Name: Jigar Siddhpura **SAPID:** 60004200155

DIV: C/C2 **Branch:** Computer Engineering

DMW EXPERIMENT 5

Aim: To implement various clustering algorithms

Theory:

The process of making a group of abstract objects into classes of similar objects is known as clustering.

One group is treated as a cluster of data objects

In the process of cluster analysis, the first step is to partition the set of data into groups with the help of data similarity, and then groups are assigned to their respective labels.

The biggest advantage of clustering over-classification is it can adapt to the changes made and helps single out useful features that differentiate different groups.

Applications of cluster analysis:

It is widely used in many applications such as image processing, data analysis, and pattern recognition.

It helps marketers to find the distinct groups in their customer base and they can characterize their customer groups by using purchasing patterns.

It can be used in the field of biology, by deriving animal and plant taxonomies and identifying genes with the same capabilities.

It also helps in information discovery by classifying documents on the web.

Clustering Methods:

- Model-Based Method
- Hierarchical Method
- Constraint-Based Method
- Grid-Based Method
- Partitioning Method
- Density-Based Method

Part A:

- Program using in-build functions
- Plot the clusters
- Plot dendrogram (for hierarchical)

K-Means Clustering Algorithm

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. In this topic, we will learn what is K-means clustering algorithm, how the algorithm works, along with the Python implementation of k-means clustering.

K-Means Clustering is an Unsupervised Algorithms, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

It is an **iterative algorithm** that divides the unlabeled dataset into k different clusters in sucha way that each dataset belongs only one group that has similar properties.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

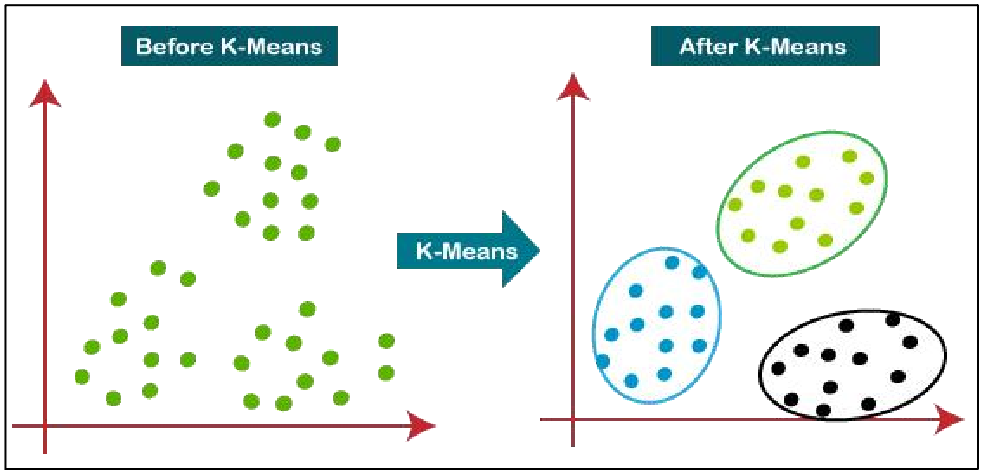
The k-means clustering algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.

- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.

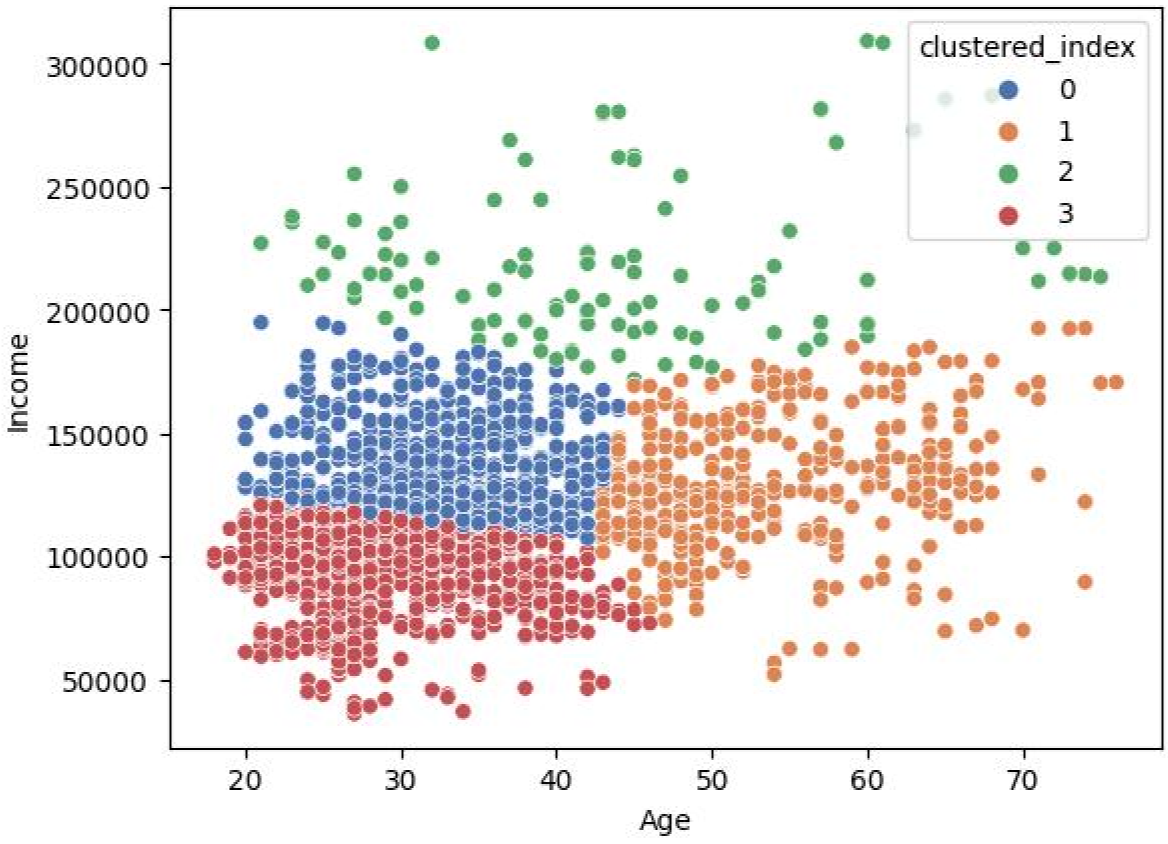
The below diagram explains the working of the K-means Clustering Algorithm:

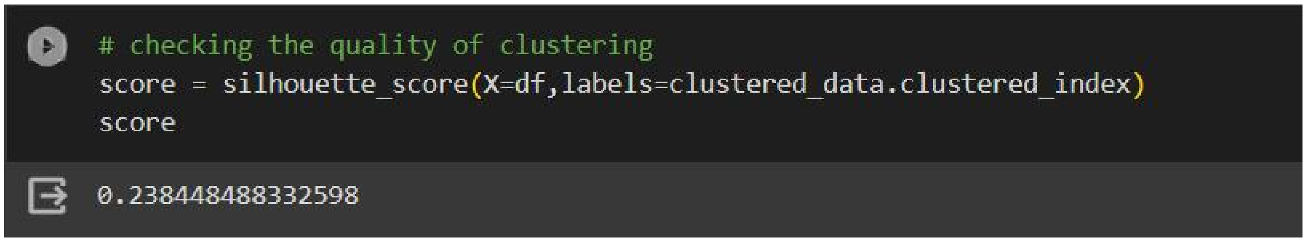


Program:

```
from google.colab import drive
drive.mount('/content/gdrive')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans, AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
df =
pd.read_csv('/content/gdrive/MyDrive/DMW/datasets/customer_segmentation.csv')
df.head()
df.drop(['ID'], inplace = True, axis = 1)
features = df[df.columns]
scaler = StandardScaler()
scaled = scaler.fit_transform(features.values)
scaled = pd.DataFrame(scaled,columns=df.columns)
scaled.head()
data = scaled[['Age','Income']]
# elbow curve
wcss = {'wcss_score':[],'no_of_clusters':[]}
for i in range(1,11):
  kmeans = KMeans(n_clusters=i,random_state=10)
  kmeans.fit(data)
  wcss['wcss_score'].append(kmeans.inertia_)
  wcss['no_of_clusters'].append(i)
plt.figure(figsize=(7,5))
plt.plot(wcss['no_of_clusters'],wcss['wcss_score'],marker='x')
plt.title("Elbow Method to determine number of clusters(K)")
plt.xlabel("K (no. of clusters)")
plt.ylabel("WCSS (Withing Cluster Sum of Squared distance )")

plt.show()
kmeans=KMeans(n_clusters=4,random_state=42)
kmeans.fit(data)
prediction = kmeans.fit_predict(data)
clustered_data = df.copy()
clustered_data['clustered_index'] = prediction
sns.scatterplot(x=clustered_data.Age, y=clustered_data.Income,
hue=clustered_data.clustered_index, palette='deep')
```

Output:





```
# checking the quality of clustering
score = silhouette_score(X=df,labels=clustered_data.clustered_index)
score
```

```
0.238448488332598
```

Hierarchical Clustering

Hierarchical clustering is another unsupervised machine learning algorithm, which is used to group the unlabeled datasets into a cluster and also known as hierarchical cluster analysis or HCA.

In this algorithm, we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the dendrogram.

Sometimes the results of K-means clustering and hierarchical clustering may look similar, but they both differ depending on how they work. As there is no requirement to predetermine the number of clusters as we did in the K-Means algorithm.

The hierarchical clustering technique has two approaches:

Agglomerative: Agglomerative is a bottom-up approach, in which the algorithm starts with taking all data points as single clusters and merging them until one cluster is left.

Divisive: Divisive algorithm is the reverse of the agglomerative algorithm as it is a top-down approach.

As we already have other clustering algorithms such as K-means, then why we need hierarchical clustering? So, as we have seen in the K-means clustering that there are some challenges with this algorithm, which are a predetermined number of clusters, and it always tries to create the clusters of the same size. To solve these two challenges, we can opt for the hierarchical clustering algorithm because, in this algorithm, we don't need to have knowledge about the predefined number of clusters.

In this topic, we will discuss the Agglomerative Hierarchical clustering algorithm.

Agglomerative Hierarchical clustering

The agglomerative hierarchical clustering algorithm is a popular example of HCA. To group the datasets into clusters, it follows the bottom-up approach. It means, this algorithm considers each dataset as a single cluster at the beginning, and then start combining the closest pair of clusters together. It does this until all the clusters are merged into a single cluster that contains all the datasets.

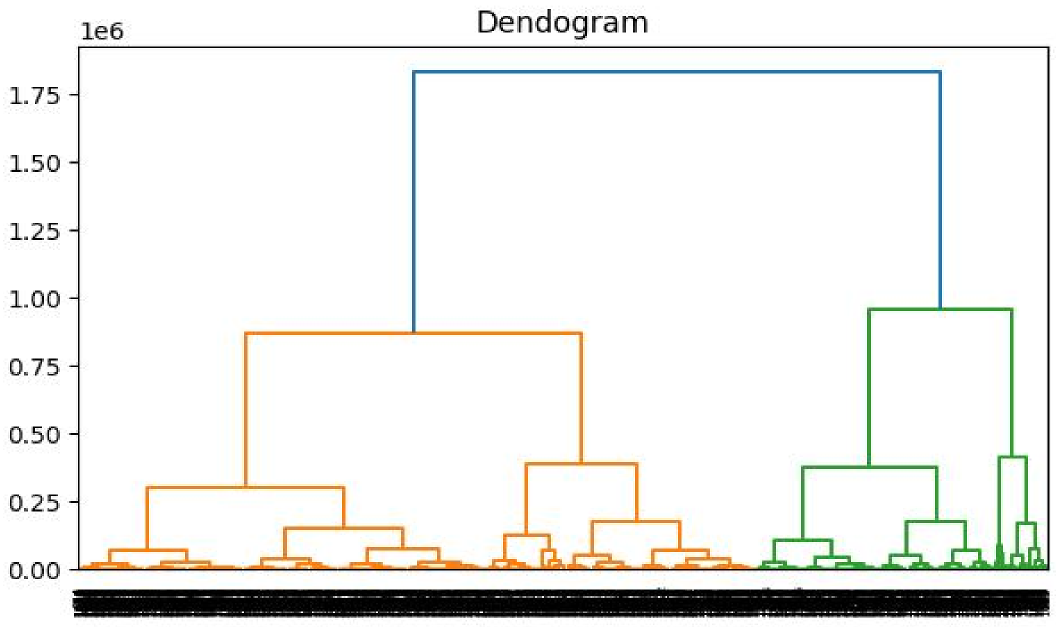
This hierarchy of clusters is represented in the form of the dendrogram.

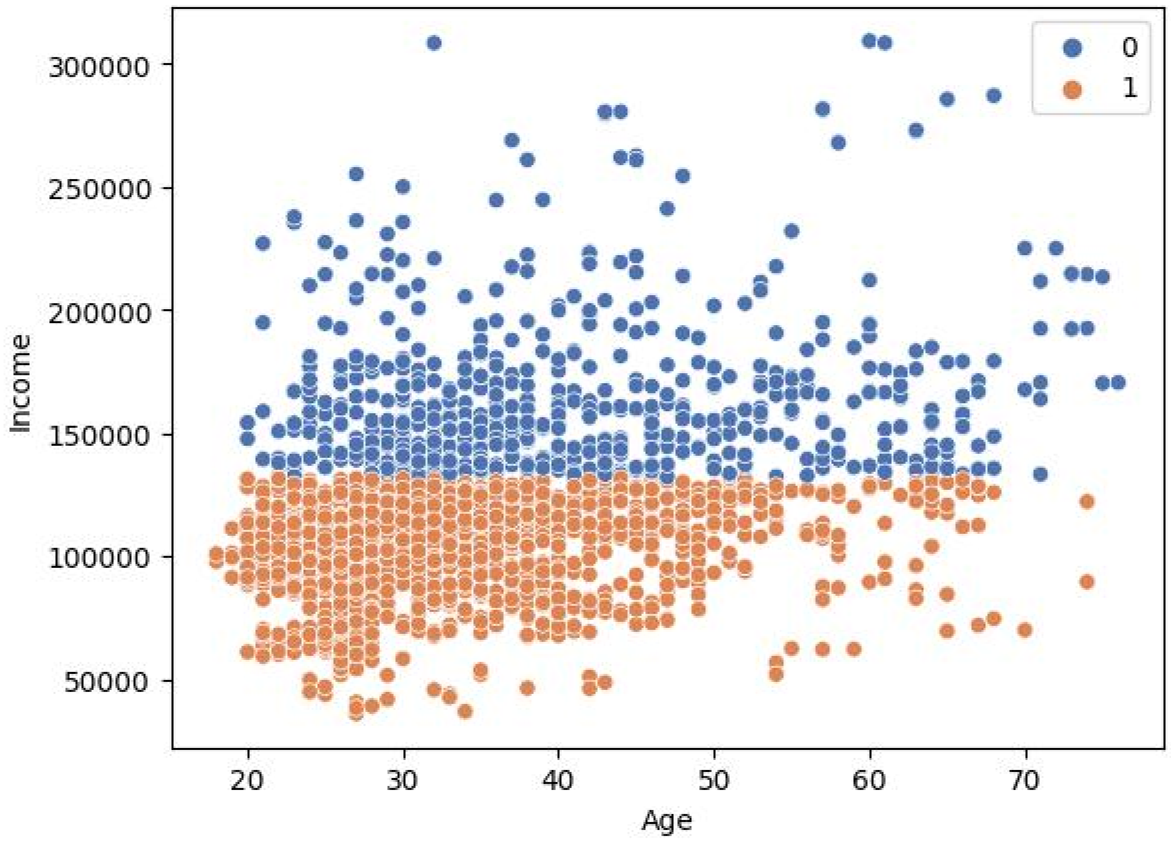
Program :

```
# Hierarchichal clustering
from scipy.cluster.hierarchy import dendrogram,linkage
data = clustered_data[['Age','Income']]

plt.figure(figsize=(10,7))
plt.title("Dendogram")
dend = dendrogram(linkage(data,method='ward'))
cluster =
AgglomerativeClustering(n_clusters=2,affinity='euclidean',linkage='ward')
labels_ = cluster.fit_predict(data)

sns.scatterplot(x=data.Age, y=data.Income, hue=labels_, palette='deep
```

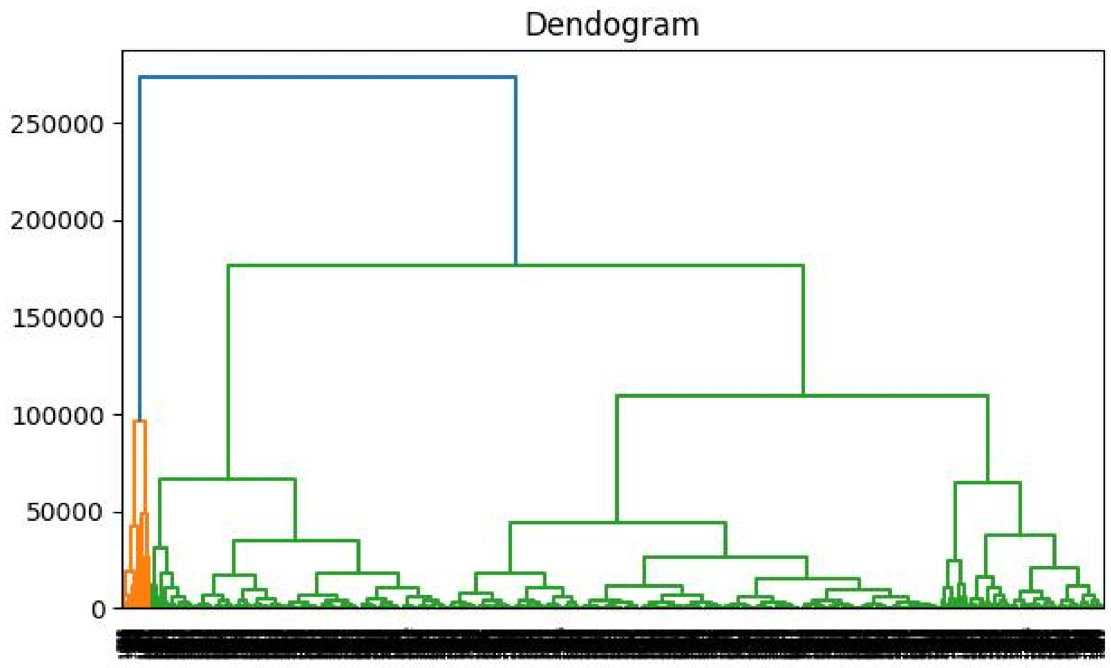
Output:

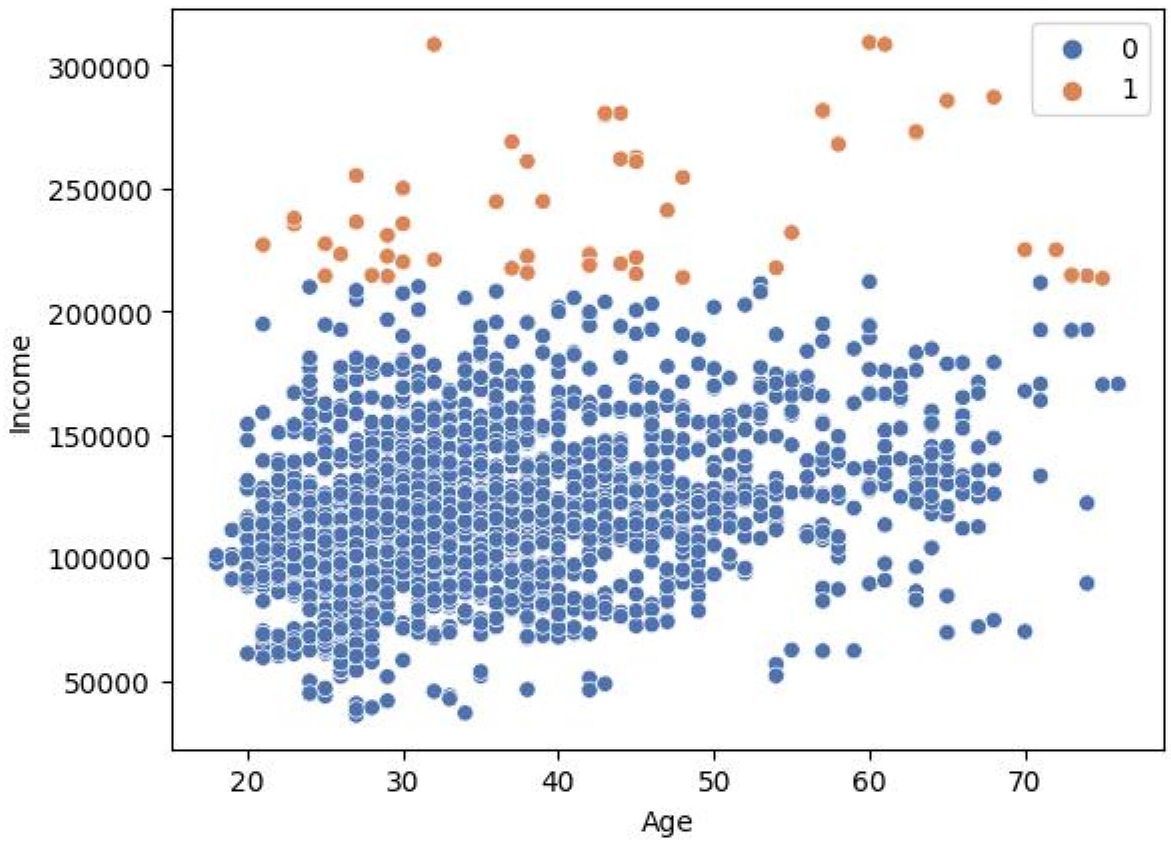
- ***Ward Hierarchical Clustering***



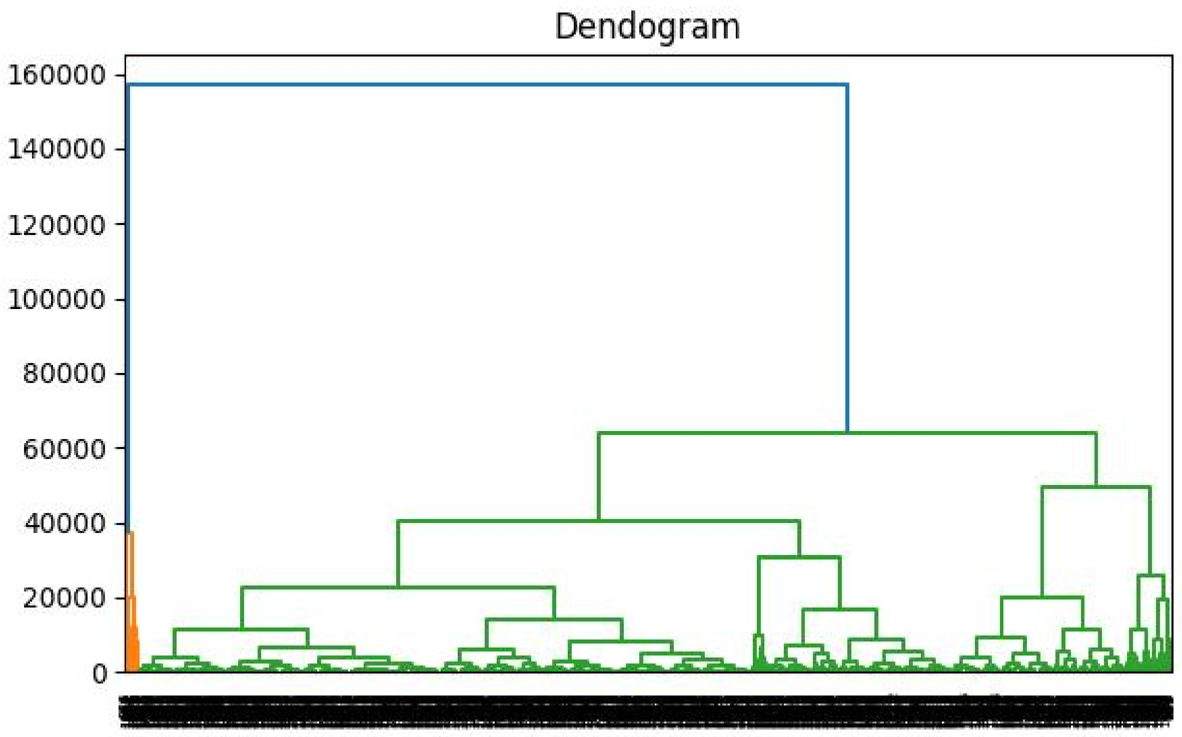


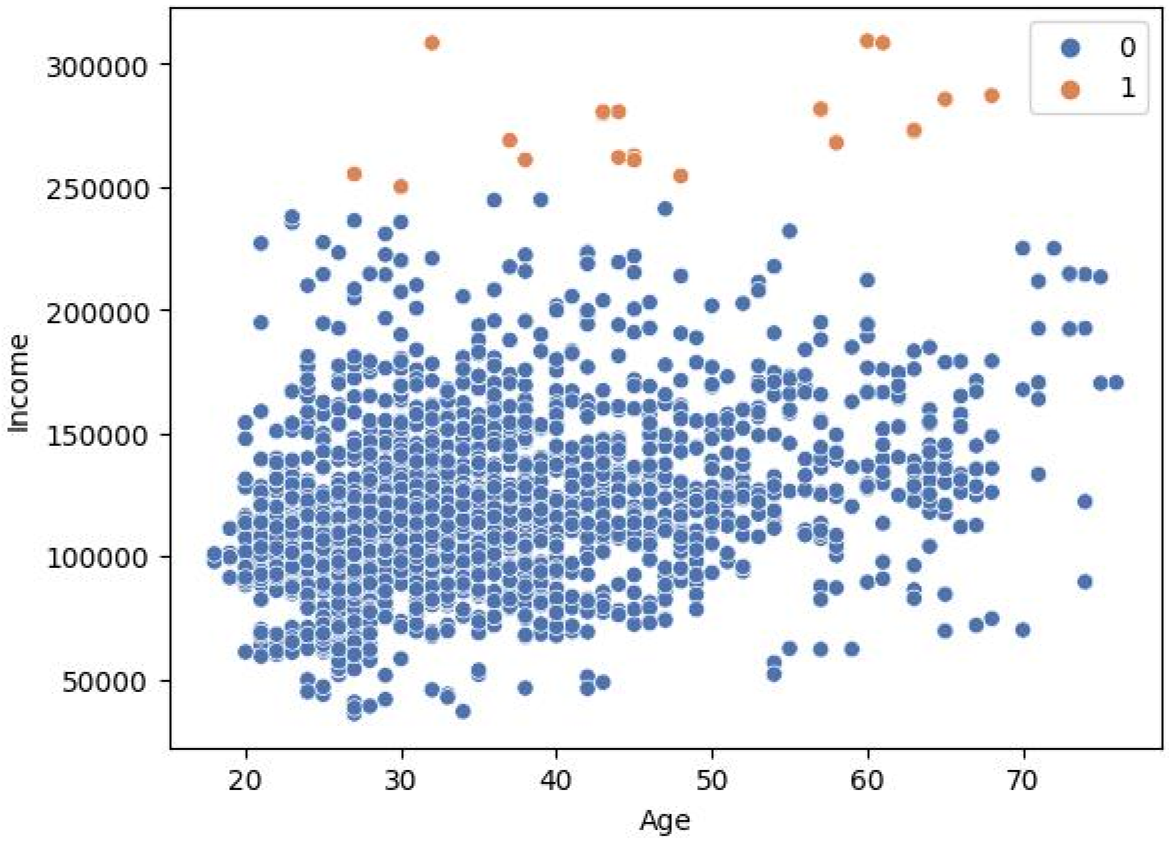
- ***Complete Hierarchical Clustering***





- ***Average Hierarchical Clustering***



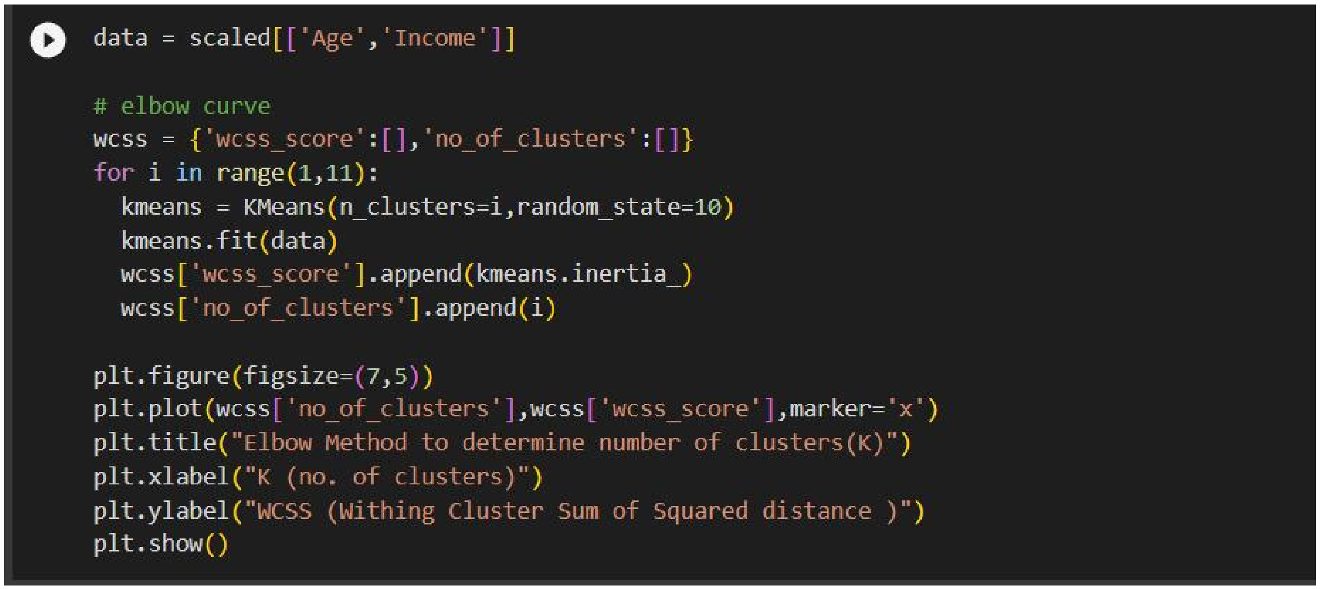


Part B:

1.Plot Elbow Method and suggest optimal number of clusters

A fundamental step for any unsupervised algorithm is to determine the optimal number of clusters into which the data may be clustered. The **Elbow Method** is one of the most popular methods to determine this optimal value of k.

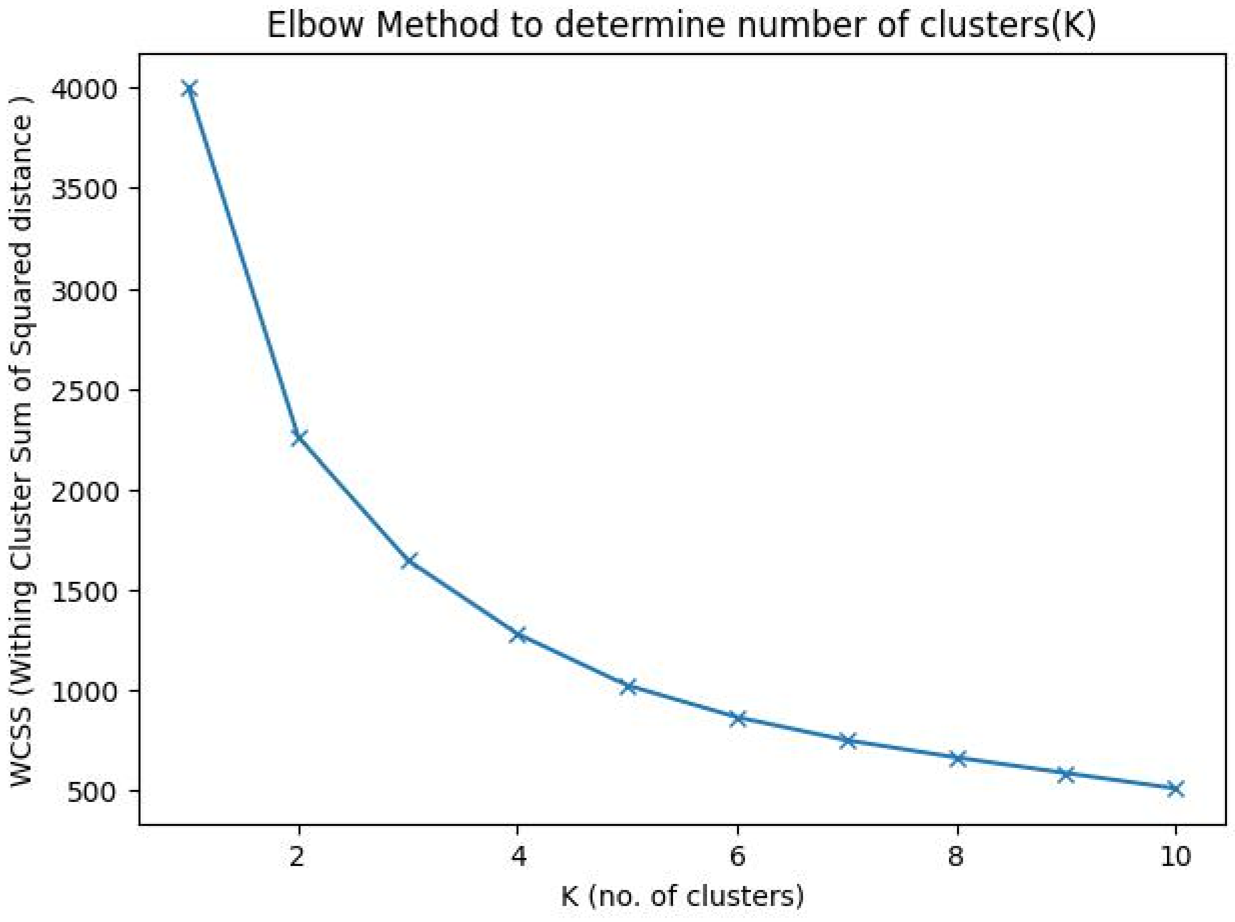
Program:



```python
data = scaled[['Age','Income']]

# elbow curve
wcss = {'wcss_score':[],'no_of_clusters':[]}
for i in range(1,11):
  kmeans = KMeans(n_clusters=i,random_state=10)
  kmeans.fit(data)
  wcss['wcss_score'].append(kmeans.inertia_)
  wcss['no_of_clusters'].append(i)

plt.figure(figsize=(7,5))
plt.plot(wcss['no_of_clusters'],wcss['wcss_score'],marker='x')
plt.title("Elbow Method to determine number of clusters(K)")
plt.xlabel("K (no. of clusters)")
plt.ylabel("WCSS (Withing Cluster Sum of Squared distance )")
plt.show()
```

To determine the optimal number of clusters, we have to select the value of k at the "elbow" i.e. the point after which the distortion/inertia start decreasing in a linear fashion. Thus, for the given data, we conclude that the optimal number of clusters for the data is 3.

Output :



Conclusion: Thus, we have successfully implemented Clustering Algorithm Using

1. k-means   2. Hierarchical(ward/complete/average)