Machine Learning

Assignment-5

Laxma Reddy Nalla

700732071

GitHub Link: [CS-5710/Assignment-5 at dev · LaxmaReddy-Nalla/CS-5710 (github.com)]

YouTube Link: https://youtu.be/qQUGYH6HLFI

Question 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 modules
Importing Datasets using read_csv() function

```
[22] from google.colab import drive
     import pandas as pd
     import numpy as np
     from sklearn.decomposition import PCA
     import seaborn as sns
     import matplotlib.pyplot as plt
```

```
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```

```
[49] cc_df = pd.read_csv('/content/gdrive/MyDrive/ML-assignment/CC.csv')
     cc_df.head()
```

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOFF_
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	

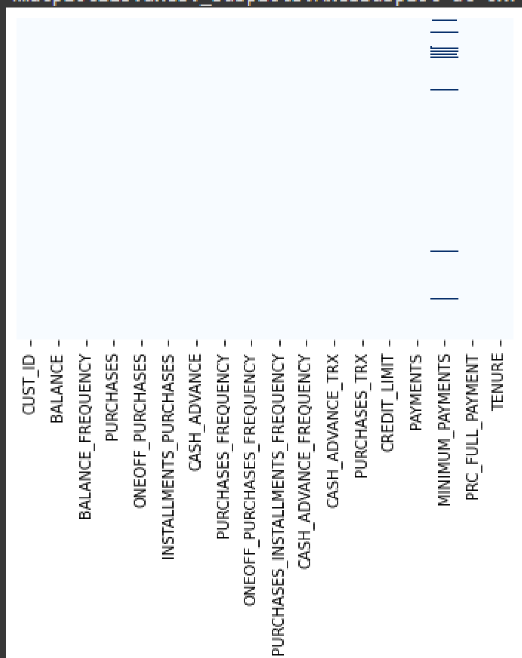
Describing Dataset using describe() function

```
cc_df.describe()
```

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOFF_PURCHASES_FREQUENCY	PURCHASES_INSTALLMENTS_FREQUENCY	CASH_A
count	8950.000000	8950.000000	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	0.202458	0.364437	
std	2081.531879	0.236904	2136.634782	1659.887917	904.338115	2097.163877	0.401371	0.298336	0.397448	
min	0.000000	0.000000	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	0.000000	0.000000	
50%	873.385231	1.000000	361.280000	38.000000	89.000000	0.000000	0.500000	0.083333	0.166667	
75%	2054.140036	1.000000	1110.130000	577.405000	468.637500	1113.821139	0.916667	0.300000	0.750000	
max	19043.138560	1.000000	49039.570000	40761.250000	22500.000000	47137.211760	1.000000	1.000000	1.000000	

```
[52] #We can get a rough idea of our missing Data using a heatmap
sns.heatmap(cc_df.isnull(),yticklabels = False,cbar = False, cmap = "Blues",linecolor = "Black")
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f76f48e0a10>



Checking for Null values in Dataset using heatmap from seaborn

```
cc_df.isnull().sum()
```

```
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
```

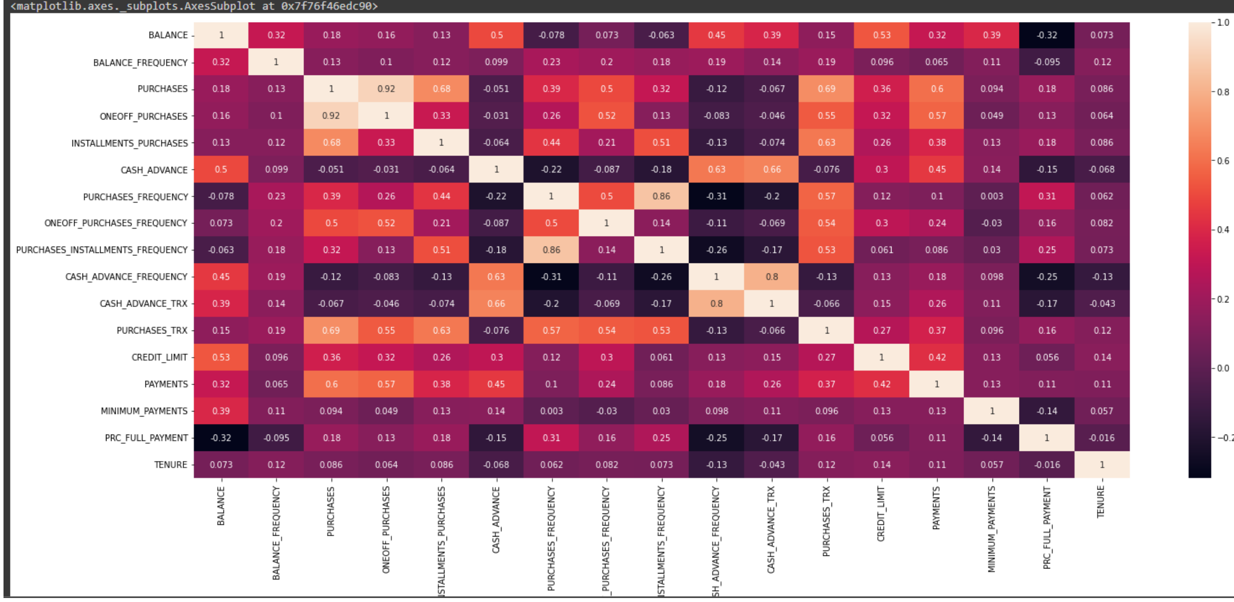
```
[54] cc_df['CREDIT_LIMIT'] = cc_df['CREDIT_LIMIT'].fillna(cc_df['CREDIT_LIMIT'].mean())
     cc_df['MINIMUM_PAYMENTS'] = cc_df['MINIMUM_PAYMENTS'].fillna(cc_df['MINIMUM_PAYMENTS'].mean())
```

Getting count null values and filling null values using mean of the null column

sf

```
f, ax = plt.subplots(figsize = (25,10))
sns.heatmap(cc_df.corr(),annot = True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f76f46edc90>
```



Heatmap of Correlation matrix

```
pca = PCA(2)
x_pca = pca.fit_transform(x)
cc_df2 = pd.DataFrame(data=x_pca, columns=['component_1', 'component_2'])
final_df = pd.concat([cc_df2, cc_df[['TENURE']]], axis=1)

final_df.head()
```

	component_1	component_2	TENURE
0	-4326.383979	921.566882	12
1	4118.916665	-2432.846346	12
2	1497.907641	-1997.578694	12
3	1394.548536	-1488.743453	12
4	-3743.351896	757.342657	12

Implementing Principal component analysis with n_components = 2

```
[64] from sklearn.cluster import KMeans
```

```
[65] score = []
     for i in range(1,11):
       model = KMeans(n_clusters=i)
       model.fit(final_df)
       score.append(model.inertia_)
```

```
[67] #Let's plot our WCSS over the range
     plt.figure(figsize= (10,10))
     plt.plot(score,'bx-')
     plt.xticks(np.arange(1,11, step = 1))
```

Implementing Elbow method to assess number of clusters

```
[68] # we can see the k before the plot get's linear is 4
     # our optimal k for our Data is k = 4
     # Let's apply KMeans
     model = KMeans(n_clusters=4)
     model.fit(final_df)
     labels = model.labels_
```

```
[72] from sklearn.metrics import silhouette_score
     sscore = silhouette_score(final_df, model.labels_, metric='euclidean')
```

```
print(sscore)
```

```
0.5025137542371804
```

Implemented KMeans classifier on pca Dataset and computed silhoutte score

Silhoutte Score: 0.5025137542371804

1.c

```
[103] from sklearn.preprocessing import StandardScaler
      scaler = StandardScaler()
```

```
# removing CUST_ID from Creditdata
# apply StandardScaler to the dataset
x = x.drop(columns=['CUST_ID'],axis=1)
creditDataScaled = scaler.fit_transform(x)
creditDataScaled
```

```
array([[-0.73198937, -0.24943448, -0.42489974, ..., -0.52897879,
        -0.31096755, -0.52555097],
       [ 0.78696085,  0.13432467, -0.46955188, ...,  0.81864213,
         0.08931021,  0.2342269 ],
       [ 0.44713513,  0.51808382, -0.10766823, ..., -0.38380474,
        -0.10166318, -0.52555097],
       ...,
       [-0.7403981 , -0.18547673, -0.40196519, ..., -0.5706145 ,
        -0.33546549,  0.32919999],
       [-0.74517423, -0.18547673, -0.46955188, ..., -0.58053567,
        -0.34690648,  0.32919999],
       [-0.57257511, -0.88903307,  0.04214581, ..., -0.57686873,
        -0.33294642, -0.52555097]])
```

```
[107] # Apply PCA for the scaled Credit Data
      pca = PCA(n_components=2)
      pca_creditdata = pca.fit_transform(creditDataScaled)
      pca_creditdata_df = pd.DataFrame(pca_creditdata, columns=['component_1', 'component_2'])
      creditfinal_df = pd.concat([pca_creditdata_df, y], axis=1)
      creditfinal_df.head()
```

Applied StandatdScaler for the dataset and applied PCA after StandardScaler

```
score = []
for i in range(1,11):
  model = KMeans(n_clusters=i)
  model.fit(creditfinal_df)
  score.append(model.inertia_)
```

```
[110] #Let's plot our WCSS over the range
      plt.figure(figsize= (10,10))
      plt.plot(score,'bx-')
      plt.xticks(np.arange(1,11, step = 1))
```

Predicting number of clusters by using Elbow method

```
# we can see the k before the plot get's linear is 4
# our optimal k for our Data is k = 4
# Let's apply KMeans
model = KMeans(n_clusters=4)
model.fit(final_df)
labels = model.labels_
```

```
[112] sscore = silhouette_score(final_df, model.labels_, metric='euclidean')
      print(sscore)

      0.43767053303119136
```

Computed Silhutte score after apply KMeans on StandardScaler_PCA dataset

Question2:

```
import pandas as pd
import numpy as np
from google.colab import drive
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
```

```
[37] drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[38] df = pd.read_csv('/content/drive/MyDrive/ML-assignment/pd_speech_features.csv')
```

```
[39] df.head()
```

	id	gender	PPE	DFA	RPDE	numPulses	numPeriodsPulses	meanPeriodPulses	stdDevPeriodPulses	locPctJitter	...	tqwt_kurtosisValue_dec_28	tqw
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	

Importing Required Modules and importing Dataset

```
[41] df.columns

Index(['id', 'gender', 'PPE', 'DFA', 'RPDE', 'numPulses', 'numPeriodsPulses',
       'meanPeriodPulses', 'stdDevPeriodPulses', 'locPctJitter',
       ...
       'tqwt_kurtosisValue_dec_28', 'tqwt_kurtosisValue_dec_29',
       'tqwt_kurtosisValue_dec_30', 'tqwt_kurtosisValue_dec_31',
       'tqwt_kurtosisValue_dec_32', 'tqwt_kurtosisValue_dec_33',
       'tqwt_kurtosisValue_dec_34', 'tqwt_kurtosisValue_dec_35',
       'tqwt_kurtosisValue_dec_36', 'class'],
      dtype='object', length=755)
```

```
[42] df.isnull().sum().sum()
     # The Value return as zero hence there are no null values

     0
```

Checking Column names And Null values inside dataset there are no null values in dataset

```
[43] def scale_inputdata(df):
       df = df.drop('id', axis=1)
       y = df['class']
       x = df.drop('class', axis=1)
       scaler = StandardScaler()

       x = pd.DataFrame(scaler.fit_transform(x), columns=x.columns, index= x.index)

       return x, y
```

```
[44] x, y = scale_inputdata(df)
```

Scaling Dataset using StandardScaler module

```
def pca_inputdata(x,y):
  pca = PCA(n_components=3)
  x_pca = pca.fit_transform(x)
  pca_df = pd.DataFrame(data=x_pca, columns=['component_1', 'component_2', 'component_3'])
  final_df = pd.concat([pca_df, y],axis=1)
  return final_df
```

```
[47] final_pca_df = pca_inputdata(x,y)
     final_pca_df
```

Apply PCA on ScaledDataset

```python
def svm_classifier(dataset):
  x = dataset.iloc[:,:-1]
  y = dataset.iloc[:,-1]

  x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.4, random_state=101)

  classifier = SVC(kernel='linear')
  classifier.fit(x_train, y_train)
  y_pred = classifier.predict(x_test)
  accuracy = accuracy_score(y_test, y_pred)
  report = classification_report(y_test, y_pred)
  return accuracy, report
```

```python
[57] accuracy, report = svm_classifier(final_pca_df)
     print("accuracy of SVM Classifier is: ", accuracy)
     print("Classification report of SVM Classifier is: \n", report)
```

```
accuracy of SVM Classifier is:  0.8085808580858086
Classification report of SVM Classifier is: 
              precision    recall  f1-score   support

           0       0.69      0.31      0.43        70
           1       0.82      0.96      0.88       233

    accuracy                           0.81       303
   macro avg       0.76      0.64      0.66       303
weighted avg       0.79      0.81      0.78       303
```

Applied SVM with linear kernel on pca dataset and reported accuracy_score and Classification_report
Accuracy_score : 0.808580

Question 3:

```python
[20] import pandas as pd
     import numpy as np
     from google.colab import drive
     from sklearn.preprocessing import StandardScaler, LabelEncoder
     from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
     import matplotlib.pyplot as plt
     import seaborn as sns
```

```python
[11] drive.mount("/content/drive")
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```python
[12] df = pd.read_csv("/content/drive/MyDrive/ML-assignment/Iris.csv")
     df.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

Importing required Dataset and importing Dataset

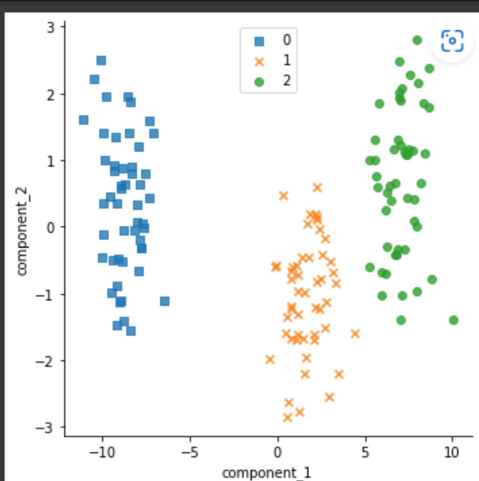
```
[13] x = df.iloc[:,:-1]
     y = df.iloc[:,-1]
     scaler = StandardScaler()
     x = scaler.fit_transform(x)
```

```
[15] le = LabelEncoder()
     y = le.fit_transform(y)
```

```
[19] lda = LinearDiscriminantAnalysis(n_components=2)
     x_lda = lda.fit_transform(x,y)
     df2 = pd.DataFrame(data=x_lda)
     df2['class'] = y
     df2.columns = ["component_1", "component_2", "class"]
     df2.head()
```

Applying StandardScaler and LinearDiscriminantAnalysis on Dataset

```
markers = ['s', 'x', 'o']
colors = ['r', 'b', 'g']
sns.lmplot(x="component_1", y="component_2", data=df2, hue='class', markers=markers, fit_reg=False, legend=False)
plt.legend(loc='upper center')
plt.show()
```



Plotted each class using scatter plot

Question 4:

Briefly identify the difference between PCA and LDA

PCA and LDA are two widely used dimensionality reduction methods for data with a large number of input features.

Both LDA and PCA are linear transformation algorithms, although LDA is supervised whereas PCA is unsupervised and PCA does not take into account the class labels.
PCA minimizes the number of dimensions in high-dimensional data by locating the largest variance.
The purpose of LDA is to determine the optimum feature subspace for class separation.

Both approaches rely on dissecting matrices of eigenvalues and eigenvectors, however, the core learning approach differs significantly. LDA is supervised, whereas PCA is unsupervised.
PCA minimizes dimensions by examining the relationships between various features. This is accomplished by constructing orthogonal axes – or principle components – with the largest variance direction as a new subspace.

PCA generates components based on the direction in which the data has the largest variation - for example, the data is the most spread out. This component is known as both principals and eigenvectors, and it represents a subset of the data that contains the majority of our data's information – or variance.

On the other hand, LDA does almost the same thing, but it includes a "pre-processing" step that calculates mean vectors from class labels before extracting eigenvalues.

PCA

1. Take the joint covariance – or correlation in some circumstances – between each pair in the supplied vector to create the covariance matrix.

LDA

1. Calculate the d-dimensional mean vector for each class label.
2. Create a scatter matrix for each class as well as between classes.