Data Mining (CSE 5334)
Assignment 4 (Clustering Report)

Student details:

- Kuldip Rameshbhai Savaliya – 1001832000
- Shivani Manojkumar Panchiwala – 1001982478
- Meghaben Ghanshyambhai Patel – 1002006777

Clustering:

An abstract object is clustered into classes based on their similarities. There are several types of data objects that can be considered one group. During cluster analysis, the data is first divided into groups based on similarity, then the labels are assigned to the groups. Overclassification by clustering has multiple advantages, including the ability to adapt to change and to distinguish between different groups with useful features.

Task 1

Import necessary packages and library.

```
# import necessary packages and library
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import MinMaxScaler
```

Read datsaet.

```
#Read dataset
df= pd.read_csv('clusteringdata.csv')
df.head()
```

	Age	WorkClass	Fnlwght	Education	EducationNumber	MaritalStatus	Occupation	Relationship	Race	Sex	CapitalGain	CapitalLoss	HoursPerWeek	N
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	

Checking for null values.

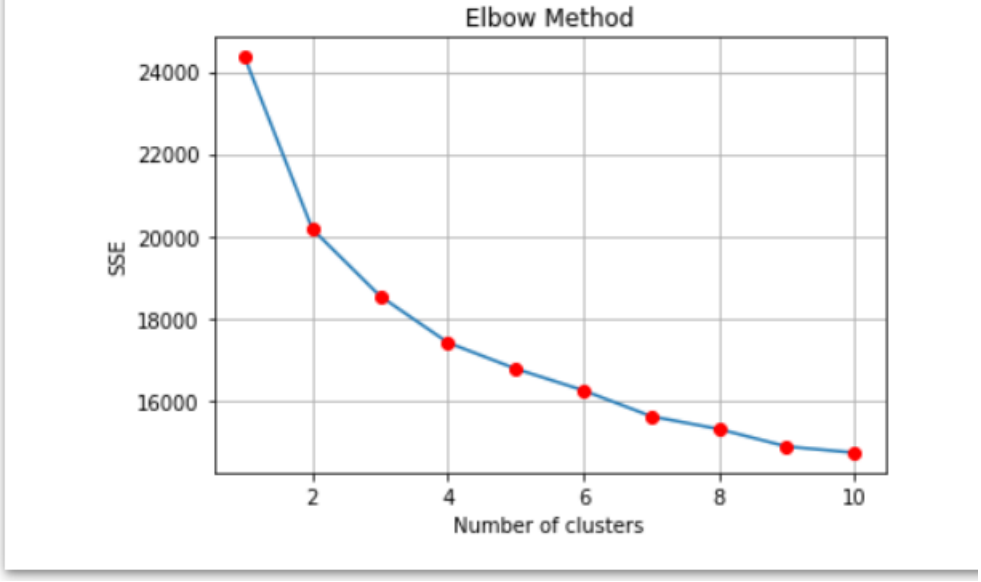
```
# checking any null values present or not
df.isnull().sum()
```

```
Age               0
WorkClass         0
Fnlwght           0
Education         0
EducationNumber   0
MaritalStatus     0
Occupation        0
Relationship      0
Race              0
Sex               0
CapitalGain       0
CapitalLoss       0
HoursPerWeek      0
NativeCountry     0
Class             0
dtype: int64
```

Elbow method for k =1 to 10.

The Elbow Method is one of the most popular methods to determine the optimal value of k.

```
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('SSE')
plt.grid()
plt.show()
```

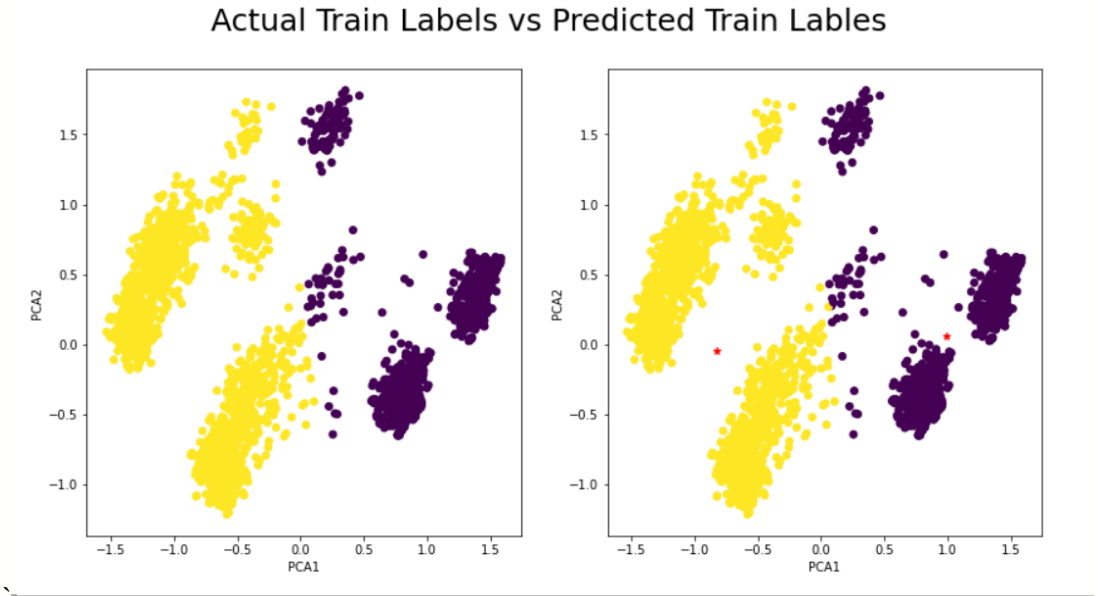


From the above graph, In elbow method we can see that the Number of clusters vary from k= 1 to 10 and for every value of K SSE is vary according to K. If the Number of clusters increas e the SSE is decrease and we can see that for k=4 graph changes rapidly and make an elbow s hape.

Visulization for K-means clusteing

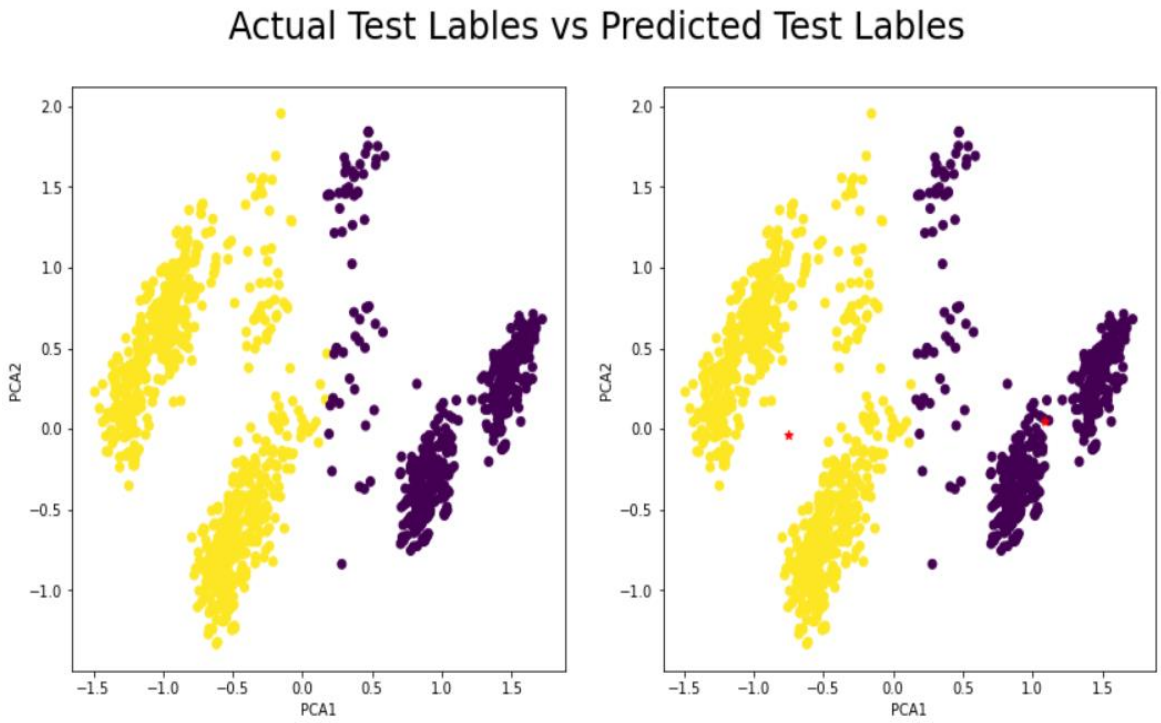
K=2

Actual Training Labels Vs Predicted Training Labels



From the Actual Train Labels vs Predicted Train Lables graph, we can see that red marks are the centroids of the clusters. Model divides the data into two clusters for k=2 which are in the subplot.

Actual Testing Labels Vs Predicted Testing Labels



Confusion Matrix

```
##################begin code for Task 1-b-4: Print out a confusion matrix
from sklearn.metrics import confusion_matrix,classification_report
print(confusion_matrix(y_test, test_kmeans))
```

```
[[612   0]
 [  2 886]]
```

Task 2

Hierarchical Agglomerative Clustering

Hierarchical Agglomerative Clustering is also known as the bottom-up approach. We begin by separating each object into its own group. In the process, close objects or groups are constantly merged. In this way, it continues until all groups are merged into one or until the termination condition is met.

Calculate pairwise distance

```
## Calculate distance
cosine_distance = pairwise_distances(train_pca,metric='cosine')
euclidean_distance = pairwise_distances(train_pca,metric='euclidean')
manhattan_distance = pairwise_distances(train_pca,metric='manhattan')
```

F-1 score for complete and average linkage

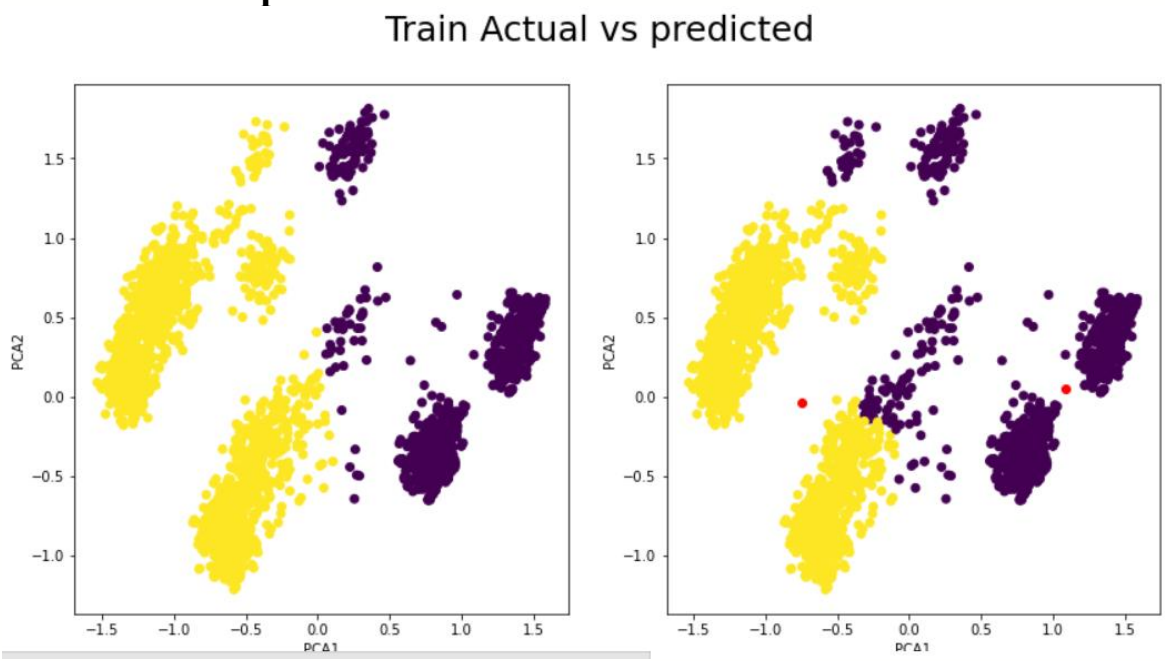
```
print("F1-score for complete linkage + cosine", f1_cos_complete)
print("F1-score for complete linkage + euclidean", f1_euc_complete)
print("F1-score for complete linkage + manhattan", f1_man_complete)
print("F1-score for average linkage + cosine", f1_cos_average)
print("F1-score for average linkage + euclidean", f1_euc_average)
print("F1-score for average linkage + manhattan", f1_man_average)

####################end code for Task 2-a

####################begin code for Task 2-b: Visualize the predicted tra
```

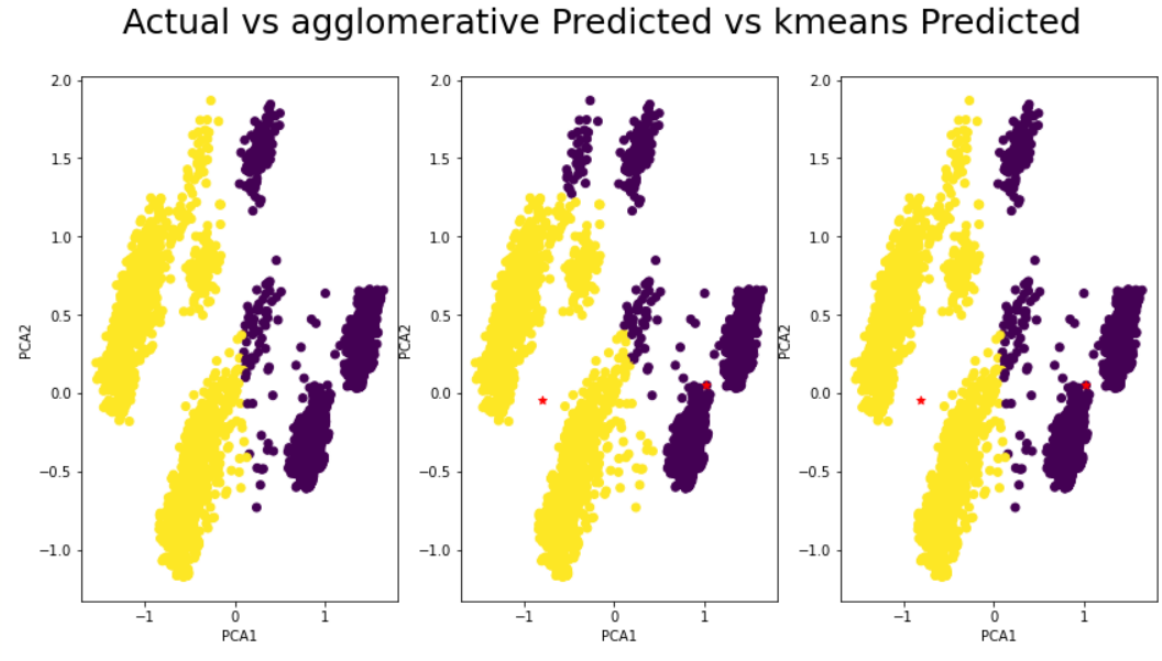
```
F1-score for complete linkage + cosine 0.7120189852615983
F1-score for complete linkage + euclidean 0.9754687683839826
F1-score for complete linkage + manhattan 0.9589398367597409
F1-score for average linkage + cosine 0.03388800309691156
F1-score for average linkage + euclidean 0.03424596409913979
F1-score for average linkage + manhattan 0.03424596409913979
```

Visualization for Hierarchical Agglomerative Clustering
Train actual Vs predicted



From the above graph displays the Train Actual Vs Predicted labels for train data. The model divides the data into two clusters for k=2 which are display in the subplot.

Compare K-means and Hierarchical Agglomerative clustering.



From the above graph we can see that the subplots of actual vs agglomerative predicted value s vs kmeans predicted values.

Confusion matrix for Kmeans and Agglomerative.

```
Confusion Matrix for KMeans:

 [[   0 2203]
 [2797    0]]
**********
Confusion Matrix for Agglomerative:

 [[2181   22]
 [  38 2759]]
**********
Precision KMeans: 1.0    Precision Agglomerative: 0.992089176555196
Recall KMeans: 1.0       Recall Agglomerative: 0.9864140150160887
F1_score KMeans: 1.0     F1_score Agglomerative: 0.9892434564359986
**********
```