CS-506 Midterm report

Introduction

This midterm presented the problem of predicting star ratings for user reviews in the Amazon Movie Reviews dataset. The primary objective was to build a model capable of accurately classifying user-generated star ratings based on available textual and non-textual features.

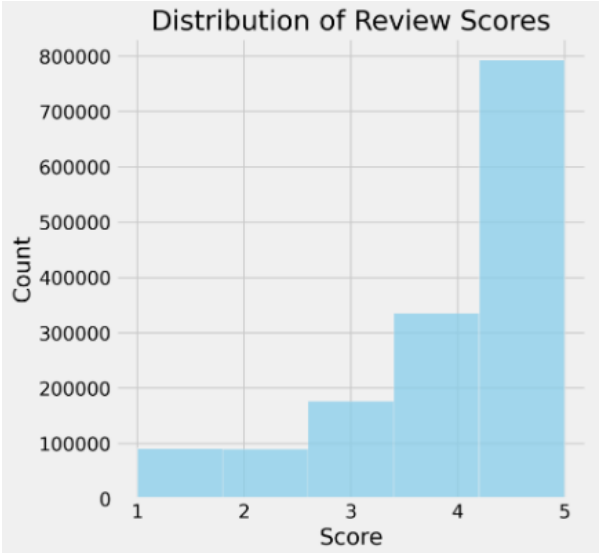
My implementation of the Amazon review classification model achieved an accuracy of 55.535% through a combination of feature engineering, memory-efficient processing, and pattern-driven optimizations. This analysis details the patterns observed and how they shaped the various decisions regarding its implementation and some tricks used to use hardware efficiently.

Assumptions

My implementation relied on several key assumptions. I anticipated that text characteristics, such as text length, would correlate with rating patterns. I also assumed that community engagement metrics, like helpfulness votes, would provide reliable rating signals. Lastly, I assumed that batch processing would offer sufficient efficiency without sacrificing model performance.This last one proved to be true since bigger batches did not improve accuracy in a significant manner.

Core Strategy

To address the prediction problem effectively, a random forest model was selected for its robustness in handling complex data patterns. One of the significant challenges of this assignment was managing the large dataset, as the `train.csv` file totaled aprox. 1.6 GB. Processing and training on such a substantial volume of data presented computational challenges, requiring careful handling to balance model training time and resource efficiency. The batch-processing approach allowed for more efficient memory utilization, reducing the computational load while maintaining data integrity during model development.



An examination of the score distribution in `train.csv` revealed a trend toward positive reviews, with a significant concentration of 5-star ratings. This observation informed a critical implementation decision: to focus on optimizing the model's ability to distinguish strongly positive reviews from others. Given the data's positive skew, prioritizing this differentiation offered a promising approach to leverage the dataset's most consistent signal for improving model accuracy.

Feature engineering based on patterns

The model's interpretation of the data primarily relied on engagement metrics, some of which were directly available in `train.csv` (such as the `Helpfulness_Denominator`), while others were manually engineered, like the `Helpfulness_Ratio` (calculated as `Helpfulness_Numerator` divided by `Helpfulness_Denominator`). These metrics highlighted a strong correlation between community engagement and review sentiment. In addition, the model incorporated text-based characteristics, such as `text_length`, and leveraged Term Frequency-Inverse Document Frequency (TF-IDF) to vectorize words. This approach enabled the model to identify patterns across reviews with varying scores, further enhancing predictive accuracy by capturing both quantitative and qualitative aspects of user feedback.

Model implementation

The core implementation leverages scikit-learn's `RandomForestClassifier`, with parameters fine-tuned through iterative testing to optimize model performance. The ensemble consists of 100 trees, chosen to balance model complexity with computational efficiency; testing showed that increasing the count to 200 trees yielded only a marginal accuracy improvement of 0.002%. This moderate number of estimators ensures diverse decision-making within the ensemble while keeping training times manageable. Each tree in the forest is limited to a maximum depth of 20, a restriction that helps mitigate overfitting by preventing the model from fully memorizing the training data.

To further enhance decision quality at each node, minimum sample constraints were set: the model requires at least 10 samples to split an internal node (`min_samples_split`) and maintains a minimum of 5 samples in each leaf node (`min_samples_leaf`). These constraints help avoid overly granular, potentially noisy decision nodes, ensuring that splits are based on meaningful observation counts.

Given the substantial dataset size, memory management was a critical factor in implementation. A batch processing system with a default batch size of 50,000 rows was employed, enabling efficient handling of large data volumes on limited hardware. This strategy effectively minimized memory strain, facilitating smoother runtime performance for the model.

The model's performance was validated using a 75-25 train-test split with stratified sampling, ensuring a representative distribution of scores. This split ratio provided approximately 346,000 reviews for testing, offering a robust evaluation set for assessing the model's generalization capabilities.

Optimization

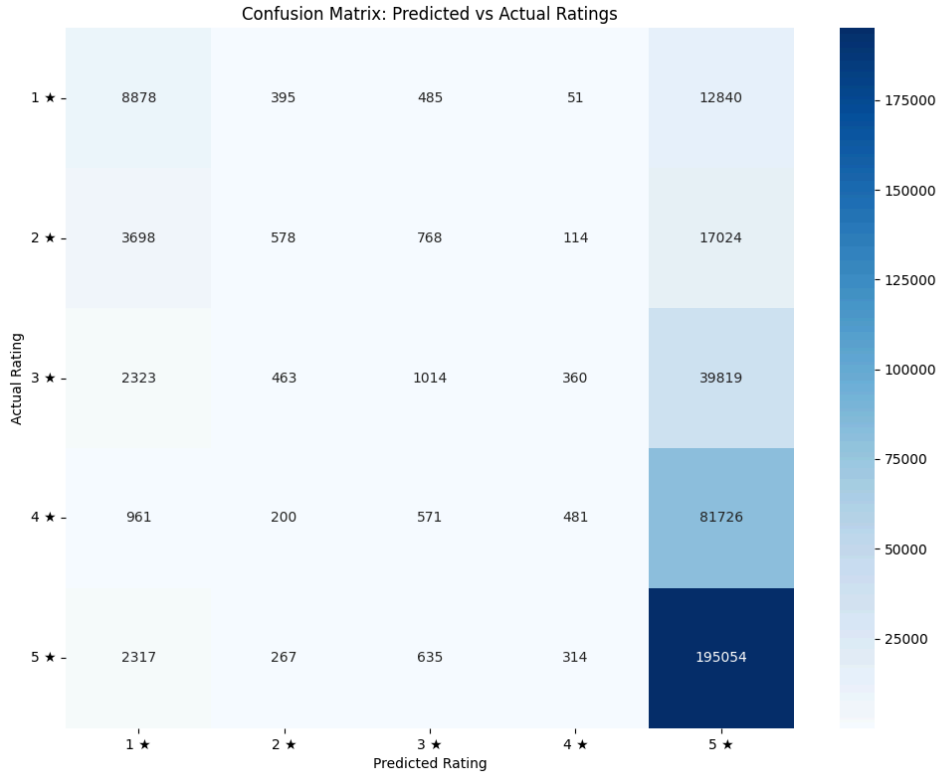
As mentioned above, the biggest optimization technique used was to split the data into batches of 50K rows of data each. Additionally feature extraction is optimized to reduce dimensionality while retaining critical semantic information. TF-IDF vectorization is limited to

the 100 most relevant terms, and vectorized preprocessing operations streamline this step. This optimization achieves notable performance gains without compromising model accuracy. Additionally, the implementation utilizes parallel processing across all CPU cores during both training and prediction, allowing processing speed to scale efficiently with the available hardware.

Results Analysis

The model's results reveal distinct patterns in its prediction tendencies, with a noticeable bias toward higher ratings, particularly favoring 5-star predictions. This bias is consistent with the dataset's actual distribution, which shows a peak at 5-star reviews, and suggests the model effectively captures this dominant trend.

The confusion matrix reveals notable misclassification rates, particularly in mid-range ratings. For example, many actual 3-star reviews are predicted as 4-star reviews, reflecting the model's tendency to overestimate ratings. The matrix also shows a concentration of predictions at the 5-star level, with the majority of actual 5-star ratings correctly identified, demonstrating strong performance with high ratings. However, the model's bias towards higher ratings suggests it struggles to distinguish subtle differences between intermediate scores. This indicates that additional feature engineering or model refinement may be necessary to improve classification accuracy across all rating categories.



Sources:

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

https://realpython.com/python-keras-text-classification/

https://pandas.pydata.org/docs/user_guide/scale.html

https://medium.com/techtofreedom/7-python-memory-optimization-tricks-to-enhance-your-codes-efficiency-5ef65bf415e7