AMAZON MOVIE RATING PREDICITON

Ritesh

Amazon Movie Rating Prediction – Github Link – Ritesh Rana

This project predicts star ratings for Amazon movie reviews by examining relationships between features such as user identifiers, helpfulness scores, timestamps, review summaries, and review text. These features are used to estimate the popularity or reputation of movies, which could support recommendation systems for unlisted movies.

You may refer to Kaggle's project page for further details about the competition.

Data Overview

The data consists of over 1.69 million records, each with columns ProductId, UserId, HelpfulnessNumerator, HelpfulnessDenominator, Time, Summary, Text, and Score, including ProductId, UserId, HelpfulnessNumerator, HelpfulnessDenominator, Time, Summary, Text, and Score. Initial visualizations reveal that no single user or product overwhelmingly dominates the dataset, making it unlikely that individual records have an outsized effect on the prediction model.

Distribution of Helpfulness Scores

Using pie charts to visualize the HelpfulnessNumerator and HelpfulnessDenominator distributions, we observed that most scores remain low, with relatively few reviews achieving high helpfulness. This suggests that a vast majority of reviews have limited feedback in terms of helpfulness scores.

Score Distribution

The dataset exhibits a high concentration of 5-star ratings, indicating a bias toward positive feedback. To counteract this in model training, we reduced the dataset size with 5-star ratings to avoid skewing the model.

Data Preprocessing and Feature Engineering

Encoding Categorical Features

One-hot encoding was chosen over label encoding for categorical features (ProductId and UserId). One-hot encoding does not impose any ordinal relationship, preventing unintended bias in model learning. However, the memory cost of one-hot encoding was mitigated by transforming the resulting matrix into a sparse format.

Missing Value Imputation

Missing values in the Text and Summary fields were filled with empty strings. This straightforward imputation approach preserved dataset integrity while avoiding complications introduced by imputation-based assumptions.

Feature Standardization

The Helpful and Unhelpful fields were created to represent the helpful and unhelpful portions of reviews, respectively. These fields, along with the Time column, were standardized to minimize outlier effects and improve model convergence.

Text Vectorization Using TF-IDF

Both Text and Summary columns were vectorized using TF-IDF (Term Frequency-Inverse Document Frequency) to capture the significance of words within individual reviews. This technique reduced overemphasis on common terms, enhancing the meaningfulness of these textual features.

Model Development and Evaluation

Concatenation of Features

To form the final feature set, we combined sparse matrices of text vectors, categorical encodings, and numerical data. This allowed us to handle large data efficiently by avoiding unnecessary memory usage for zero elements.

Train-Test Split

The dataset was split into training and testing sets based on the Score column, ensuring that records with scores were used for training, while those without scores remained in the test set.

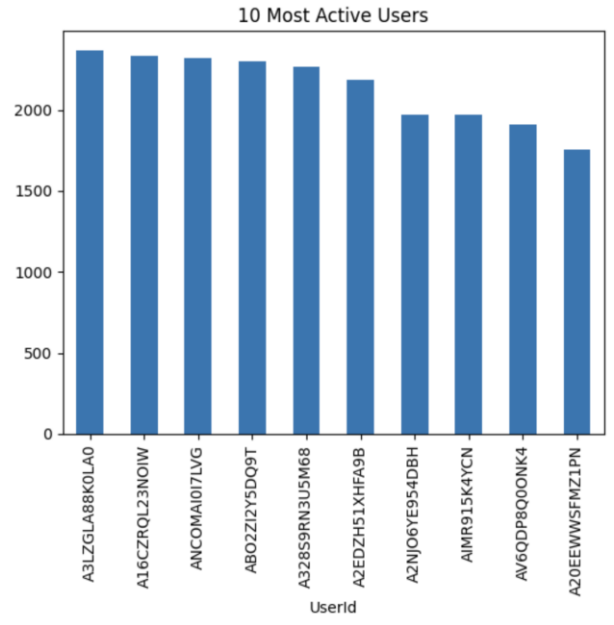
Balancing Class Distribution

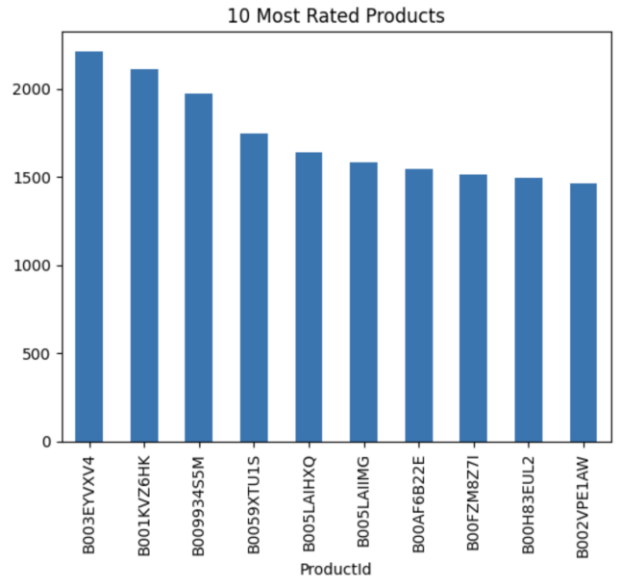
Given the imbalance in star ratings, we used Random Oversampling to balance the class distribution within the training data, reducing model bias toward the majority class.

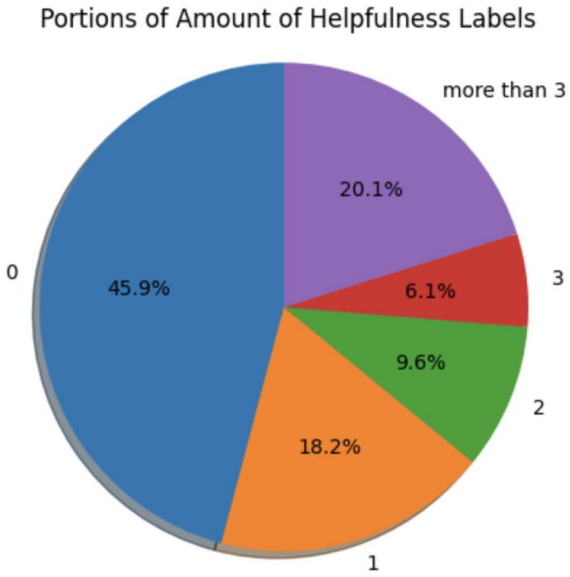
Model Selection and Cross-Validation

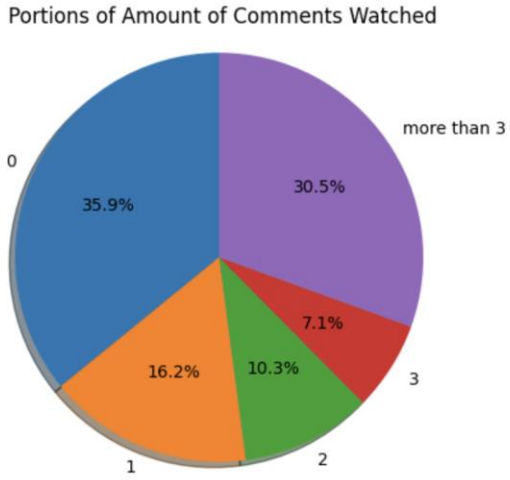
Logistic Regression was selected due to its computational efficiency and robustness for classification tasks. We implemented K-fold cross-validation (with K=5) to evaluate the model's performance across different data folds, selecting the best model based on mean squared error (MSE) from the validation sets.

Data Explorations Graphs









Results and Observations

- **Local Evaluation**: Using the K-fold cross-validation method, the model achieved an accuracy of approximately **74%** on the local dataset.
- **Kaggle Evaluation**: However, when submitting the model on Kaggle, the score dropped to around **65%**. This discrepancy may result from differences in the local test set distribution versus the unseen Kaggle test set, potentially indicating overfitting on the training data.

Conclusion

This project demonstrated the feasibility of predicting movie ratings from textual and categorical features within Amazon movie reviews. Although the model performed well locally, the Kaggle results highlight the importance of generalizability when building robust prediction systems. Future improvements could include experimenting with additional feature engineering, model tuning, and ensemble methods to enhance performance further.