Bangabandhu Sheikh Mujibur Rahman Digital University, Bangladesh

Faculty of Cyber Physical System
Department of Internet of Things and Robotics Engineering
B.Sc. in Internet of Things and Robotics Engineering
Course Title: Data Science
Course Code: IoT 4313
Assignment 02: Clustering

Submitted By:
Ummay Haney
Id: 1901028
Session: 2019-20

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

Date of Submission: 14 October 2023

Introduction: Clustering, a data analysis technique, serves the purpose of grouping similar data points together. It is a valuable tool for uncovering patterns and structures within unlabeled datasets. Clustering methods come in various forms, each tailored to specific applications and challenges. When applied to tasks such as customer segmentation and anomaly detection, clustering seeks to identify inherent groupings within the data, providing valuable insights into the underlying trends and relationships.

Here is a detailed explanation of my approaches and the results obtained for all of the clustering algorithms required in Parts A, B, and C.

Part A: K-Means Clustering

Here I implement the "Elbow Method" to find the optimal number of clusters (K) for K-means clustering. Let's break down the approach and explain the results obtained:

Approach:

01.Import Libraries:

- KMeans from sklearn.cluster: This is for implementing the K-means clustering algorithm.
- matplotlib.pyplot: This is for data visualization.

02.Feature Selection:

Select the features for clustering from the dataset. In this case, the features chosen for clustering are 'Age,' 'Annual Income,' and 'Spending Score.'

03.Initialize SSE List:

Initialize an empty list named sse (Sum of Squared Errors) to store the SSE values for different values of K. SSE is a measure of the within-cluster variation in K-means clustering.

04.Loop Over K Values:

A loop is used to iterate over a range of K values from 1 to 15.

05.K-Means Clustering for Each K: For each value of K, a K-means clustering model is created with n_clusters=k. This model will cluster the data into K clusters.The random_state=0 argument ensures that the results are reproducible.

06.Fit the K-Means Model: The K-means model is fitted to the selected features 'X' from the dataset.

07.Calculate and Store SSE: The SSE (inertia) of the K-means model for the current value of K is calculated using kmeans.inertia_. This represents the sum of squared distances of data points to their nearest cluster center.The calculated SSE is then added to the sse list for later analysis.

08.Plot SSE for Each K: After calculating the SSE for all K values, a plot is created using matplotlib.pyplot to visualize the results.The x-axis represents the number of clusters (K), and the y-axis represents the SSE.

Results Obtained:

The code follows a systematic process to determine the optimal number of clusters (K) using the Elbow Method. For each value of K, K-means clustering is performed, and the SSE is calculated and stored.The results are visualized in a plot, showing the SSE for each K value.

Part B: Hierarchical Clustering

Hierarchical clustering is a method that builds a hierarchy of clusters by successively merging or splitting existing clusters.

Approach:

01.Import Libraries:

- AgglomerativeClustering from sklearn.cluster: This is for implementing agglomerative clustering.
- dendrogram and linkage from scipy.cluster.hierarchy: These are for creating a dendrogram to visualize the clustering process.
- matplotlib.pyplot for data visualization.

02.Specify the Number of Clusters (n_clusters):

In this example, the number of desired clusters is set to 5, and this is stored in the variable n_clusters.

03.Agglomerative Clustering:

An Agglomerative Clustering model is created with the specified number of clusters (n_clusters) and the linkage method 'ward'. The 'ward' linkage is a method for measuring the distance between clusters.

04.Fit and Predict Clusters:

The Agglomerative Clustering model is fitted to the dataset 'X' to identify clusters, and the agg_labels variable stores the cluster labels assigned to each data point.

05.Create Dendrogram:

A dendrogram, which is a tree-like visualization of the hierarchical clustering process, is created using the linkage matrix. The 'ward' method is used to calculate the linkage between clusters.

06.Plot the Dendrogram:

A plot of the dendrogram is generated using matplotlib.pyplot. The x-axis represents samples, and the y-axis represents the distance between clusters. This dendrogram visually illustrates the hierarchical clustering process.

07.Display Cluster Labels:

The cluster labels assigned to each data point are printed to the console.

Results Obtained:

Agglomerative Clustering is used to group data points into a specified number of clusters.The dendrogram visually shows the hierarchical clustering process, where clusters are successively merged as the distance decreases.The cluster labels for each data point are printed to the console.

Part C: Density-Based Clustering

Here is a detailed explanation of the approach and the results obtained for applying DBSCAN (Density-Based Spatial Clustering of Applications with Noise) to the dataset.

Approach:

01.Select Features for Clustering:

The features for clustering are selected from the dataset. In this case, 'Age,' 'Annual Income (k$),' and 'Spending Score (1-100)' are chosen as the features to be used for clustering.

02.Instantiate DBSCAN:

The DBSCAN model is instantiated with specific parameters.

- eps (epsilon) is set to 3, defining the maximum distance between data points to consider them as neighbors.
- min_samples is set to 5, which specifies the minimum number of data points required to form a dense region (core point).

03.Fit and Predict Clusters:

The DBSCAN model is fitted to the dataset 'X' to identify clusters, and the cluster_labels variable stores the cluster labels assigned to each data point.

04.Add Cluster Labels to Original Data:

The cluster labels obtained from DBSCAN are added as a new column 'Cluster' to the original data, allowing for easy identification of data points in each cluster.

05.Data Visualization:

A scatter plot is created using matplotlib.pyplot to visualize the clusters. The 'Annual Income (k$)' is plotted on the x-axis, 'Spending Score (1-100)' on the y-axis, and data points are colored according to their cluster labels using the 'cmap' parameter.

06.Display Cluster Labels:

The cluster labels assigned to each data point are printed to the console, providing a numerical representation of the clusters.

Results Obtained:

DBSCAN is used to cluster data points based on density and proximity, without specifying the number of clusters beforehand.

A scatter plot visualizes the clustering results, where each point's color represents its cluster assignment.