Haider Khan
Kaggle Username: haiderkhan12
Professor Lance Galletti
CS506 Tools in Data Science

Amazon Review Prediction Model

Introduction

In this midterm project, we aim to accurately predict the star ratings of Amazon movie reviews by developing a machine learning model which analyzes the review content and associated metadata. The training dataset provided contains over 1.6 million user reviews, each of which featuring attributes such as the review text, summary, helpfulness metrics, timestamps, and unique identifiers for the movies and users. The star ratings are out of 5 and the goal was to develop a learning algorithm that predicts the ratings of the test data based on the available data.

To achieve this, I implemented a Random Forest Classifier due to its robustness and ability to handle high-dimensional data. This classifier is good for large datasets and effectively models complex interactions between features without needing extensive parameter tuning. Since raw data is not enough to develop the model, it’s really important to implement feature engineering to derive informative attributes from the data. By transforming and enriching the dataset, I was able to improve the model's predictive performance and generalizability.

This write-up covers my methodology and some insights I gained during the project, highlighting the patterns observed in the data and the reasoning behind the choices I made.

Data Exploration and Preprocessing

The training set contains 1,697,533 reviews with ten features, while the test set has 212,192 reviews with two features ('Id' and 'Score', where 'Score' is missing). The first step involved a thorough examination of the dataset to identify any inconsistencies, missing values, or anomalies that could impact the modeling process.

The distribution of star ratings is skewed towards higher ratings, with 5-star reviews being the most common. This imbalance could bias the model towards predicting higher ratings. Some reviews lack values in critical fields such as 'Text', 'Summary', or 'Score'. To handle this, I filled missing 'Text' and 'Summary' fields with empty strings to ensure consistency during text processing. Additionally, to ensure data type consistency, I ensured all numerical features are correctly typed as integers or floats to verify that no anomalies or invalid entries could disrupt the model's learning process. By cleaning the data and handling missing values appropriately, I was able to prepare a reliable dataset for feature engineering and model training.

Feature Engineering

To enhance the model's performance, I added additional features that could provide more predictive power:

1. Helpfulness ratio
    a. Calculated as HelpfulnessNumerator / HelpfulnessDenominator, this ratio represents the perceived usefulness of a review.
    b. Addressed division by zero by filling missing or zero denominators with zero in the resultant ratio.
2. Text and Summary Lengths
    a. Computed the character lengths of the 'Text' and 'Summary' fields.
    b. I Assumed that longer reviews might contain more detailed opinions, potentially influencing the star rating.
3. Word Counts
    a. Counted the number of words in the 'Text' and 'Summary' fields.
    b. Aimed to quantify the verbosity of the reviews, providing another dimension to assess content depth.

For textual data, I applied Term Frequency-Inverse Document Frequency (TF-IDF) vectorization to the review text, converting unstructured text into numerical features. To manage computational complexity, I limited the maximum number of features to 5,000. I then used Truncated Singular Value Decomposition (SVD) to reduce the dimensionality of the TF-IDF vectors from 5,000 features to 50, preserving essential information while reducing the feature space to a manageable size. By integrating these features, I provided the model with enriched data that captures various aspects of the reviews, from quantitative metrics to textual sentiment indicators.

Model Training

I chose the Random Forest Classifier for its effectiveness in handling large datasets with mixed feature types and its resistance to overfitting. The training process involved dividing the training data into training and validation sets using a 75/25 split to evaluate the model's performance on unseen data. I set the number of trees ('n_estimators') to 100 to balance between performance and computational efficiency and utilized all available processing cores ('n_jobs=-1') to expedite training.

Combining the numerical features and the reduced textual features from SVD, I created a final feature set for training, ensuring that both types of data contribute to the model's learning. I then trained the Random Forest model on the training split, allowing it to learn patterns and relationships between features and star ratings. The model achieved a validation accuracy of approximately 58%, indicating a reasonable ability to predict star ratings based on the engineered features. The starter code also generated a confusion matrix to visualize the model's performance across different star ratings.

Challenges

I definitely encountered many challenges during development. One big obstacle was the computational complexity associated with hyperparameter tuning for the Random Forest Classifier. Since the default hyperparameters might not yield optimal performance for this dataset, I attempted to use Grid Search to find the best combination of parameters. However, this was impractical due to the massive computational demands of Grid Search on such a large dataset caused my computer to literally crash (blue screen).

These computational limitations also affected the overall efficiency of model training and evaluation. For instance, when attempting to create the model using XGBoost classifier, the computational complexity was far worse. Processing the large text data for TF-IDF vectorization and applying dimensionality reduction techniques like SVD further strained the resources.

Conclusion

Despite my challenges with computational complexity and the skewed distribution of star ratings, I was able to successfully develop a model to predict star ratings for Amazon Movie Reviews by combining feature engineering with a Random Forest Classifier. By extracting meaningful features from both numerical and textual data, I enhanced the model's ability to understand the nuances of the reviews. My approach emphasized the significance of data preprocessing and the thoughtful handling of challenges such as data leakage and class imbalance.