The general steps of this midterm was data preprocessing, feature engineering, and model training. Initially, I undertook data preprocessing to transform raw data into a clean, uniform format suitable for analysis. This involved removing noise such as punctuation and standardizing text to lower case, using the NLTK library for tokenization and removal of stopwords to prepare the data thoroughly. The feature engineering phase followed, where I enhanced the model's learning potential by calculating the Helpfulness Ratio to quantify the perceived value of reviews and transforming the processed text into a new 'ProcessedText' feature via lemmatization to reduce word complexity. Lastly, the model training phase involved employing an XGBoost classifier, optimized for sparse data typical of text data represented through TF-IDF vectorization, to effectively discern patterns and classify sentiments.

Since text data requires careful preprocessing before it can be effectively used for machine learning modeling, all text was converted to lowercase to standardize the format, which helps in reducing the complexity of the model by decreasing the number of unique tokens it must learn. Non-alphanumeric characters were removed, simplifying the text data and focusing the model on relevant words. The cleaned text was split into individual elements (tokens), making it possible to apply further text processing techniques like stopwords removal and lemmatization. Commonly used words that do not contribute significant meaning to the sentences (such as "the", "is", "at") were removed to focus the model on meaningful words. Words were reduced to their base or dictionary form, aiding the model in recognizing different forms of the same word as a single entity.

In addition to the fundamental text processing tasks, I implemented a feature called the Helpfulness Ratio, which played a significant role in capturing the perceived value of each review from a user engagement perspective. The computation of this ratio as HelpfulnessNumerator / (HelpfulnessDenominator + 0.01) was a strategic choice. By adding a small constant of 0.01 to the denominator, it helped to handle cases where the HelpfulnessDenominator might be zero, which would otherwise lead to a division by zero error. This approach not only helps with potential computational errors but also slightly smooths the ratio, avoiding excessively skewed values that could result from very small denominators. Such a strategy enhances model stability and reliability, ensuring that the feature remains robust across varied data inputs.

The Helpfulness Ratio itself is there because it directly relates to how users perceive the utility of a review. A higher ratio indicates a review that many found helpful compared to the number who did not, which could correlate with the quality or

persuasiveness of the review's content. This feature serves as a proxy for the review's credibility and effectiveness, which are subtle yet powerful attributes that can influence a model's understanding of what constitutes a high-quality review.

Moving on to the cleaned and normalized text data, I stored the fully preprocessed text in a new column named ProcessedText. This column consolidates all the preprocessing steps—such as lowercasing, punctuation removal, tokenization, and lemmatization—into a single, clean textual feature ready for vectorization and further analysis. By doing so, ProcessedText becomes the cornerstone for most of the feature extraction processes that follow, particularly the application of TF-IDF vectorization. The transformation of raw text into a structured format that encapsulates the essence of the original content without the noise of unstandardized textual data is crucial. It ensures that the subsequent modeling phases have access to high-quality input data, which is critical for the performance of any machine learning model, especially those dealing with nuanced language data like product reviews. This methodical approach to feature engineering, focused on both numeric and text data, is designed to harness the full potential of the available data and drive the predictive capabilities of the model.

I also used XGBoost, an implementation of gradient boosted decision trees, for its effectiveness with sparse data and its robustness in handling various types of features. Its capability to handle large datasets efficiently and its flexibility in tuning made it suitable for this task. To convert text into a format that could be fed into the model, the following vectorization techniques were employed: the TF-IDF Vectorizer method was used to transform the text into a numeric format, emphasizing important but less frequent words in the dataset. It was used to consider unigrams and bigrams, capturing context to some extent while controlling for dimensionality with a maximum of 5000 features. The Count Vectorizer was used in parallel to TF-IDF to test if raw counts would capture different aspects of the text data. Both vectorizers were applied to the Summary field, providing a secondary textual feature set.

The dataset was split into training and testing sets, with 100,000 samples randomly selected for training to manage computational efficiency. The XGBoost model was trained using the following hyperparameters, which were chosen based on preliminary experiments and literature on similar problems: a learning rate was set to 0.1 for moderate speed and to avoid overshooting minima; the max depth was limited to 4 to prevent overfitting; the min child weight was set at 3 to avoid overly complex models that might fit noise; 1000 trees were used to allow the model to learn complex patterns; subsample and colsample_bytree both set to 0.8 to provide randomness in model training and feature selection, enhancing generalization.

Several techniques were applied to enhance model performance: Pre-trained GloVe vectors were utilized to capture semantic meanings of words, which were especially useful for capturing the sentiment and contextual nuances in review text. Additional textual features derived from the Summary were used, such as sentiment scores and text lengths, which provided the model with different perspectives of the data. An initial analysis highlighted common words and phrases associated with high and low ratings, informing the focus areas in feature engineering and the adjustment of vectorization parameters.

The confusion matrix and classification report provided deeper insights into areas where the model performed well and where it could be improved, particularly in distinguishing between adjacent rating classes. I faced challenges like overfitting, which I addressed by tuning the model's hyperparameters and employing cross-validation to ensure the model generalizes well across different data subsets. This midterm was as much about applying machine learning techniques as it was about understanding and innovating on the nuances of text processing and sentiment analysis, aiming to create a predictive system that offers insights into consumer behavior through their reviews.For instance, if the model frequently confused 4-star reviews with 5-star reviews, it indicated a need to further refine the features related to high-quality reviews or to adjust the model's sensitivity to features that differentiate high ratings.

So my final model demonstrated the effectiveness of combining traditional NLP techniques with advanced machine learning algorithms. The use of GloVe embeddings, in conjunction with TF-IDF vectorization, allowed the model to capture a broad range of features from the text data. It could have been improved by exploring deeper linguistic features, alternative embeddings like BERT, or more complex ensemble methods to further enhance performance, but this mad time consuming and I'm over it at this point. The process of feature engineering and model tuning based on exploratory data analysis proved crucial in adapting the model to the nuances of this dataset and also in helping me lose my sanity.