Introduction to Machine Learning: Work 3

Pedro Agúndez*, Bruno Sánchez*, María del Carmen Ramírez*, Antonio Lobo*

December 8, 2024

Abstract

Abstract

*Universitat de Barcelona
pedro.agundez@estudiantat.upc.edu
bruno.sanchez.gomez@estudiantat.upc.edu
maria.del.carmen.ramirez@estudiantat.upc.edu
antonio.lobo@estudiantat.upc.edu

1 Introduction

Introduction.

2 Methodology

Introduction to methodology.

2.1 Datasets

2.2 Ordering Points To Identify the Clustering Structure (OPTICS)

The OPTICS (Ordering Points To Identify the Clustering Structure) method is a density-based clustering algorithm designed to reveal the intrinsic structure of data without requiring explicit cluster assignments or fixed parameter settings. Instead of directly generating clusters, OPTICS produces an ordered list of data points annotated with metrics that reflect their density relationships, such as core distances and reachability distances. This ordering captures the clustering structure across a range of density levels, allowing hierarchical and arbitrary-shaped clusters to be identified. The algorithm is computationally efficient, with performance similar to DBSCAN, and excels in uncovering the natural distribution of complex datasets. For this work, the OPTICS method was implemented using the **sklearn** library, which provides a robust and efficient implementation for clustering tasks.

A key component of OPTICS is the choice of a distance metric, which defines how proximity between points is calculated. This metric determines the spatial relationships that underpin density-based clustering. Three different metrics were chosen for the three datasets: the Euclidean distance, Manhattan distance, and Chebyshev distance.

Moreover, three different methods were computed for the neighborhood queries, a fundamental operation in density-based clustering:

- **Brute Force:** This method computes distances between all point pairs, ensuring compatibility with all datasets but with a computational complexity of $O(n^2)$, making it less efficient for large datasets.
- **Ball Tree:** A hierarchical structure is built, partitioning data into spherical regions, which allows faster neighborhood searches for datasets with low to moderate dimensionality.
- **KD Tree:** Similar to ball trees, KD trees partition data hierarchically but with axis-aligned splits, making them particularly effective for moderately low-dimensional spaces.

Finally, several parameters in OPTICS were adjusted based on the dataset, reflecting the inherent variability between datasets. These parameters show a crucial influence on cluster definition and extraction.

- xi: This parameter influences the steepness of changes in the reachability plot used to delineate clusters. Smaller values allow for finer differentiation between clusters, whereas larger values merge clusters with more gradual density transitions.
- min_cluster_size: Sets the minimum number of points required for a group to be considered a cluster. This prevents very small, potentially noisy groups from being identified as clusters.
- min_samples: Defines the minimum number of points required to qualify as a core point, impacting the density threshold for cluster membership. Higher values favor dense and well-defined clusters, excluding sparser areas.

These parameters work in unison to balance sensitivity to noise, cluster granularity, and robustness, enabling OPTICS to adapt to diverse datasets and clustering tasks.

2.3 Spectral Clustering

Spectral Clustering is a graph-based clustering technique that leverages the spectral properties of the similarity matrix to group data points. It begins by constructing a similarity graph from the input data, where nodes represent data points, and edges encode pairwise similarities based on a given affinity function. The

graph Laplacian matrix, derived from the similarity graph, captures the structure of the data. By solving an eigenvalue problem on the Laplacian matrix, the algorithm embeds the data into a lower-dimensional space where traditional clustering methods, such as k-means, can be applied to partition the data into distinct clusters. This approach is particularly effective for non-linearly separable data or data with complex cluster shapes.

In this work, the *SpectralClustering* algorithm was implemented from the **scikit-learn** library due to its robust and efficient approach to graph-based clustering.

In all the scenarios studied, the nearest neighbors affinity method was implemented, with different values for the parameter n_neighbors. It specifies the number of closest data points considered to build the similarity graph, so it significantly affects the graph's connectivity and structure. For smaller datasets (Hepatitis), smaller values of n_neighbors were selected, whereas for the bigger datasets (Mushroom and Pen-based) larger values were chosen.

Eigen solvers are computational methods used to compute the eigenvectors and eigenvalues of the graph Laplacian, a key step in spectral clustering. The choice of solver affects the computational efficiency and scalability of the algorithm. To obtain a better analysis, the following three eigen solvers were studied for all the datasets:

- **lobpcg**: This solver is efficient for large-scale problems and works well with sparse matrices. It uses the Locally Optimal Block Preconditioned Conjugate Gradient method, making it suitable for high-dimensional datasets.
- **amg**: The Algebraic Multigrid solver is another option for large-scale problems. It is particularly effective for problems with a well-structured Laplacian matrix and leverages multilevel techniques for computational efficiency.
- **arpack**: This is the default solver and works well for small to medium-sized datasets. It employs iterative methods to compute a few eigenvalues and eigenvectors, offering a balance between accuracy and computational cost.

In the final step of spectral clustering, a clustering method is used to group points in the low-dimensional embedding obtained from the eigen decomposition. The methods analyzed for the three datasets are detailed below:

- k-means: This is the default method and groups points by minimizing the sum of squared distances within clusters. It is effective for well-separated clusters and computationally efficient for most applications.
- cluster_qr: This method uses QR decomposition to cluster points, providing a numerically stable alternative to `k-means`. It can be advantageous for datasets with unique cluster structures or when **k-means** struggles to converge.

2.4 K-Means

K-Means intro.

2.4.1 Hyperparameters

1. **k:**
    - The number of clusters to partition the dataset. Determines the complexity and granularity of the clustering.
2. **Distance Metrics:**
    - **Euclidean Distance:** Calculates the root of the sum of squared differences between feature values. Standard metric for continuous data:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- **Manhattan Distance:** Computes the sum of absolute differences between feature values, suitable for both categorical and continuous data:

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

- **Clark Distance:** Accounts for proportional differences between feature values, enhancing interpretability for attributes with varying scales:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} \left(\frac{|x_i - y_i|}{x_i + y_i + \epsilon} \right)^2}$$

where ϵ is a small constant to avoid division by zero.

3. **Additional Parameters:**

- **Initial Centroids:** Pre-defined initial cluster centers used as the starting point for the clustering algorithm.
- **Maximum Iterations:** Limits the number of iterations to prevent excessive computational time, with a default of 10 iterations.

2.4.2 Clustering Methodology

- **Clustering Process:**

1. Assign each data point to the nearest centroid using the specified distance metric.
2. Recalculate centroids by computing the mean of all points in each cluster.
3. Repeat assignment and recalculation until convergence or maximum iterations are reached.

- **Convergence Criteria:**

  – Clusters are considered stable when centroids no longer significantly change between iterations.

- **Variance Computation:**

  – Total within-cluster variance (E) is calculated by summing squared distances of points to their respective cluster centroids.

  – Provides a measure of clustering compactness and quality.

This methodology allows for flexible clustering configurations, enabling analysis across different datasets and hyperparameter values.

2.5 Global K-Means

Out of the proposed improvements to the K-Means algorithm, the first we chose to implement was the **Global K-Means** algorithm [3], which focuses on following a deterministic and systematic approach to "optimal" centroid initialization and cluster formation. Additionally, we have also implemented the improvements to the Global K-Means algorithm itself, proposed in the original article by Likas et al.: *Fast Global K-Means*, and *Initialization with k-d Trees*. By addressing the limitations of traditional K-Means, this enhanced methodology introduces novel strategies, including PCA-based data partitioning and iterative error-reduction mechanisms, to improve both accuracy and computational efficiency.

This section outlines the hyperparameter configurations and clustering methodology adopted for the Global K-Means algorithm, which was implemented in the `GlobalKMeansAlgorithm` class.

2.5.1 Hyperparameters

We consider the same hyperparameters as for the standard K-Means algorithm (Section 2.4), except for 2 significant modifications:

1. **Initial Centroids:**
   - Global K-Means no longer accepts a collection of initial centroids as a hyperparameter, since the goal of this algorithm is rooted in the deterministic calculation of the "best possible" centroids, which substitutes their random initialization.
2. **Number of Buckets:**
   - Controls initial data partitioning, by defining the number of candidate points that we will consider as possible centroids throughout the algorithm.
   - Its default value is $2 \cdot k$, but we also test values $3 \cdot k$ and $4 \cdot k$.

2.5.2 Clustering Methodology

- **Initialization with k-d Trees:**
  1. Use k-d tree partitioning based on Principal Component Analysis (PCA).
  2. Recursively partition data samples into buckets.
  3. Select candidate points based on bucket centroids.
- **Fast Global K-Means Algorithm:**
  1. Initialize first centroid as dataset mean.
  2. Iteratively add centroids by:
     - For each $k' = 2, \ldots, k$, we already have $k' - 1$ centroids.
     - Compute guaranteed error reduction for candidate points with respect to the $k' - 1$ centroids,

$$b_n = \sum_{j=1}^{N} \max\left(d^j_{k'-1} - ||x_n - x_j||^2, 0\right) \ ,$$

       where $d^j_{k'-1}$ is the squared distance between x_j and the closest centroid among the $k' - 1$ obtained so far. The pair-wise squared distances between points are precomputed at the start.
     - Select point with maximum guaranteed error reduction.
     - Run k'-means with the $k' - 1$ centroids plus the selected point, unitl convergence.
  3. Repeat until k clusters are formed.

This methodology provides a sophisticated approach to centroid initialization and clustering, leveraging PCA-based partitioning and error reduction strategies in order to achieve an improvement in consistency and speed with respect to the base K-Means algorithm.

2.6 X-Means Algorithm

The X-Means algorithm extends the traditional K-Means clustering algorithm by addressing its main limitations: the need to predefine the number of clusters K and its susceptibility to local minima. This algorithm dynamically determines the optimal number of clusters K using the Bayesian Information Criterion (BIC).

2.6.1 Core Steps of X-Means Algorithm

The algorithm alternates between two main operations until a stopping criterion is met:

1. **Improve-Parameters:** Perform standard K-Means clustering to optimize the centroid locations for a fixed number of clusters.
2. **Improve-Structure:** Dynamically decide where to split centroids to better fit the data by evaluating each potential split using BIC.

Splitting Process

- During the Improve-Structure step, a local K-Means with $K = 2$ is run until convergence for each of the clusters individually (without the rest of the clusters).
- After convergence, the model selection criterion (BIC score) determines whether the original parent centroid or the newly created children better represent the data. Only the better-performing structure is retained.

Stopping Criteria The algorithm stops when the maximum allowable number of clusters $K_{\max}$ is reached, or when no further splits improve the model selection score.

Model Selection Using BIC The Bayesian Information Criterion (BIC) for a model M_j is defined as:

$$\mathrm{BIC}(M_j) = L_j(D) - \frac{p_j}{2} \log R,$$

where:

- $L_j(D)$: Log-likelihood of the data D under model M_j,
- p_j: Number of free parameters in M_j,
- R: Total number of data points.

The BIC score is calculated globally to select the best model across all iterations and locally to decide the viability of centroid splits.

2.7 Fuzzy C-Means

We selected the **generalized suppressed Fuzzy C-Means** (gs-FCM) algorithm, an improvement over traditional FCM, which often shows multimodal behavior near cluster boundaries (Fig. 1a). This issue, where fuzzy memberships remain high for unrelated clusters, was addressed by Höppner and Klawonn [2].

The suppressed Fuzzy C-Means (s-FCM) algorithm [1] enhances convergence speed and classification accuracy without minimizing the traditional objective function J_{FCM}. It introduces a suppression step during each iteration to reduce non-winner fuzzy memberships, which is mathematically equivalent to virtually reducing the distance to the winning cluster's prototype (Fig. 1b) [5].

Szilágyi et al. [5] defined the quasi-learning rate η of s-FCM, analogous to learning rates in competitive algorithms:

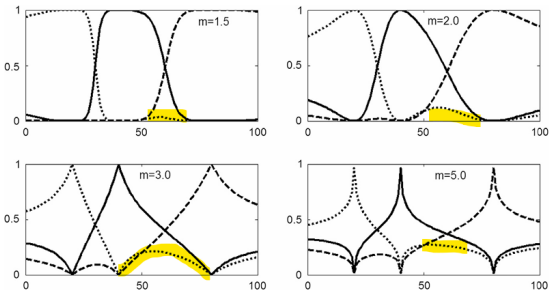
$$\eta(m, \alpha, u_{wk}) = 1 - \frac{\delta_{wk}}{d_{wk}} = 1 - \left(1 + \frac{1-\alpha}{\alpha u_{wk}}\right)^{(1-m)/2}.$$

Building on this, gs-FCM modifies the learning rate to decay linearly with increasing winner membership u_{wk} for faster convergence, as proposed in sg_ρ-FCM [4]:

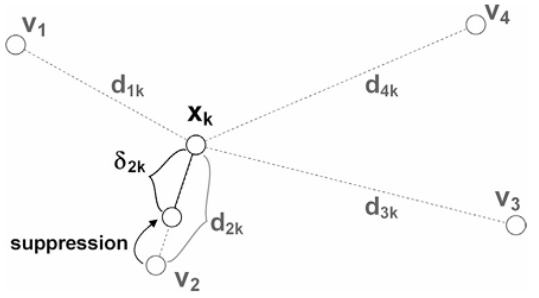
$$\eta(u_{wk}) = 1 - \rho u_{wk}, \quad \text{where } 0 \le \rho \le 1.$$

This approach ensures a logical adaptation of membership weighting, expressed as:

$$\alpha_k = \left[1 - u_w + u_w \left(1 - f(u_w)\right)^{2/(1-m)}\right]^{-(1-m)}.$$



(a) Multimodal fuzzy memberships near cluster boundaries for varying fuzzy exponent m.



(b) Suppression effect: Winner cluster ($w_k = 2$) gains increased membership while non-winners are suppressed.

Figure 1: Figures adapted from [4].

2.8 Metrics

The following metrics have been selected to evaluate clustering performance. Each metric provides a different perspective on how well the clusters reflect underlying data structure, measuring cluster separation, cohesion, or agreement with known labels.

2.8.1 Adjusted Rand Index (ARI)

The Adjusted Rand Index measures the agreement between two partitions (e.g., a clustering and ground truth labels) while correcting for random assignments. Given a pair of partitions, the ARI can be expressed as:

$$\text{ARI} = \frac{R - \mathbb{E}[R]}{\max(R) - \mathbb{E}[R]},$$

where R is the Rand Index. The ARI ranges from -1 to 1, with higher values indicating stronger agreement.

2.8.2 Normalized Mutual Information (NMI)

Normalized Mutual Information measures how much information is shared between two partitions, normalized to ensure values between 0 and 1. Let U and V be two cluster sets:

$$\text{NMI}(U, V) = \frac{2I(U; V)}{H(U) + H(V)},$$

where $I(U; V)$ is the mutual information, and $H(\cdot)$ denotes entropy. Higher values indicate greater similarity.

2.8.3 Davies-Bouldin Index (DBI)

The Davies-Bouldin Index evaluates clustering quality by comparing each cluster's dispersion to its separation from other clusters. For K clusters:

$$\text{DBI} = \frac{1}{K} \sum_{i=1}^{K} \max_{j \neq i} \frac{s_i + s_j}{d_{ij}},$$

where s_i is the average intra-cluster distance and d_{ij} is the distance between cluster centers. Lower values imply more distinct and compact clusters.

2.8.4 Silhouette Score

The Silhouette Score measures how similar each point is to points within its own cluster compared to points in other clusters. For a point x_i:

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)},$$

where a_i is the average intra-cluster distance and b_i is the smallest average distance to another cluster. The final Silhouette Score is the average s_i over all points, with values near 1 indicating well-formed clusters.

2.8.5 Calinski-Harabasz Score (CHS)

The Calinski-Harabasz Score, or Variance Ratio Criterion, compares the variance between clusters with the variance within clusters. For N data points clustered into K clusters:

$$\text{CHS} = \frac{\text{Tr}(B_K)}{\text{Tr}(W_K)} \cdot \frac{N-K}{K-1},$$

where B_K and W_K are the between- and within-cluster scatter matrices. Higher CHS values suggest better-defined clusters.

3 Results

To systematically evaluate the different configurations of each clustering algorithm, the following procedure is followed to extract results for each of the 3 data sets, in order to later perform an analysis those results:

1. **Data Preparation**: The dataset is loaded and the data samples are separated from their labels into separate 2 sets. This way, we perform the clustering analysis in a completely non-supervised way, and we then utilize the labels to extract supervised metrics of the cultering results.

2. **Parameter Configuration**: A comprehensive set of values for the algorithm's hyperparameters is defined. These combinations reflect various ways to tune the clustering algorithm.

3. **Model Evaluation**: For each parameter combination, the clustering algorithm is applied on the unlabeled data and then evaluated with different metrics. This step yields the following metrics: Adjusted Rand Score (ARI), Normalized Mutual Information (NMI), Davies-Bouldin Index(DBI), Silhouette score, Calinski-Harabasz Score (CHS), and execution time. Together, these metrics (the first 2 supervised, and the rest non-supervised) measure the effectiveness and efficiency of the clustering.

4. **Results Compilation**: The performance metrics for each parameter combination are recorded in a structured format. These results are saved as a dataset that summarizes the outcomes of all evaluations, forming a basis for analysis. We save as well the cluster labels of all samples for each clustering algorithm that we run, so we can recover the same clusters in the posterior analysis.

5. **Results Analysis**: After results are compiled across all configurations, quantitative and qualitative analysis is performed to identify common trends among the different algorithm configurations for each of the datasets. Additionally, we extract the top performing configuration according to each of the 5 evaluation metrics, in order to study common patterns and visualize the resulting clusters. This analysis helps determine the most reliable and effective parameter settings for accurate and efficient clustering for each dataset.

Due to the vast ammount of information that can be extracted from the results, we will only explicitly showcase the most clarifying plots and results that we have extracted. However, all of the information is available in the `plots_and_tables` folder for each of the datasets, in the `code` attached to this report.

Note: Since all of the tested datasets have high dimensionality, we use Principal Component Analysis (PCA) to extract the 2 principal components in order to generate the clustering scatter plots of the datasets for visualization purposes.

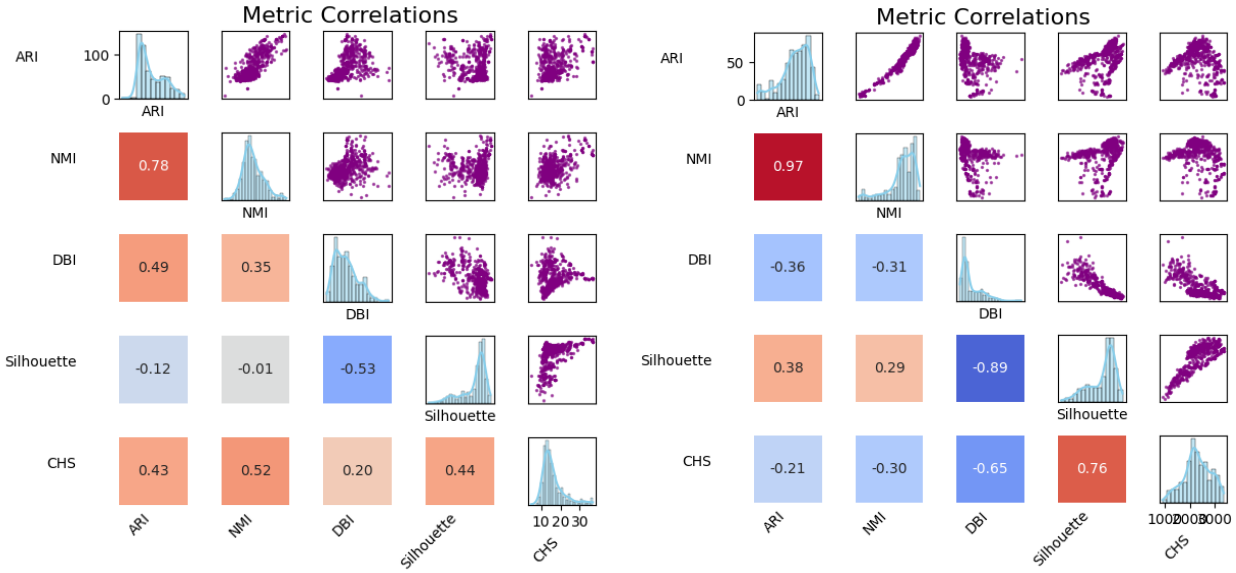
3.1 K-Means

We have tested on each dataset 57 different configurations of the K-Means algorithm, by using the 3 different distance metrics with 19 different values of the k (from 2 to 20). For each of these configurations, we have run the K-Means 10 times, in order to account for the randomness of the centroid initialization. This results in a total of 570 runs of the K-Means algorithm for each dataset. From the evaluation metrics extracted for each of these runs, we study the effect of each of the 2 hyperparameters and study the best performing runs according to each of the metrics.

3.1.1 Hyperparameter Study

First of all, let us start by observing some preliminary patterns about the measured metrics and the effect of each hyperparameter on the clustering performance.

In Figure 2 we summarize the relationship between the different metrics that were measured for two datasets. Each is a matrix plot where the lower triangle is a heatmap of the Pearson correlations between each pair of metrics, the diagonal elements are the histogram distributions of values of each metric, and the upper triangular has for each pair of metrics the plot of their values for all runs. It is interesting to observe that, while we would expect all of the metrics to agree on the identified trends, there are some cases where the opposite behavior is displayed in each dataset. We observe that most of the strongest correlations are preserved (NMI-ARI, Silhouette-DBI, CHS-Silhouette), but others are completely reversed (DBI-ARI, CHS-NMI). This could be due to changes in the hyperparameters which interact differently with the metrics, depending on the nature of the dataset.

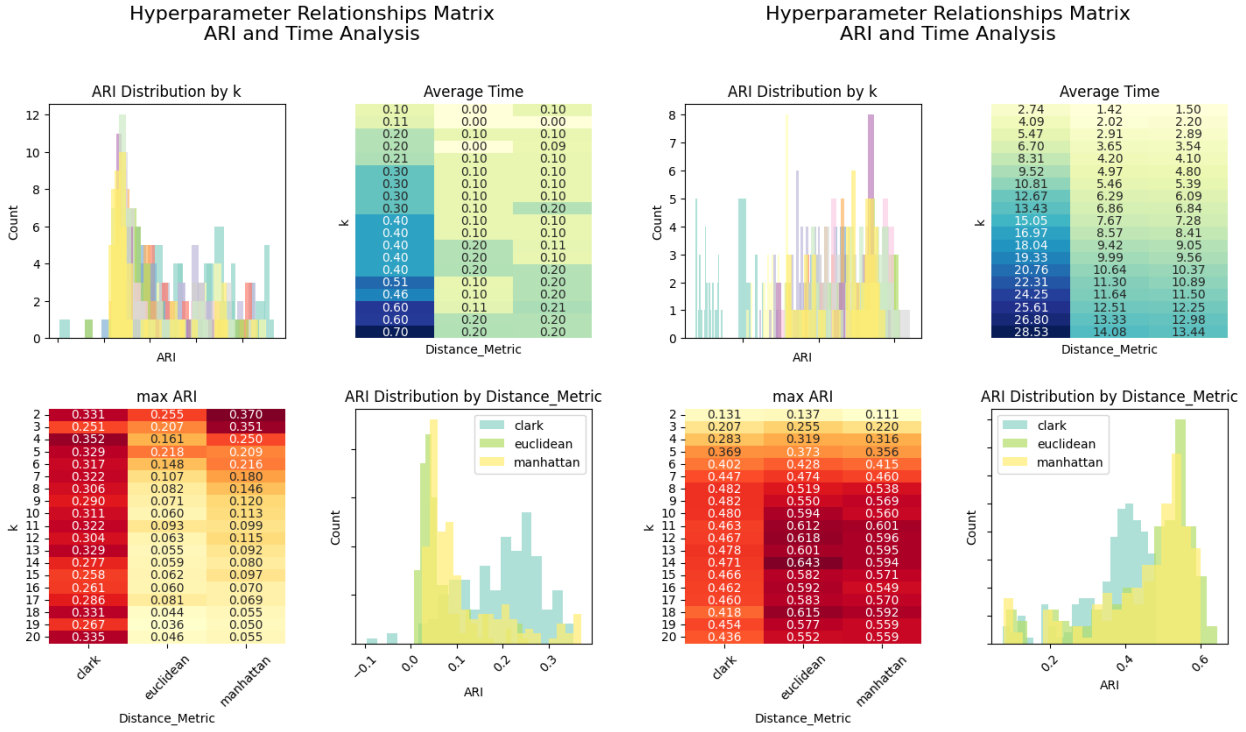


(a) Hepatitis metric correlations

(b) Pen-based metric correlations

Figure 2: Metric correlations and distributions in two datasets

Parallelly, a different set of interesting relationship are displayed in Figure 3, where we can see heatmaps of the ARI and the Time across the different pairwise hyperparameter configurations. We can observe a general trend regarding execution time: it seems to have considerably larger values for the Clark distance metric compared to those of the other 2, which reflects the higher computational cost that this distance metric has. Additionally, we see a noteworthy divergence in the ARI trends: in the Hepatitis dataset (which has 2 classes), lower values of k seem to achieve a better ARI, with a significantly better performance with the Clark distance metric than with the other two; meanwhile, in the Pen-based dataset (which has 10 classes), bigger values of k (>7) seem to achieve the best scores, with a somewhat lower performance with the Clark distance than with the other two. The behavior with respect to the k was to be expected due to the true number of labels of each dataset, yet it still is compelling to see it reflected so clearly in the results. On the other hand, it is enlightening to see opposite behaviors with respect to the distance metric, which reflects that the Clark distance fails to capture some intrinsic properties of the Pen-based dataset, while it excells to do so within the Hepatitis dataset.

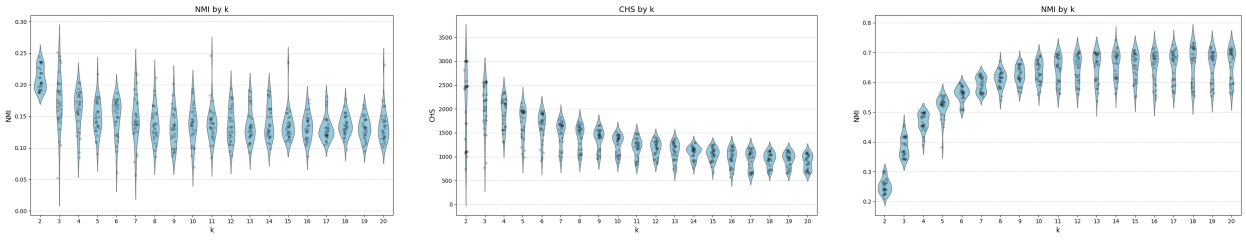


(a) Hepatitis pairplot matrix

(b) Pen-based pairplot matrix

Figure 3: Hyperparameter pairplot matrices based on F1 Score and Time

We can clearly observe the trends regarding the value of k in the violin plot in Figure 4. This time, for Hepatitis and Pen-based we plot the NMI metric, which again shows the same behavior: in Hepatitis, we observe a clear maximum of consistency for k=2, and then a constant decrease in performance as k increases; while, in Pen-based, it improves as we increase the k, with a higher consistency between values 8 and 11. As for Mushroom, the overall results are more inconclusive, but in this violin plot of the CHS we can observe that the score gets consistently worse as the k increases, which once again was to be expected due to the true number of classes being 2.



(a) Hepatitis NMI

(b) Mushroom CHS

(c) Pen-based NMI

Figure 4: Violin plots with respect to k for the different datasets

3.1.2 Best Runs

For each of the datasets, we extracted the run which achieved the best score for each of the metrics, which results in 5 K-Means algorithm configurations, which we consider to be the 5 best runs for that dataset (in no particular order). A summary of the best runs for the 3 datasets is displayed in Table 2.

From these results, we can make some deductions as to which are the best hyperparameter configurations of the K-Means for the 3 datasets:

1. **Hepatitis:** It is again clear that low values of k (2 and 3) typically achieve better scores, which we

	Hepatitis			Mushroom			Pen-based		
Metric	**k**	**Distance**	**Value**	**k**	**Distance**	**Value**	**k**	**Distance**	**Value**
ARI	2	manhattan	0.37	2	clark	0.40	14	euclidean	0.64
NMI	3	clark	0.25	15	clark	0.43	14	euclidean	0.74
DBI	20	euclidean	1.40	2	euclidean	1.20	7	euclidean	1.23
Silhouette	2	manhattan	0.21	2	euclidean	0.28	10	euclidean	0.32
CHS	2	euclidean	36.77	2	euclidean	2996.24	4	euclidean	3361.02

Table 1: Metrics with corresponding values, k, and distance metrics for three datasets.

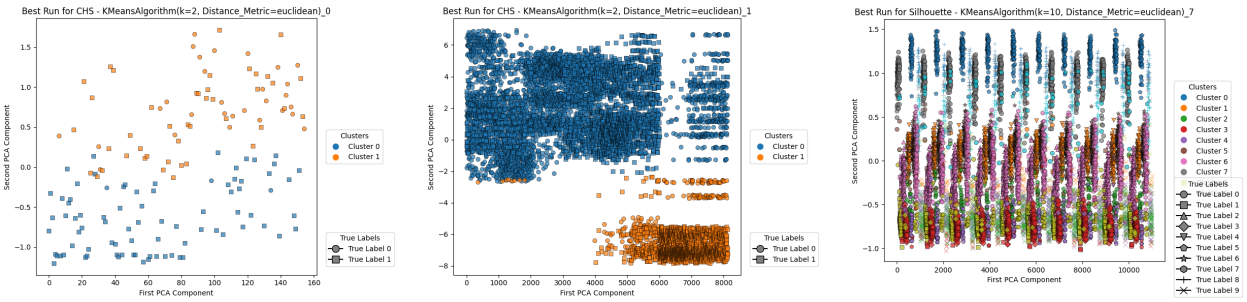
had already observed before. On the other hand, it is not so clear which of the 3 distance metrics is more appropriate for this dataset, since all of them appear in the top scoring runs.

2. **Mushroom:** We can observe once again that k=2 is dominant, probably due to the 2 classes that the dataset has. As for the distance metrics, Euclidean seems to be the most effective, followed by Clark, and Manhattan does not appear to be useful for the properties of this dataset.

3. **Pen-based:** In this case, we do not see such a clear predominance of any specific value of k, but there seems to be a trend towards intermediate values, which was to be expected due to the 10 classes of the dataset. In contrast, we do see a constant primacy of the Euclidean distance metric over the rest, that clearly appears to be better at capturing key differences in this dataset than the other two.

Additionally, we can observe that the 2 bigger datasets (Mushroom and Pen-based) have overall better top values of the metrics than the smaller, Hepatitis dataset. In particular, the Pen-based has the best results in all of the metrics except DBI (in which the difference is minimal), which leads us to the conclusion that it is the dataset for which the K-Means clustering algorithm is better suited.

Finally, we display in Figure 5 the resulting clusters for some of the best runs (one per dataset), after transforming the samples through a PCA with 2 components. Due to the high amount of samples, the Mushroom and Pen-based plots might appear somewhat cluttered, but the clusters can still be identified.



(a) Hepatitis (b) Mushroom (c) Pen-based

Figure 5: Resulting clusters for each dataset (after PCA)

3.2 Global K-Means

We have tested on each dataset 171 different configurations of the Global K-Means algorithm, by using the 3 different distance metrics with 19 different values of the k (from 2 to 20), and 3 different values for the number of buckets (2k, 3k and 4k). This time, we have only run the Global K-Means once for each of these configurations, since it is a deterministic algorithm that would return the same results every time. From the evaluation metrics extracted for each of these runs, we study the effect of each of the 3 hyperparameters and study the best performing runs according to each of the metrics.

3.2.1 Hyperparameter Study

As in the previous section, we start by observing some preliminary patterns about the effect of each hyperparameter on the clustering performance.

In Figure 6 we summarize the observed trends in each of the datasets (columns) for each of the hyperparameters (rows) with different metrics. It is not surprising to see that the patterns regarding the K-Means hyperparameters (k and distance metric) mostly repeat for the Global K-Means, since Global K-Means is mearly a modification of the same idea as the standard K-Means.

- In the first row of plots, we observe that Hepatitis again favors low values of k; Mushroom does not achieve very conclusive results, but definitely gets consistently worse with higher values of k; and Pen-based benefits from intermediate values, getting more inconsistent for the highest ones.
- In the middle row, it is evident that the Clark distance performs significantly better on the Hepatitis dataset and significantly worse on the other 2 datasets, which are similar conclusions as we had reached before.
- In the last row, we observe that the clustering performance generally does not seem to be significantly altered by the different values of the number of buckets that we use to initilize the candidate points of the algorithm.

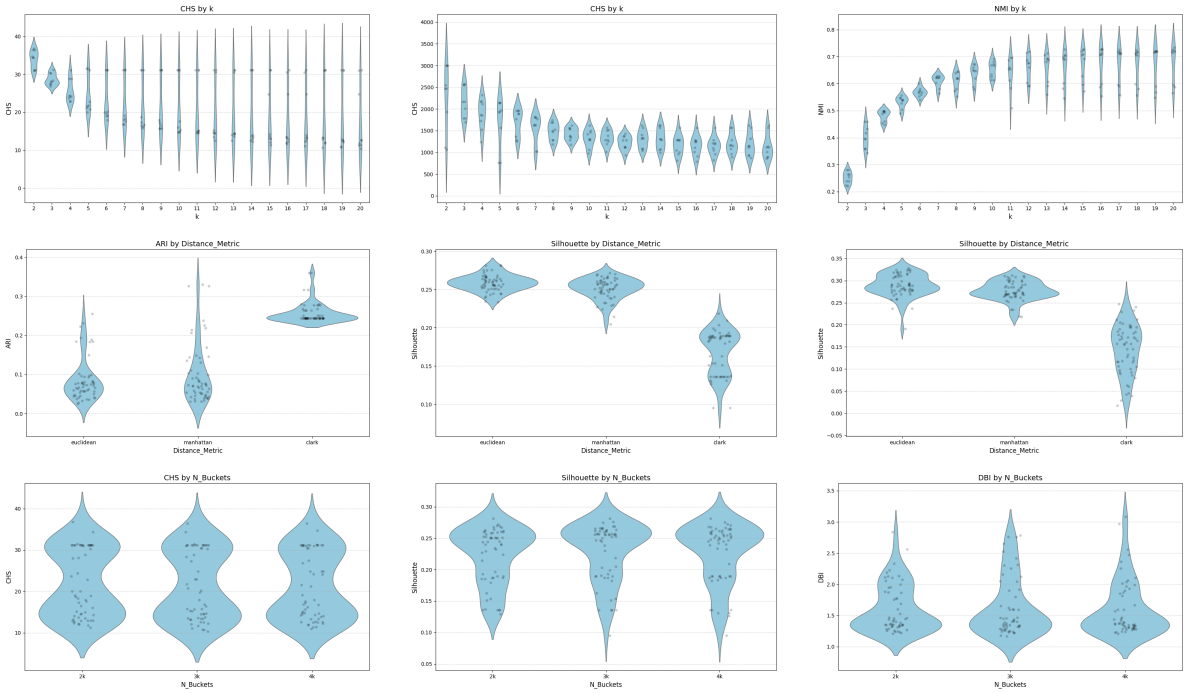


Figure 6: Violin plots of the Global K-Means: Each row is a hyperparameter (k, distance metric, number of buckets), and each column is a dataset (Hepatitis, Mushroom, Pen-based)

3.2.2 Best Runs

As for the K-Means, we extracted for each of the datasets the run which achieved the best score for each of the metrics. A summary of them is displayed in Table 2. For the sake of space efficiency, we shorten the names of the distance metrics to E (Euclidean), M (Manhattan) and C (Clark), and also we denote the number of buckets by N_B.

From these results, we can make some deductions as to which are the best hyperparameter configurations of the Global K-Means for the 3 datasets:

1. **Hepatitis:** Once again, we observe a trend towards low values of k, although this time it is slightly more doubious, with two appearances of k=4. As for the distance metric, the results, as for the K-Means algorithm, are inconclusive. It is clear, however, that the Global K-Means thrives with a lower number of buckets in the Hepatitis dataset, with almost all top configurations having N_B=2k.

	Hepatitis				Mushroom				Pen-based			
Metric	**k**	**Dist.**	**N_B**	**Value**	**k**	**Dist.**	**N_B**	**Value**	**k**	**Dist.**	**N_B**	**Value**
ARI	4	C	2k	0.36	2	C	2k	0.40	14	E	4k	0.62
NMI	4	C	2k	0.25	20	M	3k	0.38	20	E	2k	0.73
DBI	20	E	3k	1.27	2	E	2k	1.20	9	E	3k	1.17
Silhouette	2	M	2k	0.21	2	E	2k	0.28	12	E	3k	0.33
CHS	2	E	2k	36.77	2	E	2k	2996.24	4	E	4k	3357.99

Table 2: Metrics with corresponding values, and hyperparameters for three datasets.

2. **Mushroom:** We can observe once again that k=2 is dominant, as for the K-Means. On the other hand, the Euclidean distance seems to be the most effective, though it is now unclear from these results which of the other two follows (although it was very clear with the violin plots, Figure 6). As for the number of buckets, it again seems that a lower number of them (2k) tends to be the best option for Mushroom.

3. **Pen-based:** It is again unclear which value of k could be the optimal for Pen-based, but the k's seem to be slightly higher this time compared to the results of the K-Means (between 9 and 14, probably). The Euclidean distance metric still remain as the obvious choice for this dataset. The number of buckets this time seems to benefit from having greater values than for the other datasets, having a couple of 3k and 4k appearances in the best runs.

When we compare the metric values achieved by these runs to the best performing runs of the K-Means, we observe that the differences are minimal, being the most consistent improvement in the DBI metric (although the improvement is still very slight). This, along with the fact that all of the observed trends regarding hyperparameters repeat for the 2 algorithms, leads us to believe that the Global K-Means does not represent a significant improvement over the standard K-Means in terms of clustering performance.

However, the Global K-Means does provide a consistency factor that the K-Means lacks. By having a deterministic approach to centroid initilization, the Global K-Means can guarantee to provide results equally good as the best of the K-Means without taking the risk of randomness. We have observed in our experiments that (with the 2 computational improvements proposed in [3]) the 171 configurations of the Global K-Means that we ran for each dataset took significantly less amount of computation time than the 570 runs that we tested for the K-Means.

Thus, our conclusion regarding the Global K-Means is that its use should be recommended over the standard K-Means whenever possible, since it essentially eliminates the risk of randomness at no apparent cost.

Note: Since the resulting configurations are mostly similar to those of the K-Means, we do not display again the resulting clusters. These would be equivalent to those visualized in Figure 5.

3.3 X-Means

3.3.1 Hyper-parameters

We have explored the effect of the main hyper-parameter of the X-Means, which is the maximum number of clusters. The results have varied significantly for the different datasets. The results and plots presented here show the number of clusters predicted by the X-Means algorithm (red points) and their averaged value (blue line) over a range of maximum clusters. Each configuration has been run several times to reduce the effects of the stochastic nature of the algorithm.

1. **Hepatitis:**
   When analyzing the effect of increasing the maximum number of clusters when processing the hepatitis dataset, we can observe that the algorithm converges to 2 clusters and thus the mean number of predicted clusters does not vary significantly when testing in a range from 2 to 10 for the maximum number of clusters parameter, with the exception of a few runs where the algorithm found 3 clusters.

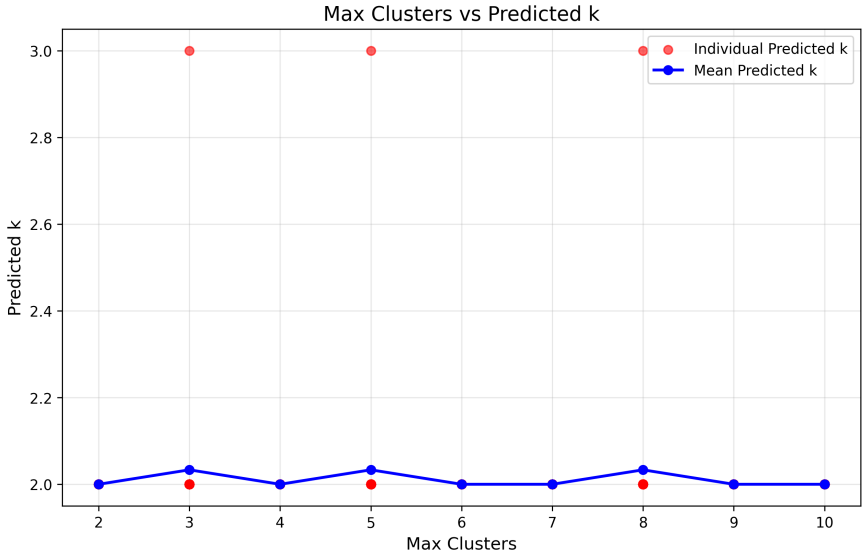


Figure 7: Max number of clusters vs predicted number of clusters (hepatitis dataset)

2. **Mushroom:**
   The results for the mushroom dataset are quite different from the ones obtained in hepatitis. For this experiment each configuration was run 10 times. The plot shows clearly how the runs with a maximum number of clusters lower than 600 always finished with the maximum number of clusters. When the maximum number of clusters was higher than 600 the algorithm would repeatedly converge to a predicted number of clusters lower than the maximum, with this effect being particularly noticeable when plotting the average number of predicted clusters (blue line).

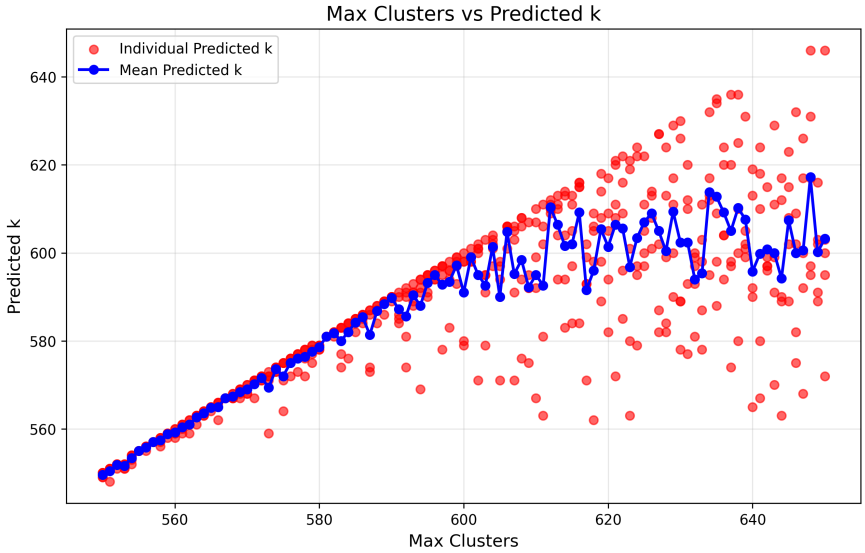


Figure 8: Max number of clusters vs predicted number of clusters (mushroom dataset)

3. **Pen-based:**
   The analysis of the effects of varying the maximum number of clusters when processing the pen-based dataset are similar to the ones obtained in mushroom, with the difference that here the algorithm converges to a lower number of clusters, around 200 (see figure).

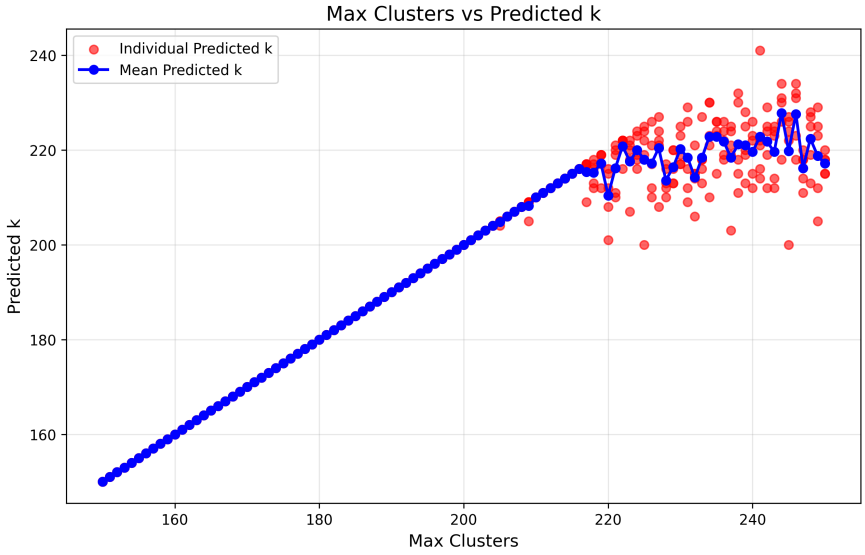


Figure 9: Max number of clusters vs predicted number of clusters (pen-based dataset

3.3.2 Visualization

Since the datasets we have chosen for this assignment have high dimensionality we have opted for doing Principal Component Analysis (PCA) to be able to plot the points and the predicted clusters.

For the hepatitis dataset, the fewer samples and the fact that the X-Means converged to a lower number of clusters allows us to easily plot it using the first two principal components. Both the mushroom and pen-based datasets are easier to explore using an interactive 3D plot.

1. **Hepatitis:**

   Plotting the points in the hepatitis dataset using the first two principal components we can observe how the points have been clustered divided in the middle.

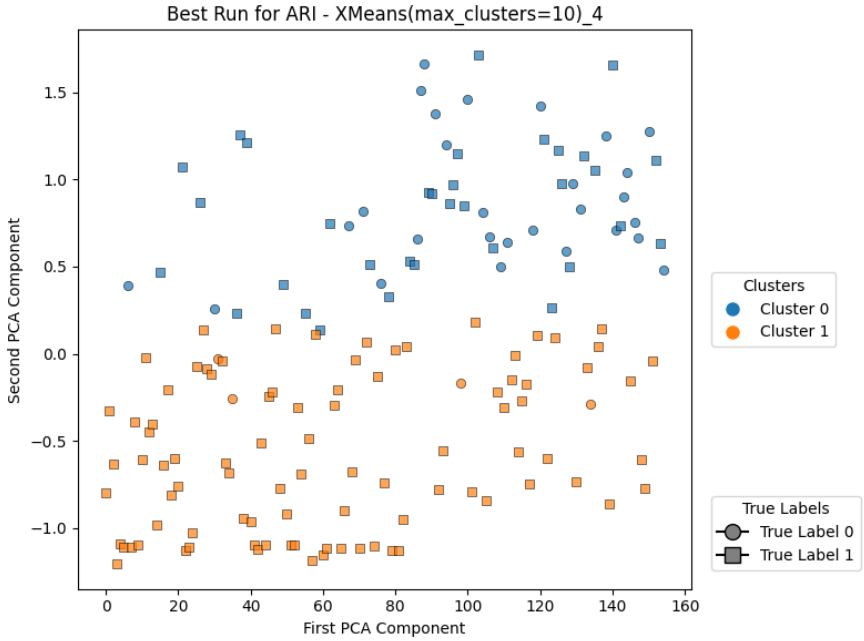


Figure 10: Hepatitis dataset 2D visualization

2. **Mushroom:**

   The plot of the mushroom dataset helps us understand why the X-Means has converged to such a high number of clusters, when seen from the distance the points seem to form few elongated clusters, however, zooming in we can observe how the big clusters are formed by smaller, clearly separated clusters.

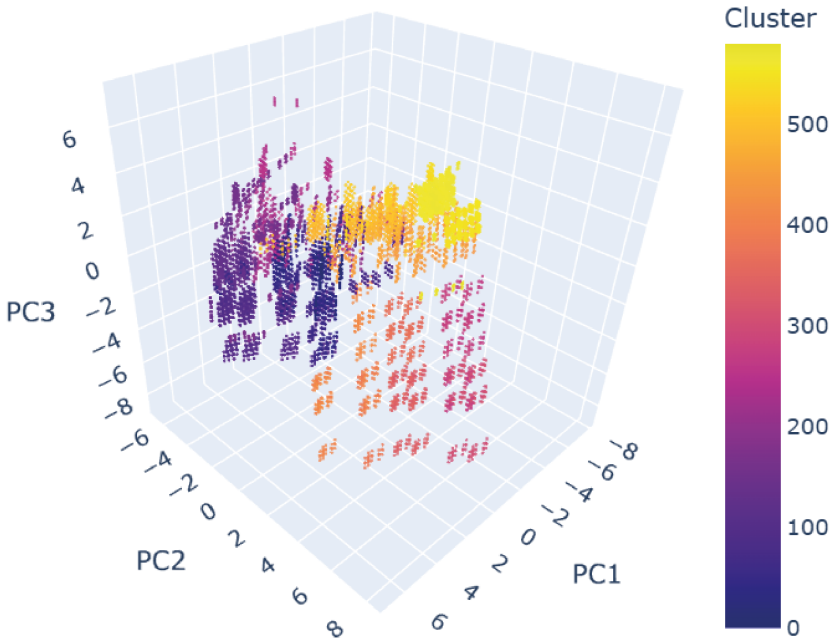


Figure 11: Mushroom dataset 3D visualization

3. **Pen-based**

   Similarly to the visualization of the mushroom dataset, visualizing the pen-based dataset in 3D helps us understand how the algorithm has clustered the data, with big clusters formed by smaller ones.

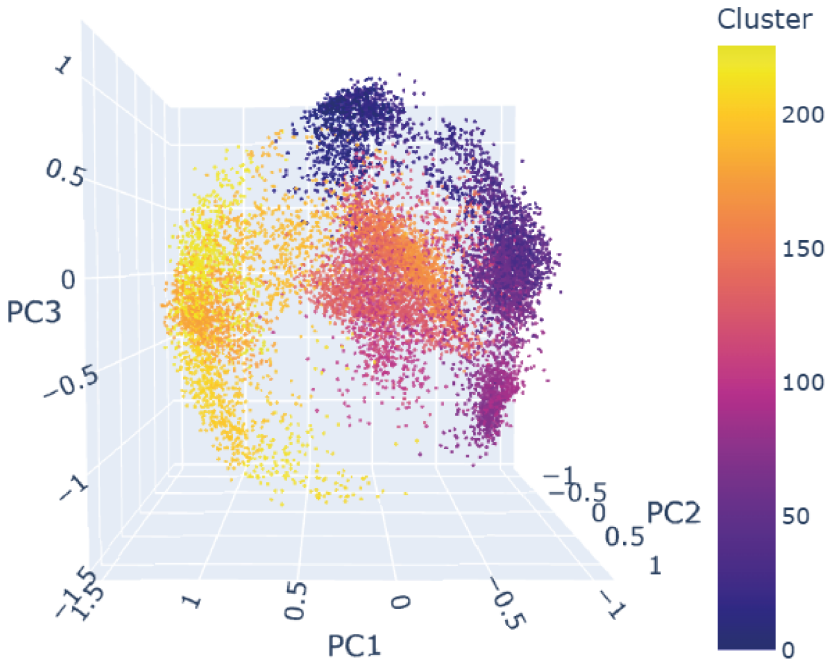


Figure 12: Pen-based dataset 3D visualization

3.4 Fuzzy C-Means

We have initially tested 216 different configurations of the Fuzzy C-Means (FCM) algorithm on each dataset varying the hyperparameters between the following values:

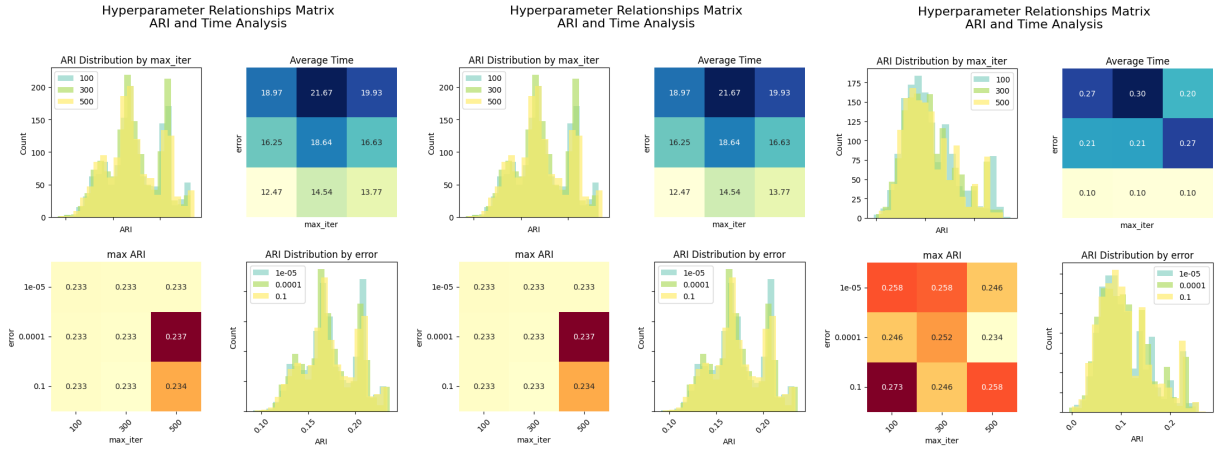
- *n_clusters*: 2, 3, 4, 5, 7, 9, 11, 13, 15
- m (Fuzziness): 1.5, 2, 3, 4, 5, 6, 7, 9
- ρ: 0.5, 0.7, 0.9

We tried the following stop rules:

1. max_iter: 100, 300, 500
2. error: 1e-1, 1e-4, 1e-5

For each configuration, we ran the algorithm 10 times to mitigate the effects of initialization randomness, resulting in a total of 2160 runs of the FCM algorithm per dataset. From the evaluation metrics extracted in these runs, we analyze the impact of the key hyperparameters and derive insights.

We decided not to include *max_iter* and *error tolerance* in the analysis of performance metrics, as they do not significantly affect the performance, aside from removing outliers with really bad performance due to a bad inizialization and slightly improving execution time. To illustrate this, we include 3 heatmaps displaying execution times for each dataset, along with the distribution of ARI results for different values of *max_iter* and *error tolerance* (see **Figure**13).



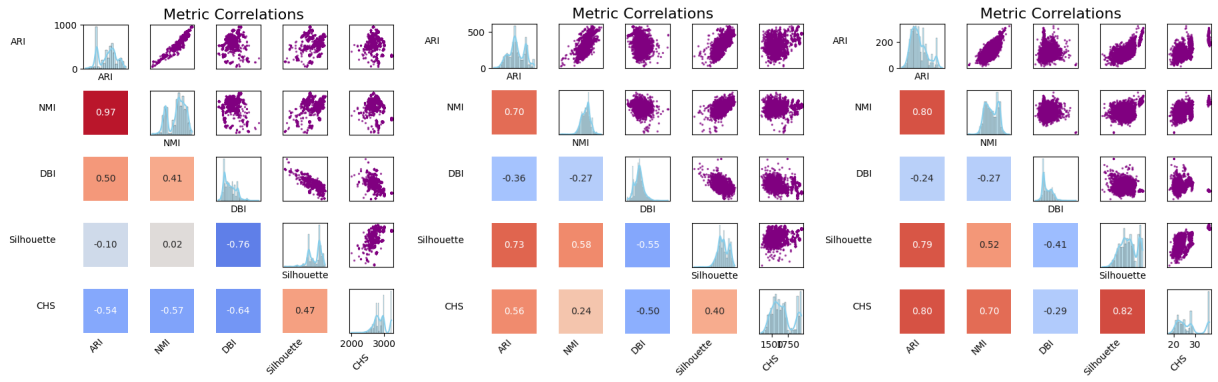
(a) Pen-based dataset. (b) Mushroom dataset. (c) Hepatitis dataset.

Figure 13: Heatmaps illustrating execution times for each dataset, showcasing performance across different configurations.

3.4.1 Preliminary Study

We first explored preliminary patterns in the measured metrics and the influence of hyperparameters on clustering performance.

Figure 14 illustrates the relationships between the various metrics for the FCM algorithm. Similar trends to the ones observed in K-means (see **Figure** 2): (1) **External metrics** (*ARI*, *NMI*) are highly correlated as both improve when the clusters assimilates to the ground truth. (2) Regarding **internal metrics**, *Silhouette* and *CHS* are negatively correlated with *DBI*, probably because the first two metrics measure cluster cohesion and separation positively, while *DBI* assesses the ratio of intra-cluster dispersion to inter-cluster separation, where lower *DBI* values indicate better clustering quality.



(a) Pen-based dataset. (b) Mushroom dataset. (c) Hepatitis dataset.

Figure 14: Metrics correlation acrross the three datasets.

Additionally, Figure 15 illustrates pairwise relationships between hyperparameters, specifically the impact of fuzziness (m), the number of clusters (`n_clusters`), and the generalized parameter (ρ) on clustering quality (evaluated using the maximum value) and execution time (evaluated using the average). The following conclusions can be drawn from these plots:

- **General Observations**:

  1. An increase in computation time is observed with higher values of *n_clusters*, *fuzziness* (m), or ρ. This aligns with the understanding that a higher number of clusters, increased fuzziness, or a lower quasi-learning rate (inversely related to ρ) slows convergence.

2. Variations in ρ do not significantly affect the ARI metric, as ρ is primarily intended to accelerate the method's convergence rather than enhance clustering performance.

- **Dataset-Specific Insights**:

1. For datasets with fewer classes (e.g., Hepatitis and Mushroom), clustering performance improves with a smaller number of clusters, whereas datasets like Pen-Based benefit from higher values of *n_clusters*.
2. Lower fuzziness (m) yields better performance for sparse datasets (e.g., Mushroom and Hepatitis), which have high dimensionality relative to the number of instances. Conversely, higher fuzziness appears advantageous for dense datasets like Pen-Based, where cluster boundaries are likely less distinct.

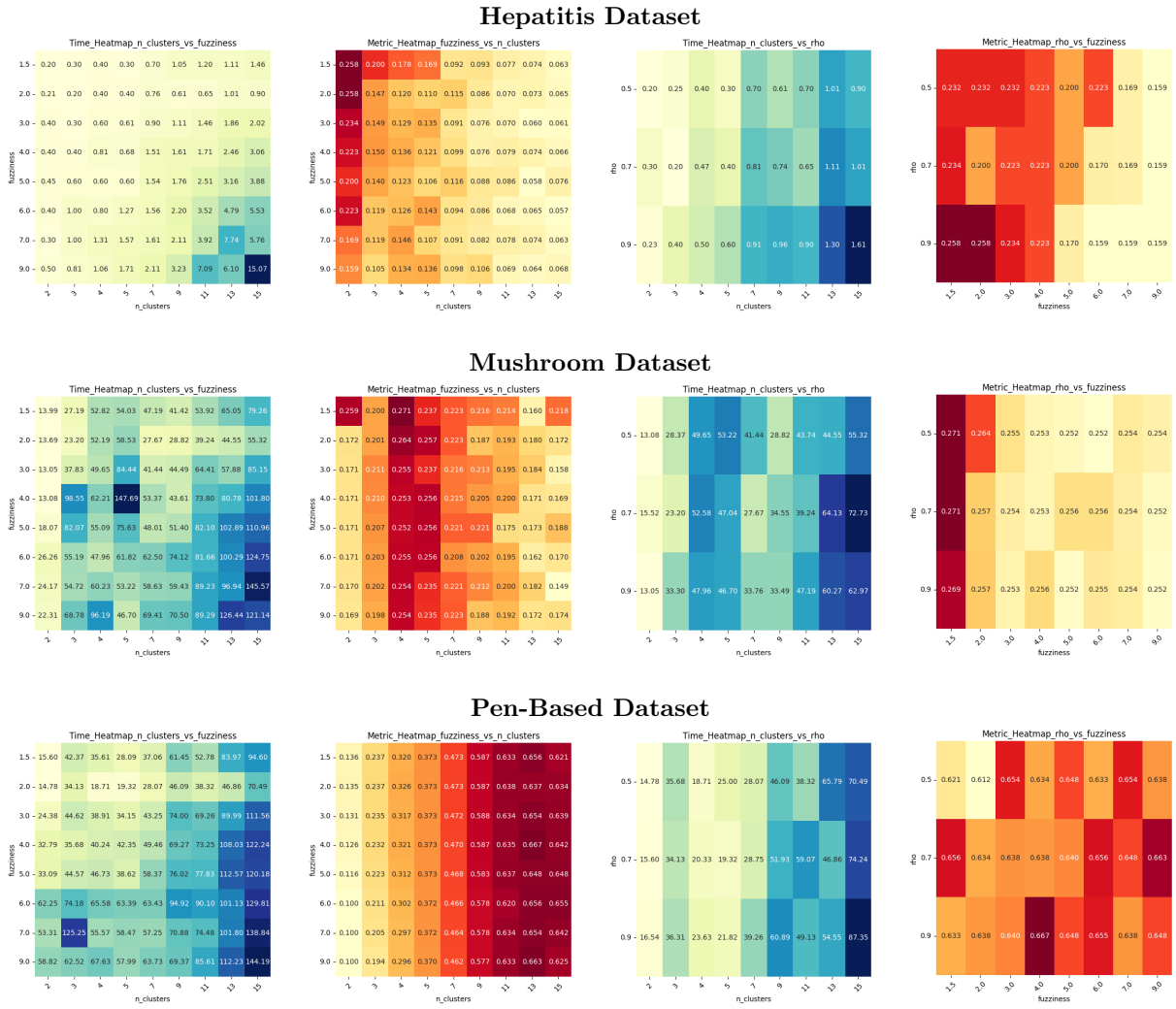


Figure 15: Comparison of heatmaps for Time and Metrics across Hepatitis, Mushroom, and Pen-Based datasets.

3.4.2 Unified Study of Parameters

To provide a comprehensive understanding of the impact of key parameters (m and `n_clusters`) on clustering performance, we analyze violin plots for a selection of metrics across the Mushroom, Pen-Based, and Hepatitis datasets. These plots reveal dataset-specific trends that inform optimal parameter selection.

- **Mushroom Dataset:**

1. Silhouette scores reach their highest value at $m = 3$, indicating that moderately sharp cluster boundaries enhance separability in this dataset.
2. Optimal NMI and CHS values are observed at `n_clusters` = 2 and 4, suggesting that the dataset's inherent structure is best represented by a smaller number of well-defined clusters.

- **Pen-Based Dataset:**

1. CHS achieves peak performance with moderate `n_clusters` values (e.g., 3 and 4), balancing granularity and overfitting. In contrast, NMI increases steadily with higher `n_clusters` values, reflecting alignment with the dataset's true labels.
2. Higher values of m result in reduced DBI, indicating improved intra-cluster cohesion—a critical factor for datasets with overlapping clusters.

- **Hepatitis Dataset:**

1. DBI consistently improves with increasing `n_clusters`, except for an initial degradation from 2 to 3, highlighting the importance of fine-grained clustering to capture the dataset's complexity. NMI peaks at `n_clusters` = 2, aligning with the dataset's labeled structure.
2. Lower m values enhance Silhouette scores, reflecting a preference for sharp cluster boundaries to effectively distinguish small and diverse groups.

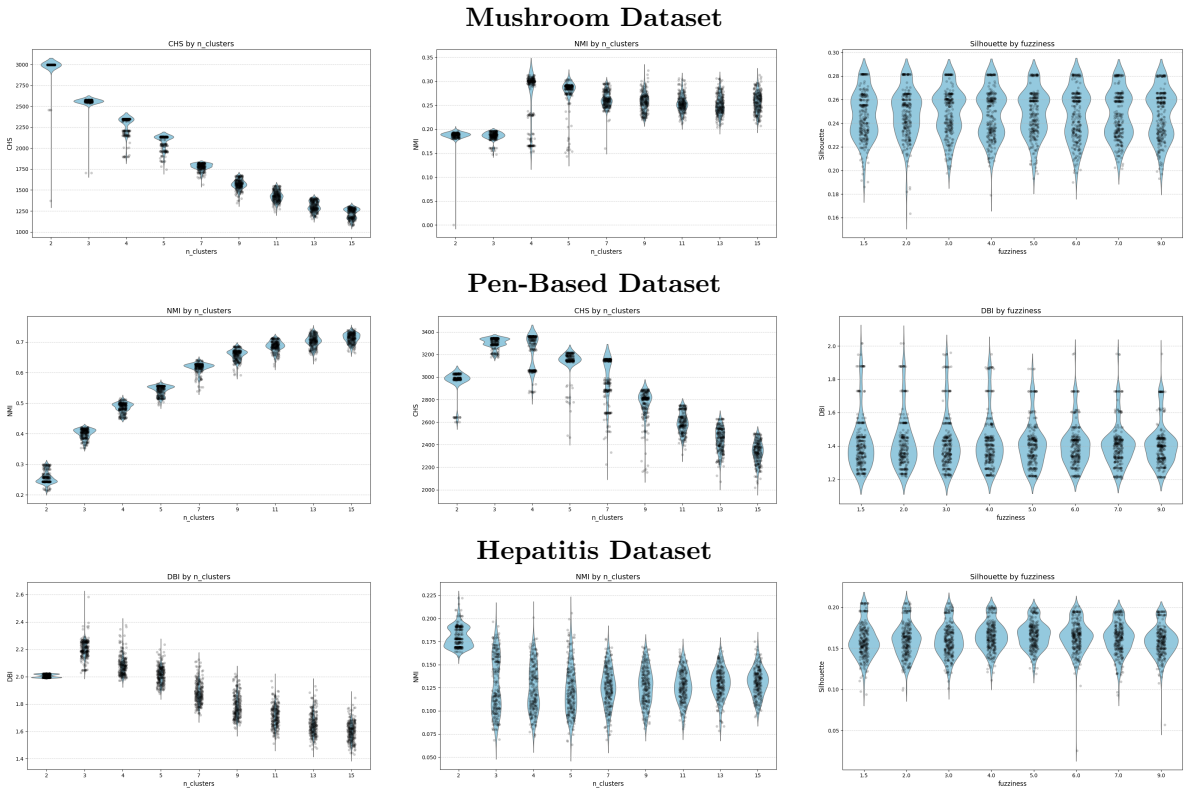


Figure 16: Violin plots illustrating the effect of m (fuzziness) and `n_clusters` on various clustering metrics across the Mushroom, Pen-Based, and Hepatitis datasets. Rows represent datasets, while columns depict specific parameter-metric relationships.

3.4.3 Best Runs

For each dataset, we extracted the configuration that achieved the best score for each metric. This results in five sg-FCM configurations per dataset, which we consider the best runs for that dataset (in no particular order). A summary of the best configurations for the three datasets is shown in Table 3.

	Hepatitis				Mushroom				Pen-Based			
Metric	`clusters`	m	ρ	**Value**	`clusters`	m	ρ	**Value**	`clusters`	m	ρ	**Value**
ARI	2	1.5	0.9	0.2585	4	1.5	0.5	0.2708	13	4.0	0.9	0.6672
NMI	2	1.5	0.9	0.2222	9	1.5	0.7	0.3224	15	6.0	0.9	0.7452
DBI	15	6.0	0.9	1.4303	3	6.0	0.7	1.1127	9	7.0	0.7	1.2040
Silhouette	2	2.0	0.9	0.2066	2	1.5	0.5	0.2816	13	3.0	0.5	0.3290
CHS	2	1.5	0.5	36.77	2	1.5	0.5	2996.24	4	1.5	0.7	3361.04

Table 3: Best configurations and their corresponding parameter values (`n_clusters`, m, ρ) and metric values for sg-FCM across the three datasets.

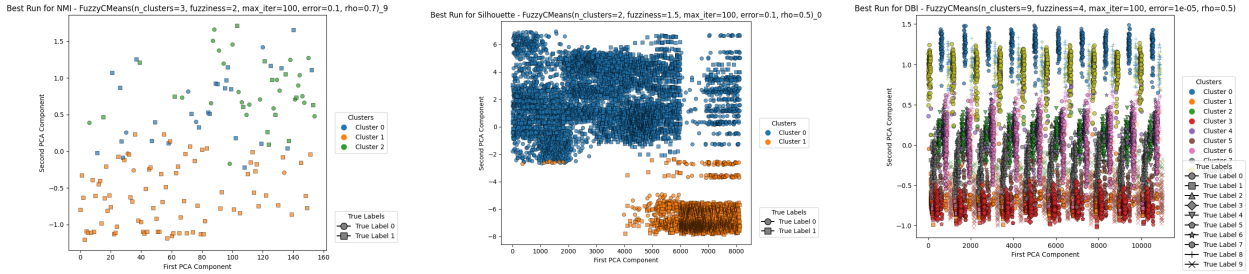
From these results, we can draw several conclusions about the best hyperparameter configurations for sg-FCM:

1. **Hepatitis:** Low values of `n_clusters` (2 and 3) dominate the top configurations, reflecting the dataset's small number of distinct classes. Higher values of m improve DBI, suggesting better intra-cluster cohesion for fuzzier assignments.

2. **Mushroom:** The best configurations favor low fuzziness values ($m = 1.5$), which is expected given the dataset's sparse nature. Lower m values (between 2 and 4) nearer to the ground truth.

3. **Pen-Based:** Intermediate to high values of `n_clusters` (9, 13, and 15) are optimal, consistent with the dataset's real number of classes. Higher values of m improve performance in this dataset as it is more dense than the others.

Additionally, the larger datasets (Mushroom and Pen-Based) achieve better metric values compared to the smaller Hepatitis dataset. Pen-Based performs particularly well across most metrics, reflecting its compatibility with the sg-FCM clustering algorithm.



(a) Hepatitis Dataset (NMI)

(b) Mushroom Dataset (CHS)

(c) Pen-Based Dataset (Silhouette)

Figure 17: Resulting clusters for each dataset using the best sg-FCM configurations (after PCA).

4 Conclusion

Conclusion.

References

[1] J.L. Fan, W.Z. Zhen, and W.X. Xie. Suppressed fuzzy c-means clustering algorithm. Pattern Recognition Letters, 24:1607–1612, 2003.

[2] F. Höppner and F. Klawonn. Improved fuzzy partitions for fuzzy regression models. International Journal of Approximate Reasoning, 32:85–102, 2003.

[3] Aristidis Likas, Nikos Vlassis, and Jakob J. Verbeek. The global k-means clustering algorithm. Pattern Recognition, 36(2):451–461, 2003.

[4] László Szilágyi and Sándor M. Szilágyi. Generalization rules for the suppressed fuzzy c-means clustering algorithm. Neurocomputing, 139:298–309, 2014.

[5] L. Szilágyi, S.M. Szilágyi, and Z. Benyó. Analytical and numerical evaluation of the suppressed fuzzy c-means algorithm: a study on the competition in c-means clustering models. Soft Computing, 14:495–505, 2010.