Ava McCormack 99121093
CIS4930 Individual Coding Assignment
Spring 2023

1. Problem Statement

In this assignment, I was given 2 datasets containing text from a social media platform and corresponding sentiment. Given the training data, I must design a sentiment analysis model that is able to detect sentiment given text data. Sentiment analysis in this context is categorized by whether the text data has a positive (1) or negative (0) sentiment. Sentiment analysis models are quite important especially as more and more aspects of life are found online through text. These models can be used to detect a customer's satisfaction or an online post's intentions. To achieve this goal, I used several different linguistic feature extraction methods and applied them to several different classification models.

2. Data Preparation

First, I read the 2 datasets from csv files into python. Next, I checked for null values of which there were none. A visual graph showed me that the sentiment of the training data was heavily negative. To combat this and the vast amount of data, I took a random sample from the training dataset.

The text preprocessing steps I took were expanding contractions, changing to lowercase, removing special characters, punctuation, and digital numbers, and removing stop words except 'not', 'no', and 'nor'. I removed stop words except those three in order to reduce my training dataset as it was very large. I also tokenized, lemmatized the data, and found the unique words and count. I also shortened my unique words to counts only above 1000 to speed up the process and keep only the most relevant occurrences.

Feature Extraction
I extracted features using 3 common ways: Bag-of-Words, TF*IDF, and Word2Vec.
Bag-of-Words is a vocabulary of words and their corresponding frequencies. The unique words list is then compared to each text passage in order to create a Bag-of-Words Vector. If a word appears in the text passage, then it is given a 1. If the unique word does not appear in the text passage, it is given a zero for that text's vector. The issue with Bag-of-Words is sparsity and the over-abundance of stop words. In order to combat the sparsity (and load times for my computer), I only kept the unique words with frequencies above 5000 which ended up being 254 words.

TF*IDF is similar to Bag-of-Words with a slight change to combat the influence of stop words. In TF*IDF, we attempt to find words that occur frequently in a sentence but rarely in the corpus because they are more likely to be informative rather than words that occur frequently in both the sentence and corpus. TF = frequency of words in a sentence / total words in a sentence. IDF = log_{10}(number of sentence in dataset / number of sentences that contain the word). These two values are multiplied together in order to create a TF*IDF vector for each sentence. I had an issue with tf*idf needing too much RAM so I limited my array features to 200 features.

Word2Vec groups together vectors of similar words. Word2Vec preserves the relationship between words. It is used to make words with similar contexts to have similar embeddings. In this assignment, I used Continuous Bag-of-Words. This predicts the target word from the context.

3. Model Development

- Model Training
  I used 3 models: Logistic Regression, Naïve Bayes, and Random Forest.
  I applied each of my 3 extraction methods to each model in order to compare and contrast the differences between different models with the same feature extraction methods and the same models with different feature extraction methods.
- Model Evaluation : **Comparing Different Classifiers with same Feature**

Bag-of-Words

	precision	recall	f1-score	support
0	0.00	0.00	0.00	177
1	0.51	1.00	0.67	182
accuracy			0.51	359
macro avg	0.25	0.50	0.34	359
weighted avg	0.26	0.51	0.34	359

Logistic Regression
Accuracy is 0.51
One f1-Score is 0 while the other is moderately high.

	precision	recall	f1-score	support
0	0.00	0.00	0.00	177
1	0.51	1.00	0.67	182
accuracy			0.51	359
macro avg	0.25	0.50	0.34	359
weighted avg	0.26	0.51	0.34	359

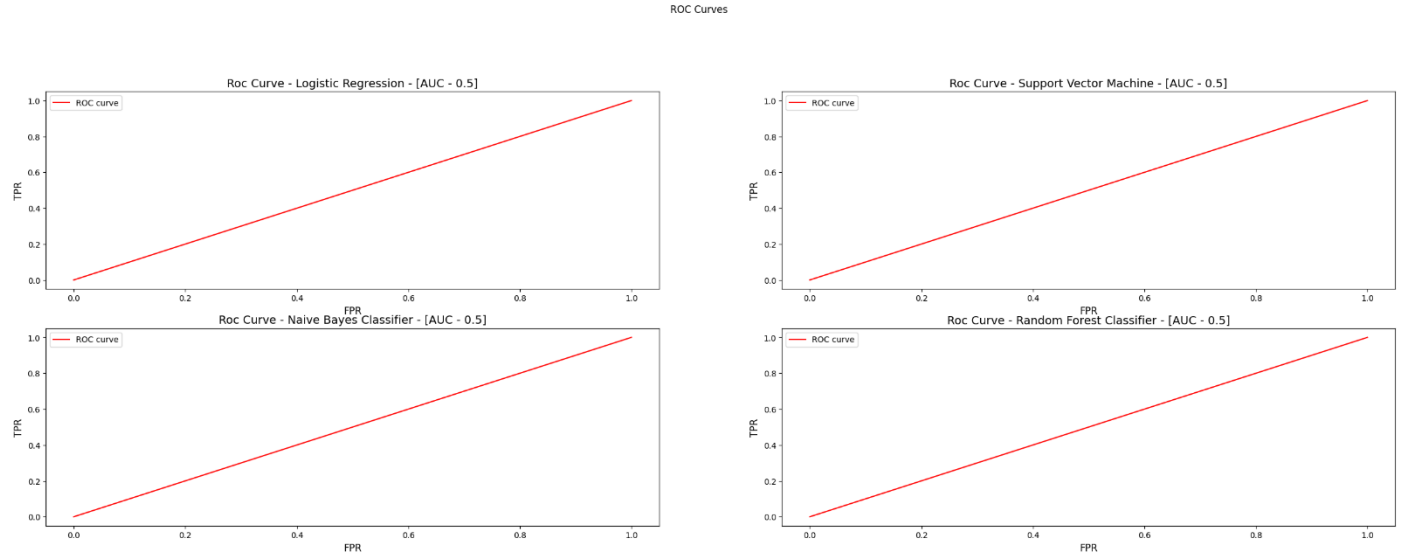
SVM
Accuracy is 0.51
One f1-Score is 0 while the other is moderately high.

	precision	recall	f1-score	support
0	0.49	1.00	0.66	177
1	0.00	0.00	0.00	182
accuracy			0.49	359
macro avg	0.25	0.50	0.33	359
weighted avg	0.24	0.49	0.33	359

Naïve Bayes
Accuracy is 0.49
One f1-Score is 0 while the other is moderately high.

	precision	recall	f1-score	support
0	0.00	0.00	0.00	177
1	0.51	1.00	0.67	182
accuracy			0.51	359
macro avg	0.25	0.50	0.34	359
weighted avg	0.26	0.51	0.34	359

Random Forest
Accuracy is 0.51
One f1-Score is 0 while the other is moderately high.



As indicated by these results, all models utilizing Bag-of-Words performed similarly as shown by the AUC scores.

TF*IDF

	precision	recall	f1-score	support
0	0.75	0.58	0.66	177
1	0.67	0.81	0.73	182
accuracy			0.70	359
macro avg	0.71	0.70	0.69	359
weighted avg	0.71	0.70	0.69	359

Logistic Regression
Accuracy is 0.70
Both f1-scores are slightly high.

	precision	recall	f1-score	support
0	0.72	0.55	0.62	177
1	0.64	0.79	0.71	182
accuracy			0.67	359
macro avg	0.68	0.67	0.67	359
weighted avg	0.68	0.67	0.67	359

SVC
Accuracy is 0.67
Both f1-scores are average.

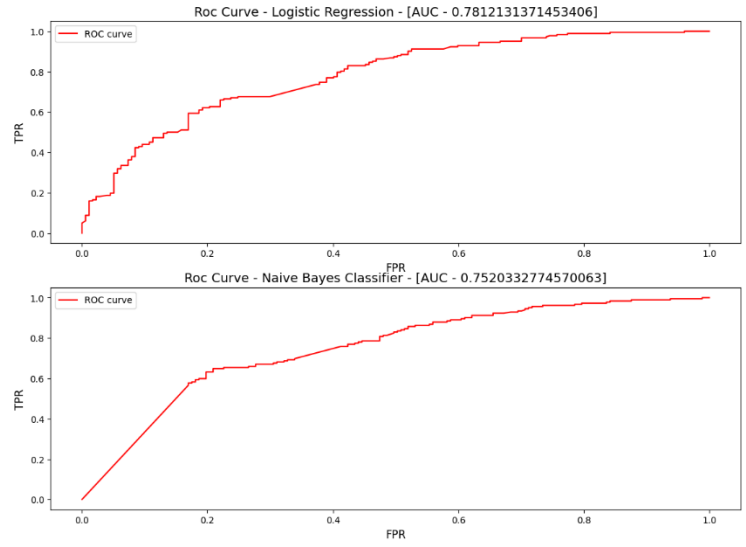
	precision	recall	f1-score	support
0	0.71	0.56	0.63	177
1	0.65	0.78	0.71	182
accuracy			0.67	359
macro avg	0.68	0.67	0.67	359
weighted avg	0.68	0.67	0.67	359

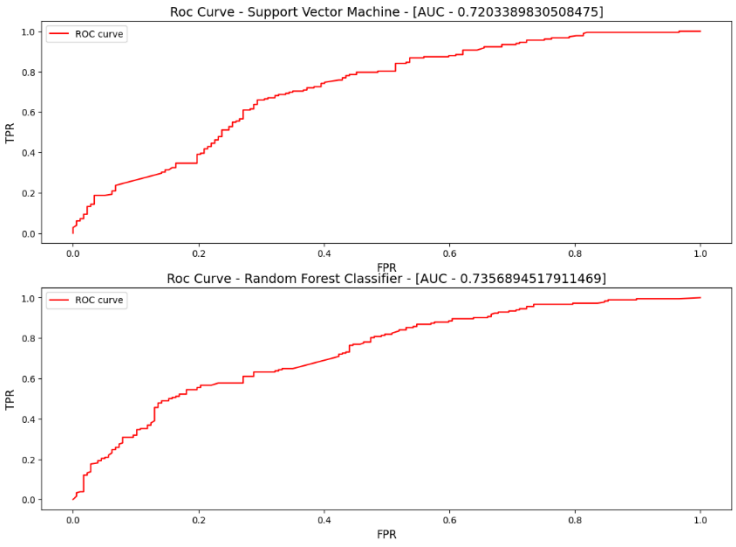
Naïve Bayes
Accuracy is 0.67
Both f1-scores are average.

	precision	recall	f1-score	support
0	0.70	0.56	0.62	177
1	0.64	0.76	0.70	182
accuracy			0.66	359
macro avg	0.67	0.66	0.66	359
weighted avg	0.67	0.66	0.66	359

Random Forest
Accuracy is 0.66
Both f1-scores are average.

ROC Curves





For classifiers utilizing TF*IDF, the Logistic Regression model performs the best as indicated by the AUC score. It is also the most accurate among the 4.

	precision	recall	f1-score	support
0	0.62	0.66	0.64	177
1	0.65	0.61	0.63	182
accuracy			0.63	359
macro avg	0.63	0.63	0.63	359
weighted avg	0.63	0.63	0.63	359

Word2Vec
Logistic Regression
Accuracy is 0.63
Both f1-scores are moderately high.

	precision	recall	f1-score	support
0	0.53	0.84	0.65	177
1	0.65	0.29	0.40	182
accuracy			0.56	359
macro avg	0.59	0.56	0.53	359
weighted avg	0.59	0.56	0.52	359

SVC
Accuracy is 0.56
The f1-score for negative sentiment is above average. The f1-score for positive sentiment is slightly below average.

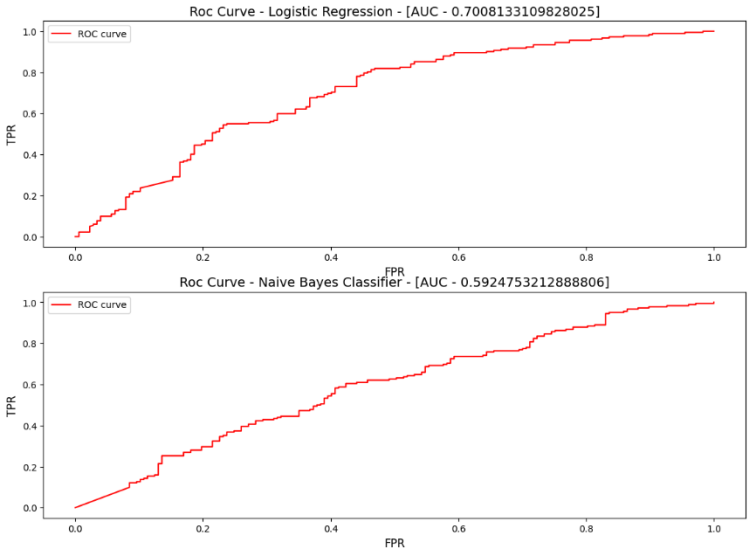
	precision	recall	f1-score	support
0	0.52	0.86	0.65	177
1	0.62	0.22	0.33	182
accuracy			0.54	359
macro avg	0.57	0.54	0.49	359
weighted avg	0.57	0.54	0.48	359

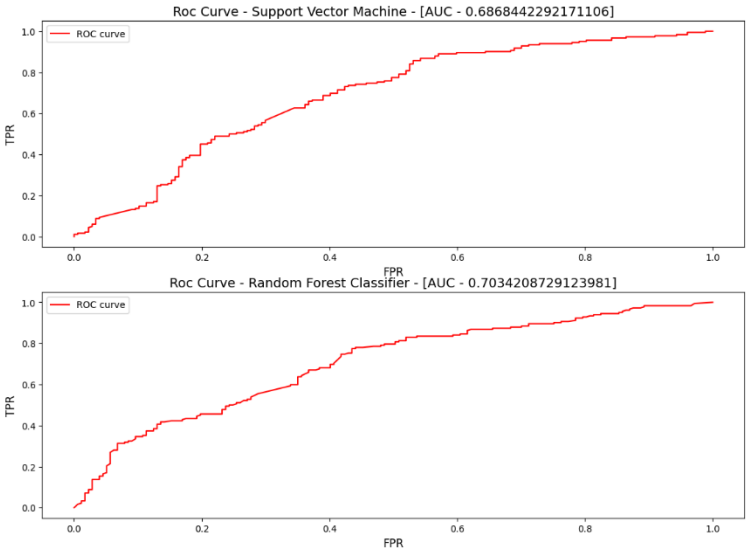
Naïve Bayes
Accuracy is 0.54
The f1-score for negative sentiment is above average. The f1-score for positive sentiment is low.

	precision	recall	f1-score	support
0	0.70	0.57	0.63	177
1	0.64	0.76	0.70	182
accuracy			0.67	359
macro avg	0.67	0.66	0.66	359
weighted avg	0.67	0.67	0.66	359

Random Forest
Accuracy is 0.67
Both f1-scores are above average.

ROC Curves





For classifiers utilizing Word2Vec, the Random Forest model performs the best as indicated by the AUC score. It is also the most accurate among the 4.

Next, we will compare the same classifiers with different feature extraction methods.

Logistic Regression
TF*IDF performed the best with Logistic Regression with an accuracy score of 0.70 and an AUC score of 0.78
It's f1-scores were also higher than all other models.

SVC
TF*IDF performed the best with SVC with an accuracy score of 0.67 and an AUC score of 0.72
It's f1-scores were also higher for all models except Word2Vec which had a slightly better f1 score for negative sentiment.

Naïve Bayes
TF*IDF performed the best with SVC with an accuracy score of 0.67 and an AUC score of 0.75
The negative sentiment f1 scores were lower than other Naïve Bayes models; however, the positive sentiment f1-scores were much higher than all other Naïve Bayes models.

Random Forest
TF*IDF performed the best with Random Forest with an accuracy score of 0.66 and an AUC score of 0.75
Word2Vec was close in performance with TF*IDF, and had a higher accuracy score of .67. it's AUC score was 0.70, and had a marginally higher f1-score for negative sentiment.

4. Discussion

- My best model performs with an accuracy of 70% and AUC of 0.78. All classifiers using TF*IDF features performed better.
- Some problems I ran into during this assignment, was the vast amount of data. I fixed this by simply taking smaller samples of the data.
- I also had issues with figuring out how to apply the extracted features to the classifiers I have used previously. This ended up being quite simple, but I was unsure how to approach at first.
- I have previously taken an NLP class; however, I have learned much more by needing to clean, extract features, apply to models, and analyze data from this assignment.

5. Appendix

https://github.com/avary8/Linear-Logistic-Regression