# Machine Learning
# Assignment -5

**Student: Sowjanya Sunkavalli**
**ID: 700731896**

## 1. Principal Component Analysis
## a. Apply PCA on CC dataset

### Principal Component Analysis

In [52]:
```
from sklearn.decomposition import PCA # importing  PCA module from sklearn.decomposition library
from sklearn.preprocessing import StandardScaler # importing StandardScaler from sklearn.preprocessing library
from sklearn.model_selection import train_test_split # Importing train_test_split module from sklearn.model_selection li
import pandas as pd # Importing Pandas as pd
from sklearn.linear_model import LogisticRegression # Importing LogisticRegression module from sklearn.linear_model libr
from sklearn.metrics import accuracy_score # Importing accuracy_score from sklearn.metrics library
import matplotlib.pyplot as plt # Importing python plot as plt from matplotlib.pyplot library
import warnings # Importing Warnings library and filtering Warnings to Ignore
warnings.filterwarnings('ignore')
```

Importing different required modules from different Libraries. Which includes following modules

PCA module from sklearn.decomposition library, StandardScaler from sklearn.preprocessing library, train_test_split module from sklearn.model_selection library, LogisticRegression module from sklearn.linear_model library, accuracy_score from sklearn.metrics library, accuracy_score from sklearn.metrics library, Warnings library and filtering Warnings to Ignore

In [53]:
```
cc = pd.read_csv('CC.csv') # Reading CC csv file using read_csv() function
cc.fillna(cc.mean(),axis=0,inplace=True) # Filling the null values with Mean of corresponding columns
```

Reading CC.csv file using read_csv() from Pandas.

Filling null values with Mean values of the Corresponding Columns in the Dataframe.

In [54]:
```
X = cc.drop(columns=['TENURE','CUST_ID']) # Dropping TENURE and CUST_ID and placing remaining columns as Features X
y = cc['TENURE'] # Assigning TENURE column as Label Y
```

Assigning Features values as all the values in the Dataframe except TENURE and CUST_ID to X

Assigning Label values as TENURE column in the Dataframe to Y.

In [55]:
```
pca2 = PCA(n_components=2) # Applying PCA with components 2
principalComponents = pca2.fit_transform(X) # Feeding X Feature data to model
#Creating a new Dataframe  principalDf with data as principal components and corresponding columns
principalDf = pd.DataFrame(data = principalComponents, columns = ['principal component 1', 'principal component 2'])
finalcc = pd.concat([principalDf, cc[['TENURE']]], axis = 1) # Concatinating principalDf and TENURE data in cc to final 
finalcc.head() # Printing the First 5 rows in the dataframe
```

Out[55]:

| | principal component 1 | principal 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 |

Creating PCA(Principal Component Analysis) datamodel with the number of components as 2.

Feeding the Features data to the model as Principal Components.

Creating new Dataframe with data items as Principal Components and Corresponding Columns as Principal Component1, Principal Component2.

Concatenating principalDf and TENURE columns values to final dataframe as finalCC.

Printing the first 5 rows from the Dataframe using head() function.

In [56]:

```
X_pca = finalcc.drop('TENURE',axis=1) # Dropping TENURE and placing remaining columns as Features X
y_pca = finalcc['TENURE'] # Assigning TENURE column as Label Y
nclusters = 3 # Declaring number of clusters as 3
```

Creating PCA features and Lables as x_pca and y_pca and assigning the corresponding columns as except TENURE remaining all columns for x_pca and TENURE column as label for y_pca

Declaring the number of clusters as 3.

**b. Apply k-means algorithm on the PCA result and report your observation if the silhouette score has improved or not?**

In [84]:

```
from sklearn.cluster import KMeans # Importing KMeans from sklearn.cluster
km = KMeans(n_clusters=nclusters)  # Assigning KMeans with n clusters
km.fit(X_pca)  # Feeding the X_pca Feature Data to model
y_pred = km.predict(X_pca) # Predicting the label from features X_pca
from sklearn.metrics import silhouette_score # importing silhoutte_score from sklearn.metrics
score = silhouette_score(X_pca, y_pred) # finding the silhouette_score
print(score) # printing silhouette_score
```

```
0.5720003159007088
```

In [39]:

```
#  Silhouette scoreIt got Improved after applying PCA
```

Importing KMeans from sklearn.cluster, Assigning KMeans with n clusters.

Feeding the X_pca to the Model KMeans using fit () method.

Predicting Y values from X_pca values using Predict () method.

Importing silhoutte_score from sklearn.metrics.

Finding the silhouette_score of X_pca, y_pred using silhouette_score() method.

Silhouette score got increased after PCA with KMeans.

**c. Perform Scaling+PCA+K-Means and report performance.**

```
In [59]:
scaler = StandardScaler() # creating a StandardScalar to scale the data
scaler.fit(X) # feeding the data to the scalar
X_scaled_array = scaler.transform(X) # scaling the data using Standard Scalar function
X_scaled = pd.DataFrame(X_scaled_array, columns = X.columns) # creating a new dataframe with scaled data

pca2 = PCA(n_components=2) # Creating a model PCA with number of components as 2.
principalComponents = pca2.fit_transform(X_scaled) # Feeding the X_scaled data to pca2
 #creating new df with columns as required
principalDf = pd.DataFrame(data = principalComponents, columns = ['principal component 1', 'principal component 2'])
finalcc = pd.concat([principalDf, cc[['TENURE']]], axis = 1)
X_pca = finalcc.drop('TENURE',axis=1) # dropping TENURE and retaining other columns and considering as X_pca
y_pca = finalcc['TENURE'] # TENURE column is considered as y_pca

km = KMeans(n_clusters=nclusters) # creating the KMeans model
km.fit(X_pca) # feeding the model with PCA data
y_cluster_kmeans = km.predict(X_pca) # predicting the y values with PCA values of x
from sklearn.metrics import silhouette_score # importing silhoutte_score from sklearn.metrics
score = silhouette_score(X_pca, y_cluster_kmeans) # finding the silhoutte_score
print(score) # printing silhoutte_score

0.4527288364707121
```

```
In [ ]:
# Performance got decreased after Scaling.
```

creating a StandardScalar to scale the data using StandardScaler() model.

Feeding the feature data to the model using fit() method.

scaling the data using Standard Scalar function using transform() method.

Creating a new dataframe with scaled data using pandas.

Creating a model PCA with number of components as 2. Using PCA(n_components=value) model.

Feeding the X_scaled data to pca2 and creating new df with columns as required with data as PrincipalComponets and columns as 'principal component 1', 'principal component 2'.

Dropping TENURE and retaining other columns and considering as X_pca using drop() function.

TENURE column is considered as y_pca

Creating the KMeans model using KMeans(n_clusters=nclusters)

Feeding the model with PCA feature data using fit() method.

Predicting the y values with PCA values of x using predict() method.

Importing silhoutte_score from sklearn.metrics

Finding the silhoutte_score using silhouette_score()

Printing silhouette_score.

After Observing the different models we can say the Performance got decreased for Scaling+PCA+K-Means combination.

## 2. Use pd_speech_features.csv

### a. Perform Scaling

In [60]:
```
speech = pd.read_csv('pd_speech_features.csv') #reading the data using pandas
```

In [61]:
```
X = speech.drop(columns = ['class']) # Considering Feature values
y = speech['class'] # Considering Label Values
```

In [62]:
```
scaler = StandardScaler() # creating a StandardScalar to scale the data
scaler.fit(X) # feeding the data to the scalar
X_scaled_array = scaler.transform(X) # scaling the data using Standard Scalar function
X_scaled = pd.DataFrame(X_scaled_array, columns = X.columns) # creating a new dataframe with scaled data
```

Reading the data pd_speech_features.csv using pandas read_csv().

Considering Features values as X from dropping column class and remaining all other columns.

Considering Label values as Y from the column class.

Creating StandardScaler model and assigning to scaler.

Feeding the data to the model using fit() method.

Scaling the data using transform function from the feature data of X.

Creating a new Dataframe with the scaled data using pandas.

### b. Apply PCA (k=3)

In [63]:
```
pca3 = PCA(n_components=3) # Creating a model PCA with number of components as 3.
principalComponents = pca3.fit_transform(X_scaled) # Feeding the X_scaled data to pca3

 #creating new df with columns as required
principalDf = pd.DataFrame(data = principalComponents, columns = ['principal component 1', 'principal component 2','prin

finalspeech = pd.concat([principalDf, speech[['class']]], axis = 1)# Concatinating principalDf and TENURE data in CC to 
finalspeech.head() # Displaying first 5 rows
```

Out[63]:

| | principal component 1 | principal component 2 | principal component 3 | class |
|---|---|---|---|---|
| 0 | -10.047372 | 1.471075 | -6.846403 | 1 |
| 1 | -10.637725 | 1.583751 | -6.830980 | 1 |
| 2 | -13.516185 | -1.253542 | -6.818697 | 1 |
| 3 | -9.155083 | 8.833602 | 15.290899 | 1 |
| 4 | -6.764469 | 4.611470 | 15.637116 | 1 |

Creating a model PCA with number of components as 3.

Feeding the X_scaled data to pca3 using transform() method.

Creating new df with columns as required as 'principal component 1', 'principal component 2','principal component 3' ad data as pricipalComponents.

Concatinating principalDf and TENURE data in CC to final dataframe using concat() function.

Printing only first 5 rows of the dataframe using head() function.

### c. Use SVM to report performance

```
In [64]:
X_pca = finalspeech.drop('class',axis=1) # Considering X_pca features
y_pca = finalspeech['class'] # Considering y_pca labels
```

```
In [65]:
# Splitting the Data into Test and Train Data
X_train, X_test, y_train, y_test = train_test_split(X_pca, y_pca, test_size = 0.2, random_state = 0)
```

Considering X_pca as Features of PCA by taking all others columns except class column.

Considering labels as Y_pca with the class column data.

Splitting the data into train and test data using train_test_split() method.

```
In [66]:
from sklearn.svm import SVC # importing SVC
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix # Importing different parameters from
classifier = SVC() # creating the model
classifier.fit(X_train, y_train) # feeding model with training dataset
y_pred = classifier.predict(X_test) # predicting the dependent variable in the test dataset
```

```
In [67]:
print(classification_report(y_test, y_pred)) # printing the classification report
print(confusion_matrix(y_test, y_pred)) # printing the confusion matrix
# Accuracy score
from sklearn.metrics import accuracy_score # importing the accuracy_score
print('accuracy is',accuracy_score(y_pred,y_test))
```

```
              precision    recall  f1-score   support

           0       0.67      0.42      0.52        38
           1       0.83      0.93      0.88       114

    accuracy                           0.80       152
   macro avg       0.75      0.68      0.70       152
weighted avg       0.79      0.80      0.79       152

[[ 16  22]
 [  8 106]]
accuracy is 0.8026315789473685
```

Importing svc from sklearn.svm library and accuracy score, classification_report and confusion_matrix from sklearn.metrics library.

Creating the model svc() and assigning to the Classifier.

Feeding the train data to the Classifier using fit() method.

Predicting the y values from the X_test values using predict() method.

Printing classification report using classification_report() by passing Y_test and y_pred as parameters.

Printing Confusion Matrix using confusion_matrix () by passing y_test, y_pred values as parameters.

Importing accuraracy score from sklearn.metrics.

Printing accuracy for y_pred and y_test values using accuracy_score() method.

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

In [68]:

```
## 3rd one
iris = pd.read_csv('Iris.csv') #reading the data using pandas
X = iris.drop(columns=['Species','Id']) #considering x values
y = iris['Species'] #considering y values
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis #importing LDA
lda = LinearDiscriminantAnalysis(n_components=2) # considering dimensions as 2
principalComponents = lda.fit_transform(X,y) #inputting x,y values
principalDf = pd.DataFrame(data = principalComponents, columns = ['principal component 1', 'principal component 2']) #cr

finaliris = pd.concat([principalDf, iris[['Species']]], axis = 1) #concatinating
finaliris.head() #displaying first 5 rows
```

Out[68]:

| | principal component 1 | principal component 2 | Species |
|---|---|---|---|
| **0** | 8.084953 | 0.328454 | Iris-setosa |
| **1** | 7.147163 | -0.755473 | Iris-setosa |
| **2** | 7.511378 | -0.238078 | Iris-setosa |
| **3** | 6.837676 | -0.642885 | Iris-setosa |
| **4** | 8.157814 | 0.540639 | Iris-setosa |

Reading the Iris data using read_csv() method in Pandas.

Considering Feature values as X with dropping Species column and remaining all other columns.

Considering Label values as Y with Species Column.

Importing LinearDiscriminantAnalysis from sklearn.discriminant_analysis library.

Creating a LDA model with 2 components.

Feeding the X, y data to the Model using fit () method.

Creating a new Dataframe with columns as principal component 1', 'principal component 2  and data as PrincipalComponents.

Concatenating principalDf and Species data to the finalIris dataframe.

Printing the top 5 rows of the finalris dataframe using head() method.

## 4. Briefly identify the difference between PCA and LDA

| **Features** | **Principal Component Analysis** | **Linear Discriminant Analysis** |
|---|---|---|
| Type | Unsupervised | Supervised |
| Goal | 1. Training faster<br>2. Visualization | Good for classification |
| Dimensions of new Data | Less than or equal to the original one's | Less than or equal to the result when subtracting 1 from the number classes |
| Computations for large datasets | PCA requires fewer computations | LDA requires significantly more computation thar PCA for Large datasets |

| | | |
|---|---|---|
| Method | Maximize the variance | Maximize between-class variance and minimize within-class variance |
| Well distributed Classes in small datasets | PCA is less superior to LDA | LDA is superior to PCA |
| Discrimination between classes | PCA deals with the data in its entirety for the principal components analysis without paying any particular attention to the underlying class structure | LDA deals directly with discrimination between classes |
| Focus | PCA searches for the directions that have largest variations | LDA maximizes the ration of between-class variation and with-in class variation |
| Directions of maximum discrimination | The directions of maximum variance are not necessarily the directions of the maximum discrimination since there is no attempt to use the class information such as the between class scatter and within-class scatter | LDA is guaranteed to find the optimal discriminant directions When the class densities are Gaussian with the same covariance matrix for all the classes |
| Applications | Application of PCA in the prominent field of criminal Investigation is beneficial | Linear Discriminant Analysis for data classification is applied for classification problem in speech recognition |

---

**GitHub Link:** **https://github.com/sunkavallisowjanya/ML_Assignment5**

**Video Link:** **https://youtu.be/e_-7m4PVNgE**