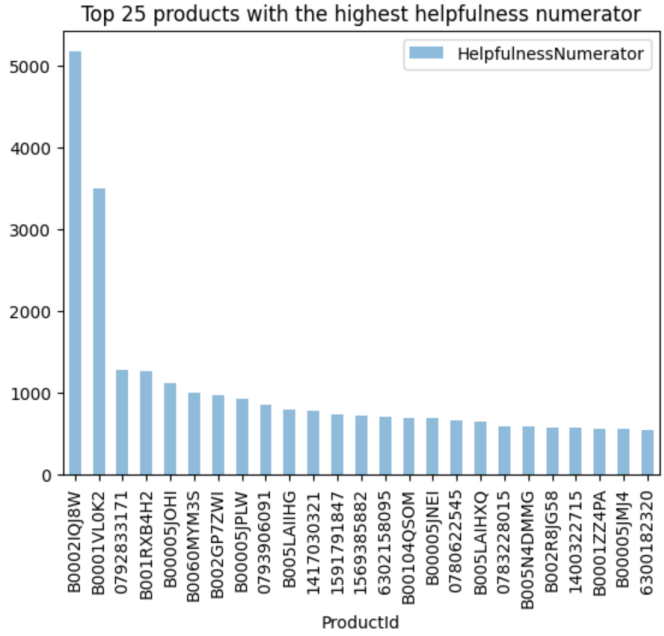
Name: Aryaan Upadhyay
Kaggle Username: aryaan_upadhyay_22
Professor Lance Galletti, CS 506
Date: 3rd March 2023

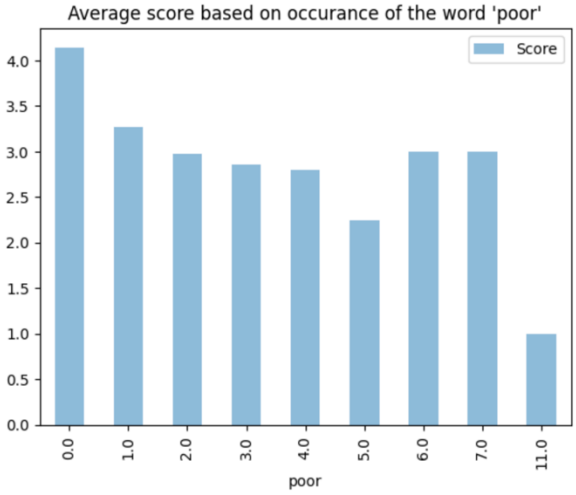
Midterm Report: Analysis and Extrapolation

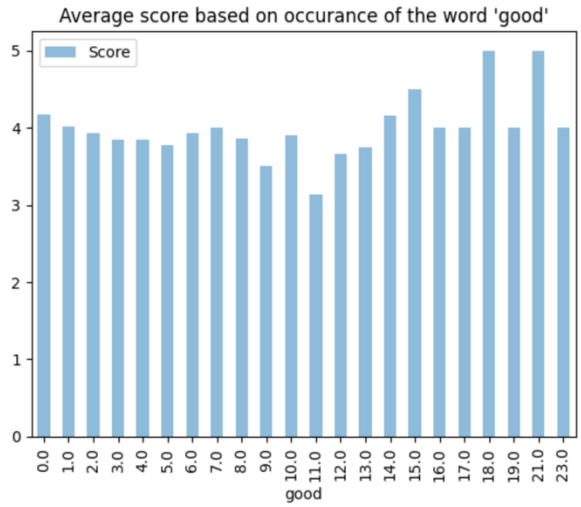
When tasked with this assignment, I originally felt overwhelmed as I did not know how to go about predicting values based on features. However, after speaking with the professor I realized that the ideal base step is to understand the trainingSet. This involved getiing familiar with the different columns and their inputs. To get more familiar with the dataset, I decided to plot different graphs to represent the correlations amongst the different columns. One of these correlations was top 25 products with the highest helpful numerator.



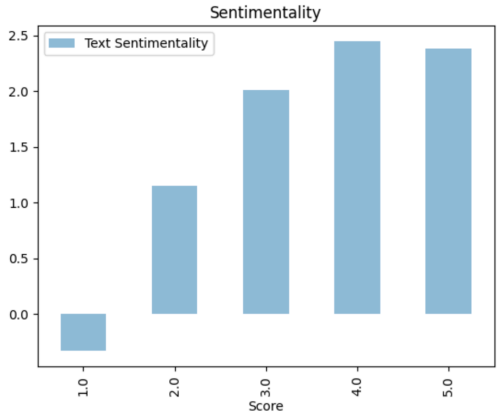
In addition to understanding how the columns interacted with one another, I was able to get a peek into understanding the process of "feature extraction." To make a strong base for predicting scores of movies, I decided to make a feature between the type of words used to describe the product and the actual rating itself - the idea behind this approach was that, ideally, if many people use more positively associated words to describe an attribute, it is more likely that the attribute being discussed is likeable. To start this, I created a list of arbitrary words, of which some are positive and some negative. While this is not an exhaustive list, I managed to encompass the frequently used adjectives such as "good" and "bad." This concept worked as it allowed me to show that if a highly rated product is critically acclaimed then the number of positive adjectives occurring must be higher than that of the number of negative adjectives. In the example below it can be seen that the number of times the word "poor" is used is inversely

related to the average score of the product. Additionally, the number of times the word “good” is used is related positively to the average score of the product.



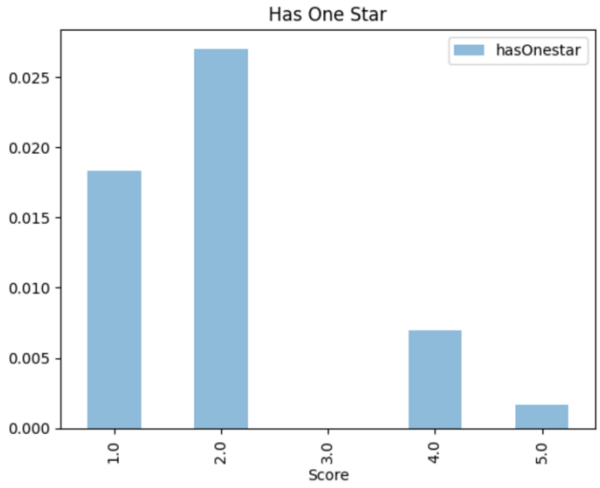


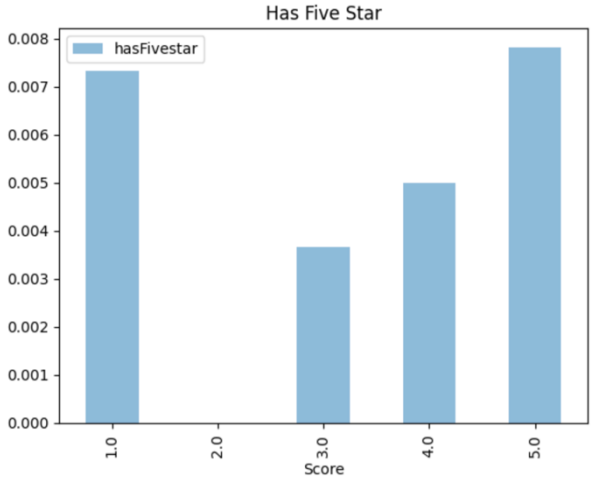
To build off the list of adjectives used to describe the product, I decided to incorporate the sentimentality value of words as a feature. The words were taken from the “Text” and “Summary” columns. To conduct this task, I used the “vader_lexicon” module part of the “nltk” library. This feature focused on assessing the words used to describe a movie while assigning different weightage to these words from [-1,1] based on whether they are neutral, positive, or negative. Based on the score of these words, the total sentimentality score of a review was calculated to predict the score of a movie. The hypothesis I proposed while doing this task was that the higher the sentimentality score, the higher the score of the movie would be. This hypothesis turned out to be right as seen from the graph below.



Like the above approach of making a proportion of adjectives to that of the score of the product, I also wanted to analyze how the phrases “one star,” “two stars,” “three stars,” “four stars,” and “five stars” within the “Text” column impact the actual average rating of the product. This was important to analyze as it helped me predict how accurate the rating of the product with respect to the review written. Like the above two approaches, the average rating

of the product was pretty like the phrases within the review. I postulated this as I wanted to see how the occurrences of these phrases would sit with the average rating. By doing this I was also able to account for anomaly cases.





Having had a few features to help process the data, I decided it was time to focus on the different classifiers that I could use. Initially, we were given a KNN Classifier, but after a few iterations I decided to move away from this mode of classifier – I adopted the DecisionTreeClassifier. This is because the KNN dataset was very biased towards giving ratings closer to 5 Star rating – I do think this is because the trainingSet data that we used was biased. Additionally, the accuracy was much lower compared to that of the DecisionTreeClassifier. I decided to stick with DecisionTreeClassifier as opposed to other models as the tuning parameters for the decision trees aligned with my feature extraction process. As most my features relied on numerical data, I thought it would make sense for the program to handle decision based on these numeric values.

Most of the issues that I faced had to do with extraction process. While trying to predict the scores, I created a lot of new columns – which were associated with the different features I had extracted. As these columns were numeric, I had to fill the null values with 0. However, this could bias the result, as by predicting scores based on columns with extra 0s, the computations might get skewed. The process of filling in null values also impacted the confusion matrix. As a result, there were 6 rows and columns in the confusion matrix printed. These issues were also compounded by the fact that sometimes my kernel would fail to run on large datasets – to solve this I decided to run the process on different batches by sampling. Another issue that I faced is that since most of my features were based on lexicon, empty strings had to be used instead of null values. This also posed as an issue because the addition of these reviews did not add to the sentimentality analysis. I do think that the sentimentality analysis could have been better if I had accounted for punctuation. However, I failed to do so because I could not find punctuation which fits with negative emotions. “!” does fit in with excitement and happiness and if I had accounted for this, I would not account for negative emotions. This would further skew the data.