Introduction

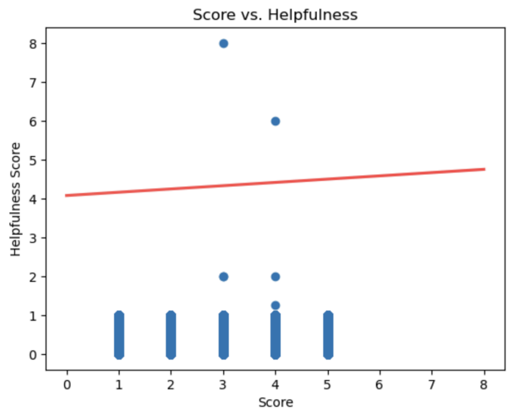
By determining important trends related to a given interest point, we can train models given information related to that feature that can then predict the value of that feature. One such example is predicting the rating of a movie using the review metadata. In this assignment, we were tasked with just that. We were given a dataset of unique reviews from Amazon Movie Reviews, each including information related to the review itself, the movie, and the user. Using this information, we then wanted to train a model that could predict the rating that the review gave to the movie.

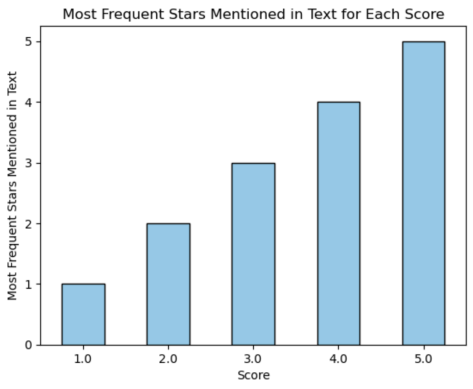
However, it is also important to note the importance of properly tuning a model so that it will respond properly to the given information. This is especially difficult in this assignment because much of the data is reliant on human sentiment, which is much harder to predict. In the following paper, I describe my process of determining features on which to train my model, how I selected my model, and any tuning and last steps that allowed me to optimize the model.

Feature Selection

The first step in my process was selecting features that would help my model determine the score given in the review. In order to do so, I analyzed the trends between the score and different features to determine whether they should be added to the model's training data.

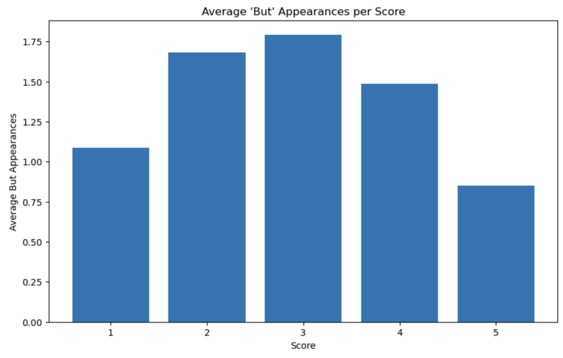
First, I analyzed the helpfulness, which is the ratio of users who found the review helpful out of the total number of users who responded regarding the review's helpfulness. In order to calculate it, I took the "HelpfulnessNumerator" feature in the data file and divided it by the "HelpfulnessDenominator" feature. From the graph below, we can see that there is a slight upward trend between the helpfulness score and the given score. While it may not be the strongest correlation, it still allows us to understand how helpfulness can affect the score, which is why I included it within the selected features on which I trained by model.

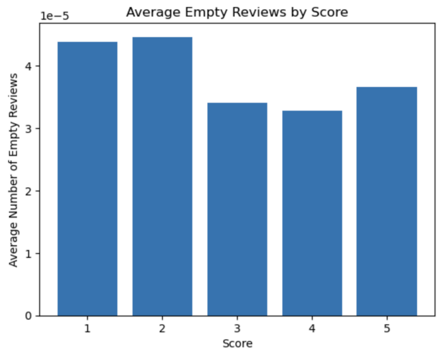




Next, I looked at whether the number of stars left on a review appears in the text of the review. For example, a user may leave a review and within the text explain their score in the text by saying something along the lines of "I left this movie 5 stars because…", which allows us to more easily classify the review. Above, we can see the mode of these appearances in the text for each of the review scores, which correlates with the score given by the user. Due to this strong correlation, I decided to include these values in the features that can be used to train the model.

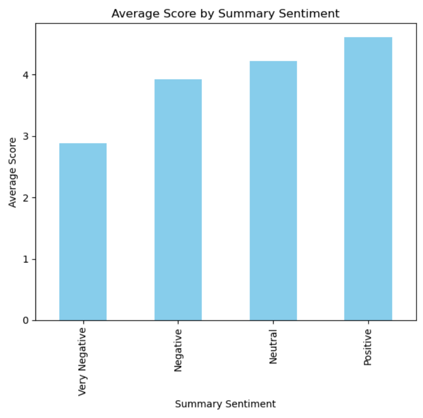
Also considering the contents of the review's text, I decided to look at the appearances of the word "but" within the review. This stemmed from my hypothesis that many of the reviews with more middle scores (i.e. 2, 3, 4) would have more sentences in which they justify their reasoning with "but". For example, they could say "I like this portion of the movie but …", explaining a more neutral rating on the film. Additionally, with the first two features, my model was doing a relatively better job predicting the extremes, scores of 5 stars and 1 star, so I wanted to implement this feature to better predict neutral reviews. In the graph below, we can see that there is a trend of having more appearances of "but" for the median values, so I added this feature to the data for the model training, allowing for better classification of median values.

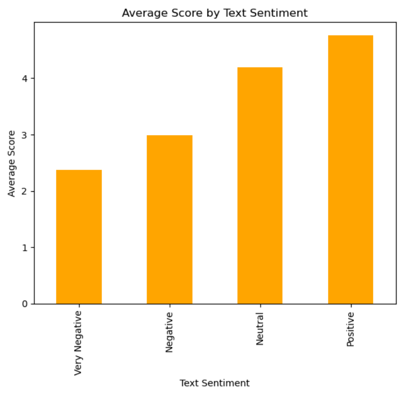




Next, I explored the trend between empty reviews and the review score. Likely, this would further classify the extremes as reviews of 1 and 5 often need less explanation than median values. From the graph above, we can see that, while the correlation does not fit perfectly with the scores, there is a slight trend where scores of 1 and 2 have more empty reviews. Therefore, it can be useful to classify these reviews with lower scores, so I added it to the features on which the model is trained.

I also wanted to explore sentiment analysis of the text itself. In order to do this, I used TextBlob functions, which return a number between -1 and 1, where -1 means the text represents a negative connotation while 1 has a positive one. Therefore, using this information, we could better classify the text as a whole on its contents. By running this on both the summary and the text, we get a good understanding of the sentiment of the review as a whole. From the graphs below, we can see the positive correlation between the text sentiment and the average score, with a noticeable difference between each rating. Therefore, this was an important feature that I added to train the model.





Finally, I wanted to use the average scores of a given user or product. A given user may be more likely to review a product when they really enjoy it or dislike it, so having their average score could allow for better classification of their reviews. Knowing a product's average score would also likely lead to better classification due to a better understanding of the general opinion of the film. Therefore, these features were also added to help train the model.

Model Selection

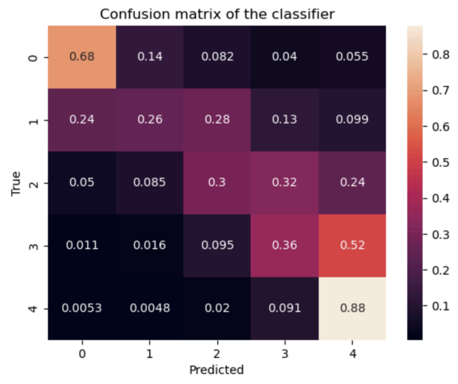
After selecting the features that best indicate the score of a given review, I decided to try a few different models that would best train and fit the dataset. When first testing the features, I started with a K-nearest neighbors (KNN) where K was set to 3. This model simply finds the K data points that are closest to the given data point and assigns a value according to the predominant value that is represented (Steinbach 2013). While this may not have led to the most accurate results, its efficiency allowed me to quickly test many of my features to see how they affect the accuracy of the dataset.

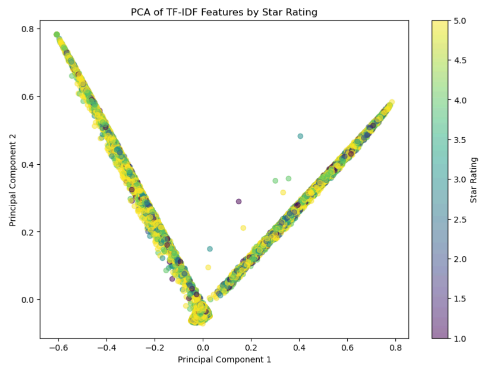
In order to increase the accuracy, I wanted to test on a different model, namely a standard vector machine (SVM). I chose this model specifically because it performs classification well on higher dimensional data (Porcello 2019). This model essentially classifies data by finding the optimal line that maximizes the distance between classes. However, due to the extensive training necessary to classify the data, I ran into issues running this model and was therefore unable to use it as my final model.

Instead, I tried to use an extreme gradient boosting (XGBoost) model. This model is known for its efficiency, so I thought it would be helpful in this assignment. Essentially, this model works by providing parallel tree boosting, meaning that it uses decision trees and an ensemble learning algorithm to make better predictions about how to classify data (NVIDIA, 2024). After training this model, my predictions had much higher accuracy.

Validation

In order to check how my model and features were working, I had an accuracy score on my testing set. However, I also wanted to see which scores were being better predicted and whether I had scores that were close to being correctly predicted. In order to do so, I used a confusion matrix like the one below which allowed me to better see how the data was being classified. By using this confusion matrix, I was able to decide what I should be focusing my features on as well as how to optimize my model as a whole.





Future Steps

While I am relatively happy with the model that I produced, I would have liked to include a few other additions if given the time. First, I would have vectorized the text of the reviews using TF-IDF, which reflects the importance of words in a document, which in this case is a review. I had initially run TF-IDF on a much smaller subset of the data, whose results you can see above. From that graph, you can see that there is a correlation between the components and the ratings of the review. However, when running the TF-IDF on all of the code, I ran out of time so I was not able to add that as a feature, despite wanting to.

As I mentioned earlier, a large constraint on the model that I built was the amount of time that it took different aspects to run. For example, after trying to run the SVM for almost 20 hours, I decided to give up, rather than continue running it, which I may have done if I had more time. Despite trying with a smaller subset of data, the training of the SVM model was still taking far too long, so I decided against implementing it, which I may have decided to do if I had more time to work on this.

Additionally, I would have liked to train an ensemble for the model, which would include XGBoost, SVM, KNN, and possibly other regression models. By using an ensemble, it would have allowed me to better classify the data points, taking into consideration the decisions of multiple different models. Through iterative testing, I would have better tuned these values to include the optimal models which led to the best accuracy.

Finally, I would have run a hyperparameter tuning so that the model could have the best results given the dataset. While I did not have enough time to do so during the duration of this midterm, I believe that this would have allowed the model to even better predict the scores of the reviews.

Citations

J. McAuley and J. Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. WWW, 2013

NVIDIA Data Science Glossary. "What Is XGBoost?" Accessed October 28, 2024. https://www.nvidia.com/en-us/glossary/xgboost/.

Porcello, J. C. "Designing and Implementing SVMs for High-Dimensional Knowledge Discovery Using FPGAs." Paper presented at the 2019 IEEE Aerospace Conference, Big Sky, MT, March 2-9, 2019.

Steinbach, Michael, and Pang-Ning Tan. "kNN: k-nearest neighbors." In *The top ten algorithms in data mining*, pp. 165-176. Chapman and Hall/CRC, 2009.