# CS506 Midterm Fall 2024

## I. Introduction

In this competition, we aim to predict the star rating associated with user reviews from Amazon Movie Reviews using the given features to train our model. I used features including sentiment analysis and a word frequency analysis. I used the word frequency analysis to create weighted values for each word in each rating class. I experimented with models including K-Nearest Neighbors, Decision Trees, and Random Forests to improve the final model. This report will describe the process/methods I used to obtain my final model.

## II. Data

The dataset used to train the model had around 1.7 million reviews, ~800,000 of which were 5 stars, ~350,000 were 4 stars, ~180,000 were 3 stars, ~85.000 were 2 stars, and ~100,000 were 1 star. Each review contained the following fields:

- **Id**: Identified the review in the dataset.
- **ProductId**: Identified the movie being reviewed.
- **UserId**: Identified the user reviewing the movie.
- **HelpfulnessNumerator**: Showed the number of people who found the review helpful.
- **HelpfulnessDenominator**: Showed the number of people who found the review either helpful or not.
- **Time**: Showed the time when the user made a review.
- **Summary**: A brief summary of the review created by the user.
- **Text**: A longer explanation as to why the review was left.
- **Score**: The number of stars, from 1-5, that the user rated the movie.

## III. Methodology

Before going into the additions that improved my model, I wanted to balance the number of each review, making it a uniform distribution since I believe that it trained the model more accurately. To balance them, I downscaled the number the size of each class to match the class with the lowest reviews. This ended up being the 2-star reviews class with around 85,000 reviews.

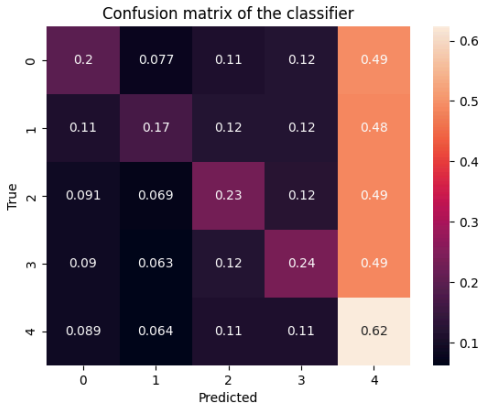
I am now going to explain the different features I explored. Initially, we were given 4 features to work off. Helpfulness Numerator, Helpfulness Denominator, Time, and Helpfulness. Helpfulness was found by diving the Helpfulness Numerator by the Helpfulness Denominator. I left them in for the majority of the testing at the start since I wanted to see how it affected the model as I added other features.

The main feature I wanted the model to focus on was the different words we would find in each of the star ratings. This included several steps.

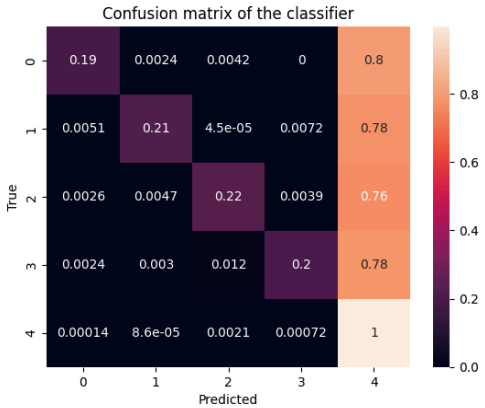
- **Text Cleaning:**
    - When creating this feature, I realized that several words, punctuations, and special characters would not contribute to the content of the review so I created a list of words that the algorithm would skip over when encountered.
    - In addition, all words were converted to lowercase to avoid repetition.

- **Sentiment Analysis:**
  - Sentiment analysis was used to assess the overall tone of the review. I created a sentiment score for each review using the pre-built sentiment analysis library TextBlob.
  - Sentiment was split into Polarity and Subjectivity. I tested these features to guide the mode however found that it had trouble distinguishing between each of the classes. It ended up lowering the accuracy of my model so I decided to not include it in my final model.
- **Word Frequency Analysis:**
  - I created a function that would count the frequency of each word across all the reviews and identified the most common words associated with each rating.
  - For each star rating, I removed all of the common words from the word list.
- **Word Weight Analysis:**
  - After calculating word frequencies, I assigned weights to each word based on how frequently they appeared in each star rating. Words that were strongly associated with a specific rating class were given higher weights.
- **Normalized Weighted Score Feature:**
  - To further refine the model, I introduced a normalized weighted score feature. This score was derived from the weighted word frequencies, normalized across the entire dataset. This allowed the model to assess the significance of each word in context, adjusting for the length of the review and the overall frequency of the word across reviews.
  - I knew that I would find difficulties accounting for negations of adjectives which would change the entire meaning of the review so I made sure to account for negation words, e.g. good versus not good.

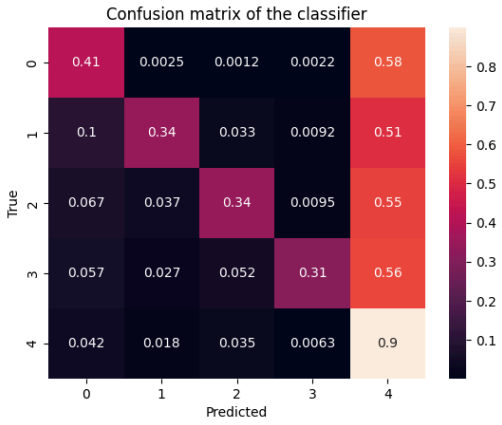
## IV. Results




- The first figure is the k-nearest neighbors model. This yielded a tested accuracy of around 51% with a k value of 15. At the original k value of 3, the accuracy hovered around 43%. I ended up not using this model since the k value needed to be higher for the model to be accurate but it would then result in too general of a model/underfitting.

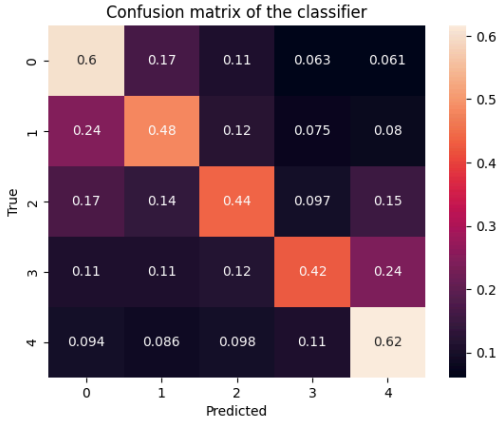



- The second figure is the decision tree model. This yielded a tested accuracy of around 64%. Even though this model also mostly predicted 5-star ratings, it identified more reviews correctly. Therefore, this is the model I decided to use.




- The third figure is the random forest model which combines the output of multiple decision trees to reach a single model. This yielded a tested accuracy of around 60%. This model was the most skewed toward the 5-star rating so I ended up not using this model.

- I believe I could've received better results if I was able to fine-tune the Word Frequency Analysis and Word Weight Analysis algorithms. They yielded decent results but still had trouble distinguishing between what was 5 stars or not. I realize now that the other review classes can have several words that would also appear in the 5-star class.




- In addition, I tried to use a sentiment intensity analyzer with the Natural Language Toolkit, which seemed to produce more accurate results overall, as seen in the 4th figure, however was less accurate in identifying 5-star reviews. Based on the testing on my device, it seemed to produce better results, but upon submission to the Kaggle competition, it ended up with the lowest score out of all of my previous submissions. Thus, I removed the sentiment analyzer from my final submission.

## V. Conclusion

In conclusion, the decision tree model yielded the highest accuracy with the normalized weighted score features. Although some features that I tried didn't produce a better model, it allowed me to eliminate patterns I previously thought existed. In addition, the process of balancing the dataset to ensure equal representation across star ratings contributed significantly to the model's performance. By refining the word frequency and weight analysis techniques, I discovered that certain words could appear in multiple rating classes, complicating the model's ability to classify reviews accurately. The findings from this project show the critical role that data preprocessing and feature engineering play in building effective machine-learning models for text classification.