# ASSIGNMENT 2

1. **Give simple descriptive statistics showing the frequency distribution for the sentiment classes for the whole dataset of 5000 tweets. What do you notice about the distribution?**

Frequency distribution:

| Sentiment | Frequency($f$) | Cumulative Frequency | Relative Frequency ($\frac{f}{n}$) |
|---|---|---|---|
| negative | 3115 | 3115 | 0.623 |
| neutral | 1063 | 4178 | 0.2126 |
| positive | 822 | 5000 = $\Sigma f$ | 0.1644 |
| | $\Sigma f$ = 5000 | $n = \Sigma f$ = 5000 | $\Sigma\frac{f}{n}$ = 1 |

*Table1.1*

Frequency distribution Bar Graph:

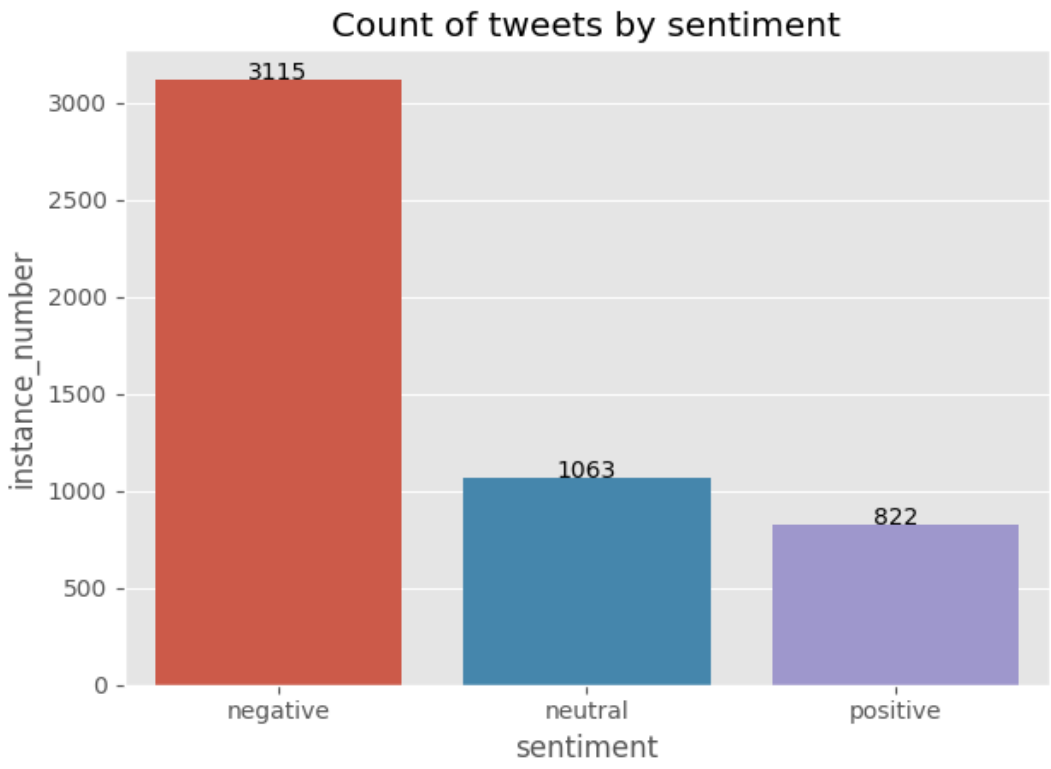


*Figure1.1*

After calculating absolute and relative frequencies and plotting the same into a bar graph clearly states that tweets with negative sentiment is most common i.e. 62.3%, tweets with neutral comments are the second most common i.e. 21.26% and the least common positive sentiment tweets are 16.44% of the whole dataset.

**2. Develop BNB and MNB models from the training set using (a) the whole vocabulary, and (b) the most frequent 1000 words from the vocabulary. Show all metrics on the test set comparing the two approaches for each method. Explain any similarities and differences in results.**

a) Models trained using the whole vocabulary

MNB

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| **negative** | 0.72 | 0.99 | 0.84 | 628 |
| **neutral** | 0.79 | 0.26 | 0.39 | 210 |
| **positive** | 0.83 | 0.39 | 0.53 | 162 |
| **accuracy** | | | 0.74 | 1000 |
| **macro avg** | 0.78 | 0.54 | 0.58 | 1000 |
| **weighted avg** | 0.76 | 0.74 | 0.69 | 1000 |

*Table2.1*

BNB

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| **negative** | 0.68 | 0.99 | 0.80 | 628 |
| **neutral** | 0.77 | 0.21 | 0.33 | 210 |
| **positive** | 0.91 | 0.12 | 0.22 | 162 |
| **accuracy** | | | 0.69 | 1000 |
| **macro avg** | 0.79 | 0.44 | 0.45 | 1000 |
| **weighted avg** | 0.73 | 0.69 | 0.61 | 1000 |

*Table2.2*

b) Models trained using the most frequent 1000 words from the vocabulary

MNB

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| **negative** | 0.84 | 0.89 | 0.86 | 628 |
| **neutral** | 0.63 | 0.54 | 0.58 | 210 |
| **positive** | 0.67 | 0.62 | 0.65 | 162 |
| **accuracy** | | | 0.77 | 1000 |
| **macro avg** | 0.71 | 0.69 | 0.70 | 1000 |
| **weighted avg** | 0.77 | 0.77 | 0.77 | 1000 |

*Table2.3*

BNB

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| **negative** | 0.87 | 0.83 | 0.85 | 628 |
| **neutral** | 0.60 | 0.67 | 0.63 | 210 |
| **positive** | 0.61 | 0.65 | 0.63 | 162 |
| **accuracy** | | | 0.76 | 1000 |
| **macro avg** | 0.69 | 0.71 | 0.70 | 1000 |
| **weighted avg** | 0.77 | 0.76 | 0.77 | 1000 |

*Table2.4*

By looking at the metrics we can clearly see the difference as both the model's performance has significantly increased when trained with 1000 most frequent words from the vocabulary.
For MNB accuracy has increased from 74% to 77%, and other statistics like precision, f1-score and recall has also increased from 74% to 77%, 69% to 77% and 74% to 77% respectively.
For BNB accuracy has increased from 69% to 76%, and other statistics like precision, f1-score and recall has also increased from 73% to 77%, 61% to 77% and 69% to 76% respectively.

3. **Evaluate the three standard models with respect to the VADER baseline. Show all metrics on the test set and comment on the performance of the baseline and of the models relative to the baseline.**

VADER

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| **negative** | 0.91 | 0.48 | 0.63 | 628 |
| **neutral** | 0.35 | 0.40 | 0.38 | 210 |
| **positive** | 0.34 | 0.88 | 0.49 | 162 |
| **accuracy** | | | 0.53 | 1000 |
| **macro avg** | 0.53 | 0.59 | 0.50 | 1000 |
| **weighted avg** | 0.70 | 0.53 | 0.55 | 1000 |

*Table3.1*

The above are the metrics from VADER baseline on the test dataset of the last 1000 tweets.
We can clearly see that the VADER baseline does not perform very well and thus we need other baseline models for NLP.
Comparing the VADER with all the three standard models with the help of heatmaps for more insights.
All the standard models below are trained on the first 4000 tweets and tested on next 1000 tweets of the given dataset as per the scope of this assignment.

## Comparison with DT:

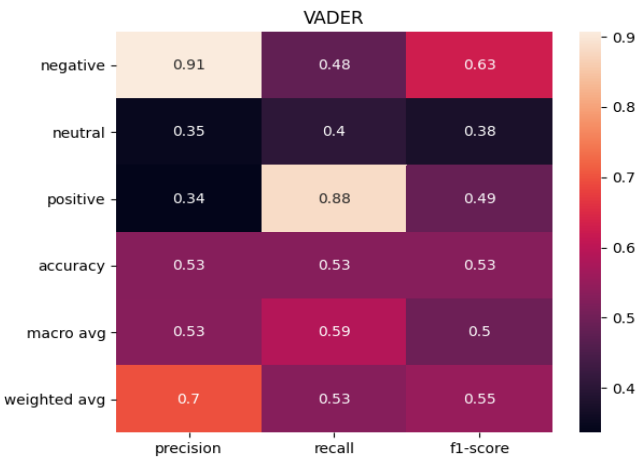


*Figure 3.1*

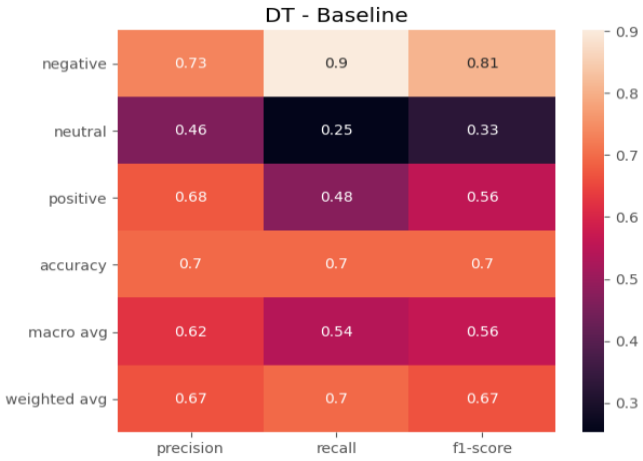


*Figure 3.2*

## Comparison with BNB:

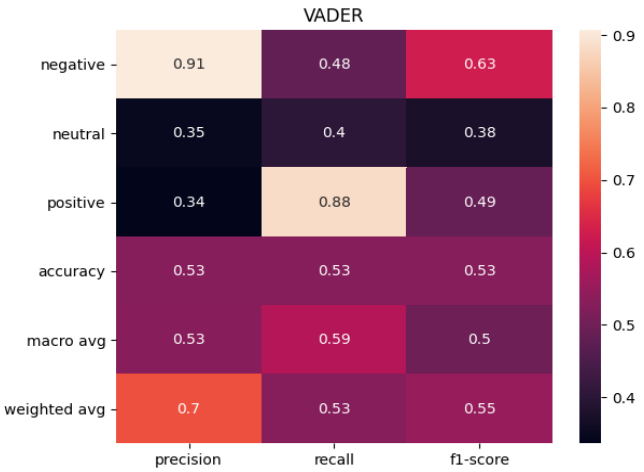


*Figure 3.3*

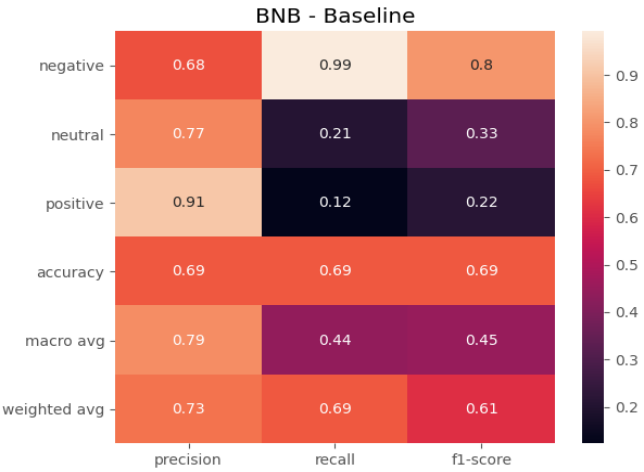


*Figure 3.4*

## Comparison with MNB:

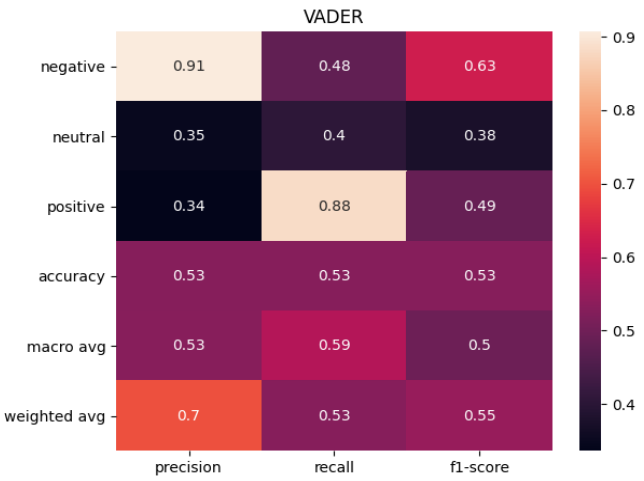


*Figure 3.5*

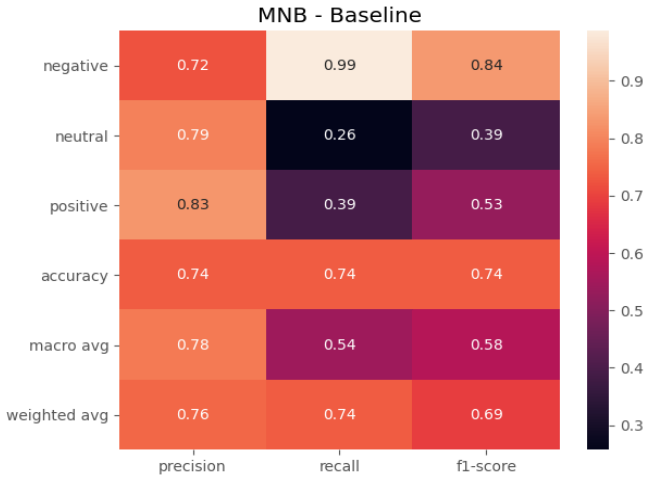


*Figure 3.6*

4. **Evaluate the effect of preprocessing the input features by applying NLTK English stop word removal then NLTK Porter stemming on classifier performance for the three standard models. Show all metrics with and without preprocessing on the test set and explain the results.**

**Stop words:**

A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query.

We can easily remove *english stopwords* in python, just by importing *stopwords* from *nltk.corpus* package and then not including them if they are present our vocabulary.

**Stemming:**

Stemming is the process of producing morphological variants of a root/base word. Stemming programs are commonly referred to as stemming algorithms or stemmers. A stemming algorithm reduces the words “chocolates”, “chocolatey”, “choco” to the root word, “chocolate”.

We can easily stem our vocabulary by simply importing *PorterStemmer* from *nltk.stem* package and then by looping throughout the vocabulary and stemming each word using the *stem()* function.

## Decision Tree (DT)

We can see from the following images (See Figure 4.1, 4.2) that after the removal of stop words there is a slight decrease in the accuracy, f1-score and recall i.e. 70% to 69%, 67% to 61% and 70% to 69% respectively but increase from 67% to 69% is seen in the precision in comparison to the baseline model. After applying porter stemmer we again observe that accuracy, precision, f1-score and recall has increased in comparison to the previous one but it is quite similar to the baseline DT model.

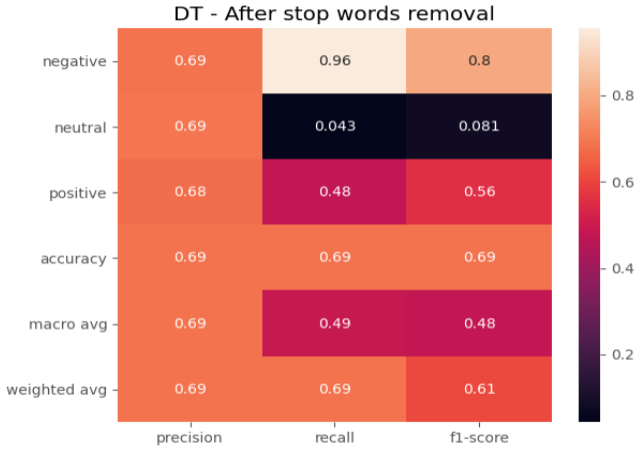


*Figure 4.1*

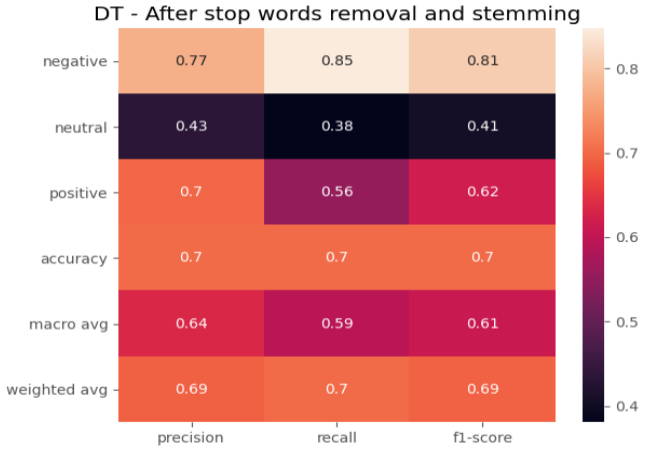


*Figure 4.2*

## Bernoulli Naive Bayes (BNB)

We can see from the following images (Figure 4.3 and 4.4) that after the removal of stop words there is a slight decrease in the accuracy, f1-score, recall and precision i.e. 69% to 67%, 61% to 58%, 69% to 67% and 73% to 72% respectively in comparison to the baseline model.
After applying porter stemmer we again can observe that accuracy, precision, f1-score and recall has increased in comparison to the previous one but it is quite similar to the baseline DT model.

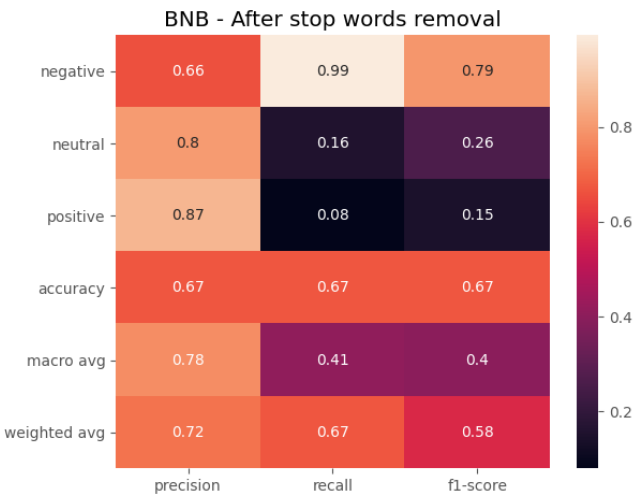


*Figure 4.3*

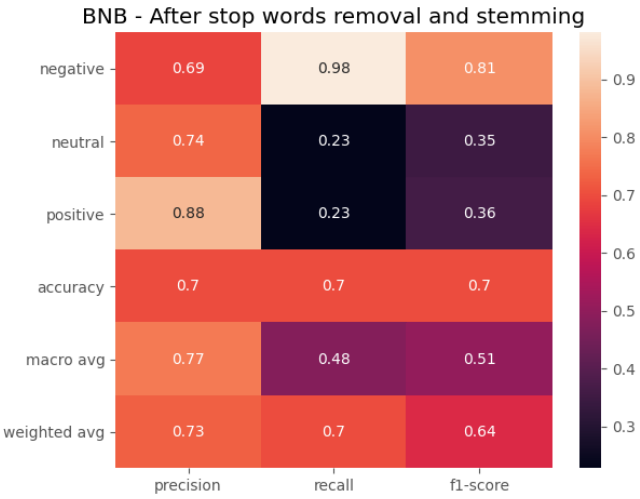


*Figure 4.4*

## Multinomial Naive Bayes (MNB)

We can see from the following images (See Figure 4.5 and 4.6) that after the removal of stop words there is a slight increase in the f1-score i.e. 69% to 71%, accuracy and recall remains the same but a decrease from 76% to 75% is seen in precision in comparison to the baseline model.
After applying porter stemmer we again can observe that accuracy, precision, f1-score and recall has increased in comparison to the baseline DT model.

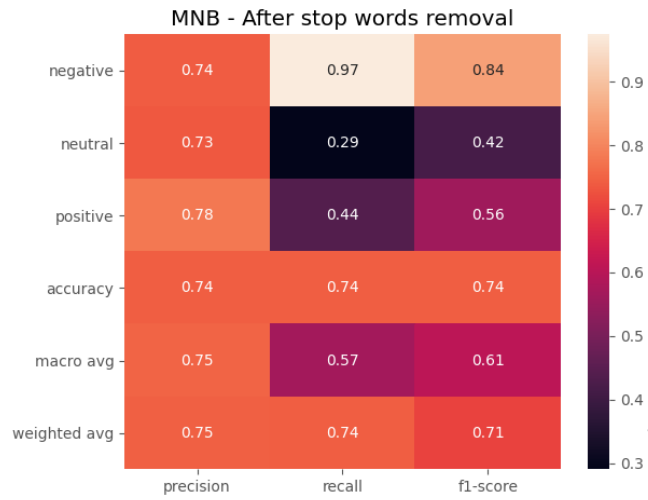


*Figure 4.5*

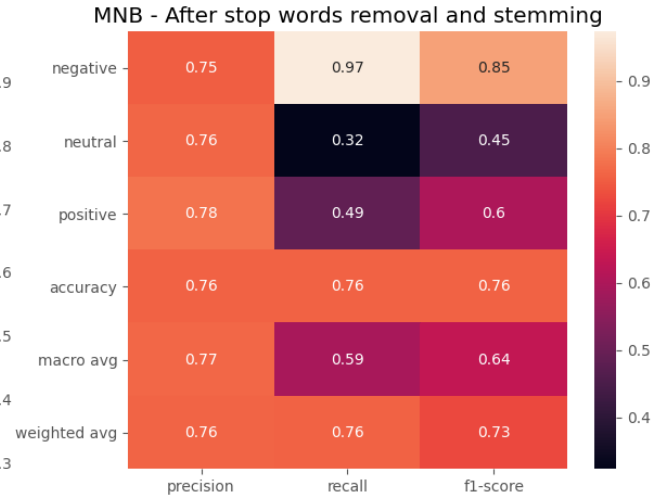


*Figure 4.6*

5. **Evaluate the effect that converting all letters to lower case has on classifier performance for the three standard models. Show all metrics with and without conversion to lower case on the test set and explain the results.**

**Lowercase Conversion:**
It is common to convert all words to one case. This means that the vocabulary will shrink in size, i.e. “Apple”, “apple”, “AppLE” etc will become one single word as “apple”. But some distinctions are definitely lost (e.g. “Apple” the company vs “apple” the fruit is now treated as same word in the vocabulary). This can be easily achieved by removing *lowercase=False* parameter from *Countvectorizer()* function. By default it takes *lowercase* as *True*.

## Decision Tree (DT)

We can see from the below images that after setting the tweets to lowercase there is a slight increase in the accuracy, precision, recall and f1-score i.e. 70% to 71%, 67% to 68%, 70% to 71% and 67% to 68% respectively.

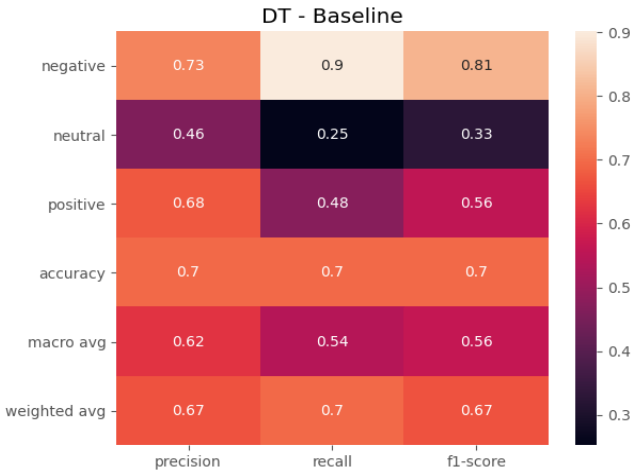


*Figure 5.1*

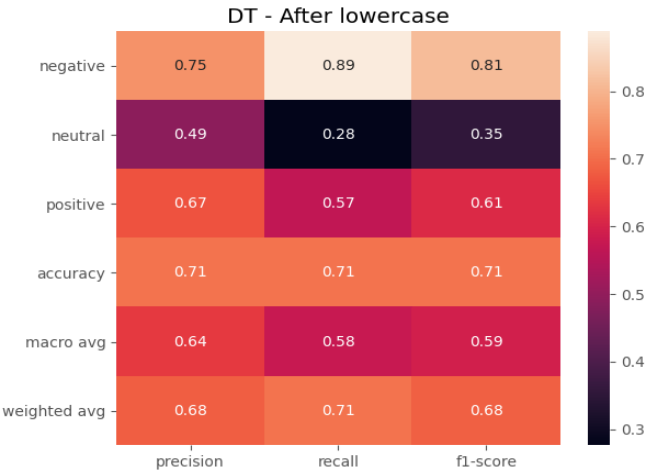


*Figure 5.2*

## Bernoulli Naive Bayes (BNB)

We can see from the below images that after setting the tweets to lowercase there is an increase in the accuracy, precision, recall and f1-score i.e. 69% to 73%, 73% to 77%, 69% to 73% and 61% to 68% respectively.

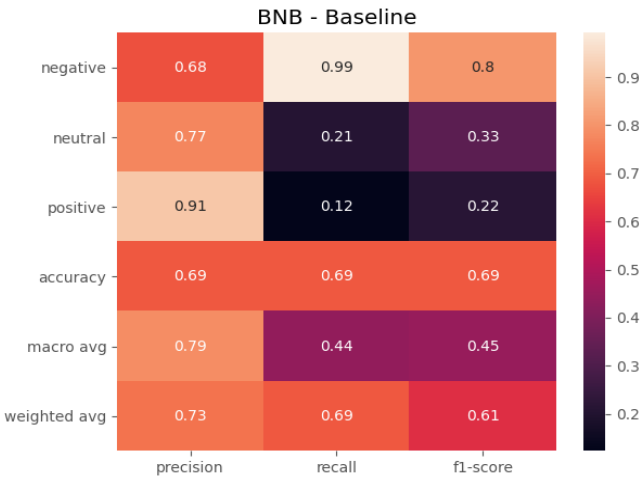


*Figure 5.3*

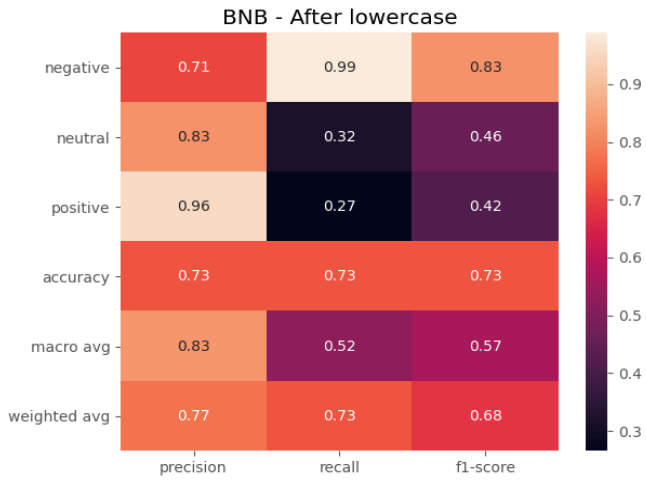


*Figure 5.4*

## Multinomial Naive Bayes (MNB)

We can see from the below images that after setting the tweets to lowercase there is an increase in the accuracy, precision, recall and f1-score i.e. 74% to 76%, 76% to 78%, 74% to 76% and 69% to 73% respectively.

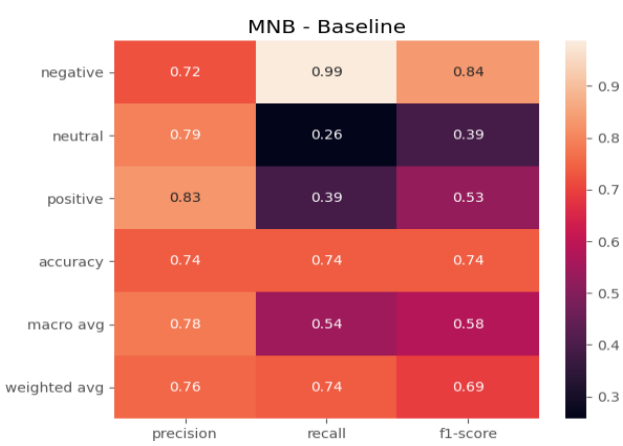


*Figure 5.5*

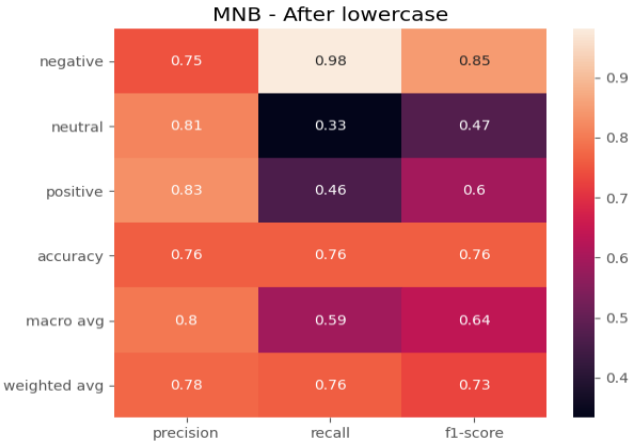


*Figure 5.6*

6. **Describe your best method for sentiment analysis and justify your decision. Give some experimental results for your method trained on the training set of 4000 tweets and tested on the test set of 1000 tweets. Provide a brief comparison of your model to the standard models and the baseline (use the results from the previous questions).**

## Integrated Stacked Model

For better performance I have decided to build an **Integrated Stacked Model**.
When using standard models as sub-models, it may be desirable to use a standard model as a meta-learner. Specifically, the sub-networks can be embedded in a larger multi-headed network that then learns how to best combine the predictions from each input sub-model. It allows the stacking ensemble to be treated as a single large model.

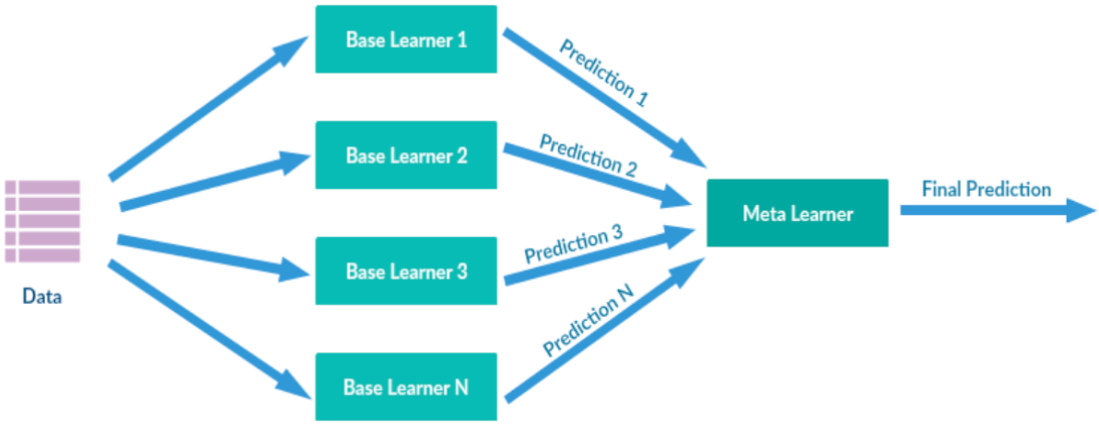


*Figure 6.1 Image from http://supunsetunga.blogspot.com/*

Using all the standard models and using integrated stacked model method and thus comparing the results from Figure 5.1, 5.2 and 5.3.
The following image clearly shows that there no increase in accuracy, recall, precision and f1-score. (See Figure 6.2)

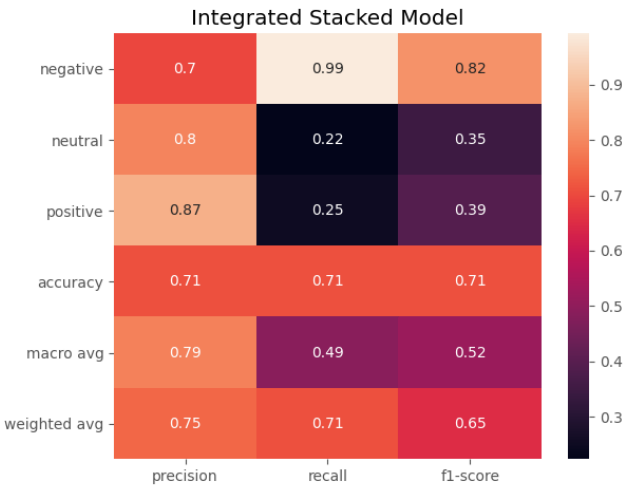


*Figure 6.2*

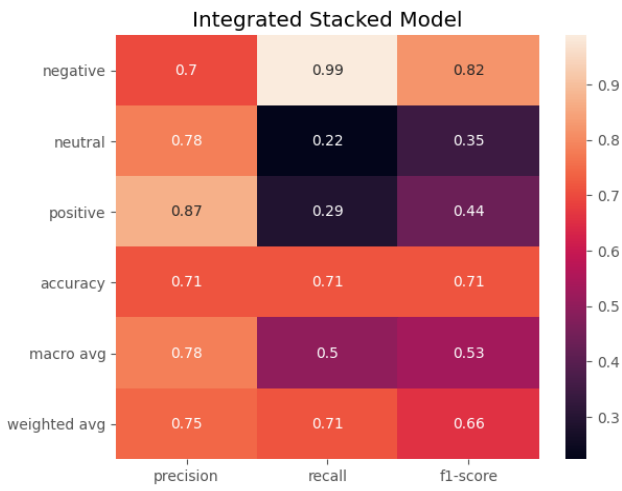


*Figure 6.3*

We can now take the preprocessing to another level by doing a deep cleaning like remove any inline emoticons (e.g. ”☺”,”☹”and so on), HTML tags, tagged people, URLs and some other not so important symbols people tend to use while tweeting on twitter. This deep cleaning is very common in NLP.
The effect is not so significant, as shown in the image above (See Figure 6.3) but still we will keep it.

Now, let’s try to train our standard models on 1000 frequent words from the vocabulary.
The effect is clear, an increased accuracy, precision, recall and f1-score is observed. (See Figure 6.4)

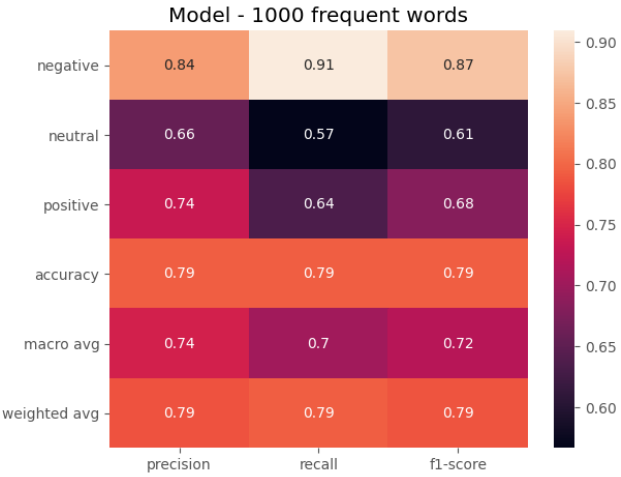


*Figure 6.4*

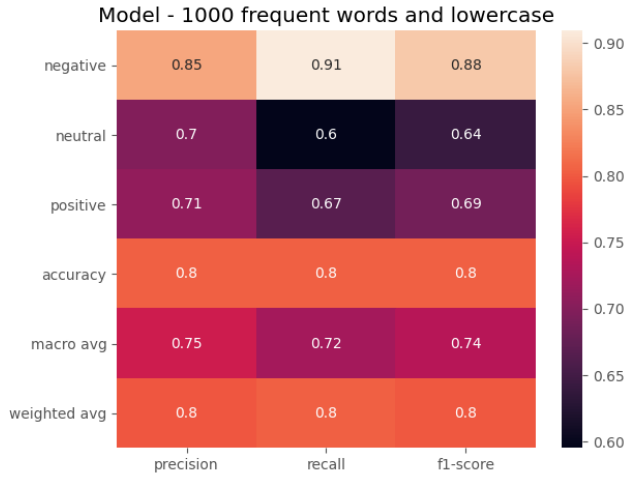


*Figure 6.5*

By now, this integrated stacked model has shown much better performance. Let’s try to lowercase every vocabulary in our model. The effect is, we now get better performance model (See Figure 6.5)

Also we have seen an increase in the performance when training it on 1000 most frequent words, let us try to lower this number and also will check the results if when increased this number of most frequent words in the vocabulary.
Training the model for 100 and 5000 most frequent words and the results are shown below.
(See Figure 6.6 and 6.7)

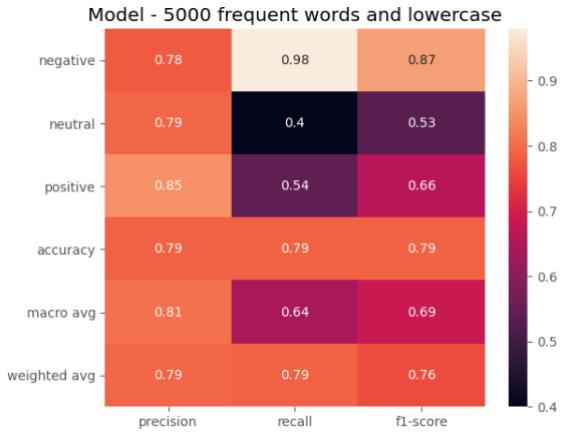


*Figure 6.6*

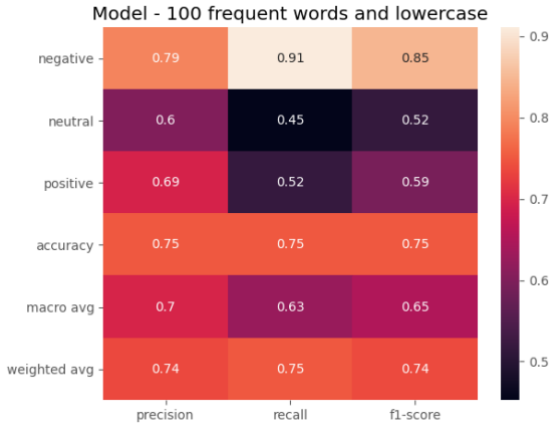


*Figure 6.7*

We have observed a significant decline in the performance. Hence reverting the changes and it is best to train our standard models on 1000 most frequent words from the vocabulary only. Now removing stop words and stemming. The effect is shown below (See Figure 6.8)

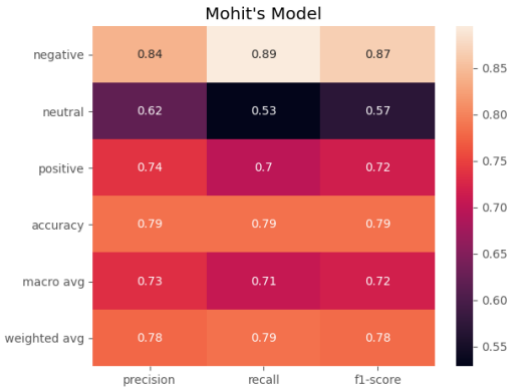


*Figure 6.8*

Not much of the difference is seen in the performance of the model.
Also, from the observations in the above questions we can say that removing stop words and stemming works best for MNB model and does not affect much on the other two models.
Therefore, removing stop words and stemming for MNB only. The effect is in the image (See Figure 6.8)

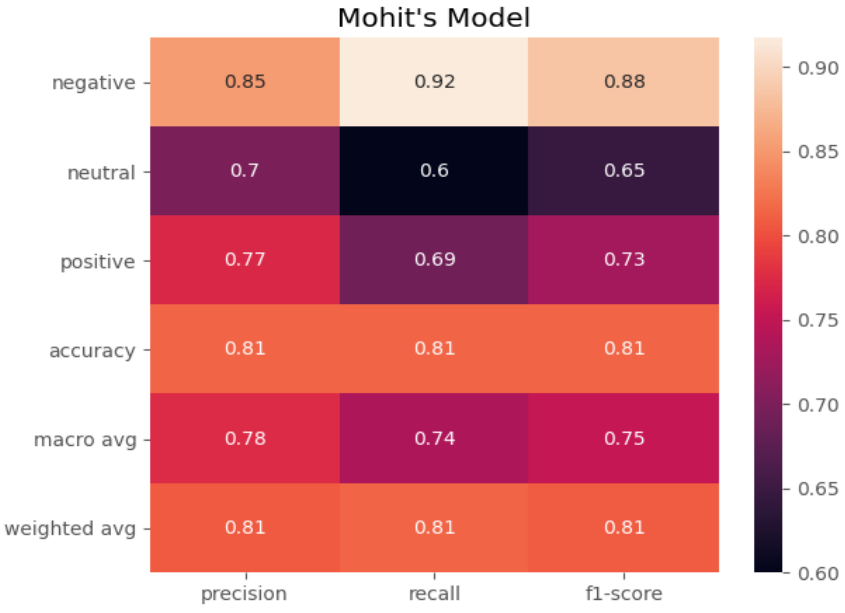


*Figure 6.9*

The final integrated stacked mechanism worked well and the performance is far better than any other standard model. Hence this is the best model with accuracy, recall, precision and f1-score all at 81%.

### Side by Side Comparison of my model with other standard models

Using Integrated Stacked mechanism on the standard model to construct a new model shows significant improvement in the performance metrics. Comparing the same with other standard models can be seen below (See Table 6.1)

| | Mohit's Model | Decision Tree | Bernoulli Naive Bayes | Multinomial Naive Bayes |
|---|---|---|---|---|
| Accuracy | **81%** | 71% | 79% | 78% |
| Precision | **81%** | 68% | 80% | 77% |
| Recall | **81%** | 71% | 79% | 78% |
| F1-Score | **81%** | 68% | 79% | 77% |

*Table 6.1*

### Side by Side Comparison of my model with baseline DT, MNB and BNB

Using Integrated Stacked mechanism on the standard model to construct a new model shows significant improvement in the performance metrics. Comparing the same with the baseline of other standard models can be seen below (See Table 6.2)

| | Mohit's Model | Decision Tree | Bernoulli Naive Bayes | Multinomial Naive Bayes |
|---|---|---|---|---|
| Accuracy | **81%** | 70% | 69% | 74% |
| Precision | **81%** | 67% | 73% | 76% |
| Recall | **81%** | 70% | 69% | 74% |
| F1-Score | **81%** | 67% | 61% | 69% |

*Table 6.2*

The final performance metrics of my final model is also shown in the table below (See Table 6.3)

### Mohit's Model

| | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| **negative** | **0.85** | **0.92** | **0.88** | **628** |
| **neutral** | **0.70** | **0.60** | **0.65** | **210** |
| **positive** | **0.77** | **0.69** | **0.73** | **162** |
| **accuracy** | | | **0.81** | **1000** |
| **macro avg** | **0.78** | **0.74** | **0.75** | **1000** |
| **weighted avg** | **0.81** | **0.81** | **0.81** | **1000** |

*Table 6.3*