Machine Learning - Assignment 5

Name: Idavalapati Vijay Taraka Ramarao
ID: 700742485
CRN: 13428

Question1

Principal Component Analysis. Apply PCA on CC dataset. Apply 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.

```python
# Question1
dataset_CC = pd.read_csv('datasets//CC.csv')
dataset_CC.info()


print(dataset_CC.head())

print(dataset_CC.isnull().any())

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

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

# 1.a Apply PCA on CC Dataset
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)
print(finalDf.head())

# 1.b Apply K Means on PCA Result
X = finalDf.iloc[:,0:-1]
y = finalDf.iloc[:,-1]
```

```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)


#1.c Scaling +PCA + KMeans
x = dataset_CC.iloc[:,1:-1]
y = dataset_CC.iloc[:,-1]
print(x.shape,y.shape)
```

```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)
print(finalDf.head())


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

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))
```

```python
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)

# 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)
```

Output:

```
C:\Users\Administrator\Documents\GitHub\ML\venv\Scripts\python.exe C:/Users/Administrator/Documents/GitHub/ML/Assignment5/Assignment5_Question_1.py
<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
```

```
  CUST_ID      BALANCE  ...  PRC_FULL_PAYMENT  TENURE
0  C10001    40.900749  ...          0.000000      12
1  C10002  3202.467416  ...          0.222222      12
2  C10003  2495.148862  ...          0.000000      12
3  C10004  1666.670542  ...          0.000000      12
4  C10005   817.714335  ...          0.000000      12

[5 rows x 18 columns]
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
```

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

```
              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   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]
 [ 188   47    1    0    0    0    0    0    0    0]
 [ 284   78    3    0    0    0    0    0    0    0]
 [5389 2069  126    0    0    0    0    0    0    0]]
```

```
Accuracy for our Training dataset with PCA: 0.0
Sihouette Score:  0.5109307274319468
(8950, 16) (8950,)
   principal component 1  principal component 2  principal component 3  TENURE
0              -1.718893              -1.072940              0.535688      12
1              -1.169305               2.509320              0.628047      12
2               0.938414              -0.382600              0.161237      12
3              -0.907502               0.045859              1.521706      12
4              -1.637830              -0.684975              0.425675      12
(8950, 3) (8950,)
              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]
 [  30    4  105    0    0    0    0    0    0    0]
 [  26    1  108    0    0    0    0    0    0    0]
 [  28    4   96    0    0    0    0    0    0    0]
 [  27    2   89    0    0    0    0    0    0    0]
 [  38    6  107    0    0    0    0    0    0    0]
 [  66   11  185    0    0    0    0    0    0    0]
 [ 842  735 3397    0    0    0    0    0    0    0]]
Accuracy for our Training dataset with PCA: 0.0
Sihouette Score:  0.38140411892789516
              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]
 [  21    3   41    0    0    0    0    0    0    0]
 [  12    0   43    0    0    0    0    0    0    0]
 [  10    1   57    0    0    0    0    0    0    0]
 [  22    0   35    0    0    0    0    0    0    0]
 [  17    5   63    0    0    0    0    0    0    0]
 [  30    4   69    0    0    0    0    0    0    0]
 [ 450  395 1765    0    0    0    0    0    0    0]]

Accuracy for our Training dataset with PCA: 0.0
Sihouette Score:  0.38364273650815006
```

<u>Question2:</u>
Use pd_speech_features.csv. Perform Scaling. Apply PCA (k=3). Use SVM to report performance

```python
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
sns.set(style="white", color_codes=True)
import warnings
warnings.filterwarnings("ignore")

dataset_pd = pd.read_csv('datasets//pd_speech_features.csv')
dataset_pd.info()

print(dataset_pd.head())

print(dataset_pd.isnull().any())

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

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

# 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 Component 3'])
```

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

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)

# 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)
```

Output:

```
C:\Users\Administrator\Documents\GitHub\ML\venv\Scripts\python.exe C:/Users/Administrator/Documents/GitHub/ML/Assignment5/Assignment5_Question_2
<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
   id  gender  ...  tqwt_kurtosisValue_dec_36  class
0   0       1  ...                    18.9405      1
1   0       1  ...                    45.1780      1
2   0       1  ...                     4.7666      1
3   1       0  ...                     4.0603      1
4   1       0  ...                     6.1164      1

[5 rows x 755 columns]
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
```

```
   principal component 1  principal component 2  Principal Component 3  class
0             -10.047372               1.471076             -6.846403      1
1             -10.637725               1.583748             -6.830976      1
2             -13.516185              -1.253542             -6.818697      1
3              -9.155083               8.833597             15.290907      1
4              -6.764470               4.611464             15.637124      1
              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:  0.25044638057045615

Process finished with exit code 0
```

Question3:

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

```
dataset_iris = pd.read_csv('datasets//Iris.csv')
dataset_iris.info()

print(dataset_iris.isnull().any())

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

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)

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)
```

Output:

```
<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
Id               False
SepalLengthCm    False
SepalWidthCm     False
PetalLengthCm    False
PetalWidthCm     False
Species          False
dtype: bool
(150, 4) (150,)
(105, 2) (45, 2)

Process finished with exit code 0
```

Question4:
Briefly identify the difference between PCA and LDA

Answer:
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.

DA finds the linear discriminants in order to maximize the variance between the different categories while minimizing the variance within the class.

Related Links:

SourceCode:

https://github.com/VijayTarakaRamarao/ML/tree/main/Assignment5

Recording:

https://github.com/VijayTarakaRamarao/ML/blob/main/Assignment4/MachineLearning_Assignment5.mp4