MSIA 414 Homework 3 – Anthony Colucci

1. My dataset consists of 500,000 reviews of video games from Amazon (*dataset_stats.py*). This is a subsample of the data available at https://snap.stanford.edu/data/web-Amazon.html. All of the reviews rate the game on a scale from 1 to 5 stars. I will be classifying games as 1, or positive, if they have a review of 4 or 5 stars and 0, or negative, otherwise.

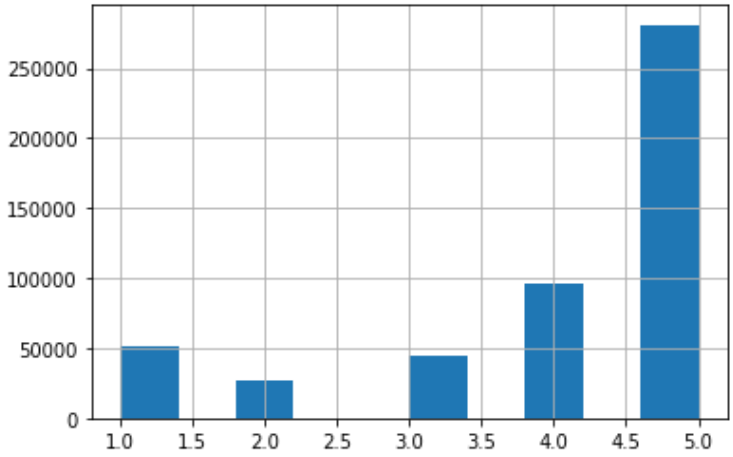

Figure 1: Rating Distribution

Number of Reviews	500,000
Number of Positive Reviews	377,479
Average Score	4.05807
Average Words per Review	116.05446
Average Characters per Review	630.353214

2. When fitting the logistic regression (*logistic_regression.py*), I ran three experiments, each to determine the effect of a different options for a TF-IDF representation of text. These options were the inclusion of bigrams, linear versus sublinear term-frequency, and the number of max_features. I was interested in the use of bigrams as it may be important for certain negations, for example, the word "not" and "good" may individually be ambiguous or indicate a positive review, the bigram "not_good" may have a different interpretation. For sub-linear term frequency weighting, I assumed that this would cause the words that appear the most in a given review to have relatively less weight on the outcome of the review. I then wanted to consider if the number of features would have an impact on the performance of the model, seeing if I had a wider array of word features, if my prediction accuracy would improve. I compared these models based on F1-score at a 50% threshold, with the results below:

Baseline (no bigrams, linear tf, 750 max features)	0.90256
With bigrams in the encoding	0.89989
Sublinear term-frequency	0.90308
500 max features	0.90256
1000 max features	0.90256

From the above table, we can see that sublinear term-frequency led to a slight increase in model accuracy on the test set. Adding bigrams slightly lowered the level of the predictions, though perhaps if that was accompanied by a different number of features, those bigrams could have provided value. Looking only at unigrams though, it's clear that we were fitting with fewer than 500 features, as updating the max features parameter had no effect on the model accuracy.

3. When creating my linear SVM (*svm.py*), I wanted to test the same parameters as I did in question 2 with logistic regression.

Baseline (no bigrams, linear tf, 400 max features)	0.89295
With bigrams in the encoding	0.89036
Sublinear term-frequency	0.89276
300 max features	0.88845
500 max features	0.89574

From the table above, we can see that this SVM performed slightly worse than logistic regression against the test data. Adding more features (up to 500) did help the accuracy of the model and decreasing the number of features reduced the accuracy, suggesting that further gains may be available by further increasing the number of max features past 500. Sub linear term frequency had a very slight negative effect and adding bigrams again decreased the accuracy, though again, adding more features in conjunction with adding bigrams may change that.

4. When training the fasttext model (in *fasttext_alc460.py*, since a file called fasttext.py caused issues when trying to load the actual fasttext package), I wanted to consider, as with previous models, how adding bigrams would affect the accuracy of the model. Additionally, I wanted to see how reducing the window size for the fasttext encoding affected the model, with my hypothesis being that it will lower accuracy by reducing how much context each embedding has. The final change I wanted to test was increasing the number of training epochs from 5 to 10, which I expected would increase the predictive power of the model.

Baseline (no bigrams, window size=5, epochs=5)	0.88121
With bigrams	0.88542
Window size=3	0.88220
Epochs=10	0.87790

Looking at the results, the baseline fasttext model is less accurate than either SVM or logistic regression, though not by a massive amount accounting for the speed with which it can create embeddings and train a predictive model. In this model, adding bigrams increased the accuracy of the model, contrasting the previous two models, and surprisingly, reducing the window size also increase the accuracy over the test set. This result is particularly surprising since the embeddings are presumably made with less information, but that may speak to how the words in this particular data set relate to one another. The results of one experiment was quite surprising as increasing the training epochs to 10 decreased the accuracy of the model on the prediction set. This suggests overfitting to the data in the training set, but I was surprised that we would encounter that with a relatively light model.