CS506 Midterm Report

Tsagaandari Battugs U62425399

1. Introduction

The goal of this midterm assignment was to predict star ratings based on Amazon Movie Reviews, specifically leveraging a balanced mix of feature engineering, advanced machine learning models, and hyperparameter optimization techniques. Given the large dataset, I focused on efficient data handling and model optimization to maximize predictive accuracy, particularly through feature extraction and fine-tuned model training.

2. Data Preparation and Exploration

The dataset consists of various review features, including ProductId, UserId, HelpfulnessNumerator, HelpfulnessDenominator, Score, Time, Summary, Text, and Id. After loading train.csv and test.csv, the distribution of the Score feature was helpful to understand the class balance across star ratings. Visualizing this distribution highlighted a skew towards certain scores, which indicated potential challenges for balancing during training.

Observations

- Imbalanced Classes: Star ratings were not evenly distributed, which guided my choice of LightGBM, known for handling imbalanced data well.

- Review-Based Patterns: Some features, such as review length and helpfulness, intuitively appeared related to Score and provided a foundation for feature engineering.

3. Feature Engineering

To capture some relationships between review attributes and star ratings, I engineered several key features:

Sentiment Analysis:

Using TextBlob, I extracted polarity and subjectivity scores from the review text. Polarity provides sentiment direction (positive, neutral, negative), and subjectivity captured the opinion level. These features enriched the model's understanding of the review's emotional tone.

- **Implementation**: I created a get_sentiment() function that applies TextBlob to the Text column, returning polarity and subjectivity. These features add a qualitative layer to quantitative text features like "ReviewLength"
- **Impact**: Reviews with high polarity and subjectivity often correlated with extreme rating (1 or 5), likely because these reviews express strong opinions. Including these features helped improve class reparation during training.

Derive Features:

- **Helpfulness Ratio:** Defined as the ratio of "HelpfulnessNumerator/HelpfulnessDemonitor", this feature measures community feedback on each review. NaN values were filled with 0 to handle reviews with no helpfulness votes
- **Review Length:** Measured as the number of words in "Text", this feature provided insight into verbosity. Longer reviews often had higher ratings, indicating more detailed opinions.
- **Intereation Terms:** I generated interaction features like "Helpfulness * ReviewLength", capturing how detailed and helpful reviews may impact scores. These interactions proved valuablenin making the model more sensitive to nuanced relatioships withi nthe data.

Observations and Patterns:

- **Helpfulness as a Proxy of Engagement:** Highly helpful reviews often received more extreme ratings. This suggested that helpfulness could correlate with review extremity.
- **Review Length and Sentiment:** Longer, highly polar reviews tended to have extreme scores, while short, neatural reviews leaned toward middle scores. Recognizing these patterns informed my feature engineer strategy, adding richness to the dataset.

4. Special Tricks and Adjustments

Given the high-dimensional feature space, I tried using LightGBM after using CatBoostClassifier. LightGBM is a gradient-boosting framework efficient for large datasets with categorical and numerical features. Optuna was selected for hyperparameters tuning due to its advanced optimization capabilities.

LightGBM Tuning with Optuna:

- Optuna's *Trial* object allowed me to specify a hyperparameter search space for key LGBM parameters, including:
    - *num_leaves:* Controlled model complexity; optimized for deeper, specific patterns in user reviews
    - *max_depth:* Balanced overfitttting by restricting tree depth
    - *learning_rate:* Fine-tuned for gradual improvements and model stability.

Special Tricks and Adjustments:

- **Early Stopping and Cross-Validation:** Implemented early stopping to prevent overfitting. Cross-validation was critical, especially with imbalanced classes, providing robust accuracy estimates for parameter choices.
- **Dynamic Parameter Space:** I defined parameter ranges for Optuna that adapted based o nobserved model behavior. For instance, after analyzing intial results, I constrained *num_leaves* to avoid excessive complexity in the model.
- **Feature Interactions and Selection:** Interaction terms were selectively tested, and non-contributing features were dropped after each trial, streamlining the model for faster convergence.

Observations:

- **Feature Importance:** LightGBM's feature importance indicated that sentiment and helpfulness were strong predictors fo "Score", validating the value of feature engineering.
- **Balanced Accuracy:** Optuna tuning improved balanced accuracy by refining the models' ability to distinguish underrepresented classes, addressing the initial skew in "Score" distribution.

5. Process

1. Initial Model with CatBoost and Feature Engineering:
    - I began with a CatBoost classifier and added several engineered features to improve prediction accuracy:
        - Helpfulness: Calculated as HelpfulnessNumerator / HelpfulnessDenominator, providing insight into how users rate the helpfulness of each review.
        - ReviewLength: The word count of the Text field, which could correlate with detailed reviews and higher ratings.
        - HelpfulnessLengthInteraction: A feature derived from Helpfulness * ReviewLength, capturing the interaction between detailed reviews and their perceived helpfulness.
        - SummaryLength: The word count of the Summary field, representing the brevity or verbosity of the summary.
        - Sentiment: Derived sentiment analysis of Text, which helped to measure the emotional tone of reviews.
        - Year and Month: Extracted from the review date to capture potential seasonal or temporal trends in ratings.
    - Using Optuna for hyperparameter tuning allowed careful adjustments to CatBoost parameters, including depth and learning rate, which increased my accuracy on the public leaderboard to 0.56149.
2. Refining Sentiment Analysis:

    - I modified the sentiment extraction by computing polarity and subjectivity from both Text and Summary fields using TextBlob. This improved the accuracy further to 0.58143 with the CatBoost model.
3. Exploring Keyword Features:
    - To test the impact of specific keywords, I created a feature based on the presence of certain positive and negative words in Summary and Text:
        - Positive Keywords: Words like "amazing," "love," "excellent," "awesome," "perfect," etc.
        - Negative Keywords: Words like "terrible," "hate," "worst," "awful," "disappointing," etc.
    - Surprisingly, this feature slightly decreased accuracy to 0.55852. In hindsight, applying these keyword features to the final LightGBM model might have yielded better results.
4. Attempted TF-IDF Feature Engineering:
    - I also attempted feature engineering with TF-IDF to represent textual data more effectively. However, my kernel repeatedly crashed due to memory limitations, preventing further experimentation with this approach.

In the end, I tried using LightGBM and finalized my model with LightGBM due to its superior performance and flexibility. This process underscored the importance of iterative testing and selecting features that truly enhance predictive power.

6. Final Model and Performance Evaluation

The final model achieved an accuracy of 0.58443 (Still under the threshold :<). Key contributions to this improvement included:

- **Sentiment-based Features:** Enhanced class separability by introducing quailitative assessments
- **Optuna's Hyperparameter Search:** Enabled precise tuning of LightGBM's parameters, resulting in robust performace without overfitting.
- **Strategic Feature Interactions: Created features that captured complex, non-linear relationships between review metrics and scores.**

7 . Future Improvements

- **Deep Text Analysis:** Integrating more sophisticated NLP methods, such as word embeddings, could further capture contextual nuances.
- **Enhanced Balancing Techniques:** Additional techniques for balancing might improve minority class predictions even further.