In [11]:
```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.naive_bayes import BernoulliNB
from sklearn import svm
from sklearn import metrics
from sklearn.metrics import confusion_matrix
import os

THIS_FOLDER = os.path.abspath('')


#setting the training and testing dataset path
class Dataset:
  train=os.path.join(THIS_FOLDER, 'wineQualityRed_train.csv')
  test=os.path.join(THIS_FOLDER, 'wineQualityRed_test.csv')

pd.options.display.max_rows = None
pd.options.display.max_columns = None

fields=["fixed acidity","volatile acidity","citric acid","residual sugar","ch
        "free sulfur dioxide","total sulfur dioxide","density","pH","sulphate
train_wine = pd.read_csv(Dataset.train, delimiter=';', header=None, skiprows=
test_wine = pd.read_csv(Dataset.test, delimiter=';', header=None, skiprows=1,

# CLASSIFING quality 0-bad AND 1-good and forming Y_TRAINING DATASET
temp = train_wine["quality"].replace([3,4,5,6] , 0)
temp1 = temp.replace([7,8] , 1)
train_wine["quality"] = temp1

tempo = test_wine["quality"].replace([3,4,5,6] , 0)
tempo1 = tempo.replace([7,8] , 1)
test_wine["quality"] = tempo1

##Training Dataset
x_train = train_wine.iloc[:,:11]
Y_train = train_wine.iloc[:,-1:]
y_train = Y_train.values.reshape(-1,1)


## Testing Dataset
x_test = test_wine.iloc[:,:11]
Y_test = test_wine.iloc[:,-1:]
y_test = Y_test.values.reshape(-1,1)


## Feature Scaling
SC = StandardScaler()
x_train_scaled = SC.fit_transform(x_train)
x_test_scaled = SC.fit_transform(x_test)

## Principal Components Analysis to get "7" attributes
Principal_7_comp = PCA(n_components = 7)
x_redwine_7_training = Principal_7_comp.fit_transform(x_train_scaled)
x_redwine_7_testing = Principal_7_comp.fit_transform(x_test_scaled)



## Principal Components Analysis to get "4" attributes
```

```
Principal_4_comp = PCA(n_components = 4)
x_redwine_4_training = Principal_4_comp.fit_transform(x_train_scaled)
x_redwine_4_testing = Principal_4_comp.fit_transform(x_test_scaled)


##############################################################################
```

In [12]:

```
## Linear Regression Classifier

print("*********************************LINEAR REGRESSION CLASSIFIER*********

## "7" Attributes.
print("-------------------7 Attributes--------------------------")
reg1 = LinearRegression()
reg1.fit(x_redwine_7_training , y_train.ravel())

x1_test_pred = reg1.predict(x_redwine_7_testing).round(3)
## HERE WE FIND OUT THE THRESHOLD ABOVE WHICH WE CAN CONSIDER THE QUALITY VAL
## "mid" IS THE VARIABLE CONTAINING THAT THRESHOLD VALUE....

## CONVERTING OUR PREDICTED RESULT AS PER THE THRESHOLD.....

#MEAN OF THE Y_PREDICTION
mid = np.mean(x1_test_pred)

print("mean: ", mid)

pred= []


#Classifing y_prediction based on the mean value
for p in x1_test_pred :
    if p < mid:
        pred.append(0)
    elif p > mid:
        pred.append(1)

print("Y_PREDICTION DATA after classificaion:(first 10 values)")
print(pred[:10])


#Getting the confusion matrix
cm1 = confusion_matrix(y_test,pred)
print('Confusion Matrix : \n', cm1)
#prints [[TP,FN],[FP,TN]]

total1=sum(sum(cm1))


accuracy1=(cm1[0,0]+cm1[1,1])/total1
print ('Accuracy : ', accuracy1)

precision1=(cm1[0,0])/(cm1[0,0]+cm1[1,0])
print ('Precision : ', precision1)


recall1=(cm1[0,0])/(cm1[0,0]+cm1[1,1])
print ('Recall : ', recall1)

f_measure=(2*recall1*precision1)/(precision1+recall1)
print('f measure: ', f_measure)

sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )
```

```
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
```

```
*********************************LINEAR REGRESSION CLASSIFIER***************
*****************************
------------------7 Attributes--------------------------
mean:  0.14567083333333333
Y_PREDICTION DATA after classificaion:(first 10 values)
[1, 0, 0, 0, 1, 1, 0, 1, 1, 0]
Confusion Matrix :
 [[179 247]
 [ 15  39]]
Accuracy :  0.45416666666666666
Precision :  0.9226804123711341
Recall :  0.8211009174311926
f measure:  0.8689320388349516
Sensitivity :  0.42018779342723006
Specificity :  0.7222222222222222
```

In [13]:

```
## "4" Attributes.
print("------------------4 Attributes--------------------------")
rege1 = LinearRegression()
rege1.fit(x_redwine_4_training , y_train.ravel())

x1_test_pred = rege1.predict(x_redwine_4_testing).round(3)
## HERE WE FIND OUT THE THRESHOLD ABOVE WHICH WE CAN CONSIDER THE QUALITY VAL
## "mid" IS THE VARIABLE CONTAINING THAT THRESHOLD VALUE....

mid = np.mean(x1_test_pred)

print("mean: ", mid)

pred= []


#Classifing y_prediction based on the mean value
for p in x1_test_pred :
    if p < mid:
        pred.append(0)
    elif p > mid:
        pred.append(1)

print("Y_PREDICTION DATA after classificaion:(first 10 values)")
print(pred[:10])


#Getting the confusion matrix
cm1 = confusion_matrix(y_test,pred)
print('Confusion Matrix : \n', cm1)
#prints [[TP,FN],[FP,TN]]

total1=sum(sum(cm1))


accuracy1=(cm1[0,0]+cm1[1,1])/total1
print ('Accuracy : ', accuracy1)

precision1=(cm1[0,0])/(cm1[0,0]+cm1[1,0])
print ('Precision : ', precision1)


recall1=(cm1[0,0])/(cm1[0,0]+cm1[1,1])
print ('Recall : ', recall1)

f_measure=(2*recall1*precision1)/(precision1+recall1)
```

```
print('f measure: ', f_measure)

sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )

specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
```

```
------------------4 Attributes--------------------------
mean:  0.14568333333333333
Y_PREDICTION DATA after classificaion:(first 10 values)
[1, 0, 0, 1, 1, 1, 1, 1, 1, 0]
Confusion Matrix :
 [[174 252]
 [ 19  35]]
Accuracy :  0.4354166666666667
Precision :  0.9015544041450777
Recall :  0.8325358851674641
f measure:  0.8656716417910447
Sensitivity :  0.4084507042253521
Specificity :  0.6481481481481481
```

In [14]:

```
##############################################################################

## Logistic Regression As Classifier

print("*****************************LOGISTIC REGRESSION CLASSIFIER**********

## "7" Attributes.
print("------------------7 Attributes--------------------------")
reg = LogisticRegression()
reg.fit(x_redwine_7_training , y_train.ravel())

pred = reg.predict(x_redwine_7_testing)

print("Y_PREDICTION:(first 10 values)")
print(pred[:10])

cm1 = confusion_matrix(y_test,pred)
print('Confusion Matrix : \n', cm1)

total1=sum(sum(cm1))
accuracy1=(cm1[0,0]+cm1[1,1])/total1
print ('Accuracy : ', accuracy1)

precision1=(cm1[0,0])/(cm1[0,0]+cm1[1,0])
print ('Precision : ', precision1)


recall1=(cm1[0,0])/(cm1[0,0]+cm1[1,1])
print ('Recall : ', recall1)

f_measure=(2*recall1*precision1)/(precision1+recall1)
print('f measure: ', f_measure)

sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )

specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
```

```
*****************************LOGISTIC REGRESSION CLASSIFIER****************
**********
------------------7 Attributes--------------------------
Y_PREDICTION:(first 10 values)
[0 0 0 0 0 0 0 0 0 0]
Confusion Matrix :
```

```
 [[419   7]
 [ 53   1]]
Accuracy :  0.875
Precision :  0.8877118644067796
Recall :  0.9976190476190476
f measure:  0.9394618834080718
Sensitivity :  0.9835680751173709
Specificity :  0.018518518518518517
```

In [15]:

```
## "4" Attributes.
print("-------------------4 Attributes--------------------------")
rege = LogisticRegression()
rege.fit(x_redwine_4_training , y_train.ravel())
pred = rege.predict(x_redwine_4_testing)

print("Y_PREDICTION:(first 10 values)")
print(pred[:10])

cm1 = confusion_matrix(y_test,pred)
print('Confusion Matrix : \n', cm1)

total1=sum(sum(cm1))
accuracy1=(cm1[0,0]+cm1[1,1])/total1
print ('Accuracy : ', accuracy1)

precision1=(cm1[0,0])/(cm1[0,0]+cm1[1,0])
print ('Precision : ', precision1)


recall1=(cm1[0,0])/(cm1[0,0]+cm1[1,1])
print ('Recall : ', recall1)

f_measure=(2*recall1*precision1)/(precision1+recall1)
print('f measure: ', f_measure)

sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )

specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
```

```
-------------------4 Attributes--------------------------
Y_PREDICTION:(first 10 values)
[0 0 0 0 0 0 0 0 0 0]
Confusion Matrix :
 [[426   0]
 [ 54   0]]
Accuracy :  0.8875
Precision :  0.8875
Recall :  1.0
f measure:  0.9403973509933775
Sensitivity :  1.0
Specificity :  0.0
```

In [16]:

```
###############################################################################

## 3. SVM Classifier.....

print("*****************************************SVM CLASSIFIER***************

## "7" Attributes.
print("-------------------7 Attributes--------------------------")
classi_svm = svm.SVC(kernel="linear")
classi_svm.fit(x_redwine_7_training , y_train.ravel())

pred = classi_svm.predict(x_redwine_7_testing)
```

```
print("Y_PREDICTION:(first 10 values)")
print(pred[:10])

cm1 = confusion_matrix(y_test,pred)
print('Confusion Matrix : \n', cm1)

total1=sum(sum(cm1))
accuracy1=(cm1[0,0]+cm1[1,1])/total1
print ('Accuracy : ', accuracy1)

precision1=(cm1[0,0])/(cm1[0,0]+cm1[1,0])
print ('Precision : ', precision1)


recall1=(cm1[0,0])/(cm1[0,0]+cm1[1,1])
print ('Recall : ', recall1)

f_measure=(2*recall1*precision1)/(precision1+recall1)
print('f measure: ', f_measure)

sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )

specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
```

```
**************************************SVM CLASSIFIER**********************
************************
-------------------7 Attributes---------------------------
Y_PREDICTION:(first 10 values)
[0 0 0 0 0 0 0 0 0 0]
Confusion Matrix :
 [[422   4]
 [ 53   1]]
Accuracy :  0.88125
Precision :  0.888421052631579
Recall :  0.9976359338061466
f measure:  0.9398663697104677
Sensitivity :  0.9906103286384976
Specificity :  0.018518518518518517
```

In [17]:

```
## "4" Attributes.
print("-------------------4 Attributes---------------------------")
classi_svm = svm.SVC(kernel="linear")
classi_svm.fit(x_redwine_4_training , y_train.ravel())

pred = classi_svm.predict(x_redwine_4_testing)


print("Y_PREDICTION:(first 10 values)")
print(pred[:10])

cm1 = confusion_matrix(y_test,pred)
print('Confusion Matrix : \n', cm1)

total1=sum(sum(cm1))
accuracy1=(cm1[0,0]+cm1[1,1])/total1
print ('Accuracy : ', accuracy1)

precision1=(cm1[0,0])/(cm1[0,0]+cm1[1,0])
print ('Precision : ', precision1)


recall1=(cm1[0,0])/(cm1[0,0]+cm1[1,1])
```

```
print ('Recall : ', recall1)

f_measure=(2*recall1*precision1)/(precision1+recall1)
print('f measure: ', f_measure)

sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )

specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
```

```
------------------4 Attributes--------------------------
Y_PREDICTION:(first 10 values)
[0 0 0 0 0 0 0 0 0 0]
Confusion Matrix :
 [[426   0]
 [ 54   0]]
Accuracy :  0.8875
Precision :  0.8875
Recall :  1.0
f measure:  0.9403973509933775
Sensitivity :  1.0
Specificity :  0.0
```

In [18]:

```
##############################################################################

## 4. Naïve Bayesian.....

print("****************************************NAIVE BAYES CLASSIFIER*******

## "7" Attributes...
print("------------------7 Attributes--------------------------")
Berno= BernoulliNB(binarize=True)
Berno.fit(x_redwine_7_training, y_train.ravel())

y_pred = Berno.predict(x_redwine_7_testing)

## Measures List.....

print("Y_PREDICTION:(first 10 values)")
print(y_pred[:10])


cm1 = confusion_matrix(y_test,y_pred)
print('Confusion Matrix : \n', cm1)

total1=sum(sum(cm1))
accuracy1=(cm1[0,0]+cm1[1,1])/total1
print ('Accuracy : ', accuracy1)

precision1=(cm1[0,0])/(cm1[0,0]+cm1[1,0])
print ('Precision : ', precision1)


recall1=(cm1[0,0])/(cm1[0,0]+cm1[1,1])
print ('Recall : ', recall1)

f_measure=(2*recall1*precision1)/(precision1+recall1)
print('f measure: ', f_measure)

sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )

specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
```

```
****************************************NAIVE BAYES CLASSIFIER**************
**************************
------------------7 Attributes-------------------------
Y_PREDICTION:(first 10 values)
[0 0 0 0 0 0 0 0 0 0]
Confusion Matrix :
 [[424   2]
 [ 53   1]]
Accuracy :  0.8854166666666666
Precision :  0.8888888888888888
Recall :  0.9976470588235294
f measure:  0.9401330376940134
Sensitivity :  0.9953051643192489
Specificity :  0.018518518518518517
```

In [19]:

```
## "4" Attributes
print("------------------4 Attributes--------------------------")
Berno= BernoulliNB(binarize=True)
Berno.fit(x_redwine_4_training, y_train.ravel())
y_pred = Berno.predict(x_redwine_4_testing)




print("Y_PREDICTION:(first 10 values)")
print(y_pred[:10])


cm1 = confusion_matrix(y_test,y_pred)
print('Confusion Matrix : \n', cm1)

total1=sum(sum(cm1))
accuracy1=(cm1[0,0]+cm1[1,1])/total1
print ('Accuracy : ', accuracy1)

precision1=(cm1[0,0])/(cm1[0,0]+cm1[1,0])
print ('Precision : ', precision1)


recall1=(cm1[0,0])/(cm1[0,0]+cm1[1,1])
print ('Recall : ', recall1)

f_measure=(2*recall1*precision1)/(precision1+recall1)
print('f measure: ', f_measure)

sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )

specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
##########################################################################################
```

```
------------------4 Attributes--------------------------
Y_PREDICTION:(first 10 values)
[0 0 0 0 0 0 0 0 0 0]
Confusion Matrix :
 [[426   0]
 [ 54   0]]
Accuracy :  0.8875
Precision :  0.8875
Recall :  1.0
f measure:  0.9403973509933775
Sensitivity :  1.0
Specificity :  0.0
```

In [ ]:

In [ ]: