COURSE PROJECT REPORT

Stock Index Prediction using Naïve Bayesian, AdaBoost and SVM classifier

A picture containing food

Description automatically generated

CHEMENG 787: Machine Learning Classification Models

Submitted to: Dr. Jeff Foruna

Submitted by: Livanshu Kashyap #400286794 [kashyapl@mcmaster.ca](mailto:kashyapl@mcmaster.ca)

**Summary**

The classifiers used in this project are Gaussian Naïve Bayesian, AdaBoost and Support Vector Machine. The dataset used in the course project is taken from Kaggle.com and it is called “Stock Index Prediction” [1] uploaded by Jiang M.Q. The idea is to predict if the stock would go up or down by training classifiers. The features are taken from Yahoo finance and FREM by the uploader. The data has 3 files for 3 different stock exchanges i.e. NASDAQ, DOW JONES and S&P 500.

For this course project I used NASDAQ data which has total 2448 data points and 14 attributes. The 14 attributes are given:

1. Date - Date of the data point collected
2. Label - The target label i.e. Stock up = 1 & Stock down = 0.
3. Open - Opening price of the stock.
4. High - Highest stock price in the day
5. Low - Lowest stock price in the day
6. Close - Closing price of the stock
7. Volume - Volume of the stock
8. Interest rate - Interest rate set by market committee
9. Exchange rate - Exchange rate of currency
10. VIX - Volatility Index
11. Gold - Gold prices
12. Oil - Oil prices
13. TEDSpread - TED spread is difference between treasury bill & LIBOR
14. EFFR - Effective federal funds rate

The data was of .csv extension and imported to python code using *pandas.read\_csv*. For training classifiers, the Date attribute was removed. All other attributes have numerical values. As the data has 2448 data points, 75% of it i.e. 1836 data points are used for training and the rest 612 data points are used for testing. The data is split using *train\_test\_split* module of *sklearn* library. Different libraries were used for all the operations which includes training classifiers, confusion matrix and ROC.

The results of all classifiers were around 54% which we can say is Random Result i.e. if predicted stock would go up there is only 54% chance that it would go up which is almost the random probability. Even 6 years of data could not predict the next day’s value which clearly states that the stock results are completely random. Adding more attributes and data points might increase the accuracy a bit but not by a significant percentage.

For this data, Gaussian Naïve Bayesian classifier has the highest accuracy i.e. 54.9%.

1. **Computation times**

Computational time for all classifiers:

|  |  |  |  |
| --- | --- | --- | --- |
| **Times** | **G. Naïve Bayesian** | **AdaBoost** | **SVM** |
| **Training** | 0.031 s | 0.156 s | 1.516 s |
| **Testing** | 0.016 s | 0.000 s | 0.234 s |
| **Accuracy** | 54.9 % | 52.12 % | 53.92 % |

Time to search best estimators in SVM = 77.51 s

Time to run the whole code ≈ 85 s

Python version: 3.8

Platform: PyCharm

**2. Cross validation for parameter selection**

Cross validation for parameter selection is done using GridSearchCV which searches over the values specified (C and gamma). It combines with K-fold cross validation with a grid of parameters. By default it uses 3-Fold fit unless specified. Only C and gamma value grid is specified therefore it uses 3-Fold fit.

Code for CV:

print("Finding best parameters...")  
start\_SVM\_param = time.time()  
param\_grid = {'C': [0.25,0.5,1,10,1e2,1e3,1e4,1e5],  
 'gamma': [10,1,0.1,0.01,0.05,0.001,0.0001,0.00001], }  
clf = GridSearchCV(  
 SVC(kernel='rbf', class\_weight='balanced'), param\_grid  
)  
clf = clf.fit(X\_train, y\_train)  
end\_SVM\_param = time.time()  
print ("Found best parameters in %0.2f seconds" % (end\_SVM\_param - start\_SVM\_param))  
print("Best C value =", clf.best\_estimator\_.C)  
print("Best Gamma value =", clf.best\_estimator\_.gamma)  
print()

Output:

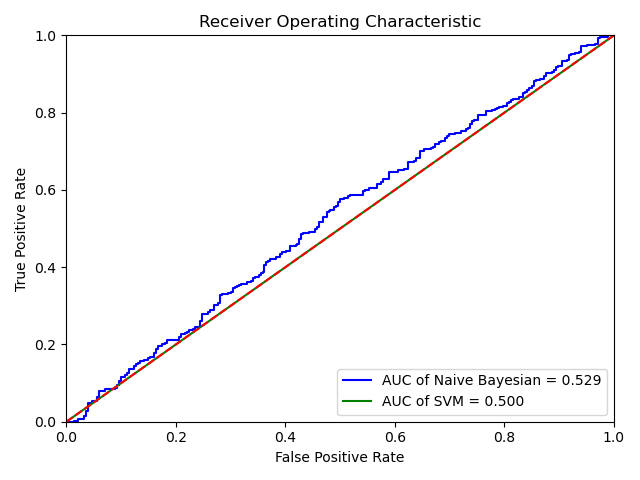
Finding best parameters...

Found best parameters in 77.51 seconds

Best C value = 1

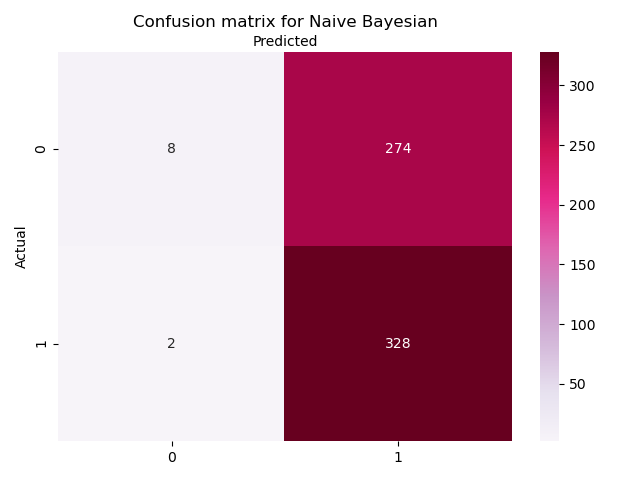
Best Gamma value = 10

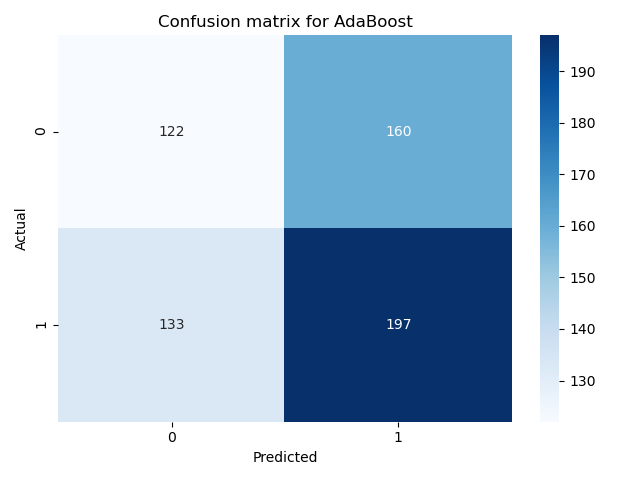
**3. ROC Curve**

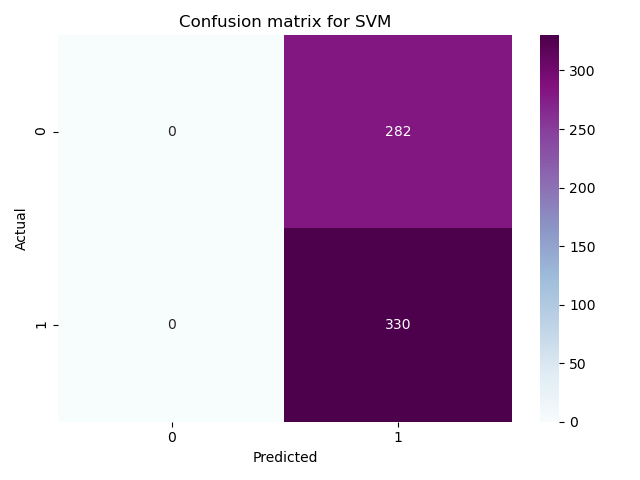
****

ROC Curve came out to be as almost random outputs. Gaussian Naïve Bayesian’s curve is very close to the random line whereas same for SVM is on the random line. The accuracy of both the classifiers is around 50% which is again a random result. ROC for AdaBoost classifier is not desirable as I used decision tree in AdaBoost.

**4. Confusion Matrix**



****

****

Confusion matrix shows the performance of the classifiers. In this project, the performance is not good for all classifiers as actual values are almost half of the predicted values.

**5. Python Code for all algorithms**

**5.1 INPUT**

### COURSE PROJECT - STOCK UP/DOWN PREDICTION ##  
  
### IMPORTING LIBRARIES ###  
import numpy as np  
import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
import time  
  
  
### DATA PREPROCESSING ###  
#Importing data  
data = pd.read\_csv (r'NASDAQ.csv')  
print (data)  
print (data['LABEL'].unique().tolist()) #To check number of variables i.e. should be only 2 (0 and1)  
  
#Deleting non-useful attributes  
del data['Date']  
  
#Setting targets  
y = data.LABEL  
print (y)  
  
  
### SPLITTING DATA ###  
#Splitting training & Testing data in 3:1 ratio (only for Naive Bayesian & AdaBoost)  
X = data.drop('LABEL',axis=1)  
print (X)  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(  
 X, y, test\_size=0.25, random\_state=42)  
  
print ("Splitting training and testing data in 3:1 ratio...")  
print ("Training data set size (Features): ",X\_train.shape)  
print ("Testing data set size (Features): ", X\_test.shape)  
print ("Training data set size (Labels): ",y\_train.shape)  
print ("Testing data set size (Labels): ",y\_test.shape)  
  
  
### CLASSIFYING ###  
## 1. Gaussian Naive Bayesian Classifier  
from sklearn.naive\_bayes import GaussianNB  
from sklearn import metrics  
  
#Training  
start\_NB\_train = time.time()  
gnb = GaussianNB()  
gnb.fit(X\_train,y\_train)  
end\_NB\_train = time.time()  
print ()  
print ("GAUSSIAN NAIVE BAYESIAN CLASSIFIER:")  
print ("GNB Processing time : Training = %0.3fs " % (end\_NB\_train - start\_NB\_train))  
  
#Testing  
start\_NB\_test = time.time()  
y\_pred\_NB = gnb.predict(X\_test)  
end\_NB\_test = time.time()  
print ("GNB Processing time : Testing = %0.3fs " % (end\_NB\_test - start\_NB\_test))  
print ("GNB Accuracy percentage = ",round(metrics.accuracy\_score(y\_test, y\_pred\_NB)\*100,2),"%")  
print ()  
  
  
  
## 2. AdaBoost Classifier  
from sklearn.ensemble import AdaBoostClassifier  
from sklearn.tree import DecisionTreeClassifier  
from sklearn import metrics  
  
#Training  
start\_AB\_train = time.time()  
ada = AdaBoostClassifier(DecisionTreeClassifier(), n\_estimators=100, random\_state = 0) #Using Decision Tree  
model = ada.fit(X\_train, y\_train)  
end\_AB\_train = time.time()  
print ("ADABOOST CLASSIFIER:")  
print ("AdaBoost Processing time : Training = %0.3fs " % (end\_AB\_train - start\_AB\_train))  
  
#Testing  
start\_AB\_test = time.time()  
y\_pred\_AB = ada.predict(X\_test)  
end\_AB\_test = time.time()  
print ("AdaBoost Processing time : Testing = %0.3fs " % (end\_AB\_test - start\_AB\_test))  
print ("AdaBoost Accuracy percentage = ",round(metrics.accuracy\_score(y\_test, y\_pred\_AB)\*100,2),"%")  
print ()  
  
  
### SVM Classifier with Cross Validation  
from sklearn.svm import SVC  
from sklearn.model\_selection import GridSearchCV  
  
#Finding best C and gamma values  
print("Finding best parameters...")  
start\_SVM\_param = time.time()  
param\_grid = {'C': [0.25,0.5,1,10,1e2,1e3,1e4,1e5],  
 'gamma': [10,1,0.1,0.01,0.05,0.001,0.0001,0.00001], }  
clf = GridSearchCV(  
 SVC(kernel='rbf', class\_weight='balanced'), param\_grid  
)  
clf = clf.fit(X\_train, y\_train)  
end\_SVM\_param = time.time()  
print ("Found best parameters in %0.2f seconds" % (end\_SVM\_param - start\_SVM\_param))  
print("Best C value =", clf.best\_estimator\_.C)  
print("Best Gamma value =", clf.best\_estimator\_.gamma)  
print()  
  
#Training  
start\_SVM\_train = time.time()  
svm = SVC(kernel='rbf', random\_state=0, gamma=clf.best\_estimator\_.gamma, C=clf.best\_estimator\_.C,probability=True) #Using best estimators  
svm.fit(X\_train, y\_train)  
end\_SVM\_train = time.time()  
print ("SVM CLASSIFIER")  
print ("SVM Processing time : Training = %0.3fs " % (end\_SVM\_train - start\_SVM\_train))  
  
#Testing  
start\_SVM\_test = time.time()  
y\_pred\_SVM = svm.predict(X\_test)  
preds\_svm = svm.predict\_proba(X\_test)[:,1]  
end\_SVM\_test = time.time()  
print ("SVM Processing time : Testing = %0.3fs " % (end\_SVM\_test - start\_SVM\_test))  
print ("SVM Accuracy percentage = ",round(metrics.accuracy\_score(y\_test, y\_pred\_SVM)\*100,2),"%")  
  
  
  
  
### Confusion Matrix  
from sklearn.metrics import confusion\_matrix  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
con\_NB = confusion\_matrix(y\_test,y\_pred\_NB)  
con\_AB = confusion\_matrix(y\_test,y\_pred\_AB)  
con\_SVM = confusion\_matrix(y\_test,y\_pred\_SVM)  
class\_names=[0,1]  
fig, ax = plt.subplots()  
tick\_marks = np.arange(len(class\_names))  
plt.xticks(tick\_marks, class\_names)  
plt.yticks(tick\_marks, class\_names)  
  
sns.heatmap(pd.DataFrame(con\_NB), annot=True, cmap="PuRd" ,fmt='g')  
ax.xaxis.set\_label\_position("top")  
plt.tight\_layout()  
plt.title('Confusion matrix for Naive Bayesian', y=1.1)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.show()  
  
sns.heatmap(pd.DataFrame(con\_AB), annot=True, cmap="Blues" ,fmt='g')  
ax.xaxis.set\_label\_position("top")  
plt.tight\_layout()  
plt.title('Confusion matrix for AdaBoost', y=1.1)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.show()  
  
sns.heatmap(pd.DataFrame(con\_SVM), annot=True, cmap="BuPu" ,fmt='g')  
ax.xaxis.set\_label\_position("top")  
plt.tight\_layout()  
plt.title('Confusion matrix for SVM', y=1.1)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.show()  
  
  
  
### ROC  
import sklearn.metrics as metrics  
preds\_nb = gnb.predict\_proba(X\_test)[:,1]  
preds\_svm = svm.predict\_proba(X\_test)[:,1]  
fpr\_nb, tpr\_nb, threshold\_nb = metrics.roc\_curve(y\_test, preds\_nb)  
fpr\_svm, tpr\_svm, threshold\_svm = metrics.roc\_curve(y\_test, preds\_svm)  
roc\_auc\_nb = metrics.auc(fpr\_nb, tpr\_nb)  
roc\_auc\_svm = metrics.auc(fpr\_svm, tpr\_svm)  
  
import matplotlib.pyplot as plt  
plt.title('Receiver Operating Characteristic')  
plt.plot(fpr\_nb, tpr\_nb, 'b', label = 'AUC of Naive Bayesian = %0.3f' % roc\_auc\_nb)  
plt.plot(fpr\_svm, tpr\_svm, 'g', label = 'AUC of SVM = %0.3f' % roc\_auc\_svm)  
plt.legend(loc = 'lower right')  
plt.plot([0, 1], [0, 1],'r--',label = 'Random guess')  
plt.xlim([0, 1])  
plt.ylim([0, 1])  
plt.ylabel('True Positive Rate')  
plt.xlabel('False Positive Rate')  
plt.show()

**5.2 OUTPUT**

C:\P\python.exe "C:/Users/Livanshu Kashyap/PycharmProjects/untitled9/Project\_ML2.py"

Date LABEL Open ... Oil TEDSpread EFFR

0 4/1/2008 0 2306.510010 ... 100.92 1.30 2.38

1 4/2/2008 1 2363.419922 ... 104.83 1.31 2.18

2 4/3/2008 1 2347.909912 ... 103.92 1.35 2.19

3 4/4/2008 0 2366.909912 ... 106.09 1.40 2.26

4 4/7/2008 0 2386.620117 ... 108.91 1.28 2.24

... ... ... ... ... ... ... ...

2443 3/23/2018 1 7170.680176 ... 65.80 0.58 1.68

2444 3/26/2018 0 7125.200195 ... 65.49 0.53 1.68

2445 3/27/2018 0 7255.470215 ... 65.21 0.56 1.68

2446 3/28/2018 1 6978.299805 ... 64.30 0.61 1.68

2447 3/29/2018 0 6984.660156 ... 64.87 0.61 1.68

[2448 rows x 14 columns]

[0, 1]

0 0

1 1

2 1

3 0

4 0

..

2443 1

2444 0

2445 0

2446 1

2447 0

Name: LABEL, Length: 2448, dtype: int64

Open High Low ... Oil TEDSpread EFFR

0 2306.510010 2362.750000 2305.399902 ... 100.92 1.30 2.38

1 2363.419922 2381.209961 2347.780029 ... 104.83 1.31 2.18

2 2347.909912 2373.989990 2339.379883 ... 103.92 1.35 2.19

3 2366.909912 2391.929932 2351.760010 ... 106.09 1.40 2.26

4 2386.620117 2390.040039 2359.540039 ... 108.91 1.28 2.24

... ... ... ... ... ... ... ...

2443 7170.680176 7194.310059 6992.669922 ... 65.80 0.58 1.68

2444 7125.200195 7225.830078 7022.339844 ... 65.49 0.53 1.68

2445 7255.470215 7255.540039 6963.680176 ... 65.21 0.56 1.68

2446 6978.299805 7036.089844 6901.069824 ... 64.30 0.61 1.68

2447 6984.660156 7120.459961 6935.779785 ... 64.87 0.61 1.68

[2448 rows x 12 columns]

Splitting training and testing data in 3:1 ratio...

Training data set size (Features): (1836, 12)

Testing data set size (Features): (612, 12)

Training data set size (Labels): (1836,)

Testing data set size (Labels): (612,)

GAUSSIAN NAIVE BAYESIAN CLASSIFIER:

GNB Processing time : Training = 0.031s

GNB Processing time : Testing = 0.016s

GNB Accuracy percentage = 54.9 %

ADABOOST CLASSIFIER:

AdaBoost Processing time : Training = 0.156s

AdaBoost Processing time : Testing = 0.000s

AdaBoost Accuracy percentage = 52.12 %

Finding best parameters...

Found best parameters in 77.51 seconds

Best C value = 1

Best Gamma value = 10

SVM CLASSIFIER

SVM Processing time : Training = 1.516s

SVM Processing time : Testing = 0.234s

SVM Accuracy percentage = 53.92 %

Process finished with exit code 0