Implementation, Evaluation, and Comparison of K-Means and Other Clustering Algorithms

Carlos Jiménez, Sheena Lang, Zachary Parent, and Kacper Poniatowski

December 1, 2024

1 Introduction

This report focuses on evaluating various clustering techniques across three distinct datasets: Hepatitis, Mushroom, and Vowel, as detailed in Section 2. The clustering algorithms examined include K-Means, Fuzzy C-Means, Global K-Means, G-Means, OPTICS, and Spectral Clustering, as described in Sections 4.1 to 4.5 . We perform multiple experimental runs with different hyperparameters and assess the performance of each method using four evaluation metrics: two internal metrics (Davies-Bouldin Index and Calinski-Harabasz Index) and two external metrics (Adjusted Rand Index and F-Measure) as seen in 4.6. The results and analysis of these experiments are presented in Section 5.

2 Data

This section describes the datasets used in our study and details the preprocessing steps applied to prepare them for analysis.

3 Dataset Selection and Characteristics

This section outlines our dataset selection criteria and analyzes the characteristics of the chosen datasets for evaluating clustering algorithms.

3.1 Dataset Selection

To evaluate clustering algorithms, we selected three datasets with diverse characteristics in size, attribute types, class distribution, and missing data patterns. This diversity allows for a comprehensive analysis of algorithm performance under varying conditions.

The datasets vary in size, including a small dataset (Hepatitis), a medium-sized dataset (Vowel), and a large dataset (Mushroom), enabling an assessment of scalability and computational efficiency. They also include diverse attribute types: nominal, numerical, and mixed, ensuring the algorithms are tested across different data representations.

Class distribution was another consideration. Although clustering is unsupervised, class distributions serve as baselines for validating cluster quality. We included a balanced dataset (Vowel), a slightly imbalanced one (Mushroom), and a heavily imbalanced one (Hepatitis). Lastly, missing data patterns were addressed, with Hepatitis containing missing values to test robustness, while Vowel and Mushroom datasets are complete, allowing performance comparisons under ideal conditions.

3.2 Dataset Characteristics

The Hepatitis dataset includes 155 instances with a mix of 13 nominal and 6 numerical attributes. It represents a binary classification problem (survive vs. die), but the classes are imbalanced, with the majority class comprising 79.35% of the instances. Approximately 6.01% of the values are missing.

The Mushroom dataset consists of 8,124 instances with 22 nominal attributes. It represents a binary classification problem (edible vs. poisonous) with nearly balanced classes and contains no missing values.

The Vowel dataset contains 990 instances with 3 nominal and 10 numerical attributes. It represents a multi-class classification problem with 11 balanced classes and no missing values.

3.3 Selection Criteria

We selected the Hepatitis, Mushroom, and Vowel datasets to provide complementary characteristics for evaluating clustering algorithms. Our selection criteria focused on:

- **Dataset size variation** (small, medium, large)
- **Attribute type diversity** (nominal, numerical, mixed)

3.4 Data Preprocessing

This section outlines the preprocessing steps applied to prepare the three datasets (*Hepatitis*, *Mushroom*, and *Vowel*) for clustering analysis.

We began by replacing all missing values, denoted as `?`, with `NaN` to facilitate appropriate imputation techniques. The class label column, which was required only for post-clustering evaluation, was temporarily removed during preprocessing. To account for the varying characteristics of the datasets, we differentiated between categorical and numerical features using a heuristic-based approach. Columns were classified as categorical if their data type was `object` or if the proportion of unique values was below 5% of the dataset's size. All other columns were treated as numerical.

Numerical features were processed by imputing missing values with the column mean to preserve the overall data distribution, followed by rescaling to the [0, 1] range using Min-Max scaling. This scaling ensured that all features contributed equally during distance-based clustering. Categorical features were handled by imputing missing values with the most frequent category (mode), thus preserving the dominant data patterns. Binary categorical features were encoded using label encoding, while non-binary categorical features were one-hot encoded, creating separate columns for each category. This encoding strategy was selected based on recommendations for clustering algorithms like K-Means, where numerical representations avoid introducing arbitrary ordinal relationships.

After processing, the numerical and categorical features were concatenated to form the final dataset, and the class labels were label-encoded for use in evaluation metrics such as Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI). The preprocessing pipeline involved imputing missing values (mean for numerical and mode for categorical features), applying Min-Max scaling to numerical features, and using label encoding (binary categories) and one-hot encoding (non-binary categories) for categorical features. This approach ensures that the preprocessing pipeline is tailored to the unique requirements of clustering algorithms, while maintaining data integrity and minimizing potential biases.

4 Methods

This section provides an overview of the clustering algorithms used in our analysis, detailing their mechanisms, key parameters, and parameter variations. The methods include K-Means (Section 4.1), Improved K-Means (Section 4.2), Fuzzy C-Means (Section 4.3), OPTICS (Section 4.4), and Spectral Clustering (Section 4.5).

4.1 K-Means

K-Means is one of the most widely used clustering algorithms due to its simplicity and efficiency. It partitions a dataset into a predefined number of clusters by iteratively refining cluster centroids based on the distance between data points and the centroids.

4.1.1 Mechanism

The K-Means algorithm operates by minimizing the within-cluster variance, which is defined as the sum of squared distances between each point and the centroid of its assigned cluster. The process begins with the initialization of k centroids, where k represents the number of clusters. These centroids can either be selected randomly or provided as input.

Following initialization, each data point x_i is assigned to the nearest centroid c_j based on a distance metric, typically the Euclidean distance. This assignment is computed using the formula:

$$\text{Cluster}(x_i) = \arg\min_j ||x_i - c_j||_2^2.$$

After assigning clusters, the centroids are updated by recalculating their positions as the mean of all points assigned to each cluster. The new centroid c_j is determined using the equation:

$$c_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i,$$

where C_j is the set of points in cluster j, and $|C_j|$ is the number of points in that cluster.

Finally, the algorithm checks for convergence by comparing the updated centroids to the previous ones. If the difference between the new and old centroids is less than a predefined tolerance (ϵ), or if the maximum number of iterations is reached, the algorithm terminates. The difference is computed as:

$$\Delta = \sum_{j=1}^{k} ||c_j^{(t)} - c_j^{(t-1)}||_2^2 < \epsilon.$$

K-Means is computationally efficient but sensitive to the initial positions of centroids, which can cause it to converge to a local minimum.

4.1.2 Parameter Grid

The parameters and their variations used for K-Means in our experiments are summarized in Table 1.

Parameter	**Values**
Number of Clusters	2, 3, 4, 5, 6, 8, 10
Maximum Iterations	100, 300, 500
Tolerance (ϵ)	$1 \times 10^{-5}, 1 \times 10^{-4}, 1 \times 10^{-3}$
Random State	1, 2, 3, 4, 5

Table 1: Parameter grid for k-means.

The *Number of Clusters*, defining the number of clusters to form. The *Maximum Iterations* parameter specifies the upper limit for iterations during convergence. The *Tolerance* (ϵ) determines the convergence threshold with values ranging from 1×10^{-5} to 1×10^{-3}. Finally, the *Random State* controls the random seed, varying between 1 and 5 for reproducibility.

4.2 Improved K-Means

This section provides an overview of the improved K-Means algorithms implemented, Global K-Means and G-Means.

4.2.1 Global K-Means

The Global K-Means algorithm is an advanced clustering approach that improves upon the traditional K-Means by introducing an intelligent centroid initialization and incremental clustering strategy. [2] Our implementation focuses on addressing key limitations of standard clustering techniques through a novel candidate selection and optimization mechanism.

Mechanism

The Global K-Means algorithm operates through a progressive clustering process with several key improvements:

The clustering process starts with a single cluster by using standard K-Means with $k = 1$, and incrementally adds clusters. For each iteration, the optimal location for the new centroid is determined by minimizing the Within-Cluster Sum-of-Squares (WCSS). [2] Candidate points for the new centroid are selected based on their minimum distance from existing centroids, with the number of candidates dynamically adjusted based on the current cluster count. An efficient selection method is used via `np.argpartition`, ensuring scability.

The algorithm refines centroid placement by using vecotorized WCSS computation, reducing the computational complexity of the algorithm by simultaneously calculating the WCSS for all candidates. The configuration with the lowest WCSS is selected as the optimal solution for the current cluster count.

A unique aspect of this implementation is the caching mechanism. Intermediate results such as cluster labels, centroids, and distance matrices are stored persistently. This allows the algorithm to resume from cached states for higher values of k, thus reducing computational overhead dramatically by preventing re-computations of lower values of k. The caching mechanism is hash-based, ensuring compatibility with different datasets and configurations.

The implementation also includes an adaptive candidate reduction strategy. As the number of clusters increases, the number of candidate points are reduced to prevent the algorithm from becoming computationally infeasible. This adaptive strategy ensures that the algorithm remains efficient and scalable for large datasets, while maintaining high-quality clustering results.

Parameter Grid

The parameter grid for Global K-Means, highlighting its key configurable parameters, is shown in Table 2.

Parameter	Values
Number of Clusters	2, 3, 5, 10, 11, 12
Maximum Iterations	100, 300, 500
Convergence Tolerance	$1\times10^{-5}, 1\times10^{-4}, 1\times10^{-3}$
Random State	1, 2, 3, 4, 5

Table 2: Parameter grid for Global K-Means.

The *Number of Clusters* determines the number of clusters to form. The *Maximum Iterations* parameter sets the upper limit for convergence iterations. The *Convergence Tolerance* defines the threshold for centroid changes. Finally, the *Random State* ensures reproducibility by controlling the random seed, varying from 1 to 5.

4.2.2 G-Means

The G-Means algorithm is an advanced clustering techniques that improves upon the standard K-Means algorithm by addressing a fundamental limitation: the need to specify a predetermined number of clusters. G-Means dynamically determines the optimal number of clusters by recursively splitting clusters based on the statistical validation of their Gaussian distribution. [6]

Mechanism

The algorithm begins the clustering process by initializing a single cluster, obtained using standard K-Means with $k = 1$. For each cluster, the data points are split into two sub-clusters by applying K-Means with $k = 2$. The resulting clusters are then evaluated using Anderson-Darling Gaussianity test. This test evaluates whether the data points in the cluster are Gaussian distributed. [6]

If the test indicates both sub-clusters have a Gaussian distribution, the split is rejected and the original cluster remains unchanged. If at least one of the sub-clusters are not of Gaussian distribution, the split is accepted and the process is repeated recursively for each sub-cluster until all clusters are of Gaussian distribution, or the user-defined maximum depth is reached. [6]

Before applying the Gaussianity test, the algorithm projects the data onto the principal components using Principal Component Analysis (PCA) to ensure the Anderson-Darling test is applied to the most significant directions in the data.

To prevent over-segmentation, a minimum number of observations ($\min_{\text{obs}}$) is enforced for each cluster. If a cluster has fewer observations than the minimum threshold, it is not split further.

Parameter Grid

The parameter grid for G-Means, highlighting key configurable parameters, is summarized in Table 3.

Parameter	Values
Strictness	[0, 1, 2, 3, 4]
Minimum Observations	[1, 5, 10]
Maximum Depth	[5, 10, 15]
Random State	[1, 2, 3, 4, 5]

Table 3: Parameter grid for G-Means.

The *Strictness* (s) controls the sensitivity of the Gaussianity test, with values ranging from 0 to 4. The *Minimum Observations* ($\min_{\text{obs}}$) determines the minimum number of data points required to prevent splitting clusters with too few observations. The *Maximum Depth* ($\max_{\text{depth}}$) limits the recursive splitting process, with values ranging from 5 to 15. The *Random State* ensures reproducibility, with values ranging from 1 to 5.

4.3 Fuzzy C-Means

To be more precise, we implemented the ***Generalized Suppressed Fuzzy C-Means*** (gs-FCM) algorithm, which extends the Suppressed Fuzzy C-Means (s-FCM) approach by introducing a time-invariant, context-sensitive suppression rule [5, 8].

The s-FCM algorithm itself incorporates a constant suppression factor to enhance clustering performance. Intuitively, the gs-FCM algorithm retains the same primary objective as the original Fuzzy C-Means (FCM) algorithm: to partition a dataset into a predefined number of clusters, allowing data points to belong to multiple clusters with varying degrees of membership [3]. The addition of the suppression mechanism improves convergence efficiency and robustness, particularly in scenarios with imbalanced or noisy data.

4.3.1 Mechanism

Initially, the user specifies the number of clusters c and sets the fuzzy exponent $m > 1$, which controls the degree of fuzziness. Cluster prototypes are then initialized, either by applying intelligent initialization principles or by randomly selecting input vectors. A suppression rule and its corresponding parameter are chosen from predefined options, typically referenced in a lookup table. For this implementation, the suppression rule chosen is defined as follows:

$$\alpha_k = \frac{1}{1 - u_w + u_w \cdot (1 - \text{param})^{\frac{2}{1-m}}},$$

where u_w is the fuzzy membership of the winning cluster, param is the suppression parameter, and m is the fuzzy exponent.

At each iteration, fuzzy memberships are calculated for each data point $\mathbf{x}_k$. The algorithm determines the winning cluster and calculates a suppression rate α_k, which is used to modify the fuzzy memberships, effectively reducing the influence of over-represented data points.

Cluster prototypes are then updated using these suppressed memberships, and the process repeats until convergence. This suppression mechanism enhances clustering robustness by addressing imbalances in data distribution and improving result quality.

4.3.2 Parameter Grid

The parameters and their variations used for gs-FCM in our experiments are summarized in Table 4.

Parameter	**Values**
Number of Clusters	2, 3, 5, 10, 11, 12
Fuzzyness	1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0
Suppression Factor	Varies per data point

Table 4: Parameter grid for gs-FCM.
The *Number of Clusters* determining the number of clusters to form. The *Fuzzyness* parameter, controls the degree of fuzziness in the membership function. The *Suppression Factor* is context-sensitive and adapts per data point based on a predefined suppression rule.

4.4 OPTICS

OPTICS (Ordering Points To Identify the Clustering Structure) is a density-based clustering algorithm that creates an augmented ordering of data points to identify cluster structures [1].

4.4.1 Mechanism

OPTICS works by computing a special ordering of points based on their density-reachability relationships. The algorithm uses core-distance (representing neighborhood density) and reachability-distance (measuring density-connectivity between points) to determine cluster structure.

The algorithm processes points in a specific order, maintaining a priority queue ordered by reachability-distance. For each point:

1. The core-distance is computed
2. For each unprocessed neighbor, the reachability-distance is calculated
3. Points are added to the priority queue based on their reachability-distance
4. The process continues with the point having the smallest reachability-distance

This ordering produces a reachability plot, where valleys in the plot represent clusters. The ξ parameter is used to identify significant drops in reachability that indicate cluster boundaries. The minimum cluster size parameter ensures that identified clusters have a meaningful number of points.

Unlike traditional clustering algorithms, OPTICS does not explicitly produce clusters but rather provides a density-based ordering that can be used to extract clusters of varying density. This makes it particularly effective at finding clusters of arbitrary shape and identifying noise points in the dataset.

4.4.2 Parameter Grid

The parameter grid for OPTICS, detailing the key configurable parameters, is summarized in Table 5.

Parameter	**Values**
Metric	euclidean, manhattan
Algorithm	auto, ball_tree
Minimum Samples	5, 10, 20
Xi	0.01, 0.05, 0.1
Minimum Cluster Size	5, 10, 20

Table 5: Parameter grid for OPTICS.
The *Metric* specifies the distance metric used for calculating point distances, with options like euclidean and manhattan. The *Algorithm* defines the method for computing nearest neighbors, with choices such as auto and ball_tree. The *Minimum Samples* refers to the number of samples in a neighborhood needed for a point to be considered a core point. The *Xi* defines the minimum steepness on the reachability plot that constitutes a cluster boundary. The *Minimum Cluster Size* specifies the minimum number of samples required in a cluster.

4.5 Spectral Clustering

Spectral clustering is a technique that performs dimensionality reduction before clustering by using the eigenvectors of a similarity matrix [4].

4.5.1 Mechanism

The algorithm proceeds in three main steps:

1. Constructs a similarity matrix using either RBF kernel or k-nearest neighbors to capture relationships between data points
2. Computes the normalized Laplacian matrix and extracts its eigenvectors, creating a lower-dimensional representation that emphasizes cluster structure

3. Applies either k-means or cluster_qr to this transformed space to obtain the final clustering

This approach is particularly effective at identifying clusters of arbitrary shape, as it does not make assumptions about the geometric structure of the clusters.

4.5.2 Parameter Grid

The parameter grid for Spectral Clustering, including its key configurable parameters, is summarized in Table 6.

Parameter	**Values**
Number of Neighbors	5, 10, 20
Affinity	nearest_neighbors, rbf
Eigen Solver	arpack, lobpcg
Assign Labels	kmeans, cluster_qr
Random State	1, 2, 3, 4, 5

Table 6: Parameter grid for Spectral Clustering.

The *Number of Neighbors* ($n_{\text{neighbors}}$) specifies the number of neighbors used for constructing the affinity matrix. The *Affinity* parameter defines the type of affinity matrix, with options like nearest_neighbors and rbf. The *Eigen Solver* determines the strategy for eigenvalue decomposition, such as arpack or lobpcg. The *Assign Labels* parameter specifies the method for label assignment in the embedding space, using strategies like kmeans or cluster_qr. Finally, the *Random State* controls reproducibility, varying from 1 to 5.

4.6 Evaluation Metrics

We evaluated our clustering approaches using two internal and two external metrics to ensure a comprehensive assessment by capturing different aspects of clustering performance. The **Davies-Bouldin Index (DBI)** measures the compactness and separation of clusters, where lower values indicate better clustering. The **Calinski-Harabasz Index (CHI)** evaluates the ratio of between-cluster variance to within-cluster variance, favoring solutions with well-separated, compact clusters; higher scores are better. These two internal metrics together provide insights into the geometric quality of clusters.

On the other hand, external metrics assess alignment with ground truth labels. The **Adjusted Rand Index (ARI)** quantifies the agreement between true and predicted labels while correcting for chance, making it robust for comparing different clustering solutions. The **F-measure**, derived from precision and recall, evaluates how well the predicted clusters capture the true ones, focusing on class overlap. Together, these external metrics assess clustering accuracy and consistency, while complementing the internal metrics by incorporating the dataset's known structure. All metrics were implemented using the Python library **scikit-learn** [7].

5 Results and Analysis

In the figures below, we present independent and interaction effects between key parameters of each clustering algorithm across three datasets: *hepatitis*, *mushroom*, and *vowel*. Each grid combines boxplots and heatmaps; the boxplots demonstrate how individual parameters affect performance, while the heatmaps highlight synergistic effects between parameter pairs.

Full results, in tabular and graphical form, are available in the appendix (Section 7).

5.1 K-Means

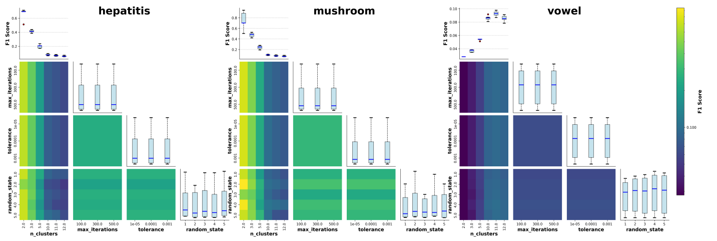

Figure 1: Parameter Interactions for K-Means Clustering.

K-Means demonstrated strong performance on the Mushroom and Hepatitis datasets. For Mushroom, it achieved an F-measure of 0.94 with the optimal parameters, highlighting its effectiveness. However, its performance on the Vowel dataset was poor, with a best F-measure of only 0.17. This low performance can likely be attributed to the dataset's higher optimal cluster count (11). As depicted in Figure 1, the optimal number of clusters varied across datasets. For Mushroom and Hepatitis, the highest F-measure was obtained with 2 clusters, matching the number of classes in these datasets. In contrast, for the Vowel dataset, the optimal result was achieved with 8 clusters, falling short of the actual number of classes (11). The inability to correctly identify the full number of clusters may explain the lower performance.

While variations in parameters such as the maximum number of iterations and tolerance had minimal effect on the F-measure, the choice of the random state significantly influenced the results. For instance, in the Mushroom dataset, the F-measure reached 0.94 with one random state but dropped dramatically to around 0.46 with another. This underscores the critical role of testing different initializations in K-Means clustering to ensure robust results.

5.2 Improved K-Means

This section provides an overview of the results obtained from the improved K-Means algorithms implemented, Global K-Means and G-Means.

5.2.1 Global K-Means

We found that the number of clusters had the most significant impact on the F-measure, as shown in Figure 2. For both the Mushroom and Hepatitis datasets, the F1 score was highest when number of clusters set to 2, and steadily decreased as the number of clusters increased. For the Vowel dataset, the F1 score was lowest with a small number of

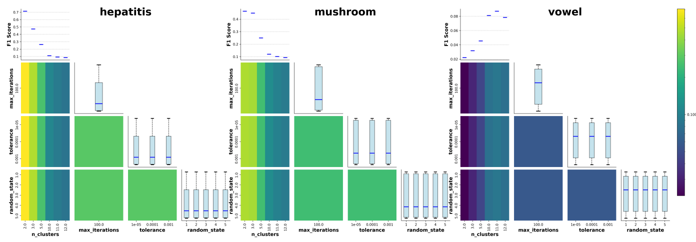

Figure 2: Hyperparameter interactions for Global K-Means

clusters, and increased as the number of clusters increased, peaking at number of clusters set to 11. These results are consistent with the number of classes in each dataset— 2 for Mushroom and Hepatitis, and 11 for Vowel.

The tolerance and $max_iteration$ parameters both had a negligable impact on the F1 score. One possible explanation for this is that, in Global K-Means, the tolerance parameter primarily influences the convergence criteria of the K-Means subroutine used at each step of the algorithm. As long as K-Means converges successfully for a given number of clusters, the overall clustering results remain largely unaffected. Similarly, the $max_iterations$ parameter acts as a safeguard to prevent excessively long runtimes but does not directly influence the final cluster placements, provided that convergence is reached within the allowed iterations. These findings suggest that factors such as the initial placement of centroids and the maximum number of clusters are more critical to the performance of Global K-Means than the tolerance or max iterations parameters.

5.2.2 G-Means

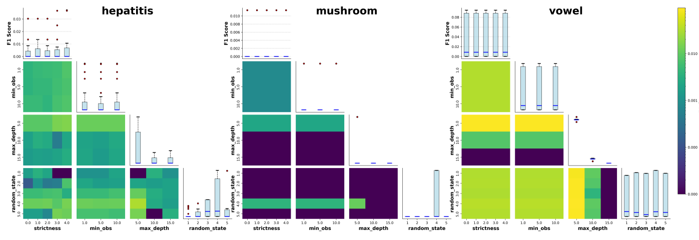

Figure 3: Hyperparameter interactions for G-Means

Compared to Global K-Means, the results are more varied for G-Means as shown in Figure 3.

The F1 score for the Mushroom dataset remains consistently low across all configurations of the hyperparameters. While there are some outliers, the overall clustering performance is poor, with F1 scores close to zero for most parameter combinations. The heatmaps show little variation across hyperparameter values, suggesting that changes to max_depth, strictness, or min_obs have negligible impact on the clustering outcome. The bar plot for $random_state$ indicates some sensitivity to initialization, but even the best cases fail to produce meaningful clusters. These results suggest that G-Means is ill-suited for the Mushroom dataset.

For the Hepatitis dataset, the F1 score shows greater variability across hyperparameter configurations. The heatmaps reveal that max_depth and $random_state$ have more noticeable impacts on clustering performance, with specific combinations producing better results. The box plots indicate significant outliers, particularly for strictness and max_depth, suggesting these parameters introduce instability. Overall, these results show G-Means hyperparameter combinations are sensitive for this dataset, highlighting the need for careful selection to optimize performance. Despite this, the F1 scores remain low, indicating G-Means cannot accurately predict the clusters in the Hepatitis dataset.

The F1 score remains consistently low across all hyperparameter configurations for the Vowel dataset. The heatmaps suggest that max_depth is the most significant hyperparameter, with lower levels producing the best results. Having said this, the results are still poor. The box plots show negligible variability for most parameters, with no significant outliers. These results indicate that G-Means struggles to produce meaningful clusters for this dataset, likely due to the dataset's complexity and the algorithm's limitations in handling high-class scenarios.

5.3 Fuzzy C-Means

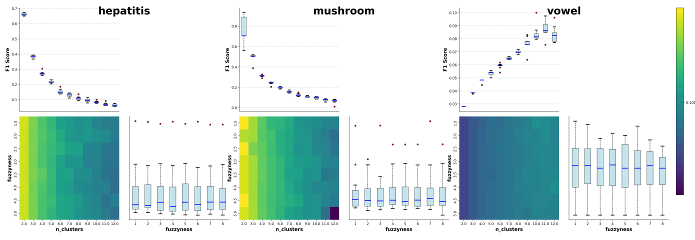

Figure 4: Parameter interactions for gs-FCM

Figure 4 presents the interaction effects between the key parameters of the Fuzzy C-Means clustering algorithm: the number of clusters (*n_clusters*) and the fuzziness coefficient, across three datasets: *hepatitis*, *mushroom*, and *vowel*.

The *hepatitis* dataset shows optimal performance with fewer clusters and lower fuzziness values (closer to 1.5), suggesting that simpler, more crisp clustering is appropriate. The *mushroom* dataset demonstrates robustness across parameter combinations, maintaining high F1 scores regardless of parameter settings. In contrast, the *vowel* dataset benefits from higher complexity, achieving better performance with larger *n_clusters* and higher fuzziness values.

These results highlight the importance of dataset-specific parameter tuning in Fuzzy C-Means clustering, as each dataset exhibits distinct optimal configurations based on its underlying characteristics.

5.4 OPTICS

OPTICS demonstrated varying performance across the three datasets. For Hepatitis, it achieved moderate results with an F-measure around 0.23 at optimal parameters. The Mushroom dataset showed very poor performance with F-measures near 0, despite its simpler binary classification na-

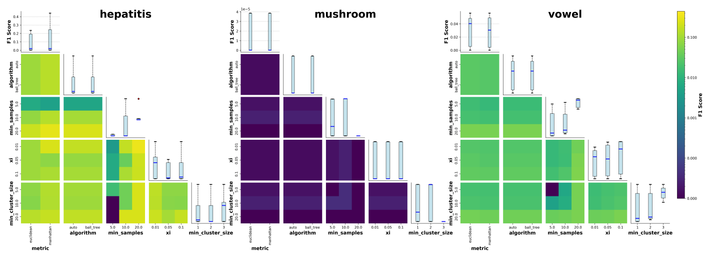
Figure 5: Parameter interactions for OPTICS

ture. The Vowel dataset achieved F-measures around 0.05, which while low, was better than the Mushroom results.

As shown in Figure 5, the algorithm's performance was significantly influenced by parameter choices. The *min_samples* parameter had the strongest effect across all datasets, while the *xi* parameter showed notable impact particularly on the Hepatitis dataset. The metric choice (euclidean vs manhattan) and algorithm implementation (auto vs ball_tree) had relatively minor effects on the results.

The *min_cluster_size* parameter demonstrated interesting interactions with other parameters, particularly visible in the Hepatitis dataset where higher values generally led to better F-measures when combined with appropriate *min_samples* settings. This suggests that OPTICS performs better when allowed to form larger, more stable clusters rather than numerous small ones.

5.5 Spectral Clustering

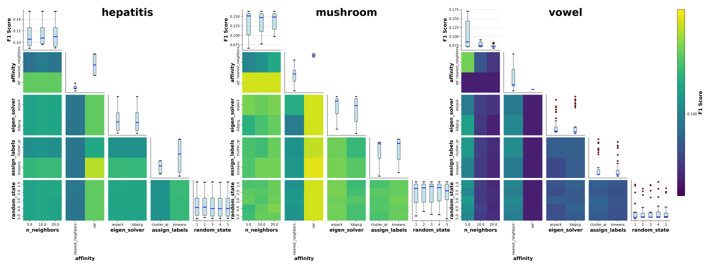
Figure 6: Parameter interactions for Spectral Clustering

As seen in 6, spectral clustering showed varying performance across different parameter configurations and datasets. It achieved roughly 0.15 F1 scores at best across the datasets, including achieving an F1 of 0.16 on the mushroom dataset using RBF affinity with ARPACK solver and assigning labels using kmeans. It was the best performing method on the difficult-to-cluster vowel dataset, with an F1 score of 0.17.

The algorithm's performance was significantly influenced by several key parameters:

- **Affinity Matrix**: Nearest-neighbors consistently outperformed RBF kernel for the vowel dataset, while RBF showed superior results for the mushroom and hepatitis datasets
- **Solver Choice**: LOBPCG solver generally provided faster execution times (0.03-0.08s) compared to ARPACK (0.1-0.4s), and the results were similar
- **Number of Neighbors**: Lower values (n_neighbors=5) produced more stable results for the vowel dataset, but it mattered less for the other datasets
- **Assignment Method**: cluster_qr and kmeans assignments showed similar clustering quality
- **Random State**: The results were very similar across different random states, indicating that the results are consistent and not highly dependent on the initialization

We saw interesting interactions between the parameters. For example, using a Kmeans assignment with the rbf affinity matrix showed particular effectiveness on the hepatitis dataset.

5.6 Summary

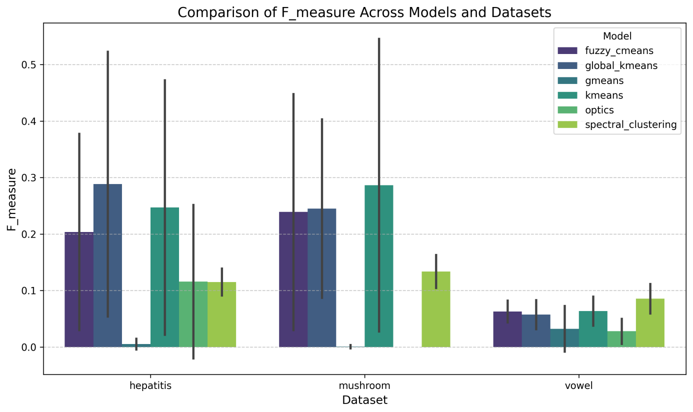

Figure 7: F1 Score Comparison Across Datasets

The Mushroom dataset, with its simple two-class structure, was best handled by K-Means, Spectral Clustering, and Fuzzy C-Means. K-Means and Spectral Clustering achieved high F1 scores when the cluster counts matched the natural class count. Fuzzy C-Means maintained consistently high performance across a range of parameter combinations, while more complex methods like G-Means and OPTICS added no benefit.

The Hepatitis dataset, with moderate complexity and imbalanced classes, showed mixed results. K-Means and Global K-Means performed well when clusters aligned with the natural classes. OPTICS and Fuzzy C-Means succeeded with careful tuning, with OPTICS favoring larger clusters and Fuzzy C-Means performing better with lower fuzziness values. G-Means, however, exhibited instability due to sensitivity to hyperparameters like max_depth and strictness.

The Vowel dataset, characterized by high-class complexity and overlapping features, posed significant challenges. All algorithms struggled, with low F1 scores across the board. Fuzzy C-Means showed some improvement with higher fuzziness values and larger cluster counts, but the gains were limited. Spectral Clustering and other methods failed to handle the dataset's complexity effectively, emphasizing the need for advanced clustering techniques.

6 Conclusion

Clustering is a core machine learning technique used in many applications, but choosing the right clustering algorithm and hyperparameters is not always straightforward. In this report, we have evaluated several clustering algorithms and their hyperparameters on three distinct datasets, exploring the strengths and weaknesses of each method. Here we present our key findings and practical implications of the results.

6.1 Key Findings

- K-means demonstrated consistent performance with moderate execution times (0.003-0.38s) and balanced clustering metrics
- Fuzzy C-means showed particularly fast runtimes even on the largest dataset (mushroom), while offering similar performance to K-means
- Global K-means exhibited longer execution times but provided more stable clustering than K-means results across different random initializations
- Gmeans did not demonstrate competitive results generally, but exhibited decent F1 scores on the vowel dataset
- OPTICS generally showed weak performance, probably due to the lack of natural density variations in the datasets
- Spectral clustering performed well with nearest-neighbors affinity, especially on the vowel dataset, achieving the best F1 score on this difficult-to-cluster dataset

6.2 Practical Implications

- For well-separated clusters: K-means offers the best balance of speed and accuracy
- For overlapping clusters: Fuzzy C-means provides more nuanced cluster assignments, and it's faster than K-means
- For datasets with varying densities and non-convex shapes: OPTICS or Spectral clustering are more suitable
- When cluster number is unknown: Gmeans, Fuzzy C-means, OPTICS or Spectral clustering are preferable
- When stability is crucial: Global K-means offers more consistent results, at the cost of longer runtimes

6.3 Limitations and Future Work

- Current analysis is limited to specific parameter ranges and could be expanded
- Computational efficiency could be improved, particularly for Global K-means and OPTICS
- Integration of multiple algorithms in an ensemble approach could potentially yield better results

6.4 Final Remarks

Each clustering algorithm demonstrates distinct strengths and weaknesses, suggesting that the choice of algorithm should be primarily driven by specific application requirements and data characteristics. While K-means and Fuzzy C-means offer good general-purpose solutions, specialized algorithms like OPTICS and Spectral clustering provide advantages for specific data distributions. Future work should focus on developing hybrid approaches that can combine the strengths of multiple algorithms while mitigating their individual weaknesses.

References

[1] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: ordering points to identify the clustering structure. *SIGMOD Rec.*, 28(2):49–60, June 1999.

[2] Jakob J. Verbeek Artistidis Likas, Nikos Vlassis. The global k-means clustering algorithm. 12, 2001.

[3] James C Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & geosciences*, 10(2-3):191–203, 1984.

[4] Anil Damle, Victor Minden, and Lexing Ying. Simple, direct and efficient multi-way spectral clustering. *Information and Inference: A Journal of the IMA*, 8(1):181–203, 06 2018.

[5] Jiu-Lun Fan, Wen-Zhi Zhen, and Wei-Xin Xie. Suppressed fuzzy c-means clustering algorithm. *Pattern Recognition Letters*, 24(9-10):1607–1612, 2003.

[6] Charles Elkan Greg Hamerly. Learning the k in k-means. 27, 2003.

[7] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

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

7 Appendix

Table 7: Best Performing Models by Dataset (Based on F-Measure)

Dataset	Model	F Measure	Ari	Chi	Dbi	Runtime (s)
hepatitis	global_kmeans	0.7122	0.2555	36.5479	1.9827	0.0625
mushroom	kmeans	0.9434	0.7881	1070.0670	2.6995	0.3844
vowel	spectral_clustering	0.1701	0.0438	19.2650	4.8690	0.0268

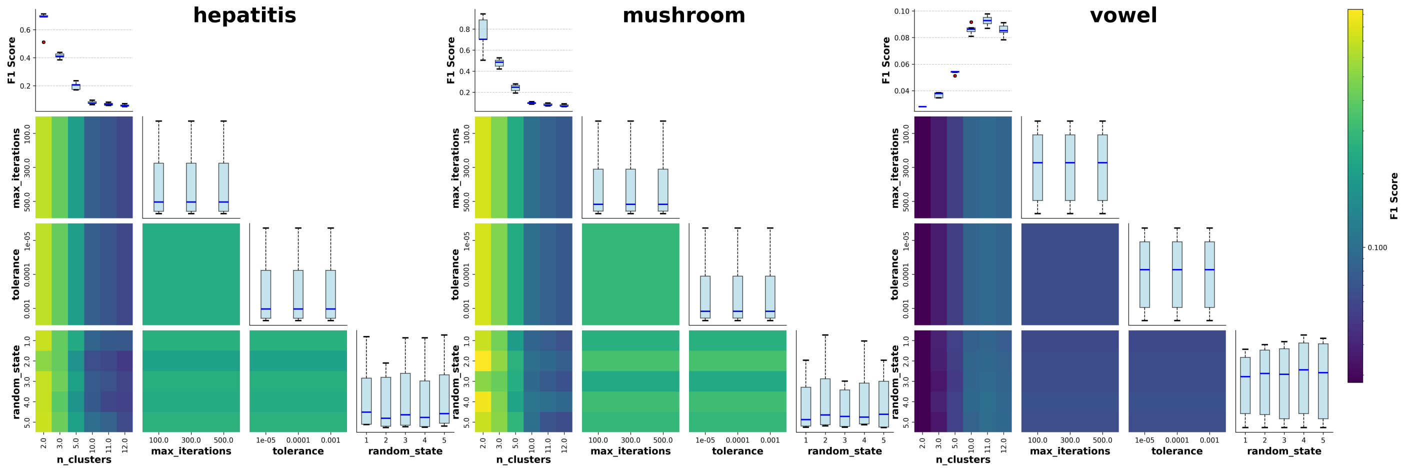

Figure 8: Parameter interactions for KMeans

Table 8: Best Parameter Configurations for Kmeans by Dataset

Dataset	N Clusters	Max Iterations	Tolerance	Random State
hepatitis	2.00000	300.00000	0.00010	5.00000
mushroom	2.00000	100.00000	0.00100	2.00000
vowel	11.00000	300.00000	0.00001	4.00000

Table 9: Performance Metrics for Best Kmeans Configurations by Dataset

Dataset	F Measure	Ari	Chi	Dbi	Runtime (s)
hepatitis	0.7122	0.2555	36.5479	1.9827	0.0033
mushroom	0.9434	0.7881	1070.0670	2.6995	0.3842
vowel	0.0978	-0.0049	139.2350	1.3573	0.2512

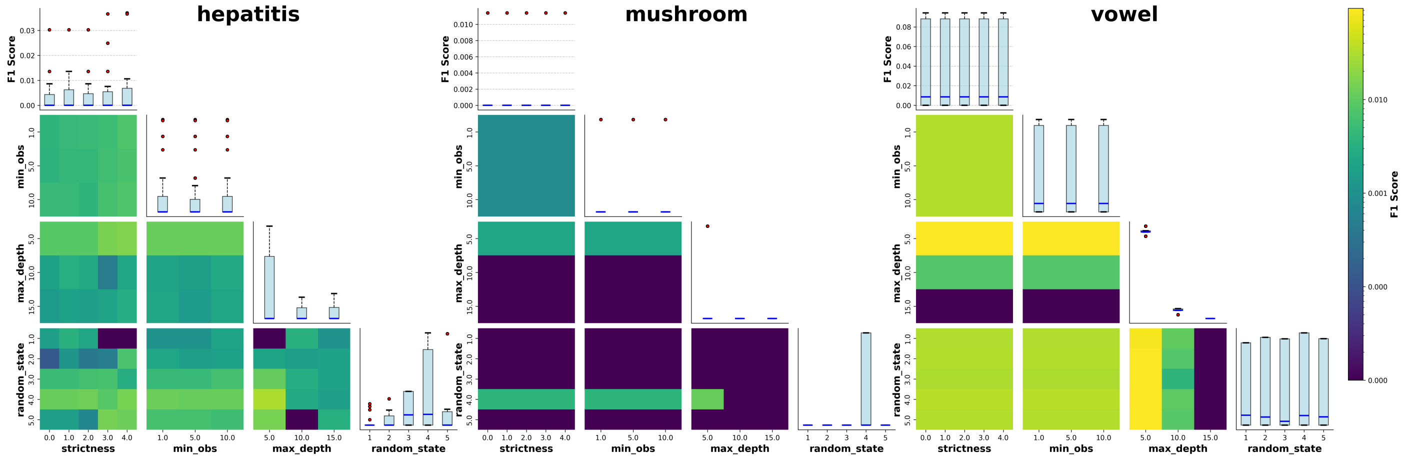

Figure 9: Parameter interactions for GMeans

Table 10: Best Parameter Configurations for Gmeans by Dataset

Dataset	N Clusters	Random State	Strictness	Min Obs	Max Depth
hepatitis	8.00000	4.00000	4.00000	1.00000	5.00000
mushroom	10.00000	4.00000	4.00000	5.00000	5.00000
vowel	10.00000	4.00000	3.00000	5.00000	5.00000

Table 11: Performance Metrics for Best Gmeans Configurations by Dataset

Dataset	F Measure	Ari	Chi	Dbi	Runtime (s)
hepatitis	0.0370	0.0355	18.6704	1.7756	0.0300
mushroom	0.0114	0.2071	701.0287	2.0688	1.3070
vowel	0.0942	-0.0042	175.3446	1.3020	0.0580

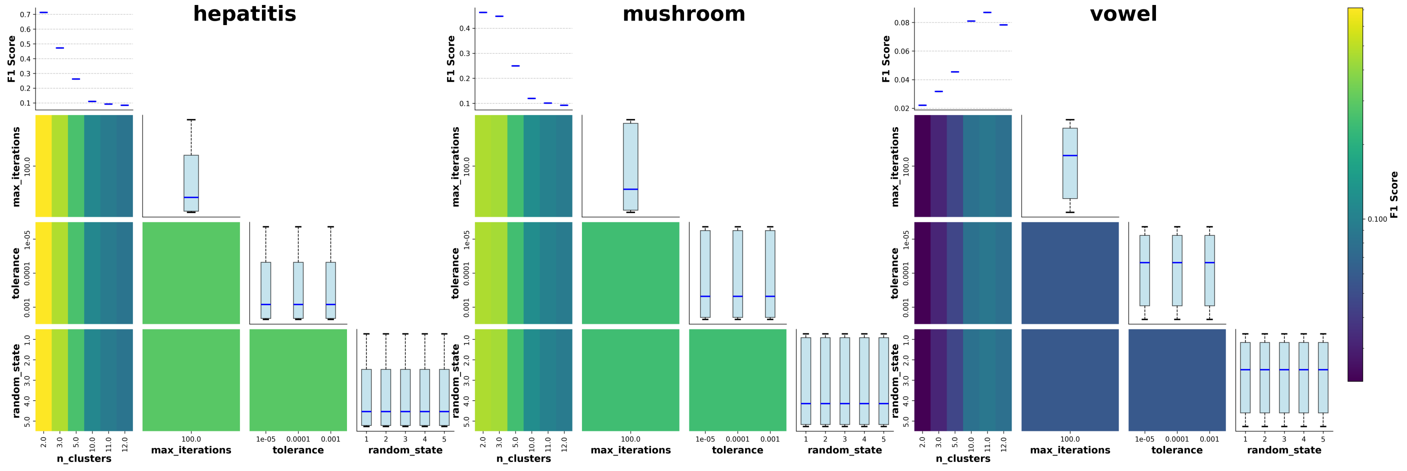

Figure 10: Parameter interactions for Global KMeans

Table 12: Best Parameter Configurations for Global Kmeans by Dataset

Dataset	N Clusters	Max Iterations	Tolerance	Random State
hepatitis	2.00000	100.00000	0.00100	4.00000
mushroom	2.00000	100.00000	0.00100	4.00000
vowel	11.00000	100.00000	0.00010	4.00000

Table 13: Performance Metrics for Best Global Kmeans Configurations by Dataset

Dataset	F Measure	Ari	Chi	Dbi	Runtime (s)
hepatitis	0.7122	0.2555	36.5479	1.9827	0.0624
mushroom	0.4626	0.0032	585.5940	2.3329	22.9219
vowel	0.0870	-0.0091	208.1905	1.1815	0.7781

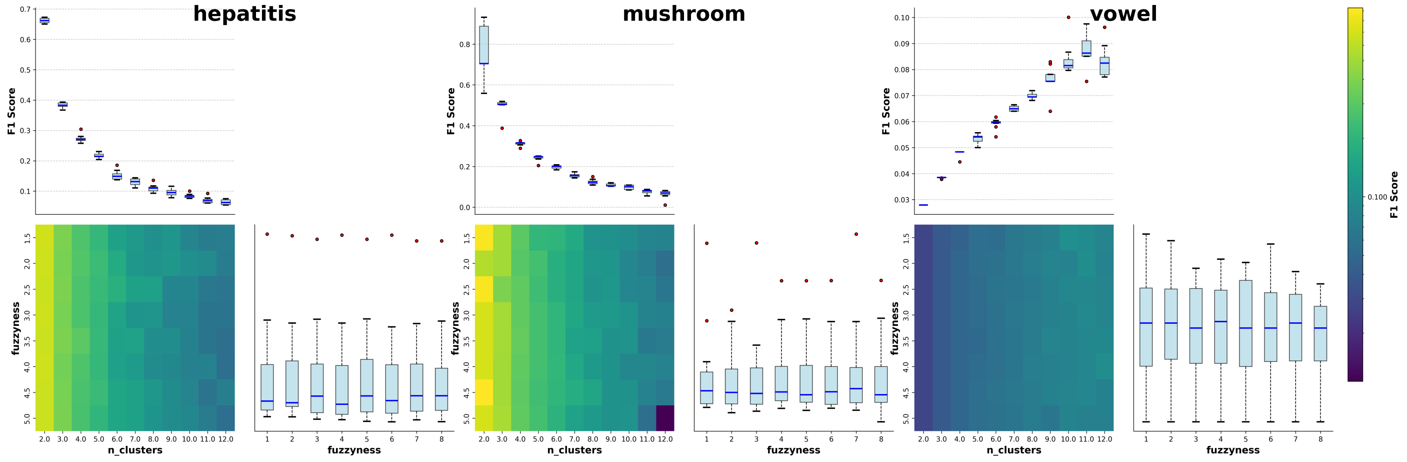

Figure 11: Parameter interactions for Fuzzy C-Means

Table 14: Best Parameter Configurations for Fuzzy Cmeans by Dataset

Dataset	N Clusters	Fuzzyness
hepatitis	2.00000	1.50000
mushroom	2.00000	4.50000
vowel	10.00000	1.50000

Table 15: Performance Metrics for Best Fuzzy Cmeans Configurations by Dataset

Dataset	F Measure	Ari	Chi	Dbi	Runtime (s)
hepatitis	0.6729	0.1703	39.6712	1.9149	0.0034
mushroom	0.9318	0.7483	1072.2265	2.6839	0.1968
vowel	0.1001	-0.0042	133.6969	1.2042	0.0138

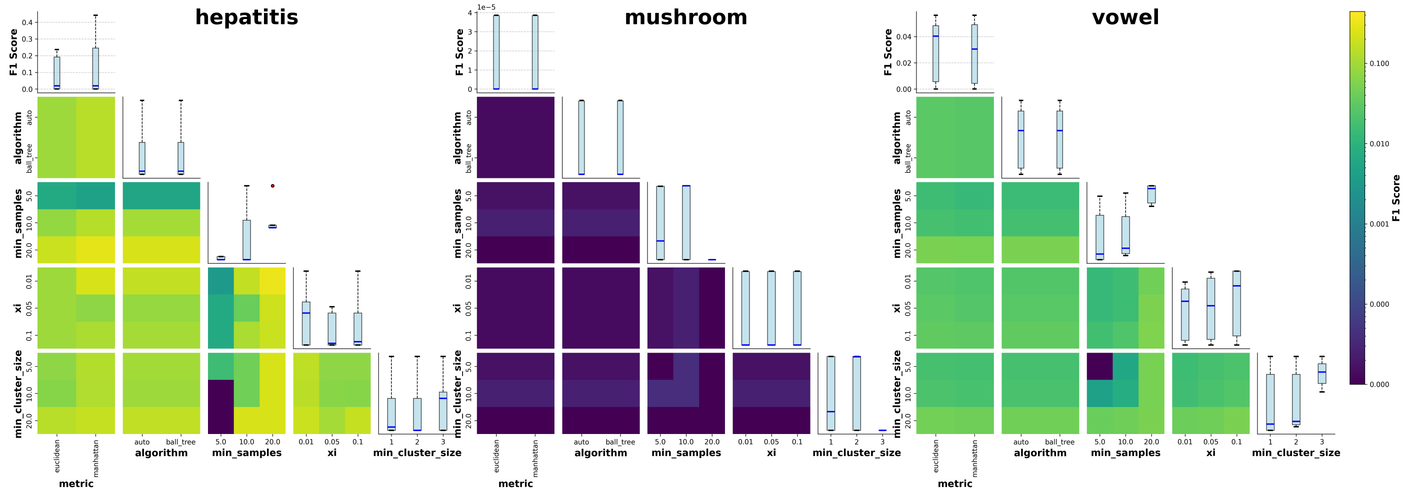

Figure 12: Parameter interactions for OPTICS

Table 16: Best Parameter Configurations for Optics by Dataset

Dataset	N Clusters	Metric	Algorithm	Min Samples	Xi	Min Cluster Size
hepatitis	1.00000	manhattan	auto	20.00000	0.01000	20.00000
mushroom	196.00000	euclidean	ball_tree	10.00000	0.10000	5.00000
vowel	15.00000	manhattan	ball_tree	20.00000	0.10000	10.00000

Table 17: Performance Metrics for Best Optics Configurations by Dataset

Dataset	F Measure	Ari	Chi	Dbi	Runtime (s)
hepatitis	0.4424	0.0000	21.1174	2.6439	0.0379
mushroom	0.0000	0.0486	172.0067	0.8884	7.3716
vowel	0.0564	-0.0119	388.3954	0.7410	0.2441

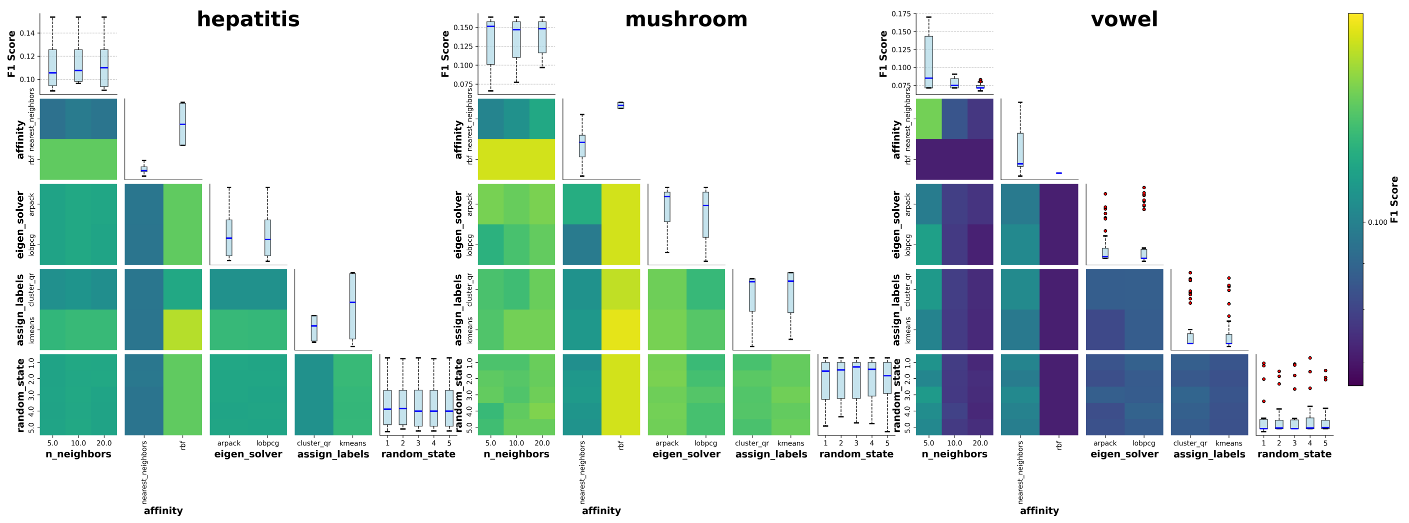

Figure 13: Parameter interactions for Spectral Clustering

Table 18: Best Parameter Configurations for Spectral Clustering by Dataset

Dataset	N Clusters	Random State	N Neighbors	Affinity	Eigen Solver	Assign Labels
hepatitis	8.00000	3.00000	10.00000	rbf	lobpcg	kmeans
mushroom	8.00000	2.00000	5.00000	rbf	arpack	kmeans
vowel	8.00000	4.00000	5.00000	nearest_neighbors	lobpcg	cluster_qr

Table 19: Performance Metrics for Best Spectral Clustering Configurations by Dataset

Dataset	F Measure	Ari	Chi	Dbi	Runtime (s)
hepatitis	0.1537	0.1916	14.3724	1.7614	0.0258
mushroom	0.1636	0.3510	770.8375	1.7358	6.1551
vowel	0.1701	0.0438	19.2650	4.8690	0.0268