MACHINE LEARNING ASSIGNMENT-5

GITHUB LINK:
https://github.com/spandanavegi/machine_learning.1/blob/main/Assignment5.ipynb
VIDEO LINK :
https://drive.google.com/drive/u/0/folders/1eVDf4XIM7LsCn_VhokeV35m_6t7JxiUR

Firstly we have imported all the required libraries
And then we imported the cc.csv dataset and to print its data frames we used dataset_CC.info().

```
dataset_CC = pd.read_csv('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
```

And then we used the head function to print the first n rows.

In [5]:
```
dataset_CC.head()
```

Out[5]:

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUEN
0	C10001	40.900749	0.818182	95.40	0.00	95.4	0.000000	0.166
1	C10002	3202.467416	0.909091	0.00	0.00	0.0	6442.945483	0.000
2	C10003	2495.148862	1.000000	773.17	773.17	0.0	0.000000	1.000
3	C10004	1666.670542	0.636364	1499.00	1499.00	0.0	205.788017	0.083
4	C10005	817.714335	1.000000	16.00	16.00	0.0	0.000000	0.083

Then we have used is null if any function. Which checks whether the given data has any null values or not. If they are any null values it prints true else false.

```
dataset_CC.isnull().any()
```

```
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
```

Then we are replacing the null value by the mean of the given data.

```
dataset_CC.fillna(dataset_CC.mean(), inplace=True)
dataset_CC.isnull().any()
```

```
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
```

Then we tried to print the shape of the dataset

```
x = dataset_CC.iloc[:,1:-1]
y = dataset_CC.iloc[:,1:-1]

print(x.shape,y.shape)
```

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

a)
Here to summarize the large dataset samples to smaller rows we used principal component analysis. To transform the data we have given different names to the columns. They are principal component 1, principal component 2, principal component 3, tenure. And then we have printed in the form of rows.

```
In [10]:
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[10]:

	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

b)
Here we have the cluster number as 3 and predicted the cluster for each data point and calculated the silhouette score.
And Silhouette 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.

```
           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]
 [   0    0    0    0    0    0    0    0    0    0]
 [ 175   28    1    0    0    0    0    0    0    0]
 [ 173   15    2    0    0    0    0    0    0    0]
 [ 169   27    0    0    0    0    0    0    0    0]
 [ 149   26    0    0    0    0    0    0    0    0]
 [ 189   46    1    0    0    0    0    0    0    0]
 [ 284   78    3    0    0    0    0    0    0    0]
 [5393 2066  125    0    0    0    0    0    0    0]]

Accuracy for our Training dataset with PCA: 0.0
Sihouette Score:  0.511279521159399
```

c) We have used standard scaler function and scaled the data and then applied principal component analysis. To transform the data we have given different names to the columns. They are principal component 1, principal component 2, principal component 3, tenure. And then we have printed in the form of rows.

```
#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()
```

	principal component 1	principal component 2	principal component 3	TENURE
0	-1.718894	-1.072940	0.535630	12
1	-1.169312	2.509318	0.627476	12
2	0.938419	-0.382598	0.161656	12
3	-0.907504	0.045858	1.521524	12
4	-1.637830	-0.684975	0.425714	12

2)

Here we have imported the pd_speech_features.csv and displayed the info

```python
dataset_pd = pd.read_csv('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
```

Then head function and check for null, then we have scaled the data using the scaler function and applied Principal Component Analysis with k=3.

```python
# 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','Princ

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

	principal component 1	principal component 2	Principal Component 3	class
0	-10.047372	1.471077	-6.846403	1
1	-10.637725	1.583749	-6.830977	1
2	-13.516185	-1.253542	-6.818698	1
3	-9.155084	8.833599	15.290899	1
4	-6.764470	4.611465	15.637116	1

Then we used the svm and displaced the silhouette Score.

```
# 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:  0.25044637751380994
```

3)

Here we have imported the dataset and displayed the info and checked for the null values. Then displayed the shape.

Then to apply linear discriminant value we have used x_train,y-train,x-test,y_test and splitted the data into the test size of 0.3

Then applied scaer function

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
le = LabelEncoder()
y = le.fit_transform(y)
```

Then we imported linear discriminant analysis and checked the shape.

```
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)
```

4) Briefly identify the difference between PCA and LDA

Both LDA and PCA rely on linear transformations and aim to maximize the variance in a lower dimension. PCA is an unsupervised learning algorithm while LDA is a supervised learning algorithm. This means that PCA finds directions of maximum variance regardless of class labels while LDA finds directions of maximum class separability
It reduces the features into a smaller subset of orthogonal variables, called principal components – linear combinations of the original variables. The first component captures the largest variability of the data, while the second captures the second largest, and so on.
LDA finds the linear discriminants in order to maximize the variance between the different categories while minimizing the variance within the class.