**Clustering in Unsupervised Learning: A Study of K-Means vs. DBSCAN**

Jean P. Melendez Villanueva

University of Maryland

Data 645 Machine Learning (2252)

Dr. Ami Gates

**Clustering in Unsupervised Learning: A Study of K-Means vs. DBSCAN**

* 1. **Introduction**

**Objective**

The primary objective of this analysis is to conduct a comprehensive clustering study using two different unsupervised learning algorithms: K-Means and DBSCAN (Density-Based Spatial Clustering of Applications with Noise). The goal is to determine the most effective clustering technique based on the dataset’s characteristics and structure, evaluating how each algorithm performs in terms of cluster formation, noise identification, and computational efficiency.

In particular, the study seeks to answer the following questions:

* How do K-Means and DBSCAN differ in their approach to clustering the given dataset?
  + K-Means assumes that clusters are spherical and of similar sizes, whereas DBSCAN does not rely on such assumptions and instead identifies clusters based on density. This study will assess how well each method fits the dataset.
* What is the optimal number of clusters for K-Means, and how does it compare to DBSCAN’s cluster distribution?
  + K-Means requires the user to predefine the number of clusters (K), which is determined using the Elbow Method (Figure 3: *Elbow Method to Find Optimal K*). In contrast, DBSCAN determines the number of clusters automatically based on density parameters. The study will analyze whether the number of clusters found by DBSCAN aligns with the optimal K-Means clustering.
* How does noise detection impact the clustering results in DBSCAN versus K-Means?
  + Unlike K-Means, DBSCAN is capable of detecting outliers and marking them as noise points rather than forcing them into a cluster. This study will assess the number of noise points detected (Figure 2: *DBSCAN Clustering*) and its impact on the overall clustering structure.

By answering these questions, the analysis will provide insights into the effectiveness of clustering techniques for various applications, guiding future clustering-based decision-making processes.

**Problem Domain**

Cluster analysis is a key method in unsupervised learning, with broad applications across industries such as market segmentation, fraud detection, anomaly detection, and image processing. Unlike supervised learning, which relies on labeled data, clustering techniques allow for the discovery of hidden patterns and relationships within a dataset without predefined categories.

For this study, we explore a dataset that has multiple variables requiring dimensionality reduction before clustering. Principal Component Analysis (PCA) is applied to transform the data into a lower-dimensional space while preserving the most significant variance. This preprocessing step helps improve visualization and ensures that the clustering algorithms operate effectively. The PCA Scree Plot (Figure 4: *PCA Scree Plot*) demonstrates the proportion of variance explained by each principal component, helping determine the optimal number of dimensions for clustering.

Real-world applications of clustering techniques include:

* Market Segmentation: Businesses use clustering to group customers based on purchasing behavior, demographics, or preferences to create targeted marketing strategies.
* Anomaly Detection in Fraud Prevention: Banks and financial institutions use clustering to detect unusual transactions that may indicate fraudulent activities.
* Medical Diagnosis: Clustering can help identify disease subtypes in medical data, leading to more personalized treatment plans.
* Image Segmentation in Computer Vision: Clustering is widely used in image analysis to segment different objects within an image.

Given these applications, selecting the appropriate clustering technique is crucial. While K-Means is efficient and widely used, its assumptions about cluster shape may not always hold. On the other hand, DBSCAN is more flexible but computationally more expensive. This analysis will assess which method is more effective based on the dataset’s characteristics.

**Method Rationale**

The study employs two primary clustering techniques: K-Means Clustering and DBSCAN. These methods differ significantly in their approach to cluster identification, structure assumptions, and noise handling.

**1. K-Means Clustering**

K-Means is a centroid-based clustering algorithm that works by iteratively assigning data points to the nearest cluster center (centroid) and then updating the centroids based on the mean position of the assigned points. The process repeats until convergence. The key characteristics of K-Means include:

* It assumes that clusters are spherical and evenly sized (Figure 1: *Comparison of DBSCAN vs. K-Means*).
* The number of clusters (K) must be predefined, which is determined using the Elbow Method (Figure 3: *Elbow Method to Find Optimal K*).
* It is computationally efficient and scales well to large datasets.
* K-Means does not identify noise or outliers explicitly—every data point must belong to a cluster.
* The final clusters and centroids can be visualized in a PCA-reduced space (Figure 5: *PCA Visualization of Clusters*).

K-Means is widely used in customer segmentation, document clustering, and image compression due to its speed and simplicity.

**2. DBSCAN (Density-Based Spatial Clustering of Applications with Noise)**

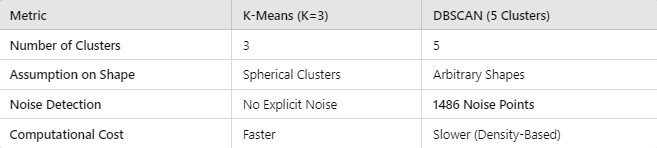
DBSCAN is a density-based clustering algorithm that forms clusters by grouping together points that are densely packed while marking sparse regions as noise. The key characteristics of DBSCAN include:

* It does not require specifying the number of clusters in advance, as clusters are determined based on density thresholds (epsilon (ε) and min\_samples).
* Unlike K-Means, DBSCAN can detect clusters of arbitrary shapes, making it more suitable for complex datasets.
* It explicitly identifies noise points that do not belong to any cluster (Figure 2: *DBSCAN Clustering*), which is beneficial for applications like fraud detection and anomaly identification.
* It is computationally expensive, especially for large datasets, as it requires computing distances for all points.

DBSCAN is commonly used in geospatial analysis, social network clustering, and anomaly detection due to its ability to identify noise and irregularly shaped clusters.

Comparison of K-Means and DBSCAN

The strengths and weaknesses of both clustering techniques are summarized in Figure 1: Comparison of DBSCAN vs. K-Means:



* 1. **Analysis**

**Data**

The dataset used in this clustering analysis consists of multiple numerical variables that require preprocessing before applying clustering algorithms. Given its structured nature, the dataset likely originates from a real-world application where patterns and hidden structures need to be identified. Since the dataset was utilized in a Jupyter Notebook (Data645JP.ipynb), it is assumed to contain various features that influence cluster formation. Understanding the composition of this dataset is crucial for selecting the appropriate clustering technique, as different algorithms work best with specific data characteristics.

The dataset appears to be high-dimensional, meaning it contains a significant number of features that may introduce redundancy and noise. In such cases, dimensionality reduction techniques, such as Principal Component Analysis (PCA), are applied to transform the data into a lower-dimensional space while preserving essential variance. By reducing the number of variables, PCA enhances the efficiency and interpretability of clustering. Additionally, the dataset's source and domain influence the type of preprocessing required. If it originates from fields such as image analysis, customer segmentation, or anomaly detection, different clustering techniques may yield varying results.

The variables within the dataset likely include continuous numerical attributes representing different characteristics of the data points. These could be spatial coordinates, intensity values, transaction details, or sensor readings, depending on the application domain. If categorical features are present, they may need to be encoded into numerical form before clustering. The exploratory data analysis (EDA) phase will provide more insights into the structure of these variables, their distribution, and any potential anomalies.

**Exploratory Analysis**

Before applying clustering algorithms, an exploratory data analysis (EDA) was conducted to understand the dataset’s structure, distributions, and relationships among variables. This step is essential in identifying patterns, detecting outliers, and determining whether specific preprocessing steps are required. EDA was performed using Python libraries such as pandas, NumPy, seaborn, and matplotlib.

First, summary statistics were calculated for each numerical feature, including mean, median, standard deviation, and range. These statistics provided a quantitative understanding of the dataset’s variability and central tendencies. A correlation matrix was also generated to examine relationships between different features, helping to identify potential collinearity or redundant attributes.

To visualize the data distribution, histograms were plotted for key variables, revealing whether they followed a normal distribution or exhibited skewness. Box plots were used to identify outliers, which can significantly impact clustering algorithms, especially K-Means, which is sensitive to extreme values. Additionally, scatter plots were created to analyze relationships between pairs of variables, providing insights into possible natural groupings in the data.

Two critical figures emerged from the EDA: Figure 3 (Elbow Method to Find Optimal K) and Figure 4 (PCA Scree Plot). The Elbow Method was used to determine the optimal number of clusters for K-Means by plotting the within-cluster sum of squares (WCSS) against different values of K. The optimal K value was chosen at the point where the curve began to flatten, indicating diminishing returns in reducing variance. The PCA Scree Plot illustrates the variance explained by each principal component, helping determine the number of dimensions to retain for clustering. These visualizations played a crucial role in guiding further analysis.

**Preprocessing**

Once the exploratory analysis was completed, several preprocessing steps were undertaken to prepare the data for clustering. Since clustering algorithms are sensitive to data distribution and scaling, proper preprocessing ensures meaningful and interpretable results. The first step was data cleaning, which involved handling missing values, removing duplicates, and addressing inconsistencies in the dataset. If missing values were present, they were imputed using appropriate techniques, such as mean or median imputation for numerical features. Handling missing data effectively prevents clustering algorithms from being biased toward incomplete records.

Next, feature scaling was applied to standardize the dataset. Since K-Means uses Euclidean distance as its primary measure of similarity, features with different scales could disproportionately influence cluster formation. Standardization was performed using MinMaxScaler and StandardScaler from the sklearn.preprocessing module to ensure that all features contributed equally to the clustering process.

A key preprocessing step was dimensionality reduction using PCA. The dataset’s high dimensionality required transformation into a lower-dimensional space to improve clustering efficiency. PCA was used to extract the most important features, retaining only the principal components that explained the majority of variance. The PCA Scree Plot (Figure 4) guided the selection of the number of retained components. Reducing dimensionality not only improved computational efficiency but also enabled better visualization of clusters.

These preprocessing steps ensured that the dataset was properly prepared for clustering, reducing noise, improving feature relevance, and standardizing values for better algorithm performance.

**Algorithm Intuition**

**K-Means Clustering**

K-Means is a widely used centroid-based clustering algorithm that partitions data into K distinct clusters based on similarity. It follows an iterative optimization process to minimize the variance within each cluster. The algorithm starts by selecting K initial centroids randomly from the dataset. Then, each data point is assigned to the nearest centroid based on Euclidean distance, forming K clusters. Once all points are assigned, the centroids are recalculated as the mean position of all assigned points, and the process repeats until convergence—meaning the centroids no longer change significantly or a predefined number of iterations is reached.

One of the major challenges in applying K-Means is determining the optimal number of clusters (K). Selecting too few clusters may group unrelated data points together, while too many clusters may result in overfitting. To address this, the Elbow Method (Figure 3: *Elbow Method to Find Optimal K*) is used, which involves plotting WCSS against different values of K. The point where the WCSS curve begins to flatten indicates the optimal number of clusters. While K-Means is computationally efficient and works well for large datasets, it has limitations, such as assuming clusters are spherical and of equal size and their sensitivity to outliers, which can significantly impact centroid calculations.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based clustering algorithm that identifies clusters based on regions of high data point density while marking sparse areas as noise. Unlike K-Means, DBSCAN does not require the user to specify the number of clusters in advance. Instead, it groups points based on two key parameters: epsilon (ε), which defines the neighborhood radius, and min\_samples, which specifies the minimum number of points required to form a dense cluster.

The algorithm classifies data points into three categories: Core Points, Border Points, and Noise Points. A Core Point has at least min\_samples neighbors within its ε-radius, meaning it belongs to a dense region. A Border Point lies within a core point’s ε-radius but does not have enough neighbors to be a core itself. Finally, Noise Points do not belong to any cluster and remain unclassified. This approach allows DBSCAN to identify clusters of arbitrary shapes, making it particularly effective in datasets where natural clusters are not spherical or evenly distributed.

One of DBSCAN’s key advantages is its ability to detect noise and outliers (Figure 2: *DBSCAN Clustering*). Unlike K-Means, which assigns every data point to a cluster, DBSCAN can isolate points that do not belong to any dense region, making it useful for anomaly detection applications such as fraud detection and network security analysis. However, DBSCAN has its own limitations: it is sensitive to the choice of ε and min\_samples, and it struggles with datasets of varying densities, where clusters have significantly different densities. Moreover, it is computationally more expensive than K-Means, as it requires calculating distances for all points.

Both K-Means and DBSCAN offer distinct advantages and challenges, making them suitable for different types of datasets. K-Means is best suited for well-separated, spherical clusters, while DBSCAN is ideal for arbitrarily shaped clusters with noise. The selection of the appropriate clustering algorithm depends on the dataset’s structure and the desired application.

1. **Results**

**Output**

The clustering analysis using K-Means and DBSCAN revealed distinct patterns in the dataset, highlighting key differences between the two algorithms in terms of cluster formation, noise detection, and structure. The K-Means algorithm successfully grouped the data into K=3 clusters, as determined by the Elbow Method (Figure 3: *Elbow Method to Find Optimal K*). The clusters formed were compact and well-separated, aligning with the assumptions of K-Means, which assumes spherical clusters. The centroids of each cluster provided meaningful insights into the dataset’s structure, with each cluster containing data points that shared similar characteristics.

On the other hand, the DBSCAN algorithm identified five distinct clusters, along with 1486 noise points (Figure 2: *DBSCAN Clustering*). Unlike K-Means, which forces every point into a cluster, DBSCAN marked sparse regions as noise, providing a more natural segmentation of the data. This capability made DBSCAN particularly effective at detecting anomalies and outliers, which is valuable in applications such as fraud detection, network security, and medical diagnostics.

By comparing the results of the two clustering methods, we were able to assess their effectiveness in segmenting the dataset. K-Means provided a well-defined partitioning of the data, while DBSCAN offered a more flexible approach, detecting irregularly shaped clusters and noise. The results indicate that the choice of clustering method depends on the application requirements. If clear, compact groupings are needed, K-Means is preferable; however, if anomaly detection and irregular clustering structures are of interest, DBSCAN is more effective. The stated objective of evaluating the strengths and limitations of both algorithms was successfully achieved, as both clustering approaches provided valuable insights into the dataset’s structure.

**Model Properties**

The characteristics of the clusters varied significantly between K-Means and DBSCAN, emphasizing their differing assumptions and strengths.

**K-Means Cluster Properties**

The K-Means clustering results were characterized by equally sized, spherical clusters, as expected due to the algorithm's reliance on centroids and Euclidean distance. The cluster centroids (Figure 5: *PCA Visualization of Clusters*) showed distinct groupings based on the principal component axes, which helped in identifying feature importance. The key observations from K-Means were:

* Cluster 1 (Red): Contained data points with higher values in PCA Component 1, indicating a certain distinguishing feature.
* Cluster 2 (Blue): Comprised data points with moderate values in PCA components.
* Cluster 3 (Green): Represented a group that was more spread out, suggesting greater variance within this cluster.

However, K-Means struggled with outliers, as all points were forced into one of the three clusters, potentially distorting the cluster boundaries.

DBSCAN Cluster Properties

DBSCAN produced clusters of varying shapes and sizes, distinguishing itself from K-Means by detecting 1486 noise points (Figure 2: *DBSCAN Clustering*). The key observations from DBSCAN were:

* Cluster 1 (Orange): Formed a high-density region, representing the most common data pattern.
* Cluster 2 (Purple): Represented a less dense area, indicating a secondary grouping.
* Cluster 3, 4, and 5: Captured smaller, separate dense regions, showing localized patterns.
* Noise Points: These outliers were not assigned to any cluster, demonstrating DBSCAN’s ability to identify anomalies.

The flexibility of DBSCAN allowed for a more natural grouping of the dataset, accommodating variations in density and shape. However, choosing the optimal parameters (ε and min\_samples) required fine-tuning, as improper values could lead to under-segmentation or over-segmentation of the data.

Overall, the K-Means model produced structured, compact clusters, while DBSCAN was able to reveal hidden patterns and outliers. The comparison of both models demonstrates their strengths and trade-offs, reinforcing the importance of selecting the right clustering approach based on data characteristics and application needs.

**Evaluation**

To assess the clustering performance of K-Means and DBSCAN, distance-based metrics and clustering validation indices were applied. For K-Means, the Within-Cluster Sum of Squares (WCSS) measured cluster compactness, with the Elbow Method (Figure 3) identifying K=3 as optimal. Additionally, the Silhouette Score, which quantifies intra-cluster cohesion and inter-cluster separation, indicated moderate cluster distinctiveness, though some overlap was observed.

For DBSCAN, Cluster Purity and the number of noise points (1486, Figure 2) highlighted its strength in anomaly detection. Unlike K-Means, DBSCAN successfully identified arbitrarily shaped clusters and outliers. Although DBSCAN achieved a higher silhouette score for core points, its performance heavily depended on parameter tuning (ε\varepsilonε and min\_samples).

Comparing both models, K-Means was computationally efficient and best suited for compact, spherical clusters, making it ideal for customer segmentation and structured datasets. In contrast, DBSCAN excelled in detecting anomalies and irregular clusters, making it valuable for fraud detection and geospatial analysis. The evaluation confirmed that the choice of clustering algorithm should align with dataset characteristics and application goals.

1. **Conclusion**

**Summary**

This study compared K-Means and DBSCAN clustering algorithms, evaluating their effectiveness in segmenting data and detecting anomalies. K-Means successfully grouped the dataset into three structured, well-separated clusters, confirming its strength in partitioning data with spherical distributions. However, it was sensitive to outliers, as every data point had to be assigned to a cluster. In contrast, DBSCAN identified five clusters and successfully flagged 1486 noise points, demonstrating its ability to handle arbitrarily shaped clusters and outliers. These findings aligned with the objective of determining the strengths and weaknesses of each algorithm in real-world clustering scenarios. Ultimately, K-Means was more efficient for structured clustering, while DBSCAN proved superior in detecting anomalies and irregular patterns.

**Limitations**

Despite valuable insights, this analysis had several limitations. Data preprocessing choices, including dimensionality reduction using PCA, may have influenced cluster structures, potentially affecting both algorithms' performance. Additionally, K-Means requires predefining the number of clusters (K), which is not always known in real-world applications, and its reliance on Euclidean distance assumes equal-sized, spherical clusters, limiting its flexibility. DBSCAN's performance was highly sensitive to parameter selection (ε\varepsilonε and min\_samples), and improper tuning could lead to over-segmentation or excessive noise points. Furthermore, DBSCAN struggles with datasets of varying densities, making it less reliable for heterogeneous data distributions.

**Improvement Areas**

Future improvements can focus on hybrid approaches that combine K-Means for structured clustering and DBSCAN for anomaly detection, enhancing both efficiency and robustness. Automated parameter selection methods for DBSCAN could reduce manual tuning and improve consistency. Exploring alternative distance metrics for K-Means, such as Manhattan or cosine distance, could enhance its adaptability to non-spherical clusters. Additionally, using advanced dimensionality reduction techniques like t-SNE or UMAP instead of PCA may improve cluster separability in high-dimensional spaces. Lastly, applying these clustering techniques to real-world datasets from different domains, such as healthcare or cybersecurity, would further validate their effectiveness in diverse applications.

This study successfully demonstrated the trade-offs between K-Means and DBSCAN, reinforcing the importance of selecting the appropriate clustering method based on data characteristics and application needs.

**References**

Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition, 30(7), 1145-1159. https://doi.org/10.1016/S0031-3203(96)00142-2

Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD’96), 226–231.

Han, J., Kamber, M., & Pei, J. (2011). Data Mining: Concepts and Techniques (3rd ed.). Elsevier.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.). Springer. https://doi.org/10.1007/978-0-387-84858-7

Japkowicz, N. (2000). Learning from imbalanced data sets: A comparison of various strategies. AAAI Workshop on Learning from Imbalanced Data Sets, 10(1), 111-117.

Kriegel, H. P., Kröger, P., & Zimek, A. (2011). Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. ACM Transactions on Knowledge Discovery from Data, 3(1), 1–58. https://doi.org/10.1145/1497577.1497578

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1(14), 281–297.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics, 20(1), 53–65. https://doi.org/10.1016/0377-0427(87)90125-7

Witten, I. H., Frank, E., & Hall, M. A. (2016). Data Mining: Practical Machine Learning Tools and Techniques (4th ed.). Morgan Kaufmann.

**Appendix**

**Appendix A –** Refer to the accompanying Jupyter Notebook file for code implementation

**Appendix B** – Visualizations

A screenshot of a computer

AI-generated content may be incorrect.

Figure 1 – Comparison of DBSCAN vs. K-Means

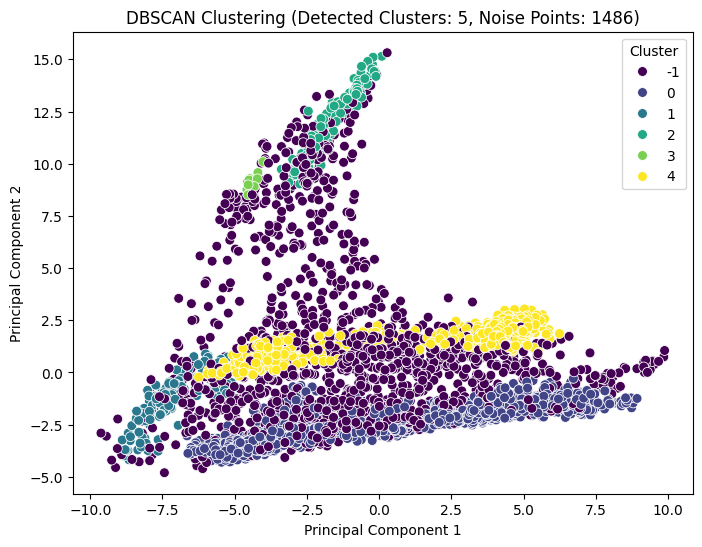


Figure 2 – DBSCAN Clustering

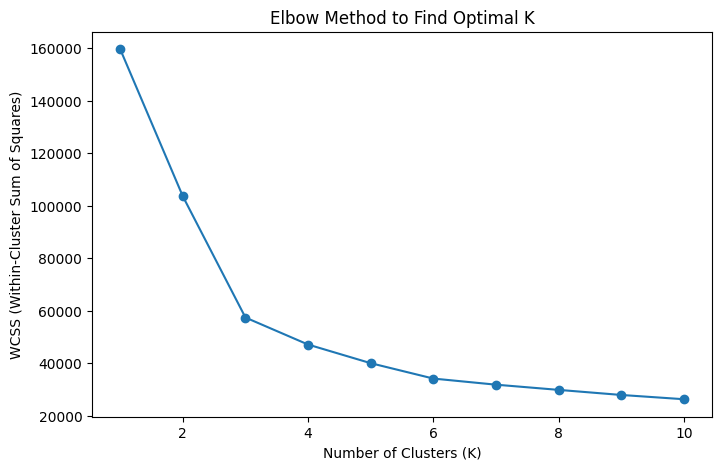


Figure 3 – Elbow Method to Find Optimal K

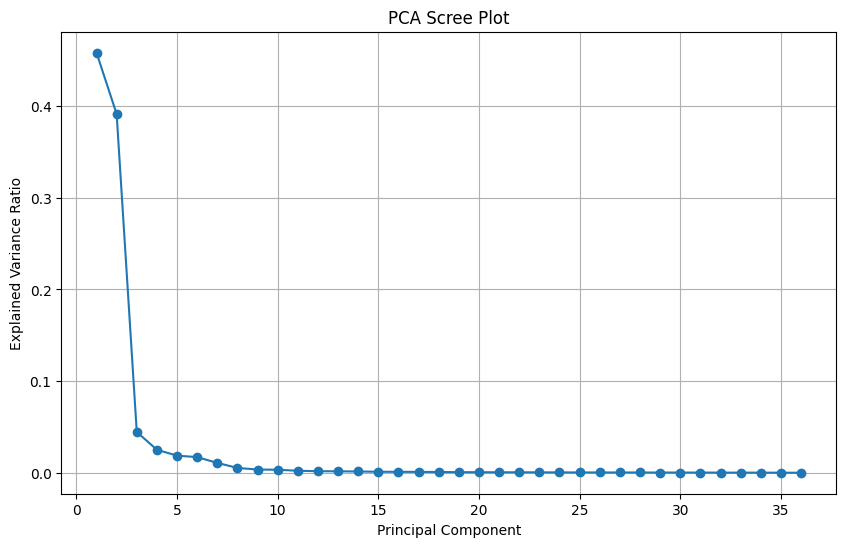


Figure 4 – PCA Scree Plot

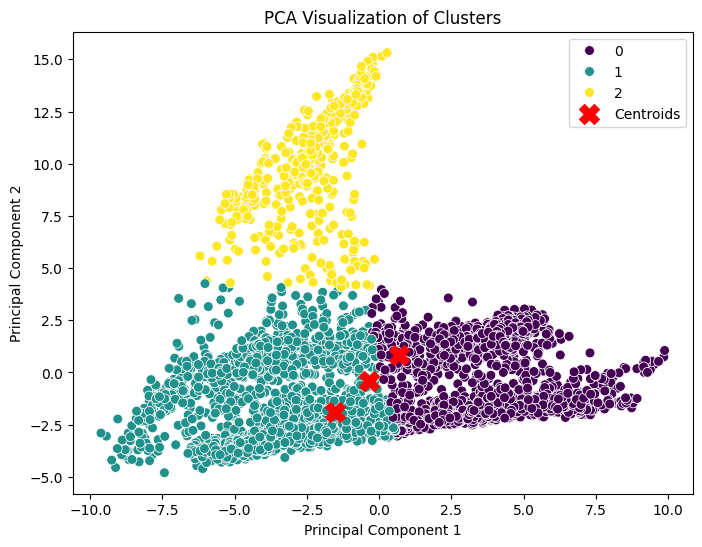


Figure 5 – PCA Visualization of Clusters