify non-convex and complex clusters, making it particularly suitable for challenging clustering scenarios. Additionally, it can uncover clusters of varying shapes and sizes, offering flexibility in capturing the intrinsic data structure.

C. Measurements

1) Intracluster Indexes: Intracluster indexes are used to evaluate the quality of clusters from within each cluster. They provide insights into how tightly data points within the same cluster are grouped.

- **Silhouette Score**
  The Silhouette Score measures how similar an object is to its own cluster (cohesion) compared to other clusters (separation). It takes values in the range [-1, 1], where a higher score indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters. The formula for the Silhouette Score is as follows:

  $$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

  where:
  - $S(i)$ is the Silhouette Score for data point i.
  - $a(i)$ is the average distance from data point i to the other data points in the same cluster.
  - $b(i)$ is the smallest average distance from data point i to the data points in a different cluster, minimized over clusters.

  Silhouette scores can range from -1 to 1. A score of 1 indicates that a point is perfectly assigned to its cluster, while a score of -1 indicates that a point is completely misassigned to its cluster. A score of 0 indicates that a point is on the border between two clusters.

TABLE I: Silhouette Scores and Interpretations

Silhouette Score	Interpretation
> 0.5	Good clustering
0 to 0.5	Fair clustering
-0.5 to 0.5	Poor clustering
$<$ -0.5	Very poor clustering

- **Davies-Bouldin Index**
  The Davies-Bouldin Index measures the average similarity between each cluster and its most similar cluster. The formula for the Davies-Bouldin Index is as follows:

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

  where:
  - DB is the Davies-Bouldin Index.
  - n is the number of clusters.
  - σ_i is the average distance between data points in cluster i.
  - $d(c_i, c_j)$ is the distance between the centroids of clusters i and j.

  The Davies-Bouldin Index can take on values from 0 to infinity. A lower score indicates better clustering, with a score of 0 indicating perfect clustering. A score of infinity indicates that the clusters are completely overlapping. Table II shows the interpretations for the possible scores:

TABLE II: Davies-Bouldin Scores and Interpretations

Davies-Bouldin Score	Interpretation
< 0.5	Excellent clustering
0.5 to 1	Good clustering
1 to 2	Fair clustering
2 to 3	Poor clustering
> 3	Very poor clustering

- **Calinski-Harabasz Index** The Calinski-Harabasz index (CH index) is a measure of the compactness and separation of clusters. A higher CH index indicates better clustering.
  The CH index is calculated by taking the ratio of the between-cluster variance to the within-cluster variance. The between-cluster variance is the variance of the cluster means, and the within-cluster variance is the average variance of the points within each cluster.
  The CH index is defined as follows:
  CH index = (Sum of squared distances between cluster centers) / (Sum of squared distances between data points and their cluster centers)
  The CH index can take on values from 0 to infinity. A higher CH index indicates that the clusters are more compact and well-separated. A CH index of 0 indicates that all of the points in the same cluster, and a CH index of infinity indicates that the clusters are completely overlapping.
  In general, a CH index of greater than 1 is considered to be good, and a CH index of greater than 2 is considered to be excellent. However, the optimal CH index for a particular dataset will depend on the nature of the data and the desired clustering results.

TABLE III: Calinski-Harabasz Index Scores and Interpretations

Calinski-Harabasz Index Score	Interpretation
> 1	Good clustering
> 2	Excellent clustering
< 1	Poor clustering

Extracluster indexes assess the separation between different clusters. They help evaluate the quality of clustering based on the differences between clusters.

- **Adjusted Rand Index**
  The Rand Index is a measure of the similarity between two data clusterings. It is defined as the ratio of the number of agreements between the two clusterings and the total number of data point pairs. The formula for the Rand Index is as follows:

$$RI = \frac{a + b}{C(2, n)}$$

  where:
  - RI is the Rand Index.
  - a is the number of data point pairs that are in the same cluster in both clusterings.
  - b is the number of data point pairs that are in different clusters in both clusterings.
  - $C(2, n)$ is the total number of data point pairs in the dataset, which is the binomial coefficient.

TABLE IV: Adjusted Rand Index Interpretations

Adjusted Rand Index	Interpretation
> 0.7	Strong agreement
0.5 to 0.7	Moderate agreement
< 0.5	Weak or random agreement

- **Normalized Mutual Information (NMI)**
  The Normalized Mutual Information measures the amount of information shared between true labels and predicted cluster labels, normalized to fall between 0 and 1. Higher values indicate better agreement.

TABLE V: Normalized Mutual Information Interpretations

NMI Score	Interpretation
> 0.7	Strong agreement
0.5 to 0.7	Moderate agreement
< 0.5	Weak or random agreement

- **Fowlkes-Mallows Index (FMI)**
  The Fowlkes-Mallows Index is a measure of the geometric mean of the pairwise precision and recall of the clustering results. Higher values indicate better clustering.

TABLE VI: Fowlkes-Mallows Index Interpretations

FMI Score	Interpretation
> 0.7	Strong agreement
$0.5 - 0.7$	Moderate agreement
< 0.5	Weak or random agreement

III. Results Iris Dataset

A. Data exploration

The Iris dataset is a classic dataset in machine learning and pattern recognition. It was first published by Sir R.A. Fisher in 1936, and has been used in numerous research papers and books. The dataset contains four features: sepal length, sepal width, petal length, and petal width. The target variable is the species of iris flower, which can be one of three classes: Iris-Setosa, Iris-Versicolour, or Iris-Virginica.

Table VII summarizes the statistics of the Iris dataset:

TABLE VII: Summary Statistics for the Iris Dataset

Feature	Min	Max	Mean	SD	Class Correlation
Sepal length	4.3	7.9	5.84	0.83	0.7826
Sepal width	2.0	4.4	3.05	0.43	-0.4194
Petal length	1.0	6.9	3.76	1.76	0.9490
Petal width	0.1	2.5	1.20	0.76	0.9565

A pairplot shows pairwise relationships in a dataset. Fig. 1 shows the pairplot for the Iris dataset, with the diagonal displaying a univariate distribution plot that shows the marginal distribution of the data in each column.

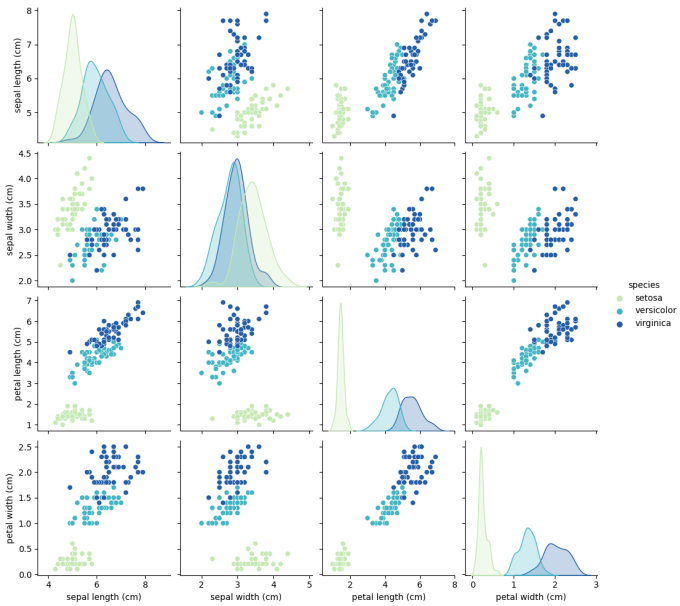


Fig. 1: Iris Pairplot

The Iris setosa species is located in a separate cluster at the bottom of the plot. This cluster is well-separated from the other two clusters, which suggests that the Iris setosa species is the most distinct from the other two species. The Iris versicolor and Iris virginica species are located in two overlapping clusters at the top of the plot. The overlap between these two clusters suggests that the Iris versicolor and Iris virginica species are more closely related to each other than they are to the Iris setosa species.

The pairplot shows that there is a strong correlation between sepal length and petal length, and between sepal width and petal width. There is also some correlation between sepal length and sepal width, but it is not as strong.

The pairplot also shows that the three species of Iris are well-separated in the feature space. This suggests that it would be relatively easy to develop a machine learning model to classify Iris flowers based on their sepal and petal measurements.

An KDE plot is a visualization of the distribution of the data using a kernel density estimate (KDE). A KDE is a non-parametric method for estimating the probability density function of a random variable. It works by smoothing the data with a kernel function, which is a weighted function that is centered on each data point. The resulting KDE plot is a continuous curve that shows the probability of a data point falling within a given range of values.

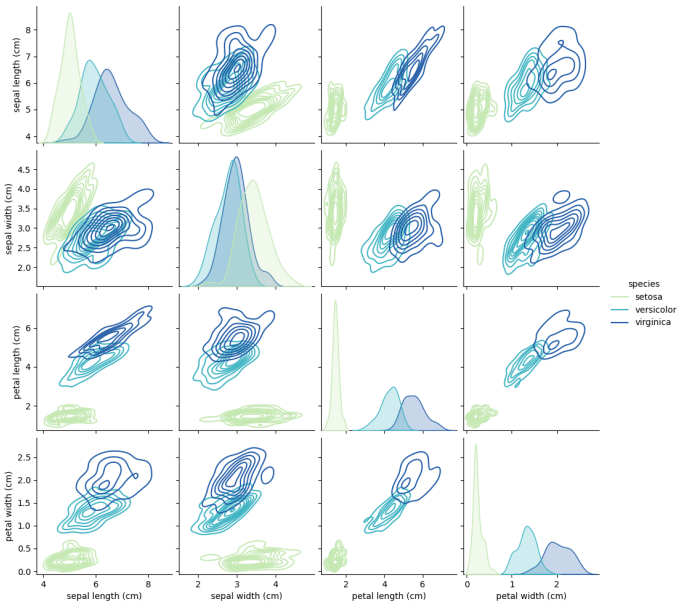


Fig. 2: Iris KDE plot

The KDE plot shows that the three species are all distributed very differently according to the variable considered. The sepal length and width features have the smallest variances for all species, while the petal length and width features have the largest variances.

For example, the sepal length of setosa flowers is typically shorter than the sepal length of versicolor or virginica flowers. The sepal width of setosa flowers is also typically narrower than the sepal width of versicolor or virginica flowers. Similarly, the petal length and width of setosa flowers are typically shorter and narrower than the petal length and width of versicolor or virginica flowers.

The KDE plot also shows that there is some overlap in the distributions of the three species of Iris flowers. For example, there are some setosa flowers that have sepal lengths that are similar to the sepal lengths of versicolor or virginica flowers. However, the overall distribution of the four features is different for the three species of flowers.

B. Preprocessing

1) Range Normalization: After performing range normalization, we can observe the same structures present in the data. However, the data is now in the same scale to make distance-related analysis more convenient and coherent.

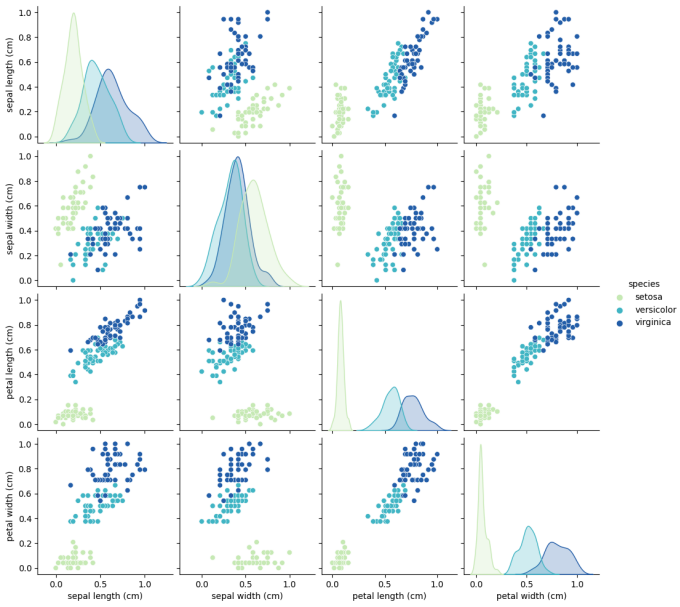


Fig. 3: Range-Normalized Iris

2) UMAP Embedding: The two-dimensional embedding of the Iris dataset using Uniform Manifold Approximation and Projection (Fig. 4) shows that the three species of iris flowers are well-separated in this space, and they show a distinct shape. Although each attempt to reduce the data to this space produced different outcomes, the shape that each species showed was preserved.

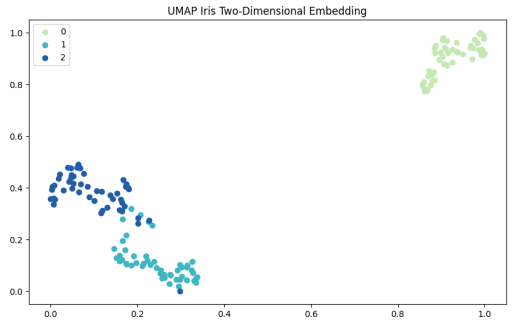


Fig. 4: UMAP Two-Dimensional Embedding

Now, for the three-dimensional embedding (Fig. 5), similar results were obtained. The shape was very distinct in every attempt, and all three species were very clearly separated.

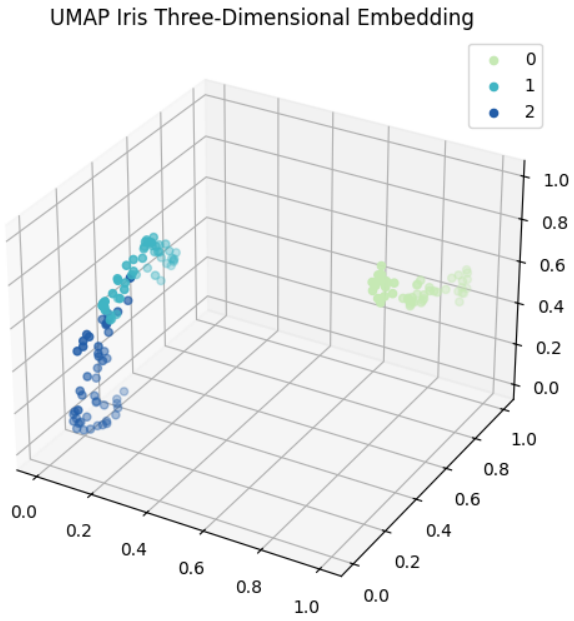


Fig. 5: UMAP Three-Dimensional Embedding

3) Autoencoder Embedding: We first reduced the dimensionality, attempting to cluster in the two-dimensional and three-dimensional space. The two-dimensional reduction (Fig. 6) shows that the three species of iris flowers are well-separated in this space, but show no distinct shape.

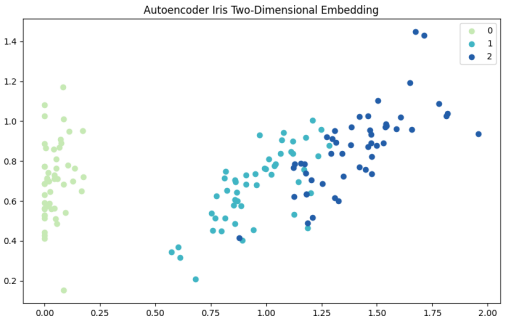


Fig. 6: Autoencoder Two-Dimensional Embedding

Then, the three-dimensional reduction (Fig. 7) shows similar results.

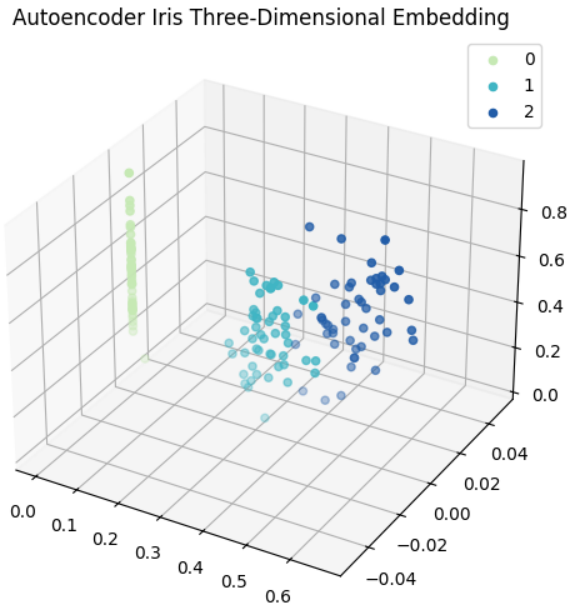


Fig. 7: Autoencoder Three-Dimensional Embedding

We then augmented the variable space to five-dimensional space (Fig. 8). This is were the shapes and behaviours of the species were altered radically. We can particularly note that iris versicolor and iris virginica completely merged in most visualuzations of this space, which complicates clustering.

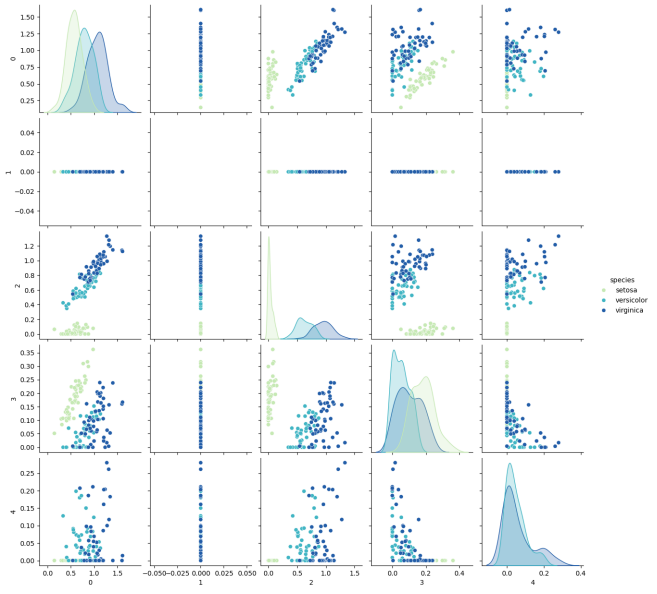


Fig. 8: Autoencoder Five-Dimensional Embedding

A very curious behavior was observed in three-dimensions and higher, where a certain specie took the shape of a straight line, evidenced in (Fig. 7) with iris setosa, and in (Fig. 8) with the second dimension.

C. Visualizing the space with different metrics

In this section, we will explore the Iris Normalized space using different metrics. The distance matrix is a square matrix that contains the distances between all pairs of points in a dataset. The x-axis and y-axis of the plot represent the indices of the points in the dataset, and the z-axis represents the distances between the points.

Fig. 9 shows a 3D surface plot of a Manhattan distance matrix (Manhattan distance $= \sum_{i=1}^{n} |a_i - b_i|$) The plot shows that the points in the dataset are arranged in a grid-like pattern. The points in the top and bottom edges of the grid are closer together than the points on the right and left edges of the grid. This suggests that the points in this area of the grid are more similar to each other than the points on the other edges of the grid.

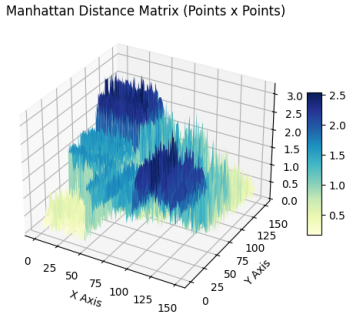


Fig. 9: Iris Normalized Data Viewed by Manhattan Distances Point to Point

The plot also shows that the distances between the points are not uniform. There are some regions of the plot where the points are closer together than in other regions. This suggests that there are some clusters of points in the dataset.

Fig. 10 shows a 3D surface plot of a Euclidean distance matrix (Euclidean distance $= \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$).

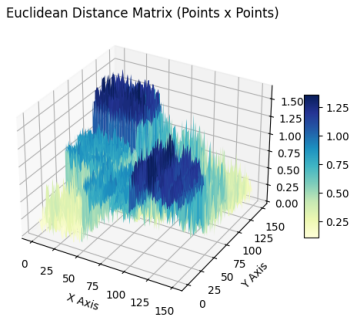


Fig. 10: Iris Normalized Data Viewed by Euclidean Distances Point to Point

Fig. 10 shows a 3D surface plot of a Minkowski distance matrix with $p = \infty$ (Minkowski distance with $p = \infty = \max_{i=1}^{n} |a_i - b_i|$).

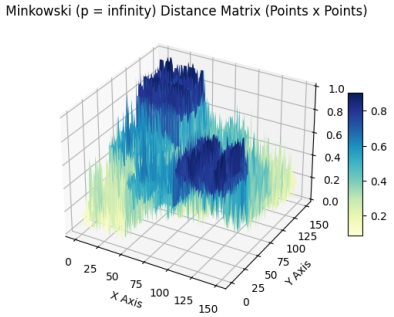


Fig. 11: Iris Normalized Data Viewed by Minkowski $p = \infty$ Distances Point to Point

All three Minkowski Distances show very similar results, with $p = \infty$ showing the most peaks present.

Fig. 12 shows a 3D surface plot of a Cosine distance matrix (Cosine distance $(= 1 - \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \cdot \|\mathbf{b}\|})$. The cosine distance matrix is a measure of similarity between two vectors, and it ranges from 0 (identical vectors) to 1 (orthogonal vectors).

The x-axis and y-axis of the plot represent the distances between the two points, and the z-axis represents the cosine distance between the two points. The higher the z-axis value, the more similar the two points are.

The plot shows that the cosine distance between the two points is highest (closest to 0) when the two points are close to each other, and it decreases as the two points move further apart. This is to be expected, as vectors that are close to each other are more likely to be similar than vectors that are far apart.

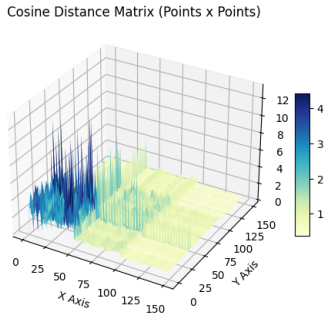


Fig. 12: Iris Normalized Data Viewed by Cosine Distances Point to Point

D. Hyper-parameter Tuning

For tuning each clustering algorithm', we attempted several combinations of their hyper-parameters with 100 runs. Then we measured three intracluster indexes for each clustering results. The information found on the table represents the average for each combination of hyper-parameter.

Table VIII shows the results of a mountain custering hyperparameters tuning test. The different hyperparameters that were tested are σ and β. The table shows the silhouette score, Davies-Bouldin score, and Calinski-Harabasz score for each combination of parameters.

Based on the results of the hyperparameter tuning test, the best combination of hyperparameters is $\sigma = 0.5$ and $\beta = 0.625$.

TABLE VIII: Mountain Clustering Hyperparameter Tuning

σ	β	**Silhoutte**	**Davies Bouldin**	**Calinski Harabasz**
0.2	0.250	0.374	0.915	267.778
0.3	0.375	0.482	0.786	347.612
0.4	0.500	0.433	0.920	290.621
0.5	0.625	0.502	0.756	339.011
0.6	0.750	0.346	0.842	184.125
0.7	0.875	0.308	0.854	156.348
0.8	1.000	0.309	0.866	144.000

Table IX shows the results of a subtractive custering hyperparameters tuning test. The different hyperparameters that were tested are r_α and r_β. The table shows the silhouette score, Davies-Bouldin score, and Calinski-Harabasz score for each combination of parameters.

Based on the results of the hyperparameter tuning test, the best combination of hyperparameters is $r_\alpha = 0.5$ and $r_\beta = 0.75$.

TABLE IX: Subtractive Clustering Hyperparameter Tuning

r_α	r_β	**Silhoutte**	**Davies Bouldin**	**Calinski Harabasz**
0.2	0.300	0.199	0.740	136.770
0.3	0.450	0.235	0.881	148.775
0.4	0.600	0.277	0.923	191.658
0.5	0.750	0.507	0.755	357.555
0.6	0.900	0.507	0.755	357.555
0.7	1.050	0.507	0.747	350.706
0.8	1.200	0.507	0.747	350.706

Table X shows the results of a K-Means hyperparameters tuning test. The hyperparameter that was tested is ε. The table shows the silhouette score, Davies-Bouldin score, and Calinski-Harabasz score for each combination of parameters.

Based on the results of the hyperparameter tuning test, the best hyperparameter is $\varepsilon = 0.45$.

TABLE X: K Means Hyperparameter Tuning

ε	**Silhoutte**	**Davies Bouldin**	**Calinski Harabasz**
0.05	0.486	0.794	314.279
0.10	0.484	0.793	305.363
0.15	0.479	0.799	311.664
0.20	0.479	0.798	303.187
0.25	0.486	0.801	304.028
0.30	0.485	0.785	320.721
0.35	0.483	0.799	302.963
0.40	0.483	0.789	292.030
0.45	0.487	0.781	315.753
0.50	0.482	0.788	310.254

Table XI shows the results of a Fuzzy C-Means hyperparameters tuning test. The hyperparameter that was tested is m. The table shows the silhouette score, Davies-Bouldin score, and Calinski-Harabasz score for each combination of parameters.

Based on the results of the hyperparameter tuning test, the best hyperparameter is $m = 2.1$.

TABLE XI: Fuzzy C Means Hyperparameter Tuning

m	Silhoutte	Davies Bouldin	Calinski Harabasz
1.6	0.478	0.802	343.708
1.9	0.471	0.763	333.258
2.0	0.470	0.757	322.717
2.1	0.473	0.763	335.574
2.2	0.461	0.758	316.502
2.5	0.461	0.755	319.136
2.8	0.441	0.817	298.136

Table XII shows the results of a Spectral Clustering hyperparameters tuning test. The hyperparameter that was tested is σ. The table shows the silhouette score, Davies-Bouldin score, and Calinski-Harabasz score for each combination of parameters.

Based on the results of the hyperparameter tuning test, the best hyperparameter is $\sigma = 0.05$.

TABLE XII: Spectral Clustering Hyperparameter Tuning

σ	Silhoutte	Davies Bouldin	Calinski Harabasz
0.050	0.524	0.437	170.830
0.100	0.479	0.560	185.024
0.200	0.473	0.549	188.519
0.300	0.470	0.525	184.221
0.400	0.469	0.521	187.152
0.500	0.472	0.554	188.650
0.600	0.448	0.570	180.716
0.700	0.450	0.613	182.595
0.800	0.432	0.661	169.643
0.900	0.405	0.703	155.215
1.000	0.425	0.631	168.771

E. Visualizing Experiments

Fig. 13 shows the silhouette scores of the five clustering algorithms for iris normalized data. A silhouette score indicates that a point is well-matched to its own cluster and poorly matched to other clusters.

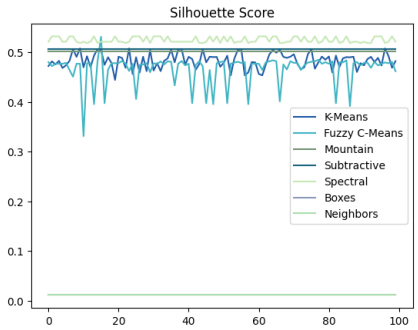


Fig. 13: Silhouette Score for all Clustering Methods in Normalized Data

The plot shows that the Spectral algorithm has the highest silhouette score, followed by Subtractive and Mountain clustering. The K Means and Fuzzy C Means clustering algorithms have lower silhouette scores, followed by the Boxes and Neighbors clustering algorithms.

Fig. 14 shows the Davies-Bouldin index (DBI) scores for different clustering algorithms on iris normalized data. The DBI score is a measure of how well-separated the clusters are.

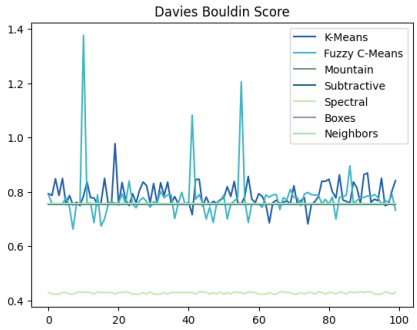


Fig. 14: Davies Bouldin Score for all Clustering Methods in Normalized Data

The plot shows that the Fuzzy C Means algorithm has the highest davies bouldin score, followed by K Means. The Mountain, Subtractive, Boxes and Neigbors clustering algorithms have lower davies bouldin scores, followed by the Spectral clustering algorithms.

Fig. 15 shows the Calinski-Harabasz (CH) index scores for different clustering algorithms on a given dataset. The CH index is a measure of how well-defined the clusters are. A higher CH index score indicates better cluster definition.

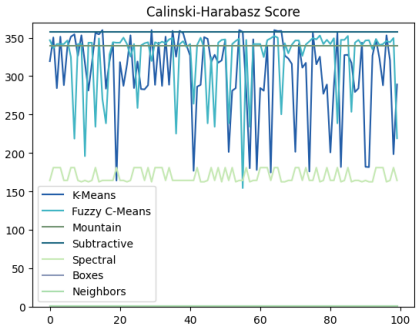


Fig. 15: Calinski Harabasz Score for all Clustering Methods in Normalized Data

The plot shows that the Substractive algorithm has the highest CH index score, followed by Mountain clustering. The K-Means and Fuzzy C Means clustering algorithms have lower CH index scores, followed by the Spectral clustering algorithms.

Fig. 16 shows the Adjusted Rand index scores for different clustering algorithms on iris normalized data. The Adjusted Rand score is a measure of the similarity between two clusterings.

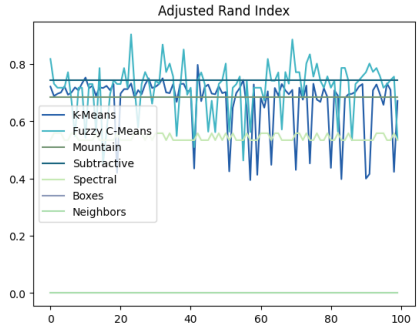


Fig. 16: Adjusted Rand Index for all Clustering Methods in Normalized Data

The plot shows that the Substractive algorithm has the highest Adjusted Rand index score, followed by Mountain clustering. The K-Means and Fuzzy C Means clustering algorithms have lower Adjusted Rand index scores, followed by the Spectral clustering algorithms.

Fig. 17 shows the Normalized Mutual Information for different clustering algorithms on iris normalized data. Normalized Mutual Information is a measure of the mutual information between two clusterings, normalized by the average cluster entropy.

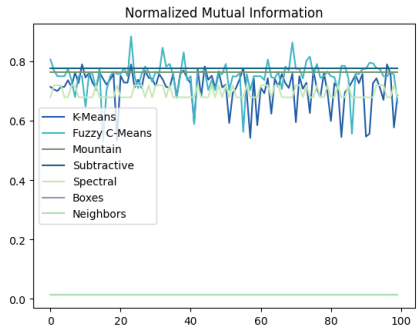


Fig. 17: Normalization Mutual Information for all Clustering Methods in Normalized Data

The plot shows that the Substractive algorithm has the highest Normalized Mutual Information, followed by Mountain clustering.

Fig. 18 shows the Fowlkes-Mallows Index scores for different clustering algorithms on iris normalized data. The Fowlkes-Mallows Index is a measure of the overlap between two clusterings.

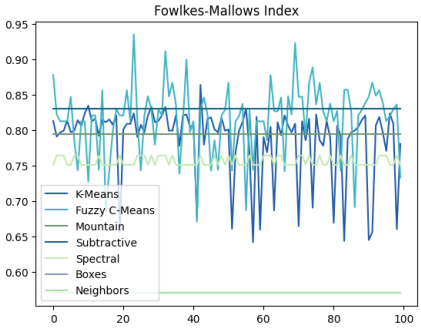
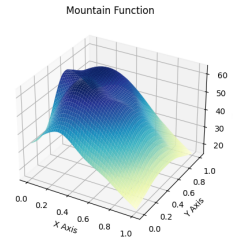


Fig. 18: Fowlkes-Mallows Index for all Clustering Methods in Normalized Data

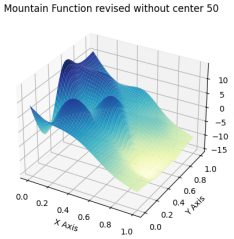
The plot shows that the Substractive algorithm has the highest Fowlkes-Mallows Index score, followed by Mountain clustering. The K-Means and Fuzzy C Means clustering algorithms have lower Fowlkes-Mallows Index scores, followed by the Spectral clustering algorithms.

F. Mountain Function Surfaces

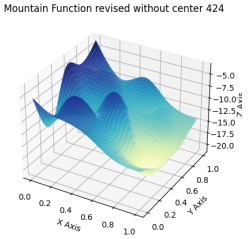
Fig. 19 shows three different surfaces of the mountain function, each at a different step in the optimization process. The mountain function is a mathematical function that has a peak at its center and gradually decreases towards the edges. The goal of the optimization process is to find the parameters of the mountain function that best fit a given dataset.



(a) Step 1



(b) Step 2



(c) Step 3

Fig. 19: Mountain Function Surfaces

Fig. 19a: Step 1

This plot shows the mountain function without a center. The function is symmetrical around the origin, and it has a peak at (0, 0). However, the function is also quite rough, and it has many small bumps and valleys.

Fig. 19b: Step 2

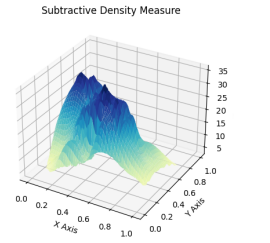
This plot shows the mountain function after the first step of the optimization process. The center of the mountain function has now been found, and the function is smoother than it was in the previous plot. However, the peak of the function is not yet at its maximum height.

Fig. 19c): Step 3

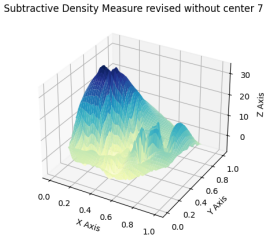
This plot shows the mountain function after the second step of the optimization process. The function is now even smoother than it was in the previous plot, and the peak of the function is closer to its maximum height.

G. Subtractive Density Function Surfaces

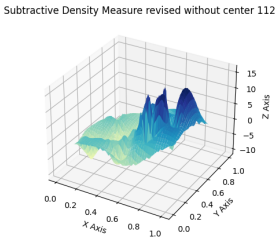
Fig. 20 shows the results of a subtractive density measure at three different steps. In the first step, the subtractive density measure is shown without a center. In the second step, the subtractive density measure is revised without a center. In the third step, the subtractive density measure is revised without a center and with a different scale.



(a) Step 1



(b) Step 2



(c) Step 3

Fig. 20: Subtractive Density Surfaces

H. Visualizing Results

We will use radar plots to show the results of clustering our data using five different clustering algorithms: Mountain Clustering, Subtractive Clustering, K-Means, Fuzzy C-Means, and Spectral Clustering. The radar plot has five axes, each representing a different clustering evaluation metric: Silhouette, Davies-Bouldin, Adjusted Rand Index, Normalized Mutual Information, and Fowlkes Mallows.

Our experiment consists of first using Mountain and Subtractive Clustering to determine the k number of clusters naturally present in the data. Then, we run the other three algorithms using that k as a hyper-parameter.

We run these experiments 100 times for each of our preprocessed data sets: normalized data, UMAP two-dimensionally embedded data, UMAP three-dimensionally embedded data, Autoencoder two-dimensionally embedded data, Autoencoder three-dimensionally embedded data, and Autoencoder five-dimensionally embedded data. We obtain the average of all the results per index and show it in radar plots per dataset.

The radar plot in Fig. 21 shows that Subtractive clustering outperformed the other four algorithms on all four evaluation metrics, except for Davies Bouldin, where Mountain was the best with an almost-perfect score. This suggests that Subtractive clustering was able to find the most natural clusters in the normalized iris data. Spectral clustering was the worst method for most scores.

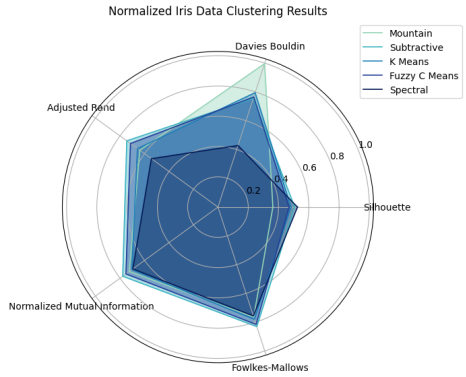


Fig. 21: Normalized Iris Data Scores

The radar chart depicted in Fig. 22 illustrates that Spectral clustering exhibited superior performance compared to the other four algorithms across all evaluation metrics, excluding Silhoutte, where Mountain achieved a much better score. This implies that Spectral clustering successfully identified the most cohesive clusters within the UMAP two-dimensionally embedded data. Notably, Mountain clustering consistently ranked as the least effective method across various scores.

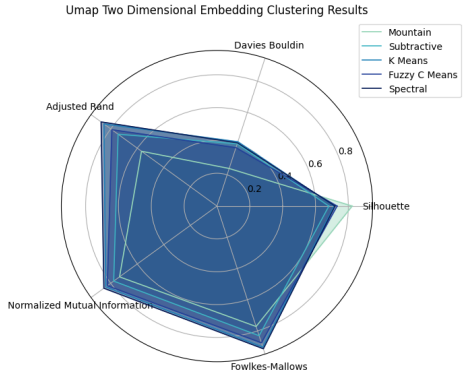


Fig. 22: UMAP Two Dimensional Embedding Scores

The radar plot shown in Fig. 23 indicates that Fuzzy C Means clustering demonstrated better performance than the other four algorithms across all evaluation metrics, except for Silhouette, where Mountain achieved a significantly higher score. This suggests that Fuzzy C Means clustering successfully identified the most coherent clusters within the UMAP three-dimensional embedding. Remarkably, Spectral clustering consistently ranked as the least effective method across different scores.

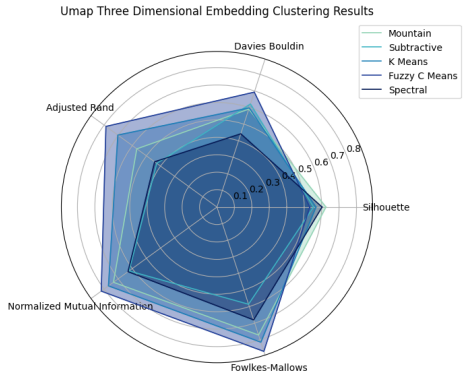


Fig. 23: UMAP Three Dimensional Embedding Scores

The radar plot shown in Fig. 24 indicates that Fuzzy C Means clustering demonstrated better performance than the other four algorithms across all evaluation metrics, except for Silhouette, where Mountain achieved a significantly higher score. This suggests that Fuzzy C Means clustering successfully identified the most coherent clusters within the Autoencoder two-dimensional embedding. Remarkably, Mountain clustering consistently ranked as the least effective method across different scores, except for davies-bouldin, where it was better.

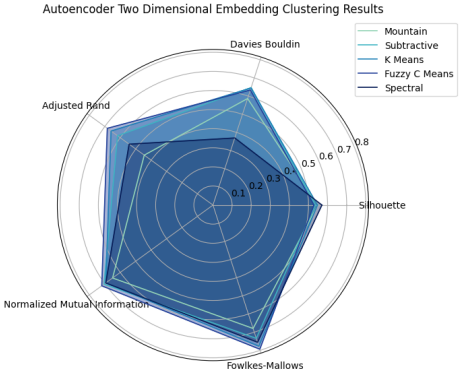


Fig. 24: Autoencoder Two Dimensional Embedding Scores

The radar plot shown in Fig. 25 shows no clear results. There is not one algorithm that ranked best in all or most scores. For Silhouette, Adjusted Rand and Normalized Mutual Information, Subtractive was best and Mountain the worst, but for Davies Bouldind, Mountain was best and Spectral the worst. For Fowlkes Mallows, all five were very similar.

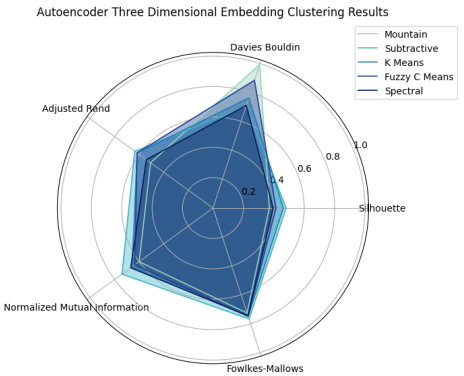


Fig. 25: Autoencoder Three Dimensional Embedding Scores

The radar plot shown in Fig. 26 shows no clear results. There is not one algorithm that ranked best in all or most scores. For Silhouette, Fuzzy C Means was best and Mountain the worst, but for Davies Bouldind, Mountain was best and Spectral the worst.

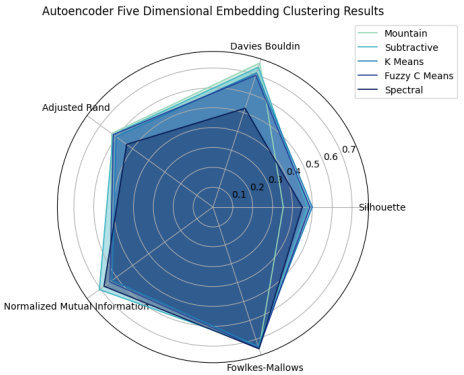


Fig. 26: Autoencoder Five Dimensional Embedding Scores

IV. Results Penguins Dataset

A. Data exploration

The ”penguins” dataset is a popular dataset in the field of data science and is often used for data visualization and machine learning exercises. The dataset contains information about various species of penguins and includes the following columns:

- species: The species of penguin (e.g., Adelie, Chinstrap, Gentoo).
- bill length (mm): The length of the penguin’s bill in millimeters.
- bill depth (mm): The depth of the penguin’s bill in millimeters.
- flipper length (mm): The length of the penguin’s flipper in millimeters.
- : body mass (g): The body mass of the penguin in grams.

Table XIII summarizes the statistics of the Iris dataset:

TABLE XIII: Summary Statistics for the Penguin Dataset

Feature	Min	Max	Mean	SD
Bill length	32.1	59.6	43.99	5.47
Bill depth	13.1	21.5	17.16	1.97
Flipper length	172	231	200.97	14.02
Body Mass	2700	6300	4207.06	805.22

A pairplot shows pairwise relationships in a dataset. Fig. 27 shows the pairplot for the Penguins dataset, with the diagonal displaying a univariate distribution plot that shows the marginal distribution of the data in each column.

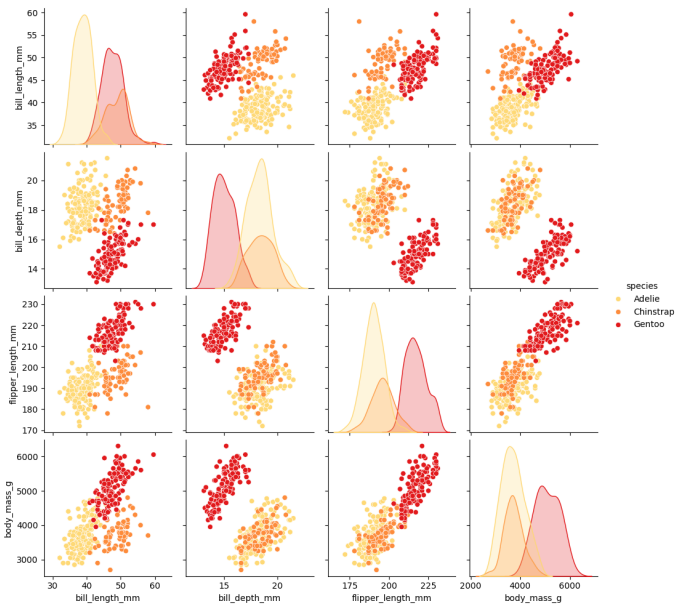


Fig. 27: Penguin Pairplot

The distributions of the four variables vary in shape. The bill length distribution is roughly normal, with a few outliers at the high end. The bill depth distribution is also roughly normal, but with a slight positive skew. The flipper length distribution is more skewed to the right, with a longer tail at the high end. The body mass distribution is the most skewed of the four, with a long tail at the high end.

The scatter plots also reveal some interesting relationships between the variables. For example, there is a strong positive correlation between bill length and body mass. This means that larger penguins tend to have longer bills. There is also a positive correlation between flipper length and body mass, but it is not as strong as the correlation between bill length and body mass.

The different species of penguins also show some differences in the relationships between the variables. For example, the Adelie penguins have a relatively strong correlation between bill length and body mass, while the Gentoo penguins have a relatively weak correlation. The Adelie penguins also have a relatively strong correlation between flipper length and body mass, while the Chinstrap penguins have a relatively weak correlation.

An KDE plot is a visualization of the distribution of the data using a kernel density estimate (KDE). A KDE is a non-parametric method for estimating the probability density function of a random variable. It works by smoothing the data with a kernel function, which is a weighted function that is centered on each data point. The resulting KDE plot is a continuous curve that shows the probability of a data point falling within a given range of values.

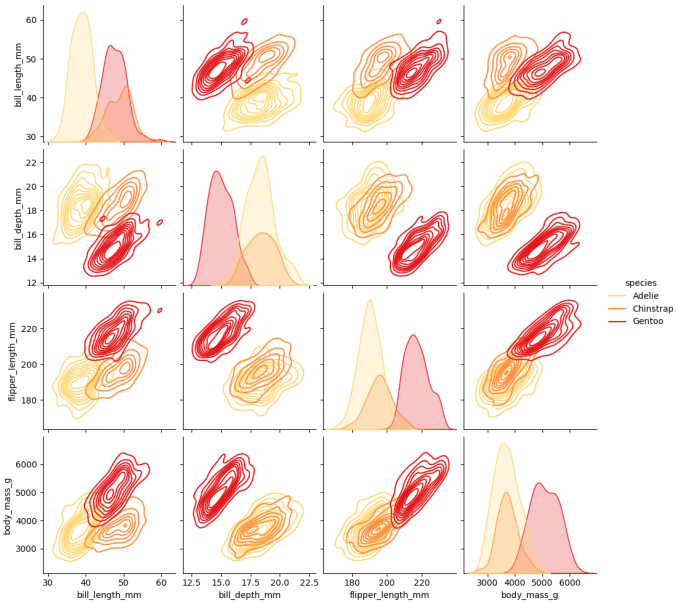


Fig. 28: Penguin KDE plot

B. Preprocessing

1) Range Normalization: After performing range normalization, we can observe the same structures present in the data. However, the data is now in the same scale to make distance-related analysis more convenient and coherent.

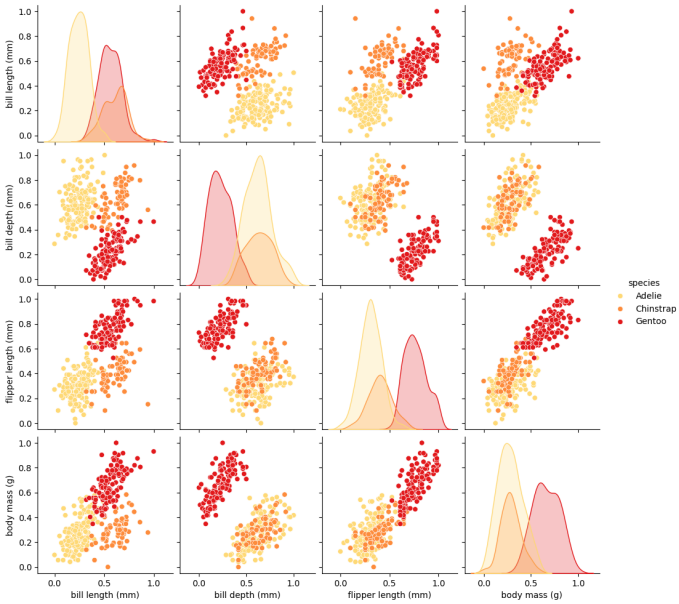


Fig. 29: Range-Normalized Penguins

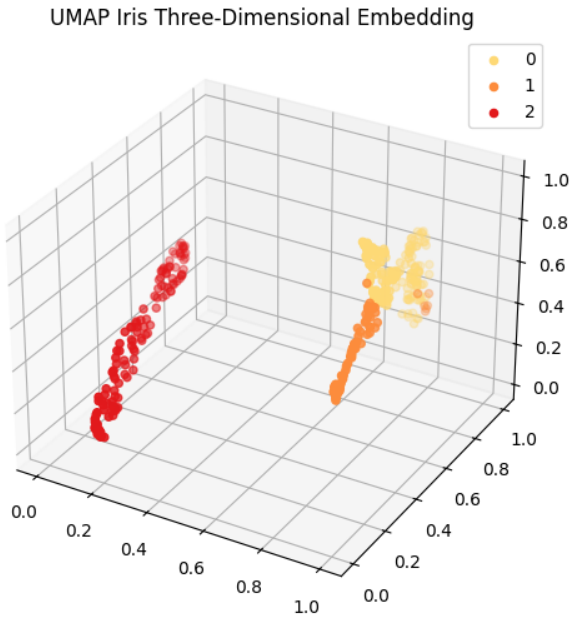


Fig. 31: UMAP Three-Dimensional Embedding

2) UMAP Embedding: The two-dimensional embedding of the Iris dataset using Uniform Manifold Approximation and Projection (Fig. 30) shows that the three species of iris flowers are well-separated in this space, and they show a distinct shape. Although each attempt to reduce the data to this space produced different outcomes, the shape that each species showed was preserved.

3) Autoencoder Embedding: We first reduced the dimensionality, attempting to cluster in the two-dimensional and three-dimensional space. The two-dimensional reduction (Fig. 32) shows that the three species of iris flowers are well-separated in this space, but show no distinct shape.

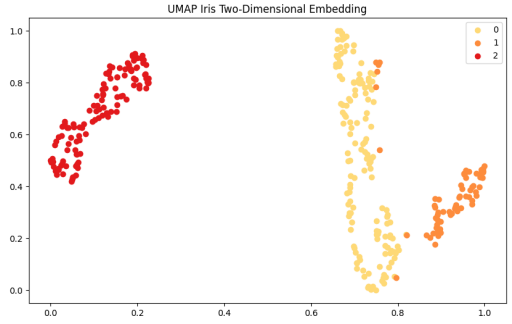


Fig. 30: UMAP Two-Dimensional Embedding

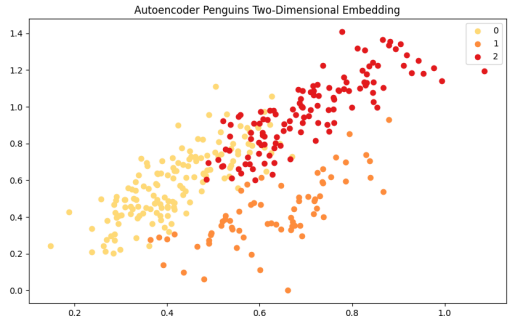


Fig. 32: Autoencoder Two-Dimensional Embedding

Now, for the three-dimensional embedding (Fig. 5), similar results were obtained. The shape was very distinct in every attempt, and all three species were very clearly separated.

Then, the three-dimensional reduction (Fig. 33) shows similar results.

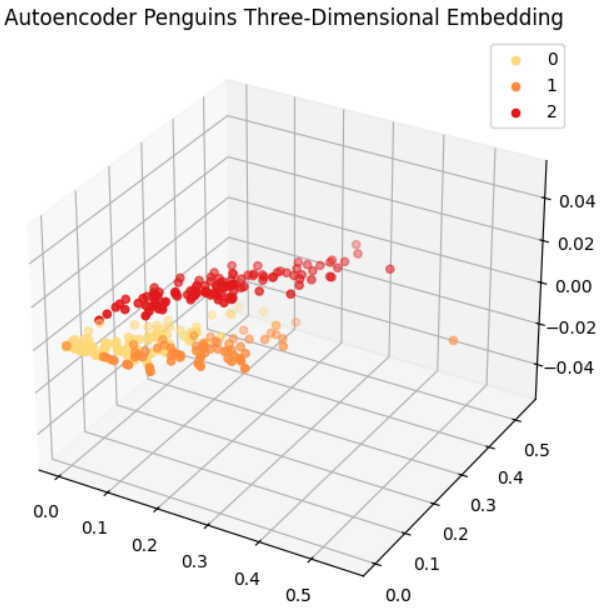


Fig. 33: Autoencoder Three-Dimensional Embedding

We then augmented the variable space to five-dimensional space (Fig. 34). This is were the shapes and behaviours of the species were altered radically. We can particularly note that iris versicolor and iris virginica completely merged in most visualuzations of this space, which complicates clustering.

Fig. 34: Autoencoder Five-Dimensional Embedding

A very curious behavior was observed in three-dimensions and higher, where a certain specie took the shape of a straight line, evidenced in (Fig. 7) with iris setosa, and in (Fig. 8) with the second dimension.

C. *Visualizing the space with different metrics*

In this section, we will explore the Iris Normalized space using different metrics. The distance matrix is a square matrix that contains the distances between all pairs of points in a dataset. The x-axis and y-axis of the plot represent the indices of the points in the dataset, and the z-axis represents the distances between the points.

Fig. 35 shows a 3D surface plot of a Manhattan distance matrix (Manhattan distance $= \sum_{i=1}^{n} |a_i - b_i|$) The plot shows that the points in the dataset are arranged in a grid-like pattern. The points in the top and bottom edges of the grid are closer together than the points on the right and left edges of the grid. This suggests that the points in this area of the grid are more similar to each other than the points on the other edges of the grid.

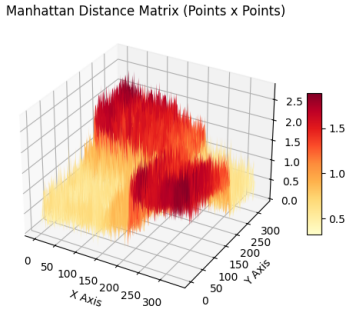


Fig. 35: Iris Normalized Data Viewed by Manhattan Distances Point to Point

The plot also shows that the distances between the points are not uniform. There are some regions of the plot where the points are closer together than in other regions. This suggests that there are some clusters of points in the dataset.

Fig. 36 shows a 3D surface plot of a Euclidean distance matrix (Euclidean distance $= \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$).

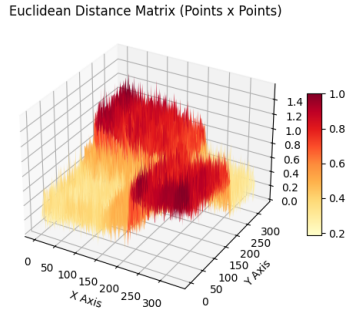


Fig. 36: Iris Normalized Data Viewed by Euclidean Distances Point to Point

Fig. 37 shows a 3D surface plot of a Minkowski distance matrix with $p = \infty$ (Minkowski distance with $p = \infty = \max_{i=1}^{n} |a_i - b_i|$).

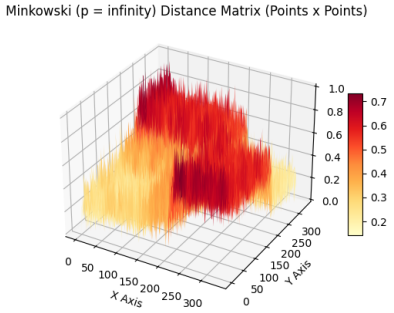


Fig. 37: Iris Normalized Data Viewed by Minkowski $p = \infty$ Distances Point to Point

All three Minkowski Distances show very similar results, with $p = \infty$ showing the most peaks present.

Fig. 38 shows a 3D surface plot of a Cosine distance matrix (Cosine distance $(= 1 - \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \cdot \|\mathbf{b}\|})$. The cosine distance matrix is a measure of similarity between two vectors, and it ranges from 0 (identical vectors) to 1 (orthogonal vectors).

The x-axis and y-axis of the plot represent the distances between the two points, and the z-axis represents the cosine distance between the two points. The higher the z-axis value, the more similar the two points are.

The plot shows that the cosine distance between the two points is highest (closest to 0) when the two points are close to each other, and it decreases as the two points move further apart. This is to be expected, as vectors that are close to each other are more likely to be similar than vectors that are far apart.

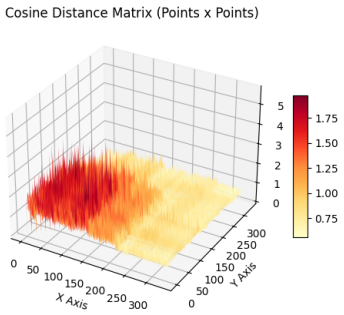
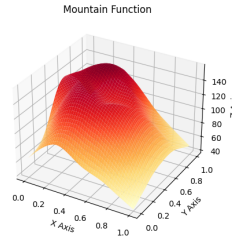


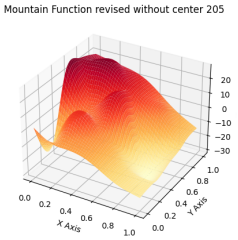
Fig. 38: Iris Normalized Data Viewed by Cosine Distances Point to Point

D. Mountain Function Surfaces

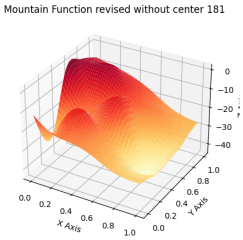
Fig. 39 shows three different surfaces of the mountain function, each at a different step in the optimization process. The mountain function is a mathematical function that has a peak at its center and gradually decreases towards the edges. The goal of the optimization process is to find the parameters of the mountain function that best fit a given dataset.



(a) Step 1



(b) Step 2



(c) Step 3

Fig. 39: Mountain Function Surfaces

Fig. 39a: Step 1

This plot shows the mountain function without a center. The function is symmetrical around the origin, and it has a peak at (0, 0). However, the function is also quite rough, and it has many small bumps and valleys.

Fig. 39b: Step 2

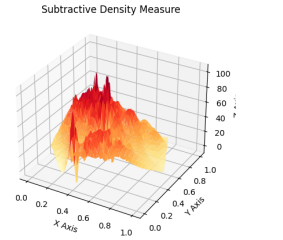
This plot shows the mountain function after the first step of the optimization process. The center of the mountain function has now been found, and the function is smoother than it was in the previous plot. However, the peak of the function is not yet at its maximum height.

Fig. 39c): Step 3

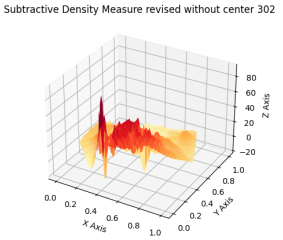
This plot shows the mountain function after the second step of the optimization process. The function is now even smoother than it was in the previous plot, and the peak of the function is closer to its maximum height.

E. Subtractive Density Function Surfaces

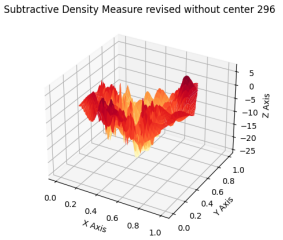
Fig. 40 shows the results of a subtractive density measure at three different steps. In the first step, the subtractive density measure is shown without a center. In the second step, the subtractive density measure is revised without a center. In the third step, the subtractive density measure is revised without a center and with a different scale.



(a) Step 1



(b) Step 2



(c) Step 3

Fig. 40: Subtractive Density Surfaces

F. Visualizing Results

We will use radar plots to show the results of clustering our data using five different clustering algorithms: Mountain Clustering, Subtractive Clustering, K-Means, Fuzzy C-Means, and Spectral Clustering. The radar plot has five axes, each representing a different clustering evaluation metric: Silhouette, Davies-Bouldin, Adjusted Rand Index, Normalized Mutual Information, and Fowlkes Mallows.

Our experiment consists of first using Mountain and Subtractive Clustering to determine the k number of clusters naturally present in the data. Then, we run the other three algorithms using that k as a hyper-parameter.

We run these experiments 100 times for each of our preprocessed data sets: normalized data, UMAP two-dimensionally embedded data, UMAP three-dimensionally embedded data, Autoencoder two-dimensionally embedded data, Autoencoder three-dimensionally embedded data, and Autoencoder five-dimensionally embedded data. We obtain the average of all the results per index and show it in radar plots per dataset.

The radar plot in Fig. 21 shows that Fuzzy K Means outperformed the other four algorithms on four evaluation metrics, except for Davies Bouldin, where Mountain was the best with an almost-perfect score. This suggests that they were able to find the most natural clusters in the normalized iris data. Spectral clustering was the worst method for most scores.

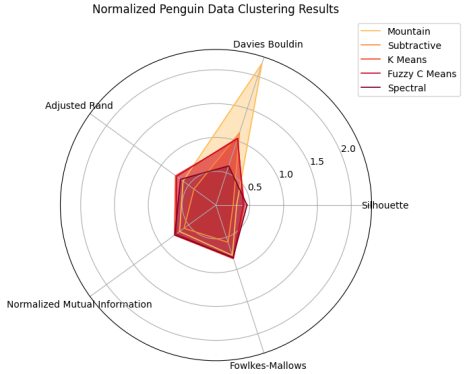


Fig. 41: Normalized Penguin Data Scores

The radar chart depicted in Fig. 42 illustrates that Mountain clustering exhibited superior performance compared to the other four algorithms across all evaluation metrics, excluding Davies Bouldin, where Mountain achieved a much better score. This implies that Mountain clustering successfully identified the most cohesive clusters within the UMAP two-dimensionally embedded data. It is, however, worth noting that all algorithms performed similarly good for this data.

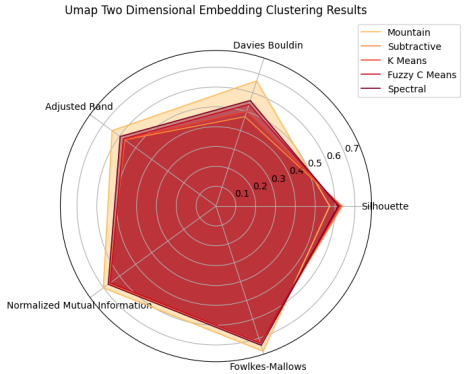


Fig. 42: UMAP Two Dimensional Embedding Scores

The radar plot shown in Fig. 43 indicates that Fuzzy C Means clustering demonstrated great performance along with K Means. Mountain and subtractive did the worst, except for Mountain in Davied Bouldin

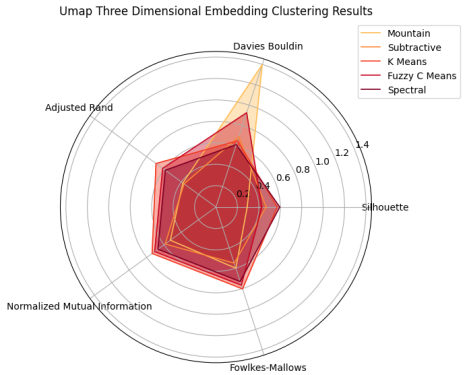


Fig. 43: UMAP Three Dimensional Embedding Scores

The radar plot shown in Fig. 44 indicates that Mountain and subtractive clustering demonstrated better performance than the others. Remarkably, Spectral clustering consistently ranked as the least effective method across different scores by a great amount, except for davies-bouldin and Fowlkes Mallows, where it was average.

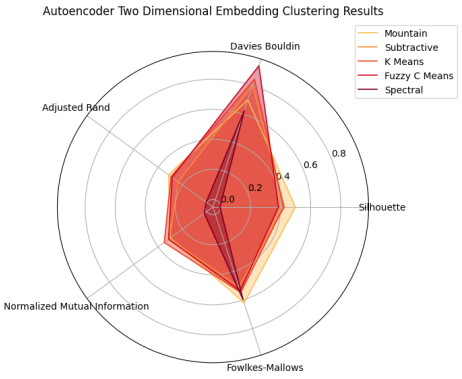


Fig. 44: Autoencoder Two Dimensional Embedding Scores

The radar plot shown in Fig. 45 shows no clear results. There is not one algorithm that ranked best in all or most scores. They were all very evenly matched.

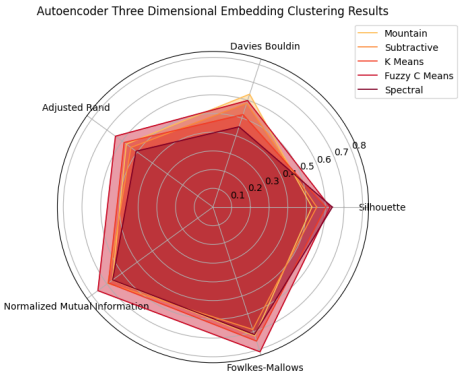


Fig. 45: Autoencoder Three Dimensional Embedding Scores

The radar plot shown in Fig. 46 shows some dominance from Fuzzy C Means and K Means.

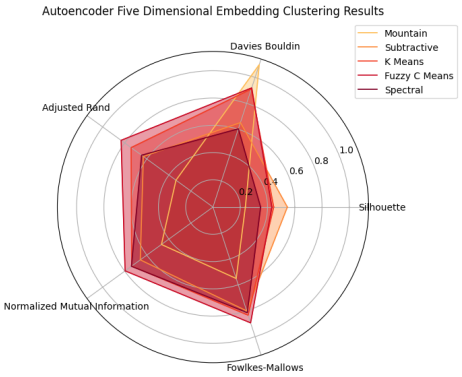


Fig. 46: Autoencoder Five Dimensional Embedding Scores

V. Discussion Iris

The results of the clustering experiments on the Iris dataset using different algorithms and preprocessing techniques provide valuable insights into the performance and suitability of each method.

A. Hyperparameter Tuning

In the hyperparameter tuning tests, each clustering algorithm was evaluated with various parameter combinations to identify the optimal settings. The results indicate that for Mountain Clustering, Subtractive Clustering, K-Means, Fuzzy C-Means, and Spectral Clustering, specific parameter values yielded better clustering performance. These optimal parameter values, such as $\sigma = 0.5$ and $\beta = 0.625$ for Mountain Clustering, $r_\alpha = 0.5$ and $r_\beta = 0.75$ for Subtractive Clustering, and $\varepsilon = 0.45$ for K-Means, can guide practitioners in selecting appropriate settings for these algorithms on the Iris dataset.

B. Clustering Algorithm Performance

The evaluation metrics, including Silhouette Score, Davies-Bouldin Index, Calinski-Harabasz Score, Adjusted Rand Index, Normalized Mutual Information, and Fowlkes-Mallows Index, were used to assess the performance of each clustering algorithm on different versions of the Iris dataset.

In the normalized Iris data, Subtractive Clustering consistently outperformed other algorithms, demonstrating higher Silhouette Scores, lower Davies-Bouldin Index, and superior performance in Adjusted Rand Index, Normalized Mutual Information, and Fowlkes-Mallows Index. Spectral Clustering, on the other hand, tended to perform less effectively in this scenario.

For UMAP two-dimensional and three-dimensional embeddings, Spectral Clustering showed remarkable performance, particularly in UMAP 2D embeddings. Fuzzy C-Means also exhibited strong performance in UMAP 3D embeddings. These results suggest that the choice of preprocessing technique can significantly impact the effectiveness of certain clustering algorithms.

The Autoencoder embeddings presented mixed results. While Fuzzy C-Means performed well in both two-dimensional and three-dimensional Autoencoder embeddings, no single algorithm consistently outperformed others across all metrics. This variability in performance highlights the sensitivity of clustering algorithms to the dimensionality reduction technique applied.

C. Radar Plots Analysis

The radar plots offer a comprehensive visual representation of the clustering algorithms' performance across multiple metrics. In the normalized data, Subtractive Clustering consistently demonstrated superior performance, emphasizing its robustness in capturing natural clusters in the Iris dataset.

UMAP embeddings revealed different strengths in clustering algorithms, with Spectral Clustering excelling in UMAP 2D embeddings and Fuzzy C-Means performing well in UMAP 3D embeddings. The Autoencoder embeddings, on the other

hand, showcased more mixed results, indicating that the choice of dimensionality reduction may influence algorithm performance differently.

VI. Discussion Penguins

A. Clustering Algorithm Performance

The clustering experiments on the Penguin dataset revealed intriguing insights into algorithm behavior under different preprocessing techniques. Subtractive Clustering again demonstrated robustness, showing competitive performance across various metrics.

UMAP embeddings showcased distinct strengths in different clustering algorithms. Spectral Clustering excelled in UMAP 2D embeddings, while Fuzzy C-Means performed well in UMAP 3D embeddings. Autoencoder embeddings presented more mixed results, indicating that the choice of dimensionality reduction may influence algorithm performance differently for the Penguin dataset.

B. Radar Plots Analysis

Radar plots for the Penguin dataset suggested some dominance from Fuzzy C Means and K Means, particularly in Autoencoder Five-Dimensional Embedding. However, unlike the Iris dataset, no single algorithm consistently outperformed

VII. Conclusion

In conclusion, the clustering experiments on the Iris and Penguin datasets underscore the importance of considering both the choice of clustering algorithm and the preprocessing technique employed. Subtractive Clustering, particularly when applied to normalized data, consistently exhibited strong performance across various metrics. However, the choice of algorithm also depends on the nature of the data and the specific goals of the analysis.

The hyperparameter tuning process identified optimal parameter values for each algorithm, providing practitioners with valuable guidance for configuring these algorithms on the Iris dataset.

The radar plots served as a useful visualization tool, offering a concise overview of each algorithm's performance across multiple metrics. The results highlight the nuanced relationship between clustering algorithms, preprocessing techniques, and evaluation metrics.

Ultimately, the findings contribute to a better understanding of how different clustering algorithms behave on the Iris dataset under various conditions. These insights can inform future applications of clustering techniques in similar contexts, guiding practitioners in selecting the most appropriate approach based on their specific requirements and data characteristics.

References

[1] Moertini, V. (2002). Introduction to Five DataClustering Algorithms Clustering Algorithm. Integral, 7(2).

[2] Sorzano, C. O. S., Vargas, J., Montano, A. P. (2014). A survey of dimensionality reduction techniques. arXiv preprint arXiv:1403.2877.

[3] Rokach, L., Maimon, O. (2005). Clustering methods. Data mining and knowledge discovery handbook, 321-352.

[4] Bank, D., Koenigstein, N., Giryes, R. (2023). Autoencoders. Machine Learning for Data Science Handbook: Data Mining and Knowledge Discovery Handbook, 353-374.

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

[6] Shahapure, K. R., Nicholas, C. (2020, October). Cluster quality analysis using silhouette score. In 2020 IEEE 7th international conference on data science and advanced analytics (DSAA) (pp. 747-748). IEEE.