Course Name: Data Science

Course Code: IoT 4313

Assignment 02: Clustering

Submitted To:

Nurjahan Nipa

Lecturer, Dept. of Internet of Things and Robotics Engineering (IRE),

Bangabandhu Sheikh Mujibur Rahman Digital University, Bangladesh.

Submitted By:

Tasnima Hamid

ID: 1901043

Session: **2019-2020**

Internet of Things (IoT) Program,

Department of Internet of Things and Robotics Engineering(IRE),

Faculty of Cyber-Physical Systems.

Submission Date: 13th October 2023

Title: Clustering

In this assignment, we dive into the critical task of customer segmentation through the application of various clustering algorithms. The purpose of customer segmentation, or market basket analysis, is to categorize customers into distinct groups based on their characteristics and purchasing behavior. This segmentation enables a targeted approach to marketing, allowing for the development of personalized strategies that resonate with each customer group.

The dataset provided for this analysis, referred to as the "Mall_Customer" dataset, encompasses data on 200 customers. It includes the following features: Customer ID, Gender, Age, Annual Income (in thousand dollars), and a Spending Score ranging from 1 to 100. The Spending Score is a crucial metric, assigned by the mall based on a customer's spending behavior and nature.

For this assignment, I have performed the following tasks,

Importing the necessary libraries:

Importing necessary libraries is essential because it provides the tools and functions needed to perform specific tasks in your analysis or modeling process. I have imported the following libraries:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster import hierarchy
from sklearn.metrics import silhouette_score
from sklearn.model_selection import ParameterGrid
from sklearn.cluster import DBSCAN
from scipy.cluster.hierarchy import dendrogram, linkage
```

Reading the dataset and displaying the dataset:

Once the libraries are imported, the next step is to read the dataset and display it. This step is necessary because the dataset is the foundation of any data analysis task.

Reading file and displaying data

```
In [2]: data = pd.read_csv("Mall_Customers.csv")
        print(data.head())
```

```
   CustomerID   Genre  Age  Annual Income (k$)  Spending Score (1-100)
0           1    Male   19                  15                      39
1           2    Male   21                  15                      81
2           3  Female   20                  16                       6
3           4  Female   23                  16                      77
4           5  Female   31                  17                      40
```

The dataset was read successfully and stored in a dataframe named ‘data’.

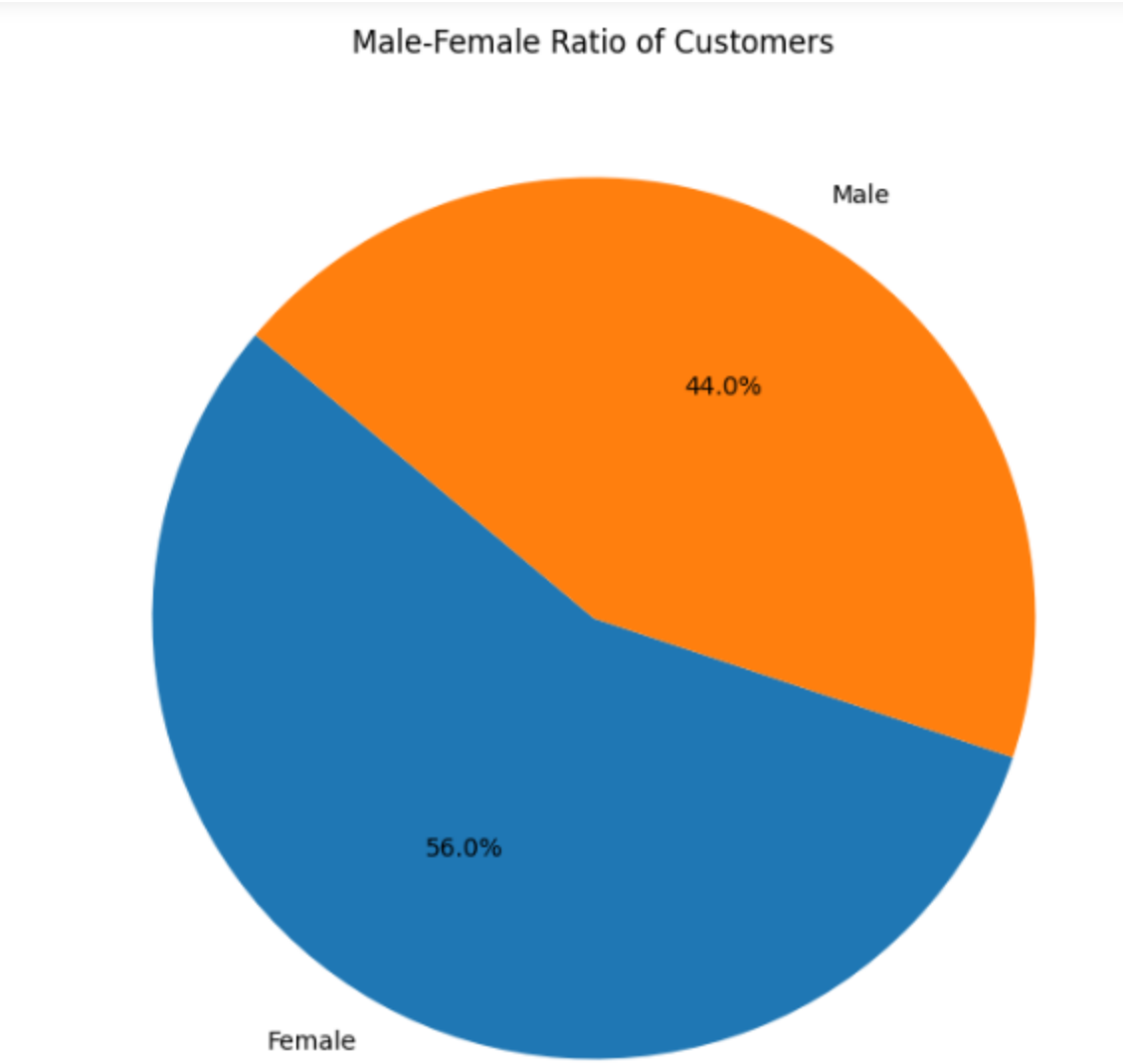
Data preprocessing and visualization:

Data preprocessing is the process of cleaning and transforming the raw data into a format that can be used for analysis. Preprocessing ensures that the data is accurate, consistent, and free of errors and outliers. I have performed the following preprocessing steps for the dataframe-

- **Checking for Missing Values:** The first step in data preprocessing is to check for missing values in the dataset. This is necessary because missing values can lead to incorrect results and misinterpretations during analysis. By using the isnull().sum() function, we can identify any columns that have missing values. In this case, there are **no missing values** in any of the columns, which simplifies the preprocessing process.

```
Out[3]: CustomerID              0
        Genre                   0
        Age                     0
        Annual Income (k$)      0
        Spending Score (1-100)  0
        dtype: int64
```

- **Label Encoding of Categorical Data:** Since clustering algorithms typically work with numerical data, we need to convert any categorical variables into numerical format. The 'Genre' column, which represents the gender of the customers, is a categorical variable. To handle this, we use label encoding, which is a technique that converts categorical values into numerical labels. This step is crucial for allowing the algorithm to process the categorical data. I also created a pie chart which shows the ratio of the genre of the customers.



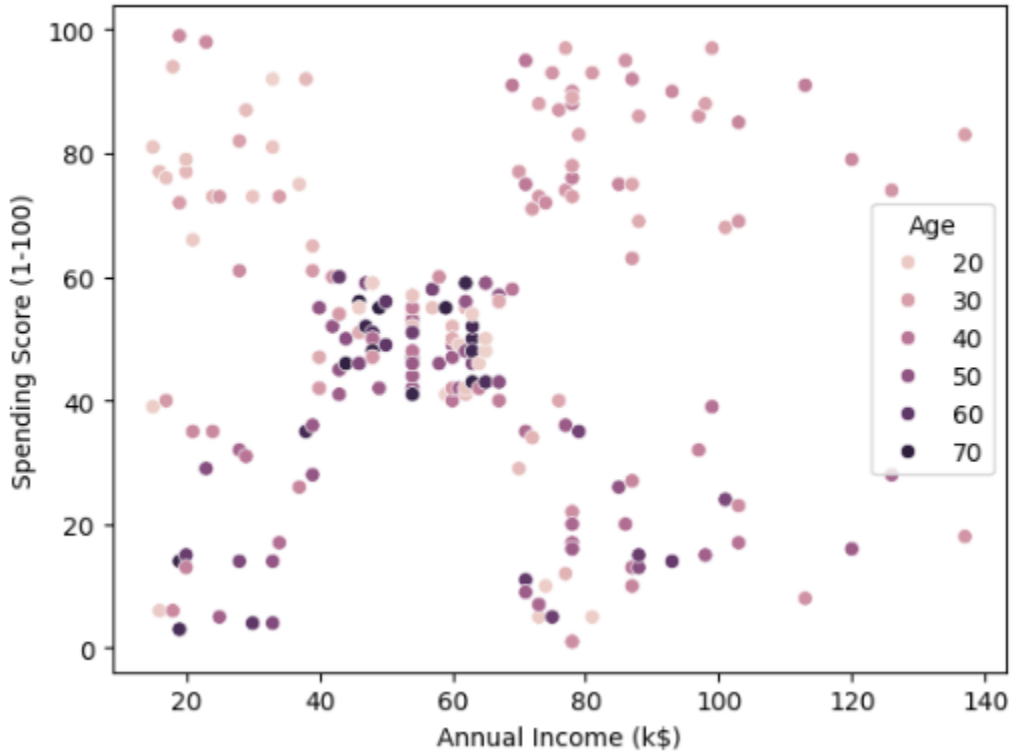
- **Feature Selection:** For clustering, we need to focus on the numerical features of the dataset. Specifically, **we are interested in the 'Annual Income (k$)' and 'Spending Score (1-100)' features** for K-means hierarchical clustering and Density Based Clustering. These features are selected because they capture the key characteristics of the customers that we want to use for segmentation. The 'main_clustering_features' DataFrame is created to hold these features.

- **Scaling the Values:** Scaling is an important step for algorithms that rely on distance calculations, such as hierarchical clustering. The 'StandardScaler' is used to standardize the features by removing the mean and scaling to unit variance. This step is necessary to ensure that all features contribute equally to the distance calculations and that the algorithm is not biased toward features with larger scales.

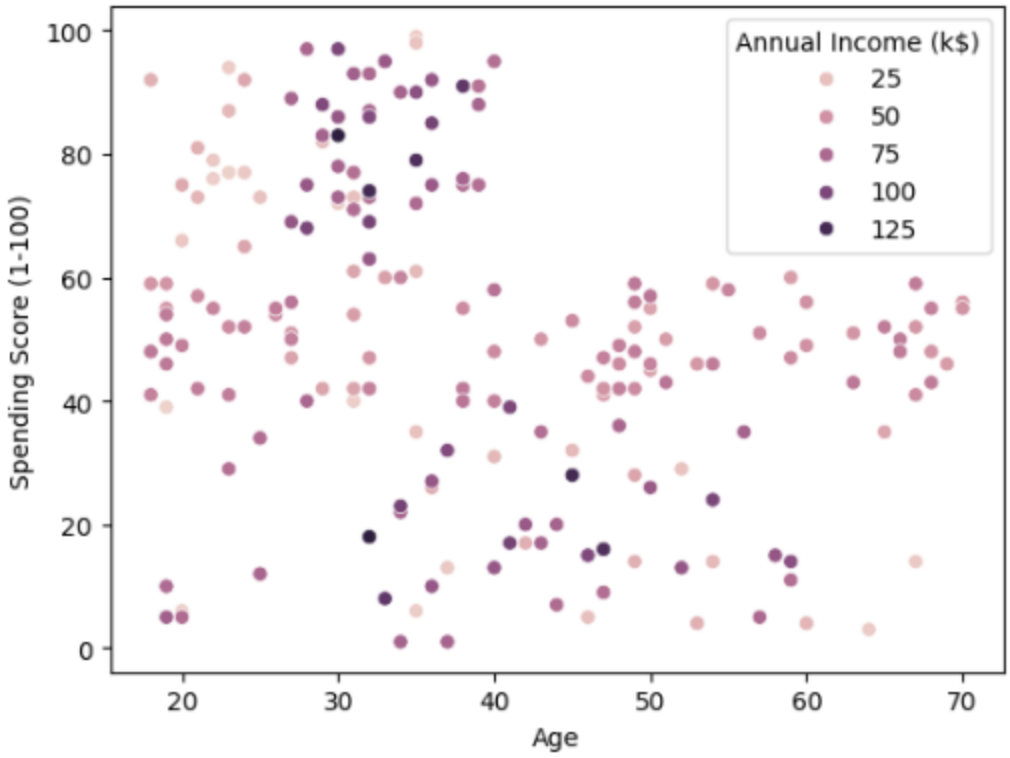
- **Visualizing the Relationship of the Features:** Visualization is an important part of data preprocessing and analysis. Scatter plots are used to visualize the relationships between

the selected features. This step is crucial for understanding the distribution of the data and identifying any patterns or trends that may exist.

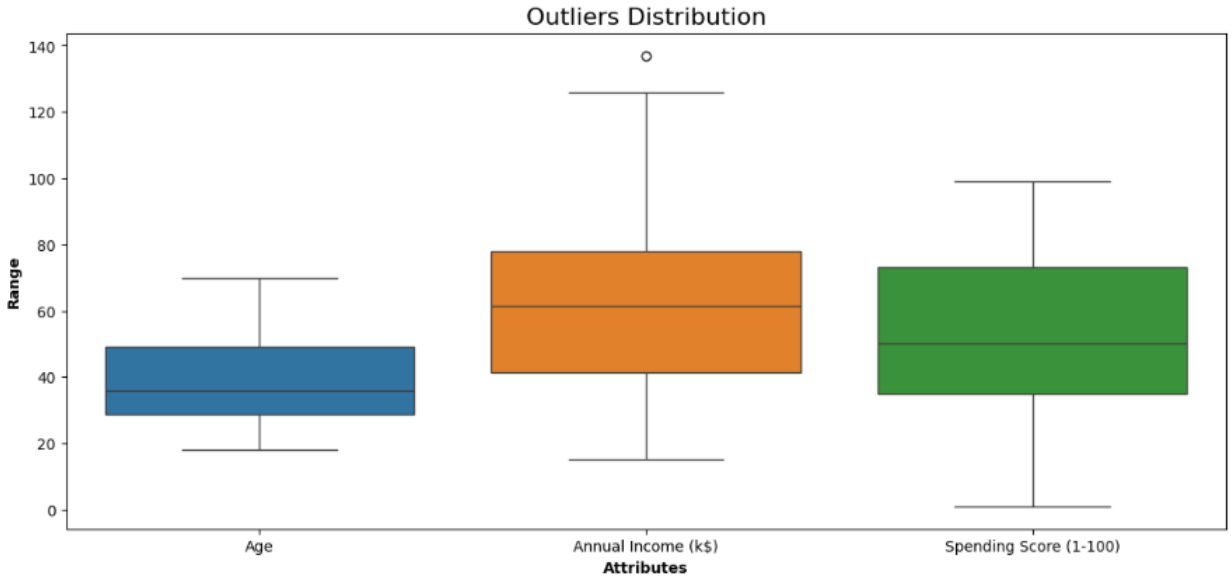
```
<Axes: xlabel='Annual Income (k$)', ylabel='Spending Score (1-100)'>
```



```
<Axes: xlabel='Age', ylabel='Spending Score (1-100)'>
```



- **Checking for Outliers:** Outliers can have a significant impact on clustering algorithms, particularly hierarchical clustering. A boxplot is used to visually inspect the distribution of the data and identify any outliers. This step is necessary to ensure that the results of the clustering analysis are not skewed by extreme values. If outliers are identified, they may need to be addressed through further preprocessing steps. There is only **one outlier** of Annual Income (k$) feature.



-

Applying the algorithms:

K-means Clustering

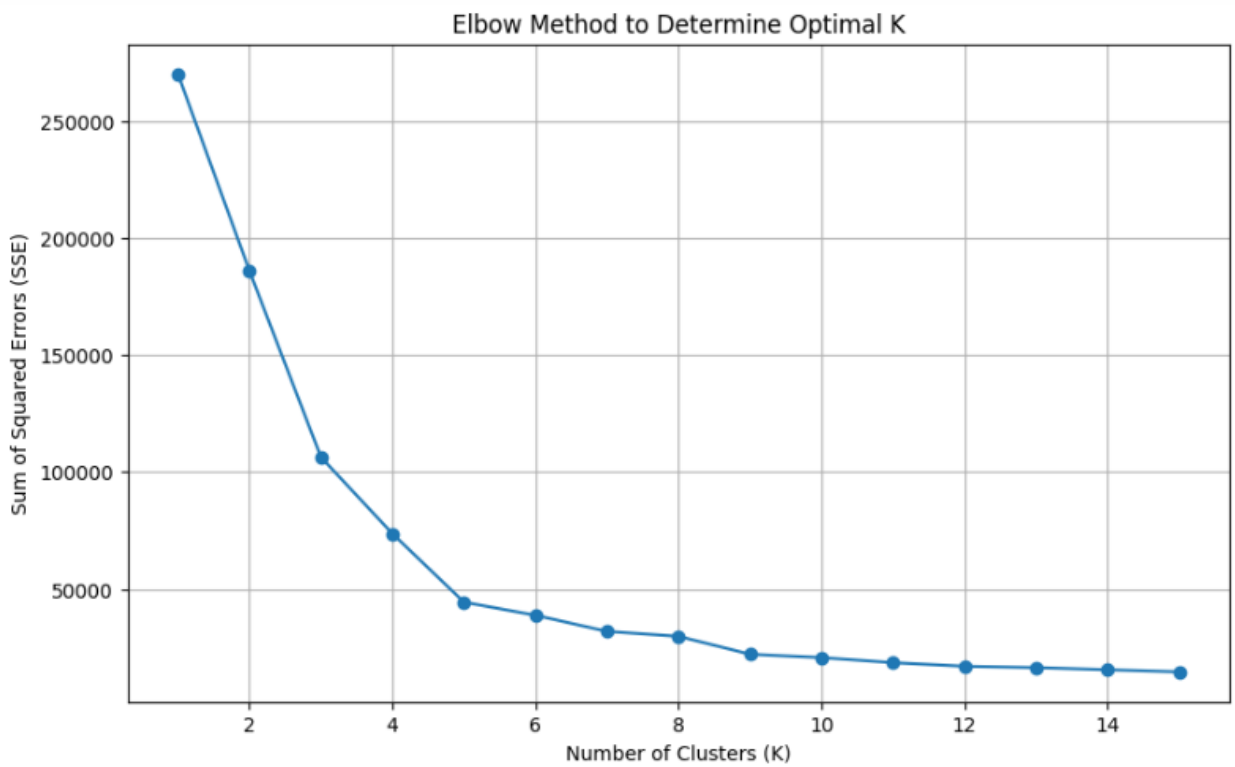
Determining the Optimal Number of Clusters (K):
The first step in K-means clustering is to determine the optimal number of clusters (K) to use for the dataset. To achieve this, we perform an iterative process where the K-means algorithm is run for different values of K (ranging from 1 to 15) and the Sum of Squared Errors (SSE) for each K is computed. The SSE, also known as the inertia, is a measure of the distance between each data point and the centroid of the cluster to which it belongs. It helps quantify how well the data points are grouped by the algorithm.

```
sse = []
for k in range(1, 16):
    kmeanModel = KMeans(n_clusters = k, init = 'k-means++', random_state =
0, n_init=1).fit(main_clustering_features)
    sse.append(kmeanModel.inertia_)
```

The reason for performing this step is to visualize the "elbow" in the plot of SSE values, which indicates the point where adding more clusters does not significantly reduce the SSE. This point represents the optimal number of clusters for the dataset.

```
SSE values for each K:
K = 1: SSE = 269981.28
K = 2: SSE = 185917.14253928524
K = 3: SSE = 106348.37306211119
K = 4: SSE = 73679.78903948834
K = 5: SSE = 44448.45544793371
K = 6: SSE = 38858.9599751439
K = 7: SSE = 31969.426550235476
K = 8: SSE = 29858.483597603947
K = 9: SSE = 22209.851608025543
K = 10: SSE = 20786.93669205916
K = 11: SSE = 18612.222868009685
K = 12: SSE = 17083.67250170932
K = 13: SSE = 16511.947370563896
K = 14: SSE = 15599.148331445118
K = 15: SSE = 14697.298899626938
```

Visualizing the Elbow Method:
Once the SSE values are computed, we plot the SSE against the number of clusters to visualize the "elbow."



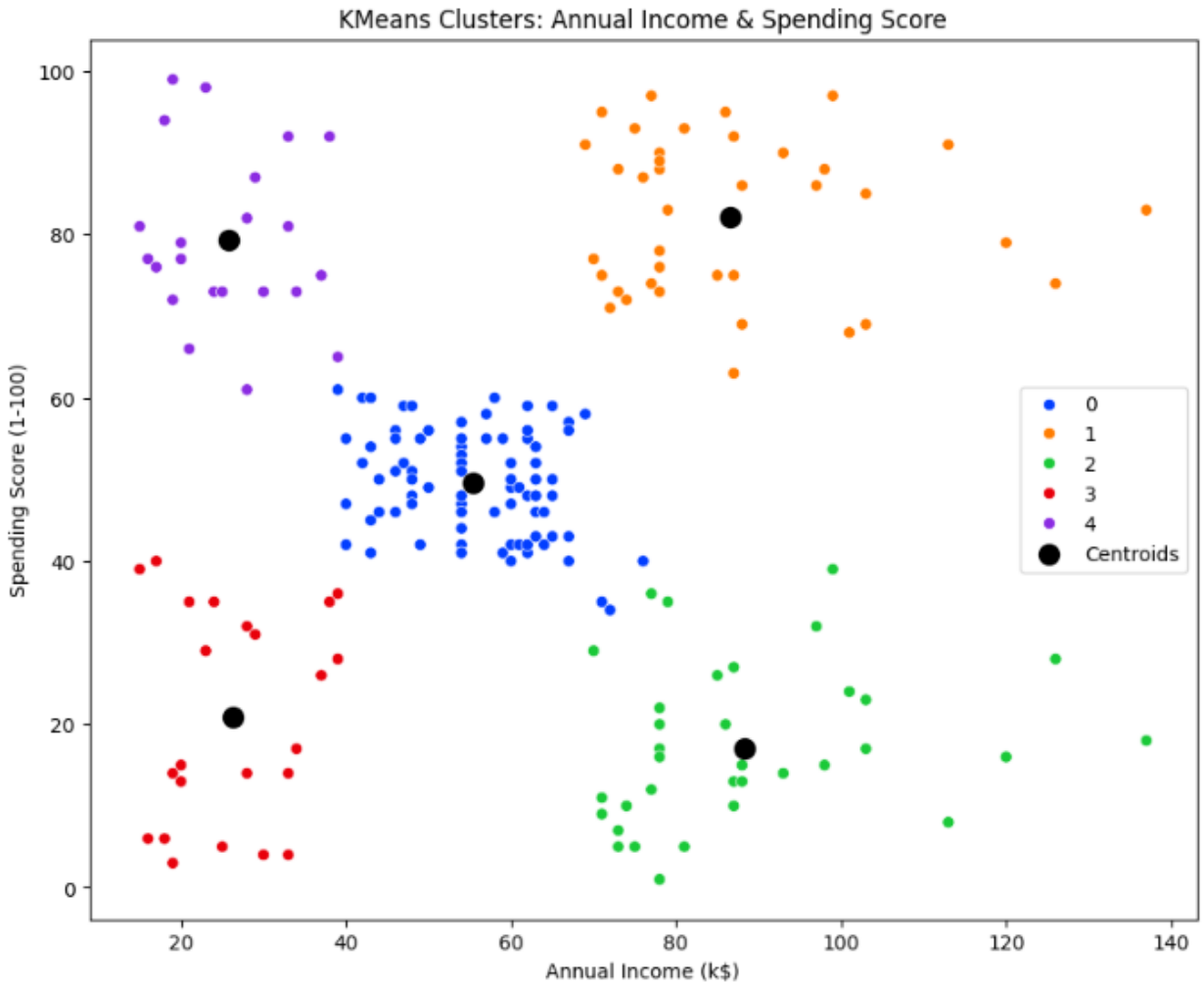
The reason for plotting this graph is to visually identify the "elbow" point, where the reduction in SSE starts to diminish. In this case, **the elbow is observed at K = 5,** indicating that this is the optimal number of clusters for the dataset.

Applying K-means Clustering with Optimal K:
Once the optimal number of clusters is determined, we apply the K-means algorithm with K = 5.

```
cluster_kmeans = KMeans(n_clusters=5, init = 'k-means++', random_state = 0,
n_init=1).fit(main_clustering_features)
data['KMeanslabel'] = cluster_kmeans.labels_
```

The reason for choosing K = 5 is that it represents the point where adding more clusters does not significantly improve the fit of the model to the data.

Visualizing the K-means Clusters:
The final step is to visualize the clusters formed by the K-means algorithm.



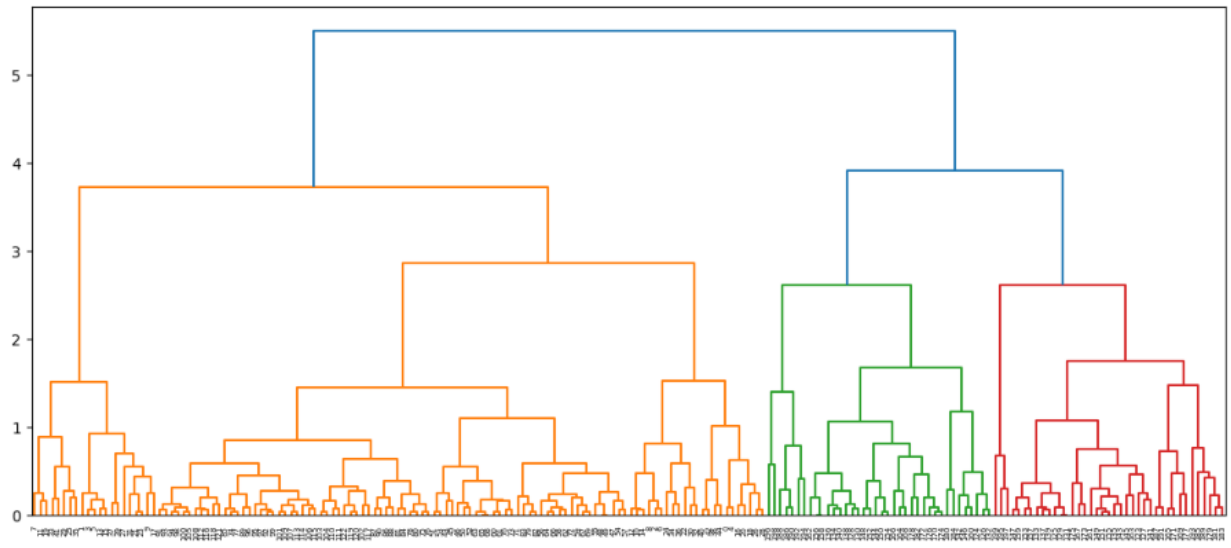
The 5 clusters detected are-

- Cluster 0: customers with **average annual income and average spending scores.** These customers may be more financially stable and have a good balance of income and expenses.
- Cluster 1: customers with **high annual income and high spending scores**. These customers may be big spenders or have a lot of disposable income.
- Cluster 2: customers with **high annual income and low spending scores.** These customers may be saving for a major purchase or retirement.
- Cluster 3: customers with **low annual income and low spending scores.** These customers may be budget-conscious or have limited financial resources.
- Cluster 4: customers with **low annual income and high spending scores.** These customers may be living beyond their means or have a lot of debt.

Hierarchical Clustering

Creating a Dendrogram:
The first step in hierarchical clustering is to create a dendrogram, which is a tree-like diagram that shows the hierarchical relationship between data points. The linkage function is used to compute the linkage matrix, which contains information about the hierarchical relationships between data points. The scaled_df dataset, which contains the standardized values of the 'Annual Income (k$)' and 'Spending Score (1-100)' features, is used as the input to the linkage function. The method parameter is set to "complete", which specifies the complete linkage method for hierarchical clustering. The complete linkage method calculates the maximum distance between clusters. The dendrogram is then created using the dendrogram function, and the result is visualized using the plt.show() function. This step is crucial for visualizing the hierarchical structure of the data and identifying the optimal number of clusters.

```
average_clustering = linkage(scaled_df, method="complete")
dendrogram(average_clustering)
plt.show()
```



By visually inspecting the dendrogram, the optimal number of clusters can be determined. In this case, the optimal number of clusters is identified as 4.

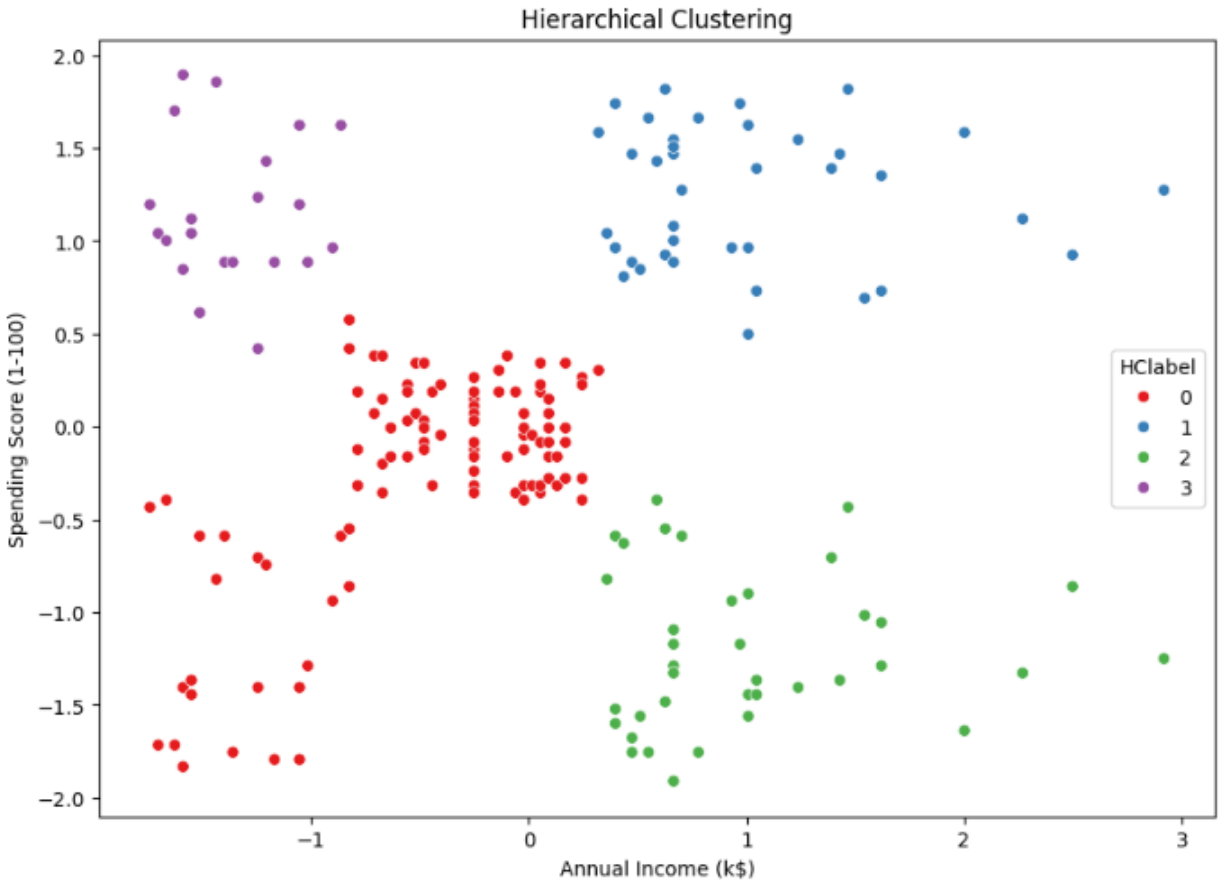
Applying Agglomerative Hierarchical Clustering:
After determining the optimal number of clusters, the next step is to apply the agglomerative hierarchical clustering algorithm. The AgglomerativeClustering class is used for this purpose. The n_clusters parameter is set to 4, which specifies the number of clusters to form. The metric parameter is set to 'euclidean', which specifies the Euclidean distance metric for calculating the distance between data points. The linkage parameter is set to 'complete', which specifies the

complete linkage method. The fit method is then called on the scaled data to perform the clustering. The resulting cluster labels are stored in a new column called 'HClabel' in the data DataFrame. This step is necessary for grouping the data points into clusters based on their similarity.

```
agc = AgglomerativeClustering(n_clusters = 4, metric = 'euclidean', linkage
= 'complete').fit(scaled_df)
data['HClabel'] = agc.labels_
```

Visualizing the Clustering Results:

Finally, the clustering results are visualized using a scatter plot. The sns.scatterplot function is used for this purpose, with the 'Annual Income (k$)' and 'Spending Score (1-100)' features on the x and y axes, respectively. The cluster labels are used to color the data points, and the color palette is set to "Set1" with the number of colors equal to the number of unique cluster labels. The plot is then displayed using the plt.show() function. This step is crucial for understanding the distribution of the clusters and assessing the quality of the clustering results.



There are four clusters of customers.

Cluster0: customers **who have low annual income and low spending score.** They are likely to be frugal and uninterested customers who buy rarely and spend little.

Cluster1: customers **who have high annual income and low spending score.** They are likely to be careful and sensible customers who buy occasionally but spend wisely.
Cluster2: customers who have **low annual income and high spending score.** They are likely to be impulsive and reckless customers who buy often but spend beyond their means.
Cluster3: customers who have **high annual income and high spending score**. They are likely to be loyal and profitable customers who buy frequently and spend a lot.

Density-based Clustering

Defining the Parameter Grid:
The first step in the density-based clustering analysis is to define the parameter grid for the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm. The ParameterGrid class is used to generate a grid of parameter combinations for hyperparameter tuning.

```
param_grid = ParameterGrid({
    'eps': np.linspace(1, 15, 15),
    'min_samples': range(4, 10)
})
```

The eps parameter defines the maximum distance between two samples for one to be considered as in the neighborhood of the other, and min_samples represents the number of samples (or total weight) in a neighborhood for a point to be considered as a core point. By defining a grid of values for these parameters, we can explore the parameter space and identify the optimal combination that maximizes the performance of the DBSCAN algorithm.

Finding the Best Parameters:
To find the best parameters for the DBSCAN algorithm, we iterate through the parameter grid and fit the DBSCAN model to the data for each parameter combination. The silhouette score is used to evaluate the quality of the clusters formed by the DBSCAN algorithm. The silhouette score ranges from -1 to 1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters.

```
best_score = -1
best_params = None
for params in param_grid:
    dbscan = DBSCAN(**params)
    labels = dbscan.fit_predict(main_clustering_features)
    # Only compute the silhouette score for clusters greater than 1
    if len(np.unique(labels)) > 1:
```

```
        score = silhouette_score(main_clustering_features, labels)
        if score > best_score:
            best_score = score
            best_params = params
```

The reason for performing this step is to identify the parameter combination that results in the best clustering solution according to the silhouette score.

```
Best Parameters: {'eps': 9.0, 'min_samples': 5}
```

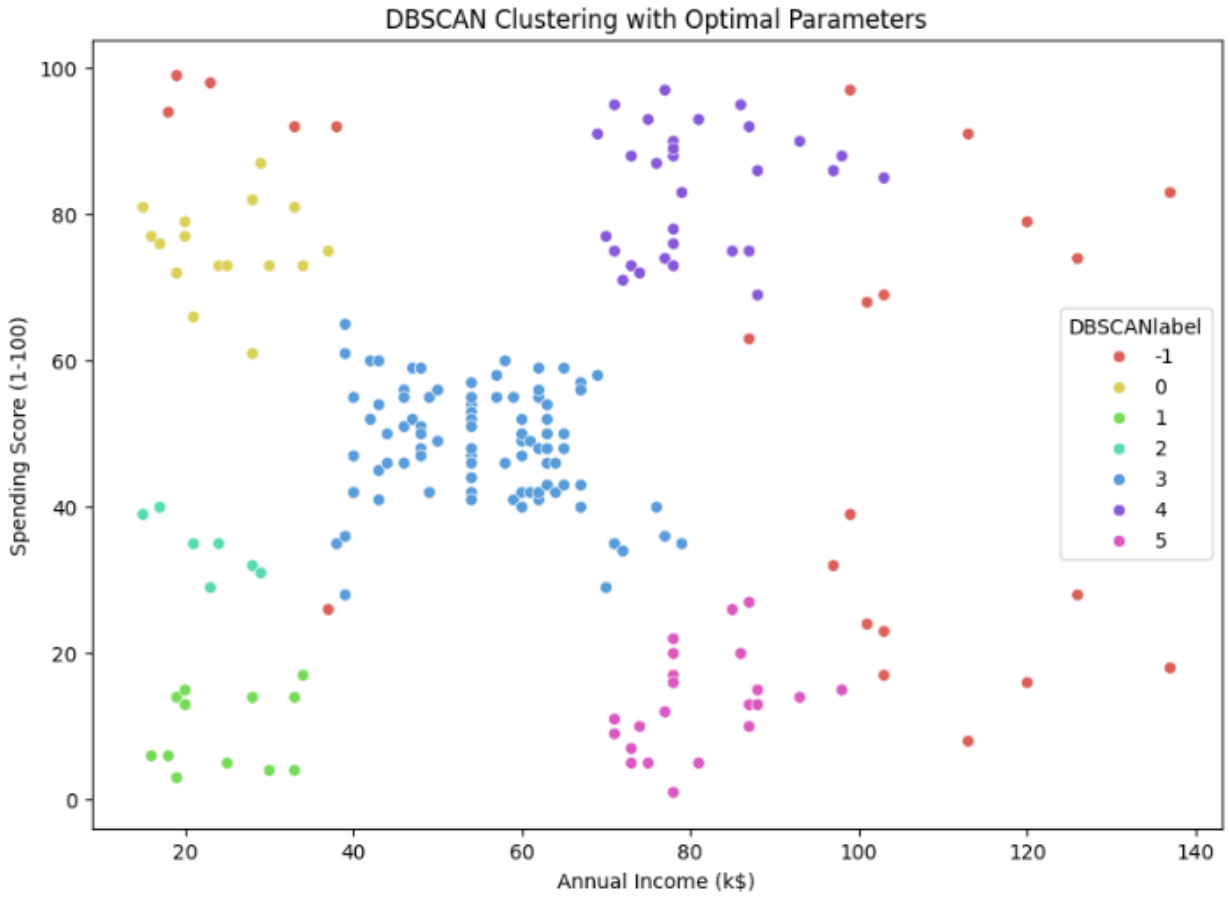
Min_samples needs to be equal or greater than 2 * selected features. As we have selected two features, it needs to be greater than 3.

Fitting DBSCAN to the Dataset with Optimal Parameters:
Once the optimal parameters are identified, the DBSCAN algorithm is fitted to the dataset using these parameters.

```
dbscan_optimal = DBSCAN(eps=9.0,
min_samples=5).fit(main_clustering_features)
data['DBSCANlabel'] = dbscan_optimal.labels_
```

The optimal parameters eps=9.0 and min_samples=3 were selected because they resulted in the highest silhouette score, indicating the best clustering solution.
The cluster labels obtained from the DBSCAN algorithm are added to the dataset for further analysis. This step is necessary to associate each data point with the cluster to which it belongs.

Visualizing the Clusters:
The final step is to visualize the clusters formed by the DBSCAN algorithm.



Cluster3: customers who have **moderate income and moderate spending**. They may be loyal and regular customers who are satisfied with the current products and services.
Cluster0: customers who have **low income and high spending**. They may be impulsive buyers who spend beyond their means or have high expectations and demands.
Cluster2 and Cluster1: customers who have **low income and low spending**. They may be frugal or dissatisfied customers who are looking for cheaper or better products and services elsewhere.
Cluster4: customers who have **high income and high spending**. They may be affluent or loyal customers who are willing to pay for premium products and services or have high satisfaction and retention rates.
Cluster5: customers who have **high income and low spending**. They may be cautious or conservative customers who have high saving rates or low trust and loyalty levels.
Label -1 are outliers.

Github repository link of the project : https://github.com/THamid02/Unsupervised_Clustering

-End of the Assignment-