Midterm Report

Predicting Star Ratings for Amazon Movie Reviews: An Approach Using LightGBM

In this project, I aimed to build a machine-learning model to predict the star rating of Amazon movie reviews using various features from the dataset. After exploring several potential algorithms, I selected LightGBM (Light Gradient Boosting Machine) as our final classifier due to its efficiency and capability to handle a mix of categorical and numerical data. LightGBM's gradient-boosting approach is well-suited for large datasets, making it a solid choice for this task.

Data Preprocessing and Feature Engineering: The dataset provided included several beneficial features, such as HelpfulnessNumerator, HelpfulnessDenominator, Time, Summary, and Text. However, to maximize the model's performance, I added some extra features, which included:

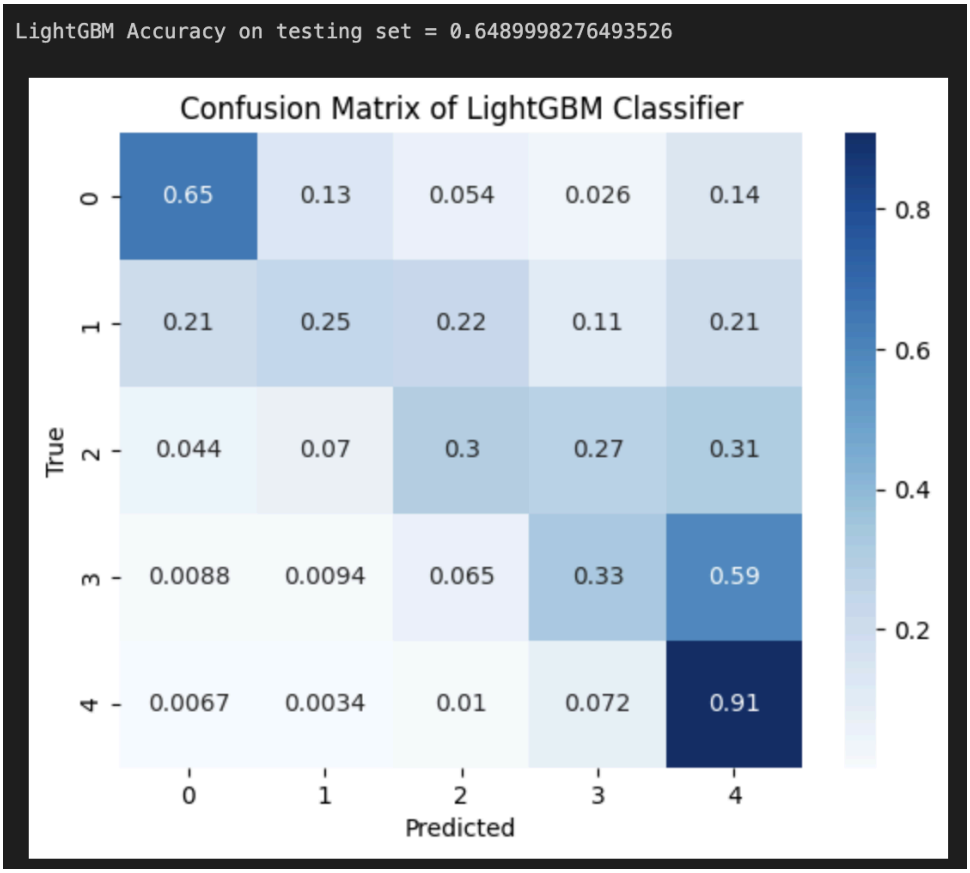
1. **Helpfulness Ratio**: One of my initial observations was that helpfulness scores might indicate the quality of reviews, as reviews deemed "helpful" by other users could be considered more reliable or insightful. I utilized the existing Helpfulness Ratio feature to quantify this, which was calculated as the ratio of HelpfulnessNumerator to HelpfulnessDenominator, providing a numerical representation of how helpful other users perceived a review. I filled any missing values in this ratio with 0, assuming that reviews without helpfulness votes had not received significant engagement.
2. **Text and Summary Length**: I also added Summary_length and Text_length as features to represent the character counts in each review's summary and main text, which allowed

the model to consider whether longer, potentially more detailed reviews tended to correlate with higher ratings.

3. **Sentiment Analysis**: Using NLTK's SentimentIntensityAnalyzer, I generated a Summary_sentiment and a Text_sentiment feature by calculating the sentiment score of each review's summary and text. The sentiment score, which ranges from -1 (very negative) to 1 (very positive), provided insights into the emotional tone of the review, allowing the model to differentiate between positive and negative sentiments, which was very helpful in distinguishing between high (positive) and low (negative) star ratings.
4. **Text Vectorization with TF-IDF**: To include the textual data in the Summary and Text column, I used TF-IDF (Term Frequency-Inverse Document Frequency) vectorization. Focusing on the top 1000 most relevant words, I captured the most essential vocabulary words without overwhelming the model. Especially with the .csv file having one million+ data entries, limiting the feature space helped maintain computational efficiency and reduced the risk of overfitting.
5. **Combining Numerical and Textual Features**: The next step was to prepare the data for input into LightGBM by combining the numerical and textual features into a single format. Since LightGBM can handle only numerical input, I converted the numerical features into sparse matrices. The TF-IDF-transformed text was stored as a sparse matrix to save memory and optimize processing. I then merged these matrices to create a unified feature set. Combining numerical and text data allowed me to exploit LightGBM's strengths in handling high-dimensional datasets without overwhelming them with excessive memory usage.

Model Selection and Training: After preparing the dataset, I trained the model using LightGBM, a choice motivated by LightGBM's ability to handle large datasets efficiently and its gradient-boosting framework, which helps reduce error rates by iteratively improving weak predictions. Key parameters used in the model included:

1. **Number of Estimators**: After experimenting with different values, I set the number of estimators to 300, representing the number of boosting rounds.
2. **Learning Rate**: The learning rate was set to 0.05, allowing the model to make gradual adjustments that minimized overfitting. A random seed was applied to ensure reproducibility of results.



In conclusion, I built a LightGBM classifier for predicting star ratings based on Amazon movie reviews. Using a combination of sentiment analysis, text length features, helpfulness ratios, and TF-IDF vectorization, I developed a model that can reasonably predict ratings with 64.9% accuracy. While the model performed well in predicting high ratings, further refinement is needed to distinguish between lower ratings accurately.