COMP9414 Assignment 2 report

Hang DONG z5227948

1. (1 mark) 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?

A: I used my program to calculate the total number of each sentiment and found that there are 882 positive tweets, 3115 negative tweets and 1063 neutral tweets in 5000 samples. The figure below shows the distribution.

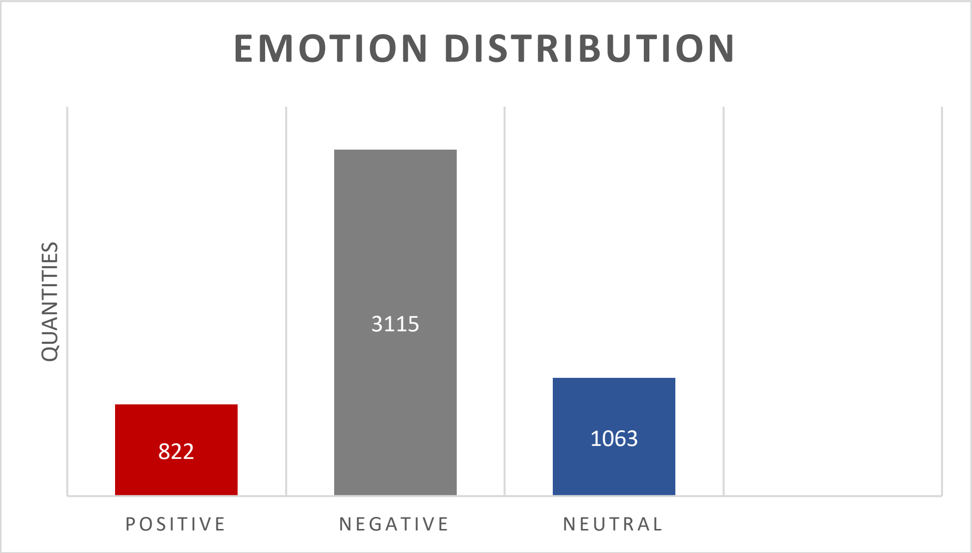
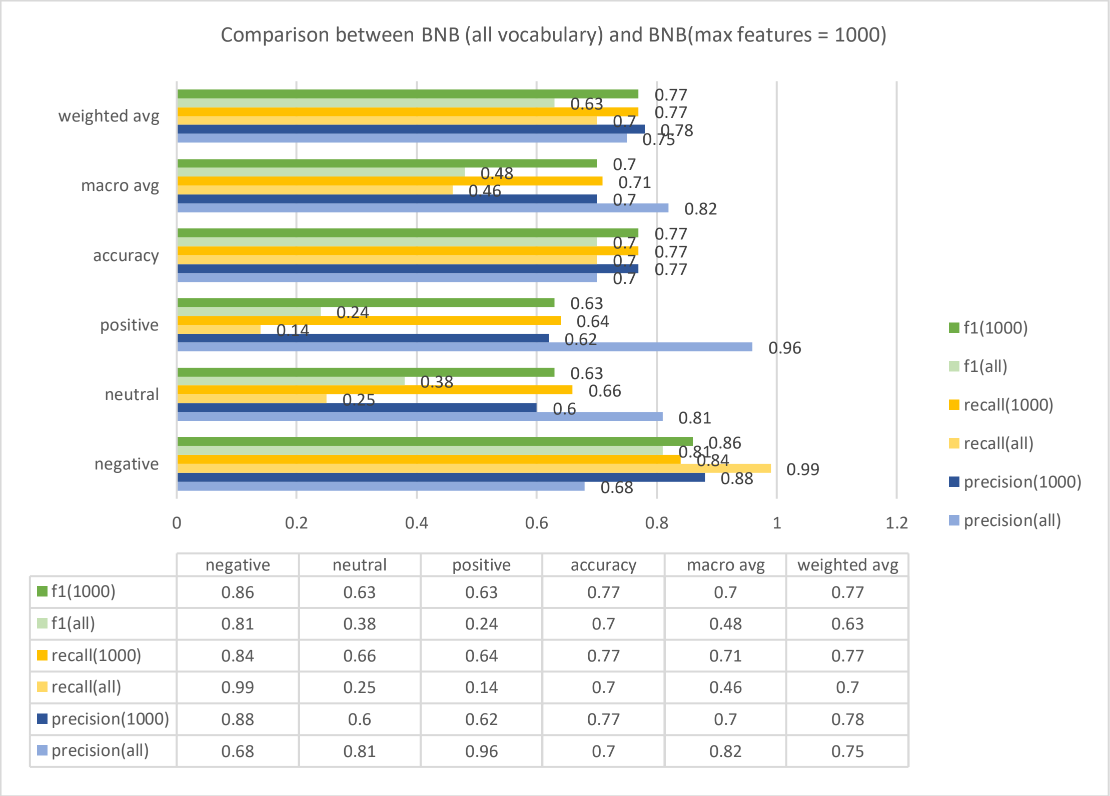


Figure 1. The frequency distribution for the sentiment

From figure 1, we found **majority** of sentiments are **negative** consisting of 62.3%. Neutral sentiment is around 21% and positive sentiment is the lowest, found to be16.4%. This means that most people (62.3%) that show negative emotion in the airlines experience feedback while the remaining subjects exhibits nearly the same amount of positive (21%) and neutral attitudes (16.4%).

2. (2 marks) Develop BNB and MNB models from the training set using (a) the whole vocabulary, and (b) the most frequent 1000 words from the vocabulary (as defined using CountVectorizer, after pre-processing by removing “junk” characters). Show all metrics on the test set comparing the two approaches for each method. Explain any similarities and differences in results.

A: By editing the max_features parameter in CountVectorizer(), the Figure 2 exemplifies how the precision, recall and f1 value changes by editing the max_feature.



	negative	neutral	positive	accuracy	macro avg	weighted avg
f1(1000)	0.86	0.63	0.63	0.77	0.7	0.77
f1(all)	0.81	0.38	0.24	0.7	0.48	0.63
recall(1000)	0.84	0.66	0.64	0.77	0.71	0.77
recall(all)	0.99	0.25	0.14	0.7	0.46	0.7
precision(1000)	0.88	0.6	0.62	0.77	0.7	0.78
precision(all)	0.68	0.81	0.96	0.7	0.82	0.75

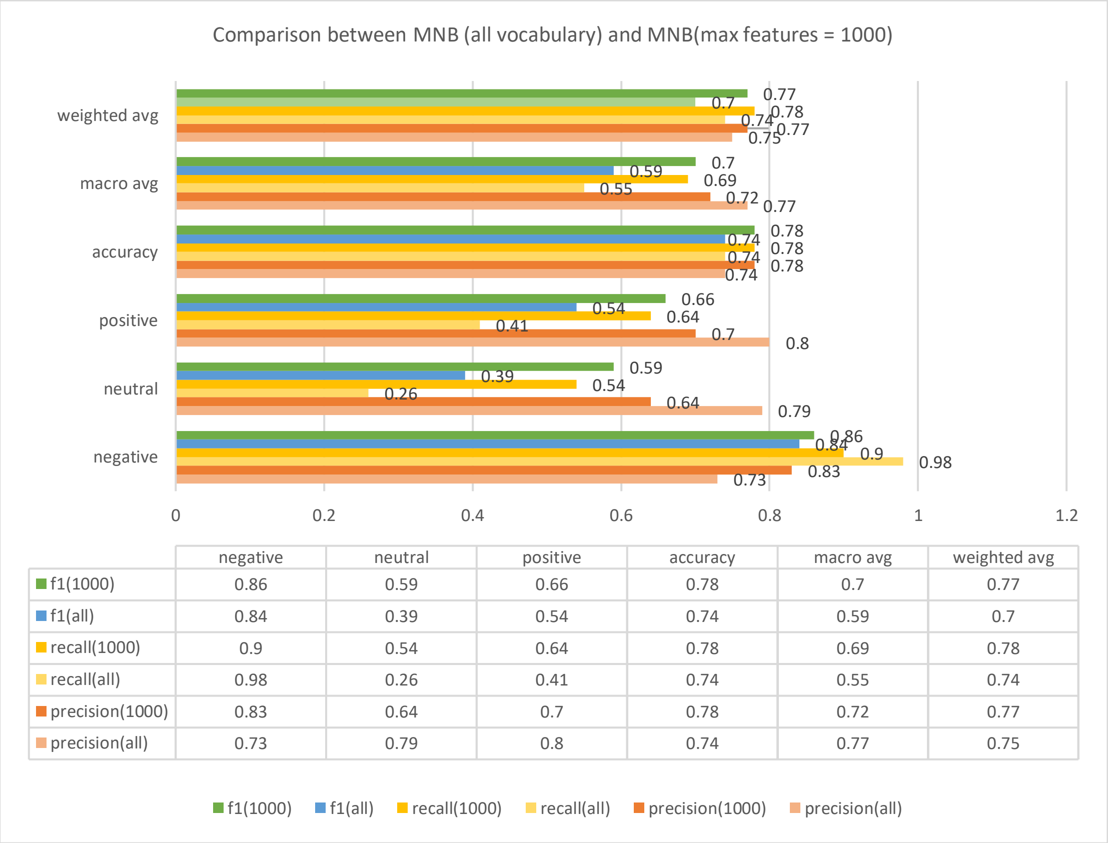
Figure 2. Comparison between BNB (all vocabulary) and BNB (max features = 1000)

From Figure 2,for the **precision performance** (represented by the blue bar in the graph), after editing the max_features from default to 1000, except for negative, the precision for the rest of the emotions decreased, while negative's shows major increased(from 0.68 to 0.88). Opposingly, the **change of recall** demonstrates an opposite situation as the negative slightly decreased (from 0.99 to 0.84) while positive and neutral sentiments displayed a huge increase (from 0.14 to 0.64 and from 0.25 to 0.66 respectively).

For **accuracy,** after the parameter was edited, the accuracy increased from 0.7 to 0.77 which means that this operation enabled a better overall prediction.

For the avg part, the macro avg of precision decreased by 0.1 and macro avg of recall increased by 0.25 which is significant while the f1 of precision increased by 0.22 which is major too; The weighted avg of precision and recall increased slightly while the weighted avg of f1 increased by 0.14.

The reason why the accuracy increased is that we **limited the max features value to 1000** which allowed the entire model to be **more compact** and to **reduce the interference of less frequent words** for the overall prediction.



	negative	neutral	positive	accuracy	macro avg	weighted avg
🟩 f1(1000)	0.86	0.59	0.66	0.78	0.7	0.77
🟦 f1(all)	0.84	0.39	0.54	0.74	0.59	0.7
🟨 recall(1000)	0.9	0.54	0.64	0.78	0.69	0.78
🟨 recall(all)	0.98	0.26	0.41	0.74	0.55	0.74
🟧 precision(1000)	0.83	0.64	0.7	0.78	0.72	0.77
🟧 precision(all)	0.73	0.79	0.8	0.74	0.77	0.75

Figure 3. Comparison between MNB (all vocabulary) and MNB (max features = 1000)

From Figure 3, first look at the precision performance, after setting the max features to 1000, the negative precision increased whilst the neutrals' and positives' decreased.

For the recall performance, it demonstrated a totally opposite situation where the negative recall declined whilst neutrals and positives inclined.

For the f1 performance, the f1 of all 3 sentiments exemplified an inclined trend.

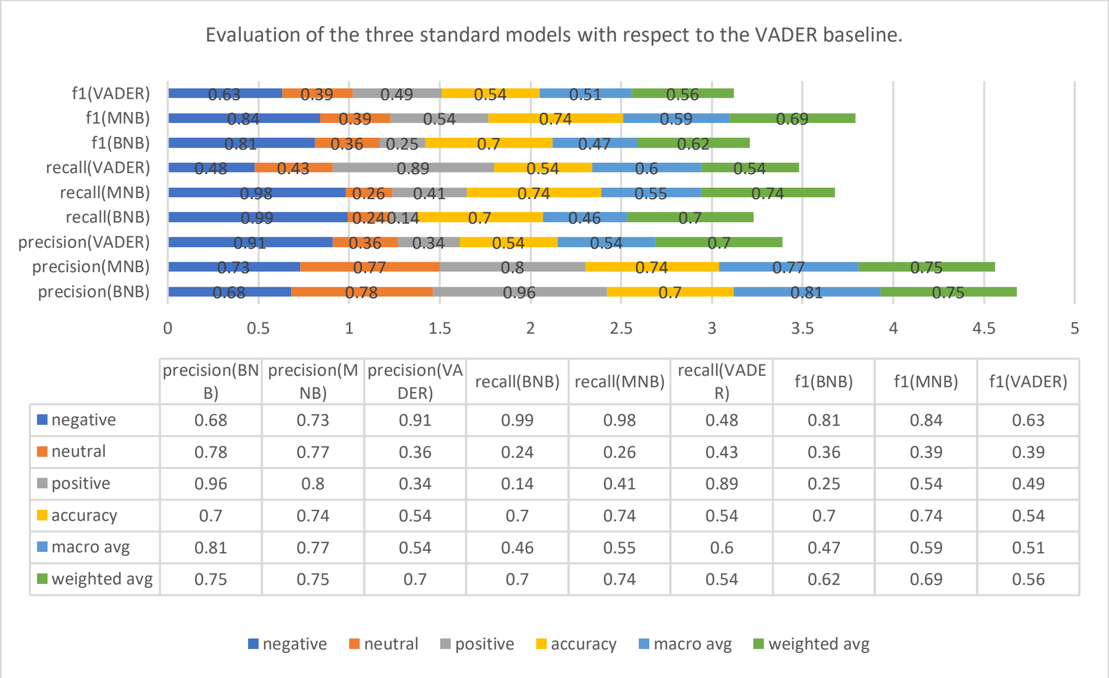
And at last, the accuracy increased from 0.74 to 0.78.

For the avg part, the macro avg of precision decreased by 0.05 and macro avg of recall increased by 0.14 whilst the f1 of precision increased by 0.21, both exhibiting major increases. The weighted avg of precision, recall and f1 score all increased slightly.

The reason why the accuracy was increased is that we limited the max features value to 1000 which led the entire model to be more **compact** and **reduced the interference** of less frequent words on the entire prediction.

3. (2 marks) 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.

A: By implement the **VADER** model and inputting the test sentence, finally analysing it. The Figure 4 displays the Evaluation of the three standard models with respect to the VADER baseline.



	precision(BNB)	precision(MNB)	precision(VADER)	recall(BNB)	recall(MNB)	recall(VADER)	f1(BNB)	f1(MNB)	f1(VADER)
■ negative	0.68	0.73	0.91	0.99	0.98	0.48	0.81	0.84	0.63
■ neutral	0.78	0.77	0.36	0.24	0.26	0.43	0.36	0.39	0.39
■ positive	0.96	0.8	0.34	0.14	0.41	0.89	0.25	0.54	0.49
■ accuracy	0.7	0.74	0.54	0.7	0.74	0.54	0.7	0.74	0.54
■ macro avg	0.81	0.77	0.54	0.46	0.55	0.6	0.47	0.59	0.51
■ weighted avg	0.75	0.75	0.7	0.7	0.74	0.54	0.62	0.69	0.56

Figure 4. Evaluation of the three standard models with respect to the VADER baseline.

From Figure 4, comparing the performance of BNB and MNB, for the negative prediction, we can find that the precision of VADER has been improved (from 0.73/0.68 to 0.91) whilst the recall of VADER significantly declined (from 0.98/0.99 to 0.46) which dominated the f1 score and resulted to a decrease in f1 of VADER (from 0.84/0.81 to 0.61).

For the neutral prediction, the precision of VADER declined significantly, and the recall of VADER increased slightly whilst the f1 of VADER has not changed much.

For the negative prediction, a huge decline occurred on the precision of VADER and the recall of VADER largely increased while the f1 of VADER is better than BNB's and worser than MNB's.

For the accuracy, after using the model of VADER, we found that compared with the other two models, VADER performed much worse (VADER: 0.52, BNB: 0.7, MNB: 0.74).

For the macro avg part, the macro avg precision of VADER declined (from 0.81/0.77 to 0.53) whilst the macro avg of recall of VADER was worser than MNB's however better than BNB's. The macro f1 of VADER is smaller than MNB's and slightly bigger than BNB's.

For the weighted avg part, the weighted avg precision of VADER is smaller than BNB's and MNB's whilst the weighted avg of recall and weighted avg of f1 of VADER shows the same result.

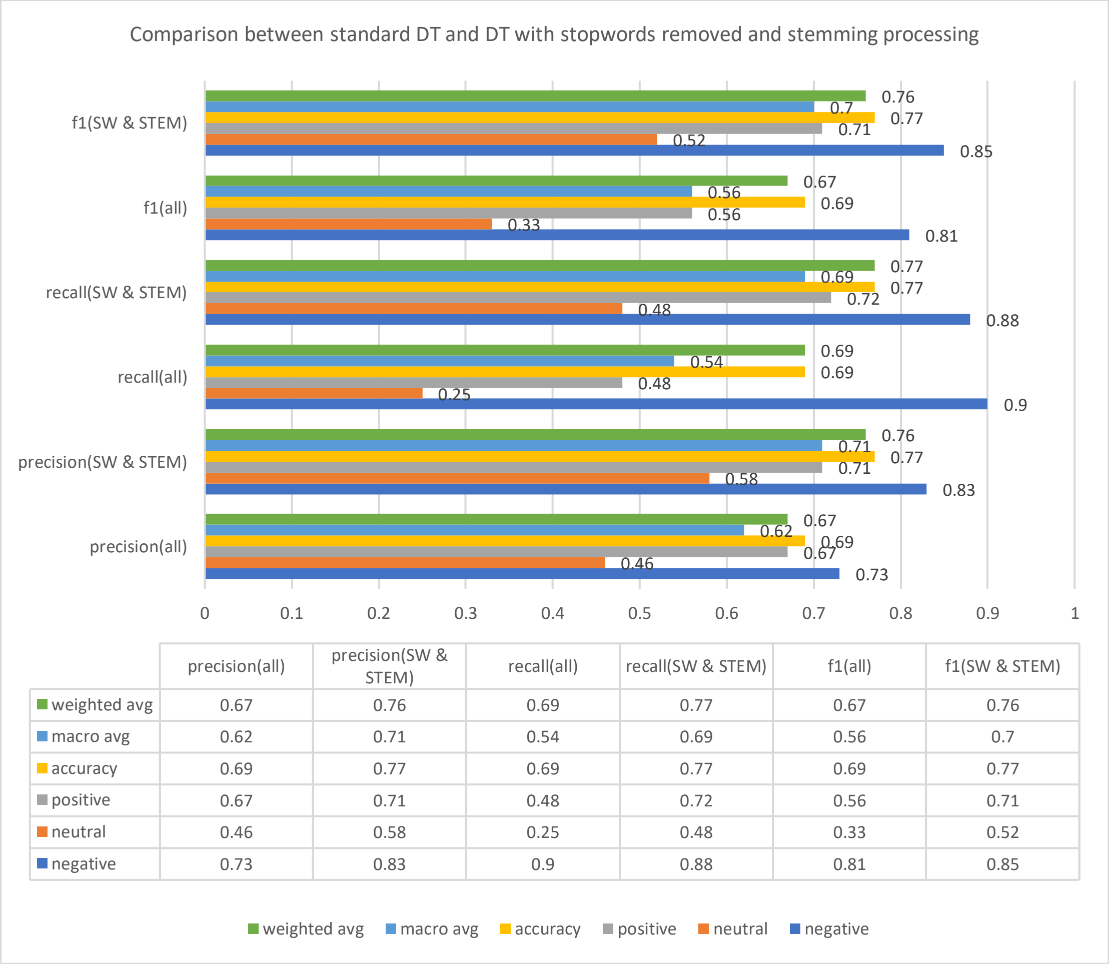
The reason why VADER performed so poor is that in the data (training and test sets) sentences, the tweets did not use many emojis or any social media features to express emotions.

Another reason is that the crowd-sourcing like VADER is not reliable.

4. (2 marks) Evaluate the effect of pre-processing 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 pre-processing on the test set and explain the results.

A: After removing the stop words and implementing the stemming words function to the 3-standard model, the figures below shows the comparison of 3 basic model performance and 3 model with removed stop words and stemmed words processing.



	precision(all)	precision(SW & STEM)	recall(all)	recall(SW & STEM)	f1(all)	f1(SW & STEM)
🟩 weighted avg	0.67	0.76	0.69	0.77	0.67	0.76
🟦 macro avg	0.62	0.71	0.54	0.69	0.56	0.7
🟨 accuracy	0.69	0.77	0.69	0.77	0.69	0.77
⬜ positive	0.67	0.71	0.48	0.72	0.56	0.71
🟧 neutral	0.46	0.58	0.25	0.48	0.33	0.52
🟦 negative	0.73	0.83	0.9	0.88	0.81	0.85

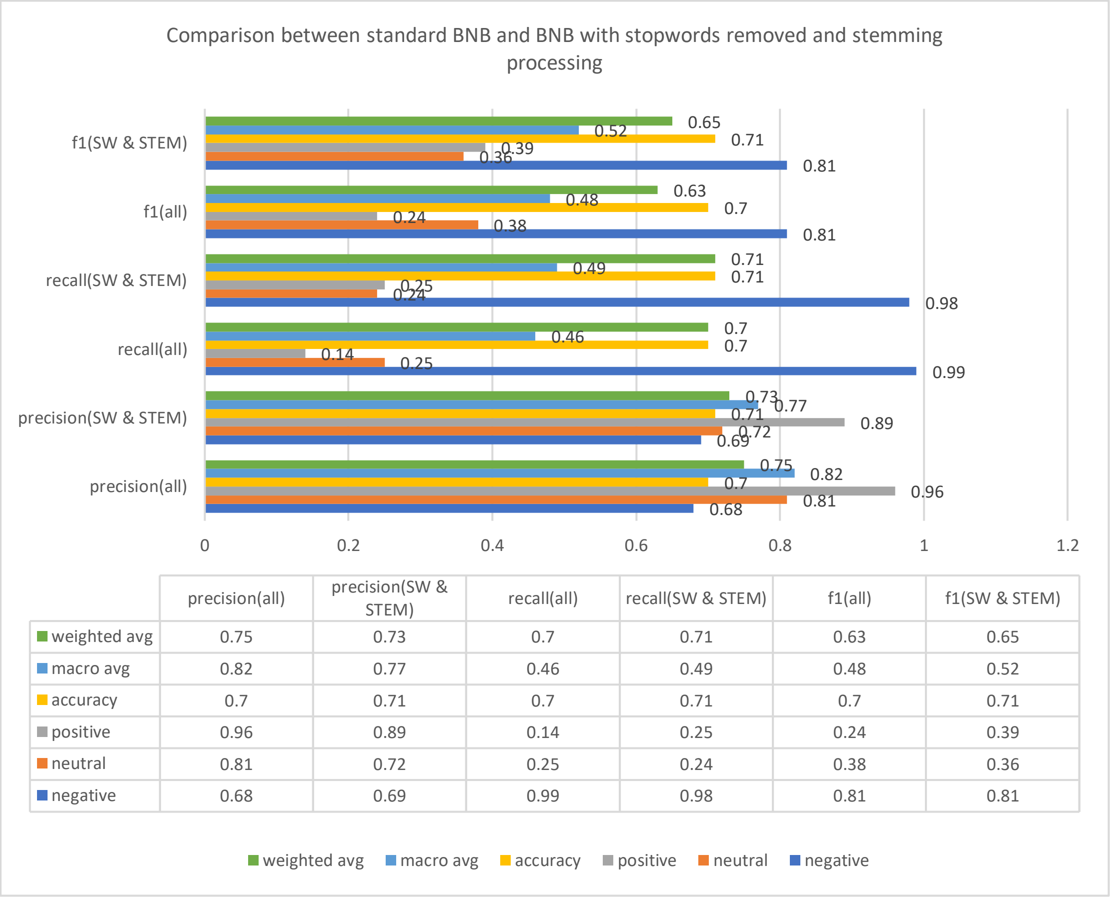
Figure 5. Comparison between standard DT and DT with stop words removed and stemming processing ('all' stand for standard DT with full vocabulary and 'SW & STEM' stand for DT with stop words and stemming preprocessing)

From Figure 5, after implementing the stop words removal and stemming sentence pre-processing, we can clearly find that the negative's precision distinctly increased from 0.73 to 0.83 and negative's f1 slightly increased from 0.81 to 0.85 whilst the negative's recall only decreased by 0.02.

For the neutral and positive part, the 3-evaluation index (precision, recall and f1); the weighted avg and macro avg all have risen to varying degrees.

The accuracy increased from 0.69 to 0.77.

So, this pre-processing shows that the improvement effect on DT is big.



	precision(all)	precision(SW & STEM)	recall(all)	recall(SW & STEM)	f1(all)	f1(SW & STEM)
weighted avg	0.75	0.73	0.7	0.71	0.63	0.65
macro avg	0.82	0.77	0.46	0.49	0.48	0.52
accuracy	0.7	0.71	0.7	0.71	0.7	0.71
positive	0.96	0.89	0.14	0.25	0.24	0.39
neutral	0.81	0.72	0.25	0.24	0.38	0.36
negative	0.68	0.69	0.99	0.98	0.81	0.81

Figure 6. Comparison between standard BNB and BNB with stop words removed and stemming processing ('all' stand for standard BNB with full vocabulary and 'SW & STEM' stand for BNB with stop words and stemming preprocessing)

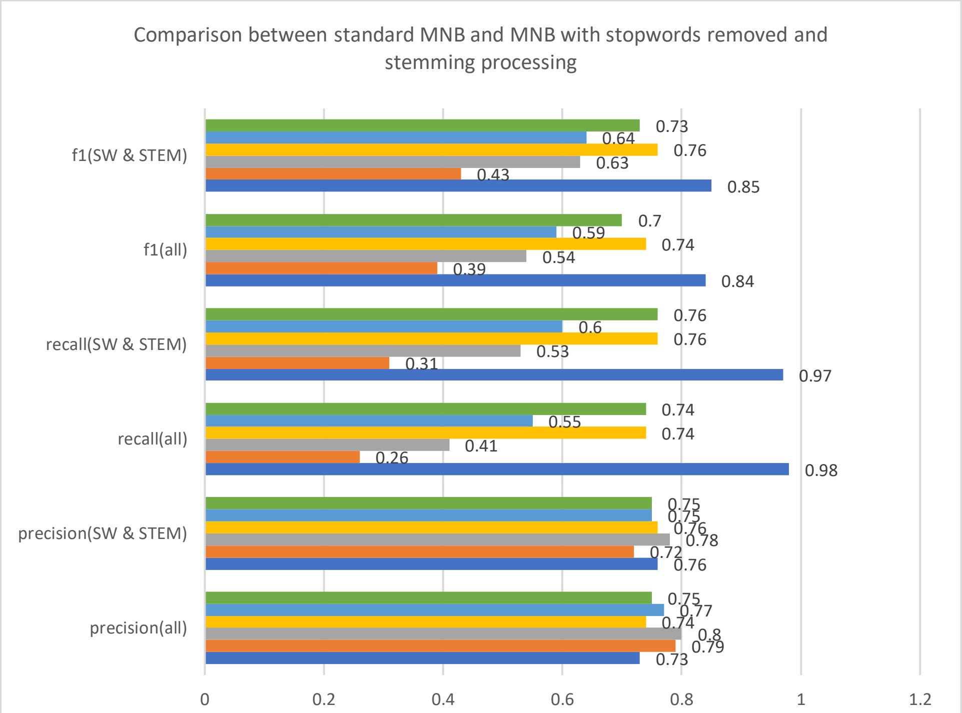
From Figure 6, for the negative part, after pre-processing, the precision and recall just increased slightly and the f1 almost stayed the same.

The neutral and positive parts share a common trend to negative.

It was also found that except that the weighted avg and macro avg of precision decreased slightly, the recall and f1 score's macro avg and weighted increased a little bit.

And at last, the accuracy increased from 0.7 to 0.71 which is small.

So, this pre-processing shows that the improvement effect on BNB is small.



	precision(all)	precision(SW & STEM)	recall(all)	recall(SW & STEM)	f1(all)	f1(SW & STEM)
weighted avg	0.75	0.75	0.74	0.76	0.7	0.73
macro avg	0.77	0.75	0.55	0.6	0.59	0.64
accuracy	0.74	0.76	0.74	0.76	0.74	0.76
positive	0.8	0.78	0.41	0.53	0.54	0.63
neutral	0.79	0.72	0.26	0.31	0.39	0.43
negative	0.73	0.76	0.98	0.97	0.84	0.85

Figure 7. Comparison between standard DT and DT with stop words removed and stemming processing ('all' stand for standard MNB with full vocabulary and 'SW & STEM' stand for MNB with stop words and stemming preprocessing)

From figure 7, for the negative part, after pre-processing, the precision and recall just increased slightly and the f1 is almost the same.

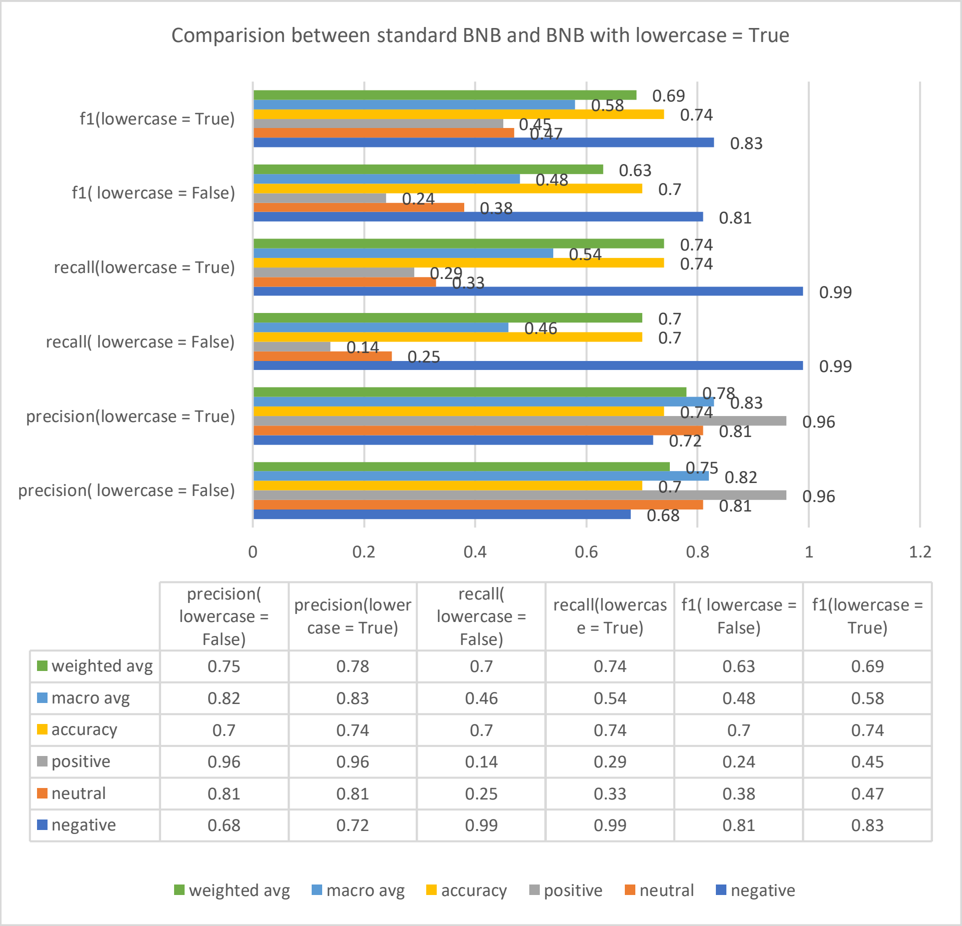
The neutral and positive part share a common trend to negative.

We can also find that except the weighted avg and macro avg of precision decreased slightly, the recall and f1 score's macro avg and weighted increased a little bit.

And at last, the accuracy increased from 0.74 to 0.76 which is small.

So, this pre-processing operation shows that the improvement effect on MNB is small but better than BNB.

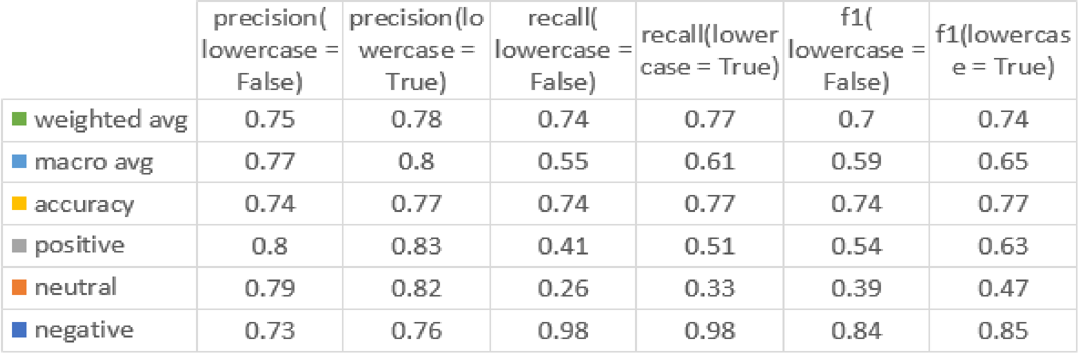
5. (2 marks) Evaluate the effect that converting all letters to lower case has on classifier performance for the three standard models. Show all metrics with and without conversion to lower case on the test set and explain the results.



	precision(lowercase = False)	precision(lower case = True)	recall(lowercase = False)	recall(lowercas e = True)	f1(lowercase = False)	f1(lowercase = True)
🟩 weighted avg	0.75	0.78	0.7	0.74	0.63	0.69
🟦 macro avg	0.82	0.83	0.46	0.54	0.48	0.58
🟨 accuracy	0.7	0.74	0.7	0.74	0.7	0.74
⬜ positive	0.96	0.96	0.14	0.29	0.24	0.45
🟧 neutral	0.81	0.81	0.25	0.33	0.38	0.47
🟦 negative	0.68	0.72	0.99	0.99	0.81	0.83

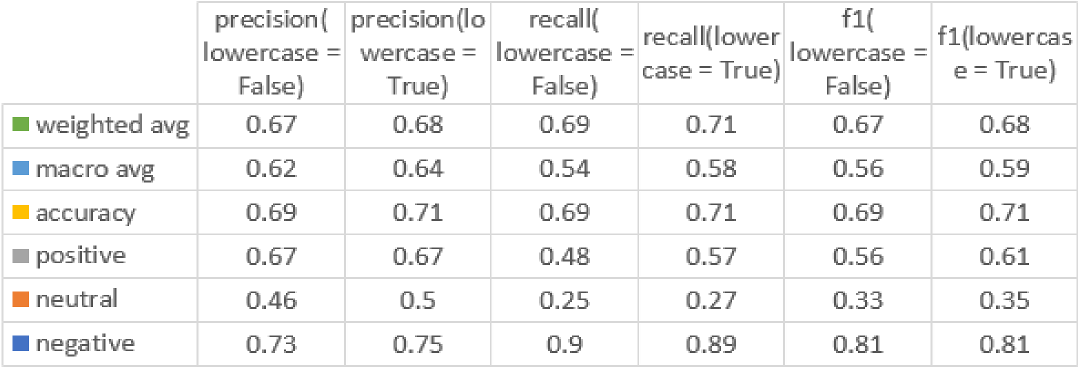
Figure 8. Comparison between standard BNB and BNB with lowercase = True

From Figure 8, most of the data (precision, recall, and f1) showed different degrees of increase, a few remained unchanged, and there was basically no decline in data.
And the accuracy increased from 0.7 to 0.74.
This is means remaining the uppercase and lowercase format can help the BNB model to perform better.



	precision(lowercase = False)	precision(lo wercase = True)	recall(lowercase = False)	recall(lower case = True)	f1(lowercase = False)	f1(lowercas e = True)
🟩 weighted avg	0.75	0.78	0.74	0.77	0.7	0.74
🟦 macro avg	0.77	0.8	0.55	0.61	0.59	0.65
🟨 accuracy	0.74	0.77	0.74	0.77	0.74	0.77
⬜ positive	0.8	0.83	0.41	0.51	0.54	0.63
🟧 neutral	0.79	0.82	0.26	0.33	0.39	0.47
🟦 negative	0.73	0.76	0.98	0.98	0.84	0.85

Figure 9. Comparison between standard MNB and MNB with lowercase = True



	precision(lowercase = False)	precision(lowercase = True)	recall(lowercase = False)	recall(lowercase = True)	f1(lowercase = False)	f1(lowercase = True)
🟩 weighted avg	0.67	0.68	0.69	0.71	0.67	0.68
🟦 macro avg	0.62	0.64	0.54	0.58	0.56	0.59
🟨 accuracy	0.69	0.71	0.69	0.71	0.69	0.71
◼ positive	0.67	0.67	0.48	0.57	0.56	0.61
🟧 neutral	0.46	0.5	0.25	0.27	0.33	0.35
🟦 negative	0.73	0.75	0.9	0.89	0.81	0.81

Figure 10. Comparison between standard MNB and MNB with lowercase = True

From figure 9 and 10, we can find that after setting the lowercase = True, MNB & DT performance shared a common trend as BNB. Most of the data (precision, recall, and f1) displayed different degrees of increase, a few remained unchanged, and there was barely any decline in data.

The accuracy increased from 0.74 to 0.77.

This is means that remaining the uppercase and lowercase format can help MNB & DT model to perform better.

6. (6 marks) 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).

According to the analysis above, we can find that: 1. Setting the lowercase to True; 2. Using the MNB model; 3. Find the best max_features value to obtain the best accuracy.

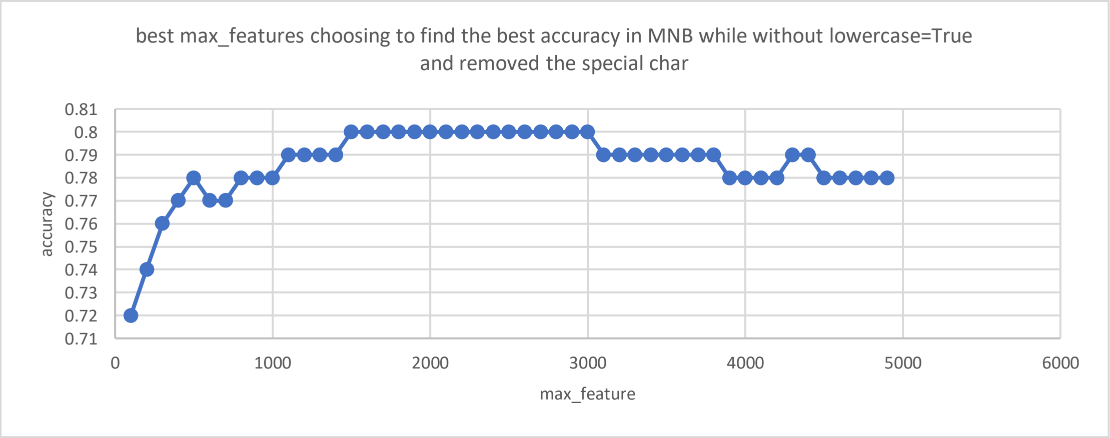
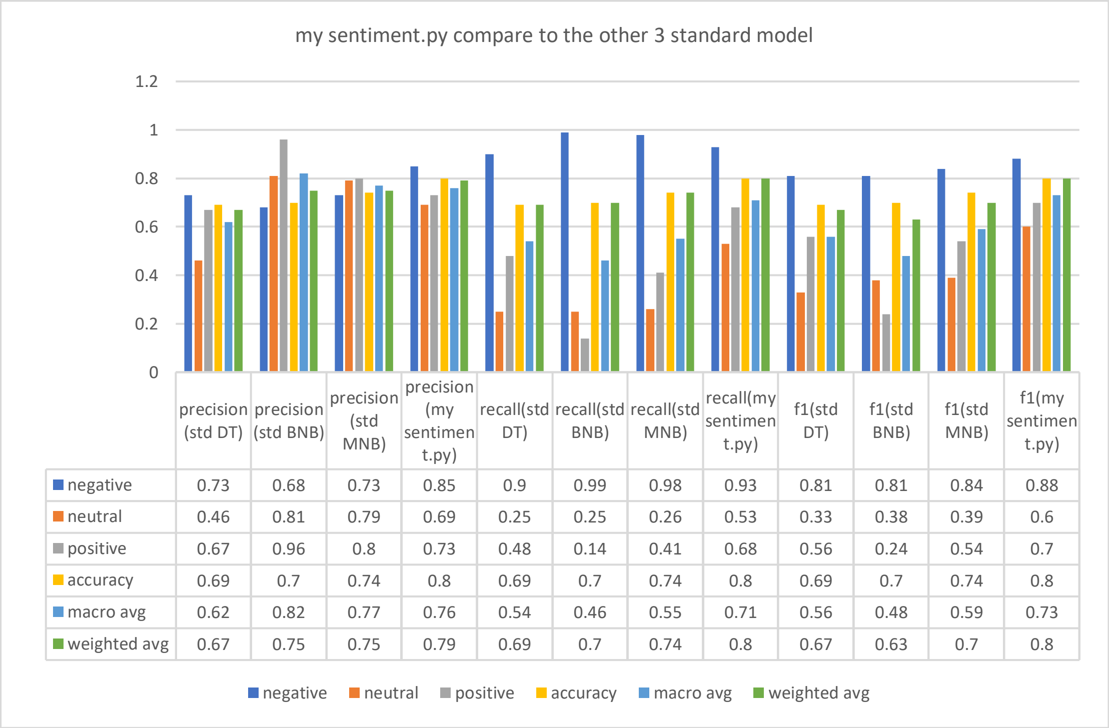


Figure 11. Best max_features value choosing to find the best accuracy in MNB while without lowercase=True and removed the special char

From the Figure 11, we can find that the best performance max_feature value is around 1600 – 3000, so one is chosen in this duration.



	precision (std DT)	precision (std BNB)	precision (std MNB)	precision (my sentimen t.py)	recall(std DT)	recall(std BNB)	recall(std MNB)	recall(my sentimen t.py)	f1(std DT)	f1(std BNB)	f1(std MNB)	f1(my sentimen t.py)
🟦 negative	0.73	0.68	0.73	0.85	0.9	0.99	0.98	0.93	0.81	0.81	0.84	0.88
🟧 neutral	0.46	0.81	0.79	0.69	0.25	0.25	0.26	0.53	0.33	0.38	0.39	0.6
⬜ positive	0.67	0.96	0.8	0.73	0.48	0.14	0.41	0.68	0.56	0.24	0.54	0.7
🟨 accuracy	0.69	0.7	0.74	0.8	0.69	0.7	0.74	0.8	0.69	0.7	0.74	0.8
🟦 macro avg	0.62	0.82	0.77	0.76	0.54	0.46	0.55	0.71	0.56	0.48	0.59	0.73
🟩 weighted avg	0.67	0.75	0.75	0.79	0.69	0.7	0.74	0.8	0.67	0.63	0.7	0.8

Figure 12. My sentiment.py performance compared with the other 3 model

From Figure 12, we can find that my sentiment.py have the best negative precision performance while the positive and neutral precision is in 3rd place. For the recall and f1 part, my sentiment.py performed well in each part.

For the macro avg, my sentiment.py is 0.8 behind first place while the recall and f1 score value is in 1st place.

For the weighted avg, my sentiment.py have the best performance in precision, recall and f1 score.

For the accuracy, my sentiment.py is 0.8 while the other 3 model (DT, BNB, MNB) is 0.69, 0.7, 0.74 which respectively improved the model and made the prediction to be more accurate.

The reason why my model has not improved greatly may be: 1. Too few training samples 2. Limited choice of models, other models will perform better.

P.S: 1. Actually, I have implemented stopwords removing, nlp() processing to the sentences and stemming sentences in my sentiment.py, but the final result is worser, so I decide not to use them.

2. I also implementing a TfidfVectorizer in my sentiment.py, but the accuracy result is lower than countVectorizer, so I give up using it.

3. I implementing some extra data cleaning process in my sentiment.py, like 'remove_backslash_n' can remove '\\n' in tweets, remove_IP can remove some ip strings and remove_Startwith_User can remove some string startwith user. These processes can make the model predict more accurate.