# Machine Learning Assignment -5


| | | |
|---|---|---|
| **Name** | **:** | **Sachit Kumar Tadishetty** |
| **700#** | **:** | **700734682** |
| **UCM ID** | **:** | **SXT46820** |


**VideoLink:**
**https://drive.google.com/drive/folders/1kj57lrUnTj2skKkZDH7UgScC-2lM5PB3?usp=share_link**

**Github Link :https://github.com/sachit46820/ML-Assignment**

**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.

STUDENT / ML Assignment 5 (SACHIT) / Notebook 1

Ready

Run notebook

```
# 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.
```

Ready

Run notebook

```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)
memory usage: 1.2+ MB
```

STUDENT / ML Assignment 5 (SACHIT) / Notebook 1

Ready

Run notebook

```python
dataset_CC.head()
```

Visualize

| | CUST_ID object | BALANCE float64 | BALANCE_FREQ... | PURCHASES floa... | ONEOFF_PURC... | INSTALLMENTS... | CASH_AI |
|---|---|---|---|---|---|---|---|
| 0 | C10001 | 40.900749 | 0.818182 | 95.4 | 0.0 | 95.4 | |
| 1 | C10002 | 3202.467416 | 0.909091 | 0.0 | 0.0 | 0.0 | 64 |
| 2 | C10003 | 2495.148862 | 1.0 | 773.17 | 773.17 | 0.0 | |
| 3 | C10004 | 1666.670542 | 0.636364 | 1499.0 | 1499.0 | 0.0 | 2 |
| 4 | C10005 | 817.714335 | 1.0 | 16.0 | 16.0 | 0.0 | |

5 rows, showing 10 per page Page 1 of 1

STUDENT / ML Assignment 5 (SACHIT) / Notebook 1

Ready

Run notebook

```
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
```

STUDENT / ML Assignment 5 (SACHIT) / Notebook 1

Ready

Run notebook

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

Visualize

| | principal compo... | principal compo... | principal compo... | TENURE int64 |
|---|---|---|---|---|
| 0 | -4326.383978558221 | 921.5668815814566 | 183.7083834739683 | 12 |
| 1 | 4118.916664523624 | -2432.846345990417 | 2369.9692893604206 | 12 |
| 2 | 1497.9076407403038 | -1997.5786942158497 | -2125.6313277233744 | 12 |
| 3 | 1394.5485361338847 | -1488.7434528532224 | -2431.799649021798 | 12 |
| 4 | -3743.351895614361 | 757.3426565700987 | 512.4764917625602 | 12 |

5 rows, showing 10 per page — Page 1 of 1

Ready

Run notebook

```
#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 clus
"""
```

STUDENT / ML Assignment 5 (SACHIT) / Notebook 1

Ready

Run notebook

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

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

```
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 clus
"""
```

Ready

Run notebook

```
              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]
 [   0  123   16    0    0    0    0    0    0    0]
 [   0  126    9    0    0    0    0    0    0    0]
 [   0  110   18    0    0    0    0    0    0    0]
 [   0  106   12    0    0    0    0    0    0    0]
 [   1  121   29    0    0    0    0    0    0    0]
 [   3  211   48    0    0    0    0    0    0    0]
 [  62 3605 1307    0    0    0    0    0    0    0]]
Accuracy for our Training dataset with PCA: 0.0
Sihouette Score:  0.5216744364662849
```

```
'\nSihouette Score- ranges from -1 to +1 , a high value indicates that the object is well matched to its own cluster
```



Ready

Run notebook

```
# 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 clus
"""
```

STUDENT / ML Assignment 5 (SACHIT) / Notebook 1

Ready

Run notebook

```
              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]
 [   0   53   12    0    0    0    0    0    0    0]
 [   1   47    7    0    0    0    0    0    0    0]
 [   0   61    7    0    0    0    0    0    0    0]
 [   0   45   12    0    0    0    0    0    0    0]
 [   0   68   17    0    0    0    0    0    0    0]
 [   0   74   29    0    0    0    0    0    0    0]
 [  37 1879  694    0    0    0    0    0    0    0]]

Accuracy for our Training dataset with PCA: 0.0
Sihouette Score:  0.5100449776852223
```

```
'\nSihouette Score- ranges from -1 to +1 , a high value indicates that the object is well matched to its own cluster
```

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

STUDENT / ML Assignment 5 (SACHIT) / Notebook 1

Ready

Run notebook

```
# 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
```

[17]

```
dataset_pd.head()
```

Visualize

| | id int64 | gender int64 | PPE float64 | DFA float64 | RPDE float64 | numPulses int64 | numPeric |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0.85247 | 0.71826 | 0.57227 | 240 | |
| 1 | 0 | 1 | 0.76686 | 0.69481 | 0.53966 | 234 | |
| 2 | 0 | 1 | 0.85083 | 0.67604 | 0.58982 | 232 | |
| 3 | 1 | 0 | 0.41121 | 0.79672 | 0.59257 | 178 | |
| 4 | 1 | 0 | 0.3279 | 0.79782 | 0.53028 | 236 | |

5 rows, showing 10 per page — Page 1 of 1

STUDENT / ML Assignment 5 (SACHIT) / Notebook 1

Ready

Run notebook

[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)
```

Ready

Run notebook

[21]

```
# 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 componer

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

Visualize

| | principal compo... | principal compo... | Principal Compo... | class int64 |
|---|---|---|---|---|
| 0 | -10.047372177383725 | 1.4710734673231525 | -6.846408954629359 | 1 |
| 1 | -10.637725107420707 | 1.583746420415047 | -6.830981593917086 | 1 |
| 2 | -13.516185371059361 | -1.2535447014964252 | -6.8187016392245035 | 1 |
| 3 | -9.155083997865033 | 8.833597009358671 | 15.290904219478573 | 1 |
| 4 | -6.764469959461754 | 4.611467295362765 | 15.637128583524104 | 1 |

5 rows, showing 10 per page Page 1 of 1

STUDENT / ML Assignment 5 (SACHIT) / Notebook 1

Ready

Run notebook

[22]

```
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)
```

[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)
```

STUDENT / ML Assignment 5 (SACHIT) / Notebook 1

Ready    Run notebook

```
              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:  0.2504463965964602
```

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

STUDENT / ML Assignment 5 (SACHIT) / Notebook 1

Ready    Run notebook

```
#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
```

[25]

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

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

STUDENT / ML Assignment 5 (SACHIT) / Notebook 1

Ready

Run notebook

[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_state=0)
```

[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 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 uns
```

[31]

```
#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 combinat
```

[32]

```
#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 minimizi
```

PCA :

- Principal component analysis (PCA) is surely the most known and simple unsupervised dimensionality reduction method.
- The first component captures the largest variability of the data, while the second captures the second largest, and so on.

LDA :

- Linear discriminant analysis (LDA) is a supervised machine learning and linear algebra approach for dimensionality reduction.
- LDA finds the linear discriminants in order to maximize the variance between the different categories while minimizing the variance within the class.