CS506 Midterm Report

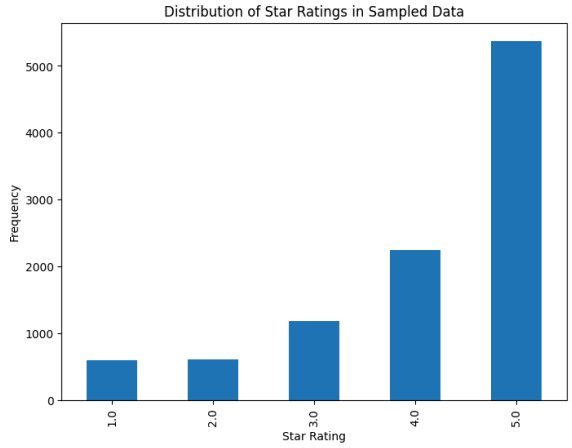
Data Exploration and Preprocessing

Initial Data Exploration

The dataset consisted of Amazon product reviews with features such as 'Summary', 'Text', 'HelpfulnessNumerator', 'HelpfulnessDenominator', and 'Time'. I first wanted to handle missing values. I completed this step by

- Filling in missing values in 'HelpfulnessNumerator' and 'HelpfulnessDenominator' with 0, assuming that missing helpfulness votes imply no votes were cast.
- Dropping rows where 'Summary' or 'Text' was missing, as these fields are crucial for text analysis.

Further, the 'Score' variable ranges from 1 to 5, but the distribution is imbalanced, with certain ratings appearing more frequently. Visualizing this:



Feature Engineering

Helpfulness Ratio

The number of users who found a review helpful ('HelpfulnessNumerator') relative to the total number of users who voted ('HelpfulnessDenominator') could indicate the review's quality.
I created a new feature, 'HelpfulnessRatio', calculated as:
HelpfulnessRatio=HelpfulnessNumerator / HelpfulnessDenominator
This ratio represents the proportion of users who found the review helpful.

Review Lengths

Capturing sentiment can rely on a lot of different things. I figured using the lengths of 'Summary' and 'Text' would be useful features as they reveal more information about the reviewer and review. Longer reviews could be more detailed, potentially reflecting extreme opinions (very positive or very negative).:

- 'SummaryLength': Number of characters in the summary.
- 'TextLength': Number of characters in the review text.

Time Features

Reviews over time might show trends, such as changes in product versions or shifts in customer satisfaction. I converted the 'Time' feature from a timestamp to datetime format and extracted components:

- 'ReviewYear'
- 'ReviewMonth'
- 'ReviewDay'

This allowed the model to consider temporal patterns in reviews.

Sentiment Analysis

The emotional tone of a review is likely linked to its rating. Using TextBlob, we computed sentiment polarity scores for 'Summary' and 'Text':

- 'SummarySentiment'
- 'TextSentiment'

These scores range from -1 (negative sentiment) to 1 (positive sentiment), providing insights into the emotional tone of the reviews. Higher sentiment scores correspond to higher ratings.

Text Preprocessing

To prepare the textual data for modeling, we performed several preprocessing steps:

- Lowercasing: Standardized all text to lowercase to reduce vocabulary size.
- Removing Non-Alphabetic Characters: Eliminated numbers and punctuation to focus on meaningful words.
- Tokenization: Split text into individual words.
- Stopword Removal and Stemming: Removed common English stopwords using NLTK and applied PorterStemmer to reduce words to their root forms.

Both 'Summary' and 'Text' provide complementary information. 'Summary' is a concise overview, while 'Text' offers detailed insights. I applied these steps to both 'Summary' and 'Text', then combined them into a single feature 'CombinedText' to capture all available textual information.

Feature Extraction

TF-IDF Vectorization

We transformed the combined text data into numerical features using Term Frequency-Inverse Document Frequency (TF-IDF) vectorization:

- Limited the vocabulary to the top 5,000 features to balance computational efficiency and model performance.
- The TF-IDF representation captures the importance of words relative to the entire corpus, enhancing the model's ability to differentiate between reviews.

Preparing the Dataset

Selecting Numerical Features

We selected relevant numerical features based on domain knowledge and exploratory analysis, that are most likely to influence the 'Score'.:

- Helpfulness Features: 'HelpfulnessNumerator', 'HelpfulnessDenominator', 'HelpfulnessRatio'

- Length Features: 'SummaryLength', 'TextLength'
- Sentiment Features: 'SummarySentiment', 'TextSentiment'
- Time Features: 'ReviewYear', 'ReviewMonth', 'ReviewDay'

Feature Scaling

We standardized numerical features using MinMaxScaler to ensure that all features contribute equally to the model. Scaling helps algorithms converge faster and improves performance by preventing features with larger scales from dominating the model.

Combining Features

We concatenated the scaled numerical features with the TF-IDF text features to form the final feature matrix X.Combining both numerical and textual features provides the model with comprehensive information to make accurate predictions. The target variable y was the 'Score' from the dataset.

Model Selection and Training

Handling Class Imbalance

The 'Score' distribution is imbalanced, with some ratings occurring more frequently. To address this:

- I computed class weights using compute_class_weight from scikit-learn Applying class weights helps the model pay equal attention to minority classes, improving overall performance..

Model Choice: Multinomial Naive Bayes

Upon research on which models are able suitable for text classification and capable of handling high volumes of data, I selected Multinomial Naive Bayes (MNB)

Hyperparameter Tuning

Fine-tuning model parameters can significantly improve performance. We performed GridSearchCV to optimize the 'alpha' parameter of MNB. Fine-tuning model parameters can significantly improve performance, the 'alpha' parameter in MNB controls smoothing; adjusting it can prevent overfitting or underfitting.:

- Tested values [0.1, 0.5, 1.0].
- Used 3-fold cross-validation to assess model performance.
- Selected the model with the best average accuracy across folds.

Model Evaluation

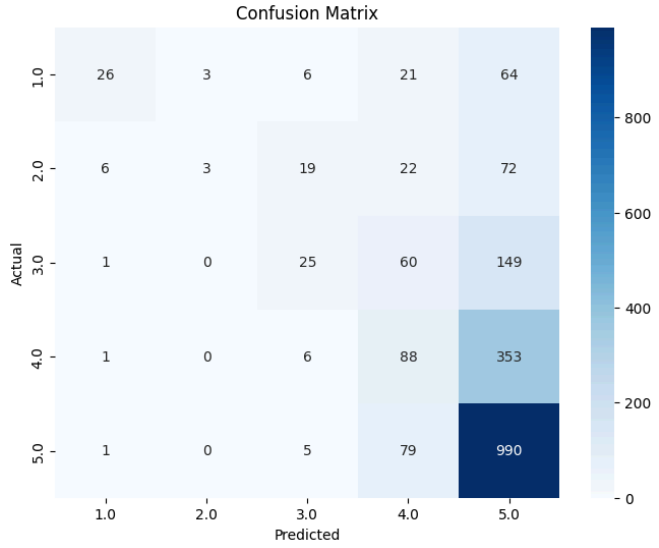
Performance Metrics

We evaluated the model on a validation set using:

- Accuracy Score: Overall correctness of predictions.
- Classification Report: Precision, recall, and F1-score for each class.
- Confusion Matrix: Visualization of true vs. predicted class distributions.

Observations

- The model achieved a reasonable accuracy, indicating effective learning from the features.
- Analysis of the classification report and confusion matrix helped identify classes where the model underperformed, guiding further refinements.



Preparing the Test Set and Making Predictions

We applied the same preprocessing and feature transformation steps to the test set:

- Ensured consistency in feature scaling and vectorization.
- Used the trained TF-IDF vectorizer and scaler without refitting to prevent data leakage.
- Predicted 'Score' for the test set using the optimized MNB model.

Assumptions Made

- Consistency Between Training and Test Data: Assumed that the test data follows the same distribution and characteristics as the training data.
- Effectiveness of Sentiment Analysis: Relied on TextBlob's sentiment analysis to generalize well across different reviews.
- Sufficiency of Selected Features: Presumed that the engineered features capture the essential information needed for prediction.