<u>506 Midterm Writeup</u>

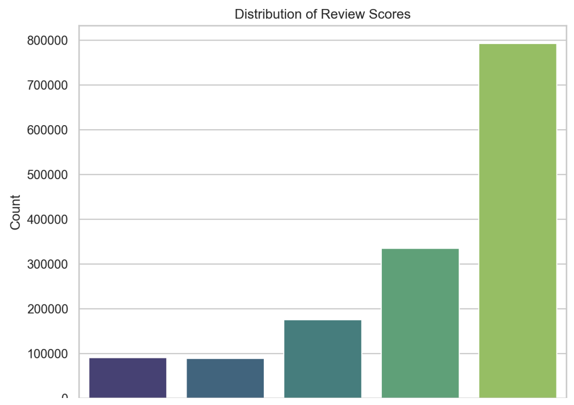
<u>Project Overview:</u>The primary purpose of this midterm was to develop a predictive model for movie reviews with optimal classification performance. This performance was measured on Kaggle, and I used data preprocessing, feature engineering, and data analysis to get my final submission. There were a few constraints put on us for this midterm, using neural networks and deep learning.

Data Preprocessing

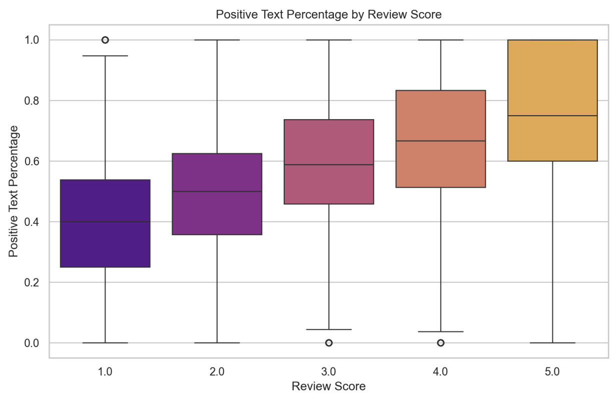
The data preprocessing script prepares the dataset by creating sentiment based features and review statistics that contributed to the models performance. I utilized publicly available positive and negative word lists from GitHub as the basis for sentiment analysis. Additionally, I handled negation words like "not" and "never" by creating combined terms (e.g., "not good" becomes `notJOINEDKEYWORDgood`). This step allows the model to differentiate between negated sentiments ("good" vs. "not good") and interpret them correctly, reducing the likelihood of misclassifying reviews with negations.

Sentiment Feature Engineering:

1. <u>Positive and Negative Word Counts and Ratios:</u>These are the counts of the positive and the negative words, as well as the positive-to-total word ratio, providing insights into the general sentiment of each review. I also included **text length** and **exclamation count** to capture the intensity and structure of the reviews.

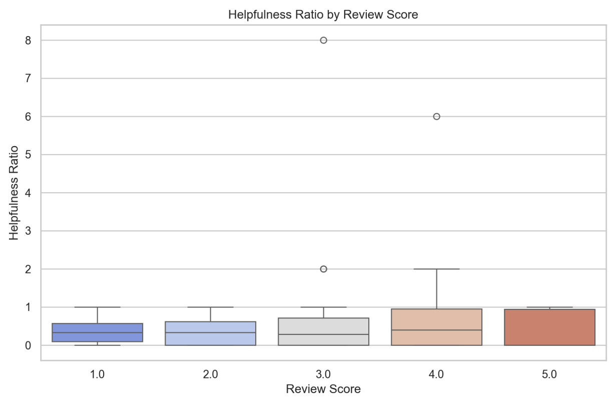


2. <u>Distribution of Review Scores</u>:This chart shows us that the higher scores (especially 5) are more frequent than lower scores. Recognizing this imbalance was essential, as it suggested that the model would benefit from techniques addressing class imbalance to ensure accurate representation of lower scores.



3. <u>Positive Text Percentage by Review Score:</u> This box plot demonstrates that higher review scores correlate with higher percentages of positive words in the text, in an increasing fashion from 1 to 5. This confirmed for me that the model can use sentiment ratios to effectively distinguish between positive and negative reviews.

Aggregating Review Statistics:



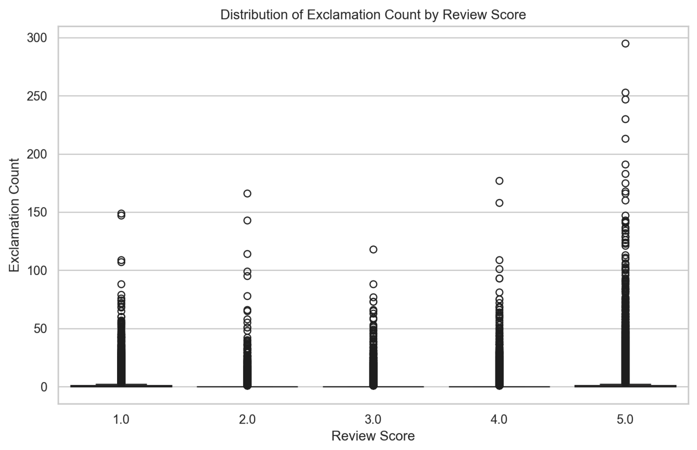
1. Helpfulness Ratio:This plot shows us that the helpfulness ratios tend to be higher for more extreme scores. This suggests that polarized views may be seen as more helpful, possibly because they constrain stronger language or clearer opinions. By including helpfulness, the model can learn to weight reviews based on their perceived use to users.

2. AverageReview and StdReview:The average score and standard deviation for each user's reviews indicate consistency or variability in user ratings. This background information is important, as it helps the model account for user behavior patterns.

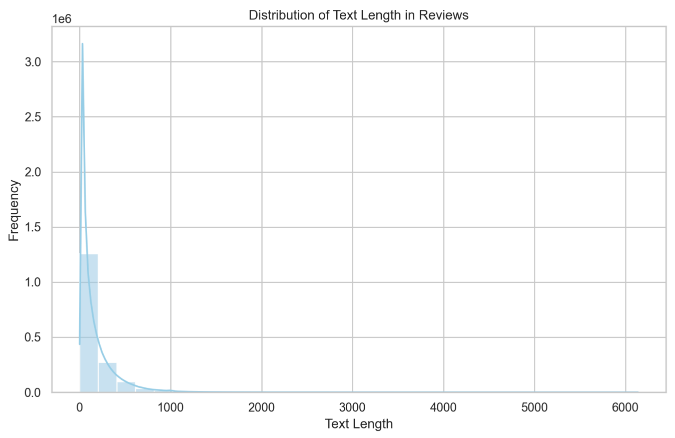
Feature Engineering and Model Training

After preprocessing, I selected relevant features for model training that capture sentiment, structure, and user tendencies.

1. Selected Features: Key features included **Helpfulness**, **NumNegativeText**, **PercPositiveText**, **TextLength**, **ExclamationCount**, and aggregate review counts.



2. Exclamation Count by Review Score: This distribution indicates that higher rated reviews tend to use more exclamation points, except for 1 star reviews. This supports our decision to use it as a feature, as it contributes to sentiment intensity, recognizing enthusiasm or strong feelings as associated with very high or very low scores.



3. Text Length Distribution: The histogram reveals that most reviews are short, with fewer longer reviews. Text length helps differentiate between brief, direct reviews and more detailed, often extreme reviews, which can correlate with higher or lower scores.

4. <u>Model training with XGBoost:</u>

For this project, I decided to use XGBoost. I was already familiar with the tool from a previous conversation with my older brother who works in data science. I chose XGBoost for many reasons, good handling of high-dimensional data (which KNN struggles with), built-in regularization, and efficient memory usage and speed. It also allowed me to see feature importance, making it easier to decide which features I wanted to keep and which were not important for my model (another feature that KNN does not have inherently). I used it by setting objective=”multi:softmax”, and configured XGBoost to classify reviews into one of five possible scores. After training, I evaluated the models accuracy on the test set and analyzed some feature importance. This allowed us to give features like PercPositiveText and Helpfulness more priority, leading to more accurate score predictions. Each of the features was chosen because it provided important information: Sentiment Features (NumNegativeText, NumPositiveText, PercPositiveText, etc.) to capture the sentiment of the review text and summary, Structural Features (TextLength, ExclamationCount) to reflect the tone and verbosity of the review, and Contextual Features (NumProductReviews. NumUserReviews, etc) to provide background information on the reviewers tendencies. By focusing on these I was able to reduce noise, allowing the model to focus on more important indicators that contributed to more accurate predictions.

Conclusion

The most significant challenge in this project was managing processing time. I implemented several techniques to improve efficiency:

- **Data Structures**: Using sets for `positive_words` and `negative_words` instead of lists made sentiment lookup operations faster.
- **Regular Expressions**: Transforming negated words with regular expressions reduced the need for complex loops, saving processing time.
- **Efficient CSV Handling**: I read the `train.csv` file once, processing each entry on the go instead of loading everything into memory, which minimized memory usage.
- **Feature Selection**: By selecting a subset of relevant features instead of all available ones, the model trained faster without compromising accuracy.

By narrowing this down the model trains faster. XGBoost is also inherently efficient because it uses a gradient-boosting algorithm. One of the things that helped me make the most progress on this assignment was the two public github repositories I found with the list of positive and negative words which I will cite here:
Positive:
https://gist.githubusercontent.com/mkulakowski2/4289437/raw/1bb4d7f9ee82150f339f09b5b1a0e6823d633958/positive-words.txt
Negative:
https://gist.githubusercontent.com/mkulakowski2/4289441/raw/dad8b64b307cd6df8068a379079becbb3f91101a/negative-words.txt
This assignment was challenging but rewarding, as I watched my model’s accuracy improve through feature engineering and data analysis.