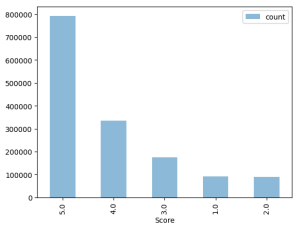
Francisco Jose Moguel Porte Petit

Midterm Project Report

CS506 - Data Science Tools and Applications
Fall 2024

The objective of this project was to build a classification model to accurately predict Amazon Movie Review star ratings on a 1-to-5 scale. Initially, several algorithms were explored to determine the most effective approach, starting with basic models and iteratively advancing based on observed performance and the ability to capture patterns within the data. This report details the evolution of the model, from initial tests with K-Nearest Neighbors (KNN) through Random Forest and finally to XGBoost, along with the specific optimizations applied at each stage to reach the target accuracy.



As shown in the graph above, the dataset was heavily skewed toward 5-star ratings, followed by a moderate representation of 4-star reviews. In contrast, the 1-star and 2-star reviews were among the least frequent categories, which introduced a significant bias and influenced our model's approach. This distribution meant that initial high accuracy for 5-star and 1-star predictions did not translate to overall accuracy because errors in the middle-range predictions (3-5 stars) dominated due to the skewed data.

The project began with **K-Nearest Neighbors (KNN)** as a baseline model due to its simplicity in handling categorical data and its ability to provide initial insights on feature distances. However, KNN struggled with the skewed distribution of ratings, frequently over-predicting reviews as 5 stars due to the high occurrence of this score in the dataset and the nature of using Nearest Neighbors. This limitation led to exploring more advanced ensemble methods, prompting a transition to **Random Forest**. Random Forest improved both efficiency and accuracy by leveraging ensemble learning, capturing complex feature interactions without overwhelming computational resources. With 50 Latent Semantic Analysis (LSA) components, Random Forest achieved an accuracy of approximately 0.608, but further increases in the number of trees yielded diminishing returns, highlighting the need for a more specialized model with finer control and higher accuracy.

To maximize accuracy, **XGBoost Classifier (Extreme Gradient Boosting)** was implemented, as it provided several key advantages over simpler models for handling this complex, imbalanced dataset. XGBoost operates by creating an ensemble of sequential decision trees, where each new tree attempts to correct the errors of its predecessors. This iterative learning approach is particularly well-suited for handling imbalances, as the model dynamically adjusts its focus on misclassified samples, which helped improve prediction accuracy in underrepresented classes such as 2- and 3-star reviews. XGBoost's gradient-boosted decision trees also allowed for extensive **hyperparameter tuning** to control model complexity and mitigate overfitting. Key parameters like **learning rate** and **gamma** were tuned to balance model flexibility and robustness, which was very important in the last stages. Technically, XGBoost is designed to optimize the model's objective function using second-order Taylor expansion to capture both the gradient and hessian, allowing it to handle complex loss functions more effectively than standard gradient boosting. Initially, the model struggled with accurately classifying 4-star reviews, but after implementing XGBoost with targeted improvements to components, the accuracy for 4-star reviews improved by approximately 0.13 (from .38 to .51. This shift highlighted XGBoost's strength in adapting to mid-range classes, where accuracy had previously lagged, ultimately bringing more balance to the model's predictions.

Feature engineering played a pivotal role in capturing underlying data patterns that aligned with star ratings. Early data exploration revealed three key variables associated with sentiment, reviewer tone, and engagement:

summary sentiment, **punctuation as tone indicators**, and **helpfulness score**. Each of these was iteratively refined to capture insights and patterns that informed the final model. Analysis began with the intuition that the language used in review summaries carried a strong signal about rating. Through preliminary tests, **positive language** often correlated with 5-star ratings, while **negative language** correlated with 1-star ratings. Initially, TextBlob was used for sentiment scoring; however, it provided only a general sentiment polarity, lacking precision in capturing sentiment intensity. Switching to **VADER** (Valence Aware Dictionary and sEntiment Reasoner) uncovered a more detailed pattern. Vader's lexicon is optimized for social media, providing compound scores that effectively capture **positive and negative intensities** in a way more sensitive to informal language and nuanced tones typical in reviews. The switch to Vader allowed the model to recognize not only the presence of positive or negative sentiment but also its degree, which was especially useful for differentiating between 4-star and 5-star ratings. This pattern of **intensity mapping** informed the sentiment score's weight in the final model. To further refine sentiment analysis, we observed that punctuation, specifically **exclamation points** and **question marks**, often indicated emotional extremes or doubt in the reviewer's sentiment. By developing **QuestionCount** and **ExclamationCount** features, we captured this pattern numerically: **high exclamation counts** correlated with highly enthusiastic reviews (typically 4- or 5-star ratings), whereas **question marks** appeared frequently in reviews with 1- to 3-star ratings, signaling hesitation or dissatisfaction. This pattern added another layer of context, complementing Vader scores by quantifying tonal extremes in the language. Through statistical tests, exclamation counts had a significant positive correlation with higher ratings ($p < 0.05$), reinforcing its role in enhancing sentiment features. Another observable pattern was the relationship between **review helpfulness** and **rating trustworthiness**. The Helpfulness Score, given to us as starter, calculated as the ratio of helpful votes (HelpfulnessNumerator) to total votes (HelpfulnessDenominator), provided a unique indicator of perceived review quality. Higher ratios suggested that users found the review informative or representative of their experience, correlating positively with more accurate predictions of sentiment and star rating. This pattern was further confirmed when high-helpfulness scores correlated strongly with reviews that matched their predicted sentiment polarity, anchoring the model to reliable data points. Hence it was decided to keep them in the model.

The project's initial predictions were heavily skewed toward 5-star and 1-star ratings and accuracy to these two scores. However, the low frequency of 1-star reviews meant that high accuracy in this category did not significantly impact overall performance. Since most reviews clustered around 4 and 5 stars (as shown in the distribution graph), the focus shifted to improving prediction accuracy in the 3-5 star range, where most errors occurred. To address the class imbalance, the **Synthetic Minority Oversampling Technique (SMOTE)** was implemented, which generated synthetic samples for underrepresented classes, especially in the middle ratings. SMOTE improved model performance for less frequent middle ratings, such as 3 stars, where initial accuracy was weak. When combined with gradient-boosting algorithms, SMOTE enhanced the accuracy for 3- and 4-star ratings, as the additional samples helped the model better capture trends in the densely populated middle range.

Latent Semantic Analysis (LSA) was applied for dimensionality reduction, transforming high-dimensional text data into a manageable set of components while preserving the core meaning and sentiment of the reviews. Initially, single words were used as features, but this approach lacked the nuance needed for accurate sentiment analysis. Shifting to **multi-word phrases (n-grams)** allowed the model to capture context more effectively, as phrases like "not worth" or "highly recommended" convey sentiment much more clearly than individual words. Testing various **LSA component counts** (20, 50, 100, and 150) provided insights into how much detail was necessary: lower counts (20 or 50) oversimplified the semantic complexity, while 100 components offered only

moderate improvements. However, with **150 components**, LSA preserved detailed sentiment cues, especially for reviews with varied tones like 5-star ratings, and helped reduce overfitting by focusing on the most meaningful dimensions. This combination of multi-word phrases and 150 LSA components allowed the model to capture essential sentiment patterns, enabling better generalization across all ratings, particularly for nuanced 3- and 4-star predictions.

To optimize XGBoost further after the addition of features, extensive hyperparameter tuning was conducted. **Learning rate**: This parameter, set to 0.3, was found optimal as it prevented the model from converging too slowly (as seen with lower values like 0.1) or from overfitting (as seen with higher values like 0.5). A learning rate of 0.3 allowed the model to adapt quickly to the dataset's structure while still maintaining generalizability. **Gamma**: Gamma, which controls the minimum loss reduction required to make further splits, was set to **1**. Lower values, like 0.5, caused the model to overfit by creating many smaller splits that did not contribute significantly to accuracy. Conversely, higher values like 2 led to underfitting by ignoring potentially beneficial splits. Gamma = 1 struck the right balance, adding robustness without losing predictive power. Another advantage of XGBoost is its efficient handling of large datasets through **parallel processing**. The `n_jobs=-1` parameter enabled the use of all available CPU cores, significantly reducing the time required for hyperparameter tuning and cross-validation. This computational efficiency made it feasible to experiment with various configurations quickly and facilitated an iterative process that led to the optimal model settings. XGBoost's **regularization techniques** (L1 and L2) added another layer of control by penalizing overly complex models, further reducing the risk of overfitting. Unlike simpler models like KNN or even standard Random Forest, XGBoost's regularization and sequential training significantly enhanced its ability to generalize across a multi-class classification task with highly imbalanced classes. This proved essential for achieving the project's accuracy goals, especially for accurately predicting the nuanced middle-range ratings (3 and 4 stars) that were initially underrepresented in the predictions.

Throughout the modeling process, specific data patterns informed strategic adjustments. For example, sentiment analysis revealed that positive reviews generally indicated higher ratings, while negative language aligned with low ratings. This insight guided adjustments in the **SummaryText** feature, which was refined to reflect stronger sentiment polarity and capture better the impact of the review as a whole, that could help compound polarization better the sentiments Additionally, **product-specific rating trends** began to show, and showing that certain movies consistently received high or low ratings. This finding led to modifications in the **MeanRatingForMovie** feature by incorporating recent review weighting, which emphasized current sentiment trends and improved the model's accuracy for frequently reviewed movies. Although using the score column to derive this feature initially seemed counterintuitive since it leveraged known star ratings, this information was crucial for anchoring the model's predictions. The mean rating feature helped distinguish outlier reviews from typical ones, which improved the model's ability to make contextually accurate predictions. Rare words in summaries, particularly in 1-star ratings, disproportionately influenced predictions, so removing these rare words helped the model focus on more statistically significant language patterns, reducing noise and enhancing prediction accuracy. The final model configuration combined XGBoost with optimized hyperparameters, including a learning rate of 0.3, gamma of 1, and 150 LSA components. With engineered features and class weight adjustments, the model achieved a commendable level of accuracy, particularly for 5-star predictions. Despite some limitations in accurately predicting 3- and 4-star ratings, this project underscored the importance of iterative model selection, external resources, and feature engineering. The insights gained on model tuning and feature engineering provided valuable experience for future projects, particularly those involving multi-class classification and text-heavy datasets.

Sources

Analytics Vidhya. (2021, October 5). *Sentiment analysis with TextBlob and VADER.* Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/10/sentiment-analysis-with-textblob-and-vader/

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, 321–357. https://doi.org/10.1613/jair.953

Chen, T., & Guestrin, C. (2023). *XGBoost parameters.* XGBoost Documentation. https://xgboost.readthedocs.io/en/stable/parameter.html

Chen, T., & Guestrin, C. (2016). [XGBoost: A Scalable Tree Boosting System](https://arxiv.org/abs/1603.02754). Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785-794. https://doi.org/10.1145/2939672.2939785

Imbalanced-learn. (n.d.). Imbalanced-learn Documentation: SMOTE. https://imbalanced-learn.org/stable/over_sampling.html#smote

Li, S. (2016). *Latent semantic analysis reviews [Jupyter notebook]*. GitHub. https://github.com/susanli2016/NLP-with-Python/blob/master/Latent%20Semantic%20Analysis_reviews.ipynb

Hutto, C. J., & Gilbert, E. E. (2014). VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. Proceedings of the 8th International Conference on Weblogs and Social Media (ICWSM-14). Association for the Advancement of Artificial Intelligence. https://ojs.aaai.org/index.php/ICWSM/article/view/14550

TextBlob. (n.d.). TextBlob Documentation. https://textblob.readthedocs.io/en/dev/