MIdterm Paper

Introduction and Background:

In this competition, my final model uses features such as 'HelpfulnessNumerator,' 'Helpfulness,' 'HelpfulnessDenominator,' 'user_avg_score,' 'neg,' 'pos,' 'neu,' and 'compound,' with LightGBM as the model. All other processes align with the starter code. My primary strategy in adjusting the code was to identify the most relevant features, test the accuracy across a few selected models, and then fine-tune the model's parameters to maximize accuracy.

Algorithm Overview and Implementation:

The overall code modifications I made to the starter code were relatively minimal, mainly involving changes in the feature-processing function and switching to a different model. The final set of features fed into the model included ['HelpfulnessNumerator,' 'Helpfulness,' 'HelpfulnessDenominator,' 'user_avg_score,' 'neg,' 'pos,' 'neu,' 'compound']. Of these, 'Helpfulness,' 'user_avg_score,' 'neg,' 'pos,' 'neu,' and 'compound' are new features I created.

Here's a quick breakdown:

- 'Helpfulness' was already in the starter code, calculating the percentage of 'HelpfulnessNumerator' within total helpfulness, assisting in assessing a review's significance and accuracy (suggesting a positive correlation with its rating).

- 'user_avg_score' captures score-related data without directly using the 'score' column.

- 'neg,' 'pos,' 'neu,' and 'compound' are sentiment analysis features processed from the review text. These sentiment scores represent negative, positive, and neutral sentiment intensities in the text (from 0 to 1), with 'compound' summarizing overall sentiment (from -1 to 1).

Due to time constraints, I only applied sentiment analysis to the text column but would add sentiment-derived features from summaries if I had more time.

Although the final and most accurate model was LightGBM, I tested five models during development: KNN, LogisticRegression, RandomForestClassifier, SVM, and LightGBM. As I don't fully understand the differences and characteristics of each model, I tested each with the same features, recording the highest score and further fine-tuning the parameters.

****Special Techniques and Improvements:****

Initially, I assumed the path to achieving the highest accuracy was using all possible features and finding the most suitable model. To begin, I consulted ChatGPT and Claude for suggestions on predictive features and an initial set of models. From these recommendations, I learned that features like 'Helpfulness' and 'user_avg_score' were valuable and that the top five models were KNN, LogisticRegression, RandomForestClassifier, SVM, and LightGBM. LightGBM was particularly recommended for this problem due to its suitability and efficiency. Therefore, I added 'Helpfulness' and 'user_avg_score' as initial features and tested each of the five models in the notebook, recording the accuracy. Unfortunately, initial results were unsatisfactory, with scores around 0.4-0.5.

At this point, I felt my chosen features might lack relevance, so I reviewed the dataset and noticed that most columns related to user reviews. I believed that a relationship existed between review content and 'Helpfulness,' and that converting review content into a computable feature could work well when paired with 'Helpfulness.' I found and implemented sentiment analysis to generate features like 'neg,' 'pos,' 'neu,' and 'compound.'

Running these additional sentiment features slowed down the code, impacting the implementation of further ideas. However, the results improved accuracy to 0.58-0.59, which was a noticeable increase.

Due to time constraints and the slower processing time from adding sentiment-based features, I attempted to add new features directly to the already processed sentiment-

based submission and train files but encountered issues I couldn't resolve, resulting in unsuccessful feature integration.

****Improvements Considered but Not Achieved:****

First, I wanted to explore relationships between features by draw plots for them, then use this insight to select features. However, I didn't have the time and didn't research how to implement this idea effectively. Second, I tried optimizing the model by adjusting LightGBM parameters, yielding a slight accuracy increase of about 0.0004, which is too minor to be meaningful. While I used ChatGPT to find suitable parameters, I didn't have enough understanding of the model to conduct my own parameter tuning.