****

**CS 634 Data Mining - Fall 2021**

**Professor Yasser Abduallah**

**Final term Project Report (Option 1)**

**Supervised Data Mining (Classification)**

**And Evaluations**

**Nishi Rani (nr36@njit.edu)**

**NJIT ID : 31524700**

**Contents:**

Description…………………………………………………………3

Naïve Bayes Model…………………………………………………6

Random Forest Model…………………………………………….12

LSTM Model………………………………………………………18

Complete Source Code…………………………………………….27

Result table………………………………………………………....37

Github Link………………………………………………………...37

Comparison/Discussion……………………………………………38

Conclusion…………………………………………………………..39

**Option 1:**

2 Machine Learning Algorithms selected for supervised classification were :

1. Naïve Bayes
2. Random Forest

1 Deep learning algorithm selected:

1. LSTM (Long Short Term Memory)

Evaluation metrics used were :

TPR, FPR, Recall, Precision,

Data was taken from <https://archive.ics.uci.edu/ml/datasets/Wine>

For Tensorflow installation I used : !pip install tensorflow in jupyter notebook.

Data was available in.data format I read the binary file using below python command as:

Code:

file\_handler=open('wine.data','rb')

lines\_array = file\_handler.readlines()

for l in lines\_array:

print(l)

Sample output:

b'1,14.23,1.71,2.43,15.6,127,2.8,3.06,.28,2.29,5.64,1.04,3.92,1065\n'

b'1,13.2,1.78,2.14,11.2,100,2.65,2.76,.26,1.28,4.38,1.05,3.4,1050\n'

b'1,13.16,2.36,2.67,18.6,101,2.8,3.24,.3,2.81,5.68,1.03,3.17,1185\n'

b'1,14.37,1.95,2.5,16.8,113,3.85,3.49,.24,2.18,7.8,.86,3.45,1480\n'

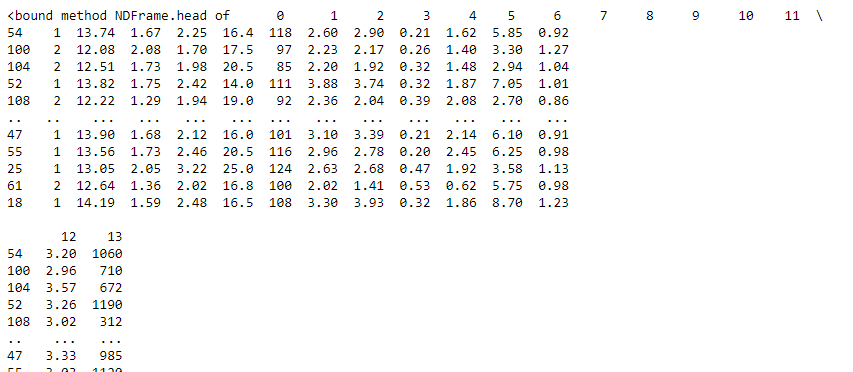
b'1,13.24,2.59,2.87,21,118,2.8,2.69,.39,1.82,4.32,1.04,2.93,735\n'

Saved the data file as .csv using notepad++ to easily access the rows and columns using pandas dataframe. Moreover Wine.data had 3 labels for which multi-label classification/ confusion matrix was required which was bit too complex to handle for each fold of cross validation, So with the suggestion from professor, I took the data for only 2 labels i.e. to go ahead with the binary classification.

data=pd.read\_csv('wine.csv',header=None)

df=data.sample(frac=1) # To shuffle the data I have used the sample function.

print(df.head)



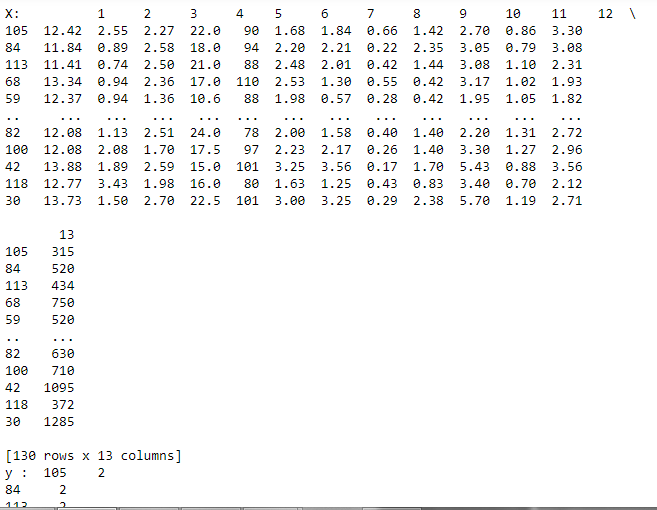
Next, separate the features and labels, There are overall 14 columns, first column represents the quality of wine i.e. “Label” with index 0. And rest of the columns are “features” which helps in predicting the label i.e. index 1 to index 13.

labels=df.iloc[:,0]

features=df.iloc[:,1:14]

X=features

y=labels



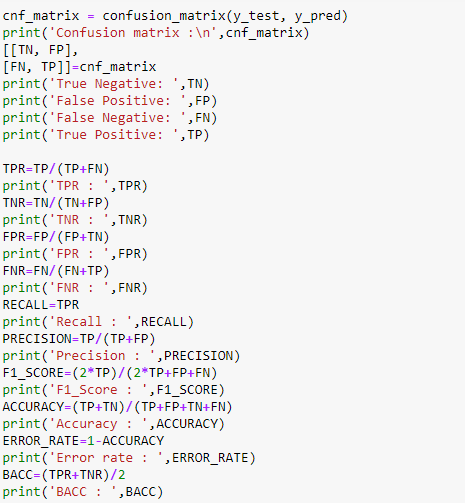
Next step was to import all the necessary python libraries and implementing the first model for each fold of k-fold cross validation:

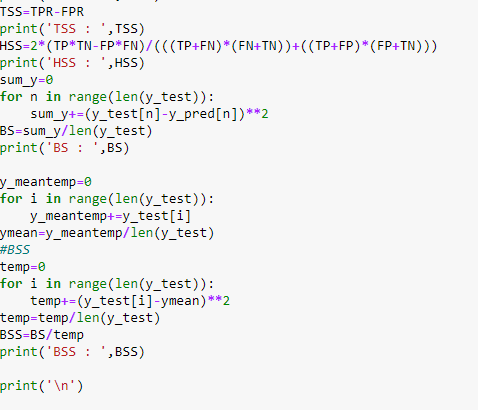
I have used k=10.

Below is the snapshot consisting of importing all the essential libraries and implementing the 1st model

**Model 1 : Naïve Bayes**







Output Snapshot:



Complete Output:

Accuracy achieved by Naive bayes in 1st fold: 1.0

Confusion matrix :

[[6 0]

[0 7]]

True Negative: 6

False Positive: 0

False Negative: 0

True Positive: 7

TPR : 1.0

TNR : 1.0

FPR : 0.0

FNR : 0.0

Recall : 1.0

Precision : 1.0

F1\_Score : 1.0

Accuracy : 1.0

Error rate : 0.0

BACC : 1.0

TSS : 1.0

HSS : 1.0

BS : 0.0

BSS : 0.0

Accuracy achieved by Naive bayes in 2nd fold: 1.0

Confusion matrix :

[[9 0]

[0 4]]

True Negative: 9

False Positive: 0

False Negative: 0

True Positive: 4

TPR : 1.0

TNR : 1.0

FPR : 0.0

FNR : 0.0

Recall : 1.0

Precision : 1.0

F1\_Score : 1.0

Accuracy : 1.0

Error rate : 0.0

BACC : 1.0

TSS : 1.0

HSS : 1.0

BS : 0.0

BSS : 0.0

Accuracy achieved by Naive bayes in 3rd fold: 0.9230769230769231

Confusion matrix :

[[4 0]

[1 8]]

True Negative: 4

False Positive: 0

False Negative: 1

True Positive: 8

TPR : 0.8888888888888888

TNR : 1.0

FPR : 0.0

FNR : 0.1111111111111111

Recall : 0.8888888888888888

Precision : 1.0

F1\_Score : 0.9411764705882353

Accuracy : 0.9230769230769231

Error rate : 0.07692307692307687

BACC : 0.9444444444444444

TSS : 0.8888888888888888

HSS : 0.8311688311688312

BS : 0.07692307692307693

BSS : 0.3611111111111111

Accuracy achieved by Naive bayes in 4th fold: 0.9230769230769231

Confusion matrix :

[[7 1]

[0 5]]

True Negative: 7

False Positive: 1

False Negative: 0

True Positive: 5

TPR : 1.0

TNR : 0.875

FPR : 0.125

FNR : 0.0

Recall : 1.0

Precision : 0.8333333333333334

F1\_Score : 0.9090909090909091

Accuracy : 0.9230769230769231

Error rate : 0.07692307692307687

BACC : 0.9375

TSS : 0.875

HSS : 0.8433734939759037

BS : 0.07692307692307693

BSS : 0.3249999999999999

Accuracy achieved by Naive bayes in 5th fold: 1.0

Confusion matrix :

[[4 0]

[0 9]]

True Negative: 4

False Positive: 0

False Negative: 0

True Positive: 9

TPR : 1.0

TNR : 1.0

FPR : 0.0

FNR : 0.0

Recall : 1.0

Precision : 1.0

F1\_Score : 1.0

Accuracy : 1.0

Error rate : 0.0

BACC : 1.0

TSS : 1.0

HSS : 1.0

BS : 0.0

BSS : 0.0

Accuracy achieved by Naive bayes in 6th fold: 1.0

Confusion matrix :

[[6 0]

[0 7]]

True Negative: 6

False Positive: 0

False Negative: 0

True Positive: 7

TPR : 1.0

TNR : 1.0

FPR : 0.0

FNR : 0.0

Recall : 1.0

Precision : 1.0

F1\_Score : 1.0

Accuracy : 1.0

Error rate : 0.0

BACC : 1.0

TSS : 1.0

HSS : 1.0

BS : 0.0

BSS : 0.0

Accuracy achieved by Naive bayes in 7th fold: 0.9230769230769231

Confusion matrix :

[[4 0]

[1 8]]

True Negative: 4

False Positive: 0

False Negative: 1

True Positive: 8

TPR : 0.8888888888888888

TNR : 1.0

FPR : 0.0

FNR : 0.1111111111111111

Recall : 0.8888888888888888

Precision : 1.0

F1\_Score : 0.9411764705882353

Accuracy : 0.9230769230769231

Error rate : 0.07692307692307687

BACC : 0.9444444444444444

TSS : 0.8888888888888888

HSS : 0.8311688311688312

BS : 0.07692307692307693

BSS : 0.3611111111111111

Accuracy achieved by Naive bayes in 8th fold: 1.0

Confusion matrix :

[[6 0]

[0 7]]

True Negative: 6

False Positive: 0

False Negative: 0

True Positive: 7

TPR : 1.0

TNR : 1.0

FPR : 0.0

FNR : 0.0

Recall : 1.0

Precision : 1.0

F1\_Score : 1.0

Accuracy : 1.0

Error rate : 0.0

BACC : 1.0

TSS : 1.0

HSS : 1.0

BS : 0.0

BSS : 0.0

Accuracy achieved by Naive bayes in 9th fold: 1.0

Confusion matrix :

[[5 0]

[0 8]]

True Negative: 5

False Positive: 0

False Negative: 0

True Positive: 8

TPR : 1.0

TNR : 1.0

FPR : 0.0

FNR : 0.0

Recall : 1.0

Precision : 1.0

F1\_Score : 1.0

Accuracy : 1.0

Error rate : 0.0

BACC : 1.0

TSS : 1.0

HSS : 1.0

BS : 0.0

BSS : 0.0

Accuracy achieved by Naive bayes in 10th fold: 0.9230769230769231

Confusion matrix :

[[6 1]

[0 6]]

True Negative: 6

False Positive: 1

False Negative: 0

True Positive: 6

TPR : 1.0

TNR : 0.8571428571428571

FPR : 0.14285714285714285

FNR : 0.0

Recall : 1.0

Precision : 0.8571428571428571

F1\_Score : 0.9230769230769231

Accuracy : 0.9230769230769231

Error rate : 0.07692307692307687

BACC : 0.9285714285714286

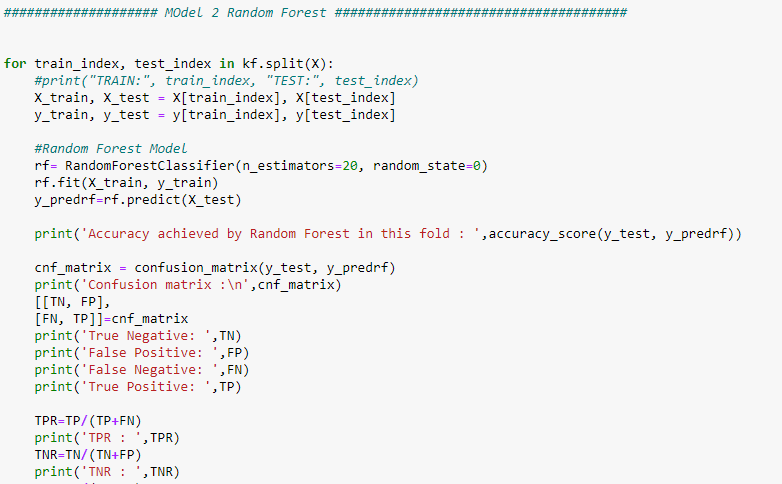
TSS : 0.8571428571428572

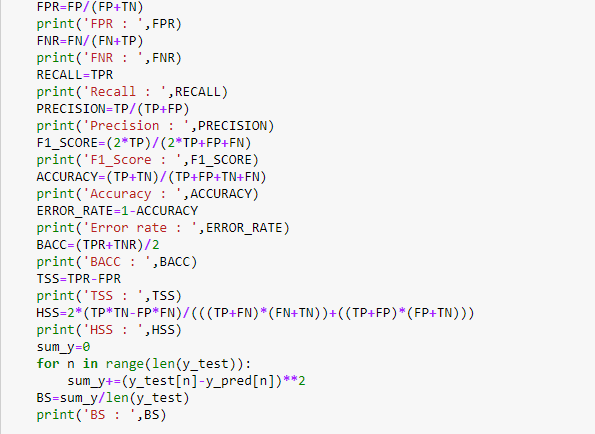
HSS : 0.8470588235294118

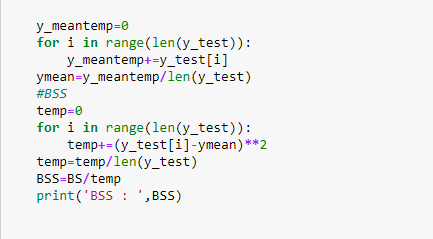
BS : 0.07692307692307693

BSS : 0.30952380952380965

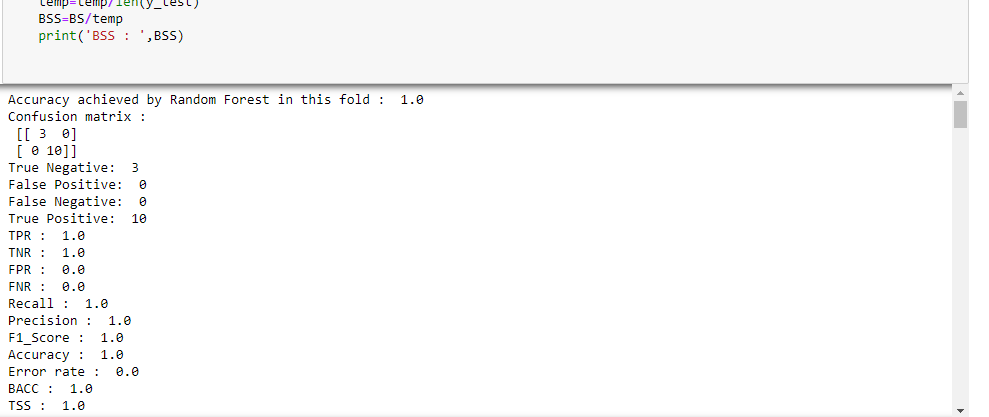
**Model 2: Random Forest**

****

****

****

Output Snapshot:



Output:

Accuracy achieved by Random Forest in 1st fold : 1.0

Confusion matrix :

[[6 0]

[0 7]]

True Negative: 6

False Positive: 0

False Negative: 0

True Positive: 7

TPR : 1.0

TNR : 1.0

FPR : 0.0

FNR : 0.0

Recall : 1.0

Precision : 1.0

F1\_Score : 1.0

Accuracy : 1.0

Error rate : 0.0

BACC : 1.0

TSS : 1.0

HSS : 1.0

BS : 0.6153846153846154

BSS : 2.476190476190477

Accuracy achieved by Random Forest in 2nd fold : 1.0

Confusion matrix :

[[9 0]

[0 4]]

True Negative: 9

False Positive: 0

False Negative: 0

True Positive: 4

TPR : 1.0

TNR : 1.0

FPR : 0.0

FNR : 0.0

Recall : 1.0

Precision : 1.0

F1\_Score : 1.0

Accuracy : 1.0

Error rate : 0.0

BACC : 1.0

TSS : 1.0

HSS : 1.0

BS : 0.38461538461538464

BSS : 1.8055555555555556

Accuracy achieved by Random Forest in 3rd fold : 0.9230769230769231

Confusion matrix :

[[4 0]

[1 8]]

True Negative: 4

False Positive: 0

False Negative: 1

True Positive: 8

TPR : 0.8888888888888888

TNR : 1.0

FPR : 0.0

FNR : 0.1111111111111111

Recall : 0.8888888888888888

Precision : 1.0

F1\_Score : 0.9411764705882353

Accuracy : 0.9230769230769231

Error rate : 0.07692307692307687

BACC : 0.9444444444444444

TSS : 0.8888888888888888

HSS : 0.8311688311688312

BS : 0.46153846153846156

BSS : 2.1666666666666665

Accuracy achieved by Random Forest in 4th fold : 0.9230769230769231

Confusion matrix :

[[7 1]

[0 5]]

True Negative: 7

False Positive: 1

False Negative: 0

True Positive: 5

TPR : 1.0

TNR : 0.875

FPR : 0.125

FNR : 0.0

Recall : 1.0

Precision : 0.8333333333333334

F1\_Score : 0.9090909090909091

Accuracy : 0.9230769230769231

Error rate : 0.07692307692307687

BACC : 0.9375

TSS : 0.875

HSS : 0.8433734939759037

BS : 0.3076923076923077

BSS : 1.2999999999999996

Accuracy achieved by Random Forest in 5th fold : 1.0

Confusion matrix :

[[4 0]

[0 9]]

True Negative: 4

False Positive: 0

False Negative: 0

True Positive: 9

TPR : 1.0

TNR : 1.0

FPR : 0.0

FNR : 0.0

Recall : 1.0

Precision : 1.0

F1\_Score : 1.0

Accuracy : 1.0

Error rate : 0.0

BACC : 1.0

TSS : 1.0

HSS : 1.0

BS : 0.46153846153846156

BSS : 2.1666666666666665

Accuracy achieved by Random Forest in 6th fold : 1.0

Confusion matrix :

[[6 0]

[0 7]]

True Negative: 6

False Positive: 0

False Negative: 0

True Positive: 7

TPR : 1.0

TNR : 1.0

FPR : 0.0

FNR : 0.0

Recall : 1.0

Precision : 1.0

F1\_Score : 1.0

Accuracy : 1.0

Error rate : 0.0

BACC : 1.0

TSS : 1.0

HSS : 1.0

BS : 0.3076923076923077

BSS : 1.2380952380952384

Accuracy achieved by Random Forest in 7th fold : 0.9230769230769231

Confusion matrix :

[[4 0]

[1 8]]

True Negative: 4

False Positive: 0

False Negative: 1

True Positive: 8

TPR : 0.8888888888888888

TNR : 1.0

FPR : 0.0

FNR : 0.1111111111111111

Recall : 0.8888888888888888

Precision : 1.0

F1\_Score : 0.9411764705882353

Accuracy : 0.9230769230769231

Error rate : 0.07692307692307687

BACC : 0.9444444444444444

TSS : 0.8888888888888888

HSS : 0.8311688311688312

BS : 0.3076923076923077

BSS : 1.4444444444444444

Accuracy achieved by Random Forest in 8th fold : 1.0

Confusion matrix :

[[6 0]

[0 7]]

True Negative: 6

False Positive: 0

False Negative: 0

True Positive: 7

TPR : 1.0

TNR : 1.0

FPR : 0.0

FNR : 0.0

Recall : 1.0

Precision : 1.0

F1\_Score : 1.0

Accuracy : 1.0

Error rate : 0.0

BACC : 1.0

TSS : 1.0

HSS : 1.0

BS : 0.46153846153846156

BSS : 1.8571428571428577

Accuracy achieved by Random Forest in 9th fold : 1.0

Confusion matrix :

[[5 0]

[0 8]]

True Negative: 5

False Positive: 0

False Negative: 0

True Positive: 8

TPR : 1.0

TNR : 1.0

FPR : 0.0

FNR : 0.0

Recall : 1.0

Precision : 1.0

F1\_Score : 1.0

Accuracy : 1.0

Error rate : 0.0

BACC : 1.0

TSS : 1.0

HSS : 1.0

BS : 0.5384615384615384

BSS : 2.2749999999999995

Accuracy achieved by Random Forest in 10th fold : 1.0

Confusion matrix :

[[7 0]

[0 6]]

True Negative: 7

False Positive: 0

False Negative: 0

True Positive: 6

TPR : 1.0

TNR : 1.0

FPR : 0.0

FNR : 0.0

Recall : 1.0

Precision : 1.0

F1\_Score : 1.0

Accuracy : 1.0

Error rate : 0.0

BACC : 1.0

TSS : 1.0

HSS : 1.0

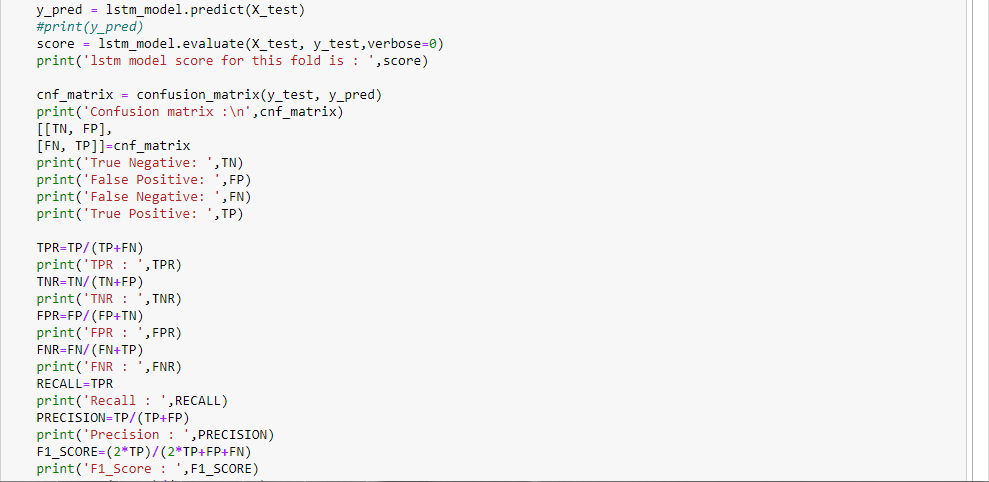
BS : 0.07692307692307693

BSS : 0.30952380952380965

**Model 3 : LSTM**

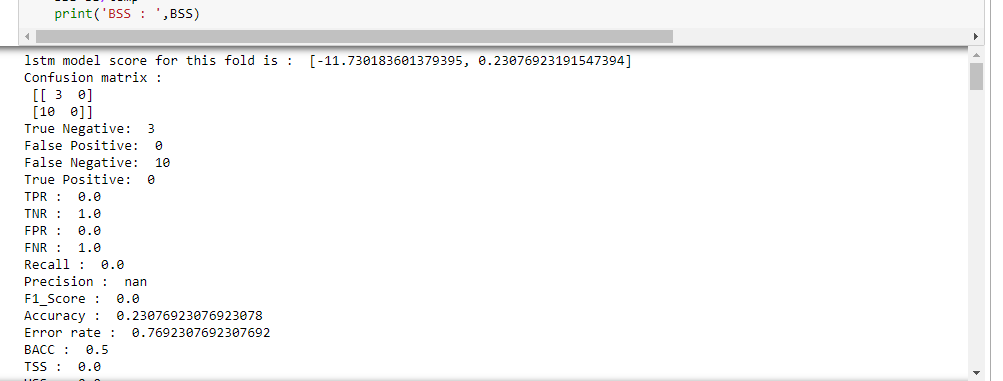
LSTM-Long Short Term Memory







Snapshot of output:



Complete Output:

Evaluation metrics for lstm model is :

[-5.865091800689697, 0.6153846383094788]

Confusion matrix :

[[8 0]

[5 0]]

True Negative: 8

False Positive: 0

False Negative: 5

True Positive: 0

TPR : 0.0

TNR : 1.0

FPR : 0.0

FNR : 1.0

Recall : 0.0

Precision : nan

F1\_Score : 0.0

Accuracy : 0.6153846153846154

Error rate : 0.3846153846153846

BACC : 0.5

TSS : 0.0

HSS : 0.0

BS : [0.3846154]

BSS : [1.625]

Evaluation metrics for lstm model is :

[-9.384146690368652, 0.38461539149284363]

Confusion matrix :

[[5 0]

[8 0]]

True Negative: 5

False Positive: 0

False Negative: 8

True Positive: 0

TPR : 0.0

TNR : 1.0

FPR : 0.0

FNR : 1.0

Recall : 0.0

Precision : nan

F1\_Score : 0.0

Accuracy : 0.38461538461538464

Error rate : 0.6153846153846154

BACC : 0.5

TSS : 0.0

HSS : 0.0

BS : [0.61538464]

BSS : [2.6000001]

Evaluation metrics for lstm model is :

[-10.557165145874023, 0.3076923191547394]

Confusion matrix :

[[4 0]

[9 0]]

True Negative: 4

False Positive: 0

False Negative: 9

True Positive: 0

TPR : 0.0

TNR : 1.0

FPR : 0.0

FNR : 1.0

Recall : 0.0

Precision : nan

F1\_Score : 0.0

Accuracy : 0.3076923076923077

Error rate : 0.6923076923076923

BACC : 0.5

TSS : 0.0

HSS : 0.0

BS : [0.6923077]

BSS : [3.2500002]

Evaluation metrics for lstm model is :

[-7.03810977935791, 0.5384615659713745]

Confusion matrix :

[[7 0]

[6 0]]

True Negative: 7

False Positive: 0

False Negative: 6

True Positive: 0

TPR : 0.0

TNR : 1.0

FPR : 0.0

FNR : 1.0

Recall : 0.0

Precision : nan

F1\_Score : 0.0

Accuracy : 0.5384615384615384

Error rate : 0.46153846153846156

BACC : 0.5

TSS : 0.0

HSS : 0.0

BS : [0.46153846]

BSS : [1.8571428]

Evaluation metrics for lstm model is :

[-9.384146690368652, 0.38461539149284363]

Confusion matrix :

[[5 0]

[8 0]]

True Negative: 5

False Positive: 0

False Negative: 8

True Positive: 0

TPR : 0.0

TNR : 1.0

FPR : 0.0

FNR : 1.0

Recall : 0.0

Precision : nan

F1\_Score : 0.0

Accuracy : 0.38461538461538464

Error rate : 0.6153846153846154

BACC : 0.5

TSS : 0.0

HSS : 0.0

BS : [0.61538464]

BSS : [2.6000001]

Evaluation metrics for lstm model is :

[-7.03810977935791, 0.5384615659713745]

Confusion matrix :

[[7 0]

[6 0]]

True Negative: 7

False Positive: 0

False Negative: 6

True Positive: 0

TPR : 0.0

TNR : 1.0

FPR : 0.0

FNR : 1.0

Recall : 0.0

Precision : nan

F1\_Score : 0.0

Accuracy : 0.5384615384615384

Error rate : 0.46153846153846156

BACC : 0.5

TSS : 0.0

HSS : 0.0

BS : [0.46153846]

BSS : [1.8571428]

Evaluation metrics for lstm model is :

[-7.03810977935791, 0.5384615659713745]

Confusion matrix :

[[7 0]

[6 0]]

True Negative: 7

False Positive: 0

False Negative: 6

True Positive: 0

TPR : 0.0

TNR : 1.0

FPR : 0.0

FNR : 1.0

Recall : 0.0

Precision : nan

F1\_Score : 0.0

Accuracy : 0.5384615384615384

Error rate : 0.46153846153846156

BACC : 0.5

TSS : 0.0

HSS : 0.0

BS : [0.46153846]

BSS : [1.8571428]

Evaluation metrics for lstm model is :

[-8.211128234863281, 0.4615384638309479]

Confusion matrix :

[[6 0]

[7 0]]

True Negative: 6

False Positive: 0

False Negative: 7

True Positive: 0

TPR : 0.0

TNR : 1.0

FPR : 0.0

FNR : 1.0

Recall : 0.0

Precision : nan

F1\_Score : 0.0

Accuracy : 0.46153846153846156

Error rate : 0.5384615384615384

BACC : 0.5

TSS : 0.0

HSS : 0.0

BS : [0.53846157]

BSS : [2.1666667]

Evaluation metrics for lstm model is :

[-8.211128234863281, 0.4615384638309479]

Confusion matrix :

[[6 0]

[7 0]]

True Negative: 6

False Positive: 0

False Negative: 7

True Positive: 0

TPR : 0.0

TNR : 1.0

FPR : 0.0

FNR : 1.0

Recall : 0.0

Precision : nan

F1\_Score : 0.0

Accuracy : 0.46153846153846156

Error rate : 0.5384615384615384

BACC : 0.5

TSS : 0.0

HSS : 0.0

BS : [0.53846157]

BSS : [2.1666667]

Evaluation metrics for lstm model is :

[-10.557165145874023, 0.3076923191547394]

Confusion matrix :

[[4 0]

[9 0]]

True Negative: 4

False Positive: 0

False Negative: 9

True Positive: 0

TPR : 0.0

TNR : 1.0

FPR : 0.0

FNR : 1.0

Recall : 0.0

Precision : nan

F1\_Score : 0.0

Accuracy : 0.3076923076923077

Error rate : 0.6923076923076923

BACC : 0.5

TSS : 0.0

HSS : 0.0

BS : [0.6923077]

BSS : [3.2500002]

**Evaluation metrics for the average 10 fold cross validation using LSTM :**

TN\_AVG : 5.9

TP\_AVG : 0.0

FN\_AVG : 7.1

FP\_AVG : 0.0

TPR : 0.0

TNR : 1.0

FPR : 0.0

FNR : 1.0

Recall : 0.0

Precision : nan

F1\_Score : 0.0

Accuracy : 0.4538461538461539

Error rate : 0.5461538461538461

BACC : 0.5

TSS : 0.0

HSS : 0.0

BS : [0.6923077]

BSS : [3.2500002]

**Complete Source code:**

import pandas as pd

import numpy as np

import tensorflow as tf

from tensorflow.python.ops.math\_ops import reduce\_prod

data=pd.read\_csv('wine.csv',header=None)

df=data.sample(frac=1)

print(df.head)

labels=df.iloc[:,0]

features=df.iloc[:,1:14]

X=features

y=labels

print('X: ',X)

print('y : ',y)

def evaluation\_metrics(TP,TN,FP,FN):

TPR=TP/(TP+FN)

print('TPR : ',TPR)

TNR=TN/(TN+FP)

print('TNR : ',TNR)

FPR=FP/(FP+TN)

print('FPR : ',FPR)

FNR=FN/(FN+TP)

print('FNR : ',FNR)

RECALL=TPR

print('Recall : ',RECALL)

PRECISION=TP/(TP+FP)

print('Precision : ',PRECISION)

F1\_SCORE=(2\*TP)/(2\*TP+FP+FN)

print('F1\_Score : ',F1\_SCORE)

ACCURACY=(TP+TN)/(TP+FP+TN+FN)

print('Accuracy : ',ACCURACY)

ERROR\_RATE=1-ACCURACY

print('Error rate : ',ERROR\_RATE)

BACC=(TPR+TNR)/2

print('BACC : ',BACC)

TSS=TPR-FPR

print('TSS : ',TSS)

HSS=2\*(TP\*TN-FP\*FN)/(((TP+FN)\*(FN+TN))+((TP+FP)\*(FP+TN)))

print('HSS : ',HSS)

sum\_y=0

for n in range(len(y\_test)):

sum\_y+=(y\_test[n]-y\_pred[n])\*\*2

BS=sum\_y/len(y\_test)

print('BS : ',BS)

y\_meantemp=0

for i in range(len(y\_test)):

y\_meantemp+=y\_test[i]

ymean=y\_meantemp/len(y\_test)

#BSS

temp=0

for i in range(len(y\_test)):

temp+=(y\_test[i]-ymean)\*\*2

temp=temp/len(y\_test)

BSS=BS/temp

print('BSS : ',BSS)

print('\n')

X=np.array(X)

y=np.array(y)

from sklearn.model\_selection import KFold

from sklearn.model\_selection import cross\_val\_score

from sklearn.naive\_bayes import GaussianNB

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import multilabel\_confusion\_matrix

kf = KFold(n\_splits=10)

################# Model 1 Naive Bayes #####################

TN\_TOTAL=0

TP\_TOTAL=0

FP\_TOTAL=0

FN\_TOTAL=0

for train\_index, test\_index in kf.split(X):

#print("TRAIN:", train\_index, "TEST:", test\_index)

X\_train, X\_test = X[train\_index], X[test\_index]

y\_train, y\_test = y[train\_index], y[test\_index]

model = GaussianNB()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

print('Evaluation metrics for each fold :\n')

cnf\_matrix = confusion\_matrix(y\_test, y\_pred)

print('Confusion matrix :\n',cnf\_matrix)

[[TNs, FPs],

[FNs, TPs]]=cnf\_matrix

print('True Negative: ',TNs)

print('False Positive: ',FPs)

print('False Negative: ',FNs)

print('True Positive: ',TPs)

evaluation\_metrics(TPs,TNs,FPs,FNs)

TN\_TOTAL+=TNs

TP\_TOTAL+=TPs

FP\_TOTAL+=FPs

FN\_TOTAL+=FNs

TN=TN\_TOTAL/10

TP=TP\_TOTAL/10

FN=FN\_TOTAL/10

FP=FP\_TOTAL/10

print('Evaluation metrics for the average 10 fold cross validation using Naive Bayes : \n')

print('TN\_AVG : ',TN)

print('TP\_AVG : ',TP)

print('FN\_AVG : ',FN)

print('FP\_AVG : ',FP)

TPR=TP/(TP+FN)

print('TPR AVG : ',TPR)

TNR=TN/(TN+FP)

print('TNR AVG: ',TNR)

FPR=FP/(FP+TN)

print('FPR AVG: ',FPR)

FNR=FN/(FN+TP)

print('FNR AVG: ',FNR)

RECALL=TPR

print('Recall : ',RECALL)

PRECISION=TP/(TP+FP)

print('Precision : ',PRECISION)

F1\_SCORE=(2\*TP)/(2\*TP+FP+FN)

print('F1\_Score : ',F1\_SCORE)

ACCURACY=(TP+TN)/(TP+FP+TN+FN)

print('Accuracy : ',ACCURACY)

ERROR\_RATE=1-ACCURACY

print('Error rate : ',ERROR\_RATE)

BACC=(TPR+TNR)/2

print('BACC : ',BACC)

TSS=TPR-FPR

print('TSS : ',TSS)

HSS=2\*(TP\*TN-FP\*FN)/(((TP+FN)\*(FN+TN))+((TP+FP)\*(FP+TN)))

print('HSS : ',HSS)

sum\_y=0

for n in range(len(y\_test)):

sum\_y+=(y\_test[n]-y\_pred[n])\*\*2

BS=sum\_y/len(y\_test)

print('BS : ',BS)

y\_meantemp=0

for i in range(len(y\_test)):

y\_meantemp+=y\_test[i]

ymean=y\_meantemp/len(y\_test)

temp=0

for i in range(len(y\_test)):

temp+=(y\_test[i]-ymean)\*\*2

temp=temp/len(y\_test)

BSS=BS/temp

print('BSS : ',BSS)

print('\n')

df1=pd.DataFrame({"TP": TP,

"FP": FP,

"FN": FN,

"TN": TN,

"TPR":TPR,

"FPR":FPR,

"TNR":TNR,

"FNR":FNR,

"RECALL":RECALL,

'PRECISION':PRECISION,

'F1\_SCORE':F1\_SCORE,

'Accuracy':ACCURACY,

'Error rate':ERROR\_RATE,

'BACC':BACC,

'TSS':TSS,

'HSS':HSS,

'BS' :BS,

'BSS':BSS

},

index=["Naive Bayes"])

#################### MOdel 2 Random Forest #######################

TN\_TOTAL=0

TP\_TOTAL=0

FP\_TOTAL=0

FN\_TOTAL=0

for train\_index, test\_index in kf.split(X):

#print("TRAIN:", train\_index, "TEST:", test\_index)

X\_train, X\_test = X[train\_index], X[test\_index]

y\_train, y\_test = y[train\_index], y[test\_index]

rf= RandomForestClassifier(n\_estimators=20, random\_state=0)

rf.fit(X\_train, y\_train)

y\_predrf=rf.predict(X\_test)

print('Evaluation metrics achieved by Random Forest in each fold : ')

cnf\_matrix = confusion\_matrix(y\_test, y\_predrf)

print('Confusion matrix :\n',cnf\_matrix)

[[TNs, FPs],

[FNs, TPs]]=cnf\_matrix

print('True Negative: ',TNs)

print('False Positive: ',FPs)

print('False Negative: ',FNs)

print('True Positive: ',TPs)

evaluation\_metrics(TPs,TNs,FPs,FNs)

TN\_TOTAL+=TNs

TP\_TOTAL+=TPs

FP\_TOTAL+=FPs

FN\_TOTAL+=FNs

TN=TN\_TOTAL/10

TP=TP\_TOTAL/10

FN=FN\_TOTAL/10

FP=FP\_TOTAL/10

print('Evaluation metrics for the average 10 fold cross validation using Random Forest : \n')

print('TN\_AVG : ',TN)

print('TP\_AVG : ',TP)

print('FN\_AVG : ',FN)

print('FP\_AVG : ',FP)

TPR=TP/(TP+FN)

print('TPR AVG : ',TPR)

TNR=TN/(TN+FP)

print('TNR AVG: ',TNR)

FPR=FP/(FP+TN)

print('FPR AVG: ',FPR)

FNR=FN/(FN+TP)

print('FNR AVG: ',FNR)

RECALL=TPR

print('Recall : ',RECALL)

PRECISION=TP/(TP+FP)

print('Precision : ',PRECISION)

F1\_SCORE=(2\*TP)/(2\*TP+FP+FN)

print('F1\_Score : ',F1\_SCORE)

ACCURACY=(TP+TN)/(TP+FP+TN+FN)

print('Accuracy : ',ACCURACY)

ERROR\_RATE=1-ACCURACY

print('Error rate : ',ERROR\_RATE)

BACC=(TPR+TNR)/2

print('BACC : ',BACC)

TSS=TPR-FPR

print('TSS : ',TSS)

HSS=2\*(TP\*TN-FP\*FN)/(((TP+FN)\*(FN+TN))+((TP+FP)\*(FP+TN)))

print('HSS : ',HSS)

sum\_y=0

for n in range(len(y\_test)):

sum\_y+=(y\_test[n]-y\_pred[n])\*\*2

BS=sum\_y/len(y\_test)

print('BS : ',BS)

y\_meantemp=0

for i in range(len(y\_test)):

y\_meantemp+=y\_test[i]

ymean=y\_meantemp/len(y\_test)

temp=0

for i in range(len(y\_test)):

temp+=(y\_test[i]-ymean)\*\*2

temp=temp/len(y\_test)

BSS=BS/temp

print('BSS : ',BSS)

print('\n')

df2=pd.DataFrame({"TP": TP,

"FP": FP,

"FN": FN,

"TN": TN,

"TPR":TPR,

"FPR":FPR,

"TNR":TNR,

"FNR":FNR,

"RECALL":RECALL,

'PRECISION':PRECISION,

'F1\_SCORE':F1\_SCORE,

'Accuracy':ACCURACY,

'Error rate':ERROR\_RATE,

'BACC':BACC,

'TSS':TSS,

'HSS':HSS,

'BS' :BS,

'BSS':BSS

},

index=["Random Forest"])

import warnings

warnings.filterwarnings("ignore")

import os

os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '3'

try:

tf.compat.v1.logging.set\_verbosity(tf.compat.v1.logging.ERROR)

except Exception as e:

print('')

################# Model 3 LSTM (Deep Learning) #####################

TN\_TOTAL=0

TP\_TOTAL=0

FP\_TOTAL=0

FN\_TOTAL=0

for train\_index, test\_index in kf.split(X):

X\_train, X\_test = X[train\_index], X[test\_index]

y\_train, y\_test = y[train\_index], y[test\_index]

#reshape the data to match 3 dimension for LSTM layers.

X\_train = X\_train.reshape(X\_train.shape[0], X\_train.shape[1],1)

X\_test = X\_test.reshape(X\_test.shape[0], X\_test.shape[1],1)

# print('X\_train.shape:', X\_train.shape)

# print('y\_train.shape:', y\_train.shape)

# print('X\_test.shape:', X\_test.shape)

# print('y\_test.shape:', y\_test.shape)

lstm\_model = tf.keras.Sequential()

lstm\_model.add(tf.keras.layers.LSTM(64,return\_sequences=True, return\_state=False,input\_shape=(X\_test.shape[1],X\_test.shape[2]))) # Instruction: in case you change the data format

lstm\_model.add(tf.keras.layers.LSTM(64, return\_sequences=True, return\_state=False))

lstm\_model.add(tf.keras.layers.LSTM(64, return\_sequences=True, return\_state=False))

lstm\_model.add(tf.keras.layers.LSTM(64, return\_sequences=True, return\_state=False))

lstm\_model.add(tf.keras.layers.LSTM(64, return\_sequences=True, return\_state=False))

#when we return the sequence, we change the shape, so last layer should not return sequnce to the Dense layer

lstm\_model.add(tf.keras.layers.Flatten())

lstm\_model.add(tf.keras.layers.Dense(1, activation='sigmoid'))

# Compile the Model

optimizer = tf.keras.optimizers.Adam(learning\_rate=0.0001)

lstm\_model.compile(optimizer='adam', loss="binary\_crossentropy", metrics=['accuracy'])

#lstm\_model.summary()

lstm\_model.fit(X\_train, y\_train,epochs=100,batch\_size=1, verbose = 0)

#more epochs the better and we can add EarlyStopping in callbacks so that if the accuracy is not improving, it will stop.

y\_pred = lstm\_model.predict(X\_test)

#print(y\_pred)

score = lstm\_model.evaluate(X\_test, y\_test,verbose=0)

print('Evaluation metrics for lstm model is : \n',score)

cnf\_matrix = confusion\_matrix(y\_test, y\_pred)

print('Confusion matrix :\n',cnf\_matrix)

[[TNs, FPs],

[FNs, TPs]]=cnf\_matrix

print('True Negative: ',TNs)

print('False Positive: ',FPs)

print('False Negative: ',FNs)

print('True Positive: ',TPs)

evaluation\_metrics(TPs,TNs,FPs,FNs)

TN\_TOTAL+=TNs

TP\_TOTAL+=TPs

FP\_TOTAL+=FPs

FN\_TOTAL+=FNs

TN=TN\_TOTAL/10

TP=TP\_TOTAL/10

FN=FN\_TOTAL/10

FP=FP\_TOTAL/10

print('Evaluation metrics for the average 10 fold cross validation using LSTM : \n')

print('TN\_AVG : ',TN)

print('TP\_AVG : ',TP)

print('FN\_AVG : ',FN)

print('FP\_AVG : ',FP)

TPR=TP/(TP+FN)

print('TPR AVG : ',TPR)

TNR=TN/(TN+FP)

print('TNR AVG: ',TNR)

FPR=FP/(FP+TN)

print('FPR AVG: ',FPR)

FNR=FN/(FN+TP)

print('FNR AVG: ',FNR)

RECALL=TPR

print('Recall : ',RECALL)

PRECISION=TP/(TP+FP)

print('Precision : ',PRECISION)

F1\_SCORE=(2\*TP)/(2\*TP+FP+FN)

print('F1\_Score : ',F1\_SCORE)

ACCURACY=(TP+TN)/(TP+FP+TN+FN)

print('Accuracy : ',ACCURACY)

ERROR\_RATE=1-ACCURACY

print('Error rate : ',ERROR\_RATE)

BACC=(TPR+TNR)/2

print('BACC : ',BACC)

TSS=TPR-FPR

print('TSS : ',TSS)

HSS=2\*(TP\*TN-FP\*FN)/(((TP+FN)\*(FN+TN))+((TP+FP)\*(FP+TN)))

print('HSS : ',HSS)

sum\_y=0

for n in range(len(y\_test)):

sum\_y+=(y\_test[n]-y\_pred[n])\*\*2

BS=sum\_y/len(y\_test)

print('BS : ',BS)

y\_meantemp=0

for i in range(len(y\_test)):

y\_meantemp+=y\_test[i]

ymean=y\_meantemp/len(y\_test)

temp=0

for i in range(len(y\_test)):

temp+=(y\_test[i]-ymean)\*\*2

temp=temp/len(y\_test)

BSS=BS/temp

print('BSS : ',BSS)

print('\n')

df3=pd.DataFrame({"TP": TP,

"FP": FP,

"FN": FN,

"TN": TN,

"TPR":TPR,

"FPR":FPR,

"TNR":TNR,

"FNR":FNR,

"RECALL":RECALL,

'PRECISION':PRECISION,

'F1\_SCORE':F1\_SCORE,

'Accuracy':ACCURACY,

'Error rate':ERROR\_RATE,

'BACC':BACC,

'TSS':TSS,

'HSS':HSS,

'BS' :BS,

'BSS':BSS

},

index=["LSTM"])

# Creating Table to have all the evaluation metrics in one place.

frames=[df1,df2,df3]

result=pd.concat(frames)

result\_transpose=result.T

print(result\_transpose)

**Result Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| Naive Bayes Random Forest LSTM |  |  |  |
| TP 7.100000 7.000000 0.000000 |  |  |  |
| FP 0.200000 0.000000 0.000000 |  |  |  |
| FN 0.000000 0.100000 7.100000 |  |  |  |
| TN 5.700000 5.900000 5.900000 |  |  |  |
| TPR 1.000000 0.985915 0.000000 |  |  |  |
| FPR 0.033898 0.000000 0.000000 |  |  |  |
| TNR 0.966102 1.000000 1.000000 |  |  |  |
| FNR 0.000000 0.014085 1.000000 |  |  |  |
| RECALL 1.000000 0.985915 0.000000 |  |  |  |
| PRECISION 0.972603 1.000000 NaN |  |  |  |
| F1\_SCORE 0.986111 0.992908 0.000000 |  |  |  |
| Accuracy 0.984615 0.992308 0.453846 |  |  |  |
| Error rate 0.015385 0.007692 0.546154 |  |  |  |
| BACC 0.983051 0.992958 0.500000 |  |  |  |
| TSS 0.966102 0.985915 0.000000 |  |  |  |
| HSS 0.968877 0.984505 0.000000 |  |  |  |
| BS 0.076923 0.076923 0.538462 |  |  |  |
| BSS 0.309524 0.309524 2.166667 |  |  |  |

**Github Link**: <https://github.com/nr36/CS634-FinalProject>

**Comparison/Discussion:**

**Random Forest outperforms among the three models.**

Below are the references made while comparing the three model evaluation metrics:

* Data was almost balanced and no missing values were there that’s why accuracy and balanced accuracy i.e. BACC are almost same.
* The result from random forest and Naïve Bayes are close but few thing should be noted
  1. In Naïve Bayes model, In 10 folds there were 2 False positives noted i.e. there is a chance of 20% false positive values. For eg. The output label was 0 but predicted as 1.
  2. Whereas in Random forest model, Out of 10 folds only in one fold, one false negative was detected i.e. there is a chance of 10% false negative values For eg. The output label was 1 but predicted as 0.
  3. Random forest Model performed best out of the 3 models I have selected in all the aspects like accuracy, BACC, F1-score and more. The random Forest and Naïve bayes almost take similar amount of execution time.
  4. While using LSTM model, I have used LSTM with 4 hidden layers other than input and the output layers each with 64 hidden units and activation function as sigmoid and using Adam optimizer with learning rate 0.0001 and loss as binary\_crossentropy but the deep neural network couldn’t perform well with 100 epochs in each of the 10 folds of cross validation.

It got too slow in my laptop and I couldn’t use GPU as my laptop doesn’t support that. May be if we increase number of layers to some higher numbers It could have performed decently.

But with the result of LSTM we can conclude that:

No true positive or false positive values have been detected only true negative and false negatives labels were predicted. So it can be inferred that LSTM only has predictions with probability less than 0.5 for all the data so it considered predictions of all the data as 0.

LSTM didn’t perform well here.

* 1. So, For the given wine dataset I would preferably choose Random Forest over Naïve Bayes and LSTM.

**Conclusion:**

For the given wine dataset I would preferably choose Random Forest over Naïve Bayes and LSTM after comparing the evaluation metrics.