MACHINE LEARNING ASSIGNMENT 5

NAME: SUPRIYA SAMA
700744510

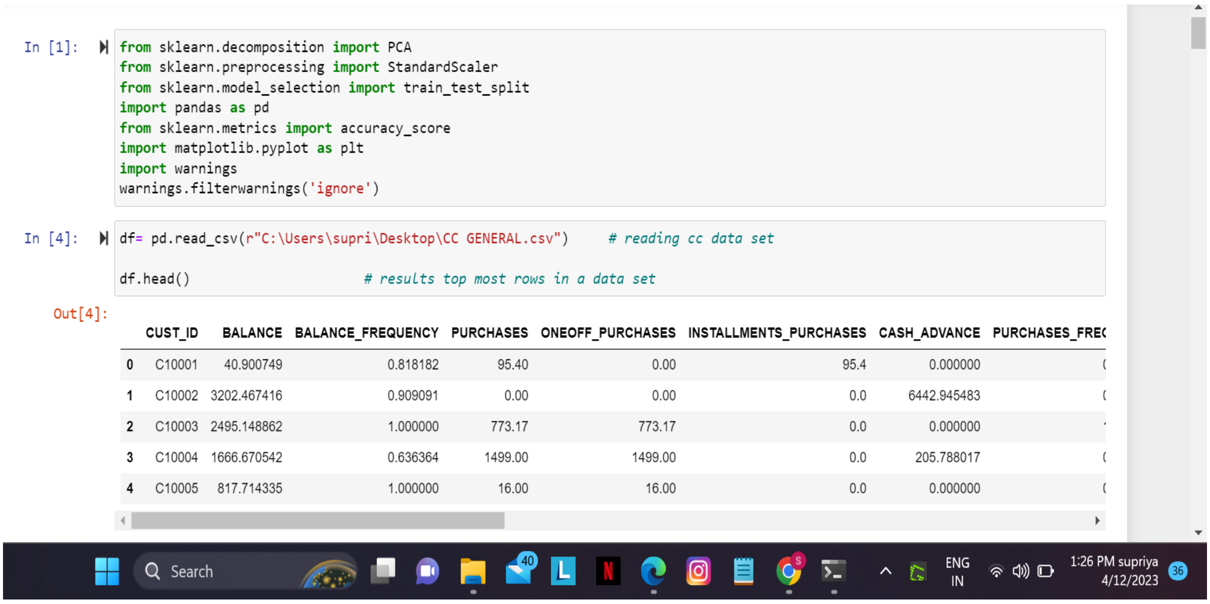
Video link:
https://drive.google.com/file/d/1TsVWovJEi8sMhgMhuhUSWspXUC0Pihdv/view?usp=share_link

1. Principal Component Analysis

a. Apply PCA on CC dataset.

I have imported few python libraries for data analysis and to apply machine learning algorithms.

Read_csv method is used to import the “cc general” data set. **head( )** method of pandas library results topmost rows of data set.



```
In [1]:
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import pandas as pd
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

```
In [4]:
df= pd.read_csv(r"C:\Users\supri\Desktop\CC GENERAL.csv")     # reading cc data set

df.head()                    # results top most rows in a data set
```

Out[4]:

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREC
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	(

Isnull( ) method of pandas library checks for any values present in data set.

```
In [6]:  df.isnull().sum()    #checking any null values are present

Out[6]: CUST_ID                              0
        BALANCE                              0
        BALANCE_FREQUENCY                    0
        PURCHASES                            0
        ONEOFF_PURCHASES                     0
        INSTALLMENTS_PURCHASES               0
        CASH_ADVANCE                         0
        PURCHASES_FREQUENCY                  0
        ONEOFF_PURCHASES_FREQUENCY           0
        PURCHASES_INSTALLMENTS_FREQUENCY     0
        CASH_ADVANCE_FREQUENCY               0
        CASH_ADVANCE_TRX                     0
        PURCHASES_TRX                        0
        CREDIT_LIMIT                         1
        PAYMENTS                             0
        MINIMUM_PAYMENTS                   313
        PRC_FULL_PAYMENT                     0
        TENURE                               0
        dtype: int64
```

We can see that null values is present at minimum payments and credit limit in the data set.

So, null values are replaced by their column mean value using **fillna( )** method.

```
In [7]:  mean1=df['CREDIT_LIMIT'].mean()
         mean2=df['MINIMUM_PAYMENTS'].mean()
         df['CREDIT_LIMIT'].fillna(value=mean1, inplace=True)    # replacing null values with mean of a column
         df['MINIMUM_PAYMENTS'].fillna(value=mean2, inplace=True)

In [8]:  df.isnull().sum()

Out[8]: CUST_ID                              0
        BALANCE                              0
        BALANCE_FREQUENCY                    0
        PURCHASES                            0
        ONEOFF_PURCHASES                     0
        INSTALLMENTS_PURCHASES               0
        CASH_ADVANCE                         0
        PURCHASES_FREQUENCY                  0
        ONEOFF_PURCHASES_FREQUENCY           0
        PURCHASES_INSTALLMENTS_FREQUENCY     0
        CASH_ADVANCE_FREQUENCY               0
        CASH_ADVANCE_TRX                     0
        PURCHASES_TRX                        0
        CREDIT_LIMIT                         0
        PAYMENTS                             0
        MINIMUM_PAYMENTS                     0
        PRC_FULL_PAYMENT                     0
        TENURE                               0
        dtype: int64
```

We can see that there is no null values.

drop( ) method is used to remove few columns.

```
X = df.drop(['TENURE','CUST_ID'],axis=1).values    # preprocessing the data by removing the columns
y = df['TENURE'].values
```

I have imported PCA method to perform PCA on the data set.

Here we reduced the dimensionality of data into two components by keeping k value is equal to 2.

In [11]:

```python
# performing PCA
pca2 = PCA(n_components=2)
principalComponents = pca2.fit_transform(X)   # pca is applied on the data set without output labels
# creating a data frame for the pca results
principalDf = pd.DataFrame(data = principalComponents, columns = ['principal component 1', 'principal component 2'])
# adding a new column to the data frame
finalDf = pd.concat([principalDf, df[['TENURE']]], axis = 1)
finalDf   # printing the results
```

Out[11]:

	principal component 1	principal component 2	TENURE
0	-4326.383979	921.566882	12
1	4118.916665	-2432.846346	12
2	1497.907641	-1997.578694	12
3	1394.548536	-1488.743453	12
4	-3743.351896	757.342657	12
...	...	...	...
8945	-4208.357725	1122.443291	6
8946	-4123.923788	951.683820	6
8947	-4379.443989	911.504583	6
8948	-4791.117531	1032.540961	6
8949	-3623.702535	1555.134786	6

8950 rows × 3 columns

b. Apply k-means algorithm on the PCA result and report your observation if the silhouette score has improved or not?

To perform k-means algorithm on a data set first we need to find the number of clusters required to fit our data together into clusters by using elbow method.

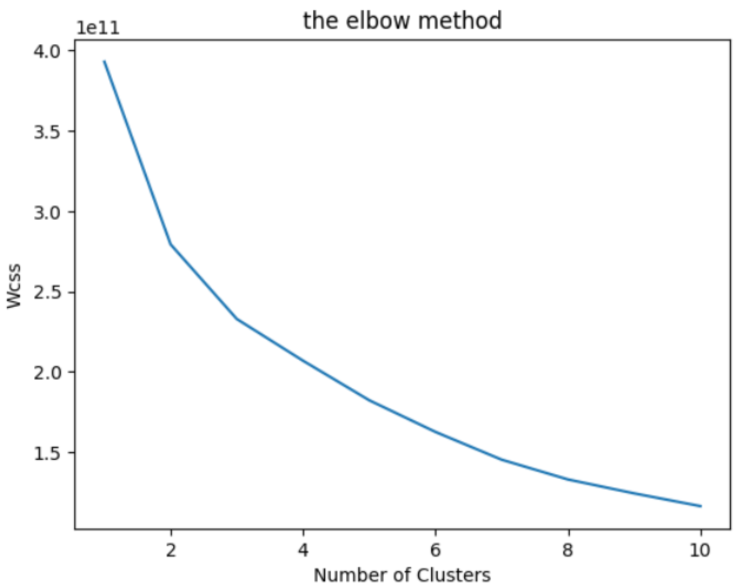
In [12]:

```python
# Use the elbow method to find a good number of clusters with the K-Means algorithm

from sklearn.cluster import KMeans
wcss = []
for i in range(1,11):
    kmeans = KMeans(n_clusters=i,init='k-means++',max_iter=300,n_init=10,random_state=0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

plt.plot(range(1,11),wcss)
plt.title('the elbow method')
plt.xlabel('Number of Clusters')
plt.ylabel('Wcss')
plt.show()
```

The elbow method results in a graph. From graph, the next point to point where the wcss value starts decreasing linearly will be the k value.



Using KMeans method of sklearn library, I applied K- Means algorithm on data set we got after performing PCA. After performing k-means on the PCA data we got a silhouette score of 57% which is higher than the silhouette score of raw data without performing PCA.

The silhouette score has been improved when we perform PCA on the data set. when we applied kmeans on the data set without performing PCA we got a silhouette score of 46.5%. After performing PCA we got a silhouette score of 57%. The silhouette score has been improved by more than 10%.

In [13]:
```
#  Calculate the silhouette score for the above clustering

nclusters = 3  # this is the k in kmeans
km = KMeans(n_clusters=nclusters)
km.fit(finalDf)       # fitting out kmeans model with our data set

y_cluster_kmeans = km.predict(finalDf)
from sklearn import metrics
score = metrics.silhouette_score(finalDf, y_cluster_kmeans)
print(score)
```

```
0.5720391530020279
```

c. Perform Scaling + PCA + K-Means and report performance.

Using StandardScalar method we performed feature scaling on the data set. Feature scaling is used to normalize the range of all features.

Now, we are performing PCA on the feature scaled data set using the PCA method.

In [14]:
```python
scaler = StandardScaler()      # feature scaling using standard scaler
X_Scale = scaler.fit_transform(X)
```

In [15]:
```python
# performing pca
pca3 = PCA(n_components=2)
principalComponents1 = pca3.fit_transform(X_Scale)

principalDf1 = pd.DataFrame(data = principalComponents1, columns = ['principal component 1', 'principal component 2'])

finalDf2 = pd.concat([principalDf1, df[['TENURE']]], axis = 1)
finalDf2
```

Out[15]:

	principal component 1	principal component 2	TENURE
0	-1.718893	-1.072939	12
1	-1.169303	2.509325	12
2	0.938414	-0.382600	12
3	-0.907502	0.045860	12
4	-1.637830	-0.684976	12
...	...	...	...
8945	-0.025276	-2.034126	6
8946	-0.233113	-1.656652	6
8947	-0.593879	-1.828114	6
8948	-2.007671	-0.673767	6
8949	-0.217931	-0.418489	6

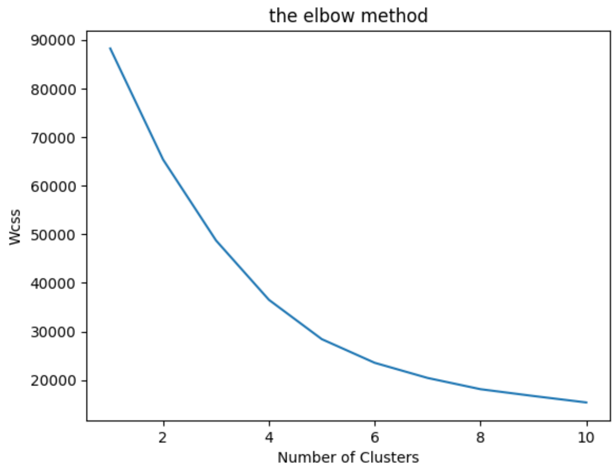
8950 rows × 3 columns

To perform k-means algorithm on a data set first we need to find the number of clusters required to fit our data together into clusters by using elbow method.

In [16]:
```python
# Use the elbow method to find a good number of clusters with the K-Means algorithm

from sklearn.cluster import KMeans
wcss = []
for i in range(1,11):
    kmeans = KMeans(n_clusters=i,init='k-means++',max_iter=300,n_init=10,random_state=0)
    kmeans.fit(finalDf2)
    wcss.append(kmeans.inertia_)

plt.plot(range(1,11),wcss)
plt.title('the elbow method')
plt.xlabel('Number of Clusters')
plt.ylabel('Wcss')
plt.show()
```



Using KMeans method of sklearn library, I applied K- Means algorithm by taking k value as 3 on data set, we got after performing feature scaling and PCA. After performing k-means on this data we got a silhouette score of 38%.

```
#  Calculate the silhouette score for the above clustering

nclusters = 3  # this is the k in kmeans
km = KMeans(n_clusters=nclusters)
km.fit(finalDf2)

y_cluster_kmeans = km.predict(finalDf2)
from sklearn import metrics
score = metrics.silhouette_score(finalDf2, y_cluster_kmeans)
print(score)
```

```
0.3837968579718024
```

2. Use pd_speech_features.csv

Using read_csv method imported a csv file. The head() method of pandas library results top most rows of a data set.

```
df1= pd.read_csv(r"C:\Users\supri\Desktop\pd_speech_features.csv")    # reading pd_speech_features csv file
df1.head()
```

	id	gender	PPE	DFA	RPDE	numPulses	numPeriodsPulses	meanPeriodPulses	stdDevPeriodPulses	locPctJitter	...	tqwt_kurtosisValue_dec_28
0	0	1	0.85247	0.71826	0.57227	240	239	0.008064	0.000087	0.00218	...	1.5620
1	0	1	0.76686	0.69481	0.53966	234	233	0.008258	0.000073	0.00195	...	1.5589
2	0	1	0.85083	0.67604	0.58982	232	231	0.008340	0.000060	0.00176	...	1.5643
3	1	0	0.41121	0.79672	0.59257	178	177	0.010858	0.000183	0.00419	...	3.7805
4	1	0	0.32790	0.79782	0.53028	236	235	0.008162	0.002669	0.00535	...	6.1727

5 rows × 755 columns

a. Perform Scaling

Using StandardScalar method we performed feature scaling on the data set. Feature scaling is used to normalize the range of all features.

```
scaler = StandardScaler()    #performing feature selection
X_Scale = scaler.fit_transform(X)
```

b. Apply PCA (k=3)

To run PCA on the data set, we imported the PCA method from the Sklearn Python package.

```python
# performing pca
pca4 = PCA(n_components=3)
principalComponents2 = pca4.fit_transform(X_Scale)

principalDf2 = pd.DataFrame(data = principalComponents2, columns = ['principal component 1', 'principal component 2',
                                                                     'principal components 3'])
finalDf3 = pd.concat([principalDf2, df1[['class']]], axis = 1)
finalDf3
```

	principal component 1	principal component 2	principal components 3	class
0	-10.047372	1.471076	-6.846403	1
1	-10.637725	1.583750	-6.830977	1
2	-13.516185	-1.253542	-6.818697	1
3	-9.155083	8.833600	15.290904	1
4	-6.764470	4.611465	15.637122	1
...	...	...	...	...
751	22.322682	6.481912	1.458753	0
752	13.442877	1.449414	9.352294	0
753	8.270264	2.391283	-0.908670	0
754	4.011761	5.412256	-0.847131	0
755	3.993114	6.072414	-2.020723	0

756 rows × 4 columns

c. Use SVM to report performance

sklearn module contains train_test_split method to split our data set into training and testing data sets.

Support vector machine algorithm is applied to the data set we got after performing PCA using sklearn module. We got an accuracy of 74.8% when we trained SVM on our data set.

```python
# training and predcting svm model on our data set
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# Support Vector Machine's
from sklearn.svm import SVC

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

y_pred = classifier.predict(X_test)

# Summary of the predictions made by the classifier
print(classification_report(y_true, y_pred))
print(confusion_matrix(y_true, y_pred))
# Accuracy score
from sklearn.metrics import accuracy_score
print('accuracy is',accuracy_score(y_pred,y_true))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        57
           1       0.75      1.00      0.86       170

    accuracy                           0.75       227
   macro avg       0.37      0.50      0.43       227
weighted avg       0.56      0.75      0.64       227

[[  0  57]
 [  0 170]]
accuracy is 0.748898678414097
```

3. Apply Linear Discriminant Analysis (LDA) on Iris.csv dataset to reduce dimensionality of data to k=2.

A csv file was imported using the read_csv method. The top rows of a data set are returned by the pandas library's head() method.

```
df2= pd.read_csv("C:\\Users\\supri\\Desktop\\Iris.csv")   # reading iris csv file
df2.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

Isnull( ) method of pandas library checks for any values present in data set.

```
df2.isnull().any()   # checking null values
```

```
Id               False
SepalLengthCm    False
SepalWidthCm     False
PetalLengthCm    False
PetalWidthCm     False
Species          False
dtype: bool
```

The Linear Discriminant Analysis of the sklearn.discriminant_analysis library can be used to Perform LDA in Python. By setting n_components value as 2 we will get the results in two linear discriminates. We execute the fit and transform methods to retrieve our results.

```
# performing lda on the data set
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
lda = LDA(n_components=2)
LinearDA = lda.fit_transform(X, y)
LinearDf = pd.DataFrame(data = LinearDA, columns = ['LD 1', 'LD 2'])   # converting our results into a dataset
finalLda = pd.concat([LinearDf, df2[['Species']]], axis = 1)   # appending species column to the data frame
finalLda
```

	LD 1	LD 2	Species
0	8.084953	0.328454	Iris-setosa
1	7.147163	-0.755473	Iris-setosa
2	7.511378	-0.238078	Iris-setosa
3	6.837676	-0.642885	Iris-setosa
4	8.157814	0.540639	Iris-setosa
...	...	...	...
145	-5.674013	1.661346	Iris-virginica
146	-5.197129	-0.365506	Iris-virginica
147	-4.981712	0.812973	Iris-virginica
148	-5.901486	2.320751	Iris-virginica
149	-4.684009	0.325081	Iris-virginica

150 rows × 3 columns

4. Briefly identify the difference between PCA and LDA

Principle Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are two main algorithms in dimensionality reduction.

PCA is an unsupervised while LDA is a supervised dimensionality reduction technique.

PCA gets the results without depending on the output labels. PCA results a data set with maximum variance between the features by ignoring the duplicates of other features. Since the variance between the features is independent of the outcome, PCA does not consider the output labels.

LDA depends on the output labels. Based on the output labels information LDA reduces the feature set dimensions and finds a decision boundary. The data points are then projected to new dimensions so that the clusters are as distinct from one another as possible, and the individual components of a cluster are as near the cluster centroid as possible