BU CS 506
Midterm, 10/28
Charles Li
U21328455

1. Introduction:

For this midterm, the goal was to develop a machine learning model to predict star ratings on a scale from 1.0 to 5.0 of Amazon movie reviews on both textual and metadata attributes. Using the provided kaggle dataset, we aimed to train a model off of 1,697,533 unique reviews, and ultimately fill in and identify the scores of reviews where the final rating was missing.

The included kaggle dataset included the following attributes:

- ProductId - unique identifier for the product
- UserId - unique identifier for the user
- HelpfulnessNumerator - number of users who found the review helpful
- HelpfulnessDenominator - number of users who indicated whether they found the review helpful
- Score - rating between 1 and 5
- Time - timestamp for the review
- Summary - brief summary of the review
- Text - text of the review
- Id - a unique identifier associated with a review

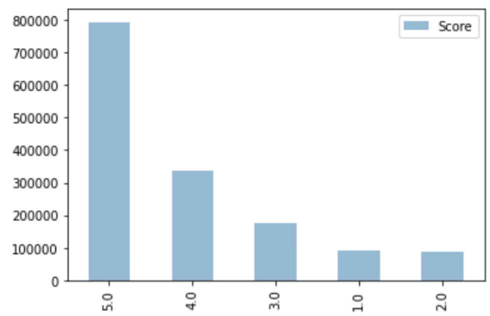
Ultimately, I made note that there are 2 general categories of the data: non-textual; Floats and Integers that we can use with less processing, and textual data, Text and Summary attributes that make up the bulk of the dataset. I attempted to experiment around with the non-textual data first. This is because I knew that manipulating the textual data for use in the model would be difficult and be a costly operation, given the size of the dataset. Hence, I experimented, and found the following features to be most important, before returning to the textual data and doing the more heavy analysis later.

2. Data and methodology

a. Features

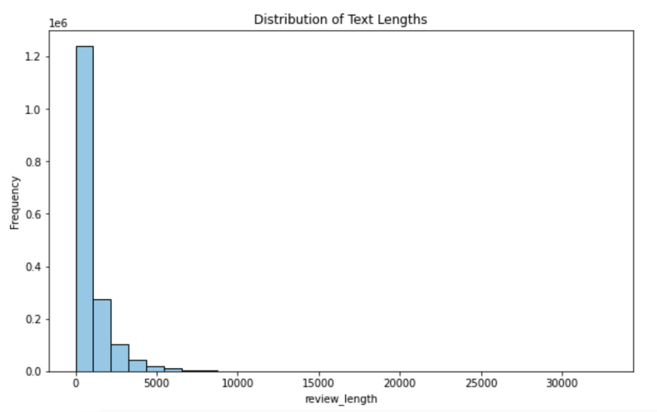
The majority of the time I spent on developing this machine learning model was in developing features; and then a smaller subset of the time was spent working with the parameters and hyperparameters of the end model I decided to use.

Firstly, regarding the features. I created several simple features that require no explanation: Length of review, time of month and year of the review. Furthermore, I decided to add an average product score attribute based on each unique UserId's given score. In cases where there were no scores to attribute, I populated with the average score of the dataset: **4.1105**.



Distribution of scores of the dataset

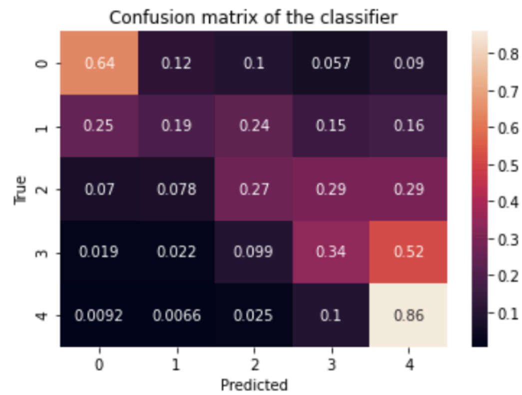
Finally of note is Helpfulness as an attribute; I used *HelpfulnessNumerator*, *HelpfulnessDenominator*, and finally an aggregate helpfulness attribute of the numerator over the denominator, which I called *Helpfulness.*



Now in terms of the text themselves, I focused on two things: Sentiment analysis, and vectorization. Vectorization was done as covered in class; running tf_idf to obtain 500 vectors of the most common words, and calculating the average occurrences of each word throughout the dataset. Sentiment analysis was done through the TextBlob library, which is a library built on python's NLTK library. It parses through each Text and assigns a given score of sentiment and polarity based on a database of keywords. Positive keywords raise the sentiment, negative keywords decrease the sentiment, and a negation word(not good, rarely impressive) reverses the sentiment of the following keyword. Together, these represented the bulk of the features that had a final impact on the outcome of the model.

b. Model

After testing, I decided to use a Random Forest Classifier.



I ultimately tested Logistic Regression models, as well as K-Nearest Neighbors classifiers, but none performed as well as RFC. RFC is better because of the amount of features used; all 500 of the vectorized words are present. In KNN this amount of features would cause it to not be able to compile the model; and in logistic regression it would cause too much noise. I believe that the Decision trees of Random Forest Classifiers are perfect for this use case.

c. Conclusion

Ultimately, I obtained an accuracy on the testing set of **0.629489.** Through some rudimentary hyper parameter tuning, I used 300 estimators of the model, over 510 features.

3. Discussion and conclusion.

I ultimately believe that the biggest constraint of the project was time. There was a lot more I could have done in terms of both the features in the model. In future projects, I will look to run TF_IDF on more samples; I could have run on both the summary and the text itself. I also would have liked to run more statistical analysis on the scores and the non-textual data; I could have found correlations between more attributes of the data, and created more creative features of the data.

I didn't have the time I would have liked to to do proper hyper parameter testing. I would've run RandomizedSearchCV to iterate through different variations of the model, expanding out of different scoring systems, max features, estimators, and more.

With more time I would have also expanded more on the model itself; I would have probably used an aggregate of different models to more accurately assess parts of the data.