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.

2.3 Spectral

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 $k' = 2, \dots, k$, 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 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:

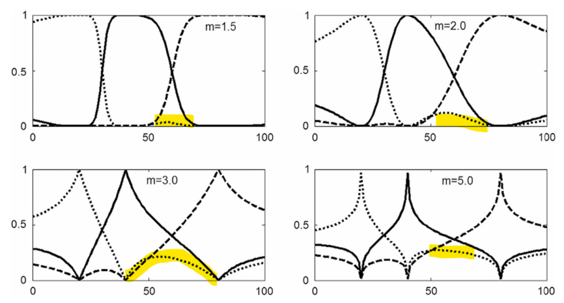
$$\eta(m, \alpha, u_{wk}) = 1 - \frac{\delta_{wk}}{d_{wk}} = 1 - \left(1 + \frac{1-\alpha}{\alpha u_{wk}}\right)^{(1-m)/2} .$$

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]:

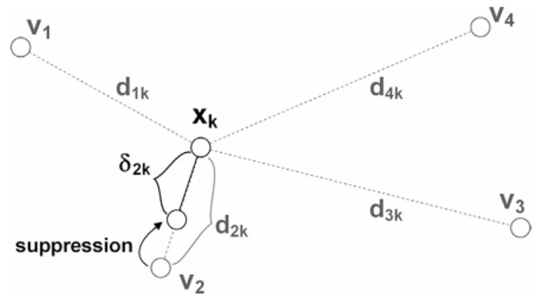
$$\eta(u_{wk}) = 1 - \rho u_{wk}, \quad \text{where } 0 \leq \rho \leq 1.$$

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].

2.7 Metrics

2.7.1 Adjusted Rand Score (ARI)

2.7.2 Normalized Mutual Information (NMI)

2.7.3 Davies-Boulding Index (DBI)

2.7.4 Silhoutte Score

2.7.5 Calinski-Harabasz Score (CHS)

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.

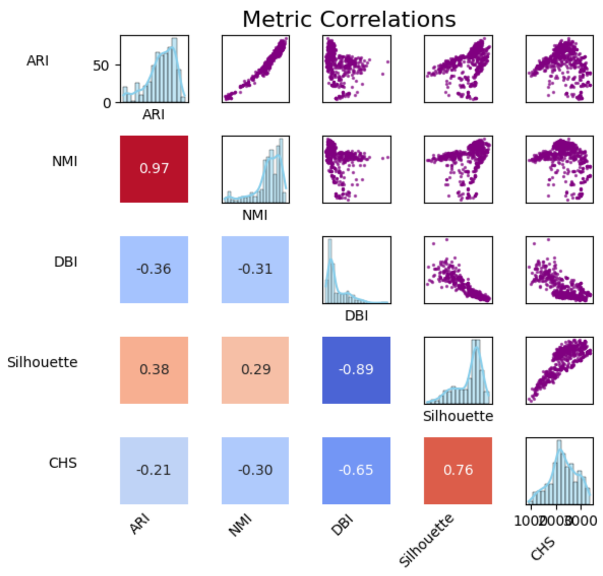
3.1 K-Means

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.

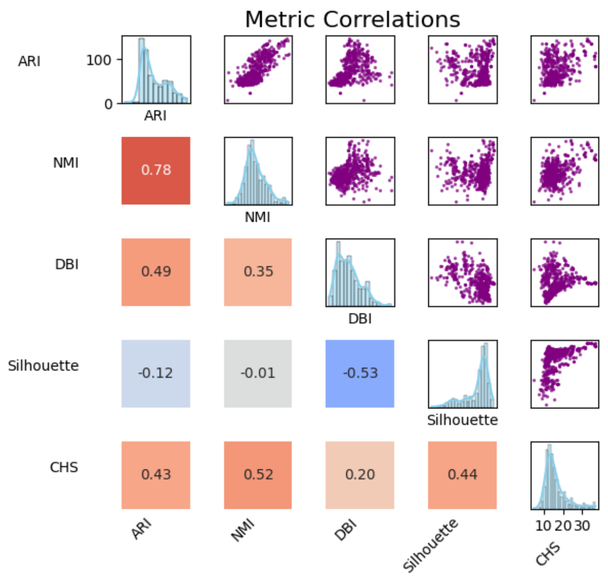
3.1.1 Hyperparameter Study

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.



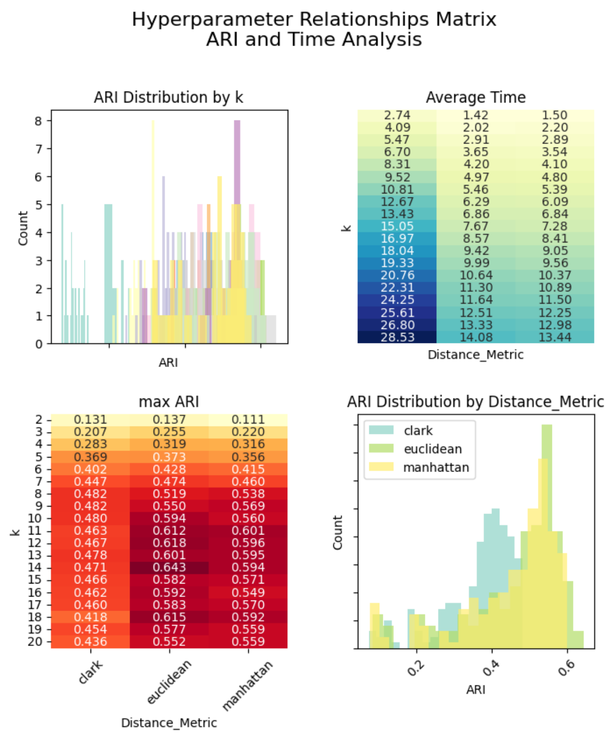
(a) Pen-based metric correlations



(b) Hepatitis 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 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; meanwhile, 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. 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.



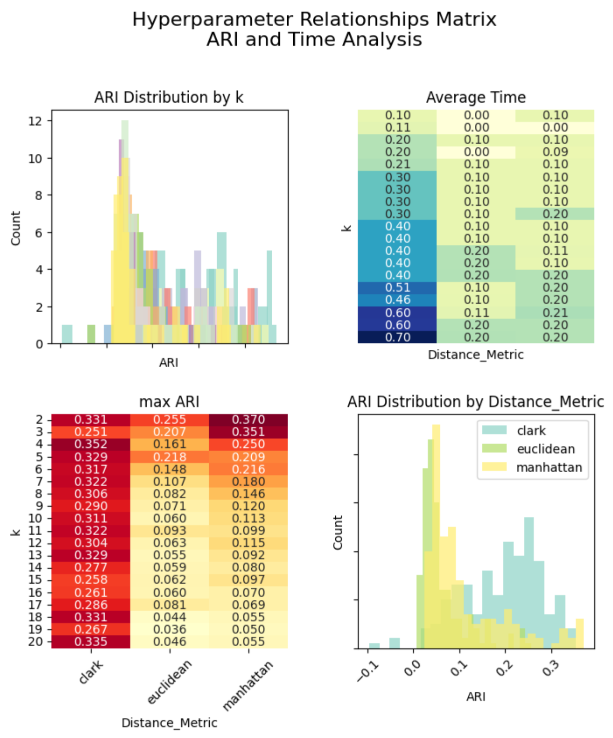
Average Time

2.74	1.42	1.50
4.09	2.02	2.20
5.47	2.91	2.89
6.70	3.65	3.54
8.31	4.20	4.10
9.52	4.97	4.80
10.81	5.46	5.39
12.67	6.29	6.09
13.43	6.86	6.84
15.05	7.67	7.28
16.97	8.57	8.41
18.04	9.42	9.05
19.33	9.99	9.56
20.76	10.64	10.37
22.31	11.30	10.89
24.25	11.64	11.50
25.61	12.51	12.25
26.80	13.33	12.98
28.53	14.08	13.44

max ARI

k	clark	euclidean	manhattan
2	0.131	0.137	0.111
3	0.207	0.255	0.220
4	0.283	0.319	0.316
5	0.369	0.373	0.356
6	0.402	0.428	0.415
7	0.447	0.474	0.460
8	0.482	0.519	0.538
9	0.482	0.550	0.569
10	0.480	0.594	0.560
11	0.463	0.612	0.601
12	0.467	0.618	0.596
13	0.478	0.601	0.595
14	0.471	0.643	0.594
15	0.466	0.582	0.571
16	0.462	0.592	0.549
17	0.460	0.583	0.570
18	0.418	0.615	0.592
19	0.454	0.577	0.559
20	0.436	0.552	0.559

(a) Pen-based pairplot matrix



Average Time

0.10	0.00	0.10
0.11	0.00	0.00
0.20	0.10	0.10
0.20	0.00	0.09
0.21	0.10	0.10
0.30	0.10	0.10
0.30	0.10	0.10
0.30	0.10	0.10
0.30	0.10	0.20
0.40	0.10	0.10
0.40	0.10	0.10
0.40	0.20	0.11
0.40	0.20	0.10
0.40	0.20	0.20
0.51	0.10	0.20
0.46	0.10	0.20
0.60	0.11	0.21
0.60	0.20	0.20
0.70	0.20	0.20

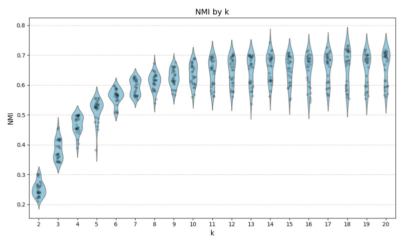
max ARI

k	clark	euclidean	manhattan
2	0.331	0.255	0.370
3	0.251	0.207	0.351
4	0.352	0.161	0.250
5	0.329	0.218	0.209
6	0.317	0.148	0.216
7	0.322	0.107	0.180
8	0.306	0.082	0.146
9	0.290	0.071	0.120
10	0.311	0.060	0.113
11	0.322	0.093	0.099
12	0.304	0.063	0.115
13	0.329	0.055	0.092
14	0.277	0.059	0.080
15	0.258	0.062	0.097
16	0.261	0.060	0.070
17	0.286	0.081	0.069
18	0.331	0.044	0.055
19	0.267	0.036	0.050
20	0.335	0.046	0.055

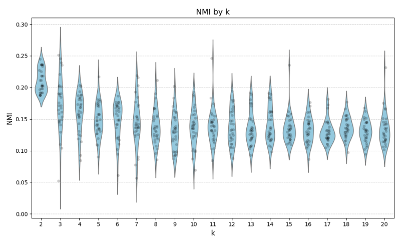
(b) Hepatitis pairplot matrix

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

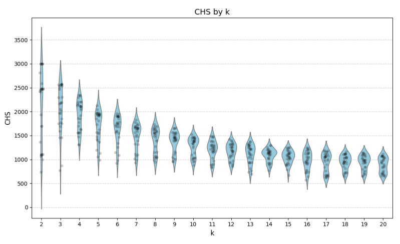
We can clearly observe the trends regarding the value of k in the violin plot in Figure 4. This time, for Pen-based and Hepatitis we plot the NMI metric, which again shows the same behavior: in Pen-based, it improves as we increase the k, with a higher consistency between values 8 and 11; while, in Hepatitis, we observe a clear maximum of consistency for k=2, and then a constant decrease in performance as k increases. 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) Pen-based NMI



(b) Hepatitis NMI



(c) Mushroom CHS

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 1

	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.

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 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.

3.2 Fuzzy C-Means

We have tested 432 different configurations of the Fuzzy C-Means (FCM) algorithm on each dataset by using the fuzziness parameter m (with values from 1.5 to 4.0) and varying the number of clusters (n_clusters) between the following values:

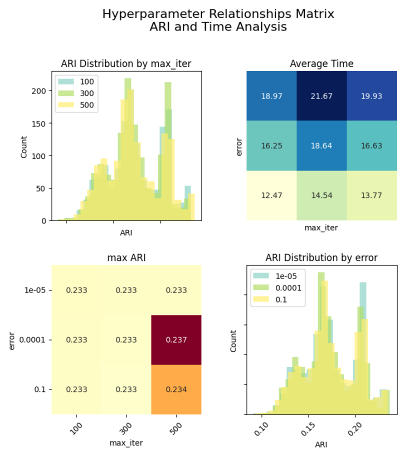
- Pen-based: 6, 8, 9, 10
- Mushroom and Hepatitis: 2, 3, 4, 5

The following parameters were also varied:

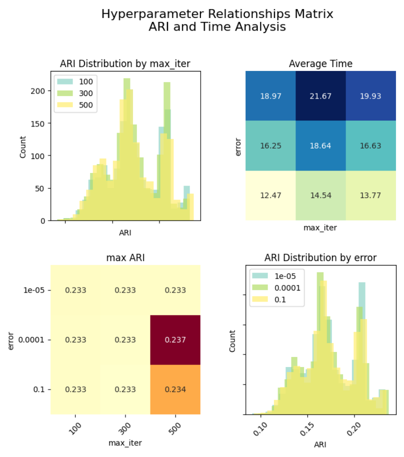
- max_iter: 100, 300, 500
- error: 1e-1, 1e-4, 1e-5
- ρ: 0.5, 0.7, 0.9

For each configuration, we ran the algorithm 10 times to mitigate the effects of initialization randomness, resulting in a total of 4320 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 through statistical analysis.

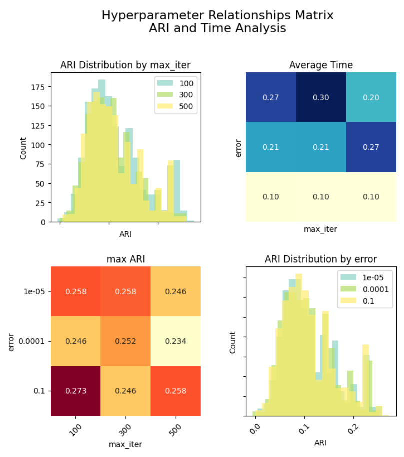
We decided not to include *max_iter* and *error tolerance* in the analysis of performance metrics, as they do not significantly affect the performance, aside from removing outliers and slightly improving execution time. To illustrate this, we include 3 heatmaps displaying execution times for each dataset, along with the distribution of ARI results for different values of *max_iter* and *error tolerance* (see **Figure**5).



(a) Pen-based dataset.



(b) Mushroom dataset.



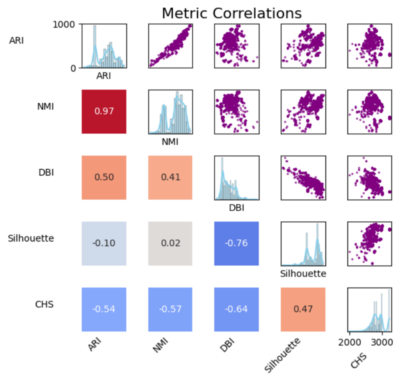
(c) Hepatitis dataset.

Figure 5: Heatmaps illustrating execution times for each dataset, showcasing performance across different configurations.

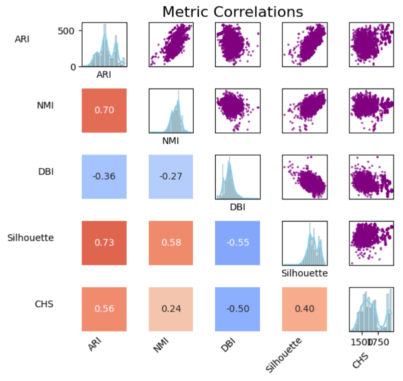
3.2.1 Preliminary Study

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

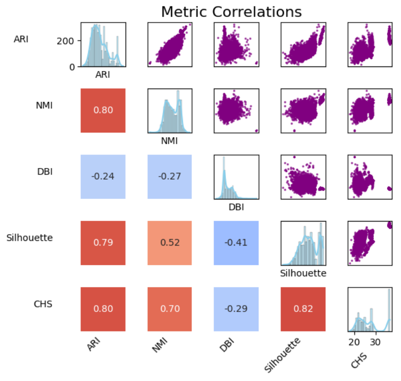
Figure 8 illustrates the relationships between the various metrics for the FCM algorithm. This matrix plot highlights the correlations between metrics (lower triangle), histogram distributions (diagonal), and scatterplots of metric values (upper triangle). A notable observation is the interaction between the fuzziness parameter and metrics like Silhouette and DBI. For instance, while lower fuzziness (mm) often improves silhouette scores due to sharper cluster boundaries, it can lead to higher DBI values, which might indicate a trade-off between cluster compactness and interpretability. Similar trends are observed across all datasets, though the degree of correlation varies.



(a) Pen-based dataset.



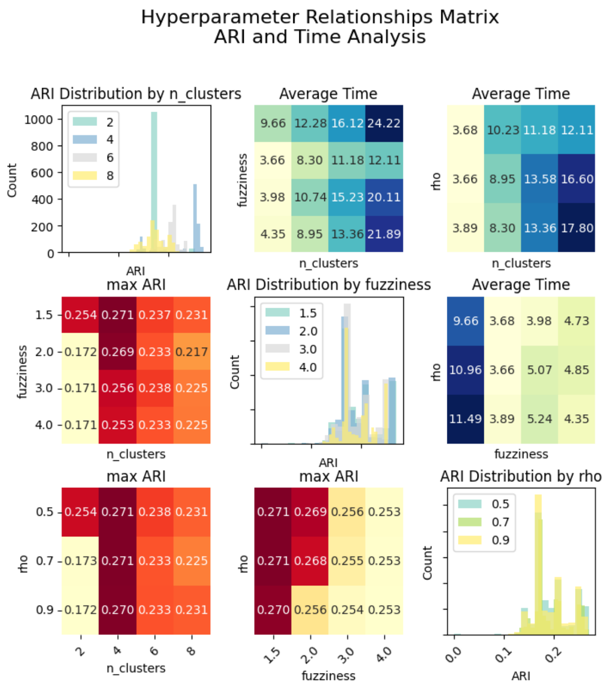
(b) Mushroom dataset.



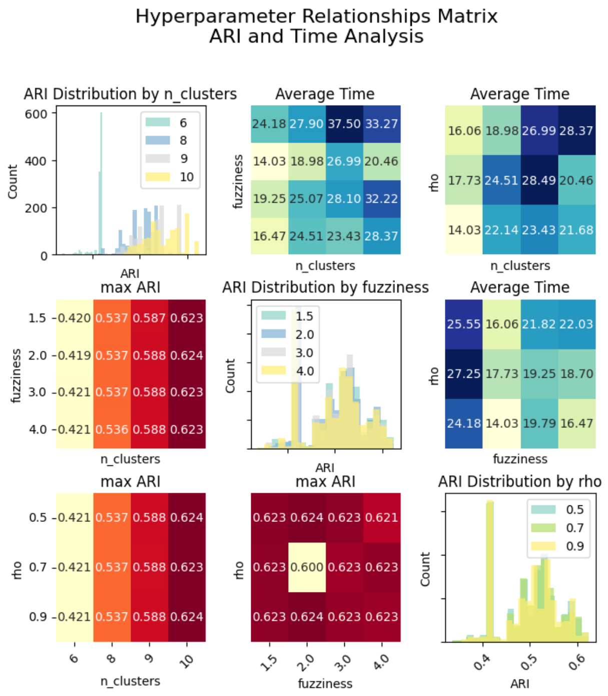
(c) Hepatitis dataset.

Figure 6: Metrics correlation acrross the three datasets.

Additionally, Figure 7 shows hyperparameter pairwise relationships, particularly the impact of fuzziness m and number of clusters n_clusters on clustering quality and execution time. Across datasets, higher fuzziness values consistently resulted in more computationally expensive runs, likely due to the increased complexity of assigning membership values. Interestingly, for datasets like Pen-based (10 classes), high cluster counts (e.g., n_clusters= 10) aligned closely with ground truth and yielded better metrics, such as ARI and NMI. Conversely, for Mushroom (2 classes), smaller values of n_clusters and lower fuzziness performed better.



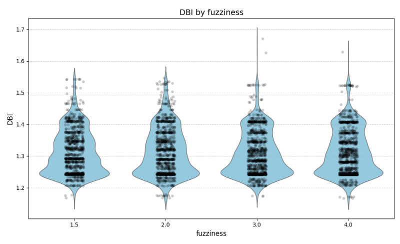
(a) Mushroom pairplot matrix for FCM



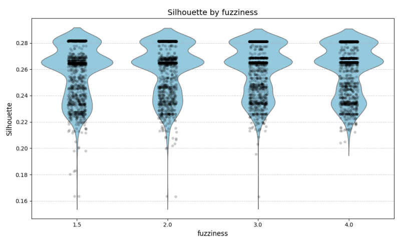
(b) Pen-based pairplot matrix for FCM

Figure 7: Hyperparameter pairplot matrices based on clustering metrics and execution time for FCM

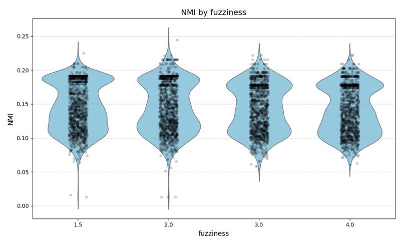
3.2.2 Fuzziness Parameter (m):



(a) Pen-based dataset.



(b) Mushroom dataset.



(c) Hepatitis dataset.

Figure 8: Metrics correlation acrross the three datasets.

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.