MACHINE LEARNING ASSIGNMENT 5

Name: Rupa Mallempati

Student id:700740339

Video link : https://drive.google.com/file/d/1TeO_e6pmHf5FdG9w43fdlRLXr-vrnhyh/view?usp=share_link

Github link : https://github.com/rupamallempati/ML_Assignment5.git

1. Principal Component Analysis

a. Apply PCA on CC dataset.

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

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

In [2]:
```
df= pd.read_csv(r"CC.csv")      # reading cc data set

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

Out[2]:

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQ
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	

In [3]:
```
df.shape
```

Out[3]: (8950, 18)

In [4]:
```python
df.isnull().sum()    #checking any null values are present
```

Out[4]:
```
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
```

In [5]:
```python
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 [6]:
```python
df.isnull().sum()
```

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                        0
PAYMENTS                            0
MINIMUM_PAYMENTS                    0
PRC_FULL_PAYMENT                    0
TENURE                              0
dtype: int64
```

In [7]:
```python
df['TENURE'].value_counts()
```

Out[7]:
```
12    7584
11     365
10     236
6      204
8      196
7      190
9      175
Name: TENURE, dtype: int64
```

In [8]:
```python
X = df.drop(['TENURE','CUST_ID'],axis=1).values   # preprocessing the data by removing the columns
y = df['TENURE'].values
```

In [9]:
```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[9]:

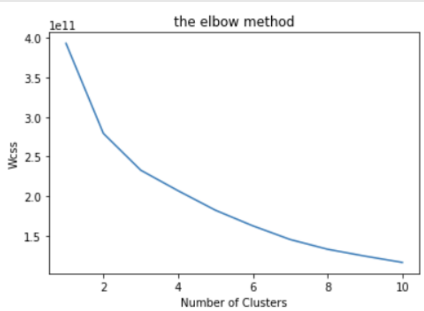
	principal component 1	principal component 2	TENURE
0	-4326.383979	921.566882	12
1	4118.916665	-2432.846346	12
2	1497.907641	-1997.578694	12
3	1394.548536	-1488.743453	12
4	-3743.351896	757.342657	12
...	...	...	...
8945	-4208.357725	1122.443291	6
8946	-4123.923788	951.683820	6
8947	-4379.443989	911.504583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

8950 rows × 3 columns

In [10]:
```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()
```



In [11]:
```python
#  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)
```

0.5720001633684872

In [13]:
```python
scaler = StandardScaler()      # feature scaling using standard scaler
X_Scale = scaler.fit_transform(X)
```

In [149]:
```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[149]:

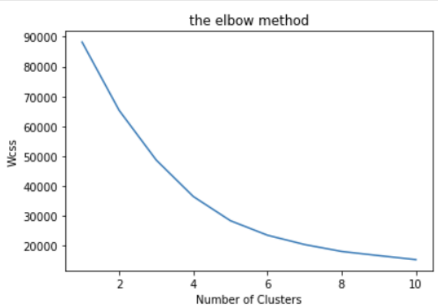
	principal component 1	principal component 2	TENURE
0	-1.718893	-1.072942	12
1	-1.169304	2.509342	12
2	0.938416	-0.382582	12
3	-0.907502	0.045862	12
4	-1.637830	-0.684980	12
...	...	...	...
8945	-0.025275	-2.034118	6
8946	-0.233112	-1.656646	6
8947	-0.593879	-1.828108	6
8948	-2.007672	-0.673770	6
8949	-0.217931	-0.418471	6

8950 rows × 3 columns

In [150]:
```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()
```



In [11]:
```python
#  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)
```

```
0.5720001633684872
```

2. Use pd_speech_features.csv

a. Perform Scaling

b. Apply PCA (k=3)

c. Use SVM to report performance.

In [152]:
```python
df1= pd.read_csv(r"pd_speech_features.csv")     # reading pd_speech_features csv file
df1.head()
```

Out[152]:

	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

In [153]:
```python
X = df1.drop('class',axis=1).values    # preprocessing the data
y = df1['class'].values
```

In [154]:
```python
scaler = StandardScaler()     #performing feature selection
X_Scale = scaler.fit_transform(X)
```

In [155]:
```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
```

Out[155]:

	principal component 1	principal component 2	principal components 3	class
0	-10.047372	1.471076	-6.846404	1
1	-10.637725	1.583748	-6.830977	1
2	-13.516185	-1.253542	-6.818698	1
3	-9.155084	8.833601	15.290902	1
4	-6.764470	4.611468	15.637120	1
...	...	...	...	...
751	22.322682	6.481910	1.458754	0
752	13.442877	1.449412	9.352295	0
753	8.270264	2.391286	-0.908671	0
754	4.011760	5.412257	-0.847132	0
755	3.993114	6.072417	-2.020724	0

756 rows × 4 columns

In [156]:
```python
# splitting our data into training and testing part
X_train, X_test, y_train, y_true = train_test_split(finalDf3[::-1], finalDf3['class'], test_size = 0.30, random_state = 0)
```

In [157]:
```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
```

Question 3

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

In [159]:
```python
df2= pd.read_csv("Iris.csv")   # reading iris csv file
df2.head()
```

Out[159]:

	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

In [160]:
```python
df2.isnull().any()   # checking null values
```

Out[160]:
```
Id               False
SepalLengthCm    False
SepalWidthCm     False
PetalLengthCm    False
PetalWidthCm     False
Species          False
dtype: bool
```

In [161]:
```
X = df2.iloc[:, 1:5].values   # preprocessing the data
y = df2.iloc[:, 5].values
```

In [162]:
```
# 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
```

Out[162]:

	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

Question 4:

4. Briefly identify the difference between PCA and LDA.

Answer:- Both PCA and LDA are linear transformation techniques. However, PCA is an unsupervised while LDA is a supervised dimensionality reduction technique.

Principal Component Analysis

PCA summarizes the feature set without relying on the output. PCA tries to find the directions of the maximum variance in the dataset. In a large feature set, there are many features that are merely duplicate of the other features or have a high correlation with the other features. Such features are basically redundant and can be ignored. The role of PCA is to find such highly correlated or duplicate features and to come up with a new feature set where there is minimum correlation between the features or in other words feature set with maximum variance between the features. Since the variance between the features doesn't depend upon the output, therefore PCA doesn't take the output labels into account.

Linear Discriminant Analysis

LDA tries to reduce dimensions of the feature set while retaining the information that discriminates output classes. LDA tries to find a decision boundary around each cluster of a class. It then projects the data points to new dimensions in a way that the clusters are as separate from each other as possible and the individual elements within a cluster are as close to the centroid of the cluster as possible. The new dimensions are ranked on the basis of their ability to maximize the distance between the clusters and minimize the distance between the data points within a cluster and their centroids. These new dimensions form the linear discriminants of the feature set.