Bangabandhu Sheikh Mujibur Rahman Digital University,

Bangladesh.

Faculty: Cyber Physical Systems

Department: IoT AND Robotics Engineering (IRE)

Course Title: Data Science

Course Code: IoT 4313

Assignment-02: Clustering

Submitted To:
Nurjahan Nipa
Lecturer
Department of IRE, BDU.

Submitted By:
Md. Mostafizur Rahaman
Id: 1901024
Session: 2019-20
Department of IRE

Submission Date: 24 October 2023

Part A:

K-means Clustering: In this part, you will be utilizing K-means clustering algorithm to identify the appropriate number of clusters. You may use any language and libraries to implement K-mean clustering algorithm. Your K-mean clustering algorithm should look for appropriate values of K at least in the range of 0 to 15 and show their corresponding sum of-squared errors (SSE).

Procedure:

Step 01: Importing Libraries:

It begins by importing the necessary Python libraries:

- pandas for data manipulation. numpy for numerical operations.
- matplotlib.pyplot for data visualization.
- sklearn.cluster.KMeans for K-means clustering.

Step 02: Loading the Dataset:

The code loads the dataset "Mall_Customers.csv" using pd.read_csv() and stores it in the data DataFrame (Data).

Step 03: Calculating Sum of Squared Errors (SSE) for Various K Values:

It initiates an empty list sse to store the sum of squared errors for different values of K. The range of values for K (from 1 to 15) is defined in k_range.

Step 04: Loop Over Different K Values:

The code iterates over each value of K using a for loop. Inside the loop, a K-means clustering model is created with the current value of K.

kmeans = KMeans(n_clusters=k, random_state=42) creates a K-means model with the specified number of clusters (K).

The fit() method is used to fit the K-means model to the data. It clusters the data points into K clusters.

kmeans.inertia_ retrieves the sum of squared errors (SSE) for the current clustering, which is a measure of how far data points are from their cluster's centroid.

Step 05: Plotting the Elbow Curve:

After calculating the SSE for different K values, the code creates a plot to visualize the results.

plt.figure(figsize=(10, 5)) sets the size of the plot.

plt.plot(k_range, sse, marker='o') creates a line plot where K values are on the x-axis, and SSE values are on the y-axis.

plt.title('Elbow Method for Optimal K') sets the title of the plot.

plt.xlabel('Number of Clusters (K)') labels the x-axis.

plt.ylabel('Sum of Squared Errors (SSE)') labels the y-axis.

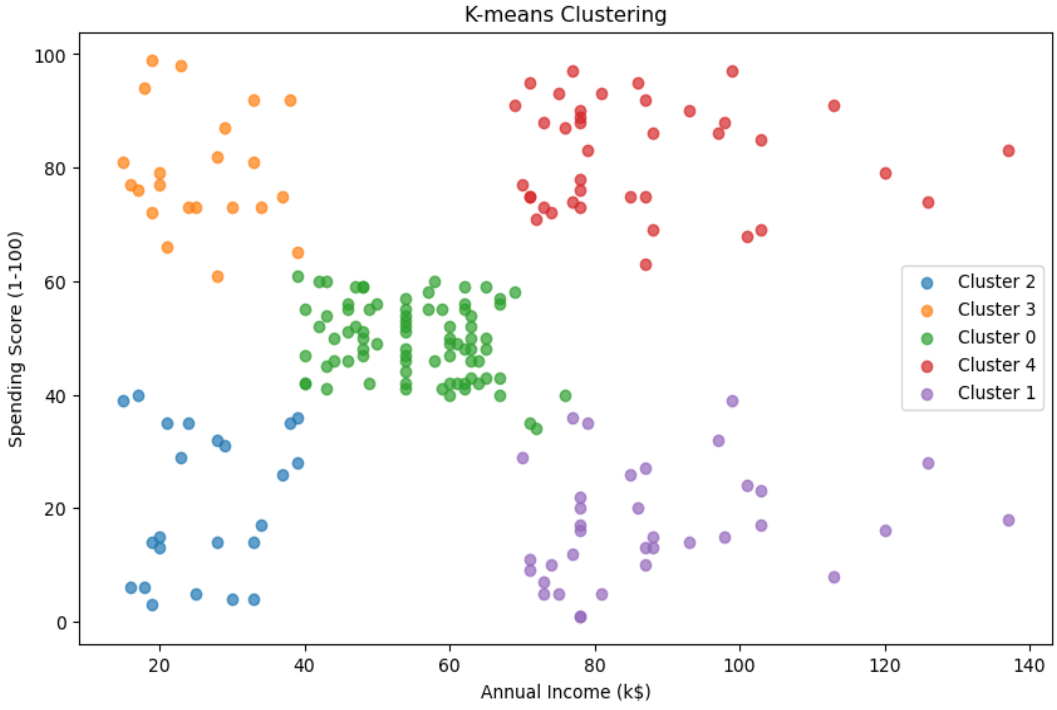
plt.show() displays the plot.

Step 06: Elbow Method Visualization:

The plot shows an "elbow" pattern, where the SSE decreases as K increases. The elbow point represents the optimal K. It is the point where increasing the number of clusters does not significantly reduce SSE.

Step 07:

Now just plot the Cluster and the cluster size is 5



Part 02:

Hierarchical Clustering: In this part, you will apply hierarchical clustering algorithm (agglomerative or divisive) to the provided mall dataset.

Procedure:

Step 01:Import Relevant Libraries:

- from scipy.cluster.hierarchy import dendrogram, linkage: This imports the necessary functions and classes from SciPy for hierarchical clustering and dendrogram visualization.

Step 02: Select Data for Clustering:

X = data[['Annual Income (k$)', 'Spending Score (1-100)']]: This line selects the two features, 'Annual Income (k$)' and 'Spending Score (1-100),' from the data DataFrame. These are the features you want to use for clustering.

Step 03: Perform Agglomerative Clustering:

agg_clustering = AgglomerativeClustering(n_clusters=None, linkage='ward', distance_threshold=0).fit(X): Here, Agglomerative Clustering is applied to the data. The parameters used are:

n_clusters=None: This indicates that the algorithm should not specify a fixed number of clusters but will instead be controlled by the distance threshold.

linkage='ward': The 'ward' linkage method is used to measure the distance between clusters.

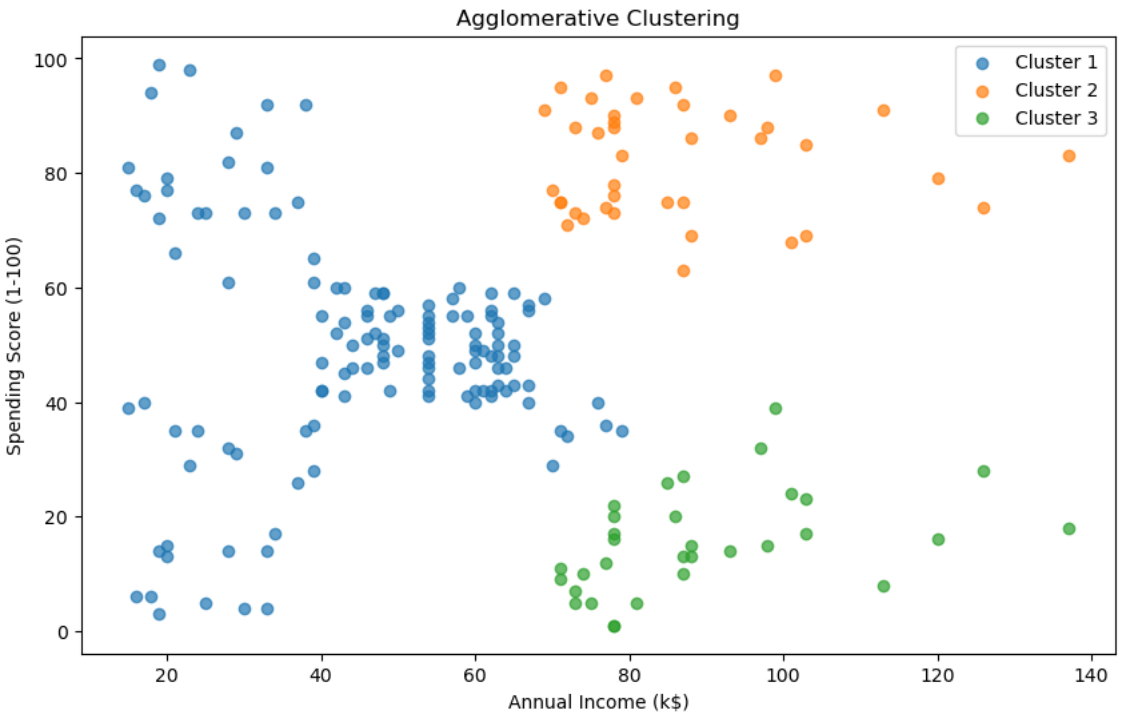
distance_threshold=0: The distance threshold is set to 0, indicating that the dendrogram will be created without a predefined height cutoff.

Step 04: Generate the Dendrogram:

linked = linkage(X, 'ward'): The linkage function computes the hierarchical clustering of data points using the 'ward' linkage method, and the result is stored in the linked variable.

Step 05: plotting the cluster

Here cluster size is 3



Part c:

Density-based Clustering: In this part, you will apply density-based clustering algorithm to the provided dataset.

Procedure:

Step 01: Import Relevant Libraries:

The code begins by importing the necessary libraries:

- matplotlib.pyplot for data visualization,
- numpy for numerical operations
- sklearn.cluster.DBSCAN for DBSCAN clustering.

Step 02: Prepare the Data:

df = df.iloc[:, [3,4]].values: This line selects columns 3 and 4 (indexing starts from 0) from the DataFrame df and converts them into a NumPy array. These columns are assumed to be the features used for clustering.

Step 03: Initial Scatter Plot:

plt.scatter(df[:,0], df[:,1], s=10, c="red"): This command creates an initial scatter plot of the data points in red color. The data points are plotted in a 2D space based on the two selected features (Annual Income and Spending Score).

Step 04: DBSCAN Clustering:

dbscan = DBSCAN(eps=5, min_samples=5): It initializes a DBSCAN clustering model with the specified parameters.

eps (epsilon) is the maximum distance between two samples for one to be considered as in the neighborhood of the other.

min_samples is the number of samples (data points) in a neighborhood for a point to be considered a core point.

Step 05: Fit and Predict:

labels = dbscan.fit_predict(df): This line fits the DBSCAN model to the data (df) and assigns cluster labels to each data point.

Step 06: Separate Data Points by Cluster Labels and display the plot:

The cluster size is 5