Assignment-5

Name: Sammeta Jaya Sai Charan
Student ID: 700739775

Github link: https://github.com/saicharan255/Assignment5.git
Video link: https://drive.google.com/file/d/1q2rj-ejiSnqOe6WP17Gy-EiiNlbS0OZL/view?usp=sharing

1. Principal Component Analysis

```python
# importing 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")
```

```python
# (1)Principal Component Analysis
dst_CC = pd.read_csv('CC GENERAL.csv')
dst_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
```

Firsty we have imported the required libraries and also imported the dataset CC GENERAL.csv, by using info() it shows the informationof the dataset.

```
dst_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
```

```
[6] dst_CC.fillna(dst_CC.mean(), inplace=True)
    dst_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
```

We next checked the dataset for null values, if they have null values or true for null values then it replaces with NA because we are using fillna().

a. Apply PCA on CC dataset

```
#(1a) Apply PCA on CC Dataset
pca = PCA(3)
a_pca = pca.fit_transform(a)
prplDf = pd.DataFrame(data = a_pca, columns = ['principal cmpnt 1', 'principal cmpnt 2', 'principal cmpnt 3'])
finalDf = pd.concat([prplDf, dst_CC.iloc[:,-1]], axis = 1)
finalDf.head()
```

	principal cmpnt 1	principal cmpnt 2	principal cmpnt 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

Now we are going to apply Principal component analysis on the CC dataset, we are using fit transform() and getting principal dataframe with principal component 1,2,3 and we get final dataframe is displayed by using head().

b. Apply k-means algorithm on the PCA result

```python
#1b) Apply K Means on PCA Result
A = finalDf.iloc[:,0:-1]
b = finalDf.iloc[:,-1]
nclusters = 3 # this is the k in kmeans
km = KMeans(n_clusters=nclusters)
km.fit(A)

# predict the cluster for each data point
b_cluster_kmeans = km.predict(A)


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


train_acc = accuracy_score(b, b_cluster_kmeans)
print("\nAccuracy of our training dataset with PCA:", train_acc)


#Calculate sihouette Score
scr = metrics.silhouette_score(A, b_cluster_kmeans)
print("Sihouette Score: ",scr)
```

```
              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]
 [   0    0    0    0    0    0    0    0    0    0]
 [ 175    1   28    0    0    0    0    0    0    0]
 [ 173    2   15    0    0    0    0    0    0    0]
 [ 169    0   27    0    0    0    0    0    0    0]
 [ 149    0   26    0    0    0    0    0    0    0]
 [ 188    1   47    0    0    0    0    0    0    0]
 [ 284    3   78    0    0    0    0    0    0    0]
 [5389  126 2069    0    0    0    0    0    0    0]]

Accuracy of our training dataset with PCA: 0.0
Sihouette Score:  0.5109307274319468
```

In this step we apply k means algorithm on the obtained final data frame here we take k as 3 clusters and predicted the clusters for each data point. We then calculated the silhouette score using metrics.silhouette_score().

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

```
#Scaling
scaler = StandardScaler()
scaler.fit(a)
X_scaled_array = scaler.transform(a)
#PCA
pca = PCA(3)
x_pca = pca.fit_transform(X_scaled_array)
prplDf = pd.DataFrame(data = x_pca, columns = ['principal cmpt 1', 'principal cmpt 2','principal cmpt 3'])
finalDf = pd.concat([prplDf, dst_CC.iloc[:,-1]], axis = 1)
finalDf.head()
```

	principal cmpt 1	principal cmpt 2	principal cmpt 3	TENURE
0	-1.718893	-1.072939	0.535710	12
1	-1.169308	2.509310	0.627778	12
2	0.938415	-0.382596	0.161292	12
3	-0.907503	0.045856	1.521631	12
4	-1.637829	-0.684972	0.425753	12

In the scaling we used StandardScaler() and scaler.transform(A) for the dataframe a. In PCA we used pca.fit_transform and got the principal dataframe with principal component 1,2,3 and we get final dataframe is displayed by using head().

```
A_train, A_test, b_train, b_test = train_test_split(A,b, test_size=0.34,random_state=0)
nclusters = 3
# this is the k in kmeans
km = KMeans(n_clusters=nclusters)
km.fit(A_train,b_train)


# predict the cluster for each training data point
b_clust_train = km.predict(A_train)

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

train_acc = accuracy_score(b_train, b_clust_train)
print("Accuracy of our training dataset with PCA:", train_acc)

#Calculate sihouette Score
scr = metrics.silhouette_score(A_train, b_clust_train)
print("Sihouette Score: ",scr)
```

```
              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    4   30    0    0    0    0    0    0    0]
 [ 108    1   26    0    0    0    0    0    0    0]
 [  96    4   28    0    0    0    0    0    0    0]
 [  89    2   27    0    0    0    0    0    0    0]
 [ 107    6   38    0    0    0    0    0    0    0]
 [ 185   11   66    0    0    0    0    0    0    0]
 [3393  739  842    0    0    0    0    0    0    0]]
Accuracy of our training dataset with PCA: 0.0
Sihouette Score:  0.38120810623375717
```

We applied k means algorithm by taking k=3 clusters, predicted the each cluster from each training data point then the predictions were made by the classifier. We calculated the silhouette score using metrics.silhouette_score().

2. Use pd_speech_features.csv

```
# 2)Use pd_speech_features.csv
dst_pd = pd.read_csv('pd_speech_features.csv')
dst_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
```

```
dst_pd.head()
```

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

5 rows × 755 columns

Firstly we have imported the required pd_speech_features.csv dataset by using pd.read_csv().

a. Perform Scaling

```
#a)Scaling Data
scaler = StandardScaler()
X_Scale = scaler.fit_transform(X)
```

We have done scaling using StandardScalar().

b. Apply PCA (k=3)

```
#b) Apply PCA with k =3
pca3 = PCA(n_components=3)
principalComponents = pca3.fit_transform(X_Scale)

prplDf = pd.DataFrame(data = principalComponents, columns = ['principal cmpnt 1', 'principal cmpnt 2','Principal cmpnt 3'])

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

	principal cmpnt 1	principal cmpnt 2	Principal cmpnt 3	class
0	-10.047372	1.471078	-6.846402	1
1	-10.637725	1.583750	-6.830976	1
2	-13.516185	-1.253541	-6.818696	1
3	-9.155083	8.833600	15.290899	1
4	-6.764469	4.611465	15.637116	1

we applied the PCA with k=3, used pca.fit_transform() and got the principal dataframe with principal component 1,2,3 and we get final dataframe is displayed by using head().

c. Use SVM to report performance

```
#c) Support Vector Machine(SVM)

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
scr= metrics.silhouette_score(X_test, y_pred)
print("Sihouette Score: ",scr)
```

```
              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:  0.2504463884705572
```

In the above we used support vector machine to obtain the performance svmClassifier.predict(). Predictions were made by the classifier and accuracy is calculated. Finally we calculated the silhouette score and got 25%.

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

```
#3) Apply Linear Discriminant Analysis (LDA) on Iris.csv dataset to reduce dimensionality of data to k=2.
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
dst_iris = pd.read_csv('Iris.csv')
dst_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
```

We imported the required libraries linear discriminant analysis, loaded Iris.csv dataset from local drive.

```
dst_iris.isnull().any()
```

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

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

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

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

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

```
[ ] 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)
```

Firstly we have checked for null values, located the rows and columns, we used standardScalar() to perform the operation and trained, tested the dataset. Now we applied linear discriminant analysis LDA algorithm trained and tested the dataset and printed the result.

4. Briefly identify the difference between PCA and LDA

Both LDA and PCA are based on linear transformations and aim to maximize the variance of the lower dimension. PCA is an unsupervised learning algorithm while LDA is a supervised learning algorithm.That is, PCA finds indications of maximum variance regardless of class labels, while LDA finds indications of maximum class separation