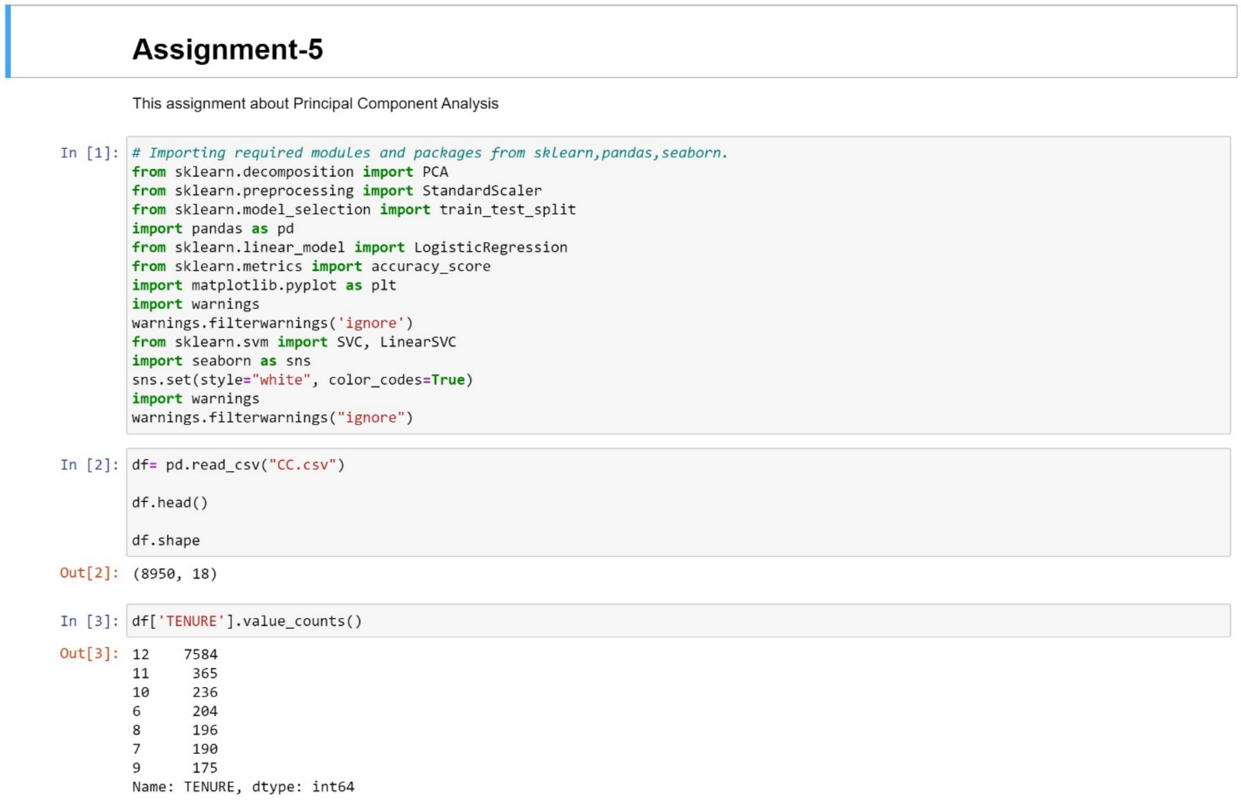
Assignment – 5

Name: Satya Ishyanth Kadali

Student id: #700735513

CRN: 11813, Subject code: CS 5710

Programming elements: Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA)



Assignment-5

This assignment about Principal Component Analysis

In [1]:
```
# Importing required modules and packages from sklearn,pandas,seaborn.
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
from sklearn.svm import SVC, LinearSVC
import seaborn as sns
sns.set(style="white", color_codes=True)
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
df= pd.read_csv("CC.csv")

df.head()

df.shape
```

Out[2]:
```
(8950, 18)
```

In [3]:
```
df['TENURE'].value_counts()
```

Out[3]:
```
12    7584
11     365
10     236
6      204
8      196
7      190
9      175
Name: TENURE, dtype: int64
```

Importing required modules and packages to visualize the data.

To perform scaling on dataset columns

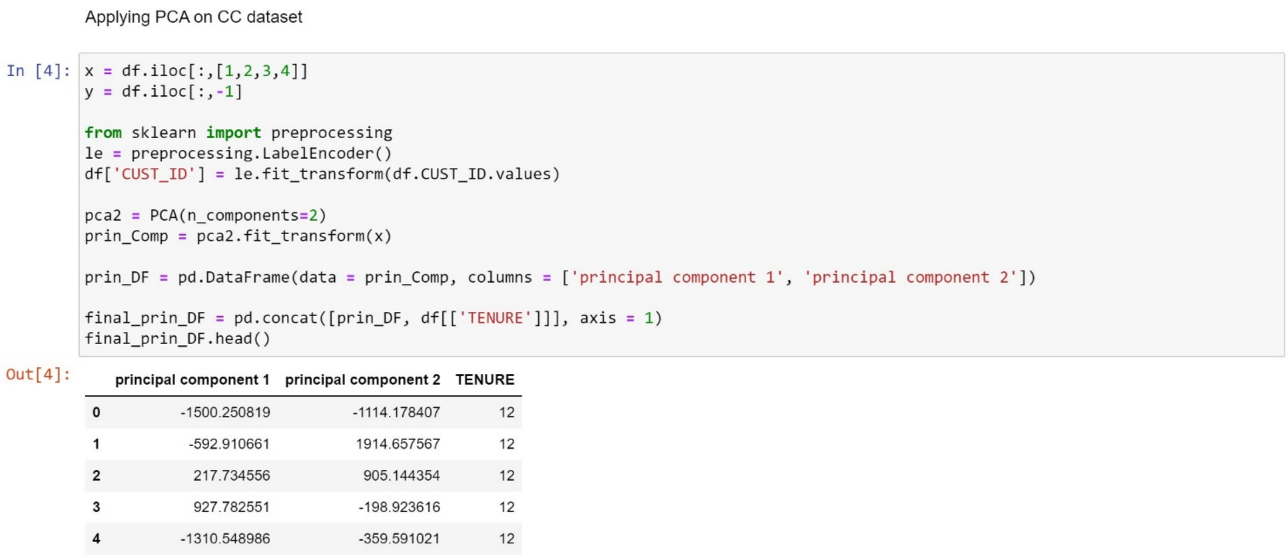
To perform PCA and LDA

To perform K-means and Support Vector Machine

And Reading dataset into pandas dataframe.

PCA simplifies the complexity in high-dimensional data while retaining patterns. It does this by transforming the data into fewer dimensions.

We are applying PCA on CC dataset here:



Applying PCA on CC dataset

In [4]:
```
x = df.iloc[:,[1,2,3,4]]
y = df.iloc[:,-1]

from sklearn import preprocessing
le = preprocessing.LabelEncoder()
df['CUST_ID'] = le.fit_transform(df.CUST_ID.values)

pca2 = PCA(n_components=2)
prin_Comp = pca2.fit_transform(x)

prin_DF = pd.DataFrame(data = prin_Comp, columns = ['principal component 1', 'principal component 2'])

final_prin_DF = pd.concat([prin_DF, df[['TENURE']]], axis = 1)
final_prin_DF.head()
```

Out[4]:

	principal component 1	principal component 2	TENURE
0	-1500.250819	-1114.178407	12
1	-592.910661	1914.657567	12
2	217.734556	905.144354	12
3	927.782551	-198.923616	12
4	-1310.548986	-359.591021	12

After applying PCA we are not applying K-means algorithm on the PCA result.

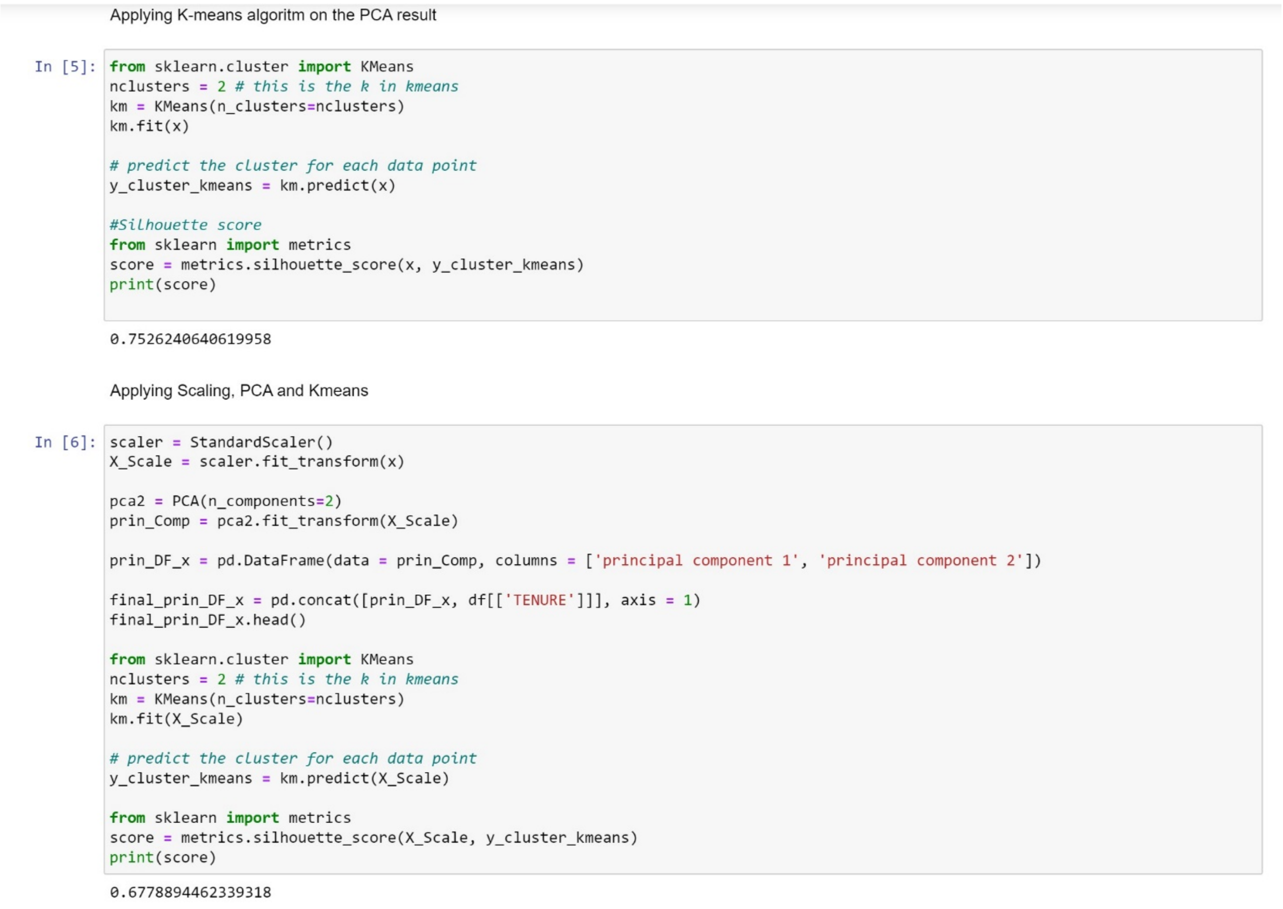
K-means algorithm is an iterative algorithm which tries to partition the dataset into Kpre-defined clusters where each data point belongs to only one group.

By using Silhouette metrics, we can calculate performance of the model score.

Now, we applying Scaling to transform the data to fit in a specific range. It will helps the algorithms like K-means and support vector machine.

Then after we applying Principal component analysis.

And finally predicting the result by K-means algorithm and using Silhouette metrics, we get the performance of the model.



Applying K-means algoritm on the PCA result

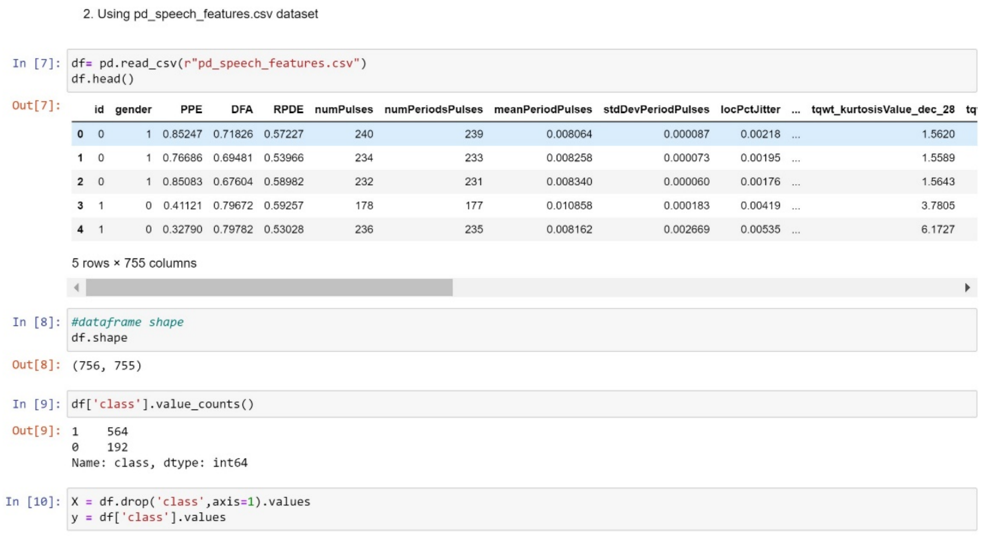
```
In [5]: from sklearn.cluster import KMeans
        nclusters = 2 # this is the k in kmeans
        km = KMeans(n_clusters=nclusters)
        km.fit(x)

        # predict the cluster for each data point
        y_cluster_kmeans = km.predict(x)

        #Silhouette score
        from sklearn import metrics
        score = metrics.silhouette_score(x, y_cluster_kmeans)
        print(score)

        0.7526240640619958
```

Applying Scaling, PCA and Kmeans

```
In [6]: scaler = StandardScaler()
        X_Scale = scaler.fit_transform(x)

        pca2 = PCA(n_components=2)
        prin_Comp = pca2.fit_transform(X_Scale)

        prin_DF_x = pd.DataFrame(data = prin_Comp, columns = ['principal component 1', 'principal component 2'])

        final_prin_DF_x = pd.concat([prin_DF_x, df[['TENURE']]], axis = 1)
        final_prin_DF_x.head()

        from sklearn.cluster import KMeans
        nclusters = 2 # this is the k in kmeans
        km = KMeans(n_clusters=nclusters)
        km.fit(X_Scale)

        # predict the cluster for each data point
        y_cluster_kmeans = km.predict(X_Scale)

        from sklearn import metrics
        score = metrics.silhouette_score(X_Scale, y_cluster_kmeans)
        print(score)

        0.6778894462339318
```

Coming to the 2nd dataset we have pd_speech_features.

Importing dataset into pandas dataframe.



2. Using pd_speech_features.csv dataset

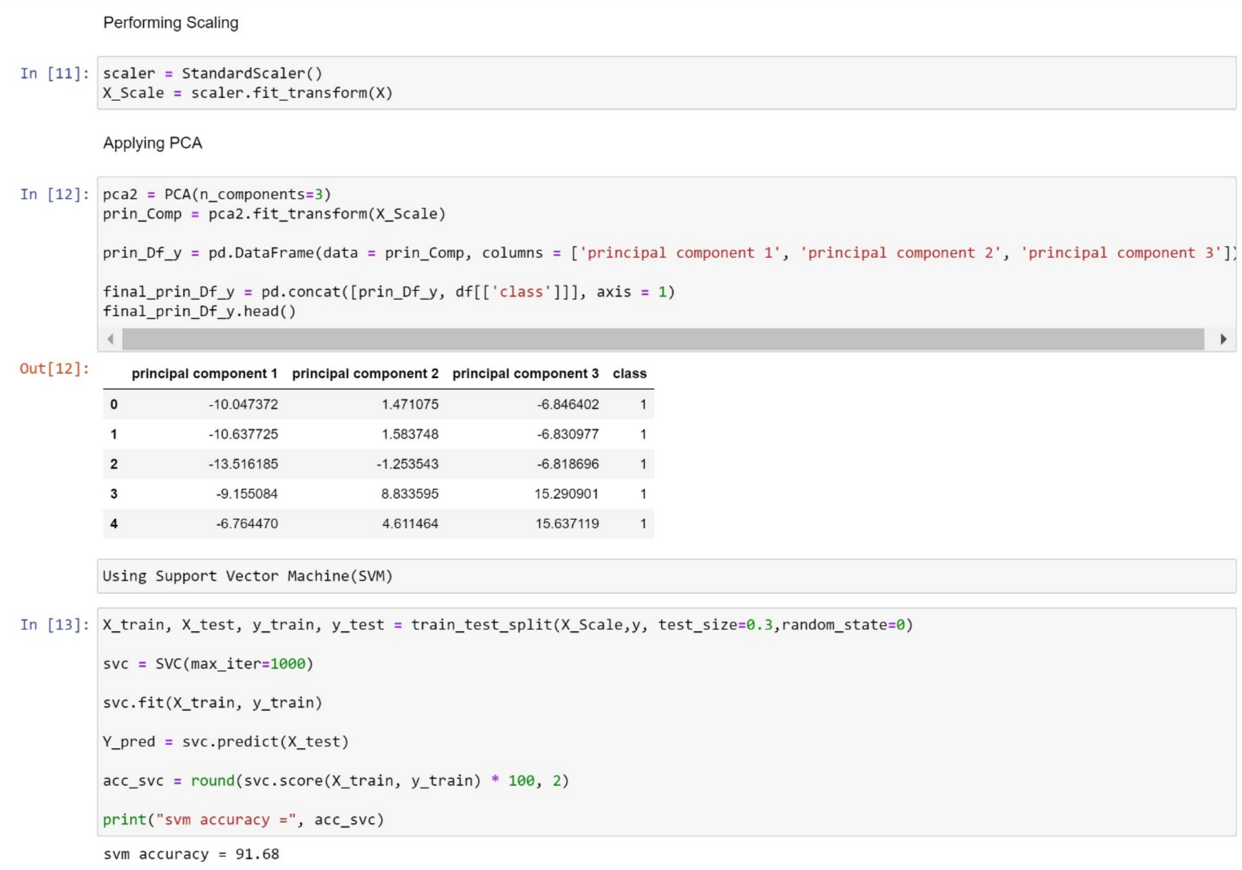
```
In [7]: df= pd.read_csv(r"pd_speech_features.csv")
        df.head()
```

Out[7]:

	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 [8]: #dataframe shape
        df.shape

Out[8]: (756, 755)

In [9]: df['class'].value_counts()

Out[9]: 1    564
        0    192
        Name: class, dtype: int64

In [10]: X = df.drop('class',axis=1).values
         y = df['class'].values
```



Performing Scaling

```
In [11]: scaler = StandardScaler()
         X_Scale = scaler.fit_transform(X)
```

Applying PCA

```
In [12]: pca2 = PCA(n_components=3)
         prin_Comp = pca2.fit_transform(X_Scale)

         prin_Df_y = pd.DataFrame(data = prin_Comp, columns = ['principal component 1', 'principal component 2', 'principal component 3']

         final_prin_Df_y = pd.concat([prin_Df_y, df[['class']]], axis = 1)
         final_prin_Df_y.head()
```

Out[12]:

	principal component 1	principal component 2	principal component 3	class
0	-10.047372	1.471075	-6.846402	1
1	-10.637725	1.583748	-6.830977	1
2	-13.516185	-1.253543	-6.818696	1
3	-9.155084	8.833595	15.290901	1
4	-6.764470	4.611464	15.637119	1

```
Using Support Vector Machine(SVM)
```

```
In [13]: X_train, X_test, y_train, y_test = train_test_split(X_Scale,y, test_size=0.3,random_state=0)

         svc = SVC(max_iter=1000)

         svc.fit(X_train, y_train)

         Y_pred = svc.predict(X_test)

         acc_svc = round(svc.score(X_train, y_train) * 100, 2)

         print("svm accuracy =", acc_svc)

         svm accuracy = 91.68
```

Now I performed scaling and applying Principal Component Analysis.

By using Support vector Machine algorithm on PCA result.

3. Applying Linear Discriminant ANALYSIS ON Iris dataset.

LDA is a type of linear combination, a mathematical process using various data items and applying functions to that set to separately analyze multiple classes of objects.

We are reducing the dimensionality of the data to k=2.

In the Visualization we can see that LDA separated the classes with maximum separability.

```
Applying Linear Discriminant Analysis (LDA) on Iris dataset
```

In [14]:
```
import math
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt


df = pd.read_csv("iris.csv")

df.head()
```

Out[14]:

	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

In [15]:
```
#Applying scaling
from sklearn.preprocessing import StandardScaler
stdsc = StandardScaler()
X_train_std = stdsc.fit_transform(df.iloc[:,range(0,4)].values)
```

In [16]:
```
from sklearn.preprocessing import LabelEncoder
class_le = LabelEncoder()
y = class_le.fit_transform(df['Species'].values)
```

Reducing the Dimensionality to 2

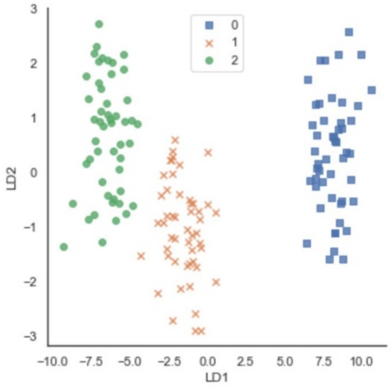
In [17]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis(n_components=2)
X_train_lda = lda.fit_transform(X_train_std,y)
```

In [18]:
```
data=pd.DataFrame(X_train_lda)
data['class']=y
data.columns=["LD1","LD2","class"]
data.head()
```

Out[18]:

	LD1	LD2	class
0	9.423452	-0.513976	0
1	8.751900	-1.591678	0
2	8.973004	-1.068204	0
3	8.170186	-1.435135	0
4	9.249789	-0.136869	0

In [19]:
```
markers = ['s', 'x', 'o']
colors = ['r', 'b', 'g']
sns.lmplot(x="LD1", y="LD2", data=data, hue='class', markers=markers, fit_reg=False, legend=False)
plt.legend(loc='upper center')
plt.show()
```



Differences between PCA and LDA

1. PCA is an unsupervised learning algorithm while LDA is a supervised learning algorithm.
2. PCA as a technique that finds the directions of maximal variance
3. Both LDA and PCA rely on linear transformations and aim to maximize the variance in a lower dimension.
4. In contrast to PCA, LDA attempts to find a feature subspace that maximizes class separability
5. Principal Component Analysis is a commonly used unsupervised linear transformation technique.
6. PCA reduces the number of dimensions by finding the maximum variance in high dimensional data.
7. Linear Discriminant Analysis is a supervised method that takes class labels into account when reducing the number of dimensions. The goal of LDA is to find a feature subspace that best optimizes class separability.