Machine Learning Assignment 5

Name: Akhila Boddu

Student ID: 700742171

GitHub link: https://github.com/AkhilaBoddu/ML-Assignment-5.git

Video Link:
https://drive.google.com/file/d/1C2pMczVfhQ05On2QEVYYD4ZUTOfcB__K/view?usp=share_link

Clustering & Dimensionality reduction

Question1

Principal Component Analysis

a. Apply PCA on CC dataset.

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

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

To do data analysis and apply machine learning algorithms on data, first I imported a few python libraries.

```
# importing required libraries for assignment 5 here
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")
```

```
# Principal Component Analysis
# a. Apply PCA on CC dataset.
# b. Apply k-means algorithm on the PCA result and report your observation if the silhouette score
# has improved or not?
# c. Perform Scaling+PCA+K-Means and report performance.
```

Using read_csv method imported “CC GENERAL.csv“ dataset.

```
dataset_CC = pd.read_csv('CC GENERAL.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
dtypes: float64(14), int64(3), object(1)
memory usage: 1.2+ MB
```

The head() method of pandas library results top most rows of a data set.

```
dataset_CC.head()
```

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY
0	C10001	40.900749	0.818182	95.40	0.00	95.4	0.000000	0.166667
1	C10002	3202.467416	0.909091	0.00	0.00	0.0	6442.945483	0.000000
2	C10003	2495.148862	1.000000	773.17	773.17	0.0	0.000000	1.000000
3	C10004	1666.670542	0.636364	1499.00	1499.00	0.0	205.788017	0.083333
4	C10005	817.714335	1.000000	16.00	16.00	0.0	0.000000	0.083333

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

```
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
```

In this data set there are a few null values present in minimum payments and credit limit columns. So, these null values are replaced with their column mean value using fillna() method.

```
[12] 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
```

Next we extract the input features and output labels from pandas dataframe and we print the shapes, here x is input features and y is output labels.

a. Apply PCA on CC dataset

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

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

```
[14] #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)
finalDf.head()
```

	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. **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 value i.e., k value.

```
[48]
#1.b Apply K Means on PCA Result
X = finalDf.iloc[:,0:-1]
y = finalDf.iloc[:,-1]
```

```
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)

"""
Sihouette Score- ranges from -1 to +1 , a high value indicates that the object is well matched to its own cluster and poorly matc
```

```
              precision    recall  f1-score   support

           0       0.51      0.53      0.52       192
           1       0.76      0.41      0.53       564
           2       0.00      1.00      0.00         0

    accuracy                           0.44       756
   macro avg       0.42      0.65      0.35       756
weighted avg       0.70      0.44      0.53       756

[[102  74  16]
 [ 98 232 234]
 [  0   0   0]]

Accuracy for our Training dataset with PCA: 0.4417989417989418
Sihouette Score:  0.30375402761352255
'\nSihouette Score- ranges from -1 to +1 , a high value indicates that the object is well matched to its own cluster and poorly m
atched to neighboring clusters.\n'
```

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

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

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

Here we perform Scaling and PCA

```
#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	-1.718893	-1.072941	0.535701	12
1	-1.169306	2.509323	0.627974	12
2	0.938414	-0.382601	0.161226	12
3	-0.907503	0.045859	1.521691	12
4	-1.637830	-0.684977	0.425702	12

Here we extract the features and target variable from finalDf dataframe and X contains all columns of dataframe except last one, y contains values from the last column

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

(8950, 3) (8950,)
```

Here we perform k-means

```
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))

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)

"""
Sihouette Score- ranges from -1 to +1 , a high value indicates that the object is well matched to its own cluster and poorly matc
"""
```

```
              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   30    4    0    0    0    0    0    0    0]
 [ 108   26    1    0    0    0    0    0    0    0]
 [  96   28    4    0    0    0    0    0    0    0]
 [  89   27    2    0    0    0    0    0    0    0]
 [ 107   38    6    0    0    0    0    0    0    0]
 [ 185   66   11    0    0    0    0    0    0    0]
 [3397  842  735    0    0    0    0    0    0    0]]
Accuracy for our Training dataset with PCA: 0.0
Sihouette Score:  0.38140434191330214
'\nSihouette 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 c
lusters.\n'
```

```python
# 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)

"""
Sihouette Score- ranges from -1 to +1 , a high value indicates that the object is well matched to its own clust
"""
```

```
              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]
 [  41   21    3    0    0    0    0    0    0    0]
 [  43   12    0    0    0    0    0    0    0    0]
 [  57   10    1    0    0    0    0    0    0    0]
 [  35   22    0    0    0    0    0    0    0    0]
 [  63   17    5    0    0    0    0    0    0    0]
 [  69   30    4    0    0    0    0    0    0    0]
 [1765  450  395    0    0    0    0    0    0    0]]

Accuracy for our Training dataset with PCA: 0.0
Sihouette Score:  0.3836430453216196
'\nSihouette 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 c
lusters.\n'
```

Question2

Use pd_speech_features.csv

a. Perform Scaling

b. Apply PCA (k=3)

c. Use SVM to report performance

```
[75] # Use pd_speech_features.csv
     # a. Perform Scaling
     # b. Apply PCA (k=3)
     # c. Use SVM to report performance
```

```
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
```

```
dataset_pd.head()
```

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

5 rows × 755 columns

```
dataset_pd.isnull().any()
```

```
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
class                      False
Length: 755, dtype: bool
```

a. Perform Scaling

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

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

b. Apply PCA (k=3)

```
# 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 componen

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	-6.846405	1
1	-10.637725	1.583750	-6.830976	1
2	-13.516185	-1.253541	-6.818699	1
3	-9.155084	8.833602	15.290902	1
4	-6.764470	4.611467	15.637120	1

```
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)
```

c. Use SVM to report performance

```
#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)
```

```
              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:  0.25044638820643456
```

Question3

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

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

```python
dataset_iris.isnull().any()
```

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

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

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

```
[90] 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)
```

Question4

Briefly identify the difference between PCA and LDA

```
[93] #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 :
```

```
'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 dire
ctions of maximum variance regardless of class labels while LDA finds directions of maximum class separability'
```

```
#PCA
"""It reduces the features into a smaller subset of orthogonal variables, called principal components – linear
```

```
'It reduces the features into a smaller subset of orthogonal variables, called principal components – linear co
mbinations of the original variables. The first component captures the largest variability of the data, while t
he second captures the second largest, and so on.'
```

```
#LDA
"""LDA finds the linear discriminants in order to maximize the variance between the different categories while
```

```
'LDA finds the linear discriminants in order to maximize the variance between the different categories while mi
nimizing the variance within the class.'
```