Predicting Amazon Star Ratings: Model Development and Insights

Introduction

This report outlines the development of a predictive model for Amazon star ratings based on customer reviews. The objective was to accurately predict star ratings using the provided data, emphasizing feature engineering and model optimization. We discuss the steps taken, key decisions made, and patterns observed that informed the final model.

Data Overview

The dataset consists of Amazon customer reviews with the following features:

- **Id**: Unique identifier for each review.
- **HelpfulnessNumerator**: Number of users who found the review helpful.
- **HelpfulnessDenominator**: Total number of users who rated the review's helpfulness.
- **Time**: Timestamp of the review.
- **Text**: Full text of the review.
- **Summary**: Brief summary of the review.
- **Score**: The star rating (1 to 5 stars).

The training set includes all features and the `Score`, while the test set lacks the `Score`, which we aim to predict.

Exploratory Data Analysis

Star Rating Distribution

Analysis revealed an imbalance in the `Score` distribution, with a higher frequency of positive reviews (4 and 5 stars). This skewness suggests the need to ensure the model doesn't become biased towards predicting higher ratings. (Figure 1)

Missing Values

Missing values were found in the `Text` and `Summary` columns. Addressing these was important for effective text processing.

Data Preprocessing and Feature Engineering

Handling Missing Values

- **Text and Summary**: Replaced missing values with empty strings to facilitate text vectorization.

Creating the Helpfulness Ratio

- **HelpfulnessRatio**: Calculated as `HelpfulnessNumerator` divided by `HelpfulnessDenominator`. This normalized measure corresponds to the perceived helpfulness of a review.
- **Handling Undefined Values**: Replaced infinite and NaN values resulting from division by zero with zero, assuming no helpfulness votes imply zero helpfulness.

Adding Textual Features

- **Length of Text and Summary**: Computed character lengths to capture verbosity.
- **Word Count**: Counted words in `Text` and `Summary` to quantify content richness.

Vectorizing Text Data with TF-IDF

Implemented Term Frequency-Inverse Document Frequency (TF-IDF) vectorization on the `Text` column to convert textual data into numerical features, capturing the importance of words relative to the corpus.

Dimensionality Reduction with Truncated SVD

Applied Truncated Singular Value Decomposition (SVD) to reduce the high-dimensional TF-IDF vectors to 50 components. This balances information retention with computational efficiency and mitigates overfitting.

Combining Features

Merged numerical features with the reduced text features to form the final dataset for modeling.

Model Selection and Training

Choosing the Random Forest Classifier

Switched from the initial K-Nearest Neighbors (KNN) model to a Random Forest Classifier due to its advantages:

- Handles large feature sets efficiently.
- Captures complex, non-linear relationships.
- Reduces variance through ensemble learning.

Training and Validation

- **Data Split**: Divided the data into training and validation sets (75% train, 25% validation).
- **Model Training**: Trained the Random Forest with 100 estimators and a set random state for reproducibility.
- **Evaluation**:
    - **Validation Accuracy**: Achieved approximately 58% accuracy on the validation set.
    - **Confusion Matrix Analysis**: The model demonstrated higher accuracy in predicting higher star ratings (especially for 4-star ratings), though there is noticeable misclassification among adjacent ratings, indicating some overlap in feature representation across close rating levels. (Figure 2)

Key Insights and Techniques

Importance of Text Data
Incorporating the review `Text` significantly enhanced the model's predictive power by providing sentiment and context not captured by numerical features alone.

Feature Engineering Enhancements
- **HelpfulnessRatio**: Offered a normalized helpfulness metric, more informative than raw counts.
- **Length and Word Count Features**: Allowed the model to consider the level of detail in reviews, which may correlate with satisfaction levels.

Addressing Data Imbalance
Awareness of the skewed `Score` distribution informed the evaluation of model performance and highlighted the need for potential techniques to handle class imbalance in future work.

Dimensionality Reduction Benefits
Reducing dimensionality via SVD improved computational efficiency and reduced the risk of overfitting due to the high dimensionality of text features.

Assumptions Made

- **HelpfulnessRatio Zero Handling**: Assumed reviews with no helpfulness votes have zero helpfulness.
- **Dimensionality Reduction Components**: Selected 50 components for SVD without exhaustive tuning; further optimization may improve results.
- **Text Independence**: Treated each review independently, without considering interactions or temporal effects.

Conclusion

By enhancing the initial model through strategic feature engineering and selecting a more suitable algorithm, we improved the accuracy of predicting Amazon star ratings. The inclusion of textual data and the use of a Random Forest Classifier were instrumental in capturing the nuances of customer reviews.

Future Work

- **Hyperparameter Tuning**: Adjusting model parameters could yield better performance.
- **Advanced Text Processing**: Techniques like word embeddings or recurrent neural networks may capture deeper linguistic patterns.
- **Handling Class Imbalance**: Implementing resampling methods or adjusting class weights could improve predictions for less frequent ratings.

Figures:

Figure 1: \

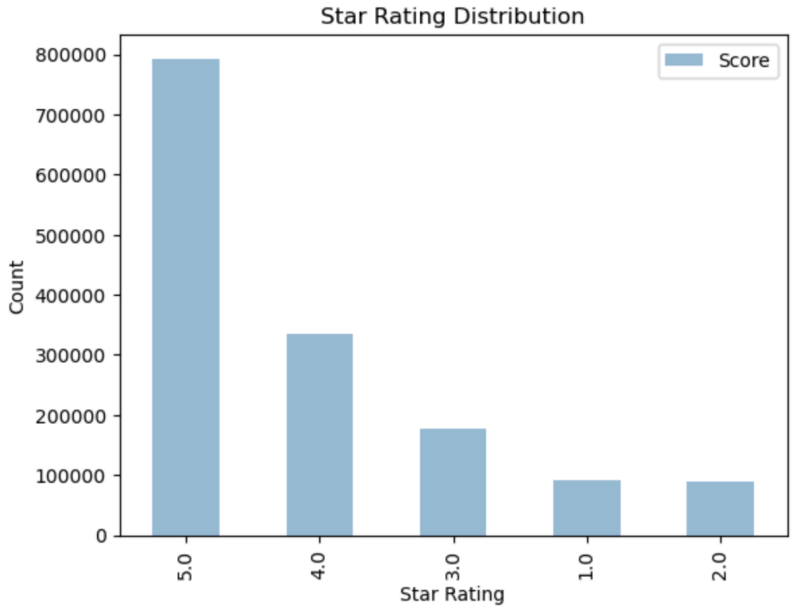
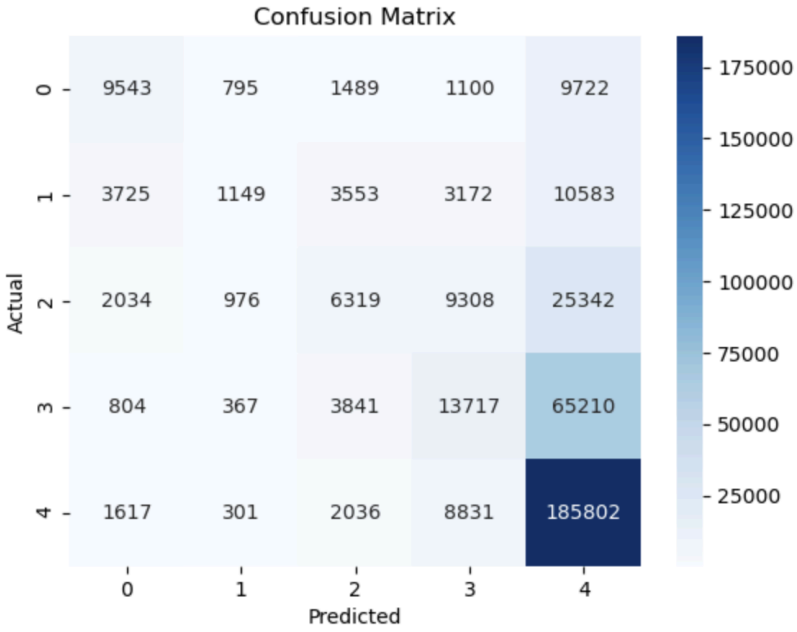


Figure 2:



References:

- Scikit-learn documentation: Random Forest Classifier
- Scikit-learn documentation: Truncated SVD
- TF-IDF Vectorization