Automated Density-Based Splitting of Merged Clusters

1st Md. Mahir Uddin
ID: 0122430022
Computer Science and Engineering
United International University
Dhaka, Bangladesh

***Abstract*—K-means clustering, a widely applied unsupervised learning algorithm, is directly affected by the user-defined number of clusters value (K) which leads to suboptimal results when incorrectly estimated. This paper introduces a novel density-based recursive splitting mechanism to automatically detect clusters from the dataset. Our method initially assumes the entire dataset a single cluster and iteratively refines clusters by identifying low-density regions, pinpointing potential splits, and recursively applying k-means to further partition them. To find low-density regions, we examined three different approaches. This algorithm demonstrates robustness to irregularly shaped data and density-varying clusters, thereby eliminating the need to prior knowledge of the ideal cluster count. Experimental validation on real-world datasets show that our method provides accurate and adaptive cluster assignments. This makes the proposed method particularly valuable for applications where true cluster structure is unknown and dynamic adjustments are critical.**

***Index Terms*—K-means clustering, density-based clustering, recursive clustering, unsupervised learning.**

I. Introduction

Clustering is a fundamental technique in unsupervised learning, widely used in various domains such as image processing, bioinformatics, anomaly detection, and market segmentation. Among clustering methods, k-means is one of the most widely adopted due to its simplicity, computational efficiency, and scalability [1]. The algorithm partitions a dataset into a predefined number of clusters, where each point is assigned to the nearest cluster centroid. Through an iterative refinement process, k-means minimizes intra-cluster variance, producing compact and well-separated clusters in many scenarios.

Despite its widespread use, k-means clustering has several inherent limitations that can impact its effectiveness, particularly in complex real-world datasets. A major drawback of k-means is its dependence on a predefined number of clusters (k), which must be specified before execution [2]. Selecting an inappropriate k value can lead to suboptimal clustering, where distinct clusters are merged or a single cluster is unnecessarily split [3]. Heuristic methods such as the Elbow Method [4] and Silhouette Score [5] attempt to estimate the optimal k, but these approaches do not guarantee the best possible clustering for all datasets. Furthermore, k-means assumes that clusters are convex and roughly spherical [6], making it ineffective for datasets containing irregularly shaped or density-varying clusters.

Another major challenge of k-means is its sensitivity to centroid initialization. Poorly initialized centroids can result in convergence to local optima, leading to unstable clustering results [7]. While initialization techniques such as k-means++ [8] mitigate this issue to some extent, they do not fully resolve the problem. Additionally, k-means is susceptible to outliers, which can disproportionately influence centroid positions and distort cluster assignments [9]. More critically, k-means lacks an inherent mechanism to detect when two distinct clusters have been mistakenly merged into one due to centroid placement or density distribution [10].

A critical unresolved challenge in k-means clustering is its inability to detect and correct clusters that are erroneously merged due to overlapping densities or initialization artifacts. To address this, we propose a density-aware splitting mechanism that enhances k-means by automatically identifying merged clusters and recursively partitioning them into sub-clusters without requiring additional user-defined parameters.

Our method operates as follows: the entired dataset is assumed to be a single cluster at the beginning, gradually each cluster is analyzed by dividing its bounding box into a grid of squares, where the grid resolution and density thresholds are derived dynamically from the cluster's spatial distribution. Low-density regions are mapped as "chains" spanning the cluster, indicating potential splits. If such chains are detected, the cluster is subdivided using k-means with k=2, and the process repeats recursively until no further splits are warranted.

The key advantages of our approach are threefold. First, it eliminates manual parameter tuning by inferring thresholds directly from data. Second, it enables the detection of non-spherical or density-varying clusters. Third, it operates as a post hoc refinement layer, making it compatible with existing k-means variants (e.g., k-means++). By bridging the gap between parametric efficiency and density-based robustness, our method extends k-means' applicability to complex real-world datasets.

II. LITERATURE REVIEW

Researchers have long sought to address the limitations of k-means, particularly its dependence on predefined cluster counts and its inability to handle complex cluster geometries. Early approaches focused on improving initialization robustness. For instance, Arthur and Vassilvitskii introduced k-means++ (2007), which probabilistically seeds initial centroids to reduce sensitivity to poor initialization, though it still requires an explicit k value [11]. To automate cluster count selection, heuristic methods like the Elbow Method [12] and Silhouette Analysis [13] were proposed. These techniques iteratively evaluate clustering quality across candidate k values but struggle with datasets where clusters vary significantly in density or shape.

For non-convex or density-varying clusters, density-based methods like DBSCAN (Ester et al., 1996) and OPTICS (Ankerst et al., 1999) emerged as alternatives. These algorithms identify clusters based on local point density, enabling detection of arbitrarily shaped structures. However, they lack the computational efficiency of k-means and often require tuning density thresholds, limiting their scalability [14],[15].

Hierarchical extensions to k-means, such as bisecting k-means (Steinbach et al., 2000), recursively partition clusters to refine results. While this approach mitigates initialization sensitivity, it inherits k-means' bias toward spherical clusters and struggles with overlapping distributions [16]. More recently, Gaussian Mixture Models (GMMs) (Reynolds, 2009) and spectral clustering (Ng et al., 2002) have been used to model complex cluster geometries. However, GMMs assume parametric distributions, and spectral clustering's reliance on graph Laplacians introduces computational overhead [17],[18].

Efforts to dynamically determine k include x-means (Pelleg and Moore, 2000), which uses Bayesian Information Criterion (BIC) to split clusters during runtime, and G-means (Hamerly and Elkan, 2003), which employs statistical tests to validate cluster normality. While these methods reduce dependency on predefined k, they often fail to detect merged clusters caused by subtle density variations or irregular boundaries [19],[20]. Similarly, dip-means (Kalogeratos and Likas, 2012) uses multimodality tests to split clusters but requires user-defined significance thresholds [21].

Recent advances in deep learning, such as deep embedded clustering (DEC) (Xie et al., 2016), combine autoencoders with k-means to learn latent representations for improved clustering. While DEC achieves state-of-the-art results on high-dimensional data, it introduces complexity and training instability, limiting its applicability to resource-constrained environments [22].

Despite these advancements, no method fully resolves the challenge of post hoc detection and correction of merged clusters in k-means. Most approaches either require prior assumptions about cluster geometry, sacrifice computational efficiency, or lack interpretability. This gap motivates our work.

III. METHODOLOGY

The proposed method refines traditional k-means clustering by iteratively detecting and splitting clusters using density-based analysis. Unlike conventional k-means, which relies on a predefined number of clusters, this approach dynamically adjusts k by analyzing the density distribution within clusters. Through a recursive process, the method systematically identifies and separates incorrectly merged clusters without requiring user-defined parameters. The complete workflow consists of five key stages, detailed below.

A. Datasets

We used multiple dataset to check the validity of our proposed algorithm. We used both synthetic and real-world datasets.

- Make blobs dataset: we generated a synthetic two dimensional dataset using the make_blobs function from Scikit-learn. The dataset consists of 5000 data points distributed across 8 clusters, with each cluster following a Gaussian distribution. The standard deviation of clusters was set to 0.60, ensuring moderate overlap between clusters. A fixed random seed (random_state=46) was used to maintain reproducibility.
- Mall customer segmentation dataset: It's a real-world dataset. It contains 5 feature columns(customer id, age, gender, annual income and spending score). Among them 4 are numeric columns. After performing EDA, from the pairplot we go to know that annual income and spending score gives clear clusters. We used the dataframe with these two features to fit and test our proposed algorithm. Applied feature scaling before training the model.
- Penguin dataset: The dataset contains various attributes and measurements of three types of penguins. It has 17 columns in total. Among them 7 are numeric columns. There are some columns containing NaN values. We replaced the NaN values of numeric columns with media value of the particular column. Then performed standard scaler to bring the data in the similar scale. We used this dataset to check whether our model can find the clusters correctly in 3d space.

B. Outlier Removal and Bounding Box Construction

To focus on the core structure of each cluster, outliers and edge points are removed before further analysis. Outliers and edge points are defined as points that deviate significantly from the cluster center. Specifically, any point located beyond 1.5σ from the cluster centroid—where σ is the standard deviation of distances from the centroid—is considered an outlier or edge points and discarded.

Once outliers are removed, a bounding box is constructed around the remaining cluster points. This bounding box is defined by the minimum and maximum coordinates in both dimensions:

(x_min, y_min), (x_min, y_max), (x_max, y_min), (x_max ,y_max)

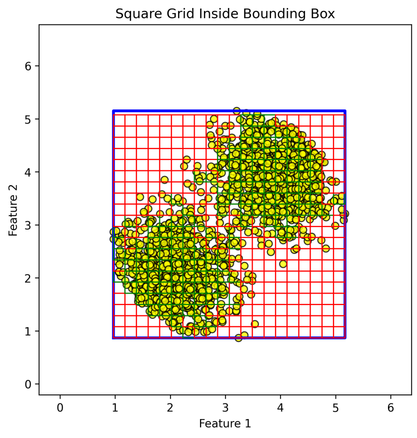

Fig. 1. Bounding Box

C. Density Matrix Generation

To analyze the internal structure of each cluster, we first generate a density matrix. If the data has more than two dimensions, PCA is applied to project the cluster data into 2D for visualization and spatial analysis. Then, a bounding box is created around the projected cluster, which is divided into a 10×10 grid, resulting in 100 equal-sized square regions. Each square is assigned a density value based on how populated it is, giving us a coarse-grained view of how points are distributed across the cluster.

1) Count based approach: In this method, the density of each square is calculated by simply counting the number of data points that fall within it. The initial value assigned to each square directly reflects this count. To determine whether a square is considered low-density, a threshold is calculated using the following formula: Points Box Area of One SquareThreshold=0.5×(Bounding Box AreaTotal Points)×Area of One Square Here, 50% of the average density is used as the threshold. Where count is less than this threshold, that region is considered as low density and high density otherwise. This parameter can be adjusted to control the sensitivity of the approach.
2) KNN-Based Approach: In this method, density estimation is carried out using a k-nearest neighbors (KNN) approach. The number of neighbors, k, is selected as the square root of the number of data points in the cluster. For each small square within the 10×10 grid, the points located inside it are identified, and for each point, the distance to its k-th nearest neighbor is computed. The density of a point is then defined as the inverse of this distance. The total density of a square is obtained by summing the densities of all points it contains. Finally, a threshold is computed as the average density across all squares. Squares with density below this threshold are assigned a value of 0 (indicating low density), while others are assigned 1 (high density).
3) KDE-Based Approach: In this method, Kernel Density Estimation (KDE) is employed to estimate the point density across the 2D grid. For each of the 10×10 squares within the bounding box, the density is computed using a Gaussian kernel applied over the data points in the cluster. The resulting density values are then normalized to the range [0, 1] to ensure consistency across different clusters. A fixed threshold of 0.25 is applied in this experiment, where squares with normalized density below the threshold are classified as low-density (assigned 0), and others as high-density (assigned 1). This threshold is a tunable parameter that controls the sensitivity of the approach, allowing adaptability based on the density characteristics of the dataset.

D. Low-Density Chain or Plane Detection

Since edge artifacts can influence density calculations, the outermost 15% of rows and columns are trimmed from the density matrix. This ensures that the analysis focuses on the core region of the cluster, reducing boundary effects.

The remaining density matrix is then analyzed for low-density chains—continuous sequences of connected low-density squares (0s) that indicate potential cluster boundaries. These chains are detected using 8-directional breadth-first search (BFS) to find connected low-density paths. A cluster is flagged for splitting if a low-density chain satisfies either of the following conditions:

- Top-to-bottom chain: A continuous low-density path spans from the top edge to the bottom edge of the trimmed density matrix.
- Left-to-right chain: A continuous low-density path spans from the left edge to the right edge of the trimmed density matrix.

The presence of such chains suggests that the cluster actually contains multiple sub-clusters, requiring further subdivision.

E. Recursive Cluster Splitting

If a low-density chain is detected, the cluster undergoes further subdivision using k-means with k=2. The original cluster is split into two sub-clusters, each with its own centroid and assigned data points. The two resulting sub-clusters replace the original cluster, and both are independently analyzed from Step B onward.

This process—outlier removal, density matrix generation, and low-density chain detection—recursively repeats for each newly formed sub-cluster. If a sub-cluster is found to contain additional low-density chains, it is further subdivided. This iterative refinement continues until no further splits are necessary, meaning all clusters are free of incorrectly merged sub-clusters.

IV. RESULTS AND DISCUSSION

This section presents the clustering results obtained using three different density matrix approaches across four datasets: one synthetic and one real-world dataset in 2D, and one synthetic and one real-world dataset in 3D. PCA was applied prior to clustering for 3D data. The results are visualized and discussed for each dataset separately, followed by a comparison of model performance across different data types, highlighting each approach's strengths and limitations.

We begin with the synthetic 2D dataset, which was generated to contain clearly separable clusters with uniform density. The original distribution of the data is visualized below, followed by the clustering results obtained using each of the three density matrix approaches(Fig.2).

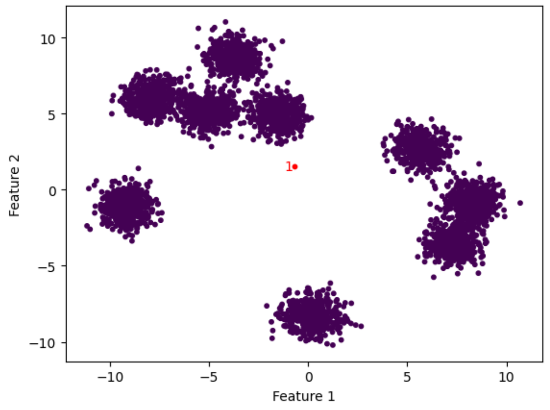

Fig. 2. Make Blobs Dataset Without Clustering

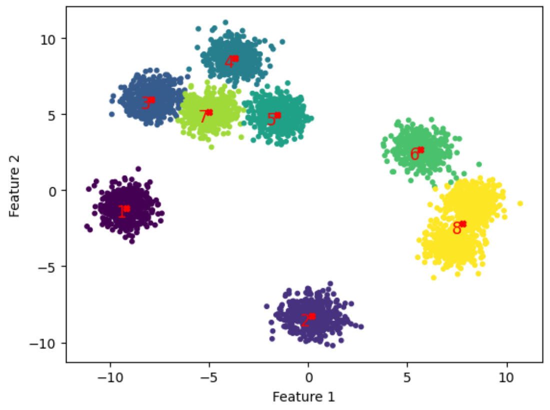

Fig. 3. Make Blobs Dataset Using KMeans KDE

Among the three approaches, the KDE-based method produced the most accurate result, correctly identifying all five clusters with minimal over- or undersplitting. Its strength lies in estimating density by accounting for surrounding points, which helps reduce false classification of sparse cells as low-density and leads to more stable cluster boundaries. However, in very sparse regions, it may occasionally undersplit due to smooth density estimation. The KNN-based approach also considers neighboring density and performs better than the count-based method, but it still showed both oversplitting and undersplitting in some clusters due to its sensitivity to local variations. The count-based approach performed the worst, as it treats each cell in isolation. In sparse datasets, many grid

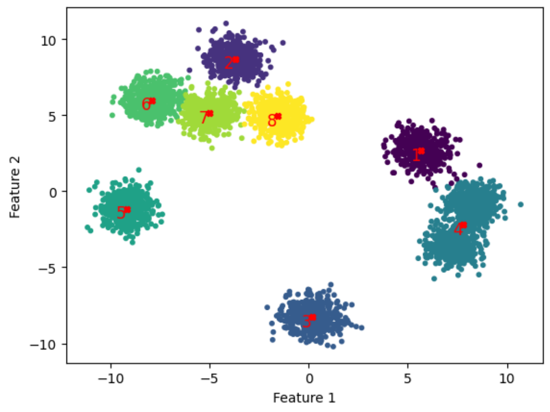

Fig. 4. Make Blobs Dataset Using KMeans KNN

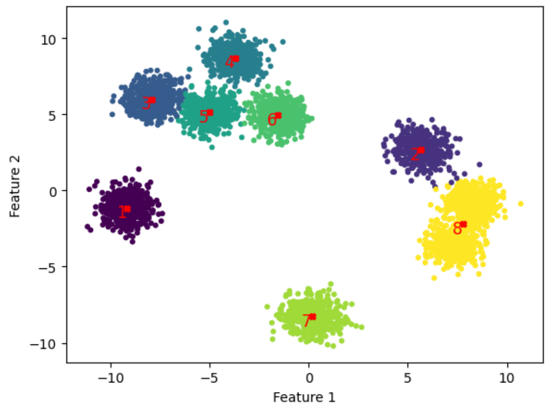

Fig. 5. Make Blobs Dataset Using KMeans Count

cells contain only 0 or 1 points and are misclassified as low-density, leading to repeated and inaccurate splitting. Its inability to consider nearby density makes it prone to oversplitting, especially as clusters get smaller. Then, It continues to split until a stopping criteria is met (Fig.3, Fig.4, Fig.5).

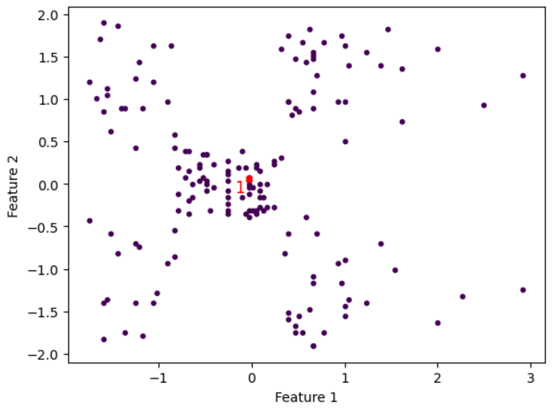

Fig. 6. Mall Customer Dataset Without Clustering

All three approaches produced identical clustering results on the 2D real dataset, which contains well-separated and

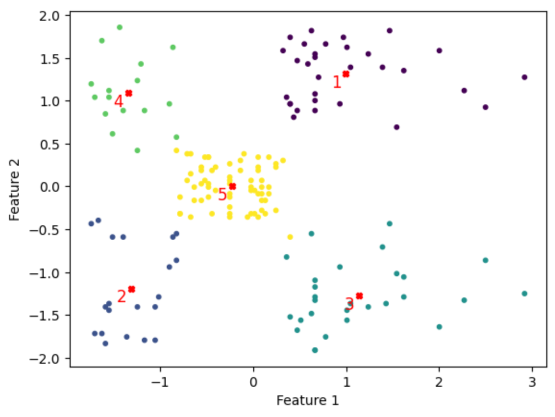

Fig. 7. Mall Customer Dataset Using KMeans KDE

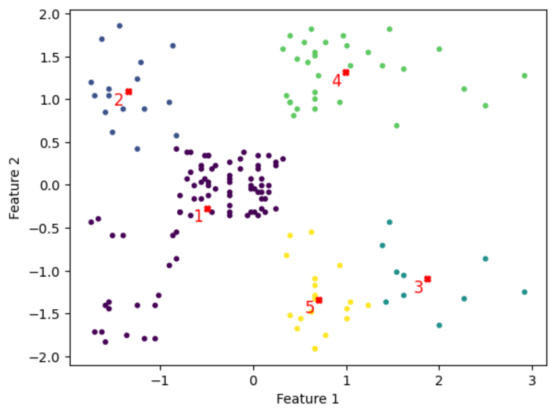

Fig. 8. Mall Customer Dataset Using KMeans KNN

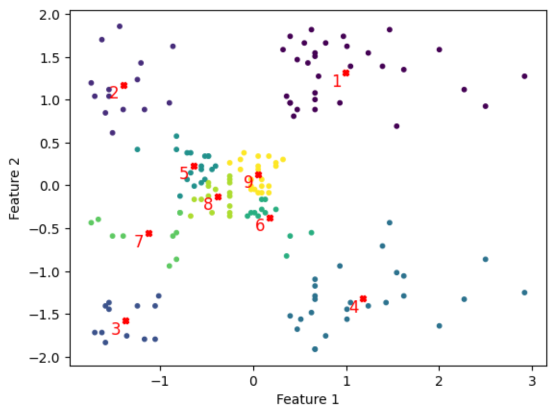

Fig. 9. Mall Customer Dataset Using KMeans Count

densely packed clusters. Due to minimal overlap and high concentration of data points, the clustering algorithms performed effectively, as confirmed visually. In this high concentration scenario, the count-based approach performed well since the higher density minimized its tendency to misclassify sparse cells, and clear low-density regions between clusters were correctly identified. The KDE-based method, while generally robust, may occasionally fail to split across very narrow gaps, as surrounding points smooth out local density variations, leading to possible undersplitting but doesn't do oversplitting like count based. The KNN-based approach, though effective, sometimes oversplit dense clusters due to local sensitivity. Overall, the KDE-based method continued to show strong performance on larger datasets, indicating better adaptability across data types (Fig.6, Fig.7, Fig-8, Fig-9).

We now present the results for the 3D synthetic and real-world datasets, along with their interpretations.

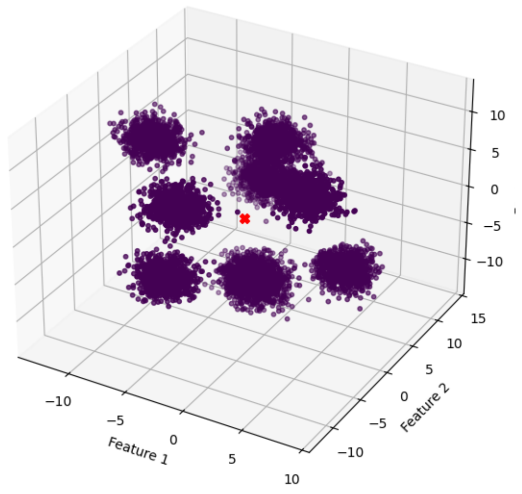

Fig. 10. Make Blobs Dataset Without Clustering

For the 3D synthetic dataset, generated using the make_blobs function with clearly separated and highly concentrated clusters, all three approaches produced identical clustering results. The clusters were easily distinguishable based on density, and the models showed similar behavior to their performance on the 2D synthetic dataset. Although the density matrix was computed after applying PCA to reduce dimensionality, the outcome remained consistent. As in the 2D case, the count-based method performes well due to the high concentration and clear boundaries, while the KDE and KNN-based approaches showes stable results without significant under- or oversplitting. This confirms that all three methods are effective in handling clean, well-structured datasets with less overlapping(Fig.10, Fig.11, Fig.12, Fig.13).

In the 3D real-world dataset, the true number of clusters is three. Among the three approaches, only the KDE-based method correctly identified all three clusters, while both the KNN-based and count-based methods produced only two

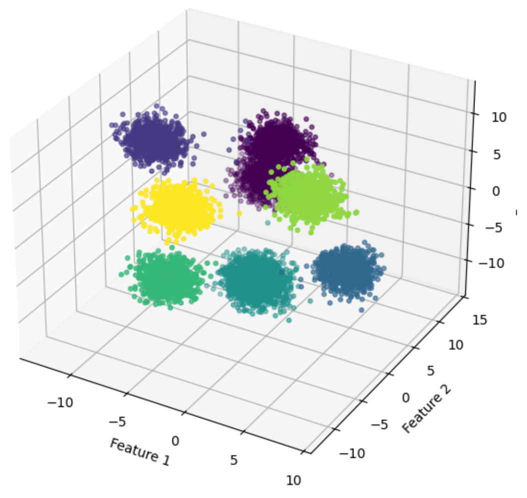

Fig. 11. Make Blobs Dataset Using KMeans KDE

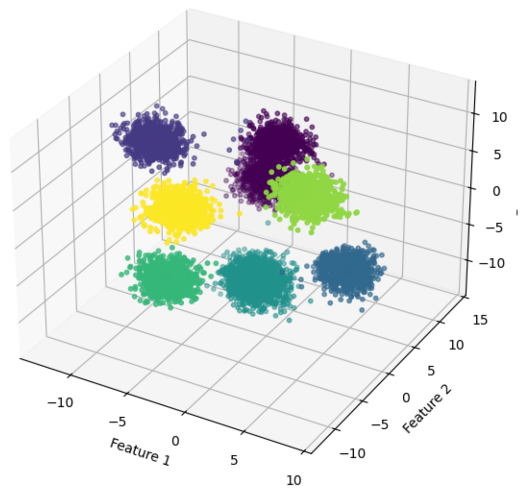

Fig. 13. Make Blobs Dataset Using KMeans Count

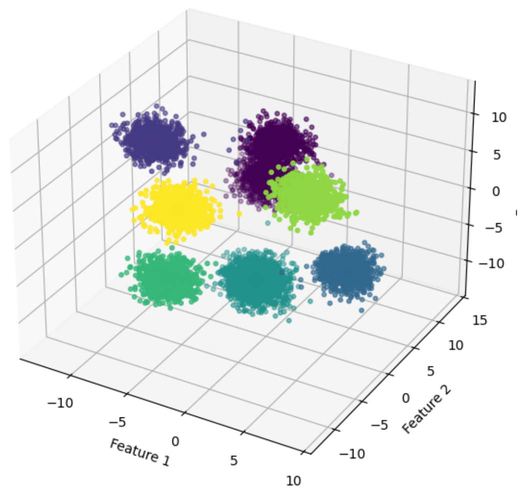

Fig. 12. Make Blobs Dataset Using KMeans KNN

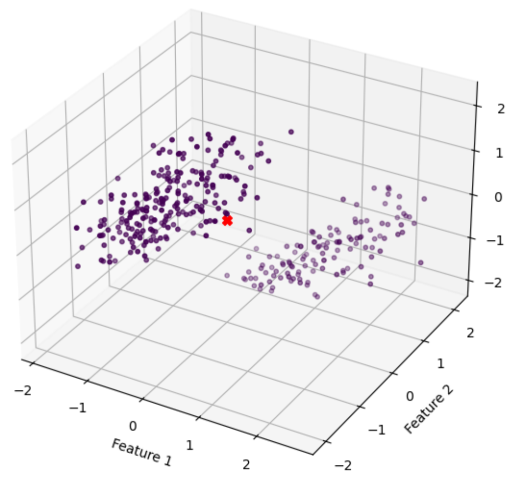

Fig. 14. Penguin Dataset Without Clustering

clusters. This indicates that the KDE-based approach handled the low-density and sparse nature of the real data more effectively. Its ability to incorporate surrounding point influence in density estimation allowed for more accurate detection of subtle cluster boundaries. In contrast, the count-based method misclassified sparse regions due to its limited sensitivity to context, and the KNN-based method failed to resolve the weaker density transitions. Consistent with the results from the 2D real dataset, the KDE-based method again demonstrated its ability in handling sparse, real-world data(Fig.14, Fig.15, Fig.16, Fig.17).

Across all datasets, the KDE-based density matrix approach consistently delivered the most reliable clustering results. It accurately identified the correct number of clusters in both sparse and dense settings, in both 2D and 3D, demonstrating strong adaptability and robustness. The KNN-based approach showed moderate performance, performing reasonably well on dense data but struggling with oversplitting or undersplitting in more complex, sparse scenarios. The count-based approach performed acceptably on highly concentrated datasets with clearly defined boundaries but failed in sparse or real-world data due to its lack of contextual awareness, often leading to overclustering. Overall, while all methods performed similarly on clean, synthetic data with well-separated clusters, the KDE-based approach stood out as the most versatile and reliable across varying data complexities and dimensionalities.

V. LIMITATIONS AND FUTURE WORK

Computation and time are still huge constraints in our model. Overall our model performs well when it deals with dense datasets. It's performance needs sufficient improvement in case of sparse datasets. We performed PCA to deal with

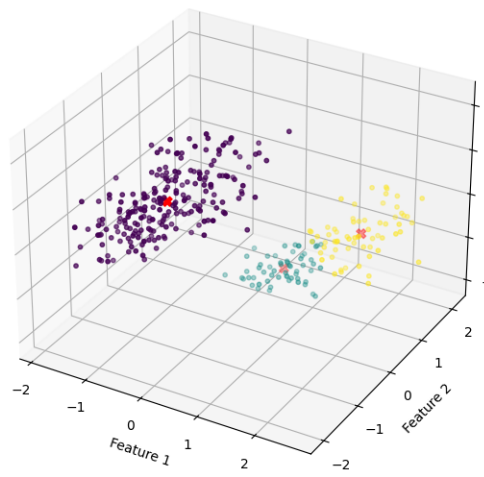

Fig. 15. Penguin Dataset Using KMeans KDE

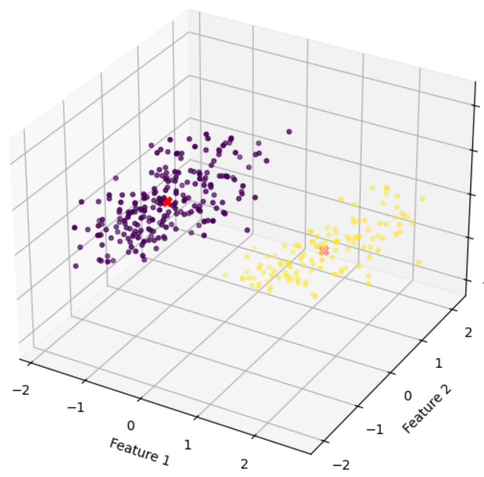

Fig. 17. Penguin Dataset Using KMeans Count

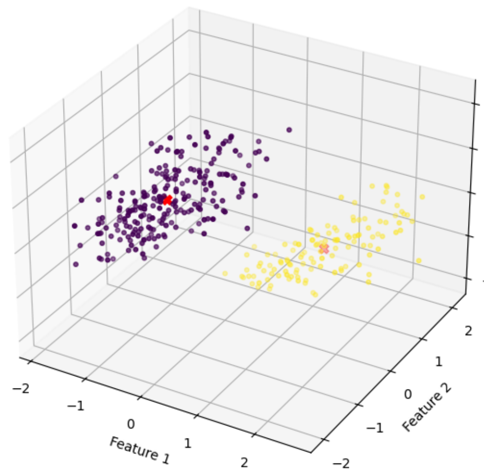

Fig. 16. Penguin Dataset Using KMeans KNN

higher dimensional datasets. However, our target was to find out plane or sphere containing 0s for splitting, which we couldn't achieve yet. It made our model less robust for higher dimensional datasets.

Future work will address these challenges by optimizing the algorithm to reduce computational overhead, perform clustering appropriately for both sparse and dense datasets and create a robust approach to handle higher dimensional datasets.

VI. Conclusion

This study successfully introduced a novel density-based recursive splitting mechanism, effectively addressing the K-means limitation of pre-defined cluster numbers by automatically identifying and partitioning data based on low-density regions. Among the three density matrix approaches explored, the KDE-based method consistently demonstrated superior performance, particularly in accurately detecting clusters in diverse, real-world, and sparse datasets across 2D and 3D. While robust to unknown cluster structures and density variations, the current model faces limitations in computational efficiency and struggles with highly sparse data and complex higher-dimensional geometries. Future work will focus on optimizing these aspects to further enhance the model's versatility and robustness.

VII. References

References

[1] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics* (Vol. 5, pp. 281–298). University of California press.

[2] Jain, A. K. (2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, *31*(8), 651–666. Elsevier.

[3] Reddy, C. K. (2018). *Data clustering: algorithms and applications*. Chapman and Hall/CRC.

[4] Thorndike, R. L. (1953). Who belongs in the family?. *Psychometrika*, *18*(4), 267–276. Springer-Verlag.

[5] Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, *20*, 53–65. Elsevier.

[6] Celebi, M. E., Kingravi, H. A., & Vela, P. A. (2013). A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*, *40*(1), 200–210. Elsevier.

[7] Fränti, P., & Sieranoja, S. (2018). K-means properties on six clustering benchmark datasets. *Applied Intelligence*, *48*, 4743–4759. Springer.

[8] Arthur, D., & Vassilvitskii, S. (2006). *k-means++: The advantages of careful seeding*. Stanford. (Technical Report).

[9] Kriegel, H.-P., Kröger, P., Schubert, E., & Zimek, A. (2012). Outlier detection in arbitrarily oriented subspaces. In *2012 IEEE 12th International Conference on Data Mining* (pp. 379–388). IEEE.

[10] Hamerly, G., & Elkan, C. (2003). Learning the k in k-means. *Advances in Neural Information Processing Systems*, *16*.

[11] Arthur, D., & Vassilvitskii, S. (2007). k-means++: The Advantages of Careful Seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

[12] Thorndike, R. L. (1953). Who Belongs in the Family?. *Psychometrika*, *18*(4), 267-276.

[13] Rousseeuw, P. J. (1987). Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Journal of Computational and Applied Mathematics*, *20*, 53-65.

[14] Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD)*.

[15] Ankerst, M., Breunig, M. M., Kriegel, H. P., & Sander, J. (1999). OPTICS: Ordering Points to Identify the Clustering Structure. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data (SIGMOD)*.

[16] Steinbach, M., Karypis, G., & Kumar, V. (2000). A Comparison of Document Clustering Techniques. In *Proceedings of the KDD Workshop on Text Mining*.

[17] Reynolds, D. A. (2009). Gaussian Mixture Models. In S. Z. Li & A. Jain (Eds.), *Encyclopedia of Biometrics* (pp. 523-528). Springer.

[18] Ng, A. Y., Jordan, M. I., & Weiss, Y. (2002). On Spectral Clustering: Analysis and an Algorithm. In *Advances in Neural Information Processing Systems*, *15* (NIPS 2002).

[19] Pelleg, D., & Moore, A. W. (2000). X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*.

[20] Hamerly, G., & Elkan, C. (2003). Learning the k in k-means. *Advances in Neural Information Processing Systems*, *16*.

[21] Kalogeratos, A., & Likas, A. (2012). Dip-means: An Incremental Clustering Method for Estimating the Number of Clusters. In *Advances in Neural Information Processing Systems*, *25* (NIPS 2012).

[22] Xie, J., Girshick, R., & Farhadi, A. (2016). Unsupervised Deep Embedding for Clustering Analysis. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*.