Parsa Yadollahi
COMP 550 - Assignment 1
260869949

Problem setup:
For the purpose of this exercise, I split the training and testing data into 80% respectively. I split the data into sentences to create a dataset and put these into an array and randomize the order. For the basic preprocessing, I cleaned the data (sentences) by removing all extra white spaces, replacing all capitalized characters for their respective lower case. Using the Natural Language toolkit (nltk) library, I tokenized the sentences and removed all stopwords as well as all punctuations. Next, I implemented custom preprocessors that stem and lemmatize the sentences using Porter Stemmer and WordNet corpus. I then create a feature vector for each sentence using CountVectorizer with a specified preprocessor from the sklearn.feature_extraction.text library. I proprocessed the testing data in the same fashion but used transformed it to fit the features from the training data.

Experimental Procedure:
I ran the experiment on four different models and random baseline. The four models given by the assignment were Logistic Regression, Support Vector Machine, Naive-Bayes and, the optional classifier, random forest classifier. The data seen above was acquired by reading the positive and negative data from the files provided, randomized and split into 80/20 - training/testing. Once the files were read, the training data was fed into these five classifiers and then evaluated against the testing data to return their accuracy.

Range of parameter settings:
I kept the training and testing data constant throughout the different runs as well as basic preprocessing such as stopwords the same. I changed 3 parameters between each run while testing the model. The first is “min_df” which is used for removing terms that appear too infrequently, the second is “max_df which is used for removing terms that appear too frequently, also known as "corpus-specific stop words". For example, adding a value of 2 to min_df and max_df would remove all words that either appear less than twice or appear more than twice respectively. In this case I went with decimals (0.5 and 0.8) which represents percentages for “max_df” which implies to ignore terms that appear in more than x% of the documents (where x is the value passed to max_df). And the final one was the preprocessing step for the feature, between using a lemmatizater or stemming.

Results:
The following table represents the performance in accuracy of the classifiers with respect to the parameters settings that we're changed.

Classifiers	Lemmatizing				Stemming			
	Min_df = 2		Min_df = 4		Min_df = 2		Min_df = 4	
	Max_df = 0.5	Max_df = 0.8	Max_df = 0.5	Max_df = 0.8	Max_df = 0.5	Max_df = 0.8	Max_df = 0.5	Max_df = 0.8
Naive-Bayes	0.702	0.693	0.717	0.712	0.678	0.679	0.694	0.732
Logistic Regression	0.747	0.754	0.743	0.753	0.751	0.753	0.767	0.752
Support Vector Machine	0.729	0.737	0.720	0.732	0.716	0.731	0.749	0.735
Random Baseline	0.482	0.499	0.508	0.521	0.499	0.519	0.496	0.497
Random Forest	0.719	0.729	0.706	0.719	0.700	0.720	0.727	0.715

Conclusion:
From the table above, we can conclude that, on average, the logistic regression classifier returns the best performance between all models with an accuracy of 0.767%. More specifically, there is a stronger performance when stemming each word instead of using a lemmatizer and when ignoring all words that appear less than 4 times (min_df = 4) and ignoring all words that appear in more than 50% of the documents (max_df = 0.5). For the test accuracy, all models performed within 70-77% excluding the random baseline classifier.

	Positive (T)	Negative (T)
Positive (P)	813	269
Negative (P)	279	772

Note: (P) denotes Predicted. (T) denotes True.

By looking at the confusion matrix of the logistic regression classifier, we note that there are more true positives then true negatives. We also note that the classifier predicted 813 true positive, 772 true negatives, 279 false positives and 269 false negatives. The false discovery of the model is still high. A solution to this could be tuning test-train split or to train the model with more data which increases the data set size or even to change the preprocessing and feature extraction steps on the data.