Predictions using Feature Reduction Techniques PCA and LDA

In [11]:

```
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score,cohen_kappa_score

#csv file
url = 'C:/Users/Prerna/Desktop/ML_jupyter_notenooks/datasets/titanic.csv'

#Creating a dataframe
dataframe = pd.read_csv(url).fillna(0)

#DATA CLEANING
#Dropping columns
dataframe = dataframe.drop('Name',axis=1)
dataframe = dataframe.drop('SexCode',axis=1)

# Create mapper
pclass_mapper = {"1st":1,"2nd":2,"3rd":3}
gender_mapper = {"male":1 ,"female":2}

# Replace feature values with scale
dataframe["PClass"] = dataframe["PClass"].replace(pclass_mapper)
dataframe["Sex"] = dataframe["Sex"].replace(gender_mapper)

#Replacing missing values of Age with mean of age
dataframe["Age"] = np.where(dataframe['Age']==0,np.mean(dataframe['Age']),dataframe['Age'])

#print(dataframe)
# Input features
x = dataframe.iloc[:, :3].values
# Output class
y = dataframe.iloc[:, 3].values


'''
xtrain = training features
xtest = testing features
ytrain = classes of training data
ytest = classes of testing data
```

```
'''
#Splitting the data into training and test
xtrain, xtest, ytrain, ytest = train_test_split( x, y, test_size = 0.3, random_state = 0)

#Applying PCA
pca = PCA(n_components = 0.99)   #(variance = 0.99,0.95)

# apply feature Reduction
x_train = pca.fit_transform(xtrain)
x_test = pca.transform(xtest)

print("Original number of features:", x.shape[1])
print("Reduced number of features:", x_test.shape[1])

#Creating Model
gnb = GaussianNB()
#Fitting a model
gnb.fit(x_train, ytrain)

#Perform Predictions
y_pred = gnb.predict(x_test)

#Confusion Matrix
cm = confusion_matrix(ytest, y_pred)

print ("Confusion Matrix : \n", cm)
error_rate = 1 - accuracy_score(ytest, y_pred)
print ("Classification Report:\n")
print(classification_report(ytest,y_pred))
print("Accuracy: \n", accuracy_score(ytest, y_pred))
print("Error Rate: \n",error_rate )
print("Kappa Score: \n", cohen_kappa_score(ytest, y_pred))


```

```
Original number of features: 3
Reduced number of features: 1
Confusion Matrix :
 [[229  35]
 [111  19]]
Classification Report:
```

```
              precision    recall  f1-score   support

           0       0.67      0.87      0.76       264
           1       0.35      0.15      0.21       130

    accuracy                           0.63       394
   macro avg       0.51      0.51      0.48       394
weighted avg       0.57      0.63      0.58       394

Accuracy:
 0.6294416243654822
Error Rate:
 0.37055837563451777
Kappa Score:
 0.015943615710962145
```

In [12]:

```
import pandas as pd
import numpy as np
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score,cohen_kappa_score


#csv file
url = 'C:/Users/Prerna/Desktop/ML_jupyter_notenooks/datasets/titanic.csv'

#Creating a dataframe
dataframe = pd.read_csv(url).fillna(0)

#DATA CLEANING
#Dropping columns
dataframe = dataframe.drop('Name',axis=1)
dataframe = dataframe.drop('SexCode',axis=1)

# Create mapper
pclass_mapper = {"1st":1,"2nd":2,"3rd":3}
gender_mapper = {"male":1 ,"female":2}

# Replace feature values with scale
dataframe["PClass"] = dataframe["PClass"].replace(pclass_mapper)
dataframe["Sex"] = dataframe["Sex"].replace(gender_mapper)

#Replacing missing values of Age with mean of age
dataframe["Age"] = np.where(dataframe['Age']==0,np.mean(dataframe['Age']),dataframe['Age'])


# Input features
x = dataframe.iloc[:, :3].values
# Output class
y = dataframe.iloc[:, 3].values


'''
xtrain = training features
xtest = testing features
```

```
ytrain = classes of training data
ytest = classes of testing data
'''
#Splitting the data into training and test

xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.3, random_state = 0)

#Using LDA
lda = LDA(n_components=None)

# Transform the features
x_train = lda.fit_transform(xtrain,ytrain)
x_test = lda.transform(xtest)


print("Original number of features:", x.shape[1])
print("Reduced number of features:", x_test.shape[1])

#Create a model
gnb = GaussianNB()

#Fit a model
gnb.fit(x_train, ytrain)

#Perform Predictions
y_pred = gnb.predict(x_test)

#Confusion Matrix
cm = confusion_matrix(ytest, y_pred)

print ("Confusion Matrix : \n", cm)
error_rate = 1 - accuracy_score(ytest, y_pred)
print ("Classification Report:\n")
print(classification_report(ytest,y_pred))
print("Accuracy: \n", accuracy_score(ytest, y_pred))
print("Error Rate: \n",error_rate )
print("Kappa Score: \n", cohen_kappa_score(ytest, y_pred))

```

```
Original number of features: 3
Reduced number of features: 1
```

```
Confusion Matrix :
 [[223  41]
 [ 41  89]]
Classification Report:

              precision    recall  f1-score   support

           0       0.84      0.84      0.84       264
           1       0.68      0.68      0.68       130

    accuracy                           0.79       394
   macro avg       0.76      0.76      0.76       394
weighted avg       0.79      0.79      0.79       394

Accuracy:
 0.7918781725888325
Error Rate:
 0.20812182741116747
Kappa Score:
 0.5293123543123543
```

In [ ]:

```
1
```