Predicting Amazon Review Ratings with Natural Language Processing

Ryan Gilbert

October 28, 2024

Introduction

In this project, I aimed to predict star ratings for Amazon movie reviews using various features extracted from review text, metadata, and user and product identifiers. The primary goal was to develop a model that accurately classifies the reviews' star ratings, providing insight into the factors that drive user satisfaction. The dataset includes approximately 1.7 million reviews with fields such as 'UserId', 'ProductId', 'Helpfulness', 'Summary', 'Text', and the target variable 'Score'. Handling natural language data and sparse categorical information presented unique challenges, especially when balancing feature selection and model accuracy. This project was completed for my CS506 midterm. Details can be found here.

Data Description

The training dataset, 'train.csv', includes the 'Score' column, which I used as the target variable. Other fields, like 'HelpfulnessNumerator' and 'HelpfulnessDenominator', indicated the helpfulness ratings of reviews. I used the 'Summary' and 'Text' fields as primary sources of textual information for feature extraction. The test dataset, 'test.csv', consists of similar fields, excluding 'Score', which I aimed to predict. To prepare the data, I addressed missing values and encoded categorical variables such as 'UserId' and 'ProductId'.

Final Algorithm

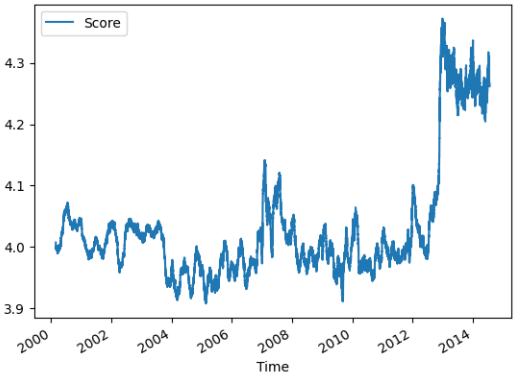
After testing multiple models, I selected a LightGBM classifier, known for its ability to handle large datasets efficiently and for its built-in categorical handling. LightGBM also offers gradient boosting, which helped improve my model's accuracy by iteratively refining predictions.

The final model pipeline includes:

1. **Text Processing:** I tokenized and applied a TF-IDF transformation to the 'Text' and 'Summary' columns. Using term frequency-inverse document frequency reduced the vocabulary size and sparsity, making the text data more manageable. In addition, I applied LDA and LSA to this transformation to reduce dimensionality. I discovered LDA by researching supervised decomposition tools. I also used Vader and TextBlob for sentiment analysis as well as extracting other text based information to provide additional features for the model. I checked the texts for substrings like 'five star' as it showed promise as a feature in the model.

Ultimately, I passed in TF-IDF features directly into the model alongside further dimensionality reduced vectors. I found this increased my accuracy; however, I was heavily limited by my hardware in the amount of max features I could extract.

2. **Feature Engineering:** I gathered descriptive statistics about 'UserId' and 'ProductId' which ended up being extremely important to the model. I also extracted time based features from the timestamp, as there appeared to be non-stationarity between time and scores.



3. **Model Training:** I trained the LightGBM model on the processed data, tuning hyperparameters to optimize accuracy with Flaml automl. A final model was trained on all data, which resulted in the highest Kaggle private LB score of 0.67323.

Methods Explored

Several methods were attempted but ultimately did not yield significant improvements for this task. These include:

- **Named Entity Recognition (NER):** I explored identifying named entities within the review text using part-of-speech tagging, with the goal of extracting product-specific keywords or sentiment-indicating terms. However, this approach increased computational complexity without boosting predictive accuracy. I generated a few lists of words or terms that may be helpful. It is possible the text processing with TFIDF made this redundant.

- **Feature Elimination:** I applied recursive feature elimination with the LightGBM model to refine feature selection further, but was unable to complete the running. I also tried manually retraining on subsets of features using the feature importance provided from the all-feature-model. Feature removal decreased performance.

- **Resampling:** I tested upsampling so that there were an equivalent amount of data for each score class. This reduced model performance significantly likely due to LightGBM handling imbalances and the nature of the test and submission set.

- **Ensemble Technique:** A secondary prediction is generated by creating a model for each class (ratings 1-5). Probabilities are inferred from these models. A final model is trained on these probabilities and actual score. The Kaggle private LB score ended up being higher for the regular classifier model.

Each of these methods provided insights into different aspects of the data and its feature space, but due to practical constraints and limited gains, they were omitted from the final model.

Special Techniques and Observations

I applied several special techniques to enhance the final model:

- **Descriptive Stats:** The descriptive statistics on each User and Product, such as mean and standard deviation were the most helpful to my model. By exploring important features, I was able to improve my model.
- **Parallelization:** I leveraged parallelization to speed up computation through the use of joblib.
- **AutoML Model:** I used FLAML AutoML to tune an optimal model.
- **Multi-class Model:** I created a base model for each rating type (1-5), and combined probability predictions in a final model. This technique yielded interesting results.
- **Iterative Feature Engineering:** By understanding which features were important to the LightGBM model, I was able to create similar features to benefit the model. For example, I added more descriptive statistics for product and user after figuring out their importance to the model.
- **LDA for Dimensionality Reduction:** LDA was particularly helpful in reducing the TF-IDF feature space, focusing the model on high-variance components. This improved both computational efficiency and predictive accuracy.

Conclusion

This project highlighted the use of LightGBM combined with TF-IDF text features and additional metadata in predicting Amazon review ratings. Techniques like hyperparameter tuning and LDA dimensionality reduction contributed to a more robust and accurate model. Further improvements might come from exploring domain-specific embeddings or deep learning, but this approach provided reliable results within practical constraints (and competition rules).

Citations

For this project, I relied heavily on my prior experience with machine learning competitions, where I have encountered many of the foundational techniques used, particularly around model selection, feature engineering, and data processing. However, I explored several new natural language processing (NLP) techniques, such as Latent Dirichlet Allocation (LDA) and named entity recognition(NER) to enhance text-based feature representations.

I consulted various sources for additional ideas and implementations. ChatGPT provided helpful suggestions, including feature engineering techniques and exploratory methods to improve model performance. I also referred to the scikit-learn documentation for information on implementing specific techniques like LDA, TF-IDF, and recursive feature elimination (RFE). All code and modifications were based on these combined resources, as well as my understanding from previous ML experience.