In [1]:
```
##Linear Regression Classifier

#importing all the packages
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,r2_score
from sklearn.metrics import confusion_matrix
from sklearn import svm
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB


import os

#getting the directory
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')

#declaring features
fields=["fixed acidity","volatile acidity","citric acid","residual sugar","ch
fields1=["fixed acidity","volatile acidity","citric acid","residual sugar","c
```

In [2]:
```
#X_TRAINING DATASET
train_wine = pd.read_csv(Dataset.train, delimiter=';', header=None, skiprows=

x_train=train_wine[fields1]

print("X_TRAINING DATA:")
x_train.head(5)
```

Out[2]:

X_TRAINING DATA:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	10.6	0.28	0.39	15.5	0.069	6.0	23.0	1.0026	3.12	0.66	9.2
1	9.4	0.30	0.56	2.8	0.080	6.0	17.0	0.9964	3.15	0.92	11.7
2	10.6	0.36	0.59	2.2	0.152	6.0	18.0	0.9986	3.04	1.05	9.4
3	10.6	0.36	0.60	2.2	0.152	7.0	18.0	0.9986	3.04	1.06	9.4
4	10.6	0.44	0.68	4.1	0.114	6.0	24.0	0.9970	3.06	0.66	13.4

In [3]:
```
print("**********************************************************************
# CLASSIFING quality 0-bad AND 1-good and forming Y_TRAINING DATASET
y_train = train_wine["quality"].apply(lambda  q:0 if q<7 else 1)
```

```
print("Y_TRAINING DATA:")
y_train.head(5)
```

```
**************************************************************************
**********************
Y_TRAINING DATA:
```

Out[3]:
```
0    0
1    1
2    0
3    0
4    0
Name: quality, dtype: int64
```

In [4]:
```
#similarly for testing data

#X_TESTING DATASET
test_wine = pd.read_csv(Dataset.test, delimiter=';', header=None, skiprows=1,
x_test=test_wine[fields1]
print("X_TESTING DATA:")
x_test.head(5)
```

```
X_TESTING DATA:
```

Out[4]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.70	0.00	1.9	0.076	11.0	34	0.9978	3.51	0.56	9.4
1	7.8	0.88	0.00	2.6	0.098	25.0	67	0.9968	3.20	0.68	9.8
2	7.8	0.76	0.04	2.3	0.092	15.0	54	0.9970	3.26	0.65	9.8
3	11.2	0.28	0.56	1.9	0.075	17.0	60	0.9980	3.16	0.58	9.8
4	7.4	0.70	0.00	1.9	0.076	11.0	34	0.9978	3.51	0.56	9.4

In [5]:
```
print("**************************************************************************

#Y_TESTING DATASET
y_test = test_wine["quality"].apply(lambda  q:0 if q<7 else 1)
print("Y_TESTING DATA:")
y_test.head(10)
```

```
**************************************************************************
**************************
Y_TESTING DATA:
```

Out[5]:
```
0    0
1    0
2    0
3    0
4    0
5    0
6    0
7    1
8    1
9    0
Name: quality, dtype: int64
```

In [17]:
```
#declaring linear regression model

print("*********************************LINEAR REGRESSION CLASSIFIER*********
l_r= LinearRegression()

# training the model with X_TRAINING AND Y_TRAINING
l_r.fit(x_train, y_train)
```

```
#Getting Y_PREDICTION
pred1=l_r.predict(x_test)

print("Y_PREDICTION DATA:(first 10 values)")
pred1[:10]
```

```
**********LINEAR REGRESSION CLASSIFIER**********
**********
Y_PREDICTION DATA:(first 10 values)
```

Out[17]:
```
array([-0.12480267, -0.04486957, -0.02591986,  0.17223449, -0.12480267,
       -0.12215259, -0.07652599,  0.01531073, -0.01059442,  0.22524849])
```

In [18]:
```
#MEAN OF THE Y_PREDICTION
mid = np.mean(pred1)

print("mean: ", mid)
```

```
mean:  0.10580178551233062
```

In [19]:
```
pred= []


#Classifing y_prediction based on the mean value
for p in pred1 :
    if p < mid:
        pred.append(0)
    elif p > mid:
        pred.append(1)

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

```
Y_PREDICTION DATA after classificaion:(first 10 values)
[0, 0, 0, 1, 0, 0, 0, 0, 0, 1]
```

In [20]:
```
#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)
```

```
Confusion Matrix :
 [[280 146]
 [  3  51]]
```

```
Accuracy :  0.6895833333333333
Precision :  0.9893992932862191
Recall :  0.8459214501510574
f measure:  0.9120521172638437
Sensitivity :  0.6572769953051644
Specificity :  0.9444444444444444
```

In [9]:
```
##Logistic Regression Classifier

print("*****************************LOGISTIC REGRESSION CLASSIFIER**********
#declaring logistic regression model
logisticRegr = LogisticRegression()

#training the model with X_TRAINING AND Y_TRAINING
logisticRegr.fit(x_train,y_train)

#Y_PREDICTION
pred=logisticRegr.predict(x_test)

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

```
*****************************LOGISTIC REGRESSION CLASSIFIER****************
**********************
Y_PREDICTION:(first 10 values)
[0 0 0 0 0 0 0 0 0 0]
/home/thrishik/.local/lib/python3.6/site-packages/sklearn/linear_model/_logis
tic.py:764: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

In [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)
```

```
Confusion Matrix :
 [[414  12]
 [ 43  11]]
Accuracy :  0.8854166666666666
Precision :  0.9059080962800875
Recall :  0.9741176470588235
f measure:  0.9387755102040816
```

```
Sensitivity :  0.971830985915493
Specificity :  0.2037037037037037
```

In [11]:
```
##SVM Classifier
print("**************************************SVM CLASSIFIER**************
#declaring svm model
cls=svm.SVC(kernel="linear")

#training the model with X_TRAINING AND Y_TRAINING
cls.fit(x_train,y_train)

#Y_PREDICTION
pred=cls.predict(x_test)


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

```
**************************************SVM CLASSIFIER**********************
************************
Y_PREDICTION:(first 10 values)
[0 0 0 0 0 0 0 0 0 0]
```

In [21]:
```
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)
```

```
Confusion Matrix :
 [[280 146]
 [  3  51]]
Accuracy :  0.6895833333333333
Precision :  0.9893992932862191
Recall :  0.8459214501510574
f measure:  0.9120521172638437
Sensitivity :  0.6572769953051644
Specificity :  0.9444444444444444
```

In [22]:
```
##Naive Bayes Classifier

print("**************************************NAIVE BAYES CLASSIFIER*******
#declaring Naive Bayes model
BernNB= BernoulliNB(binarize=True)

#training the model with X_TRAINING AND Y_TRAINING
BernNB.fit(x_train,y_train)
print(BernNB)
```

```
#Y_PREDICTION
y_pred=BernNB.predict(x_test)


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

```
****************************************NAIVE BAYES CLASSIFIER**************
**************************
BernoulliNB(binarize=True)
Y_PREDICTION:(first 10 values)
[0 0 0 0 0 0 0 0 0 0]
```

In [23]:

```
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)
```

```
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 [ ]: