Predicting Amazon Movie Review Ratings Using An Ensemble Model and Custom Features

Laima Ozola-Szoke

Abstract

Predicting the star ratings of Amazon movie reviews offers valuable insights into customer sentiment and product perception. I constructed a diverse set of features, including text-based metrics such as review helpfulness, sentiment polarity, and TF-IDF representations reduced via TruncatedSVD. To optimize classification, I implemented a stratified train-test split and experimented with Random Forest, SGD Classifier, and Ordinal Logistic Regression models. These were subsequently combined in a soft voting ensemble, weighted to balance model strengths. The ensemble achieved robust accuracy on the test set, outperforming individual models, with a particular emphasis on distinguishing high and low ratings effectively.

1 Dataset

The dataset for this project comprises Amazon movie reviews with associated metadata, including review text, helpfulness votes, star ratings, timestamps, and unique identifiers for products and users. My goal was to predict star ratings based on these features, identifying patterns and developing insights to inform my feature engineering and model selection. Before diving into the model training, I first observed the qualities of the dataset in order to get a better understanding of the patterns that I could make use of for my models as well as what features I should include. The observations I made were that, first, the ratings were skewed toward the 4 and 5 stars in commonality whereas the 1 and 2 stars had significantly less data available for them. Next, I also saw that the textual length of the review could vary a lot as well and the actual word patterns present in these reviews could indicate a general "sentiment" that caused the user to leave their particular review (e.g. lower reviews having more critical words). After that, I also noted that the dataset had a helpfulness metric that could be used to assess how helpful a review was and this could be used to quantify a likely pattern where 4 and 5 star reviews were more helpful. Then, another feature that could be noted as a possibly helpful feature for machine learning models was the review age of the review which could possibly show some temporal patterns in ratings if combined with the helpfulness attribute. Lastly, a less impactful feature that I thought of while observing the dataset but nonetheless could be useful was the user behavior pattern, where specifically the total number of reviews left by each user could indicate differing review patterns (users with higher numbers of reviews might leave more moderate ratings). Some challenges that I noted alongside noting these patterns in the dataset were that there was missing and noisy data in this dataset, which indicated to me that there would need to be a rigorous feature extraction for the model to be able to capture the patterns in the data best.

2 Feature Engineering

To improve model accuracy and interpretability, I implemented a diverse set of features reflecting various aspects of review data, such as helpfulness, textual characteristics, sentiment, and user behavior. These engineered features are designed to capture both the direct and nuanced relationships between review attributes and star ratings. As a quick summary before diving into the specifics, the features I chose were the following: helpfulness score combined with flags and buckets to capture the gradient of helpfulness levels, review age, text length, TF-IDF features, sentiment polarity, user review count, and composite features that combined two of the aforementioned features into a combined one. Now, after summarizing the features I used, I will briefly explain more in detail how each feature was calculated and why it was used.

The helpfulness score was first calculated as the ratio of helpfulness numerator to helpfulness denominator, and this feature was hypothesized to be helpful as a review with a higher helpfulness score might be a more reliable indicator of product quality and therefore of the star rating. As a result of this reasoning, I additionally made a feature called the high helpfulness flag that was set to 1 if the helpfulness was greater than 0.5 in order to let the model easily see which reviews were particularly helpful quicker. Lastly, in order to further give some nuance and assistance to the model to really understand the patterns of helpfulness, I created another helpfulness bucket feature that categorized helpfulness into 'none' (0% helpfulness votes), 'medium' (up to 70%), and 'high' (greater than 70%).

A TF-IDF (term frequency-inverse document frequency) approach was used on the text of the reviews to capture word importance in a compressed numerical format, and this was limited to the top 200 terms by importance to reduce dimensionality. To quickly elaborate on what TF-IDF is, it's the "calculation of how relevant a word is in a given text" (GeeksforGeeks, 2021c). "The meaning increases proportionally to the number of times in the text a word appears but is compensated by the word frequency in the dataset" (GeeksforGeeks, 2021c). On top of this, truncated SVD (singular value decomposition) was applied to reduce the features further to 50 components to capture the most essential language patterns in reviews.

Another chunk of sentiment analysis features that I engineered for this included, first, basic sentiment polarity calculation through TextBlob generating a score for each review, and capturing an overall positive or negative sentiment.

The last set of features I engineered that I wanted to mention was the composite features, which were a combination of the individual features mentioned above. The first composite feature was the sentiment-helpfulness interaction, which combined sentiment polarity with the helpfulness score. The other composite feature I made was review age-helpfulness which was a combination of review age with helpfulness.

3 Model Selection and Tuning

In selecting models for this task, I aimed to balance interpretability, predictive power, and computational efficiency. I experimented with a range of models and ultimately chose a combination of Random Forest, SGD Classifier, and Ordinal Logistic Regression. Each model offered unique strengths that complemented the others, and I used a weighted ensemble approach to combine their outputs for improved performance.

3.1 Random Forest

Random Forest is a tree learning algorithm that works by "creating a number of decision trees during the training phase" and "each tree is constructed using a random subset of the dataset to measure a random subset of features in the partition" (GeeksforGeeks, 2021a). In the prediction phase, "the algorithm aggregates the results of all trees, either by voting (for classification tasks) or by averaging (for regression tasks" (GeeksforGeeks, 2021a). Random Forest was chosen as one of my models because of its capability to capture complex, non-linear interactions between features and the flexibility in handling imbalanced datasets. Additionally, it has high interpretability with being able to see the top features that contributed to the classification (called feature importance scores) as well as a robustness to overfitting which was important to be able to translate the performance of my model on training and testing data onto the submission data.

3.2 Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent is a "variant of the gradient descent algorithm" and uses "only a single random training sample (or a small batch) [...] to calculate the gradient and update the model parameters" (GeeksforGeeks, 2021b). For some more context on what gradient descent is, the primary goal of gradient descent is to iteratively search for an objective function's optimal value (minimum or maximum) and "identify the model parameters that provide the maximum accuracy" (GeeksforGeeks, 2021b). SGD was chosen for this midterm because of its scalability and efficiency with large datasets as it processes data in batches, and it has compatibility with sparse data formats (like TF-IDF features). The SGD classifier used in my midterm was configured to use a logistic loss function (for probabilistic output) and L1 regularization to promote sparsity and prevent overfitting.

3.3 Ordinal Logistic Regression

The last model I used was the ordinal logistic regression model, which is a logistic regression model designed to predict ordinal outcomes which basically means variables with a natural order (like with the star reviews which have an inherent order of $1 < 2 < 3$ ...). Logistic regression estimates the probability of a given event using a logit transformation. I used this model using logisticAT from mord as it does not require intensive tuning and it naturally handles the nature of the ordinal dataset well.

4 Ensemble Methodology

To leverage the unique strengths of each model, I implemented a weighted soft voting ensemble that combines the predictions of the Random Forest, SGD Classifier, and Ordinal Logistic Regression models. In soft voting, each model produced probabilities for each possible star rating and these probabilities were then combined, using predefined weights, to generate a final prediction. The weights assigned were as follows: 0.5 for Random Forest (due to its strong performance in capturing complex interactions and high accuracy), 0.2 for SGD (due to its stability for high-dimensional text features), and 0.3 for ordinal logistic regression (due to its ability to handle ordinal data in a natural way which adds valuable structure). By averaging the predictions from each model, the ensemble reduced variance and produced more stable results.

5 Performance Evaluation and Insights

To assess the effectiveness of my models and the final ensemble, I used classification accuracy as the primary metric, along with a confusion matrix for a more detailed analysis. The evaluation focused on both individual model performance and the ensemble's overall ability to predict the correct star rating. For the individual model performance, the random forest model achieved a strong baseline accuracy of 56.41%, then the SGD classifier achieved an accuracy of 55.91%, and finally the ordinal logistic regression classifier achieved an accuracy of 56.19% upon evaluation of training on the training set with the testing set. After performing the weighted soft voting for the ensemble method, the accuracy that was achieved was 58.76%, which was higher than any of the individual models themselves. The confusion matrix that was generated from the ensemble model's predictions on the test set shows that the ensemble model generally was able to capture well the two "extremes" of reviews, 1 and 5 star, but in-between there started to be a loss of accuracy likely due to not having features that were able to define those in-between reviews as clearly to the models. Additionally, since there was a discrepancy in the number of samples for each review, that could've also affected the way that the model learned to recognize each type of star of review.

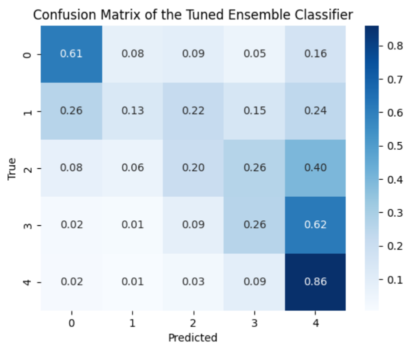

Figure 1: Confusion Matrix for Ensemble Model

6 Conclusion

This project successfully predicted star ratings for Amazon movie reviews by combining feature engineering, model selection, and ensemble learning. Through detailed data exploration, I developed a variety of features that captured nuances in the dataset. The final ensemble model achieved higher accuracy than any individual model, and its weighted soft voting approach balanced each model's unique strengths—handling non-linear relationships, sparse text data, and ordinal consistency. Another point of importance in this midterm was the use of ChatGPT for help with getting started with complex techniques such as TF-IDF as well as debugging with training the models (OpenAI, 2024).

References

GeeksforGeeks. 2021a. Random forest algorithm in machine learning. [Online; accessed 31-Oct-2024].

GeeksforGeeks. 2021b. Stochastic gradient descent (sgd). [Online; accessed 31-Oct-2024].

GeeksforGeeks. 2021c. Understanding tf-idf (term frequency - inverse document frequency). [Online; accessed 31-Oct-2024].

OpenAI. 2024. ChatGPT (October 27 Version). [Accessed: 31-Oct-2024].