Sofia Boada
CS506 - Data Science Tools and Applications
Fall 2024

CS506 Midterm Fall 2024 Report

Objective

The objective of this midterm competition is to predict star ratings for Amazon Movie Reviews by turning both review text and metadata into predictive features. The key elements include features such as review length and helpfulness scores, as well as advanced text processing using **TF-IDF** and **Latent Semantic Analysis** to capture language patterns. To address the natural rating imbalance, **Borderline SMOTE** was applied, and XGBoost was selected as the classifier for its effectiveness with structured data. This approach combines feature engineering and strategic preprocessing to build a strong model within the project's guidelines.

Data Understanding and Preparation

The dataset contains over 1.6 million Amazon Movie Reviews with features such as *ProductId*, *UserId, Helpfulness* ratings, *Time* of review, *Summary, Text*, and a *Score*.

Initial Observations:

- **Class Imbalance**: Star ratings were not evenly distributed, with some ratings appearing more frequently than others. The dataset was heavily skewed toward 5-star reviews, with 1 and 2-star reviews being rare. This imbalance posed a challenge in accurately predicting less common ratings.
- **Missing Values:** Some values in the Summary and Text fields were missing, so I filled these with empty strings to ensure consistency in text processing (32 and 62 missing, respectively).

Preprocessing:

- **Filling Missing Values:** Missing values in Summary and Text were filled with empty strings, allowing the text processing to proceed without errors.
- **Time-based Features:** Extracted *Year*, *Month*, and *Day of the Week* from the *Time* column to capture any temporal patterns in review ratings.
- **Review Length:** Created a ReviewLength feature based on character count in the Text, helping the model interpret detail in a review.
- **Sentiment Analysis**: Used **TextBlob** to calculate sentiment scores for both Summary and Text, capturing positive or negative tones linked to ratings.

These steps prepared the dataset for modeling and added useful patterns for the model to learn from.

Feature Engineering

Sentiment Analysis

Sentiment scores were calculated using **TextBlob** for both Summary and Text, revealing that higher sentiment scores generally aligned with higher ratings. This pattern was valuable for distinguishing positive and negative reviews.

TF-IDF Vectorization: Term Frequency-Inverse Document Frequency

TF-IDF was applied to the Text field with a vocabulary limit of 15,000 words to highlight the most relevant words. This helped identify important terms without using less useful ones.

Latent Semantic Analysis (LSA)

LSA was used to reduce **TF-IDF's** features to 100 components. This allowed the model to focus on broader themes rather than individual terms. This helped in understanding complex patterns without overfitting on specific words.

Custom Keyword Features

Two features were added to check if the words “excellent” or “terrible” appeared in a review. These words often signaled higher or lower ratings, helping the model identify strong positive or negative reviews.

Each feature was selected to improve the model’s input, capturing text, sentiment, and engagement patterns observed in the data.

Model Selection and Tuning

To predict star ratings accurately, it was important to choose a model that is capable of handling class imbalance and complex data patterns. **XGBoost** was picked because of its ability to capture complex relationships in the data. The process involved balancing the dataset using **Borderline SMOTE** and fine-tuning hyperparameters to improve the models accuracy.

XGBoost Classifier

XGBoost is a method that builds multiple decision trees one after another, with each tree correcting the mistakes of the previous one. XGBoost was used to improve accuracy on challenging, less common ratings like the 2 and 3-star reviews. By adjusting for mistakes in each step, the model handled challenging cases better and balanced predictions across all star ratings.The regularization in XGBoost kept the model from becoming too complex, helping it work well on new reviews without *overfitting* to the training data.

Borderline SMOTE

The dataset had a significant *imbalance*, with far more 5-star ratings than 1, 2, and 3-star ratings. This imbalance showed a pattern where the model initially struggled to predict the less common ratings accurately. To address this, **Borderline SMOTE** was used to create synthetic samples for underrepresented ratings, especially 2 and 3 stars, by adding new examples near difficult cases. This improved the model’s accuracy by giving it a more balanced set of examples to learn from.

Hyperparameter Optimization

Fine-tuning XGBoost’s hyperparameters was important for balancing learning complexity and performance. Key parameters included:

- **n_estimators**: Set to 200 to allow enough boosting rounds for the model to learn detailed patterns without overfitting.
- **gamma**: A **gamma** value of 1 helped prevent the model from making unnecessary splits, improving its focus.
- **class weights**: Custom class weights were applied, increasing the importance of less frequent ratings to further address the imbalance and improve accuracy across all star ratings.

These adjustments were made to handle patterns like class imbalance and the complexity of the data, helping the model predict all ratings more accurately. This process allowed XGBoost to learn important patterns while staying balanced and reliable.

Model Validation

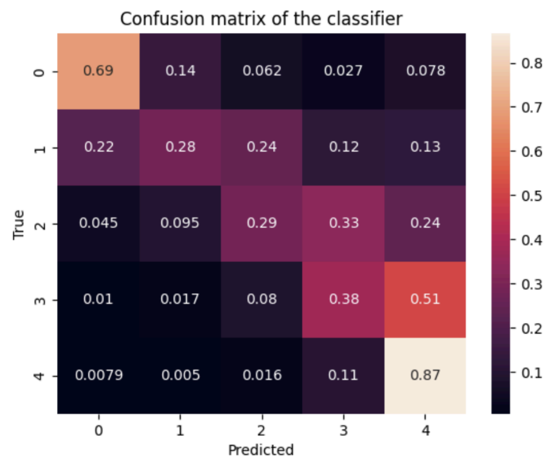
Strategy and Performance

The model was developed using the provided train.csv data, with internal validation for tracking *accuracy*. Key techniques like custom class weights and **Borderline SMOTE** improved the model’s performance, especially for the less frequent 2- and 3-star ratings.

Visualization and Insights

- **Class Distribution:** The initial distribution plot showed the unbalance in 5-star reviews, showing the need for SMOTE and custom class weights.

- **Confusion Matrix:** A heatmap of predictions revealed high accuracy in 5-star reviews but more errors in middle-range ratings. This pattern showed the need for balancing to improve accuracy across all classes.



Challenges, Insights, and Solutions

- **Data Imbalance:** The scarcity of 2- and 3-star reviews initially limited accuracy in these categories. **Borderline SMOTE** and custom class weights addressed this issue, providing more balanced learning.
- **Overfitting**: Regularization parameters in XGBoost, such as gamma and n_estimators, were adjusted to prevent overfitting and ensure generalization to new data.
- **Computation Time:** Given the large dataset, training time was managed by enabling parallel processing with n_jobs=-1, which accelerated training without sacrificing accuracy.

Conclusion and Lessons Learned

Achieving 64.2% accuracy on the testing set took a lot of testing and tuning of data processing, feature choices, and model settings. Getting this accuracy required trying different features and methods to help the model understand patterns in Amazon reviews.

Using **TextBlob** for sentiment analysis added useful insights into the text, allowing the model to pick up on positive or negative tones that often matched ratings. **TF-IDF** and **LSA** helped control the high-dimensional text data. With 100 LSA components, the model could focus on important word patterns without extra noise, showing a clear balance between performance and efficiency.

Borderline SMOTE was used to fix class *imbalance*, especially for underrepresented ratings. This targeted sampling of difficult examples helped the model improve its learning on hard cases, increasing accuracy for ratings like 2 and 3 stars.

Visualization tools like **Seaborn** and **Matplotlib** were essential for understanding the model's strengths and weaknesses. Plots, such as the heatmap and initial rating distributions, helped identify where the model struggled and guided improvements.

XGBoost tuning was challenging, with key settings like n_estimators=200 and gamma=1 playing a huge role. XGBoost also needed high computational power with text data, so normalizing features like Helpfulness and ReviewLength using **StandardScaler** helped the model treat each feature equally.

Overall, training this model required ongoing testing, adjustments, and balance between computational demands and *accuracy*. The final model, while not perfect and had room for improvement, shows how good feature engineering and sampling can be in handling the complexity of predicting star ratings.

Sources:

Brownlee, Jason. “How to Configure Xgboost for Imbalanced Classification.” *MachineLearningMastery.Com*, 20 Aug. 2020, machinelearningmastery.com/xgboost-for-imbalanced-classification/.

Dalakiari, Vassiliki. “Feature Engineering in XGBoost: A Practical Guide.” *Medium*, ITNEXT, 2 Sept. 2024, itnext.io/feature-engineering-in-xgboost-a-practical-guide-7fbafa7dbdbd.

“Understanding TF-IDF (Term Frequency-Inverse Document Frequency).” *GeeksforGeeks*, GeeksforGeeks, 19 Jan. 2023, www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/.

Navlani, Avinash. “Python LSI/LSA (Latent Semantic Indexing/Analysis).” *DataCamp*, DataCamp, 9 Oct. 2018, www.datacamp.com/tutorial/discovering-hidden-topics-python.

“Truncatedsvd.” *Scikit-learn.org*, scikit-learn.org/dev/modules/generated/sklearn.decomposition.TruncatedSVD.html.

“TfidfVectorizer.” *Scikit*-learn.org, scikit-learn.org/1.5/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html.

“Borderlinesmote#.” *BorderlineSMOTE - Version 0.12.4*, imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.BorderlineSMOTE.html.

“Guides: Text Analysis: Textblob Package.” *TextBlob Package - Text Analysis - Guides at Penn Libraries*, guides.library.upenn.edu/penntdm/python/textblob.