Clustering Non-Convex Datasets: A Comparative Study Across Hard, Soft, Model-Based, Spectral, and Density-Based Clustering Techniques

Priyanka Goel, **Siddhant Mohanty**, **Veda Sahithi Bandi**

Abstract - This project compares the performance of several clustering algorithms on a variety of simulated datasets, including both convex and non-convex structures. We evaluate classical and modern techniques—including K-Means, Probabilistic Distance Clustering Adjusted for Cluster Size (PDQ), Gaussian Mixture Models (GMM), Density-Based Spatial Clustering of Applications with Noise (DBSCAN), and Spectral Clustering—across four synthetic scenarios designed to reflect a range of cluster shapes and densities, with varying numbers of clusters. To assess scalability, we also included a high-dimensional (10D) scenario. Each simulation is repeated 20 times for robustness. Our findings show that density-based and spectral methods consistently outperform centroid-based approaches like K-Means in identifying non-convex clusters. These results highlight the importance of algorithm selection in real-world clustering tasks involving complex or irregular structures.

1 Introduction

Once an analyst has determined that clustering is the best approach to tackle a problem of interest, they are faced with the daunting task of selecting an appropriate clustering method. They must make choices about data preprocessing, how many clusters to model, distance metrics, and the clustering algorithm itself—each of which can significantly impact the quality of the final results. Given the wide range of available methods and their varying assumptions, benchmarking has become a crucial tool for evaluating the robustness and suitability of clustering techniques in different scenarios (Van Mechelen et al., 2023).

Among the most widely used clustering algorithms are K-Means (Na et al., 2010), Gaussian Mixture Models (GMM)(Z. Wang et al., 2019; McLachlan and Basford, 1988), fuzzy clustering methods like PDQ (Jimeno et al., 2021), Spectral Clustering (S. Wang et al., 2015), and DBSCAN (Ester et al., 1996). These span a variety of clustering paradigms: centroid-based, model-based, soft membership, graph-based, and density-based, respectively. While methods like

K-Means and GMM assume convex, often spherical, cluster structures, graph-based and density-based methods are more flexible, capable of detecting clusters of arbitrary and non-convex shapes. This flexibility makes them more suitable for real-world data, where well-separation and ideal boundaries between groups cannot be assumed.

In this project, we perform a controlled comparison of the aforementioned algo- rithms on simulated datasets designed to test their ability to handle non-convexity, overlap, and variations in dimensionality. We examine four 2D scenarios with varying numbers of clusters (either 3–5 or 2–4, depending on the dataset), and one 10-dimensional case to explore scalability.

To this end, we use a mix of external validation metrics including Adjusted Rand Index (ARI)(Hubert and Arabie, 1985), Normalized Mutual Information (NMI)(Knops et al., 2006), Folkes-Mallows Index (FMI)(Murugesan et al., 2021), Jaccard Index (JCI)(Desgraupes, 2018), and algorithm runtime. Using these, we aim to understand the conditions under which each method excels or fails.

2 Methodology

2.1 Simulation Design

We simulated four different scenarios in 2D, each reflecting specific challenges encountered in real-world clustering tasks. The datasets varied in structure ranging from well-separated convex shapes to complex non-convex forms like concentric circles and moons. Each dataset was generated using a combination of functions from R packages such as MASS (for multivariate normal sampling) and custom routines for curved geometries. All datasets were validated to avoid missing, infinite, or undefined values.

Each dataset was simulated 20 times to evaluate algorithm performance across multiple runs to ensure that the influence of random variation in the data generation process on our final results was minimal. For both spherical cluster datasets and the blob with moons dataset, we began with 3 clusters of 200 observations each and later extended to 5 clusters. The spherical clusters were generated using bivariate gaussian distributions. Adjusting the variance of these distributions allowed us to define the degree of separation between the clusters. The exotic clusters in the third dataset—the moons and slashes—were generated by introducing normally-distributed perturbations to portions of sinusoids and line segments respectively. We also added outliers to the clusters in this dataset to test the robustness of our algorithms (Figure 1).

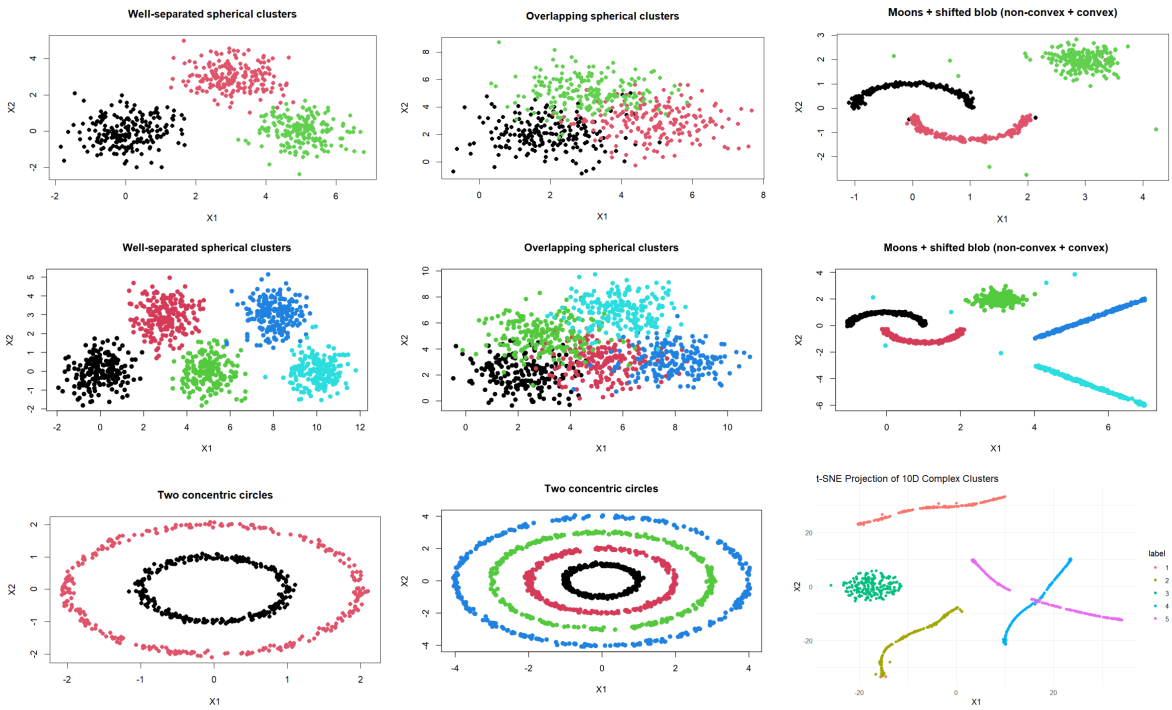

Figure 1: Example plots of the synthetic datasets. The first row shows 2D datasets with 3 clusters (spherical, overlapping, moons with a blob); the second row shows the same with 5 clusters. The third row includes concentric circles (2 and 4 clusters) and a t-SNE projection of the 10D dataset with moons and slashes.

The concentric circles dataset began with 2 clusters of 300 observations each and were extended to 4 clusters. These circles were also generated with non-uniform point density to assess how well the chosen algorithms adapted to differences in compactness.

Finally, we generated a second extension of the blob with moons and slashes dataset with 5 clusters in 10 dimensions. Pictured is a t-SNE projection of this data into 2 dimensions for the purpose of visualization. The strategy used to generate the cluster shapes was analogous to the lower dimensional scenario; we used a 10-variate normal for the blob, multidimensional sinusoids with normal noise, and two line segments with noise in all 10 dimensions.

2.2 Clustering Methods Compared

In this study, we evaluate five clustering algorithms, each representing a different methodological paradigm and implemented using established R packages (Team, 2010). These methods are chosen for their varied assumptions and strengths in handling convexity, non-convexity, overlapping clusters, and noise.

We began with K-Means, a classic centroid-based method that partitions the data by minimizing within-cluster variance. Using the *kmeans()* (Team, 2010) function with $nstart = 25$, we mitigated the deleterious effects of poorly

chosen starting cluster centers by choosing the best result from multiple random starts. K-Means assumes convex, similarly sized clusters and assigns each point exclusively to one cluster, which causes it to perform poorly on non-convex structures and overlapping data.

PDQ (Probabilistic Distance-based Clustering Adjusted to Cluster size), implemented via the *PDQ()* function from the "FPDclustering" package (Tortora and Palumbo, 2025), offers a fuzzy clustering approach; PDQ allows each point to belong to multiple clusters with varying degrees of membership, reflecting uncertainty and transitional behavior in the data. A key feature of PDQ is that it adjusts for cluster size and provides more balanced results in the presence of unequal group proportions. While it is well-suited for overlapping and imbalanced clusters, similarly to K-Means, it still requires the number of clusters to be specified and remains sensitive to initialization.

To complement these, we also used Gaussian Mixture Models (GMMs) via the *Mclust()* function from the "mclust" package (Haughton et al., 2009; Scrucca et al., 2016). GMM is a model-based clustering algorithm that assumes data is generated from a mixture of Gaussian distributions, assigning each point a probability of belonging to each cluster. We used the flexible *"VVV"* model, which allows each cluster to have its own volume, shape, and orientation. This makes GMM particularly effective for overlapping or elliptical clusters. However, its performance can degrade when the data deviates from Gaussian assumptions or exhibits strongly non-convex boundaries.

For handling such complex shapes, we turned to Spectral Clustering, implemented with *specc()* from "kernlab" (Karatzoglou et al., 2004). This graph-based method constructs a similarity matrix, computes the Laplacian, and uses its eigenvectors for dimensionality reduction, followed by a final K-Means step. Spectral Clustering is particularly strong on non-convex structures like moons or concentric circles, where traditional algorithms fail. We used the Gaussian kernel (*rbfdot*) with the kernel width parameter (σ) automatically estimated from the data. While this approach works reasonably well in many cases, performance can be greatly influenced by the choice of kernel parameters, especially in datasets with varying densities. Its main drawback lies in computational cost, especially with large datasets, due to the need for eigen-decomposition.

Lastly, we included DBSCAN using the *dbscan()* function(Hahsler, 2017). DBSCAN identifies clusters as regions of high point density and labels outliers as noise. It does not require the number of clusters to be specified, which is advantageous for exploratory analysis. Instead, it uses two key parameters —*eps* (neighborhood radius) and *minPts* (minimum points per cluster); we used $minPts = 4$ (Sander et al., 1998; Schubert et al., 2017). To determine

an appropriate value for *eps*, we employed a brute-force grid search over a range of candidate values. For each candidate *eps*, DBSCAN was run, and the clustering results were evaluated using the Adjusted Rand Index (ARI). The *eps* value that maximized the ARI was selected as the optimal value. In all 2D datasets, the brute-force method suggested that $eps = 0.5$ was optimal, and in the 10D dataset, *eps* was found to be 0.4. While the brute-force method has a higher computational cost and scales poorly with the number of observations in a dataset, it guarantees the optimal choice of parameter by exhausting all possible options.

Although we also considered using the traditional k-nearest neighbors (*kNN*) approach along with the elbow method to determine *eps* (Hahsler, 2017), the brute-force search yielded better results in terms of clustering quality and consistency. DBSCAN is capable of discovering clusters of arbitrary shapes, but its performance remains highly sensitive to the choice of *eps*, particularly when dealing with datasets that exhibit varying densities. This brute-force approach allows for more precise tuning and a more robust selection of *eps*, improving DBSCAN's ability to handle non-convex and irregular cluster structures.

Together, these methods offer a broad perspective on clustering performance, ranging from rigid assignment to probabilistic and graph-theoretic approaches, allowing us to assess how well each adapts to the unique demands of non-convex and noisy data.

2.3 Evaluation Criteria

To assess the performance of the clustering algorithms, we used several external validation metrics that compare the predicted cluster labels with the true labels available in the simulated datasets. These metrics capture different notions of clustering quality, such as pairwise agreement, mutual information, and set similarity.

We first considered the Adjusted Rand Index (ARI), calculated using the *ARI()* function from the "MixGHD" package (Tortora et al., 2021), which measures the similarity between two clusterings while adjusting for the expected similarity of random assignments. In this case, with simulated data, the second clustering is simply the true labels. The ARI is based on counting pairs of points that are assigned consistently in both clusterings. It is defined as:

$$\text{ARI} = \frac{\text{Index - Expected Index}}{\text{Max Index - Expected Index}}$$

where the numerator corrects for chance agreement. The ARI ranges from -1 (worse than random) to 1 (perfect clustering in our case), with 0 indicating

clustering no better than random.

Next, we used the Normalized Mutual Information (NMI), computed using the *NMI()* function from the "aricode" package (McDaid et al., 2011), quantifies how much information the predicted cluster assignment provides about the true labels. It is based on mutual information and entropy, and we use the max-normalized variant, where the denominator is the maximum of the entropies of the two clustering.

$$\text{NMI} = \frac{\text{I}(\text{Y};\hat{\text{Y}})}{\max(\text{H}(\text{Y}),\text{H}(\hat{\text{Y}}))}$$

where $I(Y;\hat{Y})$ is the mutual information between the true labels Y and predicted labels $\hat{Y}$, and $H(\cdot)$ denotes entropy. NMI ranges from 0 (no shared information) to 1 (perfect agreement).

We also included two pairwise comparison metrics: The Folkes-Mallows Index (FMI) and Jaccard Index (JCI), both computed using the *extCriteria()* function from the "clusterCrit" package (Desgraupes, 2018). FMI evaluates the geometric mean of precision and recall for point pairs:

$$\text{FMI} = \sqrt{\frac{TP}{TP+FP} \cdot \frac{TP}{TP+FN}},$$

where TP, FP, and FN refer to the number of true positive, false positive, and false negative pairs, respectively.

The Jaccard Index calculates the ratio of correctly clustered pairs to the total number of pairs clustered together in at least one of the assignments:

$$\text{JCI} = \frac{TP}{TP+FP+FN}$$

with a value of 1 indicating perfect pairwise agreement and 0 indicating no shared clustering.

In addition to these clustering quality metrics, we also recorded the runtime of each algorithm. While not a measure of clustering accuracy, runtime is critical for assessing the practical scalability of algorithms, especially in high-dimensional or large-scale datasets.

3 Results and Discussion

Dataset	Method	ARI	FMI	JCI	NMI	Runtime (s)
ConcentricCircles	DBSCAN	1.000	1.000	1.000	1.000	0.0036
	Fuzzy	0.0003	0.500	0.333	0.0014	0.109
	GMM	0.0018	0.505	0.338	0.0025	0.222
	KMeans	-0.0004	0.499	0.333	0.0009	0.0108
	Spectral	1.000	1.000	1.000	1.000	70.122
MoonsBlob	DBSCAN	0.940	0.960	0.923	0.897	0.0036
	Fuzzy	0.860	0.907	0.830	0.810	0.0737
	GMM	0.735	0.828	0.717	0.737	0.181
	KMeans	0.819	0.879	0.784	0.768	0.0069
	Spectral	0.842	0.907	0.835	0.815	9.090
Overlapping	DBSCAN	0.0005	0.545	0.320	0.0103	0.0015
	Fuzzy	0.613	0.742	0.590	0.541	0.1585
	GMM	0.572	0.717	0.559	0.521	0.219
	KMeans	0.611	0.740	0.588	0.538	0.0096
	Spectral	0.526	0.700	0.537	0.475	10.006
Spherical	DBSCAN	0.634	0.799	0.654	0.634	0.0013
	Fuzzy	0.986	0.991	0.981	0.975	0.0693
	GMM	0.987	0.991	0.982	0.976	0.163
	KMeans	0.985	0.990	0.980	0.974	0.0052
	Spectral	0.856	0.921	0.862	0.853	9.910

Table 1: Clustering performance on 2D datasets with 3 clusters (mean over 20 runs).

We observed clear and consistent trends in how the clustering algorithms performed across datasets with varying geometry, overlap, and dimensionality. In 2D settings with non-convex clusters—such as moons, slashes, and concentric circles—DBSCAN stood out as the most effective method, as can be seen in Tables 1 and 2. It achieved near-perfect scores on all external metrics (ARI, FMI, JCI, and NMI), particularly in cases where other methods failed to detect the curved or nested structure. Its ability to detect arbitrarily shaped clusters and ignore noise made it especially well-suited for these settings. In addition, it consistently boasted the quickest runtimes, likely making it a good option for situations that were left unexplored in this report, such as very large datasets with many observations.

It should be noted that if eps was not selected correctly—in particular, if it was too small— DBSCAN consistently terminated cluster boundaries early when faced with non-uniformly dense clusters, resulting in a very large number of clusters modeled. We encountered this issue while using the kNN selection method with elbow analysis (Figure 2) for epsilon, and as such we opted to find epsilon by brute-force. The difference in clustering outcomes is depicted in Figures 3 and 4. DBSCAN was also by far the worst performer in the Overlapping clusters dataset; frequently opting to assign most—if not all—points in the dataset to the same group.

Dataset	Method	ARI	FMI	JCI	NMI	Runtime (s)
ConcentricCircles	DBSCAN	0.977	0.983	0.967	0.967	0.0030
	Fuzzy	0.015	0.267	0.154	0.0264	1.200
	GMM	0.093	0.366	0.220	0.158	0.991
	KMeans	0.0024	0.252	0.144	0.0055	0.0229
	Spectral	0.881	0.918	0.868	0.916	81.400
MoonsBlob	DBSCAN	0.948	0.958	0.920	0.930	0.0033
	Fuzzy	0.906	0.925	0.861	0.893	0.293
	GMM	0.941	0.953	0.910	0.926	0.519
	KMeans	0.880	0.904	0.824	0.870	0.0116
	Spectral	0.840	0.885	0.800	0.849	39.300
Overlapping	DBSCAN	0.0005	0.426	0.196	0.0118	0.0022
	Fuzzy	0.672	0.737	0.584	0.663	0.600
	GMM	0.630	0.705	0.545	0.642	0.599
	KMeans	0.676	0.740	0.588	0.667	0.0186
	Spectral	0.600	0.689	0.525	0.615	39.100
Spherical	DBSCAN	0.717	0.801	0.655	0.754	0.0029
	Fuzzy	0.986	0.989	0.978	0.979	0.319
	GMM	0.986	0.989	0.978	0.980	0.539
	KMeans	0.987	0.990	0.979	0.981	0.0162
	Spectral	0.908	0.933	0.877	0.919	40.900

Table 2: Clustering performance on 2D datasets with 5 clusters (mean over 20 runs).

Dataset	Method	ARI	FMI	JCI	NMI	Runtime (s)
MoonsBlob-10D	DBSCAN	0.4742	0.6637	0.4465	0.5689	0.0056
	Fuzzy	0.3053	0.5271	0.3274	0.4826	0.7597
	GMM	0.8165	0.8573	0.7562	0.8561	0.7961
	KMeans	0.4525	0.5856	0.4072	0.5683	0.0293
	Spectral	0.5711	0.6823	0.5106	0.6609	130.093

Table 3: Clustering performance on the 10-dimensional dataset with 5 clusters (mean over 20 runs).

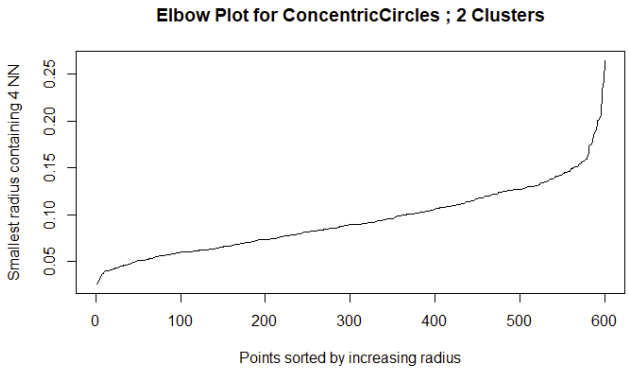

Figure 2: An elbow plot of epsilon values for the Concentric Circles dataset. The elbow suggests that the optimal value of epsilon to be passed into DBSCAN should be roughly 0.15.

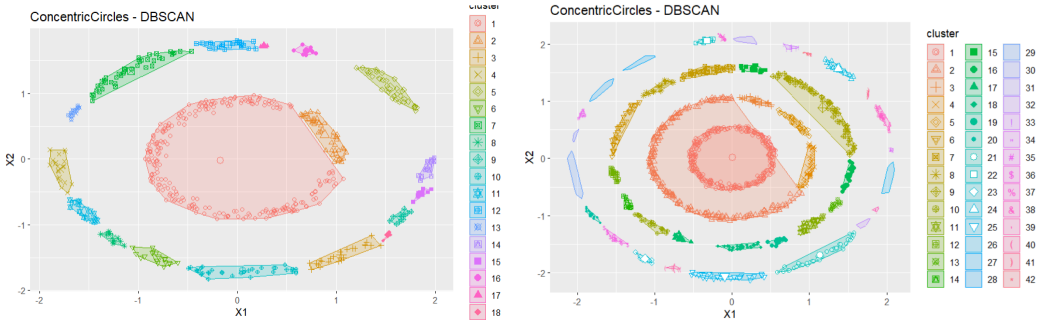

Figure 3: DBSCAN results on the Concentric Circles dataset using the kNN elbow method with `eps` = 0.15. Aside from the inner circles, there are many small clusters wherever there is a break in the data, resulting in a significantly reduced ARI.

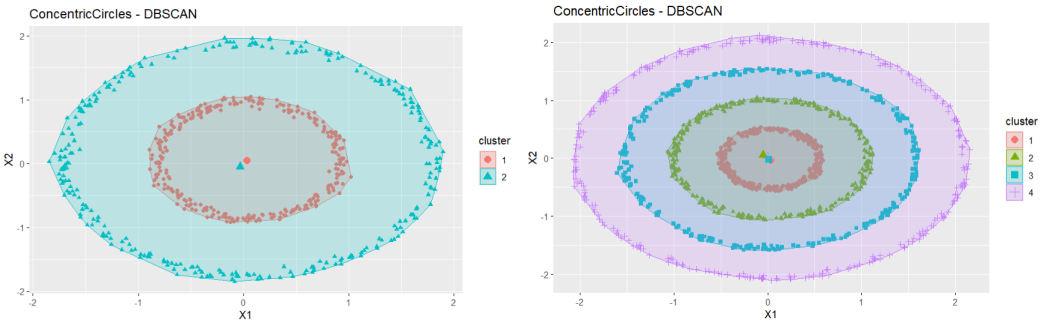

Figure 4: DBSCAN results on the Concentric Circles dataset using `eps` = 0.5, found using brute-force. Clustering is perfect.

Spectral Clustering also performed well on non-convex structures, achieving high FMI and NMI scores, especially on the moons and concentric circles datasets. However, it was significantly more computationally expensive than other models due to the eigen-decomposition step. Runtime between Spectral Clustering and other methods were often orders of magnitude different, and it clearly scales poorly with the number of clusters; runtimes (with the exception of the Concentric Circles dataset) increased roughly by a factor of 4 as clusters went from 3 to 5.

K-Means and GMM were consistently effective on well-separated, convex datasets such as the spherical clusters, where they produced high ARI, FMI, and JCI values. However, they struggled with complex geometries. K-Means was particularly limited by its assumption of equal-sized, convex clusters and performed poorly on datasets like moons and concentric circles. GMM handled overlapping clusters better than K-Means due to its probabilistic nature, but still failed in scenarios with highly non-convex boundaries.

Fuzzy clustering (PDQ) showed mixed results. It performed reasonably well in overlapping datasets, reflecting moderate scores on FMI and JCI, but did not adapt effectively to non-convex cluster shapes. NMI scores for PDQ were

often lower than expected, suggesting it did not capture the full information structure of the clusters in more complex settings.

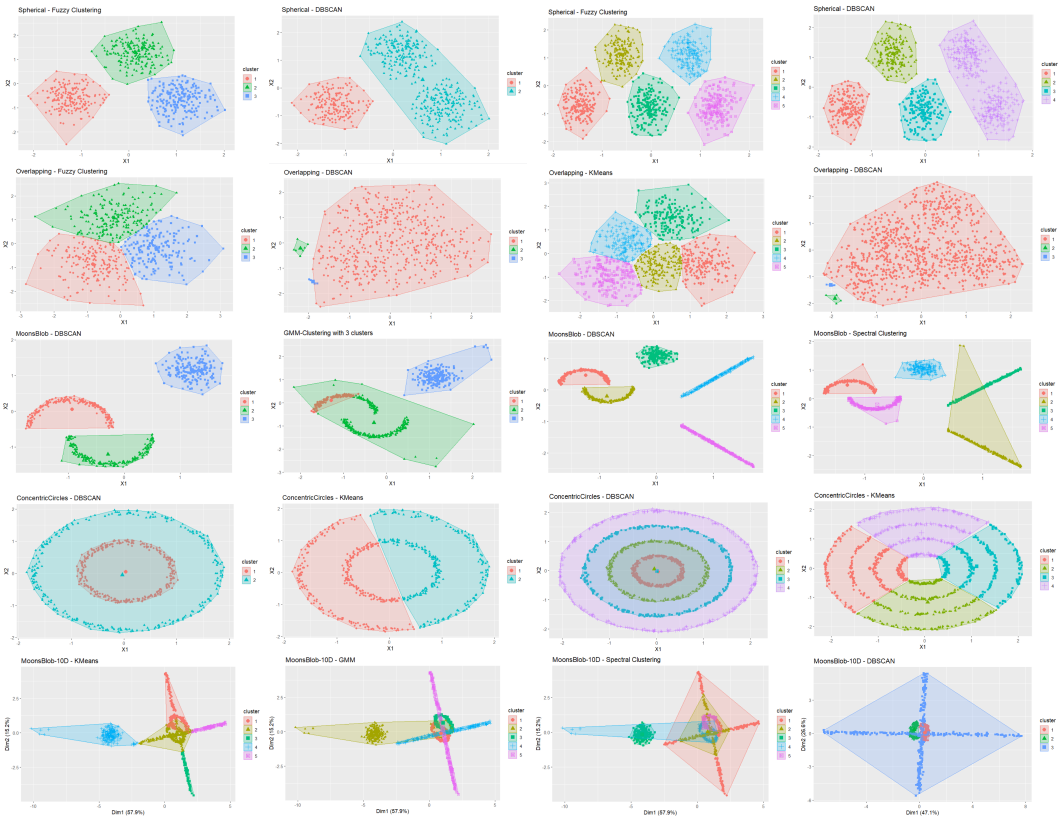

Figure 5: Example plots showing good and poor clustering results for each dataset under varying cluster settings. Last row shows 10D Moons results using K-means, GMM, Spectral, and DBSCAN (Fuzzy is excluded due to similarity with K-means in this setting).

In the 10-dimensional case, GMM emerged as the top performer across all evaluation metrics. This is likely due to the use of multivariate normal distributions in generating many of the clusters, aligning well with GMM's assumptions. Both DBSCAN and Spectral Clustering saw sharp declines in ARI and NMI in this setting, as their reliance on local neighborhoods or similarity matrices becomes less effective in high-dimensional space. DBSCAN, in particular, showed low JCI and FMI scores in 10D, indicating poor pairwise separation of points.

Overall, while ARI provided a robust measure of agreement with true labels, FMI and JCI further highlighted differences in how well methods grouped or separated individual pairs of points. For example, on overlapping clusters, PDQ and GMM achieved reasonably high FMI and JCI scores despite moderate ARI, suggesting they were effective at preserving local pairwise structure even if overall label alignment was imperfect. In contrast, DBSCAN sometimes achieved high ARI but lower JCI in high-dimensional settings, indicating it correctly clustered major groups but made more errors in pairwise point assignments.

NMI, on the other hand, emphasized how well the predicted clustering preserved the overall information content of the true labels. For instance, Spectral Clustering often achieved high NMI even when ARI was lower, indicating that it captured broader structural patterns despite differences in exact label assignments. This was especially noticeable in datasets with soft boundaries or multiple valid clustering interpretations.

4 Conclusion

Our results highlight the importance of understanding the structure and separability of the data prior to applying clustering techniques. DBSCAN and Spectral Clustering consistently outperformed centroid-based methods like K-Means in handling non-convex cluster shapes, but failed to competitively partition both the overlapping and well-separated spherical clusters. GMM proved especially effective in high-dimensional settings, likely due to its alignment with the Gaussian-based data generation. PDQ offered moderate flexibility but was sensitive to initialization and less effective on complex geometries.

Overall, among the algorithms examined in this project, Spectral Clustering was the most consistent in all scenarios. Although it was never the top performer, it was the only method that never fell below an ARI of 0.5, suggesting robustness in varied clustering scenarios. While this result presents as a quality of a reliable, general-purpose model, the practical utility of Spectral Clustering is constrained by its high computational cost relative to the other models.

Ultimately, a clear winner cannot emerge without additional criteria specific to the user and the application. Each algorithm presents distinct strengths and limitations, and the optimal choice depends on which drawbacks can be mitigated and which advantages best align with the task at hand.

5 References

References

Desgraupes, B. (2018). Clustercrit: Clustering indices. r package version 1.2. 8.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *kdd*, *96*(34), 226–231.

Hahsler, M. (2017). Piekenbrock. *M. dbscan: Density Based Clustering of Applications with Noise (DBSCAN) and Related Algorithms*.

Haughton, D., Legrand, P., & Woolford, S. (2009). Review of three latent class cluster analysis packages: Latent gold, polca, and mclust. *The American Statistician*, *63*(1), 81–91.

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of classification*, *2*, 193–218.

Jimeno, J., Roy, M., & Tortora, C. (2021). Clustering mixed-type data: A benchmark study on kamila and k-prototypes. *Data Analysis and Rationality in a Complex World 16*, 83–91.

Karatzoglou, A., Smola, A., Hornik, K., & Zeileis, A. (2004). Kernlab-an s4 package for kernel methods in r. *Journal of statistical software*, *11*(1), 1–20.

Knops, Z. F., Maintz, J. A., Viergever, M. A., & Pluim, J. P. (2006). Normalized mutual information based registration using k-means clustering and shading correction. *Medical image analysis*, *10*(3), 432–439.

McDaid, A. F., Greene, D., & Hurley, N. (2011). Normalized mutual information to evaluate overlapping community finding algorithms. *arXiv preprint arXiv:1110.2515*.

McLachlan, G. J., & Basford, K. E. (1988). *Mixture models: Inference and applications to clustering* (Vol. 38). M. Dekker New York.

Murugesan, N., Cho, I., & Tortora, C. (2021). Benchmarking in cluster analysis: A study on spectral clustering. *Data Analysis and Rationality in a Complex World*, 175.

Na, S., Xumin, L., & Yong, G. (2010). Research on k-means clustering algorithm: An improved k-means clustering algorithm. *2010 Third International Symposium on intelligent information technology and security informatics*, 63–67.

Sander, J., Ester, M., Kriegel, H.-P., & Xu, X. (1998). Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications. *Data Mining and Knowledge Discovery*, *2*(2), 169. https://doi.org/10.1023/A:1009745219419

Schubert, E., Sander, J., Ester, M., Kriegel, H., & Xu, X. (2017). Dbscan revisited, revisited: Why and how you should (still) use dbscan. *ACM Transactions on Database Systems*, *42*, 1–21. https://doi.org/10.1145/3068335

Scrucca, L., Fop, M., Murphy, T. B., & Raftery, A. E. (2016). mclust 5: Clustering, Classification and Density Estimation Using Gaussian Finite Mixture Models. *The R Journal*, *8*(1), 289–317.

Team, R. D. C. (2010). R: A language and environment for statistical computing. *(No Title)*.

Tortora, C., Browne, R. P., ElSherbiny, A., Franczak, B. C., & McNicholas, P. D. (2021). Model-based clustering, classification, and discriminant analysis using the generalized hyperbolic distribution: Mixghd r package. *Journal of Statistical Software*, *98*, 1–24.

Tortora, C., & Palumbo, F. (2025). Fpdclustering: A comprehensive r package for probabilistic distance clustering based methods. *Computational Statistics*, *40*(2), 1123–1146.

Van Mechelen, I., Boulesteix, A.-L., Dangl, R., Dean, N., Hennig, C., Leisch, F., Steinley, D., & Warrens, M. J. (2023). A white paper on good research practices in benchmarking: The case of cluster analysis. *WIREs Data Mining and Knowledge Discovery*, *13*(6). https://doi.org/10.1002/widm.1511

Wang, S., Chen, F., & Fang, J. (2015). Spectral clustering of high-dimensional data via nonnegative matrix factorization. *2015 International Joint Conference on Neural Networks (IJCNN)*, 1–8.

Wang, Z., Da Cunha, C., Ritou, M., & Furet, B. (2019). Comparison of k-means and gmm methods for contextual clustering in hsm. *Procedia Manufacturing*, *28*, 154–159.

6 Appendix

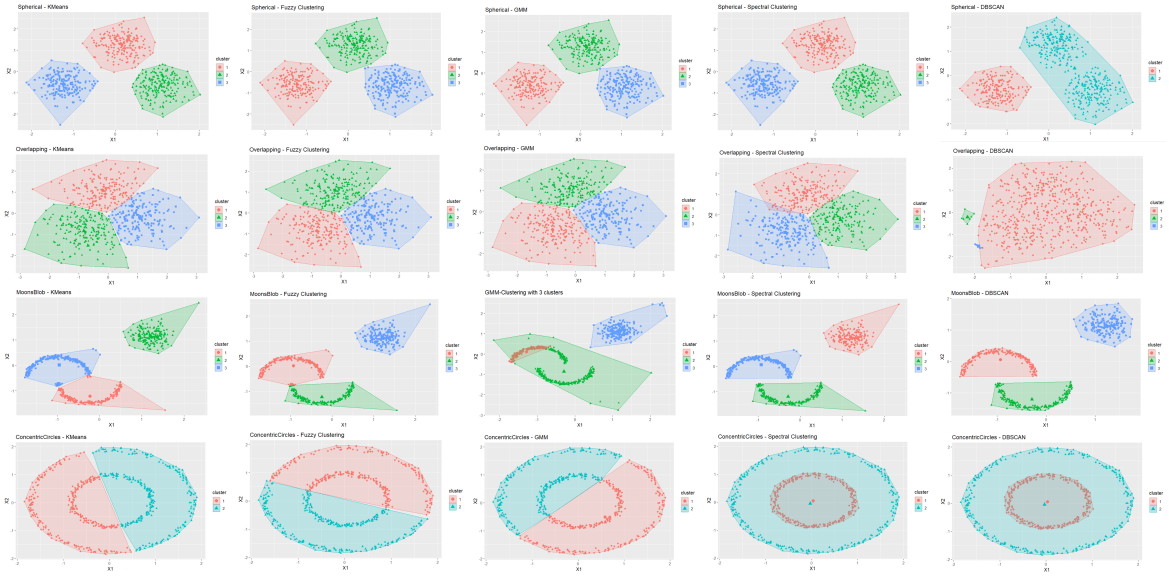

Figure 6: Example clustering results on 2D datasets with 3 clusters. Each row corresponds to a different dataset—spherical, overlapping, moons, and concentric circles. Columns represent the clustering methods used: K-Means, Fuzzy, GMM, Spectral, and DBSCAN (left to right).

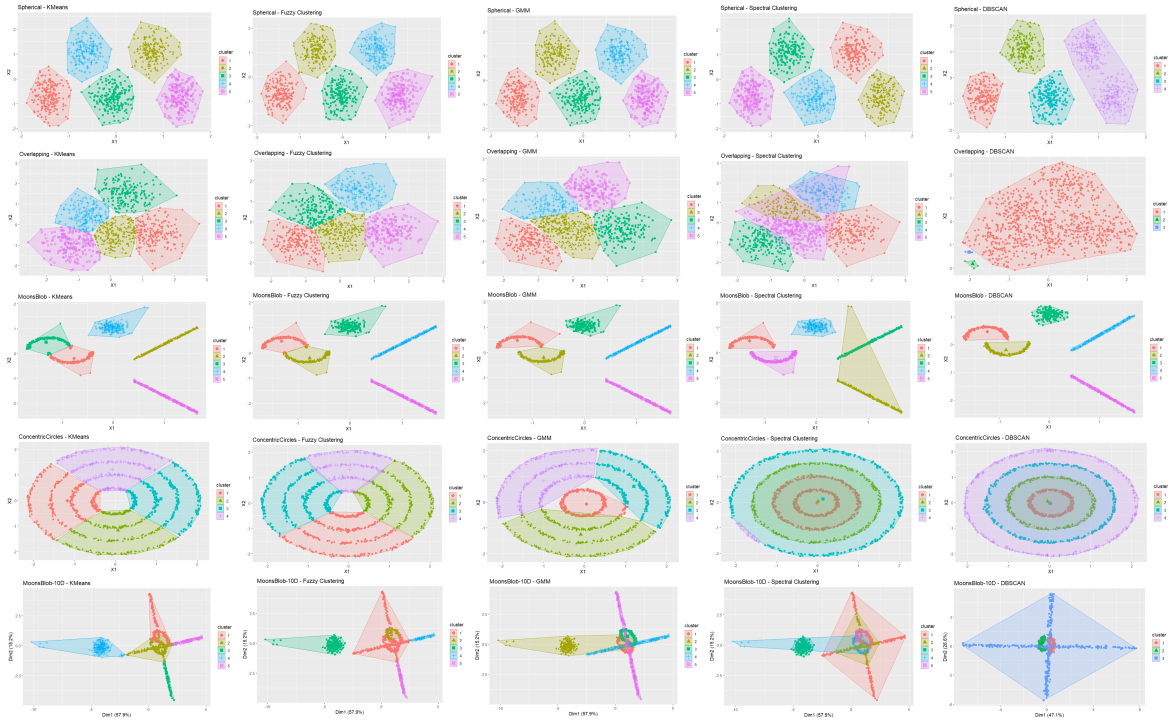

Figure 7: Example clustering results on 2D datasets with 5 clusters and 10D with 5 clusters (last row). Each row corresponds to a different dataset—spherical, overlapping, moons, and concentric circles. Columns represent the clustering methods used: K-Means, Fuzzy, GMM, Spectral, and DBSCAN (left to right).