ML Assignment 5

jupyter Assignment5 Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

In [1]:
```
# importing 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")
```

In [2]:
```
# 1)Principal Component Analysis
```

In [3]:
```
csv_file='./CC.csv'
dataf = pd.read_csv('CC.csv')
dataf.describe()
```

Out[3]:

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY
count	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000
mean	1564.474828	0.877271	1003.204834	592.437371	411.067645	978.871112	0.490351
std	2081.531879	0.236904	2136.634782	1659.887917	904.338115	2097.163877	0.401371
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	128.281915	0.888889	39.635000	0.000000	0.000000	0.000000	0.083333
50%	873.385231	1.000000	361.280000	38.000000	89.000000	0.000000	0.500000
75%	2054.140036	1.000000	1110.130000	577.405000	468.637500	1113.821139	0.916667
max	19043.138560	1.000000	49039.570000	40761.250000	22500.000000	47137.211760	1.000000

jupyter Assignment5 Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

In [4]:
```
dataf.head(5)
```

Out[4]:

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUEN
0	C10001	40.900749	0.818182	95.40	0.00	95.4	0.000000	0.166
1	C10002	3202.467416	0.909091	0.00	0.00	0.0	6442.945483	0.000
2	C10003	2495.148862	1.000000	773.17	773.17	0.0	0.000000	1.000
3	C10004	1666.670542	0.636364	1499.00	1499.00	0.0	205.788017	0.083
4	C10005	817.714335	1.000000	16.00	16.00	0.0	0.000000	0.083

In [5]:
```
dataf.isna().sum()
```

Out[5]:
```
CUST_ID                           0
BALANCE                           0
BALANCE_FREQUENCY                 0
PURCHASES                         0
ONEOFF_PURCHASES                  0
INSTALLMENTS_PURCHASES            0
CASH_ADVANCE                      0
PURCHASES_FREQUENCY               0
ONEOFF_PURCHASES_FREQUENCY        0
PURCHASES_INSTALLMENTS_FREQUENCY  0
CASH_ADVANCE_FREQUENCY            0
CASH_ADVANCE_TRX                  0
PURCHASES_TRX                     0
CREDIT_LIMIT                      1
PAYMENTS                          0
MINIMUM_PAYMENTS                313
PRC_FULL_PAYMENT                  0
TENURE                            0
dtype: int64
```

jupyter Assignment5 Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [6]:
```
dataf.fillna(dataf.mean(), inplace=True)
dataf.isna().sum()
```

Out[6]:
```
CUST_ID                             0
BALANCE                             0
BALANCE_FREQUENCY                   0
PURCHASES                           0
ONEOFF_PURCHASES                    0
INSTALLMENTS_PURCHASES              0
CASH_ADVANCE                        0
PURCHASES_FREQUENCY                 0
ONEOFF_PURCHASES_FREQUENCY          0
PURCHASES_INSTALLMENTS_FREQUENCY    0
CASH_ADVANCE_FREQUENCY              0
CASH_ADVANCE_TRX                    0
PURCHASES_TRX                       0
CREDIT_LIMIT                        0
PAYMENTS                            0
MINIMUM_PAYMENTS                    0
PRC_FULL_PAYMENT                    0
TENURE                              0
dtype: int64
```

In [7]:
```
x = dataf.iloc[:,1:-1]
y = dataf.iloc[:,-1]
print(x.shape,y.shape)
```

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

In [8]:
```
#1.a Apply PCA on CC Dataset
```

In [9]:
```
pca = PCA(3)
x_pca = pca.fit_transform(x)
principalDataf = pd.DataFrame(data = x_pca, columns = ['principal component 1', 'principal component 2', 'principal component 3']
finalDf = pd.concat([principalDataf, dataf.iloc[:,-1]], axis = 1)
```

jupyter Assignment5 Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [7]:
```
x = dataf.iloc[:,1:-1]
y = dataf.iloc[:,-1]
print(x.shape,y.shape)
```

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

In [8]:
```
#1.a Apply PCA on CC Dataset
```

In [9]:
```
pca = PCA(3)
x_pca = pca.fit_transform(x)
principalDataf = pd.DataFrame(data = x_pca, columns = ['principal component 1', 'principal component 2', 'principal component 3']
finalDf = pd.concat([principalDataf, dataf.iloc[:,-1]], axis = 1)
finalDf.head()
```

Out[9]:

	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

In [10]:
```
#1.b Apply K Means on PCA Result
X = finalDf.iloc[:,0:-1]
y = finalDf.iloc[:,-1]
```

jupyter Assignment5 Last Checkpoint: 7 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [11]:
```
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       0.00      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
```


jupyter Assignment5 Last Checkpoint: 8 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
              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
```

Out[11]:
```
'\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 clusters.\n'
```

jupyter Assignment5 Last Checkpoint: 8 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [12]:
```
#1.c Scaling +PCA + KMeans
x = dataf.iloc[:,1:-1]
y = dataf.iloc[:,-1]
print(x.shape,y.shape)
```

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

In [13]:
```
#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, dataf.iloc[:,-1]], axis = 1)
finalDf.head()
```

Out[13]:

	principal component 1	principal component 2	principal component 3	TENURE
0	-1.718893	-1.072937	0.535666	12
1	-1.169306	2.509320	0.627969	12
2	0.938414	-0.382597	0.161192	12
3	-0.907503	0.045860	1.521680	12
4	-1.637830	-0.684972	0.425658	12

In [14]:
```
X = finalDf.iloc[:,0:-1]
y = finalDf["TENURE"]
print(X.shape,y.shape)
```

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

jupyter Assignment5 Last Checkpoint: 8 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [15]:
```
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 mat
"""
```

```
              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
```

jupyter Assignment5 Last Checkpoint: 9 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

```
              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]
 [ 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    0]
 [ 121   29    1    0    0    0    0    0    0    0]
 [ 212   47    3    0    0    0    0    0    0    0]
 [3609 1310   55    0    0    0    0    0    0    0]]
Accuracy for our Training dataset with PCA: 0.0
Sihouette Score:  0.47850291668322187
```

Out[15]:

```
'\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 clusters.\n'
```

jupyter Assignment5 Last Checkpoint: 9 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [16]:

```
# 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 cluster and poorly mat
"""
```

```
              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
```

jupyter Assignment5 Last Checkpoint: 9 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
              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]
 [  53   11    1    0    0    0    0    0    0    0]
 [  47    7    1    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    0]
 [1886  689   35    0    0    0    0    0    0    0]]

Accuracy for our Training dataset with PCA: 0.0
Sihouette Score:  0.4675667597335984
```

Out[16]:
```
'\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 clusters.\n'
```


jupyter Assignment5 Last Checkpoint: 10 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [17]:
```
# 2) Use pd_speech_features.csv
```

In [18]:
```
csv_file='./pd_speech_features.csv'
dataf_pd = pd.read_csv(csv_file)
dataf_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
```

In [19]:
```
dataf_pd.head()
```

Out[19]:

	id	gender	PPE	DFA	RPDE	numPulses	numPeriodsPulses	meanPeriodPulses	stdDevPeriodPulses	locPctJitter	...	tqwt_kurtosisValue_dec_28	tq
0	0	1	0.85247	0.71826	0.57227	240	239	0.008064	0.000087	0.00218	...	1.5620	
1	0	1	0.76686	0.69481	0.53966	234	233	0.008258	0.000073	0.00195	...	1.5589	
2	0	1	0.85083	0.67604	0.58982	232	231	0.008340	0.000060	0.00176	...	1.5643	
3	1	0	0.41121	0.79672	0.59257	178	177	0.010858	0.000183	0.00419	...	3.7805	
4	1	0	0.32790	0.79782	0.53028	236	235	0.008162	0.002669	0.00535	...	6.1727	

5 rows × 755 columns

In [20]:
```
dataf_pd.isna().sum()
```

jupyter Assignment5 Last Checkpoint: 10 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

Out[20]:
```
id                          0
gender                      0
PPE                         0
DFA                         0
RPDE                        0
                           ..
tqwt_kurtosisValue_dec_33   0
tqwt_kurtosisValue_dec_34   0
tqwt_kurtosisValue_dec_35   0
tqwt_kurtosisValue_dec_36   0
class                       0
Length: 755, dtype: int64
```

In [21]:
```
X = dataf_pd.drop('class',axis=1).values
y = dataf_pd['class'].values
```

In [22]:
```
#Scaling Data
scaler = StandardScaler()
X_Scale = scaler.fit_transform(X)
```

In [23]:
```
# Apply PCA with k =3
pca_3 = PCA(n_components=3)
principalComponents = pca_3.fit_transform(X_Scale)

principalDf = pd.DataFrame(data = principalComponents, columns = ['principal component 1', 'principal component 2','Principal Com

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

jupyter Assignment5 Last Checkpoint: 11 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

Out[23]:

	principal component 1	principal component 2	Principal Component 3	class
0	-10.047372	1.471074	-6.846406	1
1	-10.637725	1.583748	-6.830979	1
2	-13.516185	-1.253544	-6.818699	1
3	-9.155084	8.833599	15.290899	1
4	-6.764470	4.611467	15.637118	1

In [24]:
```
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)
```

In [25]:
```
#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)
```

jupyter Assignment5 Last Checkpoint: 11 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
              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:  0.2504463563500205
```

In [26]:
```
#3)Apply Linear Discriminant Analysis (LDA) on Iris.csv dataset to reduce dimensionality of data to k=2.
```

In [27]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
csv_file='./Iris.csv'
dataset_iris = pd.read_csv(csv_file)
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
```

jupyter Assignment5 Last Checkpoint: 12 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [28]:
```
dataset_iris.isna().sum()
```

Out[28]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [29]:
```
x = dataset_iris.iloc[:,1:-1]
y = dataset_iris.iloc[:,-1]
print(x.shape,y.shape)
```

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

In [30]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)
```

In [31]:
```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
le = LabelEncoder()
y = le.fit_transform(y)
```

In [32]:
```
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

Linear transformations are the foundation of both LDA and PCA, which strive to maximize variance in a lower dimension.LDA is a supervised learning algorithm, whereas PCA is an unsupervised one.This indicates that LDA searches for maximum class separability while PCA searches for maximum variance directions regardless of class labels.

#PCA

It combines the features into a smaller set of orthogonal variables known as principle components, which are linear combinations of the original variables. The first component captures the most variability in the data, the second the second most, and so on.

#LDA
LDA finds the linear discriminant in order to maximize the variance between the different categories while minimizing the variance within the class.

GIT HUB LINK:
nimmalapudisriram/ML_Assignment_5 (github.com)