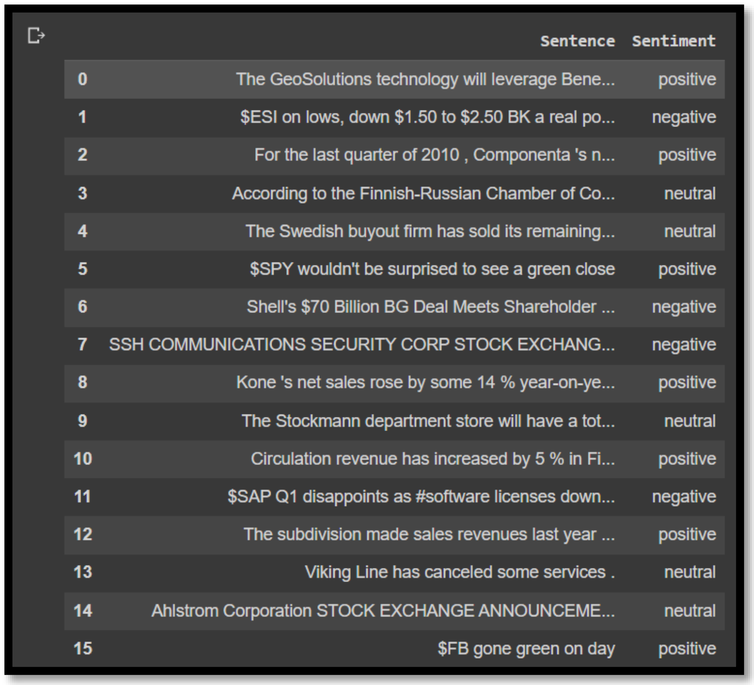
Sentiment Analysis using Naïve Bayes

By Yagna Hari Muni

Describe the task you were solving (sentiment classification) and the dataset you are using. Why did you choose that dataset?

I tried working on Sentiment Analysis using Naïve Bayes algorithm to predict inclination of people's opinions across social media platforms or any other sources. This aids in predicting people's state of mind and monitor social media activities.

I considered analyzing financial dataset due to its three sentiments positive, negative, and neutral. Rather than analyzing a dataset of movie review/tweets which contain direct words like good/bad, I wished analyzing the financial dataset was something different and decided to analyze the precision score, recall score, and f1 score.



	Sentence	Sentiment
0	The GeoSolutions technology will leverage Bene...	positive
1	$ESI on lows, down $1.50 to $2.50 BK a real po...	negative
2	For the last quarter of 2010 , Componenta 's n...	positive
3	According to the Finnish-Russian Chamber of Co...	neutral
4	The Swedish buyout firm has sold its remaining...	neutral
5	$SPY wouldn't be surprised to see a green close	positive
6	Shell's $70 Billion BG Deal Meets Shareholder ...	negative
7	SSH COMMUNICATIONS SECURITY CORP STOCK EXCHANG...	negative
8	Kone 's net sales rose by some 14 % year-on-ye...	positive
9	The Stockmann department store will have a tot...	neutral
10	Circulation revenue has increased by 5 % in Fi...	positive
11	$SAP Q1 disappoints as #software licenses down...	negative
12	The subdivision made sales revenues last year ...	positive
13	Viking Line has canceled some services .	neutral
14	Ahlstrom Corporation STOCK EXCHANGE ANNOUNCEME...	neutral
15	$FB gone green on day	positive

Fig 1: Financial Data set with Sentence and Sentiment Columns.

Based on data analysis stage, what can you say about your dataset. Is it a balanced dataset or not, if it is not balanced, what is the main problem? (e.g. something like this: an the lack of neutral sentiment samples won't give enough opportunity for our classifier to properly distinguish neutral samples from the rest)

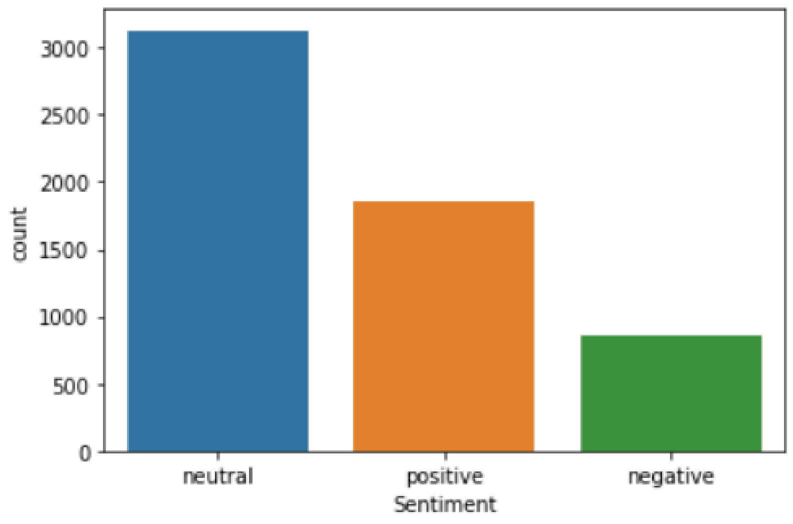


Fig 2: Count Plot for the Financial Dataset

Based on the analysis of financial dataset, it is an unbalanced dataset with less accuracy in predictions due to high amount neutral values. From my dataset, I observe that there are majority of neutral classes compared to positive and negative classes.

These imbalanced datasets are a challenge for predictive modeling since ML algorithms are designed in assumption of equal number of examples. This results in having the models with poor predictive performance especially for minor classes. The problem is that these minor classes are important and sensitive to errors compared to majority class.

What can you say about your dataset based on generated plots (wordcloud and an interactive plot). Make sure to include all generated plots.

Count Plot:

The count plot **(Fig. 2)** depicts the number of positive, negative, and neutral sentiments on a bar graph.

WordCloud:

WordCloud is a simple way to depict specific words in textual data. The larger and bolder the text appears in the cloud, the more often it is mentioned in the data file. This helps to identify the word similarities between different sentiments.

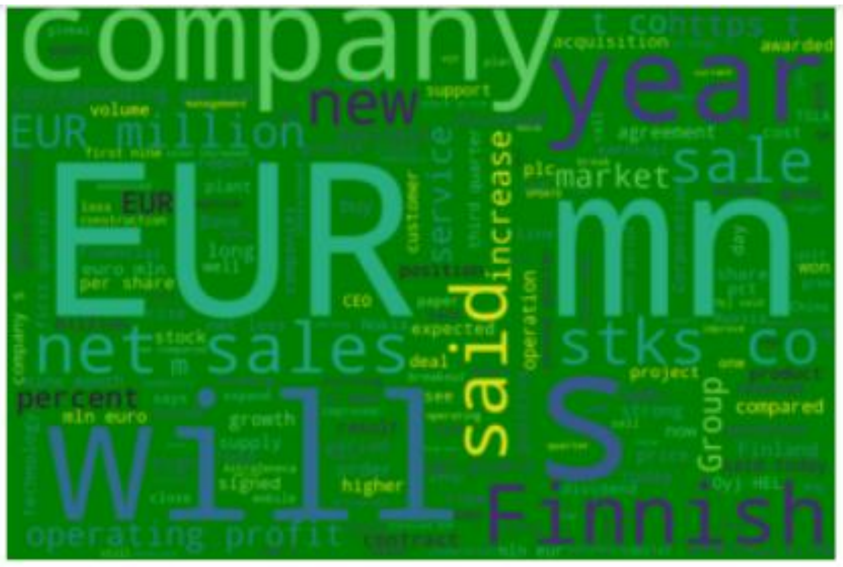


Fig 3: Positive Word Cloud

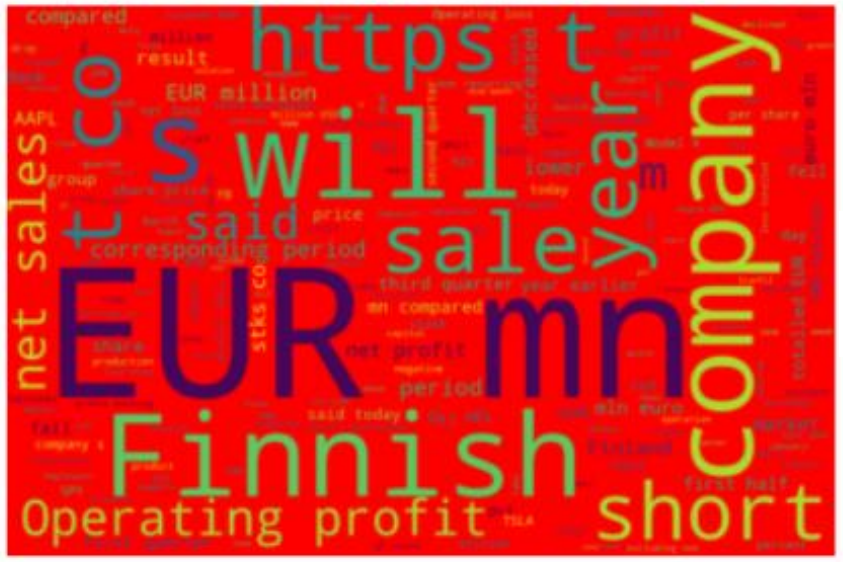


Fig 4: Negative Word Cloud

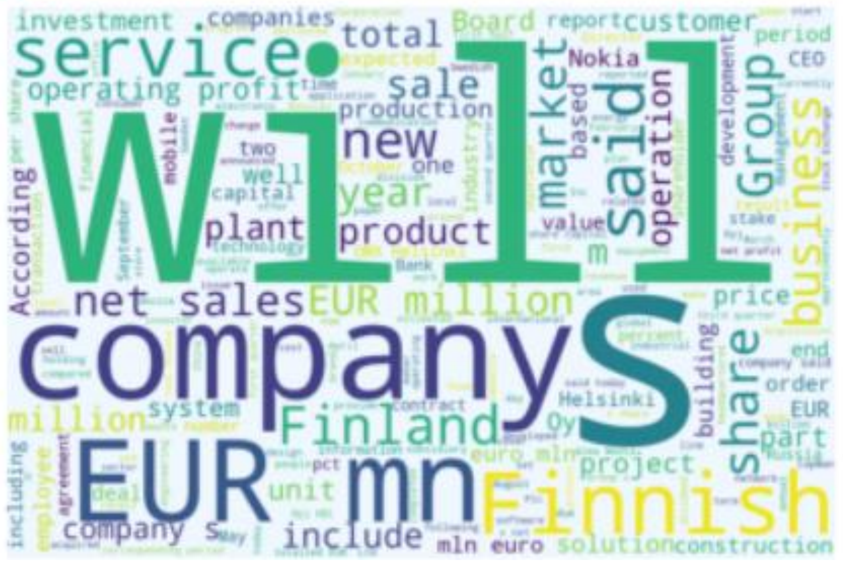


Fig 5: Neutral Word Cloud

Scatter Plot:

The scatter plot helps to analyze trends in data and define the relationship between positive and negative variables. Since, the scatterplot formed a line that is slanting upward from left to right, it is a positive relationship or positive correlation between both the variables.

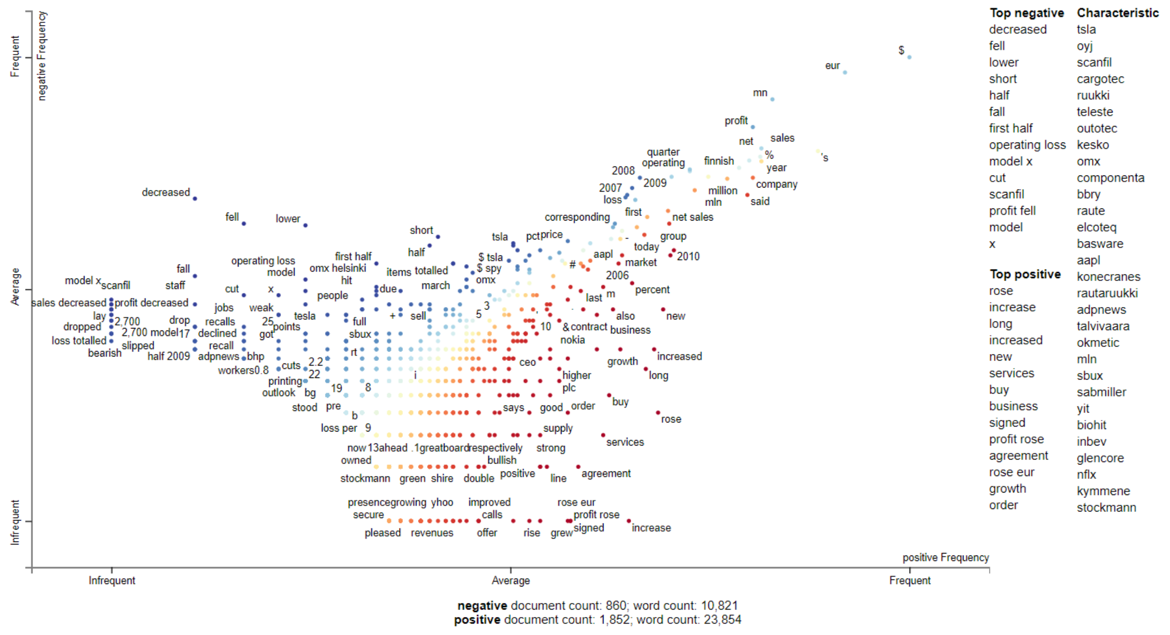


Fig 6: ScatterPlot for positive and negative classes

Confusion Matrix:

Confusion matrix is visualization of an algorithm. As per the matrix, each row represents the instances in predicted class and the column represents the instance in true class. It is useful for measuring the scores: Precision, recall, and f1.

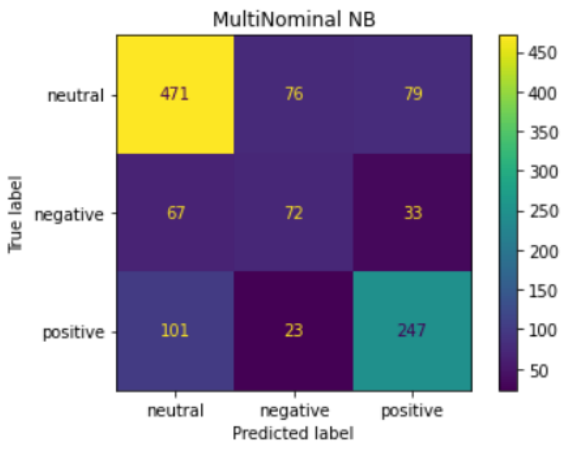
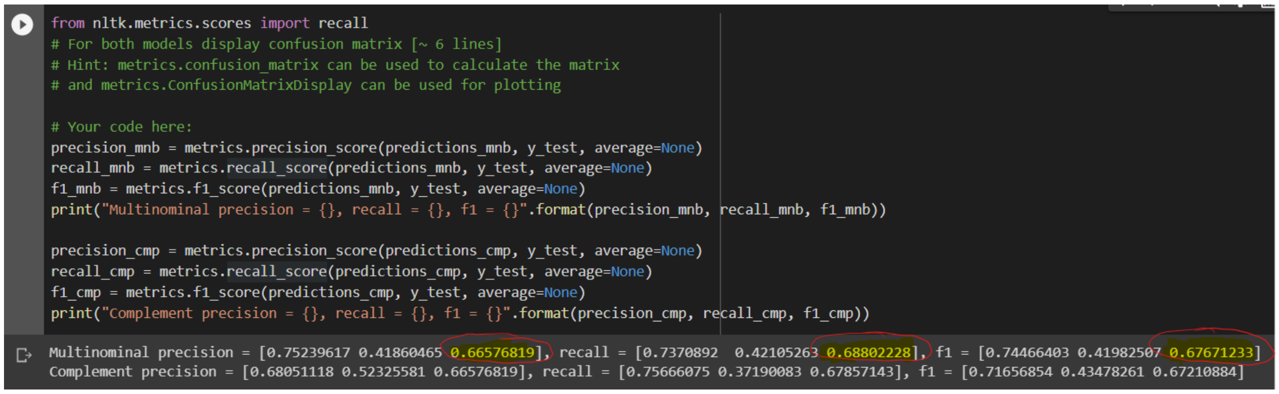


Fig 6: ScatterPlot for positive and negative classes

Calculating the precision, recall and accuracy for one of the classes:

Precision = 247 + (247 + 23 + 101) = **0.66**
Recall = 247 + (247 + 33 + 79) = **0.68**
f1 = 2 X 0.68 X 0.66 / (0.68 + 0.66) = **0.67**

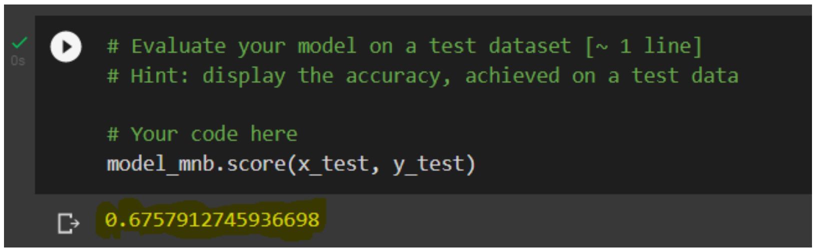


```
from nltk.metrics.scores import recall
# For both models display confusion matrix [~ 6 lines]
# Hint: metrics.confusion_matrix can be used to calculate the matrix
# and metrics.ConfusionMatrixDisplay can be used for plotting

# Your code here:
precision_mnb = metrics.precision_score(predictions_mnb, y_test, average=None)
recall_mnb = metrics.recall_score(predictions_mnb, y_test, average=None)
f1_mnb = metrics.f1_score(predictions_mnb, y_test, average=None)
print("Multinominal precision = {}, recall = {}, f1 = {}".format(precision_mnb, recall_mnb, f1_mnb))

precision_cmp = metrics.precision_score(predictions_cmp, y_test, average=None)
recall_cmp = metrics.recall_score(predictions_cmp, y_test, average=None)
f1_cmp = metrics.f1_score(predictions_cmp, y_test, average=None)
print("Complement precision = {}, recall = {}, f1 = {}".format(precision_cmp, recall_cmp, f1_cmp))
```

```
Multinominal precision = [0.75239617 0.41860465 0.66576819], recall = [0.7370892  0.42105263 0.68802228], f1 = [0.74466403 0.41982507 0.67671233]
Complement precision = [0.68051118 0.52325581 0.66576819], recall = [0.75666075 0.37190083 0.67857143], f1 = [0.71656854 0.43478261 0.67210884]
```

Fig 7: The calculated values can be compared to Actual values

Accuracy: True values / Total = 790/ 1169 = **0.67**



```
# Evaluate your model on a test dataset [~ 1 line]
# Hint: display the accuracy, achieved on a test data

# Your code here
model_mnb.score(x_test, y_test)
```

```
0.6757912745936698
```

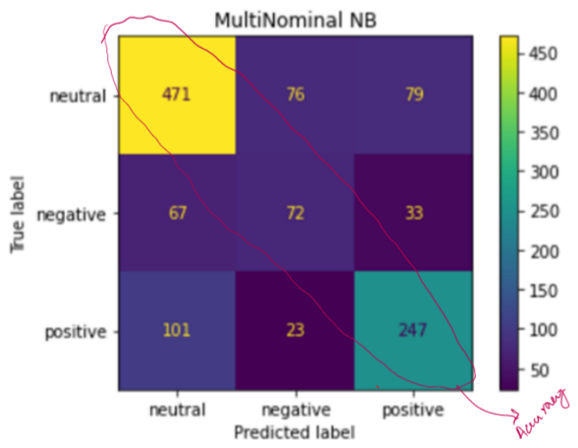


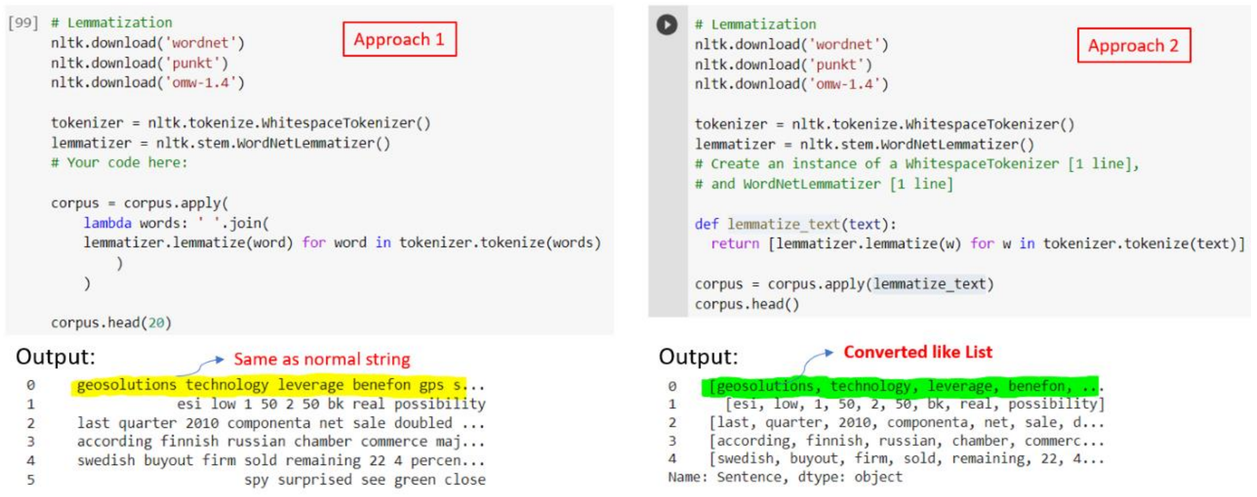
Fig 8,9: The calculated values can be compared to Actual values

Based on the Sentiment classification task, how did Multinomial Naive Bayes and Complement NB perform, include computed stats (accuracy, precision, recall, f1). It would be nice to compose a table with the results. Which classifier performed the best and why?

AND

Provide confusion tables and your observations, i.e. which class was the easiest to classify and which was the hardest? Based on your label distribution (balanced/imbalanced), is this what you've expected to see?

For the sentiment classification task, I have followed 2 approaches:
1. Approach 1: Regular approach as explained in HW 01
2. Approach 2: Changed lemmatization and tokenization task making it more granular.



```
[99] # Lemmatization
     nltk.download('wordnet')
     nltk.download('punkt')
     nltk.download('omw-1.4')

     tokenizer = nltk.tokenize.WhitespaceTokenizer()
     lemmatizer = nltk.stem.WordNetLemmatizer()
     # Your code here:

     corpus = corpus.apply(
         lambda words: ' '.join(
         lemmatizer.lemmatize(word) for word in tokenizer.tokenize(words)
             )
         )

     corpus.head(20)
```

```
0    geosolutions technology leverage benefon gps s...
1                  esi low 1 50 2 50 bk real possibility
2    last quarter 2010 componenta net sale doubled ...
3    according finnish russian chamber commerce maj...
4    swedish buyout firm sold remaining 22 4 percen...
5                           spy surprised see green close
```

```
# Lemmatization
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('omw-1.4')

tokenizer = nltk.tokenize.WhitespaceTokenizer()
lemmatizer = nltk.stem.WordNetLemmatizer()
# Create an instance of a WhitespaceTokenizer [1 line],
# and WordNetLemmatizer [1 line]

def lemmatize_text(text):
  return [lemmatizer.lemmatize(w) for w in tokenizer.tokenize(text)]

corpus = corpus.apply(lemmatize_text)
corpus.head()
```

```
0    [geosolutions, technology, leverage, benefon, ...
1      [esi, low, 1, 50, 2, 50, bk, real, possibility]
2    [last, quarter, 2010, componenta, net, sale, d...
3    [according, finnish, russian, chamber, commerc...
4    [swedish, buyout, firm, sold, remaining, 22, 4...
Name: Sentence, dtype: object
```

Fig 10: Two Approaches for Lemmatization and Tokenization

In Approach 1, the precision values are:

Multinominal precision = [0.75239617 0.41860465 0.66576819], recall = [0.7370892 0.42105263 0.68802228], f1 = [0.74466403 0.41982507 0.67671233]

Complement precision = [0.68051118 0.52325581 0.66576819], recall = [0.75666075 0.37190083 0.67857143], f1 = [0.71656854 0.43478261 0.67210884]

In Approach 2, the precision values are: **(Resolved issue with list not having lower() method)**

Multinominal precision = [0.74440895 0.40116279 0.65498652], recall = [0.73617694 0.40828402 0.66212534], f1 = 0.74027006 0.40469208 0.65853659]

Complement precision = [0.68370607 0.52906977 0.67385445], recall = [0.76428571 0.37603306 0.68119891], f1 = [0.72175379 0.43961353 0.67750678]

Theoretically, complement precisions suit well for unbalanced datasets. But comparing Multinominal and Complement precisions generated, the values depict that MultinominalNB has more precision compared to ComplementNB.

In my analysis, I understood that unbalanced dataset has less accuracy due to high amount of netutral values. The MultinominalNB and ComplementNB precisions are mostly similar and my approach in making the lemmatization and tokenization more granular has reduced the accuracy by a small factor.

I have shared the Confusion Matrix of Multinominal and Complement Precisions. The left two image show analysis for Approach 1, and right two images with title granular depict the analysis for my Approach 2.

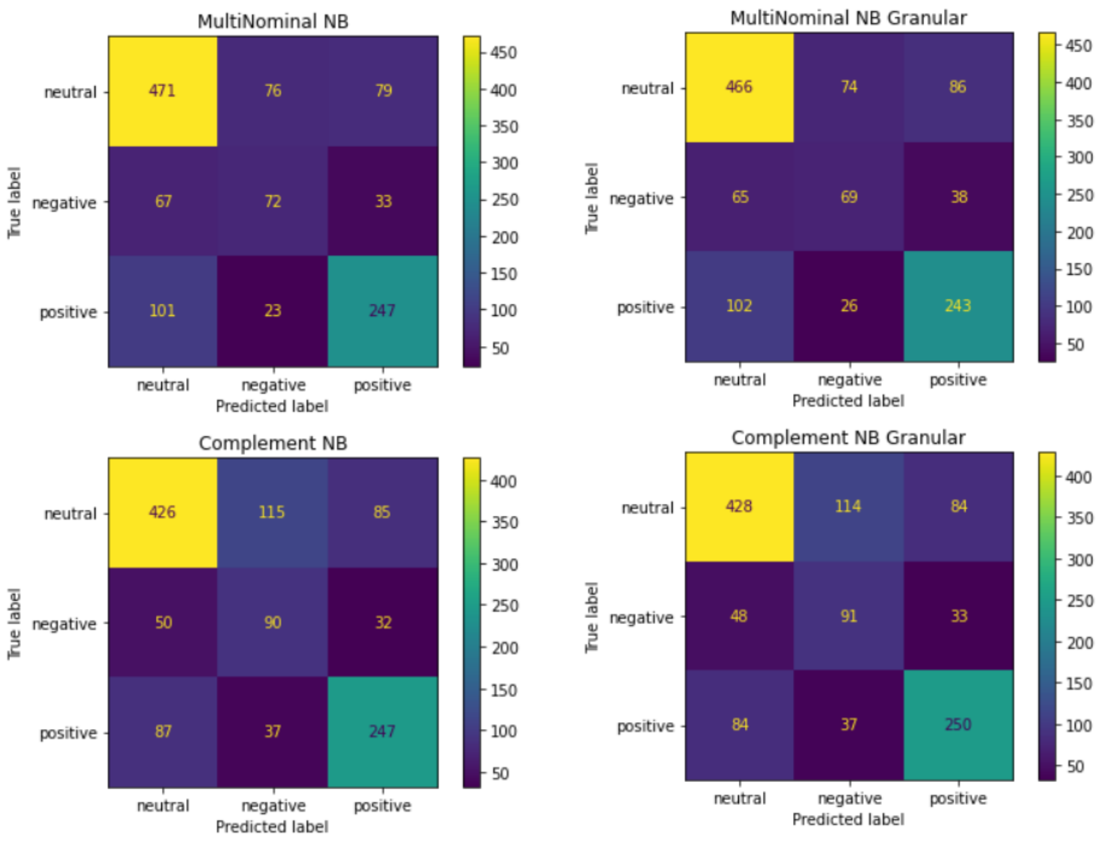


Fig 11: Confusion Matrix for Approach 1 and Approach 2