Assignment 2 Report

Patrick Li z5180847

1.

The proportion of each class is represented in the table below, which shows that the dataset is not balanced since the negative class takes the majority. Since the ratio of number of samples of positive: negative: neutral is nearly 4:16:5, the data is not really imbalanced, I will implement accuracy score instead of balanced accuracy score.

	Training set (the first 4000 tweets)	Test set (the last 1000 tweets)	Total
Positive	660 (16.5%)	162 (16.2%)	822 (16.4%)
Negative	2487 (62.2%)	628 (62.8%)	3115 (62.3%)
neutral	853 (21.3%)	210 (21.0%)	1063 (21.3%)

Table 1 Proportion of each class

The visualized distribution is shown below. As shown in the chart, highly used words are usually prepositions such as “to”, “for” and “on”, pronouns such as “you”, “it” and “my”, some verbs such as “have”, “get” and “be”, and articles “the”. Besides those words highly used in everyone’s daily life, some user names such as “@united”, “@AmericanAir”, and “USAirways” and some nouns related to the task such as “flight”, “service” and “customer”.

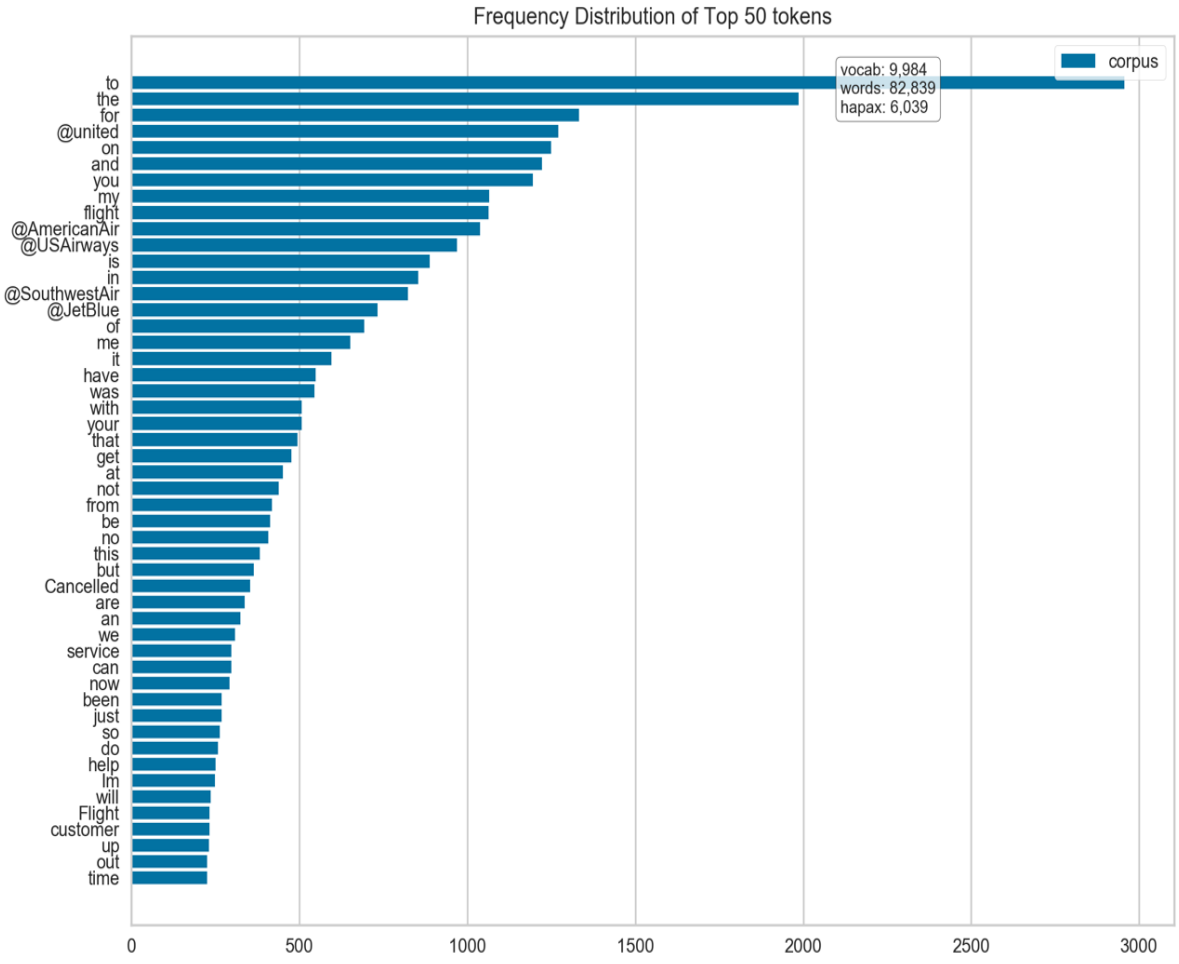

Figure 1 Distribution of the top 50 frequent tokens

2.

	BNB				MNB			
	The whole vocabulary		The most frequent 1000 words		The whole vocabulary		The most frequent 1000 words	
	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro
Accuracy	0.687	0.687	0.764	0.764	0.737	0.737	0.774	0.774
Precision	0.687	0.786	0.764	0.695	0.737	0.782	0.774	0.713
Recall	0.687	0.448	0.764	0.714	0.737	0.544	0.774	0.685
F1 Score	0.687	0.450	0.764	0.703	0.737	0.584	0.774	0.698

Table 2 Evaluation of BNB and MNB models

For both BNB and MNB models, it performs better when taking the most frequent 1000 words as training set features, instead of using the whole vocabulary which contains 9984 words according to the graph in question 1. The reason might be that the whole vocabulary contains some "junk" words which cannot help the models learn better. They maybe twitter users' user names, some infrequent hashtags or incorrectly spelled words. The models would have higher accuracy scores without these noises.

3.

The evaluation metrics of the three models using the most frequent 1000 words are shown below.

	VADER (Baseline)			DT		
	Micro	Macro	Weighted	Micro	Macro	Weighted
Accuracy	**0.529**	0.529	0.529	**0.696**	0.696	0.696
Precision	0.529	0.531	0.697	0.696	0.623	0.667
Recall	0.529	0.589	0.529	0.696	0.543	0.696
F1 score	0.529	**0.497**	0.552	0.696	**0.564**	0.667

Table 3 Evaluation metrics of the VADER and Decision Tree model

	BNB			MNB		
	Micro	Macro	Weighted	Micro	Macro	weighted
Accuracy	**0.764**	0.764	0.764	**0.774**	0.774	0.774
Precision	0.764	0.695	0.773	0.774	0.713	0.766
Recall	0.764	0.714	0.764	0.774	0.685	0.774
F1 score	0.764	**0.703**	0.768	0.774	**0.698**	0.769

Table 4 Evaluation metrics of the BNB and MNB model

Since the number of samples for each class in the test set is not balanced, the "positive" class has 162 samples out of 1000, the "negative" class has 628 and the "neutral" class has 210. Compared with micro and weighted F1 score, the macro one could show the performance more appropriately without favouring the majority class. As shown in the table above, the VADER has the lowest accuracy and macro F1 score. The VADER model's performance is as expected since crowd-sourcing is in general highly unreliable and this dataset might not include mush use of emojis and other markers of sentiment.

4.

Since weighted metrics are not suitable for the task, I will just compare micro and macro metrics.

	VADER (Baseline)		DT		BNB		MNB	
	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro
Accuracy	**0.471**	0.471	**0.702**	0.702	**0.782**	0.782	**0.776**	0.776
Precision	0.471	0.504	0.702	0.636	0.782	0.724	0.776	0.717
Recall	0.471	0.557	0.702	0.595	0.782	0.731	0.776	0.696
F1 score	0.471	**0.459**	0.702	**0.611**	0.782	**0.727**	0.776	**0.704**

Table 5 Evaluation of all the models with pre-processing

Compared with the results without stop word removal and NLTK Porter stemming, shown in question 3, all the models perform better with pre-processing except the VADER model. Increased accuracy and macro F1 score shows that with stop word removal and NLTK Porter stemming, redundant and meaningless words are removed so that models can focus more on those valuable words.

5.

	VADER (Baseline)		DT		BNB		MNB	
	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro
Accuracy	**0.53**	0.53	**0.708**	0.708	**0.787**	0.787	**0.786**	0.786
Precision	0.53	0.531	0.708	0.636	0.787	0.719	0.786	0.726
Recall	0.53	0.589	0.708	0.578	0.787	0.740	0.786	0.706
F1 score	0.53	**0.497**	0.708	**0.593**	0.787	**0.728**	0.786	**0.714**

Table 6 Evaluation of all the models with lowering words

Compared with the results without converting all words to lower case, performance of all the models are improved. The reason might be that it removed some redundant words and merged them to one word. For example, in the graph in question 1, “Flight” and “flight” are both in the top 50 tokens, and they would merge into just “flight” after lowering all words in the dataset.

6.

I implemented a multi-layer perceptron classifier, a random forest classifier and a SGD classifier from scikit-learn. For each model, a randomized search was used for hyperparameters tunning. The evaluation metrics and hyperparameters of the three models are shown below.

MLP		
Hyperparameter	Value	Description
hidden_layer_sizes	(50,)	One hidden layer with 50 nodes.
activation	tanh	Use the hyperbolic tan function as the activation function for the hidden layer.
solver	sgd	Stochastic gradient descent .
alpha	0.9655059579094672	L2 penalty (regularization term) parameter.
learning_rate	invscaling	Gradually decreases the learning rate at each time ‘t’. *effective_learning_rate = learning_rate_init/ pow(t, power_t)*
Learning_rate_init	0.44079441387656165	The initial learning rate used.

Table 7 Hyperparameters of the MLP model

Random Forest		
Hyperparameter	Value	Description
n_estimators	142	The number of trees in the forest.
criterion	entropy	Use information gain to measure the quality of a split.
max_features	auto	The number of features to consider when looking for the best split is *max_features = sqrt(n_features)*
bootstrap	False	No bootstrap samples are used when building trees.
warm_start	True	Reuse the solution of the previous call to fit and add more estimators to the ensemble.

Table 8 Hyperparameters of the Random Forest model

SGD		
Hyperparameter	Value	Description
loss	squared_hinge	Gives a linear SVM but is quadratically penalized.
penalty	l2	The penalty (regularization term) to be used.
learning_rate	adaptive	eta = eta0, as long as the training keeps decreasing.
eta0	0.007870258748593513	The initial learning rate for the 'adaptive' schedules.

Table 9 Hyperparameters of the SGD model

	MLP		Random Forest		SGD	
	Micro	Macro	Micro	Macro	Micro	Macro
Accuracy	**0.827**	0.827	**0.791**	0.791	**0.802**	0.802
Precision	0.827	0.790	0.791	0.781	0.802	0.753
Recall	0.827	0.753	0.791	0.658	0.802	0.735
F1 score	0.827	**0.769**	0.791	**0.699**	0.802	**0.743**

Table 10 Evaluation metrics of the models

MLP is a kind of neutral network, which consists of an input layer, at least one hidden layer and an output layer. It uses non-linear activation function making it can handle with non-linear separable data. Since MLPs are often used to solve really complex problems, I think it might have a good performance in this sentiment analysis task.

Random forest is an ensemble learning method, which will generate multitude decision trees on randomly selected data in training and output the best solution by means of voting. It is robust and usually has a high accuracy.

Stochastic Gradient Descent (SGD) is a simple yet efficient optimization algorithm. It is used for discriminative learning of linear classifiers under convex loss functions such as SVM and Logistic regression.
Among the three, the MLP has the best performance with an accuracy of 82.7% and macro F1 score of 0.769 as I expected, followed by the SGD with an accuracy of 80.2% and macro F1 score of 0.743. The random forest model has an accuracy of 79.1% and macro F1 score of 0.699 ranking the last.

To experiment the effect of pre-processing, I also conducted several pre-processing method on the MLP model with tuned hyperparameters to find a better performance:

1) Without stop word removal, NLTK Porter stemming or converting all words to lower case. The result is shown in the table below.

	Micro	Macro
Accuracy	**0.801**	0.801
Precision	0.801	0.760
Recall	0.801	0.704
F1 score	0.801	**0.727**

Table 11 MLP without any pre-processing

2) With only stop word removal.

	Micro	Macro
Accuracy	**0.787**	0.787
Precision	0.787	0.748
Recall	0.787	0.707
F1 score	0.787	**0.724**

Table 12 MLP with only stop word removal

3) With stop word removal and NLTK Porter stemming.

	Micro	Macro
Accuracy	**0.803**	0.803
Precision	0.803	0.770
Recall	0.803	0.722
F1 score	0.803	**0.742**

Table 13 MLP with stop word removal and NLTK Porter stemming

4) With stop word removal and converting all words to lower case.

	Micro	Macro
Accuracy	**0.800**	0.800
Precision	0.800	0.770
Recall	0.800	0.717
F1 score	0.800	**0.740**

Table 14 MLP with stop word removal and converting words to lower case

5) With NLTK Porter stemming and converting all words to lower case.

	Micro	Macro
Accuracy	**0.827**	0.827
Precision	0.827	0.790
Recall	0.827	0.753
F1 score	0.827	**0.769**

Table 15 MLP with NLTK Porter stemming and convering words to lower case

6) With stop word removal, NLTK Porter stemming and converting all words to lower case.

	Micro	Macro
Accuracy	**0.802**	0.802
Precision	0.802	0.770
Recall	0.802	0.720
F1 score	0.802	**0.741**

Table 16 With all three pre-processing methods

From the results above, with NLTK Porter stemming and converting all words to lower case, the MLP reached the highest accuracy and macro F1 score. The reason why accuracy and macro F1 score dropped after adding stop word removal might be that the stop words set from NLTK is not suitable for this task which removes some valuable words from the dataset.

The table below compares the result of MLP from table 15 with the best results of all three standard models and VADER baseline from table 6.

	VADER	DT	BNB	MNB	MLP
Accuracy	0.53	0.708	0.787	0.786	0.827
Macro F1 score	0.497	0.593	0.728	0.714	0.769

Table 17 Compare the MLP model with the standard models and the baselien

The MLP model has the highest accuracy and macro F1 score among all models implemented, as I expected before the experiment since MLP has a neural network structure implemented for complex classification task. However, the accuracy is not as high as I expected after hyperparameter tunning.