Assignment-2

we are using the “word-embeddings.feather” dataset. The dataset pertain to 512 dimensional word embedding. The first column in each row contains a word , while the second column contains the corresponding 512-dimensional embedding generated by VIT.

3) **K-Means Clustering**:

3.1) Here implementing the K-Means class

Explanation:

- The K-Means clustering algorithm partitions the dataset into k clusters by minimizing the **Within-Cluster Sum of Squares (WCSS)**, which measures the variance within each cluster.
- Your K-Means class should include the following methods:
    - **fit()**: This method finds the optimal cluster centroids for the given value of k. It iteratively updates centroids by assigning points to the nearest centroid and then recalculating the centroids based on these assignments.
    - **predict()**: Once the centroids are fixed (after the fit() method), this method assigns each data point to the nearest centroid.
    - **getCost()**: This returns the WCSS, which measures how tightly the points are clustered around their respective centroids.
- **Fit Function Calculations**:

1. Initialize k centroids randomly.
2. For each data point, calculate the Euclidean distance to each centroid.
3. Assign the point to the nearest centroid.
4. Update each centroid as the mean of the data points assigned to it.
5. Repeat the process until convergence (i.e., no further changes in cluster assignments).

- **Predict Function**:
    - After the centroids are determined, for each new data point, the algorithm assigns it to the nearest centroid by calculating the Euclidean distance.

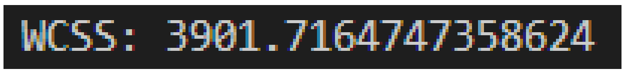
WCSS Calculation:

$$\mathrm{WCSS} = \sum_{i=1}^{k} \sum_{x_j \in C_i} ||x_j - \mu_i||^2$$

Where:

- xj is a data point,
- µi is the centroid of cluster Ci,
- $||xj−µi||^2$ is the squared distance between the data point and the centroid.

Result:



```
WCSS: 3901.7164747358624
```

3.2)

Determine the Optimal Number of Clusters

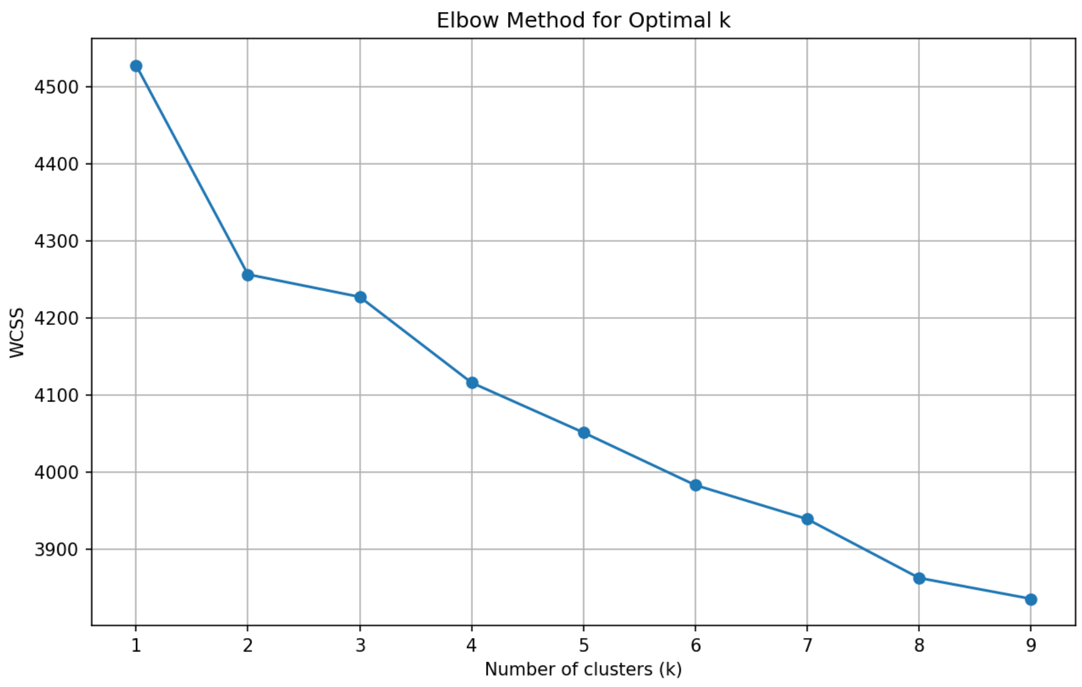
- **Explanation**:
    - The **Elbow Method** helps determine the optimal number of clusters (k) by plotting the WCSS for different values of k and identifying the "elbow" point where the rate of decrease in WCSS starts to slow.
    - As you increase k, WCSS decreases because clusters become smaller and more specific, but at some point, the reduction in WCSS becomes marginal. This point is the "elbow," and it indicates the optimal number of clusters.
    - From your plot, the WCSS starts at around 4500 for k=1 and decreases gradually as k increases.

- **Observations from Elbow Method Plot**:
    - The elbow point appears to be around **k=3** or **k=4**. After this point, the WCSS still decreases, but the rate of decrease slows down significantly, meaning adding more clusters does not improve the model much.
    - This suggests that the optimal number of clusters for the 512-dimensional dataset could be **3 or 4**, since adding more clusters doesn't significantly reduce the WCSS.

Observations for Each k Value:

- **For k=1**: The WCSS is the highest because all data points are assigned to one cluster, meaning all the variance in the data is concentrated in one centroid.
- **For k=2**: The WCSS drops significantly, as two clusters begin to capture some of the data's variance.
- **For k=3 to k=4**: The WCSS continues to drop, indicating that adding clusters helps explain more variance.
- **For k > 4**: The WCSS still decreases, but the rate of reduction is marginal, meaning the additional clusters are not adding substantial information about the data structure

Plot:



Optimal k: Based on the elbow plot, it seems like k=3 or k=4 is optimal for our dataset. And then we are performing clustering with k=3 .
so here $K_{kmeans1}=3$

5) Dimensionality Reduction and Visualization :

PCA (principal component analysis):

Principal Component Analysis (PCA) is the general name for a technique which uses sophisticated underlying mathematical principles to transform several possibly correlated variables into a smaller number of variables called principal components.

- It is a technique used to reduce the dimensionality of data while preserving as much variance as possible.

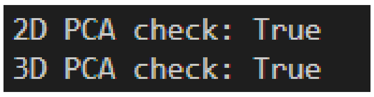
=>**Fit Method**: The fit method calculates the principal components of the dataset. The principal components are the eigenvectors of the covariance matrix of the data, and they define the directions of maximum variance.

=>**Transform Method**: The transform method projects the data onto the principal components.

=>**Check PCA Method**: The checkPCA method verifies if the PCA implementation is working correctly by comparing the explained variance ratio.

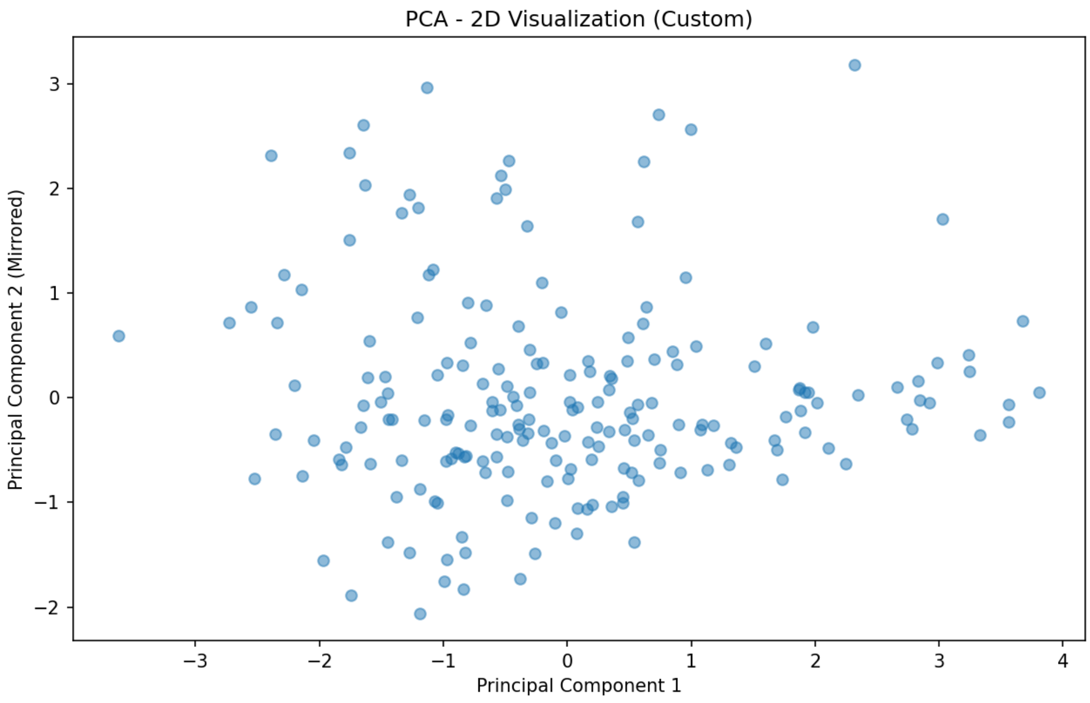
5.2) Perform Dimensionality Reduction:

Verifying the functionality of the class using the checkPCA method then we are getting the result as



```
2D PCA check: True
3D PCA check: True
```

Visualize the data in 2D:



Explanation:

- The PCA algorithm reduced the word embeddings from 512 dimensions to 2.
- The scatter plot visualizes the transformed data in a 2D plane with the principal components as axes.
- **Principal Component 1** and **Principal Component 2** are the new axes of the transformed data, capturing the directions of maximum variance.

Observations from the 2D plot:

- There doesn't appear to be any strong visible linear pattern.

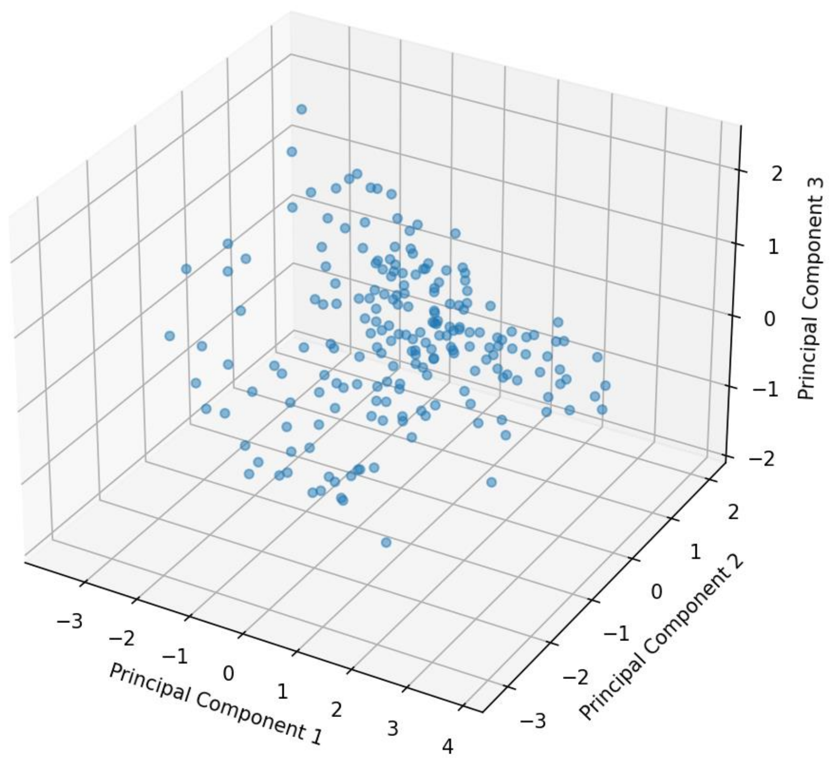
- Data points seem scattered with some minor clusters in the center.
- A few outlier points are further from the center, especially in the top right and left sections of the graph.
- The data is fairly symmetrical around the origin, with no obvious skew toward any particular direction.

Estimating clusters (k2):

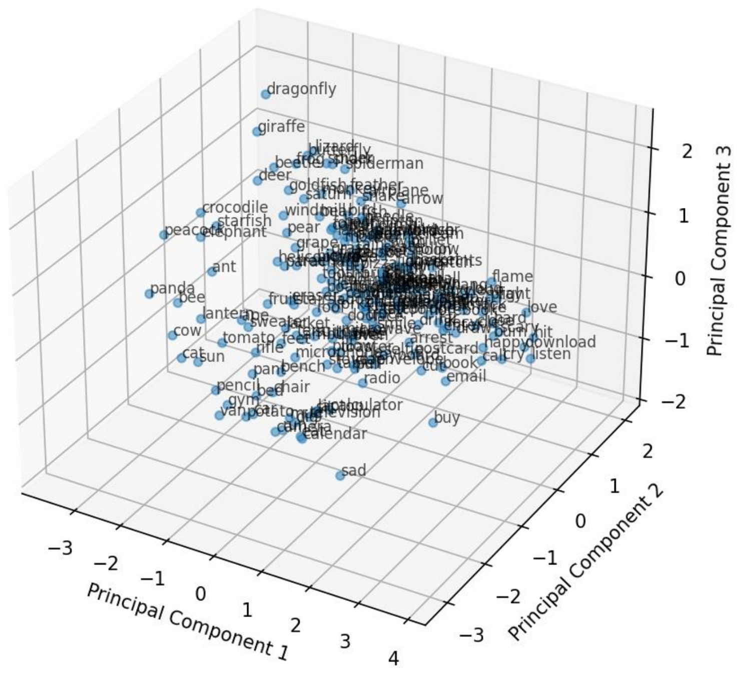
- Based on the density of the points, you could estimate around 3-4 clusters. There seem to be some denser regions near the center, but it is hard to say definitively without clustering methods. The points are loosely clustered around the center with some more spread out.

- Visualize the data in 3D:

PCA - 3D Visualization (Custom)



PCA - 3D Visualization with Word Labels (Custom)



5.3)

Data Analysis

1. **Interpreting the New Axes**:
   - **Principal Components** represent directions in the feature space where the variance of the data is maximized.
   - The first principal component captures the direction of maximum variance.
   - The second principal component captures the next highest variance orthogonal to the first one, and so on.
2. **Examining Plots for Patterns or Clusters**:
   - **2D Plot**: Look for distinct groups or clusters of points. If the points seem to form separate groups, these groups might represent clusters.
   - **3D Plot**: Provides a more detailed view, potentially revealing more complex structures.

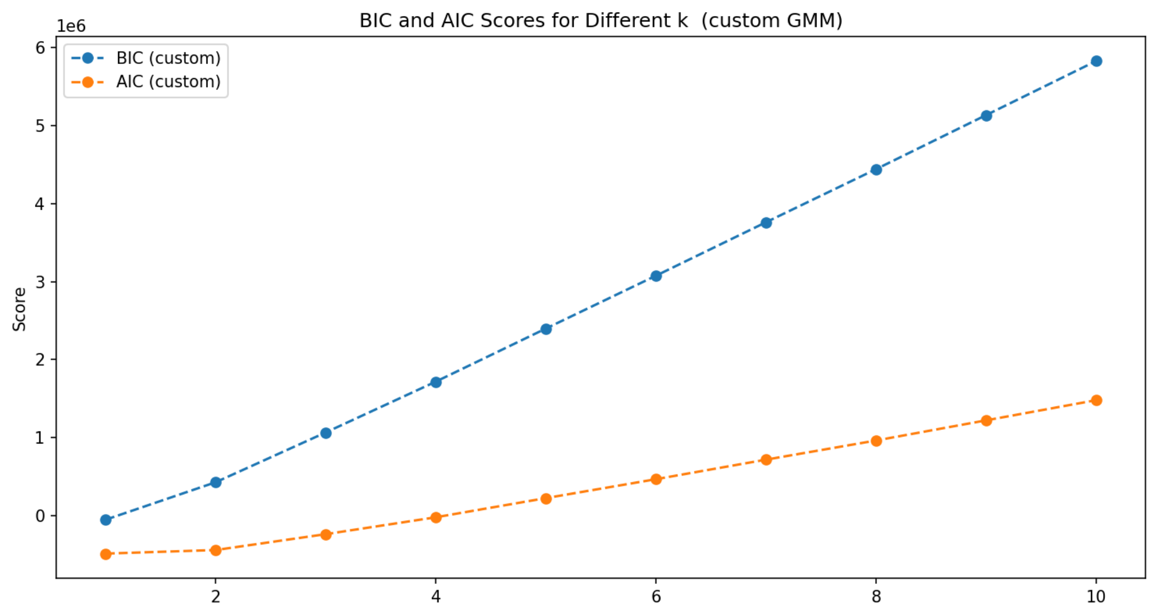
Observations:
* **Dimensionality reduction** via PCA effectively projects the data into a 2D space, which allows us to observe high-dimensional word embeddings in a simplified form.

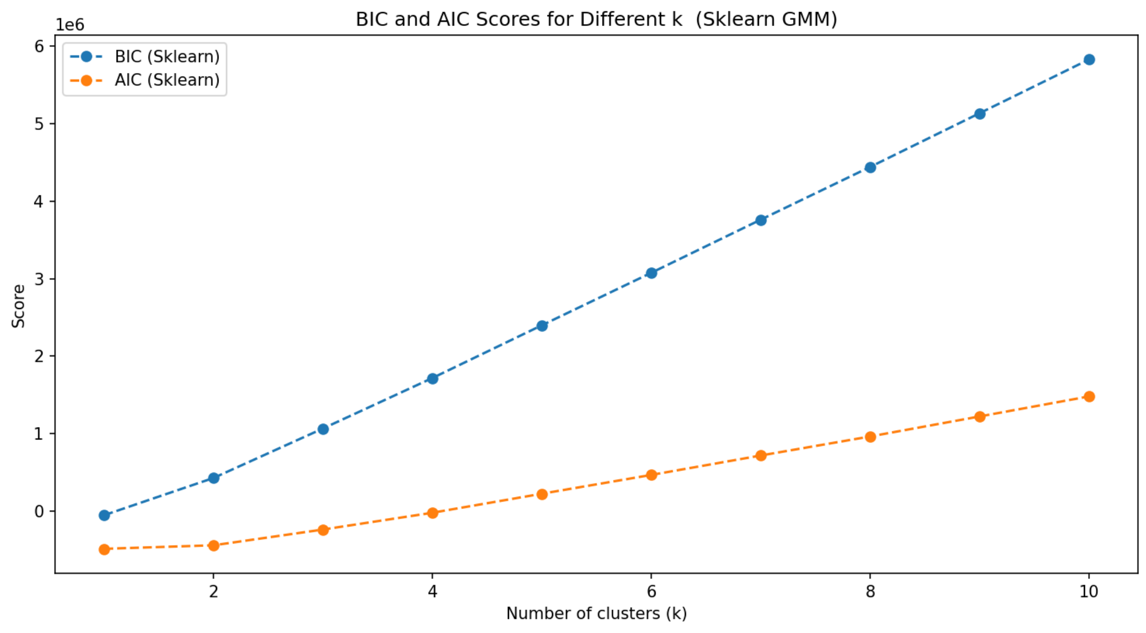
* The **scatter plot** doesn't show very clear clusters, but there are some indications of density in the center, along with some outliers.

* Compare this plot with a 3D visualization to see if more patterns or clusters become apparent.

* Use clustering algorithms like K-means to automatically identify and confirm the number of clusters.

4)





8). Hierarchical Clustering:

* It is a popular method for grouping objects.

*It creates groups so that objects within a group are similar to each other and different from objects in other groups.

1. Apply Hierarchical Clustering to the Dataset

Step 1: Compute the Linkage Matrix

We will use the scipy.cluster.hierarchy module to compute the linkage matrix. The linkage matrix contains information about the hierarchical clustering, where each row represents the merging of two clusters at a specific step. To calculate this, we need to choose a linkage method ('single', 'complete', 'average', 'ward') and a distance metric (Euclidean ,Manhattan,cosine).

* linkage(X, method='ward'): Computes the linkage matrix using Ward's method, which minimizes the variance within clusters.

* dendrogram(Z): Plots the dendrogram representing the hierarchical structure of clusters.

2. Experiment with Different Linkage Methods

We will experiment with other linkage methods such as 'single', 'complete', and 'average' to see how the linkage matrix changes

* Clusters are visually represented in a hierarchical tree called a dendrogram.

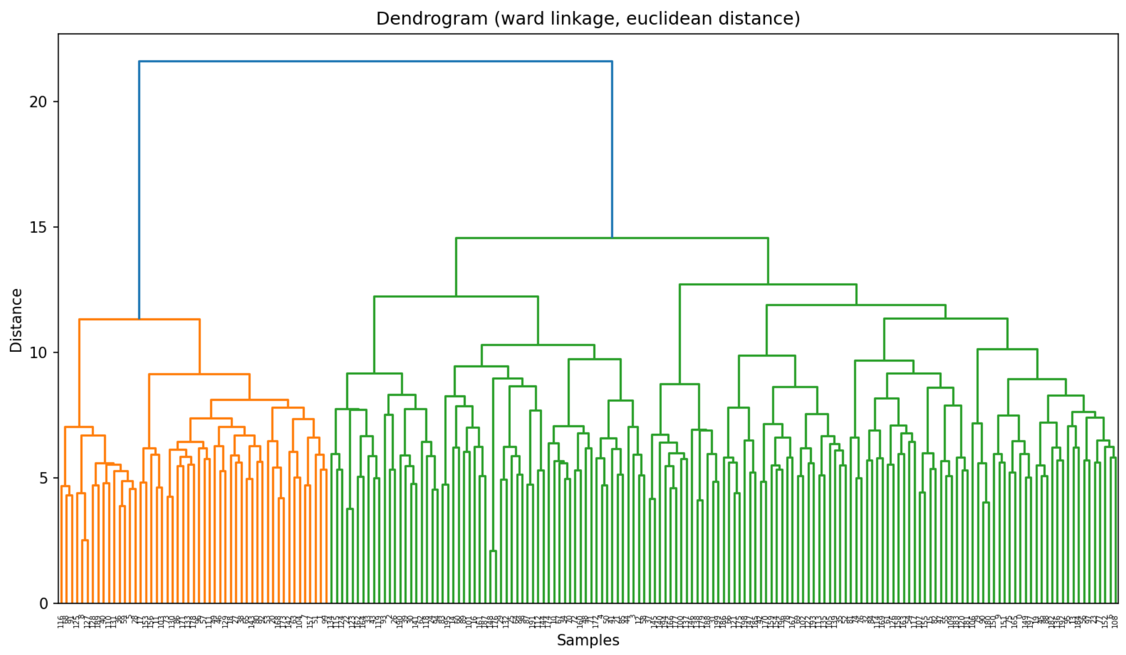
- **Single linkage**: Tends to form long, chain-like clusters, where the nearest points are merged.
- **Complete linkage**: Results in compact, spherical clusters, where the maximum distance between points in different clusters is minimized.
- **Average linkage**: Averages the distances between points, resulting in balanced clustering.
- **Ward linkage**: Minimizes the variance within clusters and typically performs better for most datasets.

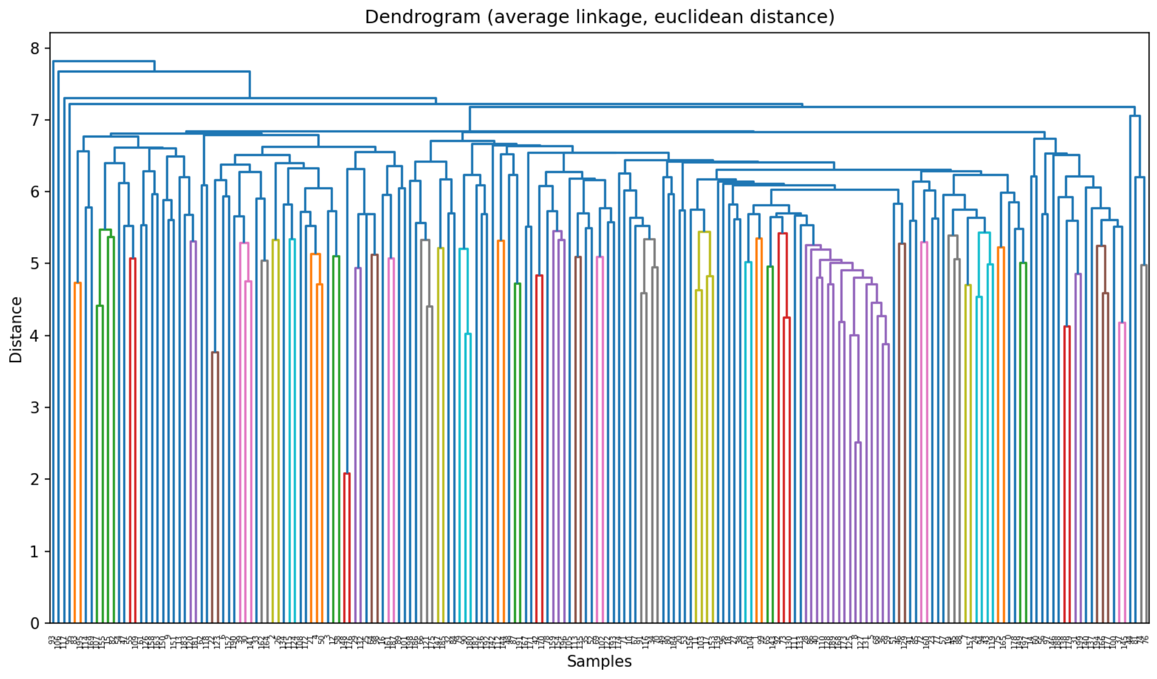
The linkage matrix (Z) is computed based on the pairwise distances between points and how clusters merge. Each row of Z contains:

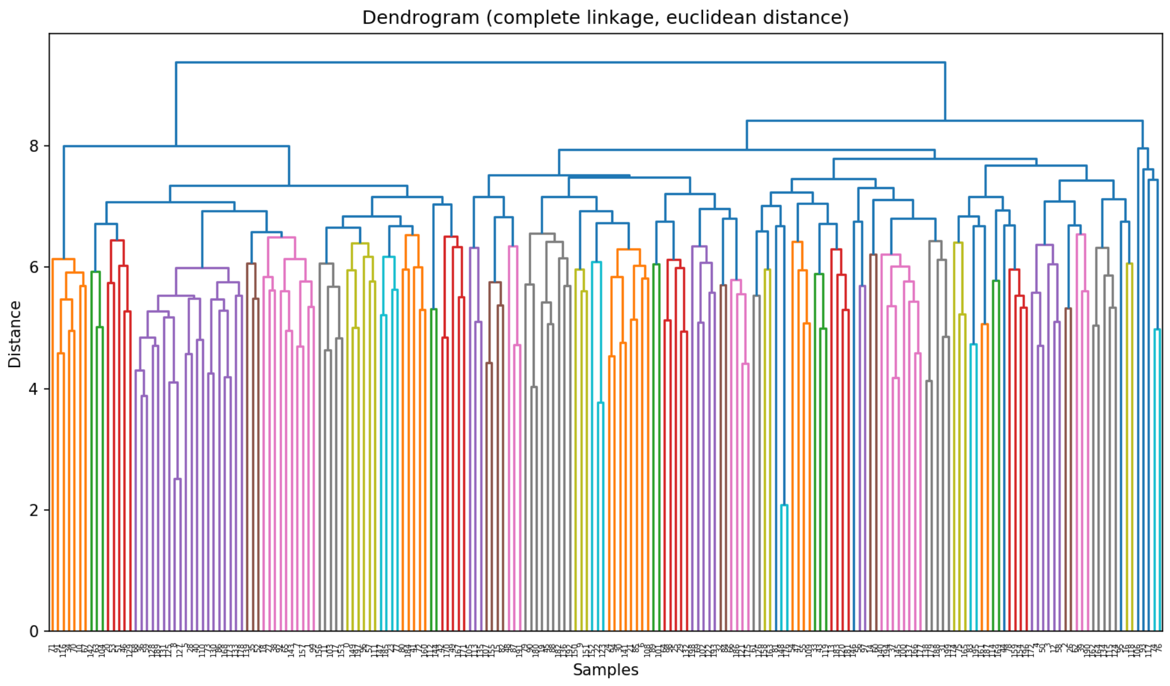
1. The indices of the clusters being merged.
2. The distance between these clusters.
3. The number of original data points in the new cluster formed by this merge.

3. Cutting the Dendrogram to Create Clusters

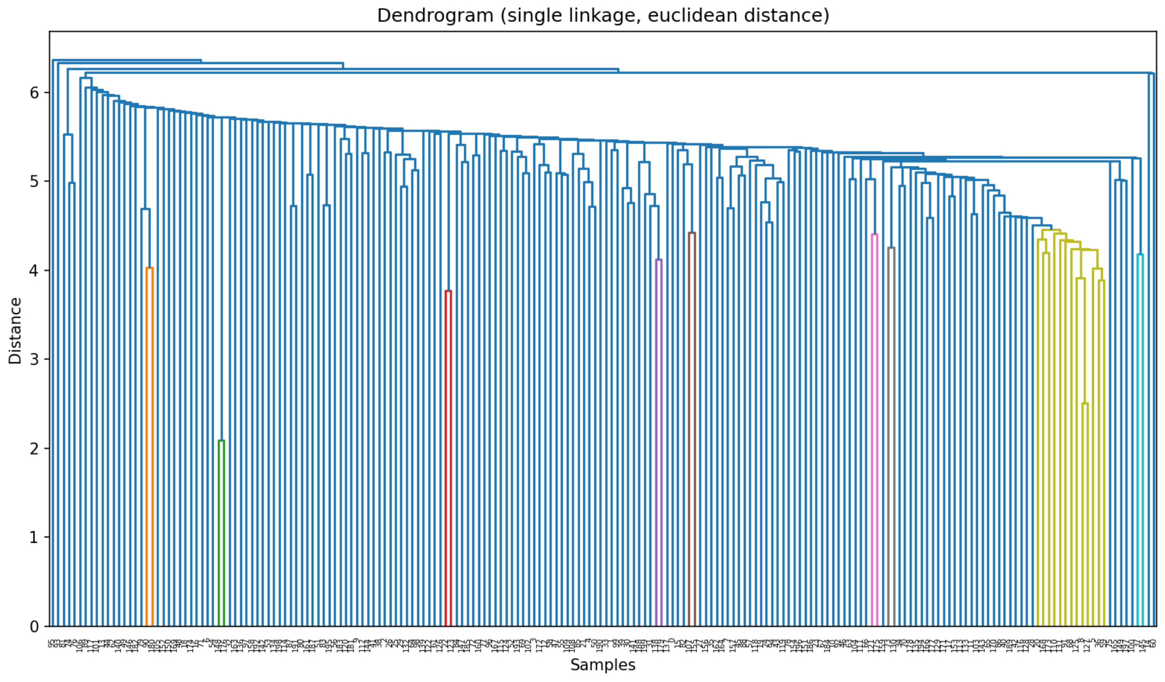
Once we have computed the linkage matrix and plotted the dendrogram, the next task is to cut the dendrogram to form clusters at specific points. We will use the kbest1 , kbest2 values to cut the dendrogram using the `fcluster()` function from `scipy.cluster.hierarchy`.


Dendrogram (ward linkage, euclidean distance)
Distance
Samples
0
5
10
15
20

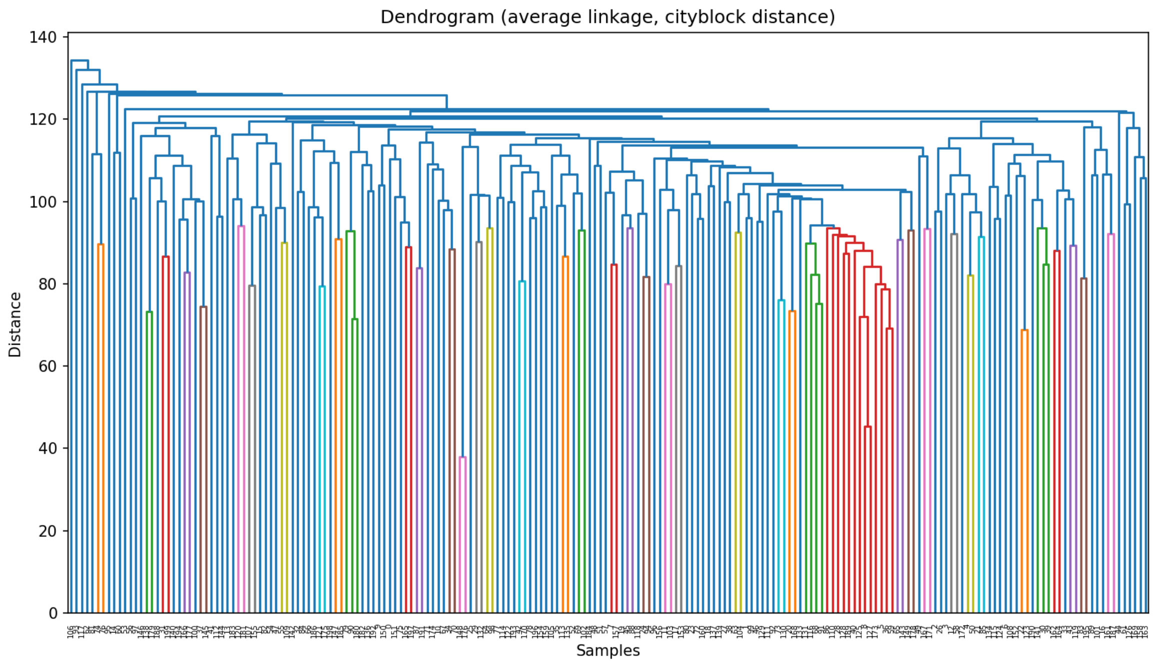

Dendrogram (average linkage, euclidean distance)
Distance
Samples
0
1
2
3
4
5
6
7
8

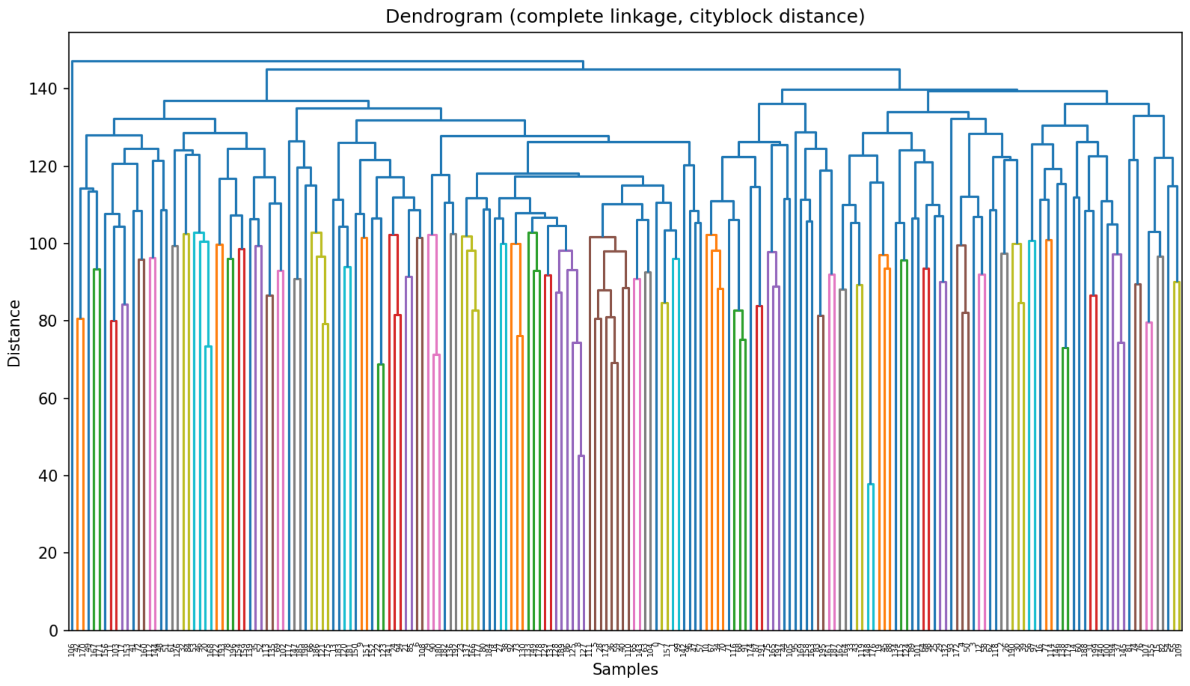


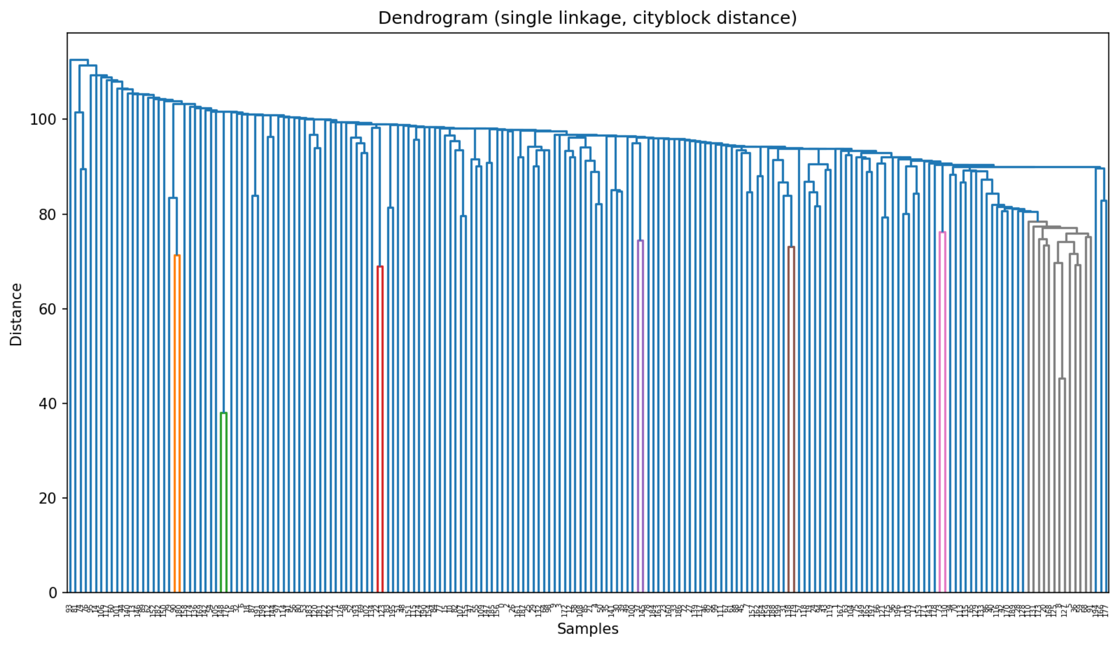
Dendrogram (complete linkage, euclidean distance)
Distance
Samples
0
2
4
6
8

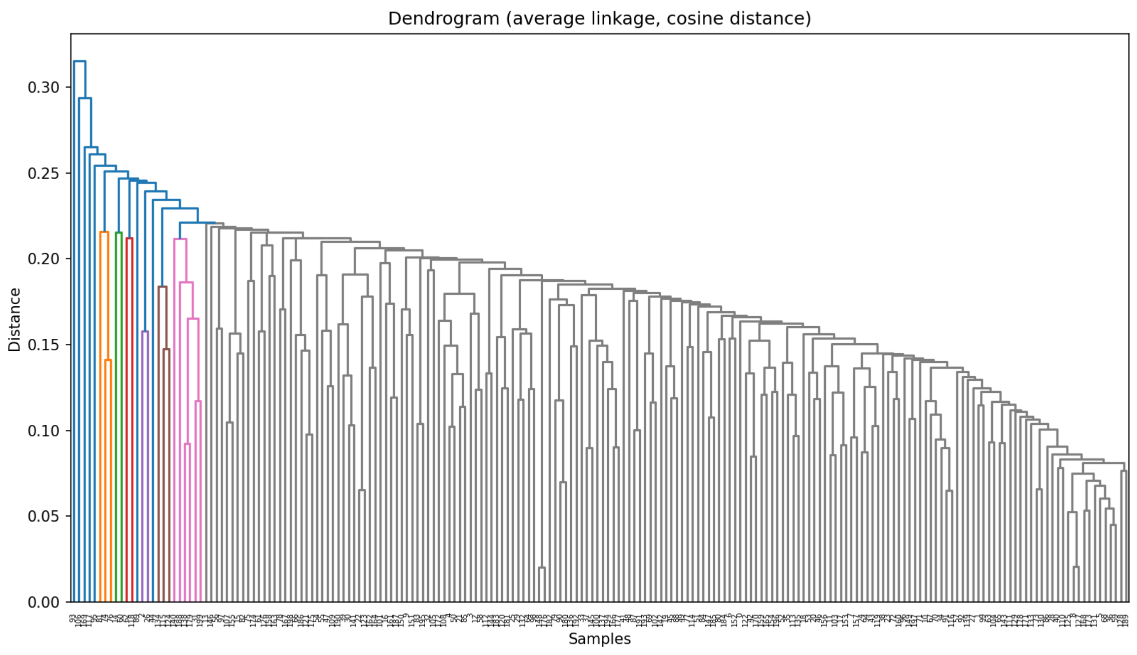


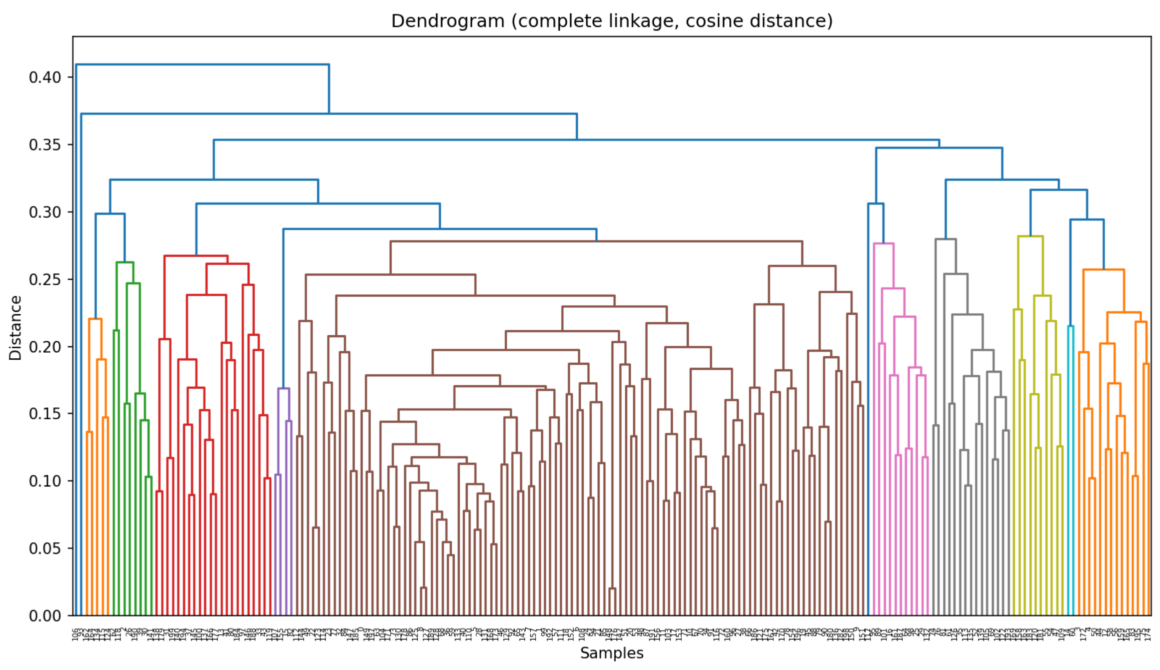
Dendrogram (single linkage, euclidean distance)
Distance
Samples
0
1
2
3
4
5
6

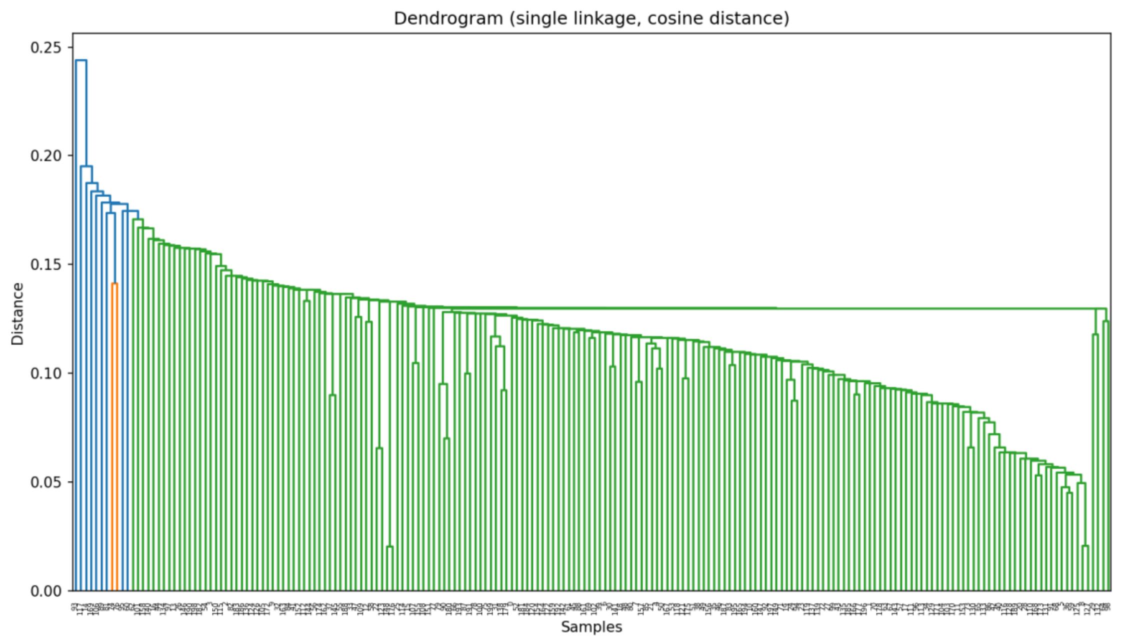

Dendrogram (average linkage, cityblock distance)
Distance
Samples
0
20
40
60
80
100
120
140


Dendrogram (complete linkage, cityblock distance)
Distance
Samples
0
20
40
60
80
100
120
140


Dendrogram (single linkage, cityblock distance)
Distance
0
20
40
60
80
100
Samples


Dendrogram (average linkage, cosine distance)
Distance
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Samples





Observations and Calculations

- **Cluster Comparisons**: After comparing the clusters from hierarchical clustering with K-means and GMM, you can analyze how similar or different the cluster groupings are.
    - If the clusters are similar, this indicates that hierarchical clustering has found similar groupings to those found by K-means and GMM.

- If the clusters are very different, it suggests that hierarchical clustering may be capturing a different structure in the data.

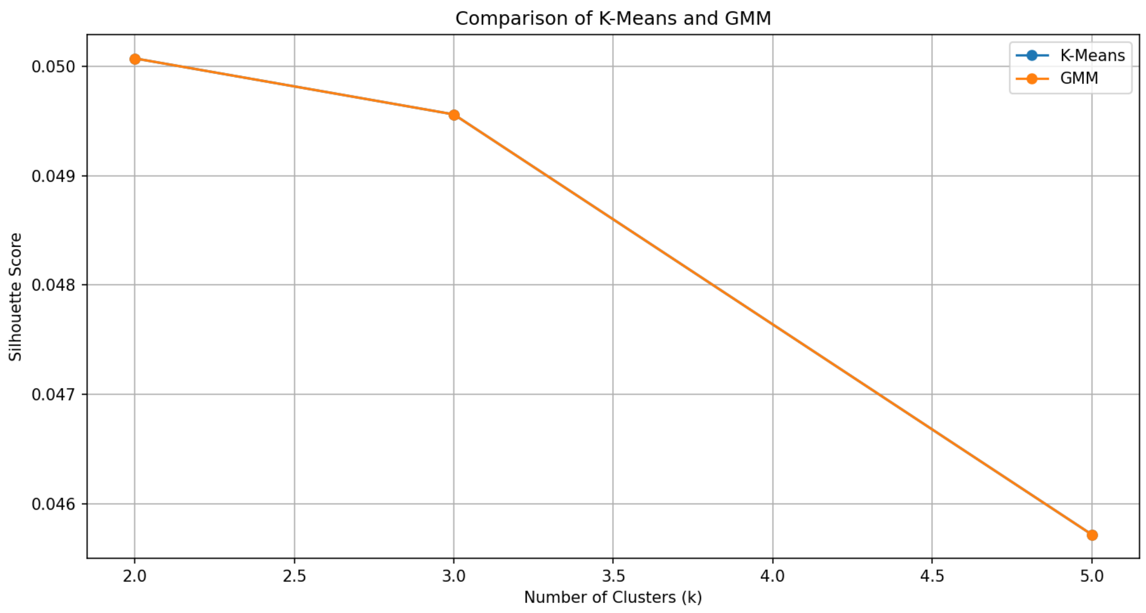
- **Distance Metric**: You used the Euclidean distance metric here (the default for Ward linkage). If necessary, you could experiment with other distance metrics (like Manhattan, Cosine) by passing them into the linkage() function as a parameter.
- **Linkage Method**: Based on your experiments with different linkage methods, you can identify the method that gives the best structure (e.g., most interpretable or most consistent with K-means or GMM results).

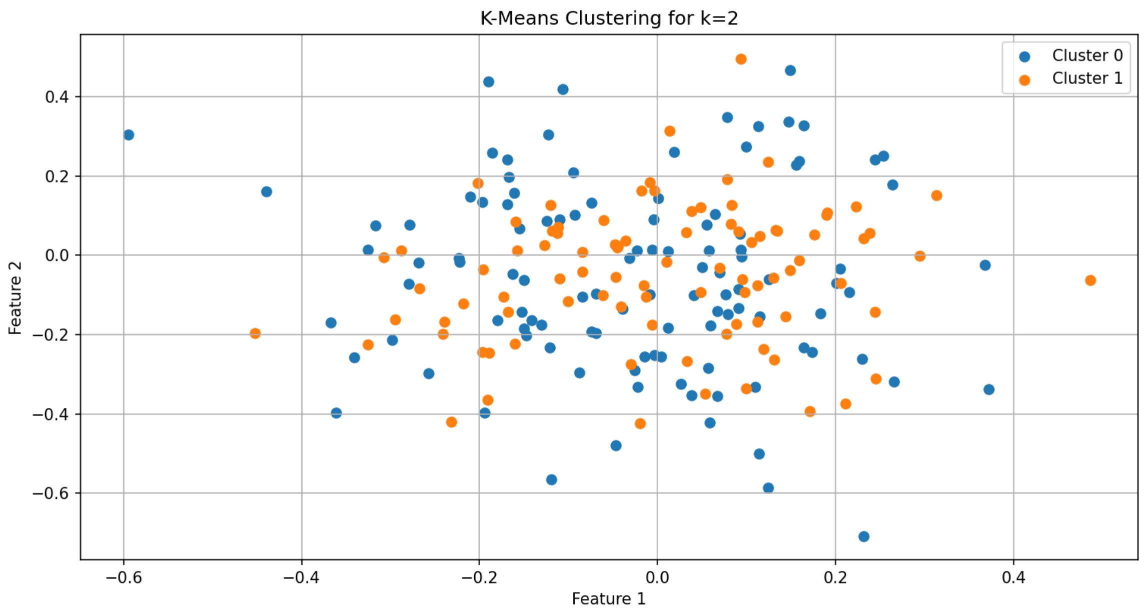
Conclusion

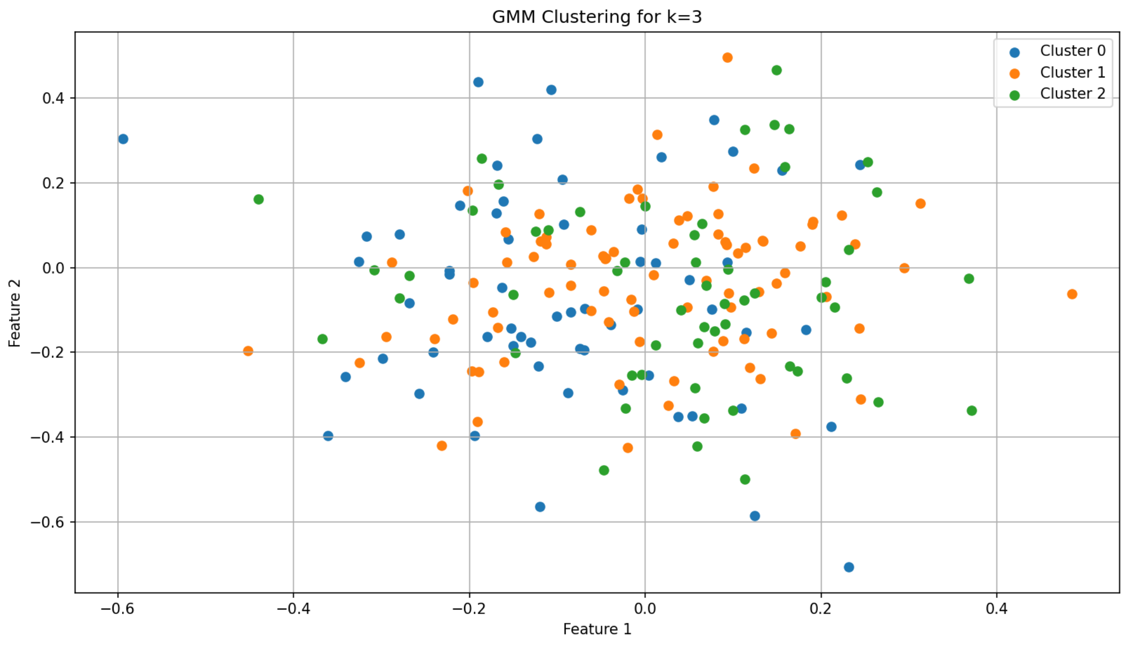
You have now:

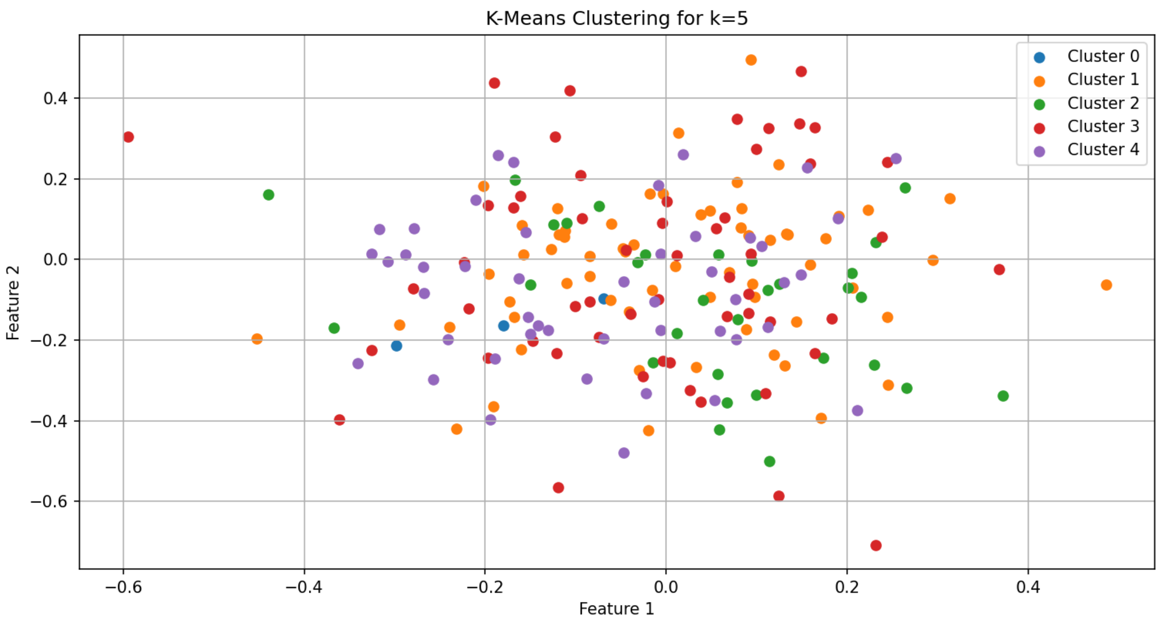
1. Applied hierarchical clustering and visualized dendrograms.
2. Experimented with different linkage methods and distance metrics.
3. Cut the dendrogram to form clusters and compared them with K-means and GMM results.

7.

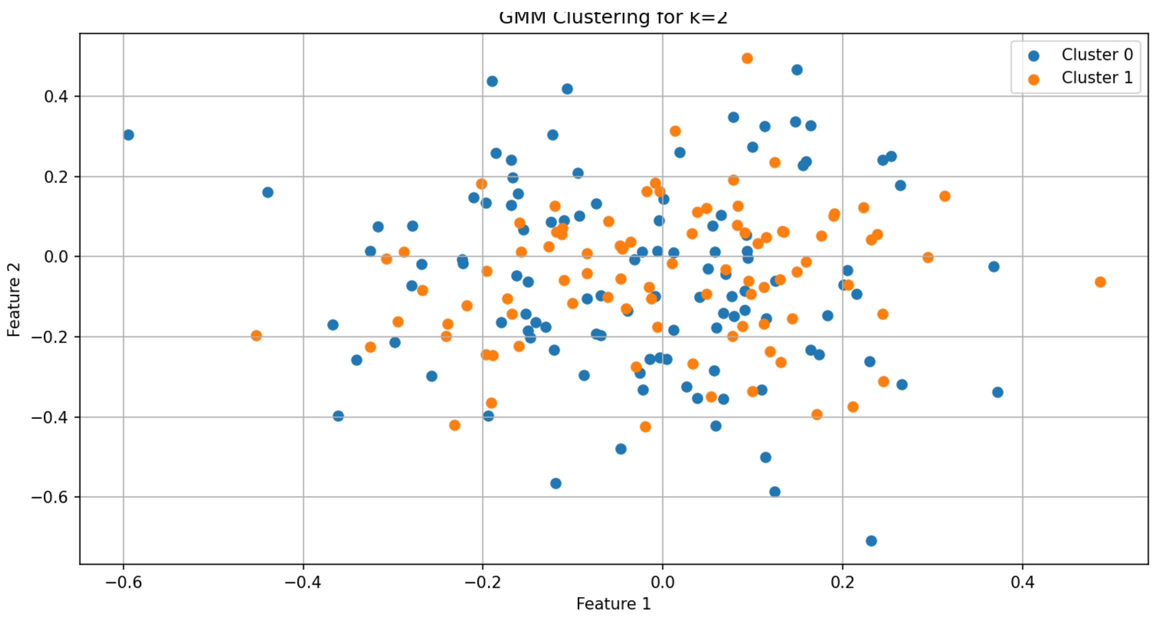



K-Means Clustering for k=2
Cluster 0
Cluster 1
Feature 1
Feature 2
−0.6
−0.4
−0.2
0.0
0.2
0.4

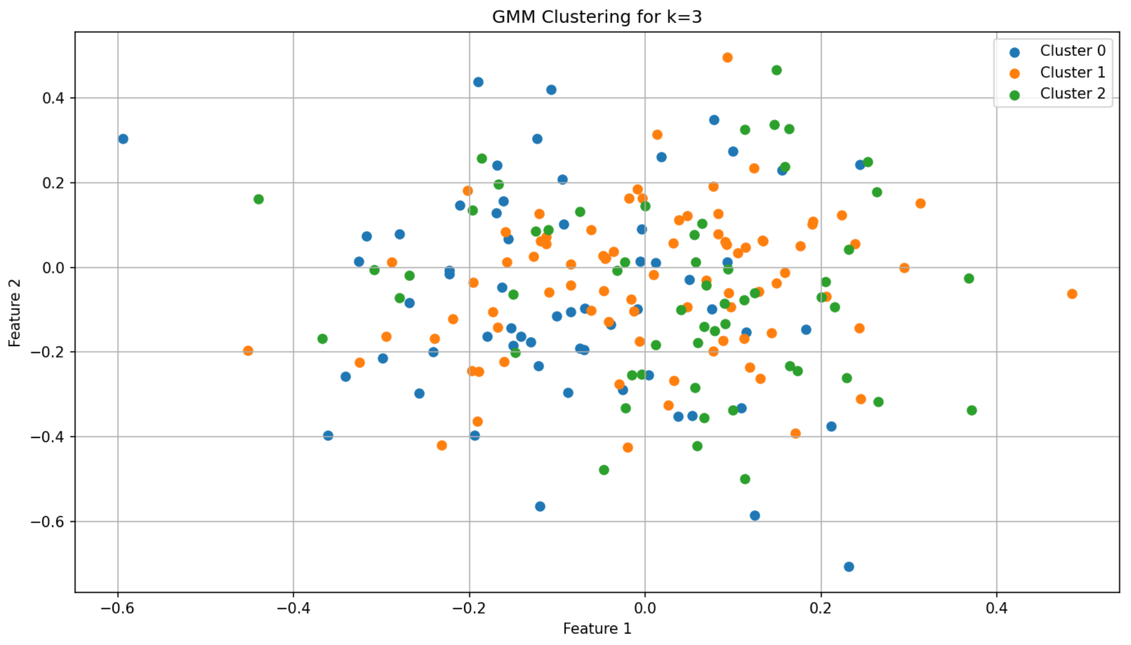

GMM Clustering for k=3
Cluster 0
Cluster 1
Cluster 2
Feature 1
Feature 2
−0.6
−0.4
−0.2
0.0
0.2
0.4

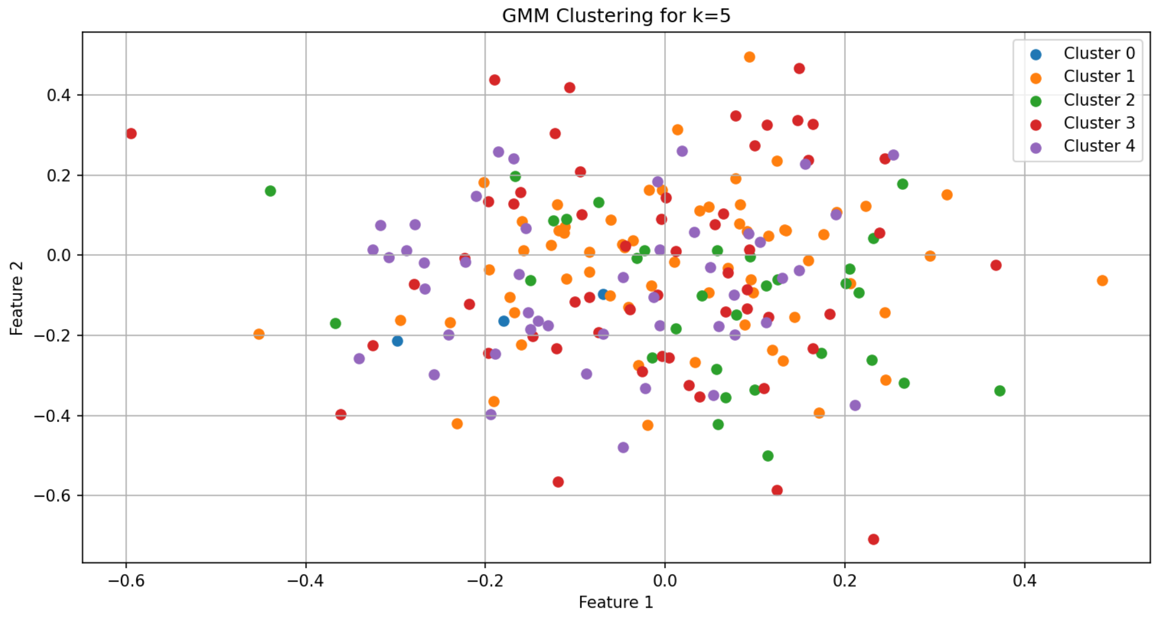


K-Means Clustering for k=5
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Feature 1
Feature 2



GMM Clustering for k=2
Cluster 0
Cluster 1
Feature 1
Feature 2





Observations:

* **Silhouette Scores**: Higher silhouette scores generally indicate more meaningful and well-separated clusters. Comparing these scores across different kkk values helps identify the optimal number of clusters.

* **Cluster Distribution**: Examining the number of points in each cluster helps understand the balance of clusters.

* **Visualization**: Visual inspection of the scatter plots can reveal how well-separated the clusters are and whether the clusters align with any meaningful groupings.

GMM Clustering Analysis

Steps Involved:

1. **Perform Clustering**:
    - Gaussian Mixture Models (GMM) are used to cluster the data. Unlike K-Means, GMM assumes data is generated from a mixture of several Gaussian distributions.
    - Each cluster in GMM is represented as a Gaussian distribution characterized by its mean and covariance.
2. **Evaluate Coherence**:
    - Similar to K-Means, the **Silhouette Score** is used to evaluate the coherence of the clusters.
3. **Examine Clusters**:
    - Analyze the clusters by counting the number of data points in each cluster.
4. **Visualize Clusters**:
    - Scatter plots for GMM clusters are generated to visualize the clusters in 2D space.

Observations:

- **Silhouette Scores**: GMM can potentially provide better clustering results if the underlying data distribution is closer to Gaussian. Compare these scores to identify the best kkk.
- **Cluster Distribution**: Analyzing cluster sizes helps in understanding the distribution of data points across clusters.
- **Visualization**: Visualization helps assess the shape and distribution of clusters, which can be more flexible in GMM compared to K-Means.

3. Comparing K-Means and GMM

Steps Involved:

1. **Compare Clustering Results**:

   - Compare the clusters obtained from K-Means and GMM for the best kkk values determined in previous steps.
   - Evaluate cluster coherence and separation.
2. **Assess Effectiveness**:
   - Compare silhouette scores and visualizations to determine which method provides more meaningful groupings.

Observations:

- **Silhouette Scores**: Compare the silhouette scores of K-Means and GMM to determine which method has better clustering quality.
- **Cluster Visualization**: Examine the scatter plots to see which method produces more distinct and meaningful clusters. GMM might handle non-spherical clusters better than K-Means, which assumes spherical clusters.

9.

Principal Component Analysis (PCA)

Objective: PCA is a dimensionality reduction technique used to transform high-dimensional data into a lower-dimensional form while preserving as much variance as possible. This is beneficial in improving the efficiency of machine learning models and reducing computational costs.

Steps for PCA:

1. **Standardize the Data**:
   - Center the data by subtracting the mean of each feature.
   - Scale the data to have unit variance if needed.
2. **Compute the Covariance Matrix**:
   - Calculate the covariance matrix to understand the relationships between features.
3. **Calculate Eigenvalues and Eigenvectors**:
   - Determine the eigenvalues and eigenvectors of the covariance matrix to find principal components.
4. **Choose the Optimal Number of Dimensions**:

    - Generate a scree plot (plot of eigenvalues) to visually inspect the proportion of variance captured by each principal component.
    - Select the number of dimensions (principal components) that captures a sufficient amount of variance (e.g., 90-95%).
5. **Transform the Data**:
    - Project the original data onto the selected principal components to obtain the reduced dataset.

2. K-Nearest Neighbors (KNN)

Objective: KNN is a supervised learning algorithm used for classification or regression based on the closest training examples in the feature space. The performance of KNN depends on the choice of distance metric and the value of kkk (number of neighbors).

Steps for KNN:

1. **Choose Distance Metric**:
    - Common metrics include Euclidean, Manhattan, and Minkowski distances.
2. **Select Best kkk**:
    - Tune the number of neighbors kkk to find the optimal value that minimizes classification errors.
3. **Training**:
    - KNN doesn't have a training phase in the traditional sense; it stores the entire dataset and uses it for making predictions.
4. **Prediction**:
    - For a new data point, find the kkk nearest neighbors using the chosen distance metric and make predictions based on the majority vote (for classification) or average (for regression).

9.1 PCA + KNN

Procedure:

1. **Perform PCA**:

  - Apply PCA on the Spotify dataset to reduce dimensionality. Determine the optimal number of principal components using a scree plot, which shows the proportion of variance captured by each component. Select a sufficient number of components that capture a desired level of variance.

2. **Apply KNN**:
    - Train and test the KNN model using the reduced dataset. Use the best kkk and distance metric pair obtained from Assignment 1.

Benefits of PCA with KNN:

- **Dimensionality Reduction**: Reduces the number of features, which can help in mitigating the curse of dimensionality, improving model performance, and reducing computation time.
- **Noise Reduction**: By focusing on principal components, PCA can help in filtering out noisy features that may negatively affect KNN performance.

9.2 Evaluation

Procedure:

1. **Split Dataset**:
    - Divide the dataset into training and validation subsets.
2. **Calculate Metrics**:
    - Compute the following metrics for the KNN model on both the complete and reduced datasets:
        - **F1 Score**: The harmonic mean of precision and recall, providing a balance between the two.
        - **Accuracy**: The proportion of correctly classified instances out of the total instances.
        - **Precision**: The proportion of true positive results among the positive predictions.
        - **Recall**: The proportion of true positive results among the actual positives.
3. **Compare Metrics**:

    - Compare these metrics with those obtained in Assignment 1 to analyze improvements or differences in performance due to dimensionality reduction.

4. **Plot Inference Time**:
    - Measure and plot the inference time for the KNN model on both the complete and reduced datasets. Inference time is the time taken to make predictions on new data.

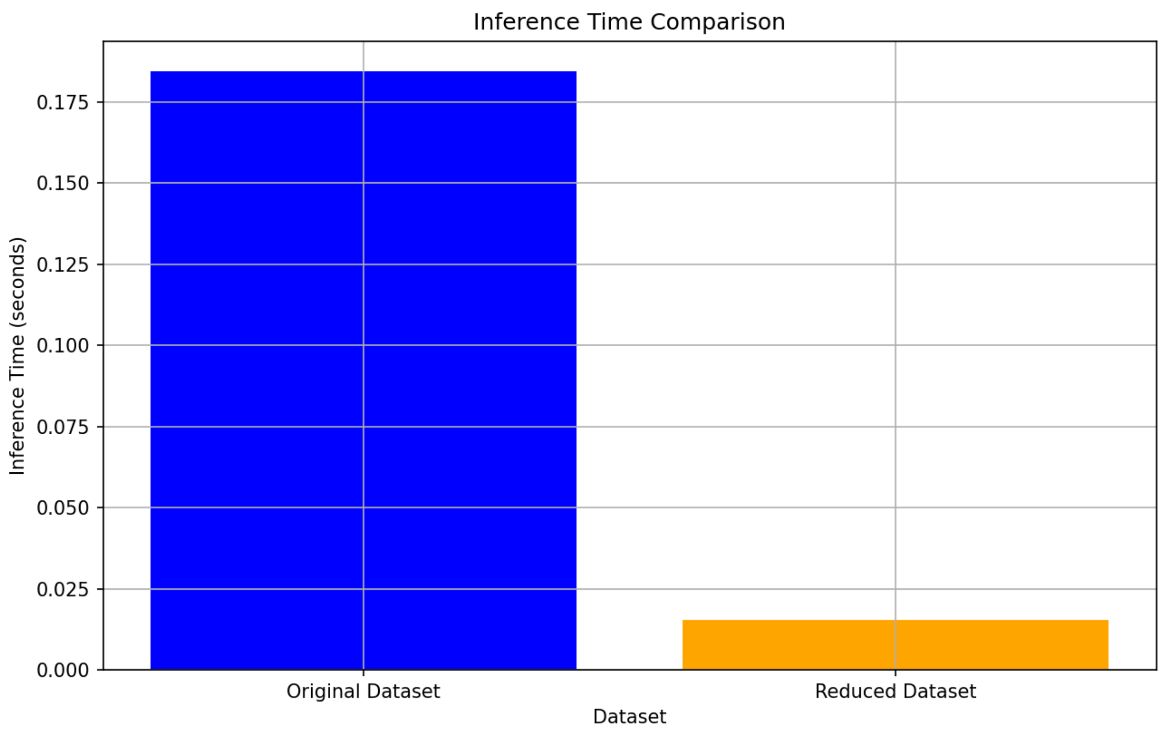
Implications of Inference Time:

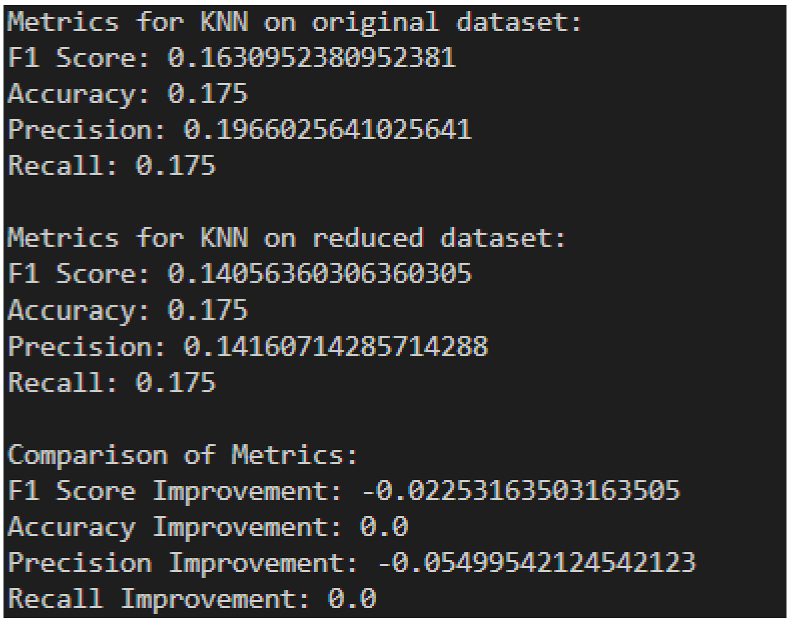
- **Reduced Dataset**: Typically, a reduced dataset will lead to faster inference times as there are fewer dimensions to consider during the distance computation.
- **Complete Dataset**: Using the complete dataset may result in longer inference times due to the higher dimensionality and computational complexity.

Analysis:

- **Improved Metrics**: If the F1 score, accuracy, precision, and recall improve with the reduced dataset, it indicates that PCA has effectively reduced dimensionality while preserving important information.
- **Inference Time**: A significant reduction in inference time for the reduced dataset suggests that dimensionality reduction with PCA is beneficial for improving the efficiency of the KNN model.

In summary, PCA helps in reducing dimensionality, which can lead to more efficient and potentially more accurate KNN models by focusing on the most informative features and reducing the computational burden. Evaluating the model's performance and inference time on both the complete and reduced datasets provides insights into the effectiveness of the dimensionality reduction.





```
Metrics for KNN on original dataset:
F1 Score: 0.1630952380952381
Accuracy: 0.175
Precision: 0.1966025641025641
Recall: 0.175

Metrics for KNN on reduced dataset:
F1 Score: 0.14056360306360305
Accuracy: 0.175
Precision: 0.14160714285714288
Recall: 0.175

Comparison of Metrics:
F1 Score Improvement: -0.022531635031635505
Accuracy Improvement: 0.0
Precision Improvement: -0.05499542124542123
Recall Improvement: 0.0
```