FAST NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

REPORT

Name:	**Muhammad Saif (22F-3165)** **Sitara Shabbir (22F-3838)**
Assignment:	**4**
Instructor:	**Dr. Hashim Yasin**
Course:	**Machine Learning**

DELIVERABLES:

1. Code report in the form of pdf.
2. “ML_A4_Q1” ipynb file containing codes for question 1.
3. “ML_A4_Q2” ipynb file containing codes for question 2.

//Each code is commented in both notebook files.

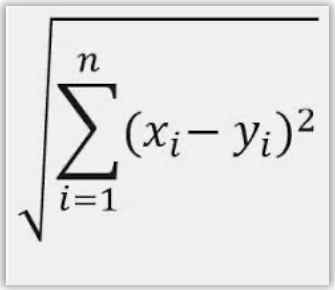
QUESTION 1:

For the first part, we read in the data:

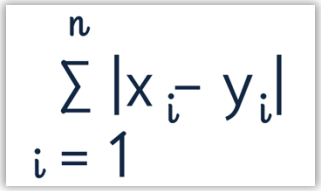
We started by loading the data from the file 'Data.xlsx' and then we set the custom headers (A1, A2, A3, A4) for the columns. We then select only these four columns to create our feature set, X, which we'll use throughout the clustering process.

For the next part, we define distance functions:

1. **Euclidean Distance:** We utilized this distance in k means which calculates straight line distance.



2. **Manhattan Distance:** We used this distance in k medoids and k medians, it sums up the absolute distances in each dimension.



Then, we initialized Centroids:

To begin clustering, we randomly selected K points from the data as initial centroids.

For K-Means Clustering:

1. **Assign Clusters**: We assigned each point to the closest centroid using Euclidian distance.
2. **Update Centroids**: Each centroid is recalculated as the mean of its assigned points.
3. **Repeat Until Stable**: We repeated until it converged.

For K-Medoids Clustering:

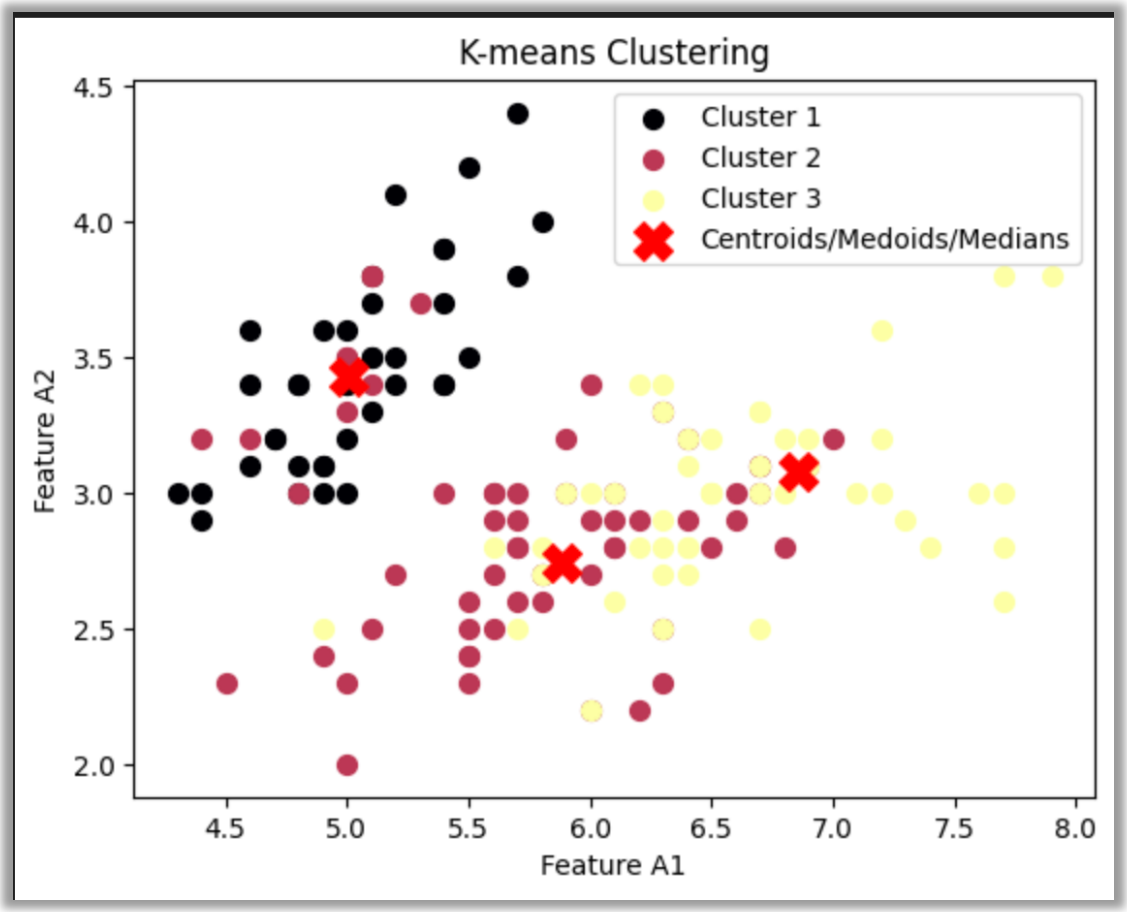
1. **Assign Clusters**: Points are assigned to the closest medoid using Manhattan distance.
2. **Update Medoids**: Each medoid is recalculated as the point within the cluster that minimizes total distance.
3. **Repeat Until Stable**: We repeated until it converged.

For K-Median Clustering:

1. **Assign Clusters**: Points are assigned to the nearest median point using Manhattan distance.
2. **Update Medians**: Each median is recalculated as the median of its assigned points.
3. **Repeat Until Stable**: Continued until medians stabilized.

K-means Clustering:

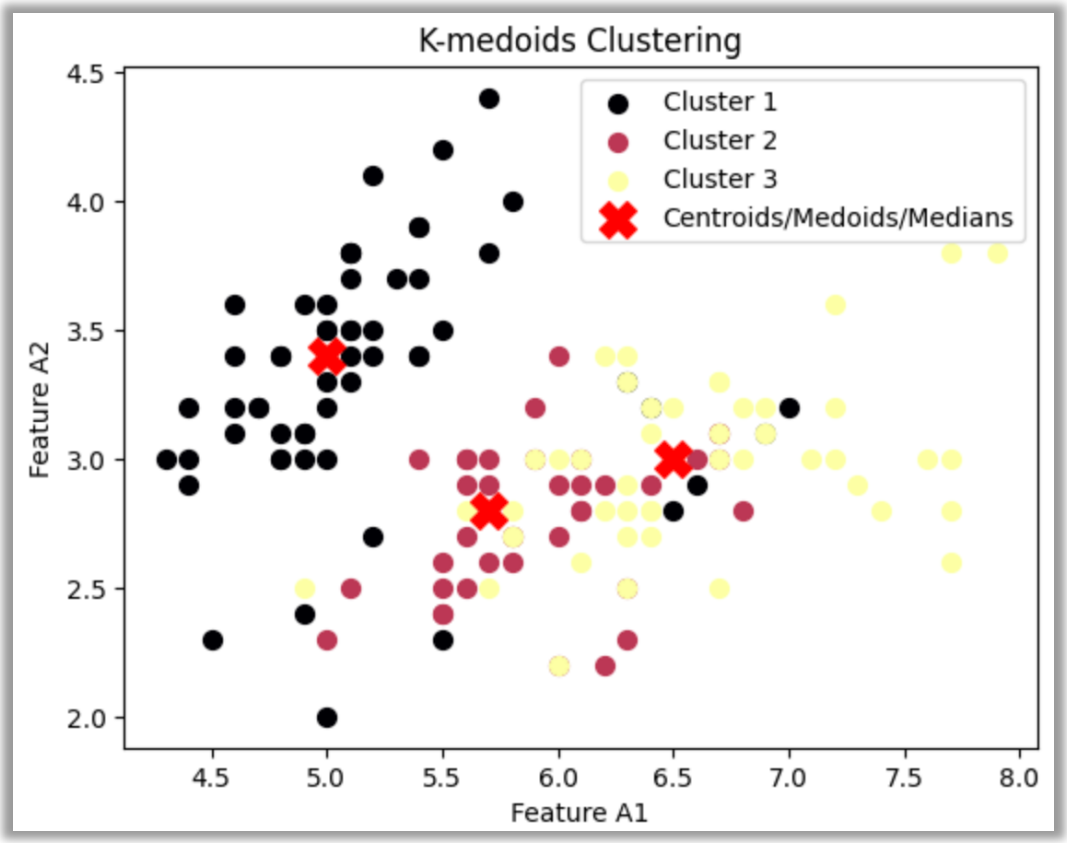
We plotted the graph using matplotlib, showing how K-means clustering grouped the data into three distinct clusters.



Each cluster is centered around calculated centroids, with minimal overlap between clusters, highlighting clear separation based on average distances.

K-medoids Clustering:

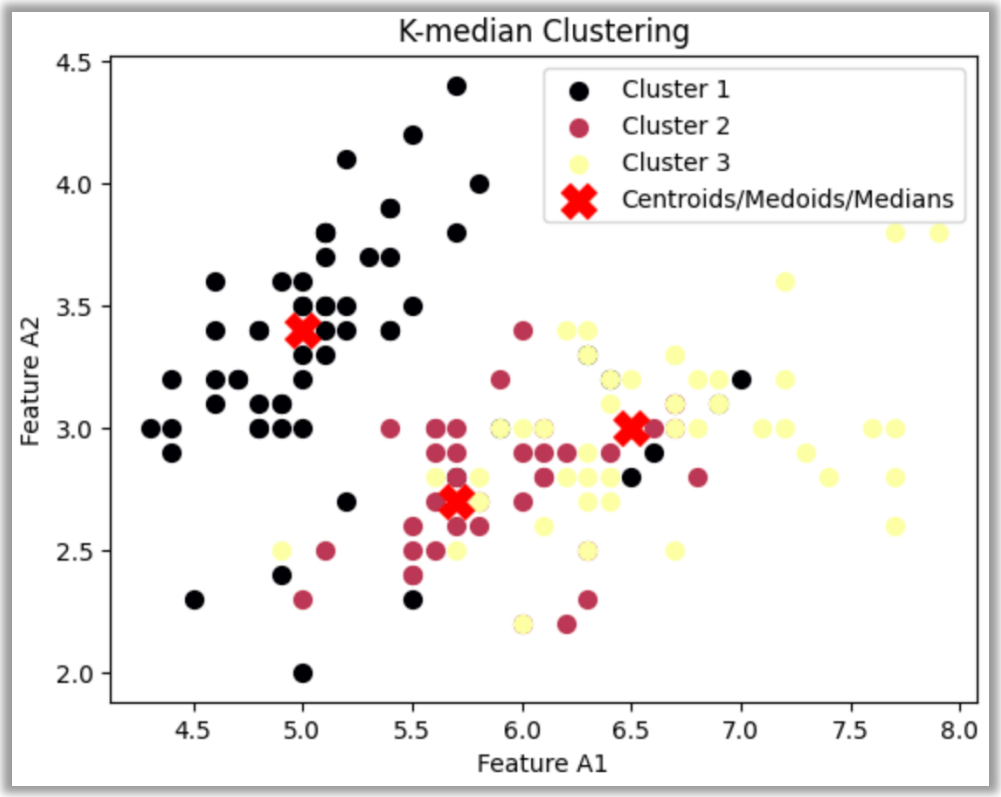
We plotted the graph using matplotlib, showing clusters based on K-medoids clustering.



Here, each cluster is represented by a medoid, making the clusters more stable in the presence of outliers, though with slight overlaps due to the median-based center points.

K-median Clustering:

We plotted the graph using matplotlib, illustrating clusters from K-median clustering.



Each cluster centers around median points, making it robust against outliers. There is visible overlap between some clusters, especially between the two larger groups.

Clustering Algorithm	Advantages	Disadvantages
K-means Clustering	• Efficient for large datasets. • Provides distinct, non-overlapping clusters. • Suitable for spherical clusters around a centroid.	• Sensitive to outliers. • Requires the number of clusters (K) to be predefined. • May produce inaccurate results with irregularly shaped clusters.
K-medoids Clustering	• More robust to outliers compared to K-means. • Uses actual data points as medoids, making clusters representative.	• Slower than K-means, especially on large datasets. • Performance may degrade if clusters aren't well-separated.
K-median Clustering	• Resistant to outliers due to median-based calculations. • Suitable for non-spherical clusters. • Centers are less influenced by extreme values.	• Computationally more complex than K-means. • Clusters may overlap, leading to less distinct boundaries. • Requires a predefined number of clusters (K).

Lastly, we calculated silhouette scores,

For three different clustering algorithms to evaluate their clustering performance. The silhouette score is a metric that measures how similar an object is to its own cluster compared to other clusters, ranging from -1 to +1. A higher score indicates better-defined clusters, where +1 represents highly dense clustering, 0 indicates overlapping clusters, and -1 suggests incorrect clustering.

Our results showed:

- K-medoids performed best with a silhouette score of 0.386
- K-median achieved a score of 0.374
- K-means showed the lowest score at 0.356

All three algorithms produced positive silhouette scores (0.36-0.39), indicating reasonable but moderate clustering quality. K-medoids achieved marginally better cluster separation with a score of 0.386, followed by K-median (0.374) and K-means (0.356). While these scores demonstrate successful clustering, the moderate values suggest some cluster overlap, which is common in real-world datasets where perfect cluster separation is uncommon.

QUESTION 2:

The preprocessing:

We loaded data from 'Data.xlsx' containing multiple features. To ensure fair comparison and eliminate scale differences, we standardized the data using z-score normalization (mean=0, standard deviation=1).

The Methodology:

We implemented hierarchical clustering using three different linkage methods:

1. **Single Linkage**
    - Merges clusters based on minimum distance between points
    - Useful for detecting non-elliptical shaped clusters
2. **Complete Linkage**
    - Merges clusters based on maximum distance between points
    - Tends to form more compact, evenly sized clusters
3. **Average Linkage**
    - Merges based on average distance between all points
    - Provides balanced results between single and complete linkage

Implementation:

The clustering algorithm follows these steps:

1. Calculate distance matrix between all points using Euclidean distance
2. Find closest clusters to merge
3. Update distance matrix after each merge
4. Track merge history including cluster sizes and merge distances

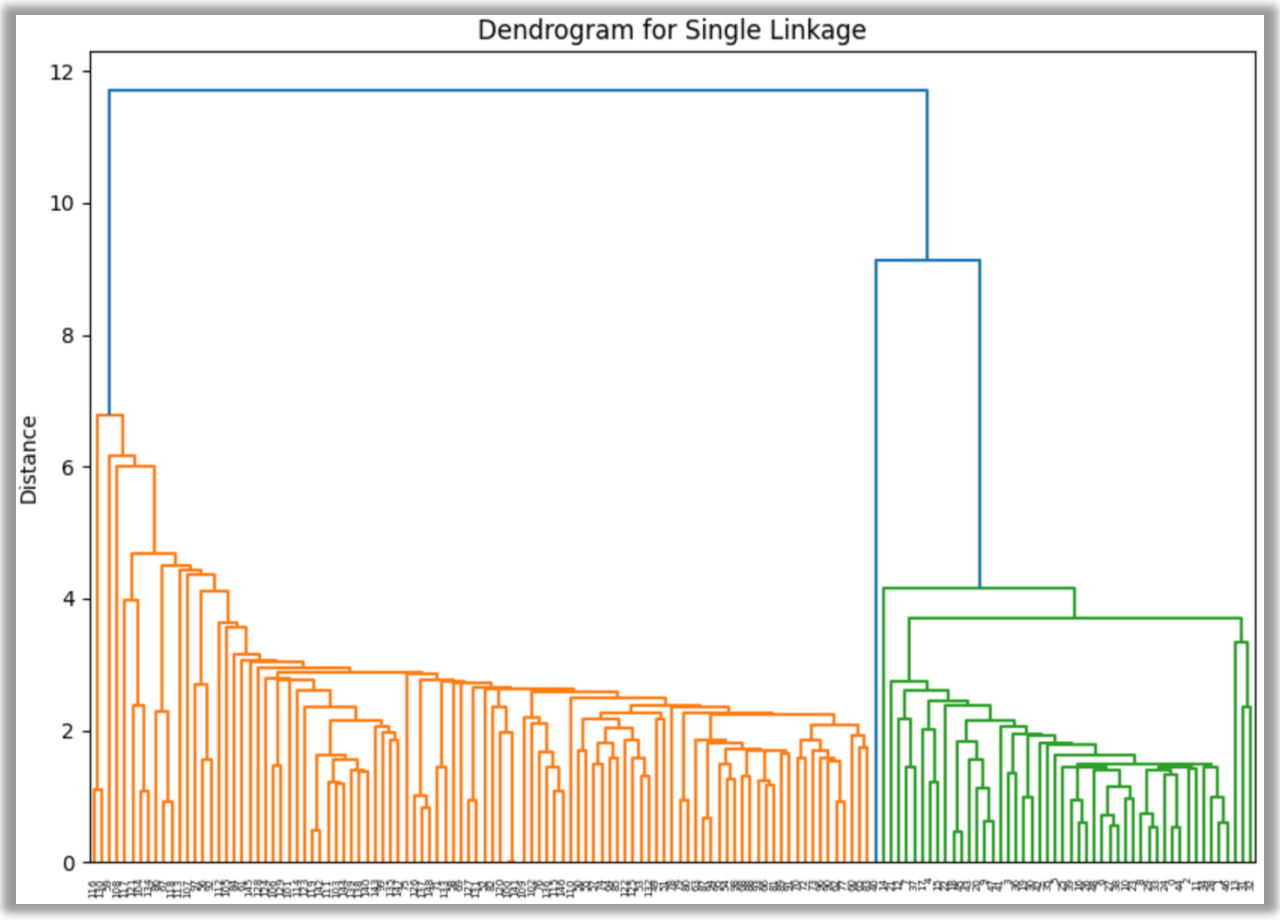
Distance Calculation:

We used Euclidean distance as our distance metric, implementing it through:

- Point-to-point distance calculation
- Full distance matrix computation
- Dynamic matrix updates during clustering

We plotted the dendrogram for single linkage clustering,

using SciPy's hierarchical clustering visualization tools and matplotlib. The dendrogram illustrates the hierarchical clustering structure using single linkage method.



The visualization shows two main clusters (orange and green), separated by about 11.5 units. The vertical axis represents the distance or dissimilarity between clusters, and the horizontal axis shows the sample indices.

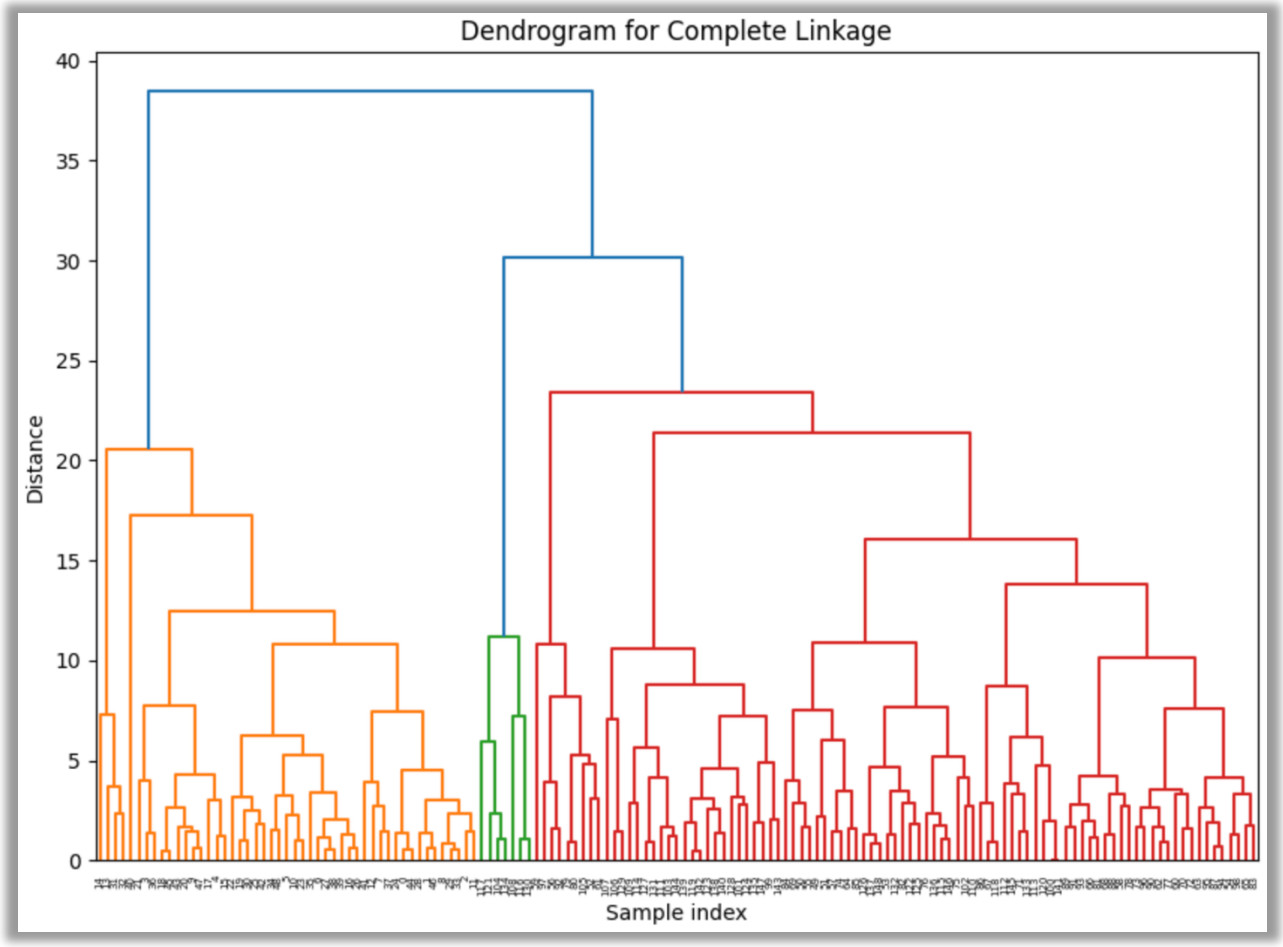
Key observations include:

1. Clear separation of two major groups
2. Smaller sub-clusters within each group
3. Gradual merging of clusters below 4 units
4. The orange cluster has more internal variation with multiple sub-branches
5. The green cluster is more compact with shorter branches

This dendrogram reveals a natural two-cluster separation with hierarchical relationships within each cluster.

Dendrogram for complete linkage clustering:

Then we plotted the dendrogram for complete linkage clustering using the same tools,



The visualization shows three main clusters (orange, green, and red), separated by significant distances in the early clustering stages. The vertical axis represents the distance or dissimilarity

between clusters, measured by the furthest distance between points, while the horizontal axis shows the sample indices.

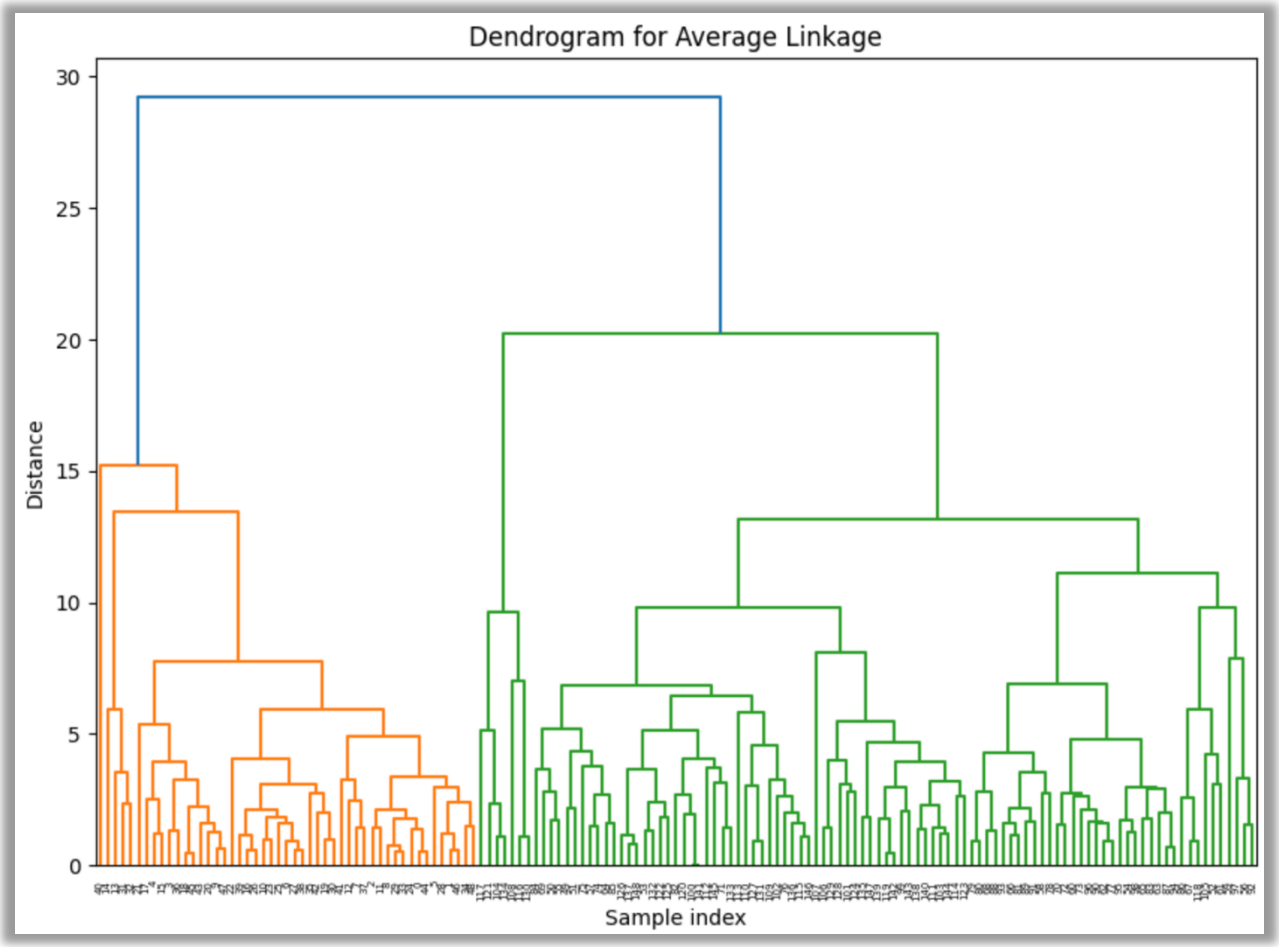
Key observations include:

1. Clear separation of three major groups, indicated by distinct color segments.
2. Smaller sub-clusters within each group, reflecting the nested structure from complete linkage.
3. Gradual merging of clusters below 5 units.
4. The orange cluster has high internal variation, with several sub-branches.
5. The green cluster is compact, merging at a shorter distance, showing high internal similarity.
6. The red cluster has substantial internal branching, indicating diverse sub-clusters.

This dendrogram reveals a natural three-cluster separation, highlighting how complete linkage clustering maximizes inter-cluster distance.

Dendrogram for average linkage clustering:

Lastly, we plotted the dendrogram for complete linkage clustering using the same tools.



The visualization shows two main clusters (orange and green), separated by a large distance at the top of the hierarchy. The vertical axis represents the distance or dissimilarity between clusters, calculated using the average distance between points in each pair of clusters, while the horizontal axis displays sample indices.

Key observations include:

1. Clear separation of two major groups, highlighted by distinct color segments.
2. Smaller sub-clusters within each group, illustrating a nested structure from average linkage.
3. Gradual merging of clusters below approximately 5 units.
4. The orange cluster has more internal variation, with several sub-branches at lower distances.
5. The green cluster is larger but more compact, merging with the orange cluster at a high distance, suggesting internal cohesion.

This dendrogram reveals a natural two-cluster division with hierarchical relationships within each cluster, showing how average linkage balances inter-cluster distances.

Conclusively,

Comparing the three linkage methods, single linkage forms clusters with minimal distance between points, leading to long, narrow clusters; complete linkage maximizes inter-cluster distances, creating more compact clusters with clear separation; and average linkage balances both approaches, yielding clusters that are moderately cohesive and well-separated, showing a balanced hierarchical structure.