Leveraging Naive Bayes with TF-IDF Vectorizer to Predict Stock Movement from Textual Headlines

Group 11: Achyut Dubey (MIT2020082), Samiksha Gupta (MIT2020105)

Introduction to Machine Learning,

II semester M.Tech, Department of Information Technology (Data Engineering),

Indian Institute of Information Technology, Allahabad

1 Introduction

1.1 Naive Bayes Classifier

The algorithm is called "Naive" because it makes a naive assumption that **each feature is independent of other features** which is not true in real life. As for the "Bayes" part, it refers to the statistician and philosopher, Thomas Bayes and the theorem named after him, Bayes' theorem, which is the base for Naive Bayes Algorithm.

Simply speaking, Naive Bayes algorithm can be defined as a **supervised classification algorithm** which is based on Bayes theorem with an assumption of independence among features.

Lets look at the equation for Bayes Theorem,

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- P(A|B) is the probability of hypothesis A given the data B. This is called the posterior probability.
- P(B|A) is the probability of data B given that the hypothesis A was true.
- P(A) is the probability of hypothesis A being true (regardless of the data). This is called the prior probability of A.
- P(B) is the probability of the data (regardless of the hypothesis).

1.2 TF-IDF Vectorizer

When dealing with texts in machine learning, it is quite common to transform the text into data that can be easily analyzed and quantify. For this, the most commonly used technique is the TF-IDF, short for **"term frequency-inverse document frequency"**, which basically reflects how important a word is to a document (email) in a collection or corpus (our set of emails or documents).

The TF-IDF is a statistic that increases with the number of times a word appears in the document, penalized by the number of documents in the corpus that contain the word.

1.3 About the Dataset

The Stock data used for this project consists of 27 columns. The last 25 columns consist of news headlines published on the date mentioned in the first column. The second column shows the label assigned to that particular tuple (either 0 or 1), where 1 means the Stock will go up and 0 means the stock will go down.

2 Code Snippet

```
import pandas as pd
df= pd.read_csv('Stock_Dataa.csv',encoding='ISO-8859-1')

// Performing Exploratory Data Analysis (EDA) on Stock dataset

# LOOKING AT TOP 5 RECORDS OF DATASET
df.head()

# LOOKING AT RANDOM 5 RECORDS OF THE DATASET
df.sample(5)

# DIVIDING THE DATASET INTO TRAINING AND TEST SETS ACCORDING TO DATE
train= df[df['Date']< '20150101']
test= df[df['Date']> '20141231']

# REMOVING PUNCTUATIONS
data= train.iloc[:,2:27]
data.replace("[^a-zA-Z]"," ",regex=True, inplace=True)

# RENAMING COLUMN NAME FOR EASE OF ACCESS
list1=[i for i in range(25)]
new_Index=[str(i) for i in list1]
data.columns= new_Index
data.head()

# CONVERTING THE HEADLINES INTO LOWER CASE
for index in new_Index:
    data[index]=data[index].str.lower()
data.head(1)

# COMBINING ALL 25 HEADLINES PRESENT IN THE FIRST ROW
' '.join(str(x) for x in data.iloc[0,0:25])

# COMBINING THE TOP 25 HEADLINES FOR EACH RECORD IN THE DATASET SO THAT WE COULD CONVERT THEM INTO VECTORS
headlines=[]
for row in range(0,len(data.index)):
    headlines.append(' '.join(str(x) for x in data.iloc[row,0:25]))

// STANDARD NAIVE BAYES CLASSIFIER

from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer
naive= MultinomialNB()

# USING BAG OF WORDS MODEL FOR CONVERTING TEXT INTO VECTORS

countvector= CountVectorizer(ngram_range=(2,2))
traindataset= countvector.fit_transform(headlines) #CONVERTING ALL THE HEADLINES INTO VECTOR
```

```
# FITTING TRAIN DATA INTO NAIVE BAYES CLASSIFIER
naive.fit(traindataset,train['Label'])

# PREDICTING FOR TEST DATASET

test_transform=[]
for row in range(0,len(test.index)):
    test_transform.append(' '.join(str(x) for x in test.iloc[row,2:27]))
test_dataset= countvector.transform(test_transform)
predictions= naive.predict(test_dataset)

# LOOKING AT ALL THE PREDICTIONS
predictions

# FOR CHECKING ACCURACY
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report

# PLOTTING CONFUSION MATRIX HEATMAP
import scikitplot as skplt
skplt.metrics.plot_confusion_matrix(
    test["Label"],predictions,
    figsize=(6,6),cmap="summer")

matrix= confusion_matrix(test["Label"],predictions)
print(matrix)
score= accuracy_score(test["Label"],predictions)
print(score)
report= classification_report(test['Label'],predictions)
print(report)

matrix= confusion_matrix(test["Label"],predictions)
print(matrix)
score= accuracy_score(test["Label"],predictions)
print(score)
report= classification_report(test['Label'],predictions)
print(report)
```

```
[[138  48]
 [ 10 182]]
0.8465608465608465
              precision    recall  f1-score   support

           0       0.93      0.74      0.83       186
           1       0.79      0.95      0.86       192

    accuracy                           0.85       378
   macro avg       0.86      0.84      0.84       378
weighted avg       0.86      0.85      0.84       378
```

```
// USING NAIVE BAYES CLASSIFIER WITH TF-IDF VECTORIZER

# IMPLEMENTING BAG OF WORDS MODEL
countvector= CountVectorizer(ngram_range=(2,2))
traindataset= countvector.fit_transform(headlines)
```

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
naive= MultinomialNB()

# IMPLEMENTING TF-IDF VECTORIZER
tfidf= TfidfVectorizer(ngram_range=(2,2))

# CONVERTING ALL THE HEADLINES INTO VECTORS using TF-IDF technique
traindataset= tfidf.fit_transform(headlines)

naive.fit(traindataset,train['Label'])

# PREDICTING FOR TEST DATASET
test_transform=[]
for row in range(0,len(test.index)):
    test_transform.append(' '.join(str(x) for x in test.iloc[row,2:27]))
test_dataset= countvector.transform(test_transform)
predictions= naive.predict(test_dataset)

# PLOTTING CONFUSION MATRIX HEATMAP
import scikitplot as skplt
skplt.metrics.plot_confusion_matrix(
    test["Label"],predictions,
    figsize=(6,6),cmap="winter")

# ACCURACY AFTER USING TF-IDF VECTORIZER IN NAIVE BAYES CLASSIFIER
matrix= confusion_matrix(test["Label"],predictions)
print(matrix)
score= accuracy_score(test["Label"],predictions)
print(score)
report= classification_report(test['Label'],predictions)
print(report)
```

```
[[130  56]
 [  0 192]]
0.8518518518518519
              precision    recall  f1-score   support

           0       1.00      0.70      0.82       186
           1       0.77      1.00      0.87       192

    accuracy                           0.85       378
   macro avg       0.89      0.85      0.85       378
weighted avg       0.89      0.85      0.85       378
```

3 Analysis

Figure 1 corresponds to the train-test split where the training set consists of entries before 1st of January, 2015 and the test set consists of entries after 31st December, 2014. The model when trained with just standard Naive Bayes gives an accuracy of 0.8465, whereas when Naive Bayes is coupled with TF-IDF vectorizer, the accuracy becomes 0.8515.

Figure 2 corresponds to the train-test split where the training set consists of entries before 1st of January, 2016 and the test set consists of entries after 31st December, 2015. The model when trained with just standard Naive Bayes algorithm as well as Naive Bayes coupled with TF-IDF vectorizer, the accuracy received is perfect 1.0000.

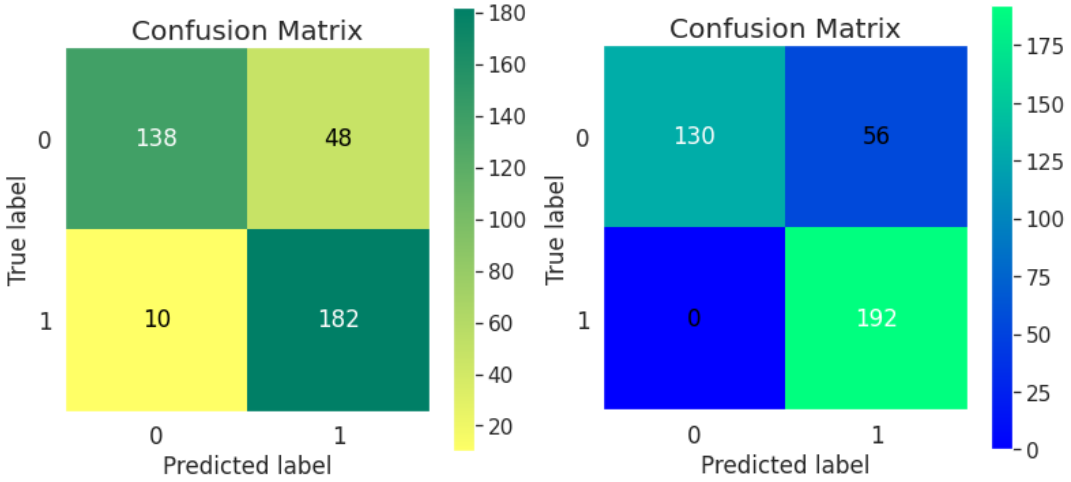

Figure 1: (a) Standard Naive Bayes (b) Naive Bayes with TF-IDF

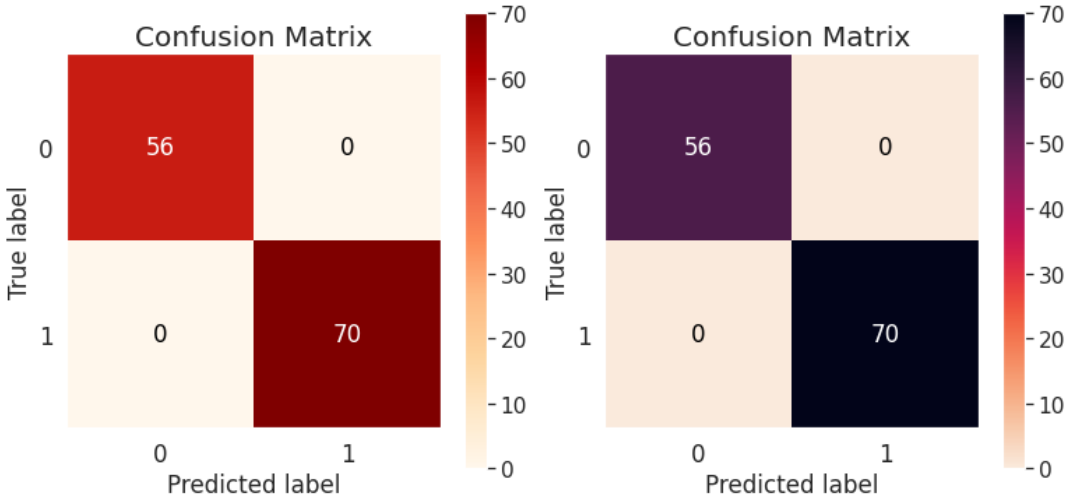

Figure 2: (a) Standard Naive Bayes (b) Naive Bayes with TF-IDF

4 Applications of Naive Bayes Algorithm:

1. Naive Bayes is widely used for text classification
2. Another example of Text Classification where Naive Bayes is mostly used is Spam Filtering in Emails
3. Other Examples include Sentiment Analysis, Recommender Systems, etc.