MACHINE LEARNING
ASSIGNMENT - 5

Name :Thandarupalli Naveen Goud
UCM ID :nxt46830
700# :700734683

Video Link:https://drive.google.com/file/d/12jD6GhkvIeV9HxbDZ40_DN5nNpiBidT4/view?usp=share_link

Github Link:https://github.com/nxt46830/ML-Assignment-1

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 silhouette score has improved or not?
c. Perform Scaling+PCA+K-Means and report performance.

```
# 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_matr
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
# has improved or not?
# c. Perform Scaling+PCA+K-Means and report performance.
```

```python
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
dtypes: float64(14), int64(3), object(1)
```

```
dataset_CC.head()
```

	CUST_ID object	BALANCE float64	BALANCE_FREQ...	PURCHASES floa...	ONEOFF_PURC
0	C10001	40.900749	0.818182	95.4	
1	C10002	3202.467416	0.909091	0.0	
2	C10003	2495.148862	1.0	773.17	7
3	C10004	1666.670542	0.636364	1499.0	1.
4	C10005	817.714335	1.0	16.0	

5 rows, showing 10 per page   Page 1 of 1

```
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
```

```
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
```

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

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

[7]

```
#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', 'pri
finalDf = pd.concat([principalDf, dataset_CC.iloc[:,-1]], axis = 1)
finalDf.head()
```

Visualize

Visualize

	principal compo...	principal compo...	principal compo...	TENURE int64
0	-4326.383978558351	921.5668815814222	183.70838347385111	12
1	4118.916664523621	-2432.8463459904533	2369.9692893604197	12
2	1497.9076407403047	-1997.5786942158522	-2125.631327723396	12
3	1394.5485361338856	-1488.7434528532237	-2431.7996490218034	12
4	-3743.351895614359	757.3426565700995	512.4764917625604	12

5 rows, showing 10 per page Page 1 of 1

```
#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
"""
```

```
              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]
 [  28  175    1    0    0    0    0    0    0    0]
 [  15  173    2    0    0    0    0    0    0    0]
 [  27  169    0    0    0    0    0    0    0    0]
 [  26  149    0    0    0    0    0    0    0    0]
 [  47  188    1    0    0    0    0    0    0    0]
 [  78  284    3    0    0    0    0    0    0    0]
 [2068 5390  126    0    0    0    0    0    0    0]]

Accuracy for our Training dataset with PCA: 0.0
Sihouette Score:  0.5109769750121257
```

```
'\nSihouette Score- ranges from -1 to +1 , a high value indicates that the object is w
```

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

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

```
#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', 'pri
finalDf = pd.concat([principalDf, dataset_CC.iloc[:,-1]], axis = 1)
finalDf.head()
```

Visualize

Visualize

	principal compo...	principal compo...	principal compo...	TENURE int64
0	-1.718892260161892	-1.0729401874379387	0.5356228915887592	12
1	-1.1693075799553465	2.5093219077357274	0.6281801074535476	12
2	0.9384162721923274	-0.3826026169765072	0.16089982444446763	12
3	-0.9075028801689199	0.045859536408029006	1.5217473664737184	12
4	-1.6378289692772323	-0.6849764077245449	0.42555813137208226	12

5 rows, showing 10 per page Page 1 of 1

	principal compo...	principal compo...	principal compo...	TENURE int64
0	-1.718892260161892	-1.0729401874379387	0.5356228915887592	12
1	-1.1693075799553465	2.5093219077357274	0.6281801074535476	12
2	0.9384162721923274	-0.3826026169765072	0.16089982444446763	12
3	-0.9075028801689199	0.045859536408029006	1.5217473664737184	12
4	-1.6378289692772323	-0.6849764077245449	0.42555813137208226	12

5 rows, showing 10 per page Page 1 of 1

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

[13]

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

[14]

```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.34,random_st
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
"""
```

```
              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]
 [ 184   11   67    0    0    0    0    0    0    0]
 [3408  723  843    0    0    0    0    0    0    0]]
Accuracy for our Training dataset with PCA: 0.0
Sihouette Score:  0.3816397521049892
```

```
'\nSihouette Score- ranges from -1 to +1 , a high value indicates that the object is w
```

[15]

```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
"""
```

	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    3   21    0    0    0    0    0    0    0]
 [  43    0   12    0    0    0    0    0    0    0]
 [  57    1   10    0    0    0    0    0    0    0]
 [  35    0   22    0    0    0    0    0    0    0]
 [  63    5   17    0    0    0    0    0    0    0]
 [  68    4   31    0    0    0    0    0    0    0]
 [1775  382  453    0    0    0    0    0    0    0]]

Accuracy for our Training dataset with PCA: 0.0
Sihouette Score:  0.38425040003352195
```

```
'\nSihouette Score- ranges from -1 to +1 , a high value indicates that the object is w
```

[16]

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

[17]

```
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
```

[18]

```
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
```

[19]

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

[20]

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

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

principalDf = pd.DataFrame(data = principalComponents, columns = ['principal comp

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

	principal compo...	principal compo...	Principal Compo...	class int64
0	-10.0473720802064	1.4710755343073976	-6.8464066262176795	1
1	-10.637725134198256	1.583748747034128	-6.830979740294065	1
2	-13.516185233777032	-1.2535425615948577	-6.818698677650746	1
3	-9.155083698658137	8.83359870249697	15.290894521409589	1
4	-6.764469799613738	4.611466268338349	15.63711379027253	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_st
```

[23]

```
#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:  0.25044636702010714
```

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

```
[24]
#3.Apply Linear Discriminant Analysis (LDA) on Iris.csv dataset to reduce dimensi
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
```

```
[25]
dataset_iris.isnull().any()
```

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

```
[26]
x = dataset_iris.iloc[:,1:-1]
y = dataset_iris.iloc[:,-1]
print(x.shape,y.shape)
```

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

```
[27]
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_s
```

```
[28]
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
le = LabelEncoder()
y = le.fit_transform(y)
```

```
[29]
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

```
[30]
#4. Briefly identify the difference between PCA and LDA

"""Both LDA and PCA rely on linear transformations and aim to maximize the varian
```

```
'Both LDA and PCA rely on linear transformations and aim to maximize the variance in a
```

```
[31]
captures the largest variability of the data, while the second captures the secon
```

```
'It reduces the features into a smaller subset of orthogonal variables, called princip
```

```
[32]
maximize the variance between the different categories while minimizing the varia
```

```
'LDA finds the linear discriminants in order to maximize the variance between the diff
```

Principal Component Analysis:

1.PCA is a technique in unsupervised machine learning that is used to minimize dimensionality. The key idea of the vital component analysis (PCA) is to minimize the dimensionality of a data set consisting of several variables, either firmly or lightly, associated with each other while preserving to the maximum degree the variance present in the dataset.

2.This is achieved by translating the variables into a new collection of variables that are a mixture of our original dataset's variables or attributes so that maximum variance is preserved.

Linear Discriminant Analysis (LDA):

1.LDA is a technique of supervised machine learning. The critical principle of linear discriminant analysis (LDA) is to optimize the separability between the two classes to identify them in the best way we can determine. LDA is similar to PCA, which helps minimize dimensionality. Still, by constructing a new linear axis and projecting the data points on that axis, it optimizes the separability between established categories.

2.LDA does not function on finding the primary variable; it merely looks at what kind of point/features/subspace to distinguish the data offers further discrimination.

Conclusion:

The PCA and LDA are implemented in dimensionality reduction, which means a linear relationship between input and output variables. But it is possible to apply the PCA and LDA together and see the difference in their outcome. While PCA and LDA work on linear issues, they do have differences.