Assignment 1

K-Means Clustering Algorithm

TRISHA MAY GONZALES , 21-08571
EDRICK CANZANA , 21-03018
JOBERT JOHN CAYAO , 21-06830
VICTOR MANUEL DALISAY , 21-07108
JONER PAUL DE SILVA , 21-05301
JOHN MICHAEL GABALFIN , 21-00856
DOMINIC REYMAR GUNIO , 21-08535
MARIUS JACOB HERNANDEZ , 21-07121

Notes:

During our consultation, we realized that the K value according to the elbow method must be 2. We had initially used 3, and Sir Davood had advised us to keep the old code and explain our learnings in this documentation.

Old: Part 1, K=3, K Means Clustering-Group 2.ipynb
New: Part 2, K=2, K Means Clustering-Group 2.ipynb

Unsupervised Learning

Unsupervised learning is a type of machine learning algorithm that operates without predefined labels. Its primary function is to autonomously discover hidden patterns within data, eliminating the need for human intervention. In contrast to supervised learning, which relies on labeled datasets for classification, unsupervised learning processes unlabeled data, grouping similar data points into clusters. As such, the key strength of unsupervised learning lies in its ability to *find hidden patterns* in data that supervised models might not detect. Furthermore, it reduces reliance on manual effort and allows the handling of a larger volume of data in real-time compared to supervised learning.

Algorithm Overview

K-means clustering is a type of unsupervised machine learning algorithm that partitions the dataset into a predefined number, K, of clusters. The algorithm begins by selecting K initial centroids. Subsequently, each data point is assigned to the cluster with the nearest centroid. Once all points are assigned, new centroids are calculated as the mean of the data points within each cluster. This process of assignment and centroid updating is repeated iteratively until cluster assignments no longer change, indicating convergence. Consequently, to better understand the mechanics of this algorithm, each step of the algorithm will be programmed by the group on their own, such as: the initialization of centroids, assigning points to the centroids, updating the centroid, and iterating until there are no more changes in the cluster. Therefore, in this implementation, we will **not** rely on existing libraries or built-in functions like *sklearn.cluster.KMeans* as the main purpose is to better understand how the algorithm works through manual implementation.

Advantages and disadvantages of K-Means Clustering

Advantages of K-Means Clustering

1. **High Efficiency -** K-Means is recognized for its computational efficiency. With a linear time complexity, it is well-suited for processing large datasets. It is particularly useful for analyzing unlabeled big data, providing valuable insights as an unsupervised clustering technique. Additionally, it can leverage parallel computing to accelerate processing, further enhancing its speed and efficiency.

2. **Ease of Implementation -** One of the key strengths of K-Means is its simplicity. The algorithm is straightforward to implement and effectively identifies hidden patterns in complex datasets. The results are presented in an intuitive manner, making them easier to interpret compared to more complex models like neural networks. This makes K-Means a practical choice for quickly deriving insights from data.

3. **Adaptability -** K-Means demonstrates notable adaptability through its customizable components, making it applicable across various domains. Its flexibility stems from the support for different distance metrics and initialization techniques. The choice of distance metric allows for tailored clustering based on data characteristics, enabling effective application in fields ranging from spatial data analysis to text mining. Moreover, diverse initialization methods enhance robustness by reducing sensitivity to starting conditions, ensuring reliable performance across different datasets. This inherent adaptability makes K-Means a valuable tool for diverse data segmentation and pattern recognition challenges.

Disadvantages of K-Means Clustering

1. **Susceptibility to Outliers -** K-Means is highly sensitive to outliers, which are data points that deviate significantly from the rest of the dataset. Since the algorithm updates cluster centroids based on the mean, outliers can distort the results, leading to inaccurate clustering. To mitigate this issue, it is crucial to preprocess the data by identifying and removing outliers before applying K-Means.

2. **Influence of Initial Centroid Selection -** The effectiveness of K-Means depends on the initial placement of cluster centroids. Poor initialization can lead to suboptimal clustering results, as the algorithm may converge to a local minimum instead of the optimal global solution. Proper centroid initialization techniques, such as K-Means++, can help improve clustering accuracy.

3. **Challenges with Cluster Selection -** K-Means work best when clusters are spherical and have similar variances. However, in real-world scenarios, data clusters often have irregular shapes and varying densities. When dealing with such datasets, K-Means may struggle to accurately capture the structure of the data. In these cases, alternative clustering methods, such as DBSCAN or hierarchical clustering may be more suitable.

Objective and Purpose of the Algorithm

K-Means Clustering is used to automatically group a dataset into **K clusters** based on similarity. The objective of this algorithm is to partition data points into K distinct clusters in such a way that the variance *within* each cluster is minimized. This minimization ensures that data points within each cluster are tightly grouped and highly similar to one another, forming cohesive and well-defined clusters. This clustering method is widely used in fields such as:

- **Customer Segmentation** – Grouping customers with similar purchasing behavior together.
- **Image Segmentation** – Identifying different objects in an image by grouping *pixels with similar characteristics* (e.g., color, intensity).
- **Anomaly Detection** – Detecting fraudulent activities by spotting unusual data points that differ significantly from typical patterns.

Also, the algorithm aims to:

1. **Identify Natural Groupings** – Automatically discover patterns or groupings in data without prior knowledge of labels.
2. **Minimize Variation Within Clusters** – Ensure that data points within the same cluster are as close as possible to the cluster's centroid.
3. **Maximize Separation Between Clusters** – Make sure clusters are well-separated from each other.

WCSS (**Within-Cluster Sum of Squares**) is a key metric used in K-Means Clustering specifically designed to evaluate the *compactness* of the clusters formed. WCSS quantifies the total variance *within* each cluster by calculating the sum of the squared distances between each data point and its assigned cluster centroid.

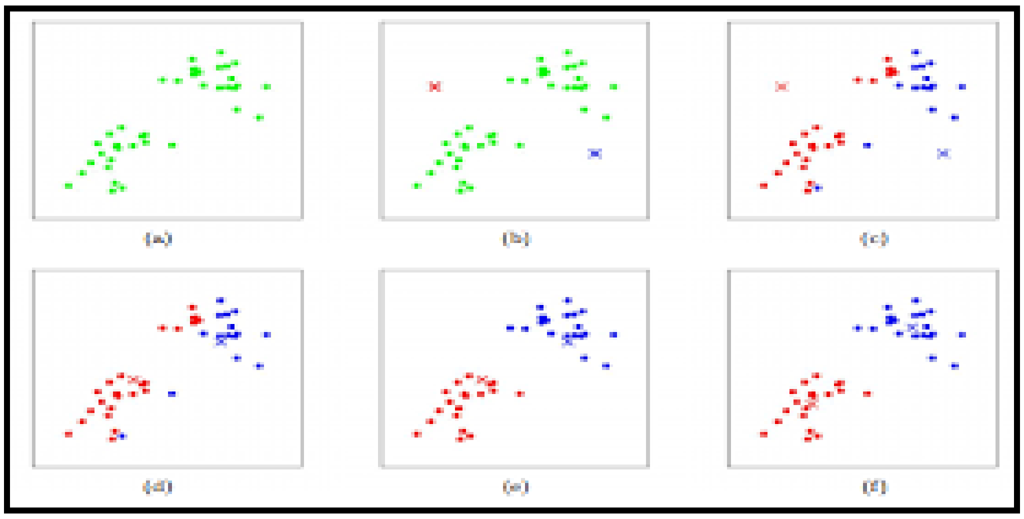
The core principle of K-Means is to <u>minimize this WCSS value</u>, effectively ensuring that data points within each cluster are as close as possible to their respective cluster centroid, thereby creating tightly packed, internally similar clusters.

Formula:

$$WCSS = \sum_{k=1}^{K} \sum_{x_i \in C_k} ||x_i - \mu_k||^2$$

Where:

- K = Total number of clusters.
- x_i = Number of data points in cluster k
- u_k = Centroid of cluster k
- $|| x_i - u_k ||^2$ = Squared Euclidean distance between a data point and the cluster centroid



(Progression of K-Means Clustering)

(a) Distribution of Initial Data
- The plot features data points without labels (green).
- It is still pending to assign clusters or centroids.

(b) Random Initialization of Centroids
- Two centroids, represented by the red and blue "X", are positioned randomly on the plot.
- The centroids serve as the starting cluster centers.

(c) Initial Allocation of Data Points to Centroids
- Each data point is allocated to the nearest centroid according to Euclidean distance.
- There are two distinct colors that emerge:
    - Red for data points nearer to the red centroid.
    - Blue for data points that are nearer to the blue centroid.

(d) First Update of Centroids
- This is the first centroid update, the update utilizes the WCSS formula.
- The centroids move to the center of their assigned points (mean position).

(e) Reassignment of Data Points to Updated Centroids
- Since the centroids have moved, some points may now be closer to a different centroid.
- Data points are reassigned to the nearest updated centroid.

(f) Final Cluster Formation (Convergence)
- The centroids stabilize, meaning they no longer move significantly.
- Data points remain in their assigned clusters.

Programming Language

Python is our preferred language for algorithms like K-Means clustering due to its simplicity, efficiency, powerful libraries, and robust visualization capabilities. Its versatility enables developers to easily preprocess data, perform clustering, and visualize results with minimal effort. Furthermore, its extensive collection of libraries significantly contributes to both the straightforward implementation and the efficient performance of K-Means.

Libraries, Dependencies, and Prerequisites

Libraries and Dependencies

- Python Libraries:
  - **numpy:** Used for efficient numerical operations and handling arrays.
    In the code, NumPy creates the dataset and helps with fast matrices.

  - **pandas:** Used for handling tabular data. In the code, it helps when working with large datasets in CSV, Excel, or databases before clustering.

  - **matplotlib:** Used for visualizing clusters by plotting data points and centroids. In the code, it is used to color clusters and mark centroids for easy interpretation.

  - **sklearn:** Provides built-in K-Means implementation. In the code, it handles clustering, training, and predictions efficiently with minimal code.

  - **seaborn:** Used for enhancing visualization with better color schemes and aesthetics. In the code, it is combined with Matplotlib to make heatmaps, pair plots, and cluster distributions clearer.

  - **scipy:** Provides advanced mathematical and statistical functions. In the code, it is used for distance calculations, optimizations, and hierarchical clustering.

Prerequisites

1. **Original dataset** - Ensure that the CSV file **sns.csv** is in the correct directory: drive/MyDrive/datasets/sns.csv
2. **Modified dataset** - Ensure that the CSV file **modified_sns.csv** is in the correct directory: drive/MyDrive/datasets/modified_sns.csv

Dataset Overview

This dataset contains mobile app usage data, capturing key numerical attributes that provide insights into user behavior. Each record consists of the following attributes:
- **App**: Name of the mobile application.
- **Usage (*minutes*)**: Total time spent on the app in minutes.
- **Notifications**: Number of notifications received from the app.
- **Times Opened**: Number of times the app was accessed.

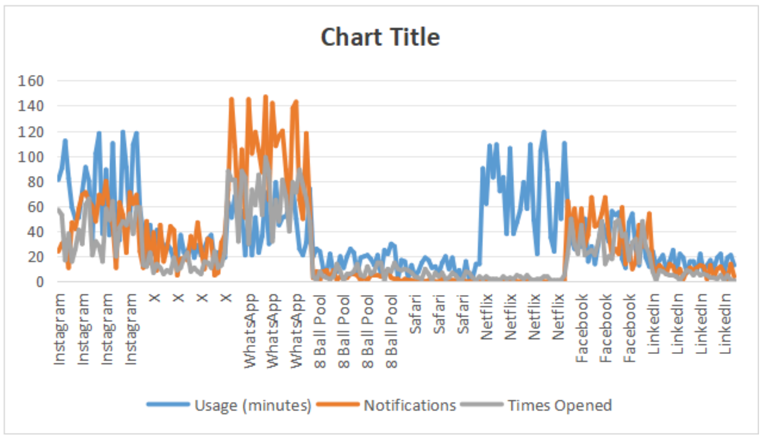
This dataset is useful for mobile app engagement analysis due to the following characteristics:
- **Continuous Numerical Attributes**: The dataset consists of numerical values, making it suitable for various analytical and machine learning models.

- **User Behavior Insights**: The data can help identify patterns in app engagement, such as which apps users spend the most time on or receive the most notifications from.
- **Feature Engineering Potential**: Additional features, such as average session duration or notification-to-open ratio, can be derived to enhance analysis.
- **Trend Analysis**: The dataset can be used to track user habits and detect trends in mobile app usage over time.

In conclusion, this dataset serves as a valuable resource for analyzing mobile app usage patterns, providing insights into user interaction and engagement that can be leveraged for research or business decision-making.

Original dataset:
https://www.kaggle.com/datasets/anandshaw2001/mobile-apps-screentime-analysis

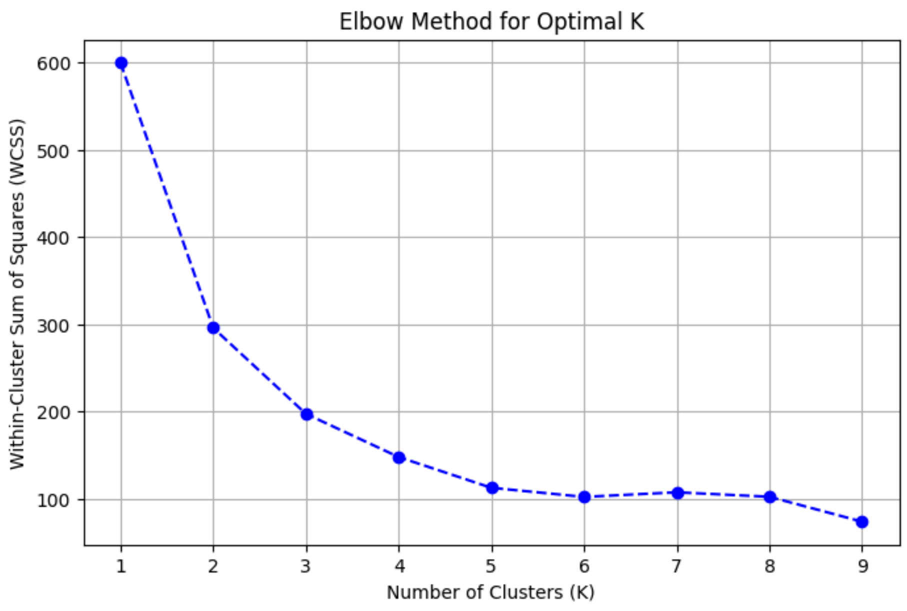
Modified dataset:
https://drive.google.com/file/d/1Wu4mcdsiurCL4gDuOCP1Ij8Vq11qI3wR/view?usp=sharing



Results and Graphs

I. Original dataset with K=3

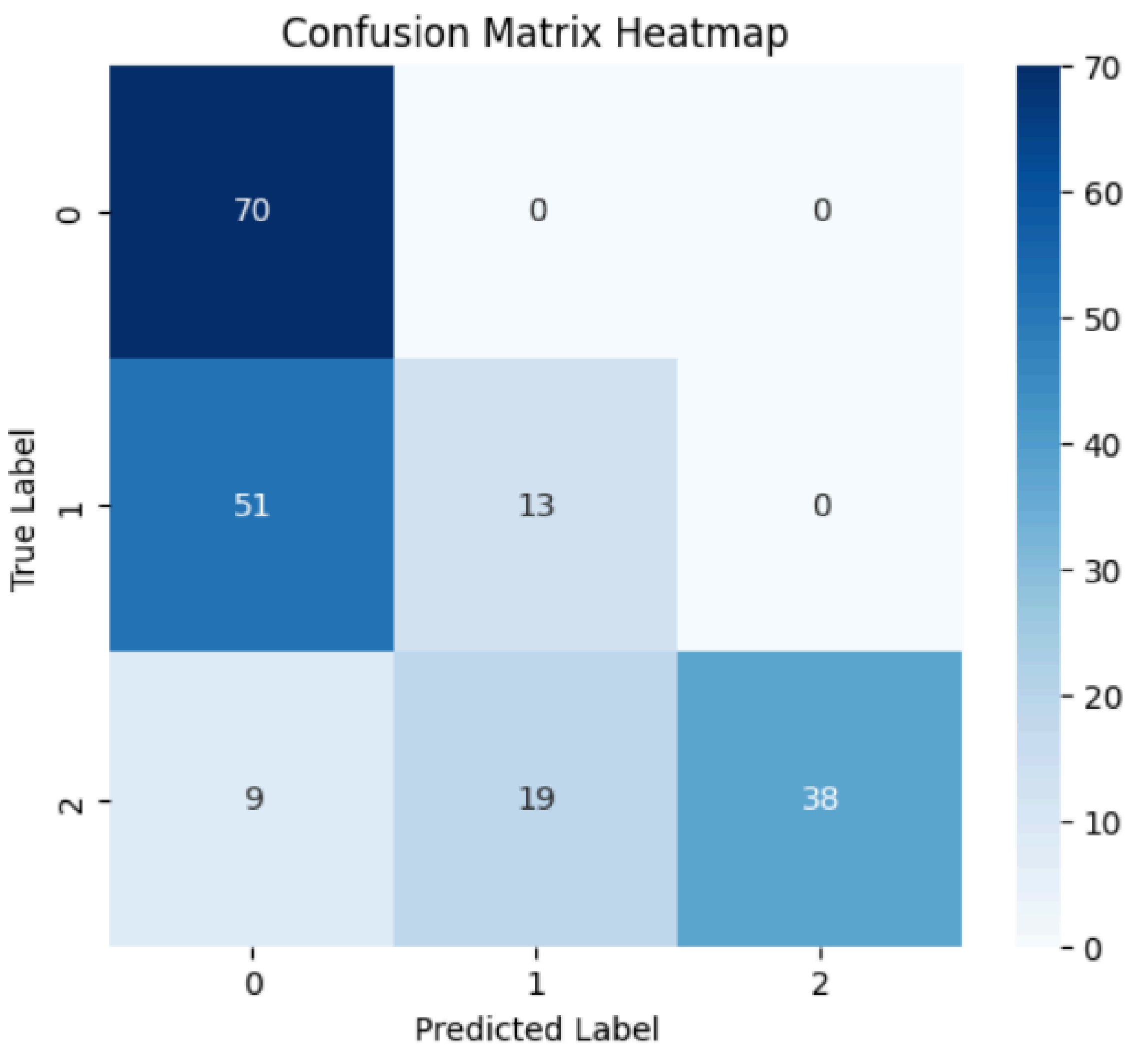
In this dataset, we first used K = 3, as the group misinterpreted the findings of the elbow graph from the image below.



Subsequently, the group then proceeded to take 3 clusters/groups into consideration. However, since the data set used within this unsupervised learning algorithm implementation has no true label, we decided to generate the true labels of the data set using the usage feature and convert it to quantiles to evaluate the accuracy of clusters.

In order to map the true labels and the predicted clusters we used a function that performs the Hungarian Algorithm. After mapping, the algorithm compared the True Labels with the Predicted labels. Consequently, the result showcased the following classification metrics:

- **Accuracy: 0.6050**
- **Precision: 0.6485**
- **Recall: 0.6050**
- **F1 Score: 0.5728**



- True Positive$_0$ (70): Correctly predicted class 0 when it was actually class 0.
- False Positive$_0$ (60): The sum of the number of incorrect predictions for class 0 when it was actually either 1 or 2.
- False Negative$_0$ (0): The sum of the number of incorrect predictions for class 1 and 2 when it was actually class 0.
- True Negative$_0$ (70): The sum of all predictions not pertaining to 0.
- True Positive$_1$ (13): Correctly predicted class 1 when it was actually class 1.

- False Positive$_1$ (19): The sum of the number of incorrect predictions for class 1 when it was actually either 0 or 2.
- False Negative$_1$ (51): The sum of the number of incorrect predictions for class 0 and 2 when it was actually class 1.
- True Negative$_1$ (117): The sum of all predictions not pertaining to 1.
- True Positive$_2$ (38): Correctly predicted class 2 when it was actually class 2.
- False Positive$_2$ (0): The sum of the number of incorrect predictions for class 2 when it was actually either 0 or 1.
- False Negative$_2$ (28): The sum of the number of incorrect predictions for class 0 and 1 when it was actually class 2.
- True Negative$_2$ (134): The sum of all predictions not pertaining to 2.

Confusion Matrix Interpretation For 0

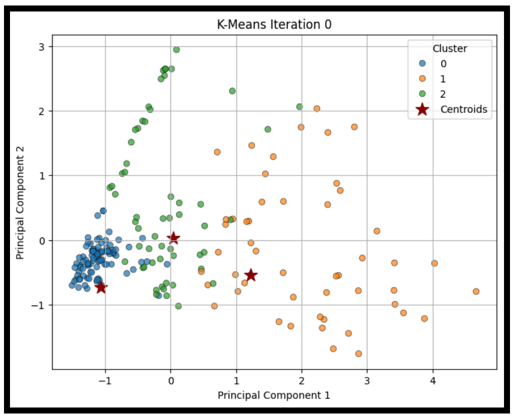
	Predicted: 0	Predicted: 1	Predicted: 2
Actual: 0	70 (True Positive)	0 (False Negative)	0 (False Negative)
Actual: 1	51 (False Positive)	13 (True Negative)	0 (True Negative)
Actual: 2	9 (False Positive)	19 (True Negative)	38 (True Negative)

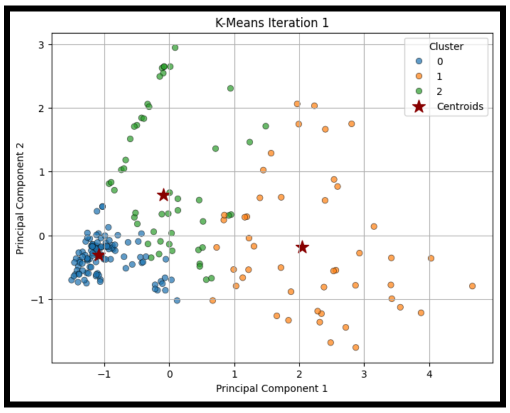
Confusion Matrix Interpretation For 1

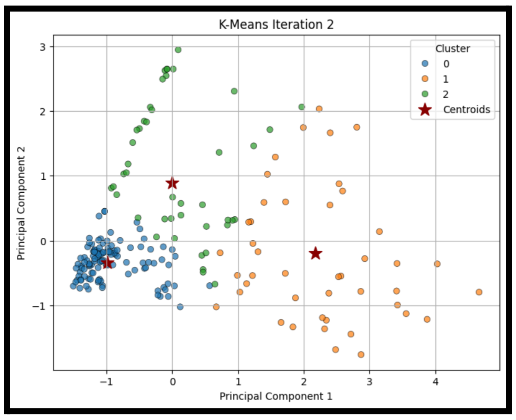
	Predicted: 0	Predicted: 1	Predicted: 2
Actual: 0	70 (True Negative)	0 (False Positive)	0 (True Negative)
Actual: 1	51 (False Negative)	13 (True Positive)	0 (False Negative)
Actual: 2	9 (True Negative)	19 (False Positive)	38 (True Negative)

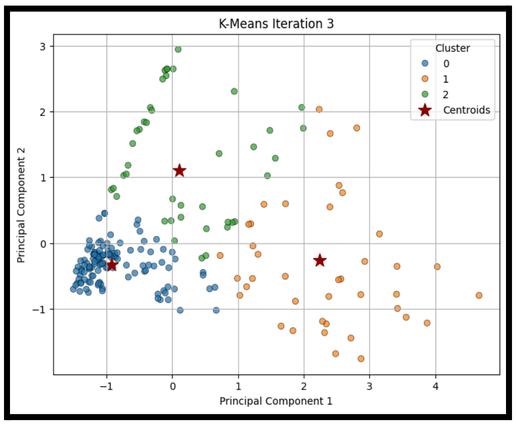
Confusion Matrix Interpretation For 2

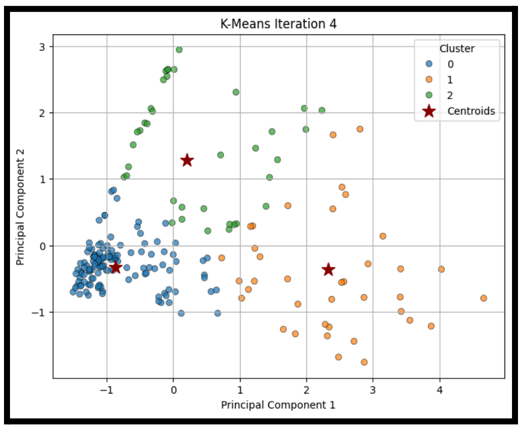
	Predicted: 0	Predicted: 1	Predicted: 2
Actual: 0	70 (True Negative)	0 (True Negative)	0 (False Positive)
Actual: 1	51 (True Negative)	13 (True Negative)	0 (False Positive)
Actual: 2	9 (False Negative)	19 (False Negative)	38 (True Positive)

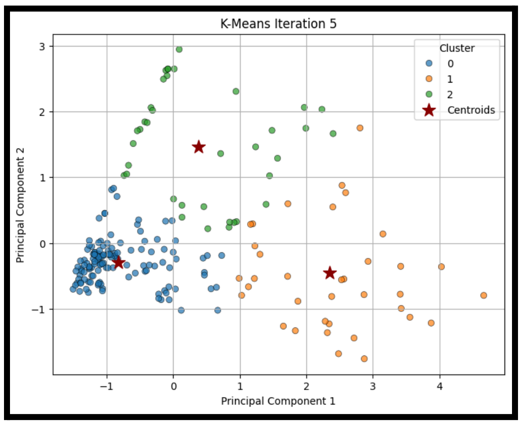

K-Means Iteration 0
Cluster
0
1
2
Centroids
Principal Component 1
Principal Component 2

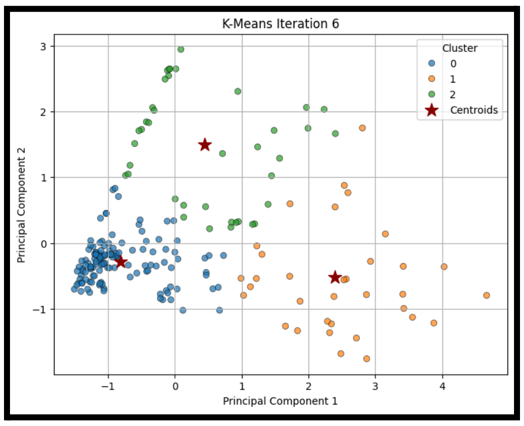

K-Means Iteration 1
Cluster
0
1
2
Centroids
Principal Component 1
Principal Component 2

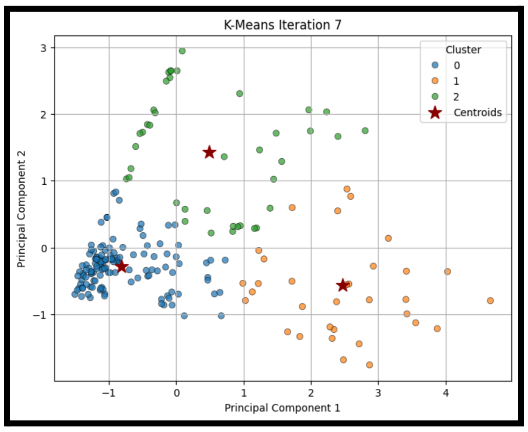

K-Means Iteration 2
Cluster
0
1
2
Centroids
Principal Component 1
Principal Component 2

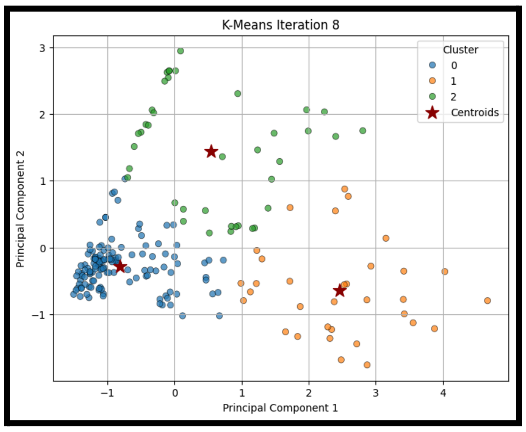

K-Means Iteration 3
Cluster
0
1
2
Centroids
Principal Component 1
Principal Component 2

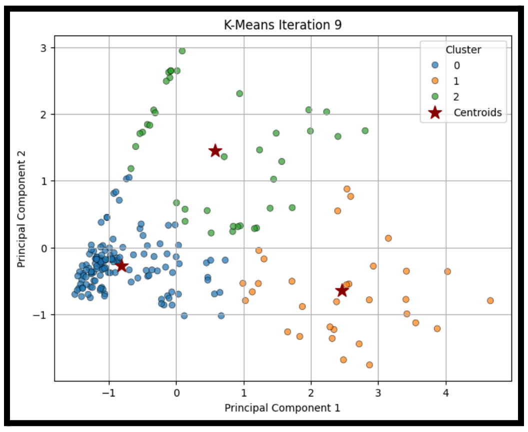

K-Means Iteration 4
Cluster
0
1
2
Centroids
Principal Component 1
Principal Component 2


K-Means Iteration 5
Cluster
0
1
2
Centroids
Principal Component 1
Principal Component 2

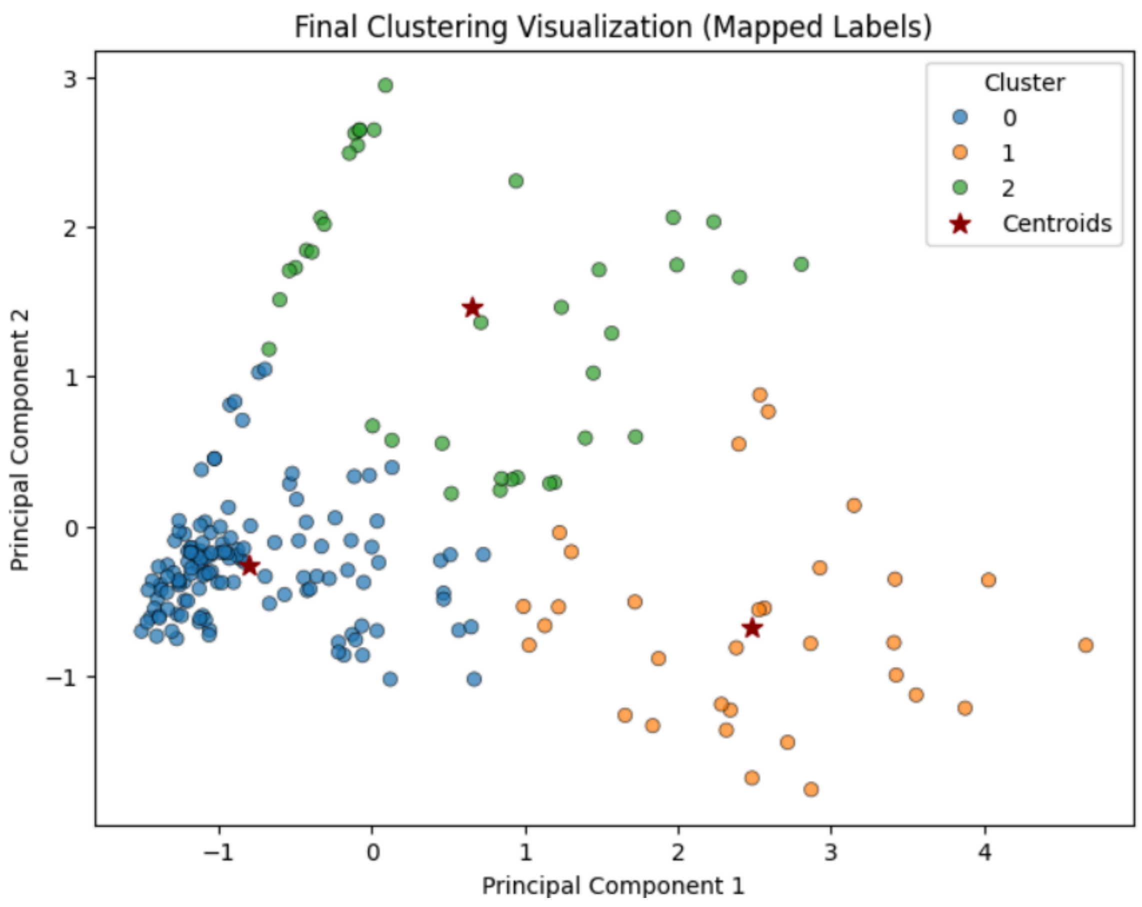

K-Means Iteration 6
Cluster
0
1
2
Centroids
Principal Component 1
Principal Component 2


K-Means Iteration 7
Cluster
0
1
2
Centroids
Principal Component 1
Principal Component 2


K-Means Iteration 8
Cluster
0
1
2
Centroids
Principal Component 1
Principal Component 2

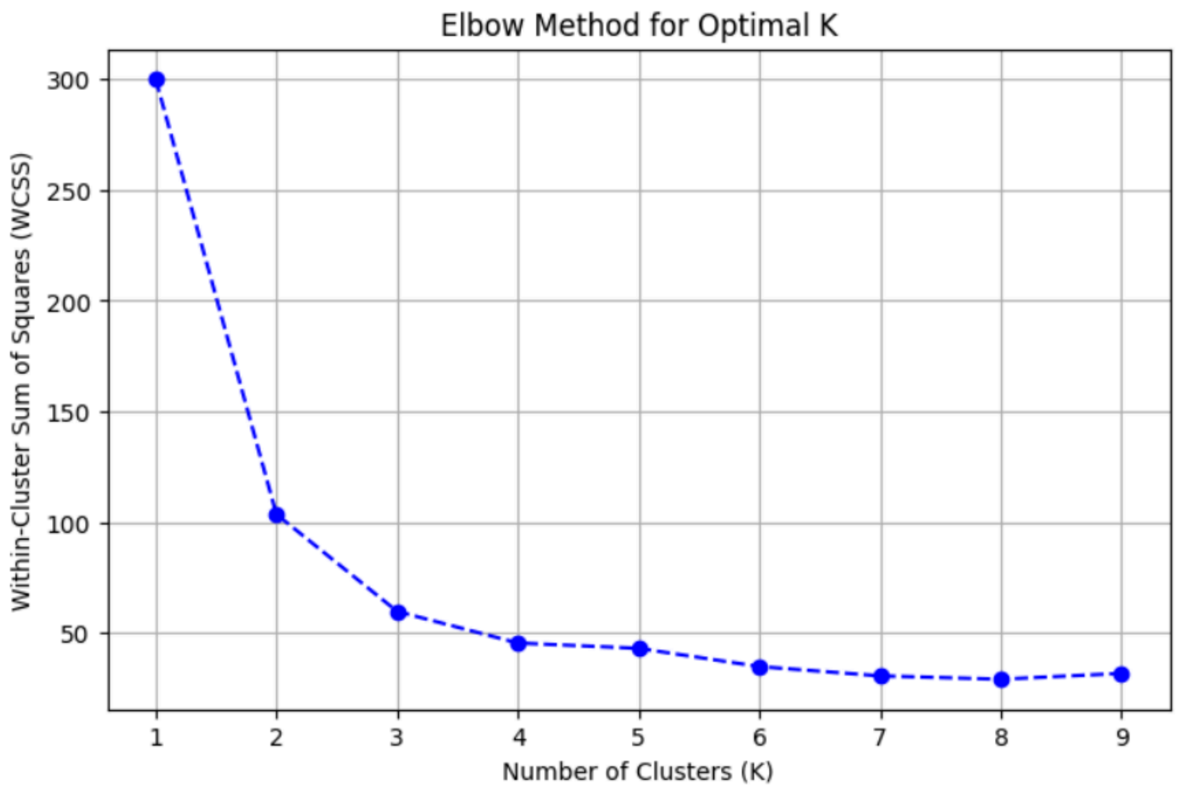

K-Means Iteration 9
Cluster
0
1
2
Centroids
Principal Component 1
Principal Component 2

The figures illustrate how the algorithm evolves with each iteration until it reaches convergence. In this instance, convergence occurred at Iteration 9. This implies that, depending on the problem's complexity and the initial setup, the algorithm may need at least 9 iterations—or possibly more—to find the optimal distribution. The point of convergence also highlights the algorithm's efficiency, as fewer iterations typically indicate faster computation and improved optimization performance.



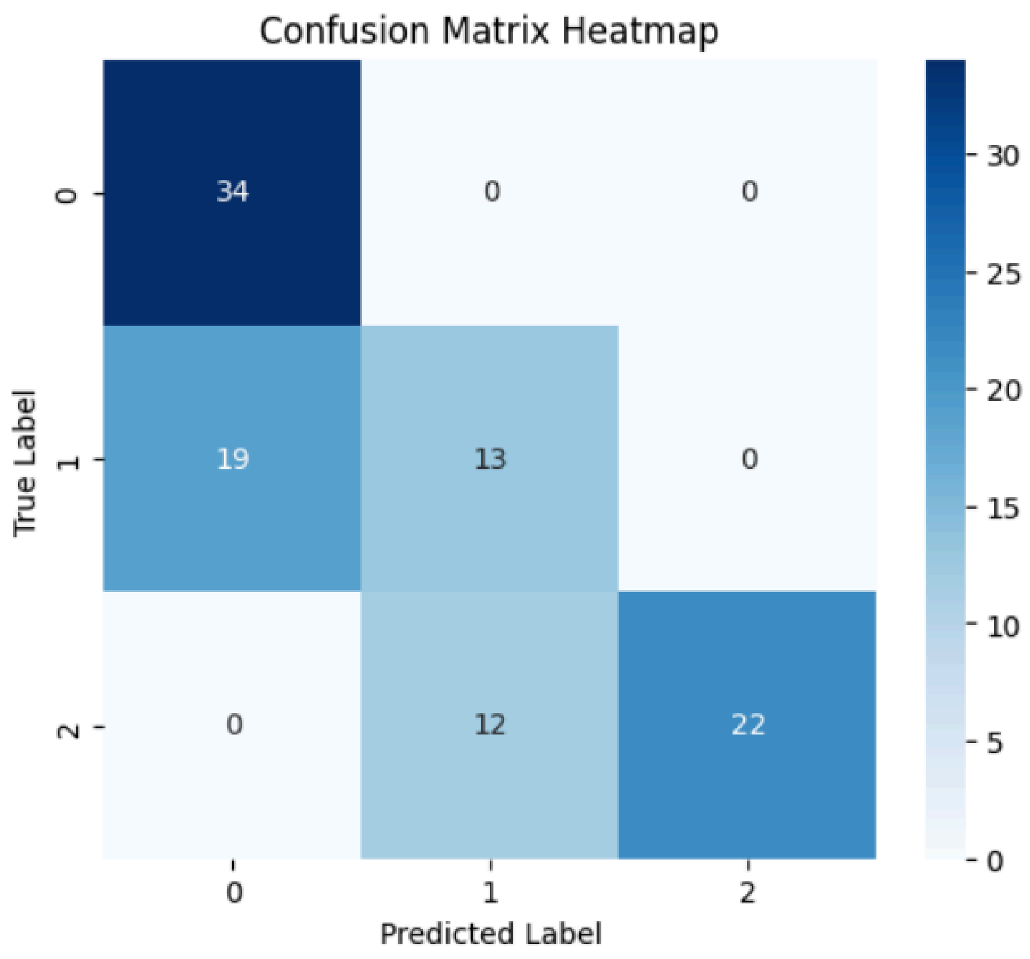
In this scatter plot we used 2 principal components to map the 3 features within a 2d space. The red stars represent the cluster centroids, which indicates the average position of points within the cluster.

II. **Modified dataset with K=3**



Similarly to the original K=3 dataset, we mapped the true labels and the predicted clusters by making a function that performs the Hungarian Algorithm. After mapping, the algorithm compared the True Labels with the Predicted labels. Consequently, the result showcased the following classification metrics:

- **Accuracy: 0.6900**
- **Precision: 0.7245**
- **Recall: 0.6900**
- **F1 Score: 0.6789**



- True Positive$_0$ (34): Correctly predicted class 0 when it was actually class 0.
- False Positive$_0$ (19): The sum of the number of incorrect predictions for class 0 when it was actually either 1 or 2.
- False Negative$_0$ (0): The sum of the number of incorrect predictions for class 1 and 2 when it was actually class 0.
- True Negative$_0$ (47): The sum of all predictions not pertaining to 0.
- True Positive$_1$ (13): Correctly predicted class 1 when it was actually class 1.
- False Positive$_1$ (12): The sum of the number of incorrect predictions for class 1 when it was actually either 0 or 2.
- False Negative$_1$ (19): The sum of the number of incorrect predictions for class 0 and 2 when it was actually class 1.
- True Negative$_1$ (56): The sum of all predictions not pertaining to 1.
- True Positive$_2$ (22): Correctly predicted class 2 when it was actually class 2.
- False Positive$_2$ (0): The sum of the number of incorrect predictions for class 2 when it was actually either 0 or 1.
- False Negative$_2$ (12): The sum of the number of incorrect predictions for class 0 and 1 when it was actually class 2.
- True Negative$_2$ (66): The sum of all predictions not pertaining to 2.

Confusion Matrix Interpretation For 0

	Predicted: 0	Predicted: 1	Predicted: 2
Actual: 0	34 (True Positive)	0 (False Negative)	0 (False Negative)

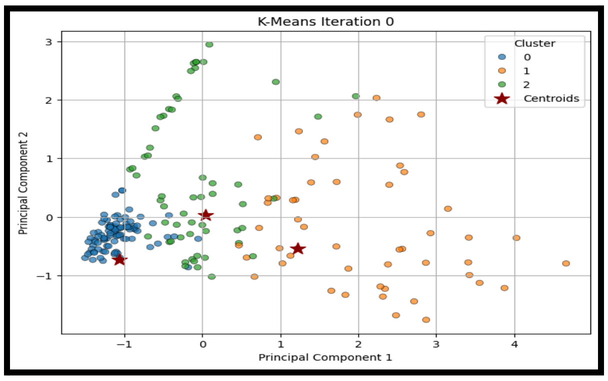
Actual: 1	19 (False Positive)	13 (True Negative)	0 (True Negative)
Actual: 2	0 (False Positive)	12 (True Negative)	22 (True Negative)

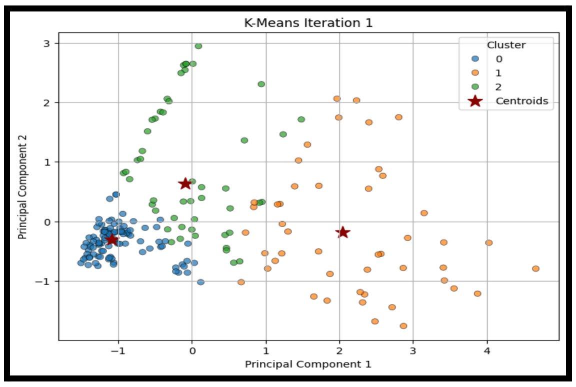
Confusion Matrix Interpretation For 1

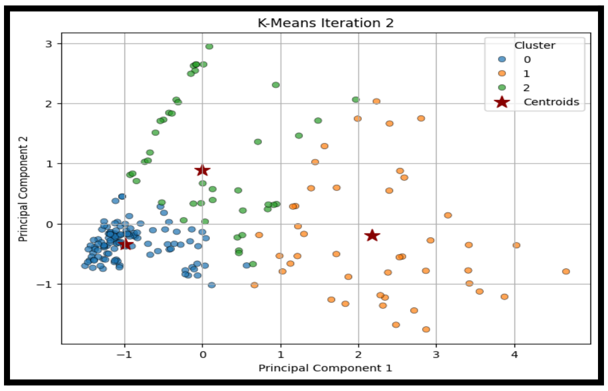
	Predicted: 0	Predicted: 1	Predicted: 2
Actual: 0	34 (True Negative)	0 (False Positive)	0 (True Negative)
Actual: 1	19 (False Negative)	13 (True Positive)	0 (False Negative)
Actual: 2	0 (True Negative)	12 (False Positive)	22 (True Negative)

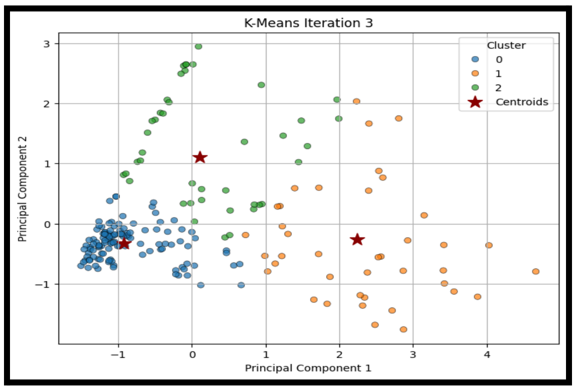
Confusion Matrix Interpretation For 2

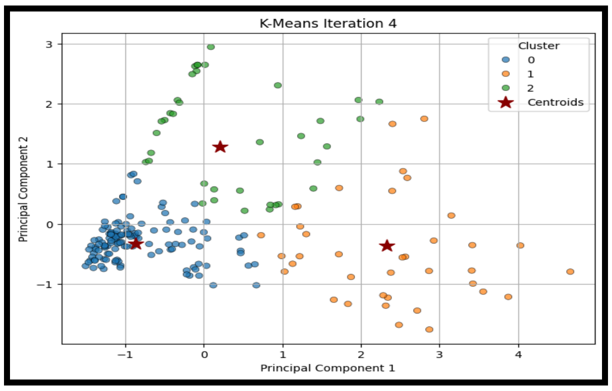
	Predicted: 0	Predicted: 1	Predicted: 2
Actual: 0	34 (True Negative)	0 (True Negative)	0 (False Positive)
Actual: 1	19 (True Negative)	13 (True Negative)	0 (False Positive)
Actual: 2	0 (False Negative)	12 (False Negative)	22 (True Positive)

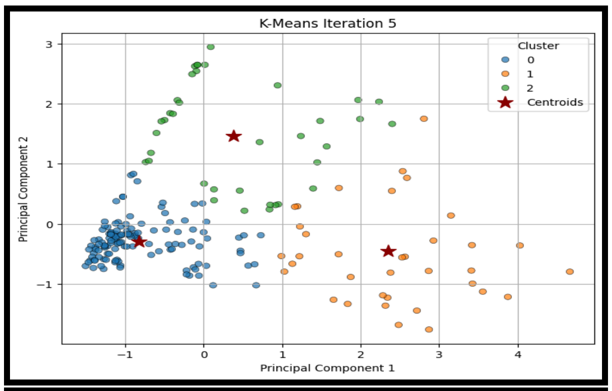


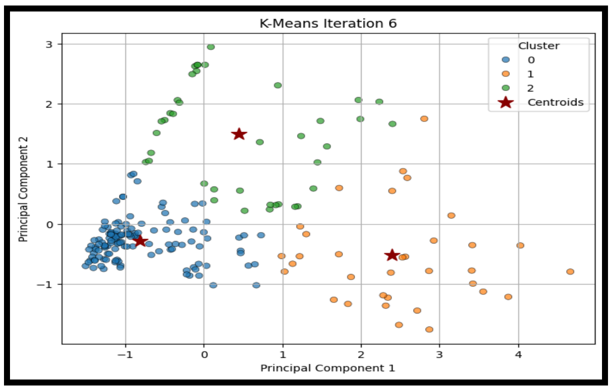


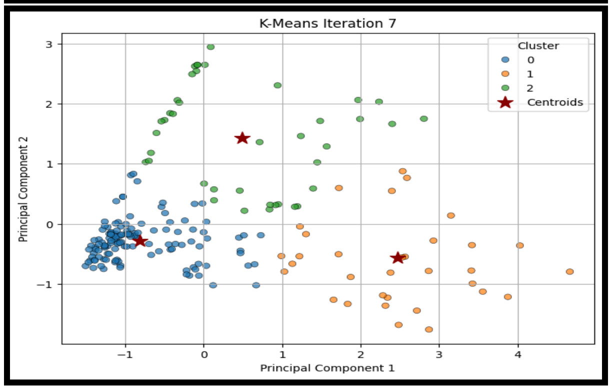


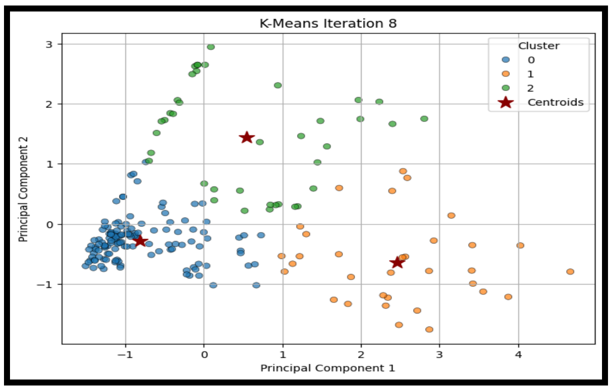


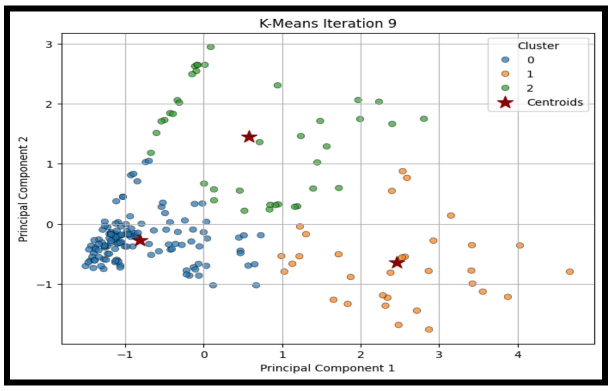




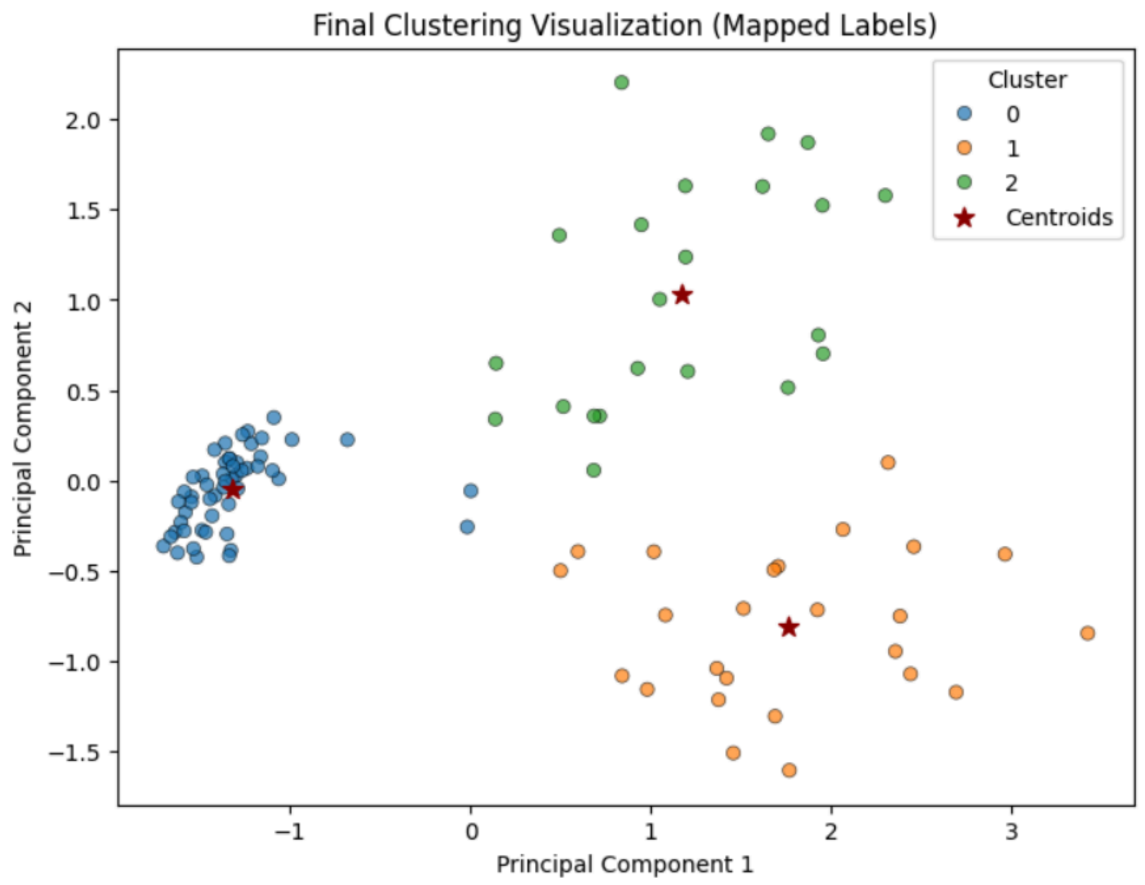








The figures display the progression of the algorithm across each iteration until convergence was achieved. In this case, the algorithm successfully converged at Iteration 9. This suggests that, depending on the complexity of the problem and the initial conditions, the algorithm may require at least 9 iterations—or potentially more—to identify the optimal distribution. The convergence point also reflects the efficiency of the algorithm, as fewer iterations generally indicate faster computation and better optimization performance.

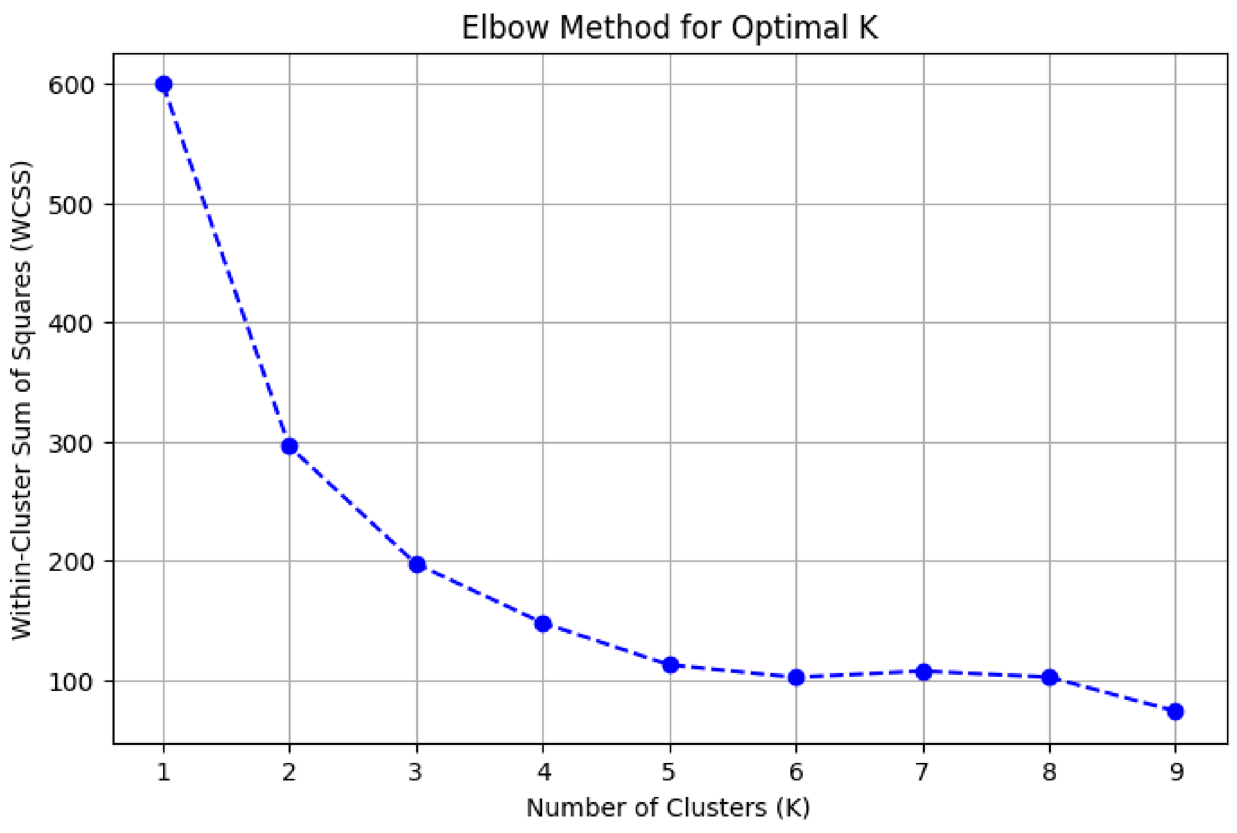


In this scatter plot we used 2 principal components to map the 3 features within a 2d space. The red stars represent the cluster centroids, which indicates the average position of points within the cluster.

III. Original dataset with K=2

In here, we tried an alternative k wherein we disregard the previous Usage column and defined our own **true labels (High Engagement, Low Engagement)** for clustering. This will compel the algorithm to categorize based on High and Low Engagement apps.

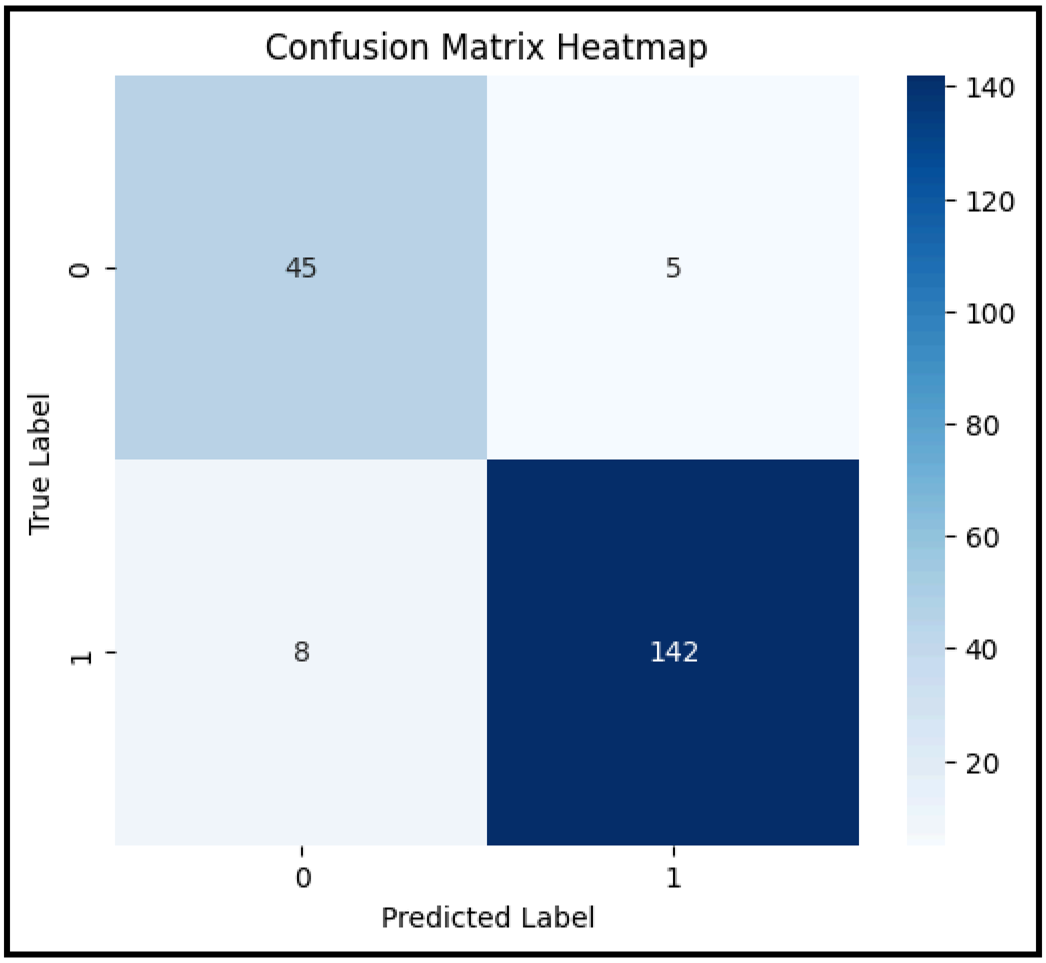
Because the original dataset had 8 labels, we chose the true labels by picking the 2 highest engagement apps and 2 from the lowest engagement apps. We used the same true labels for Original and Modified dataset K=2 to better compare the performance of the algorithm. This way, we can see if outliers play a huge role in the performance.



In this Elbow Method graph, we can see the biggest drop in WCSS happening between K=1 and K=2 and since our dataset is now a two-group structure, using 2 as our K is natural. As long as increasing K doesn't reveal meaningfully different subgroups, keeping K = 2 is justified.

Classifying the apps between High and Low Engagement, we then call them to a function where the clusters will be mapped to the manually defined labels. After the algorithm finishes mapping the respective clusters, we compare their True Labels and their Predicted Labels. We measured the results using the following classification metrics:

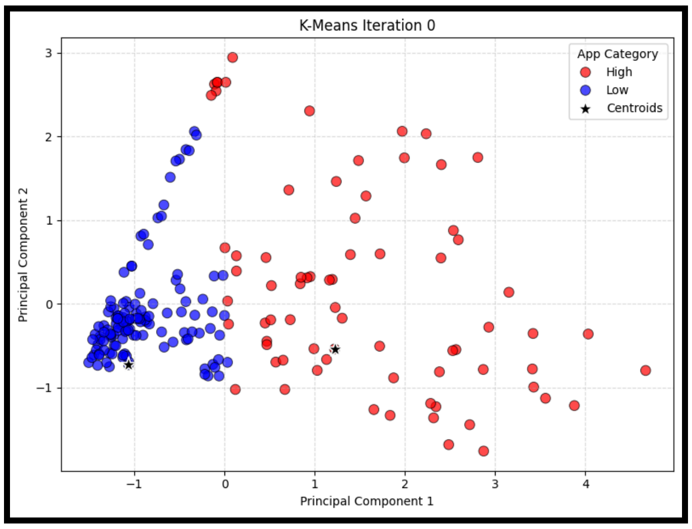
- **Accuracy: 0.9350**
- **Precision: 0.9368**
- **Recall: 0.9350**
- **F1 Score: 0.9356**

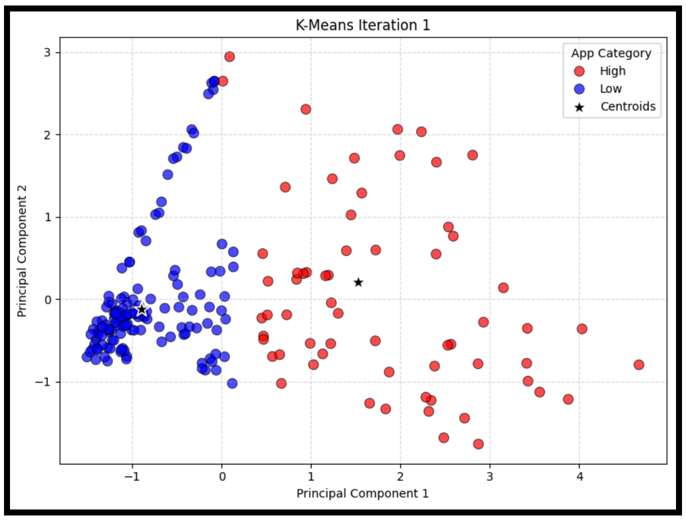


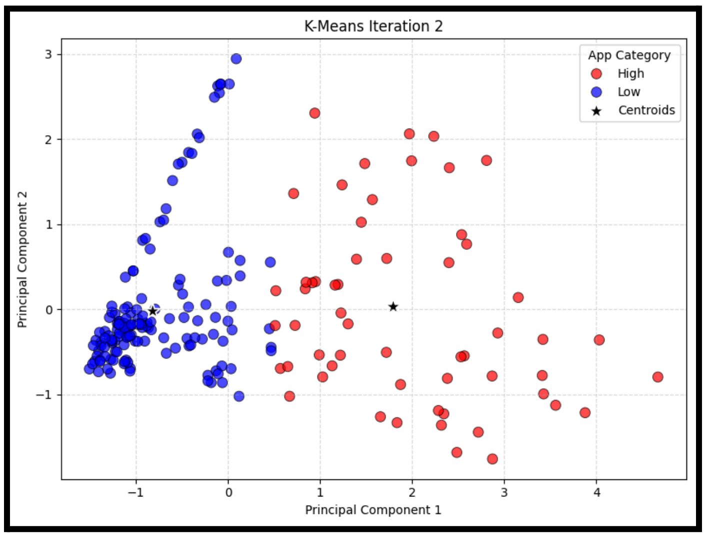
- True Negatives (45): Correctly predicted class 0 when it was actually 0.
- False Positives (5): Incorrectly predicted class 1 when it was actually 0.
- False Negatives (8): Incorrectly predicted class 0 when it was actually 1.
- True Positives (142): Correctly predicted class 1 when it was actually 1.

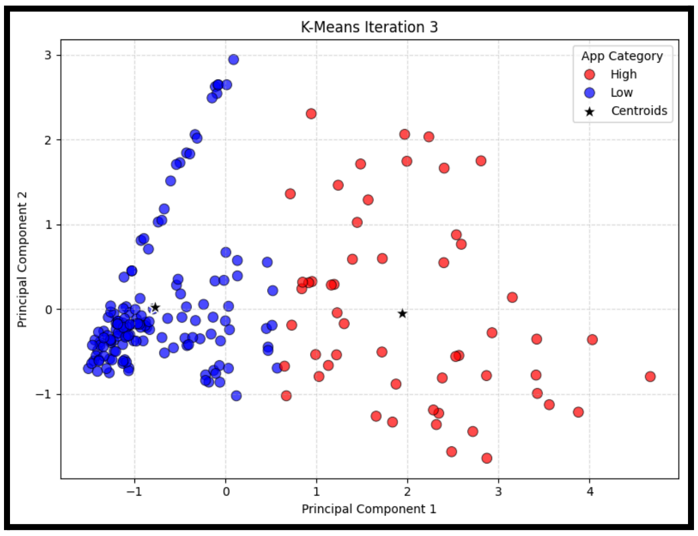
Confusion Matrix Interpretation

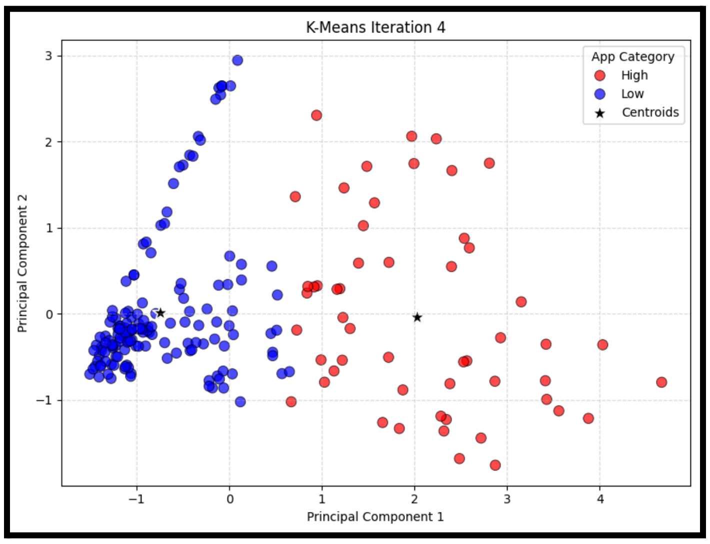
	Predicted: 0	**Predicted: 1**
Actual: 0	**45 (True Negative)**	**5 (False Positive)**
Actual: 1	**8 (False Negative)**	**142 (True Positive)**

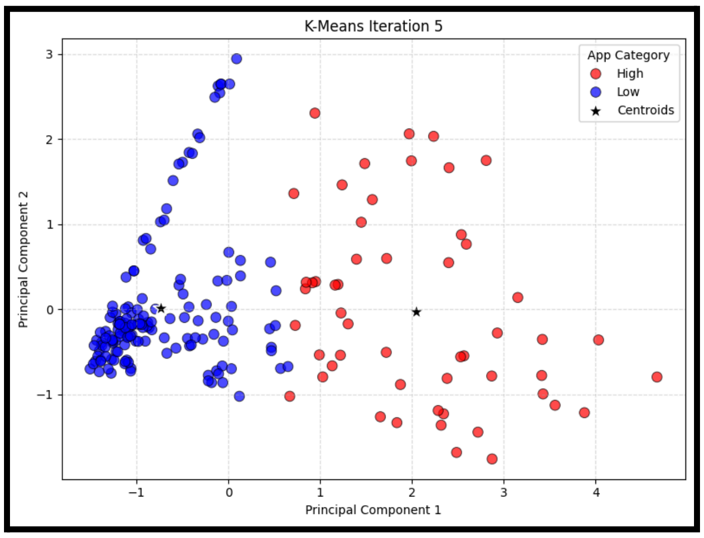




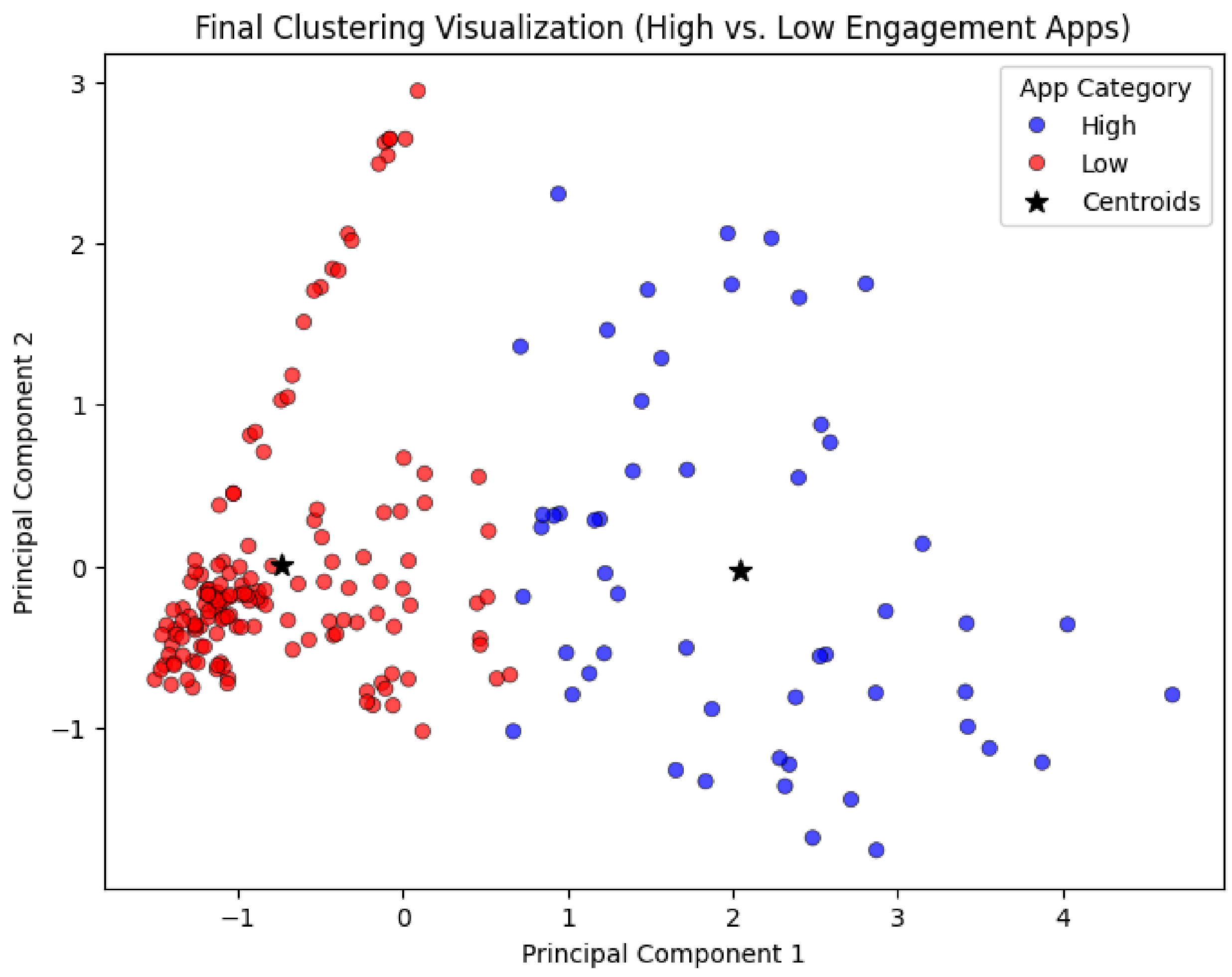








The figures illustrate the algorithm's performance over successive iterations, achieving convergence as early as Iteration 5. This rapid convergence indicates that the algorithm quickly identified an optimal or near-optimal distribution within a limited number of steps. Such early stabilization suggests either a relatively simple solution space for this instance or that the initial population was already positioned near the optimal region. The quick convergence underscores the algorithm's efficiency and potential robustness in handling specific problem cases, contributing to its overall computational effectiveness.

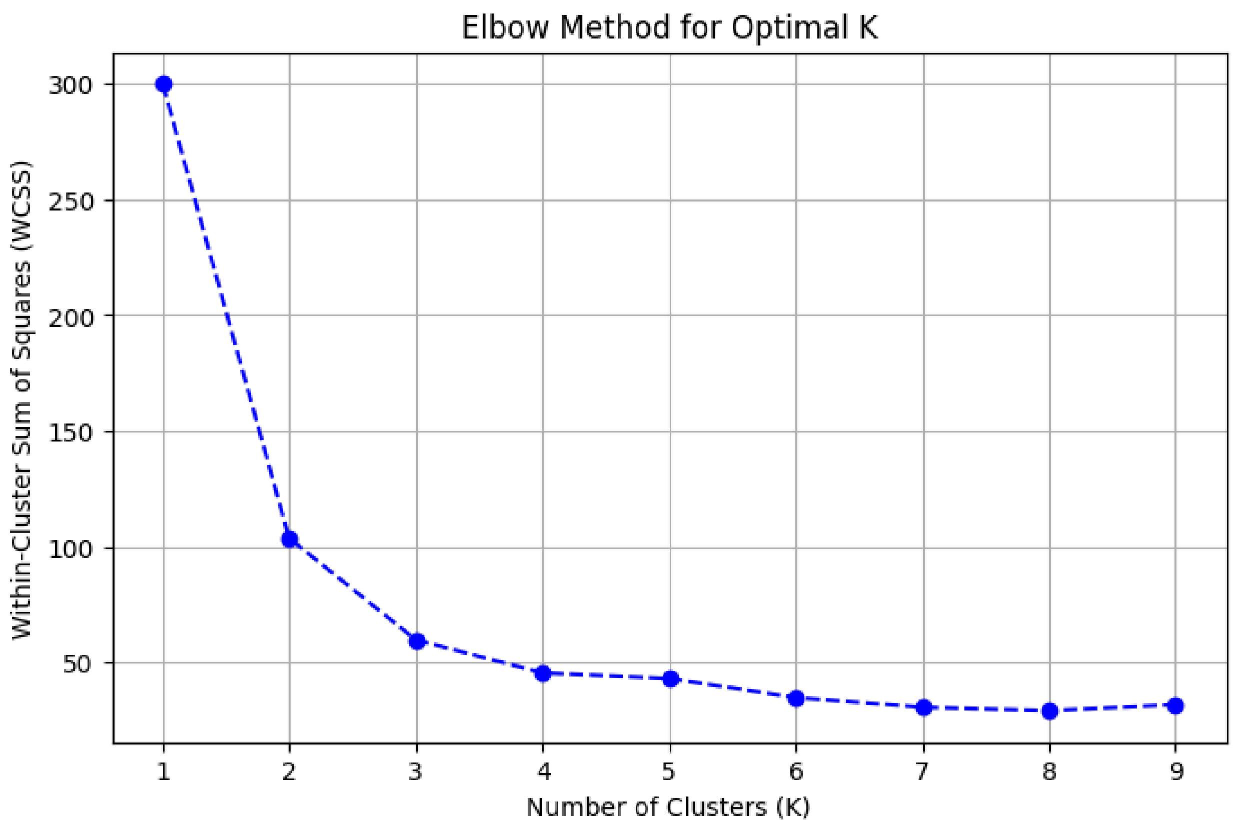


This scatter plot represents the clustering of apps based on two **Principal Components** (derived from PCA for dimensionality reduction). The black stars represent the cluster centroids, meaning the average position of points in each cluster.

IV. Modified dataset (K=2)

In this dataset, we tried using K=2 for clustering, but instead of just using the Usage (minutes) column, we decided to group apps into High Engagement and Low Engagement categories. This way, the algorithm focuses on interaction patterns instead of just screen time.

To do this, we picked two apps with the most engagement and two with the least engagement. We used the same labels for both the original and modified datasets so we could compare how well K-Means performs. This also helps us see if outliers affect the clustering results.

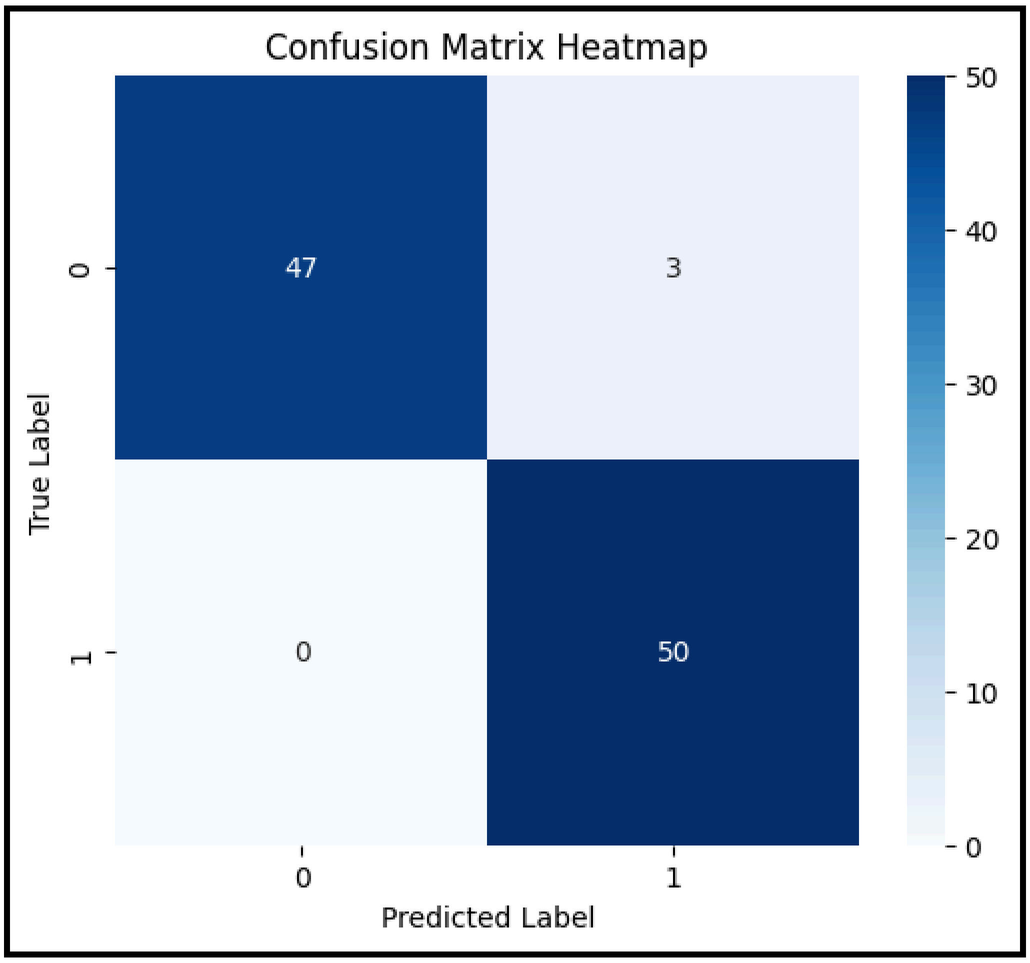


In the Elbow Method graph, we can see that the biggest drop in WCSS happens between K=1 and K=2, which makes K=2 a reasonable choice. While increasing K further still reduces WCSS, the decrease is not as steep, meaning adding more clusters might not make much of a difference.

Since we are only trying to group the apps into High Engagement and Low Engagement, sticking with K=2 makes sense. This way, we can keep things simple and still compare how well the clustering works compared to the original dataset.

After running the algorithm, we assign the clusters to their correct labels (High Engagement and Low Engagement) and check how well the predicted labels match the true labels. We then measure the results using the following metrics:

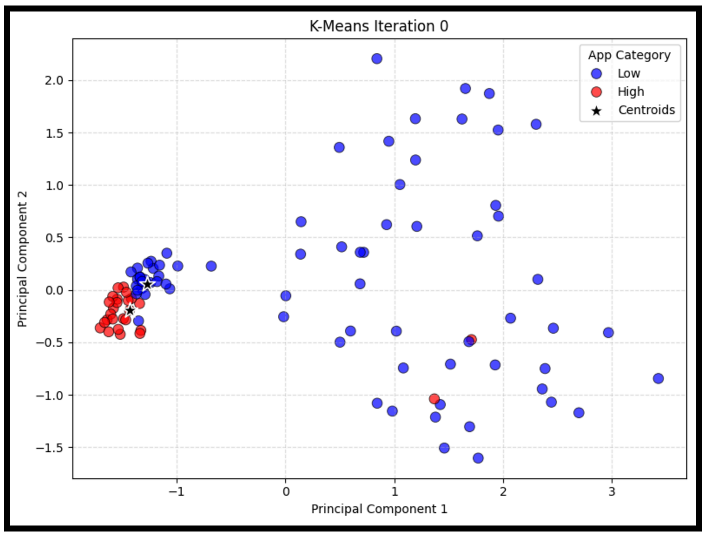
- **Accuracy: 0.9700**
- **Precision: 0.9717**
- **Recall: 0.9700**
- **F1 Score: 0.9700**

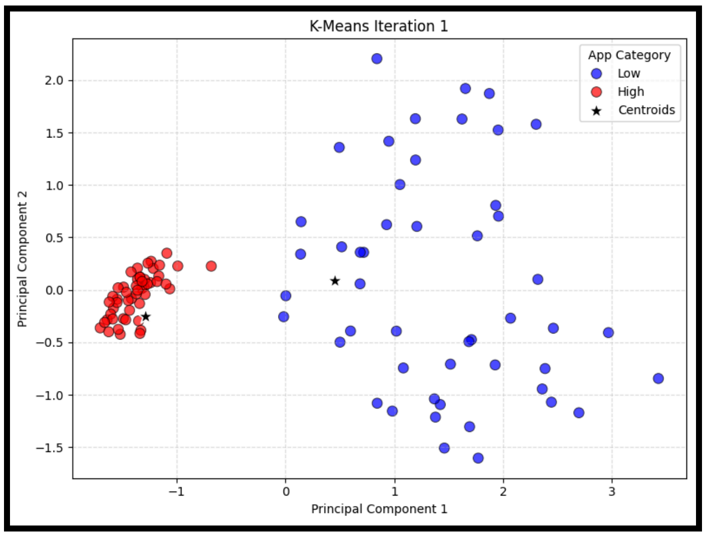


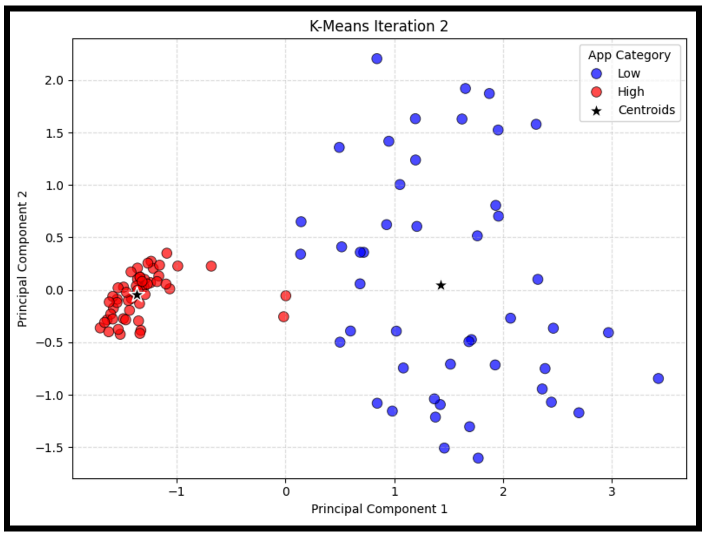
- True Negatives (47): Correctly predicted class 0 when it was actually 0.
- False Positives (3): Incorrectly predicted class 1 when it was actually 0.
- False Negatives (0): Incorrectly predicted class 0 when it was actually 1.
- True Positives (50): Correctly predicted class 1 when it was actually 1.

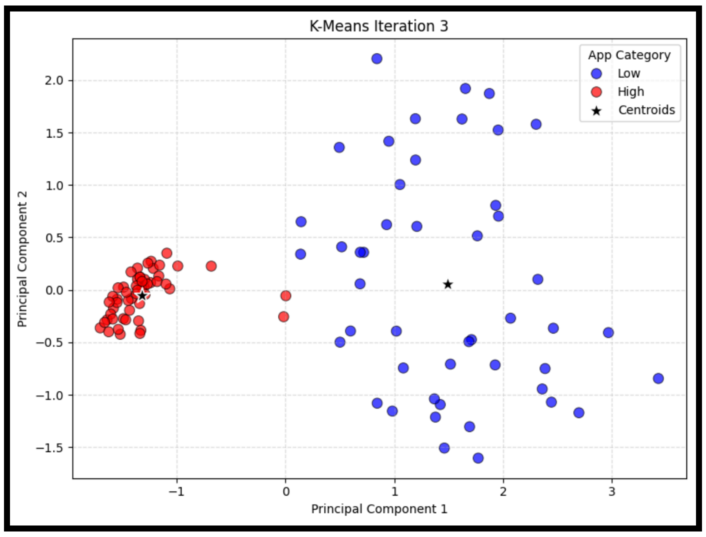
Confusion Matrix Interpretation

	Predicted: 0	**Predicted: 1**
Actual: 0	**47 (True Negative)**	**3 (False Positive)**
Actual: 1	**0 (False Negative)**	**50 (True Positive)**

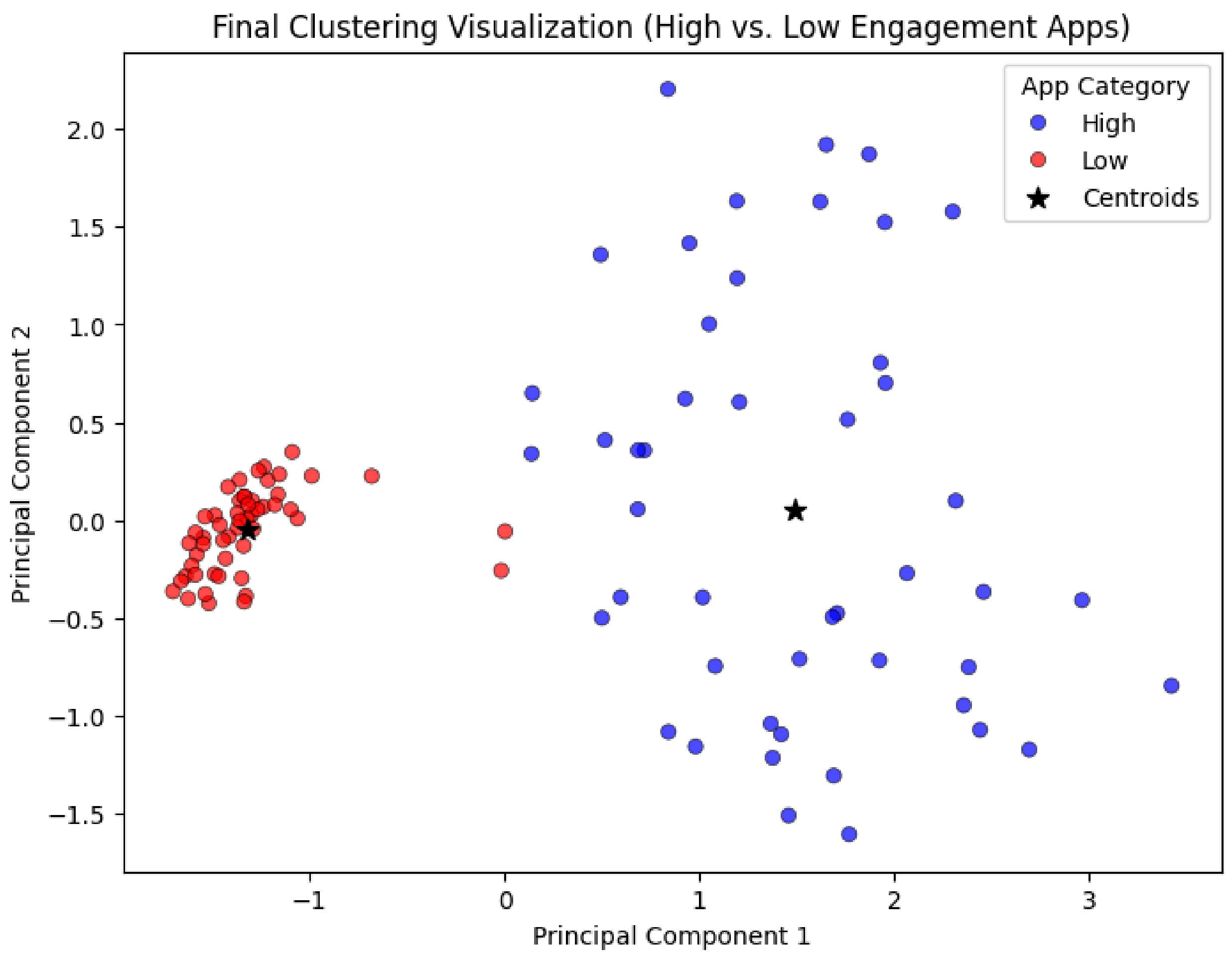








The figures display the algorithm's performance across each iteration, immediately reaching convergence at Iteration 3. This indicates that the algorithm was able to rapidly identify an optimal or near-optimal distribution within just a few iterations. Such early convergence suggests a relatively straightforward solution landscape for this particular instance, or that the initial population was already close to the optimal region. The swift convergence highlights the efficiency and potential robustness of the algorithm in handling certain problem scenarios, making it computationally efficient.



This scatter plot represents the clustering of apps based on two **Principal Components** (derived from PCA for dimensionality reduction). The black stars represent the cluster centroids, meaning the average position of points in each cluster.

Interpretation and Learnings

The group initially chose K = 3 based on a misinterpretation of the elbow graph, which led to the assumption that three clusters would best represent the data. Moreover, the group decided to generate true labels rather than manually assigning them.

Part 1:

Original Dataset, K=3

In comparing the True Label and Predicted Label of the data, the confusion matrix indicated a **60.50% accuracy** with the following key observations:

- **0 False Positives for 2:** This means that the 2nd label has not incorrectly classified the predicted values for the true value 2.
- **0 False Negatives for 0:** This means that no actual data was misclassified as 1 or 2.
- Class 0 was well-predicted, but there were **many false positives**, meaning other points were mistakenly grouped into class 0.
- **Class 1 had very poor recall (only 13 TP vs. 51 FN), indicating that most true class 1 instances were misclassified.** The model struggled to distinguish class 1 from the others.

- While there were **no false positives for class 2**, a significant number of actual class 2 points were misclassified as other classes, leading to high false negatives.
- **Precision: 64.85% and Recall: 60.50%** indicates a struggle in identifying the labels correctly.
- **F1 Score: 57.28%** shows a balance of precision and recall, suggesting some degree of misclassification.

Modified Dataset, K=3

In comparing the True Label and Predicted Label of the data, the confusion matrix indicated a **69% accuracy** with the following key observations:

- **0 False Positives for 2:** This means that the 2nd label has not incorrectly classified the predicted values for the true value 2.
- **0 False Negatives for 0:** This means that no actual data was misclassified as 1 or 2.
- Class 0 was well-predicted, but there were **many false positives**, meaning other points were mistakenly grouped into class 0.
- Similar to the original, while there were **no false positives for class 2**, a significant number of actual class 2 points were misclassified as other classes, leading to high false negatives.
- **Precision: 72.45% and Recall: 69%** indicates a better result compared to the original although it still shows a moderate struggle in identifying the labels correctly.
- **F1 Score: 67.89%** shows a better balance of precision and recall, suggesting some degree of misclassification.
- **Improved compared to the original dataset** as mostly this modified dataset for K = 3 shows better results or percentages with similar patterns for the confusion matrix.

Part 2:

The successful clustering of High and Low Engagement apps along their Principal Component suggests that the model effectively captures meaningful patterns in the data.

The final clustering visualization using PCA confirmed that the dataset naturally separates into two distinct groups:

- **Low Engagement Apps (Red Cluster)** on the left, representing apps with low user engagement.
- **High Engagement Apps (Blue Cluster)** on the right, representing apps with high user engagement.
- The **centroids** were positioned near the densest areas of their respective clusters, supporting the validity of the clustering approach.
- The **x-axis (Principal Component 1)** likely represents the dominant factor separating Low and High engagement apps.
- The **y-axis (Principal Component 2)** captures some variance but is less influential in distinguishing the groups.
- A couple of red points (**low engagement apps**) are scattered closer to the blue cluster and vice versa, these could be **hybrid apps** (e.g., apps that have fluctuating low-high or moderate user engagement.)

Original Dataset, K=2

In using the Original Dataset with K = 2, the results confirm that High and Low Engagement apps naturally form two distinct clusters, making K=2 an optimal choice. The Elbow Method graph showed a significant drop in the Within-Cluster Sum of Squares (WCSS) at K = 2, suggesting that two clusters effectively balance model simplicity and variance explanation. While additional clusters (K > 2) continued reducing WCSS, the diminishing returns indicated that K = 2 was the most reasonable choice.

Upon comparing the True Label and Predicted Label of the data, the confusion matrix indicated a significant classification **accuracy** at **93.50%,** with some key observations:

- **High True Positives (142 cases)**: Many true Low Engagement apps were apps correctly classified .
- **Low False Negatives(5 cases)**: Very few High Engagement were apps incorrectly classified as Low Engagement.
- **Precision (93.68%) and Recall (93.50%):** These scores are slightly lower than the overall accuracy, however, they still indicate that the model is performing well in identifying the True Labels of the apps.
- **The F1-score (93.56%):** This score confirms a solid balance between precision and recall, reinforcing that the model is making consistently accurate and precise predictions while only having a few misclassifications.

Modified Dataset, K=2

Lastly, let’s discuss the results we got from Modified Dataset K=2.

In comparing the True Label and Predicted Label of the data, the confusion matrix indicated a **97% accuracy** with the following key observations:

- **High True Positives (50 cases):** Many true High Engagement apps were correctly classified.
- **Low False Negatives (0 cases)**: No High Engagement apps were misclassified as Low Engagement.
- **Low False Positives (3 cases)**: Only a few Low Engagement apps were incorrectly classified as High Engagement.
- **Precision (97.17%)** and **Recall (97.00%)**: These scores indicate that the model consistently identifies the correct engagement category with high accuracy.
- The **F1-score (97.00%)**: This confirms a strong balance between precision and recall, meaning the model performs highly accurately with minimal misclassification.

Conclusion

Be cautious when analyzing the elbow graph, as depending purely on visualization may introduce subjectivity or inconsistency. The "elbow" point, indicating the ideal number of clusters, is typically identified where the decline in the cost function (e.g., inertia or SSE) begins to level off. However, this can be unclear, particularly when the curve lacks a distinct inflection. For a more precise and objective selection, consider applying a mathematical approach like the knee point detection algorithm. A reliable method involves computing the smallest angle between every three consecutive points on the graph.

Performance metrics (Accuracy, Precision, Recall, F1 score) for K-Means with K=2 were notably better on the modified dataset compared to the original. This is largely due to the modified data's characteristics: clearer separation between the chosen 'true label' groups and fewer outliers. While this scenario uses labeled data for evaluation, unlike typical unsupervised learning, it effectively demonstrates a critical lesson: a machine learning engineer's expertise lies in understanding data properties and strategically modifying datasets to maximize the efficacy of chosen algorithms.