Assignment 5 – Report

Name: Prudhvi Mahesh Meka

Network ID: 700738978

CRN: 12664

1. Principal Component Analysis
    - Apply PCA on the CC dataset.
    - Apply the k-means algorithm on the PCA result and report your observation if the silhouette score has improved or not.
    - Perform Scaling+PCA+K-Means and report performance.

In [1]:
```python
# initially importing all the required libraries.
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn import preprocessing, metrics
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
sns.set(style="white", color_codes=True)
import warnings
warnings.filterwarnings("ignore")
```

- Here, I imported all the required libraries such as NumPy, Pandas, matplotlib etc.
- After that imported the given data set and read “cc.csv” file.
- To apply PCA we need to remove all the null values that are present in the dataset. So, removed all the null values and applied PCA to the dataset

In [3]:
```python
dataset_CC = pd.read_csv('datasets//CC.csv')
dataset_CC.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8950 entries, 0 to 8949
Data columns (total 18 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   CUST_ID                           8950 non-null   object
 1   BALANCE                           8950 non-null   float64
 2   BALANCE_FREQUENCY                 8950 non-null   float64
 3   PURCHASES                         8950 non-null   float64
 4   ONEOFF_PURCHASES                  8950 non-null   float64
 5   INSTALLMENTS_PURCHASES            8950 non-null   float64
 6   CASH_ADVANCE                      8950 non-null   float64
 7   PURCHASES_FREQUENCY               8950 non-null   float64
 8   ONEOFF_PURCHASES_FREQUENCY        8950 non-null   float64
 9   PURCHASES_INSTALLMENTS_FREQUENCY  8950 non-null   float64
 10  CASH_ADVANCE_FREQUENCY            8950 non-null   float64
 11  CASH_ADVANCE_TRX                  8950 non-null   int64
 12  PURCHASES_TRX                     8950 non-null   int64
 13  CREDIT_LIMIT                      8949 non-null   float64
 14  PAYMENTS                          8950 non-null   float64
 15  MINIMUM_PAYMENTS                  8637 non-null   float64
 16  PRC_FULL_PAYMENT                  8950 non-null   float64
 17  TENURE                            8950 non-null   int64
dtypes: float64(14), int64(3), object(1)
memory usage: 1.2+ MB
```

In [4]:
```python
dataset_CC.head()
```

Out[4]:

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FRE
0	C10001	40.900749	0.818182	95.40	0.00	95.4	0.000000	
1	C10002	3202.467416	0.909091	0.00	0.00	0.0	6442.945483	
2	C10003	2495.148862	1.000000	773.17	773.17	0.0	0.000000	
3	C10004	1666.670542	0.636364	1499.00	1499.00	0.0	205.788017	
4	C10005	817.714335	1.000000	16.00	16.00	0.0	0.000000	

- dataset_CC.head() it gives first 5 records.

```
In [5]:  dataset_CC.isnull().any()

Out[5]: CUST_ID                             False
        BALANCE                             False
        BALANCE_FREQUENCY                   False
        PURCHASES                           False
        ONEOFF_PURCHASES                    False
        INSTALLMENTS_PURCHASES              False
        CASH_ADVANCE                        False
        PURCHASES_FREQUENCY                 False
        ONEOFF_PURCHASES_FREQUENCY          False
        PURCHASES_INSTALLMENTS_FREQUENCY    False
        CASH_ADVANCE_FREQUENCY              False
        CASH_ADVANCE_TRX                    False
        PURCHASES_TRX                       False
        CREDIT_LIMIT                         True
        PAYMENTS                            False
        MINIMUM_PAYMENTS                     True
        PRC_FULL_PAYMENT                    False
        TENURE                              False
        dtype: bool
```

It checks with CC having any null values.

```
In [6]:  dataset_CC.fillna(dataset_CC.mean(), inplace=True)
         dataset_CC.isnull().any()

Out[6]: CUST_ID                             False
        BALANCE                             False
        BALANCE_FREQUENCY                   False
        PURCHASES                           False
        ONEOFF_PURCHASES                    False
        INSTALLMENTS_PURCHASES              False
        CASH_ADVANCE                        False
        PURCHASES_FREQUENCY                 False
        ONEOFF_PURCHASES_FREQUENCY          False
        PURCHASES_INSTALLMENTS_FREQUENCY    False
        CASH_ADVANCE_FREQUENCY              False
        CASH_ADVANCE_TRX                    False
        PURCHASES_TRX                       False
        CREDIT_LIMIT                        False
        PAYMENTS                            False
        MINIMUM_PAYMENTS                    False
        PRC_FULL_PAYMENT                    False
        TENURE                              False
        dtype: bool
```

```
In [7]:  x = dataset_CC.iloc[:,1:-1]
         y = dataset_CC.iloc[:,-1]
         print(x.shape,y.shape)

         (8950, 16) (8950,)
```

Here, we applied the mean and got all the null values and

then recorded all the entries between 1 and -1.

```
In [9]:  pca = PCA(3)
         x_pca = pca.fit_transform(x)
         principalDf = pd.DataFrame(data = x_pca, columns = ['principal component 1', 'principal component 2', 'principal component 3'
         finalDf = pd.concat([principalDf, dataset_CC.iloc[:,-1]], axis = 1)
         finalDf.head()
```

Out[9]:

	principal component 1	principal component 2	principal component 3	TENURE
0	-4326.383979	921.566882	183.708383	12
1	4118.916665	-2432.846346	2369.969289	12
2	1497.907641	-1997.578694	-2125.631328	12
3	1394.548536	-1488.743453	-2431.799649	12
4	-3743.351896	757.342657	512.476492	12

- We got the illustration of first five entries in principle component 1, component 2, component 3 and Tenure.

In [10]:
```python
X = finalDf.iloc[:,0:-1]
y = finalDf.iloc[:,-1]
```

In [11]:
```python
nclusters = 3 # this is the k in kmeans
km = KMeans(n_clusters=nclusters)
km.fit(X)

# predict the cluster for each data point
y_cluster_kmeans = km.predict(X)


# Summary of the predictions made by the classifier
print(classification_report(y, y_cluster_kmeans, zero_division=1))
print(confusion_matrix(y, y_cluster_kmeans))


train_accuracy = accuracy_score(y, y_cluster_kmeans)
print("\nAccuracy for our Training dataset with PCA:", train_accuracy)


#Calculate sihouette Score
score = metrics.silhouette_score(X, y_cluster_kmeans)
print("Sihouette Score: ",score)

"""
Sihouette Score- ranges from -1 to +1 , a high value indicates that the object is well matched to its own cluster and poorly
"""
```

- Now, got the silhouette score between 1 and -1.
- A high value indicates that the object is well matched to its own cluster and poorly matched to neighbouring clusters.

```
              precision    recall  f1-score   support

           0       0.00      1.00      0.00       0.0
           1       0.00      1.00      0.00       0.0
           2       0.00      1.00      0.00       0.0
           6       1.00      0.00      0.00     204.0
           7       1.00      0.00      0.00     190.0
           8       1.00      0.00      0.00     196.0
           9       1.00      0.00      0.00     175.0
          10       1.00      0.00      0.00     236.0
          11       1.00      0.00      0.00     365.0
          12       1.00      0.00      0.00    7584.0

    accuracy                           0.00    8950.0
   macro avg       0.70      0.30      0.00    8950.0
weighted avg       1.00      0.00      0.00    8950.0

[[   0    0    0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0]
```

In [12]:
```python
x = dataset_CC.iloc[:,1:-1]
y = dataset_CC.iloc[:,-1]
print(x.shape,y.shape)
```

```
(8950, 16) (8950,)
```

In [13]:
```python
#Scaling
scaler = StandardScaler()
scaler.fit(x)
X_scaled_array = scaler.transform(x)
#PCA
pca = PCA(3)
x_pca = pca.fit_transform(X_scaled_array)
principalDf = pd.DataFrame(data = x_pca, columns = ['principal component 1', 'principal component 2','principal component 3']
finalDf = pd.concat([principalDf, dataset_CC.iloc[:,-1]], axis = 1)
finalDf.head()
```

Out[13]:

	principal component 1	principal component 2	principal component 3	TENURE
0	-1.718893	-1.072939	0.535670	12
1	-1.169306	2.509320	0.628027	12
2	0.938414	-0.382600	0.161198	12
3	-0.907503	0.045859	1.521689	12
4	-1.637830	-0.684975	0.425658	12

- After applying the scaling, we got the silhouette score between 1 and -1.

In [14]:
```
X = finalDf.iloc[:,0:-1]
y = finalDf["TENURE"]
print(X.shape,y.shape)
```

```
(8950, 3) (8950,)
```

In [15]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.34,random_state=0)
nclusters = 3
# this is the k in kmeans
km = KMeans(n_clusters=nclusters)
km.fit(X_train,y_train)


# predict the cluster for each training data point
y_clus_train = km.predict(X_train)

# Summary of the predictions made by the classifier
print(classification_report(y_train, y_clus_train, zero_division=1))
print(confusion_matrix(y_train, y_clus_train))

train_accuracy = accuracy_score(y_train, y_clus_train)
print("Accuracy for our Training dataset with PCA:", train_accuracy)

#Calculate sihouette Score
score = metrics.silhouette_score(X_train, y_clus_train)
print("Sihouette Score: ",score)

"""
Sihouette Score- ranges from -1 to +1 , a high value indicates that the object is well matched to its own cluster and poorly
"""
```

```
              precision    recall  f1-score   support

           0       0.00      1.00      0.00       0.0
           1       0.00      1.00      0.00       0.0
           2       0.00      1.00      0.00       0.0
           6       1.00      0.00      0.00     139.0
           7       1.00      0.00      0.00     135.0
           8       1.00      0.00      0.00     128.0
           9       1.00      0.00      0.00     118.0
          10       1.00      0.00      0.00     151.0
          11       1.00      0.00      0.00     262.0
          12       1.00      0.00      0.00    4974.0

    accuracy                           0.00    5907.0
   macro avg       0.70      0.30      0.00    5907.0
weighted avg       1.00      0.00      0.00    5907.0

[[   0    0    0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0]
 [ 105   30    4    0    0    0    0    0    0    0]
 [ 108   26    1    0    0    0    0    0    0    0]
 [  96   28    4    0    0    0    0    0    0    0]
 [  89   27    2    0    0    0    0    0    0    0]
 [ 107   38    6    0    0    0    0    0    0    0]
 [ 185   66   11    0    0    0    0    0    0    0]
 [3393  842  739    0    0    0    0    0    0    0]]
Accuracy for our Training dataset with PCA: 0.0
Sihouette Score:  0.3812076198524835
```

[15]:
```
'\nSihouette Score- ranges from -1 to +1 , a high value indicates that the object is well matched to its own cluster and poo
rly matched to neighboring clusters.\n'
```

In [16]:
```
# predict the cluster for each testing data point
y_clus_test = km.predict(X_test)

# Summary of the predictions made by the classifier
print(classification_report(y_test, y_clus_test, zero_division=1))
print(confusion_matrix(y_test, y_clus_test))

train_accuracy = accuracy_score(y_test, y_clus_test)
print("\nAccuracy for our Training dataset with PCA:", train_accuracy)

#Calculate sihouette Score
score = metrics.silhouette_score(X_test, y_clus_test)
print("Sihouette Score: ",score)

"""
Sihouette Score- ranges from -1 to +1 , a high value indicates that the object is well matched to its own cluster and poorly
"""
```

```
              precision    recall  f1-score   support

           0       0.00      1.00      0.00       0.0
           1       0.00      1.00      0.00       0.0
           2       0.00      1.00      0.00       0.0
           6       1.00      0.00      0.00      65.0
           7       1.00      0.00      0.00      55.0
           8       1.00      0.00      0.00      68.0
           9       1.00      0.00      0.00      57.0
          10       1.00      0.00      0.00      85.0
          11       1.00      0.00      0.00     103.0
          12       1.00      0.00      0.00    2610.0

    accuracy                           0.00    3043.0
   macro avg       0.70      0.30      0.00    3043.0
weighted avg       1.00      0.00      0.00    3043.0

[[   0    0    0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0]
 [  41   21    3    0    0    0    0    0    0    0]
 [  42   12    1    0    0    0    0    0    0    0]
 [  57   10    1    0    0    0    0    0    0    0]
 [  35   22    0    0    0    0    0    0    0    0]
 [  63   17    5    0    0    0    0    0    0    0]
 [  69   30    4    0    0    0    0    0    0    0]
 [1763  450  397    0    0    0    0    0    0    0]]

Accuracy for our Training dataset with PCA: 0.0
Sihouette Score:  0.383322340968964
```

Out[16]: '\nSihouette Score- ranges from -1 to +1 , a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters.\n'

In [18]:
```
dataset_pd = pd.read_csv('datasets//pd_speech_features.csv')
dataset_pd.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 756 entries, 0 to 755
Columns: 755 entries, id to class
dtypes: float64(749), int64(6)
memory usage: 4.4 MB
```

In [19]:
```
dataset_pd.head()
```

Out[19]:

	id	gender	PPE	DFA	RPDE	numPulses	numPeriodsPulses	meanPeriodPulses	stdDevPeriodPulses	locPctJitter	...	tqwt_kurtosisValue_dec_2
0	0	1	0.85247	0.71826	0.57227	240	239	0.008064	0.000087	0.00218	...	1.562
1	0	1	0.76686	0.69481	0.53966	234	233	0.008258	0.000073	0.00195	...	1.558
2	0	1	0.85083	0.67604	0.58982	232	231	0.008340	0.000060	0.00176	...	1.564
3	1	0	0.41121	0.79672	0.59257	178	177	0.010858	0.000183	0.00419	...	3.780
4	1	0	0.32790	0.79782	0.53028	236	235	0.008162	0.002669	0.00535	...	6.172

5 rows × 755 columns

- Here, we read “pd_speech_features.csv” and got the five entries from the dataset.

In [20]:
```
dataset_pd.isnull().any()
```

Out[20]:
```
id                          False
gender                      False
PPE                         False
DFA                         False
RPDE                        False
                             ...
tqwt_kurtosisValue_dec_33   False
tqwt_kurtosisValue_dec_34   False
tqwt_kurtosisValue_dec_35   False
tqwt_kurtosisValue_dec_36   False
class                       False
Length: 755, dtype: bool
```

- It collected all the Null values.

In [21]:
```
X = dataset_pd.drop('class',axis=1).values
y = dataset_pd['class'].values
```

In [22]:
```
#Scaling Data
scaler = StandardScaler()
X_Scale = scaler.fit_transform(X)
```

In [23]:
```
# Apply PCA with k =3
pca3 = PCA(n_components=3)
principalComponents = pca3.fit_transform(X_Scale)

principalDf = pd.DataFrame(data = principalComponents, columns = ['principal component 1', 'principal component 2','Principal

finalDf = pd.concat([principalDf, dataset_pd[['class']]], axis = 1)
finalDf.head()
```

Out[23]:

	principal component 1	principal component 2	Principal Component 3	class
0	-10.047372	1.471076	-6.846402	1
1	-10.637725	1.583749	-6.830976	1
2	-13.516185	-1.253542	-6.818696	1
3	-9.155084	8.833601	15.290906	1
4	-6.764470	4.611468	15.637121	1

- Here, we applied principal component analysis at three stages and class.

In [24]:
```
X = finalDf.drop('class',axis=1).values
y = finalDf['class'].values
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.34,random_state=0)
```

In [25]:
```
#2.c Support Vector Machine's

from sklearn.svm import SVC

svmClassifier = SVC()
svmClassifier.fit(X_train, y_train)

y_pred = svmClassifier.predict(X_test)

# Summary of the predictions made by the classifier
print(classification_report(y_test, y_pred, zero_division=1))
print(confusion_matrix(y_test, y_pred))
# Accuracy score
glass_acc_svc = accuracy_score(y_pred,y_test)
print('accuracy is',glass_acc_svc )

#Calculate sihouette Score
score = metrics.silhouette_score(X_test, y_pred)
print("Sihouette Score: ",score)
```

```
              precision    recall  f1-score   support

           0       0.67      0.42      0.51        62
           1       0.84      0.93      0.88       196

    accuracy                           0.81       258
   macro avg       0.75      0.68      0.70       258
weighted avg       0.80      0.81      0.79       258

[[ 26  36]
 [ 13 183]]
accuracy is 0.810077519379845
Sihouette Score:  0.2504463929631217
```

-
- Now, we imported support vector machine and got the silhouette score between 1 and -1.

In [26]:
```
#3.Apply Linear Discriminant Analysis (LDA) on Iris.csv dataset to reduce dimensionality of data to k=2.
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
dataset_iris = pd.read_csv('datasets//Iris.csv')
dataset_iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [27]:
```
dataset_iris.isnull().any()
```

Out[27]:
```
Id               False
SepalLengthCm    False
SepalWidthCm     False
PetalLengthCm    False
PetalWidthCm     False
Species          False
dtype: bool
```

In [28]:
```
x = dataset_iris.iloc[:,1:-1]
y = dataset_iris.iloc[:,-1]
print(x.shape,y.shape)
```

```
(150, 4) (150,)
```

In [29]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)
```

In [30]:
```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
le = LabelEncoder()
y = le.fit_transform(y)
```

In [31]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
lda = LDA(n_components=2)
X_train = lda.fit_transform(X_train, y_train)
X_test = lda.transform(X_test)
print(X_train.shape,X_test.shape)
```

```
(105, 2) (45, 2)
```

- LDA and PCA both use linear transformations to maximize variance in a lower dimension. The PCA algorithm is an unsupervised learning algorithm, whereas the LDA algorithm is a supervised learning algorithm. This means that P CA finds maximum variance directions regardless of class labels, whereas LDA finds maximum class separability dir ections.

- It condenses the features into a smaller set of orthogonal variables known as principal components, which are line ar combinations of the original variables. The first component captures the most variability in the data, the secon d the second most, and so on

- LDA seeks linear discriminants in order to maximize variance between categories while minimizing variance within the class.