Timothy Marro 10/25/24
CS506 - Midterm Analysis

Predicting star ratings for Amazon Movie Reviews was an engaging and challenging task. My goal was to build a machine learning model capable of accurately predicting these ratings using available review data. The task pushed me to delve into data exploration, tackle challenges, and iterate through several modeling approaches. I began with a simple Random Forest model, and that evolved into a refined and more accurate prediction system, and here's a detailed account of what I learned, the patterns I observed, and how I progressively improved my model's accuracy.

When I first started, I noticed the dataset was diverse. It included review text along with some numerical data like helpfulness scores and timestamps. The primary target was the Score, representing the star rating from 1 to 5. However, it quickly became apparent that the data was imbalanced. The 5-star ratings heavily dominated the dataset, while 1, 2, 3, and 4-star ratings were significantly underrepresented. This imbalance initially skewed my model's predictions, favoring the majority class. My first model was a straightforward Random Forest using simple features: TF-IDF (Term Frequency-Inverse Document Frequency) was applied to the review text to convert it into numerical data, numerical fields were scaled for uniformity, and categorical columns were label-encoded. This baseline model achieved an accuracy of around 56% locally. While the model had high recall for 5-star ratings, it struggled with the 1 to 4-star classes. It was clear that the model was biased, heavily favoring the most common rating. This imbalance led to poor precision for minority classes, showing that my initial approach needed more fine-tuning.

Recognizing the limitations of my initial model, I decided to delve deeper into feature engineering to enhance performance. One of the first challenges I faced was the time complexity of my model. Some of my attempts to incorporate advanced features or algorithms resulted in exceedingly long training times or even caused my system to time out. This was particularly evident when I tried using complex natural language processing techniques or more computationally intensive algorithms like gradient boosting machines without optimization.To address the time complexity issue, I turned to Truncated Singular Value Decomposition (SVD) for dimensionality reduction. By applying Truncated SVD to the TF-IDF vectors of the review text, I was able to reduce the feature space significantly without losing too much information. Specifically, I reduced thousands of textual features down to a manageable number of components. This not only decreased the training time but also helped mitigate the curse of dimensionality, which can adversely affect model performance.

In addition to dimensionality reduction, I added new features that could provide more prediction accuracy. I included features such as the length of the review text and summary, word counts, average word length, and counts of unique words. These features aimed to capture the complexity and verbosity of the reviews, which might correlate with the sentiment and, consequently, the star rating. I also noticed that certain punctuation marks and capitalization could indicate strong emotions. Therefore, I added features like the count of exclamation marks, question marks, and uppercase words. These elements often signify emphasis or heightened emotions, which could be indicative of extremely positive or negative reviews.

Understanding that sentiment plays a crucial role in review ratings, I incorporated sentiment analysis by counting positive and negative words within each review. I compiled lists of common positive and negative words and calculated their occurrences in the review text. This approach was a computationally efficient alternative to using sentiment analysis libraries, which would have increased the time complexity beyond an acceptable time. Despite these improvements, I was still encountering issues with model performance and training time. I realized that the Random Forest model's parameters needed optimization. I adjusted hyperparameters like the number of trees (n_estimators), maximum depth (max_depth), and minimum samples required to split a node (min_samples_split). Setting n_estimators to 200 and max_depth to 20 struck a balance between accuracy and training time. Additionally, I used class_weight='balanced' to address the class imbalance problem, ensuring that the model didn't overly favor the majority class.

Even with these optimizations, some of my attempts still suffered from long training times, especially when using large feature sets or more complex algorithms. To further reduce time complexity, I limited the TF-IDF vectorizer's max_features parameter to 5,000 and considered only unigrams and bigrams. This reduction in the vocabulary size helped decrease the dimensionality and, consequently, the computation time. Another significant challenge was handling missing values and ensuring data consistency. I made sure to fill missing text fields with empty strings and numeric fields with appropriate default values to avoid errors. For instance, I filled missing values in the helpfulness ratio with zero to prevent division by zero errors. After implementing these strategies, my model's validation accuracy improved to approximately 58%. While this was a modest gain over the initial 56%, it indicated that the enhancements were making a positive impact. However, I was still aiming for higher accuracy and better generalization.

I considered integrating more advanced machine learning algorithms like LightGBM or XGBoost, which are known for their efficiency and performance on large datasets. However, these algorithms often require careful parameter tuning and can be

sensitive to the specifics of the data. Moreover, incorporating them would have increased the complexity and potentially the time required for training, which I was trying to minimize. Reflecting on the model's performance, I recognized that there were still shortcomings. The model continued to struggle with accurately predicting the minority classes (1 to 4-star ratings). This issue suggested that further steps were necessary to handle the class imbalance effectively. Techniques such as resampling, using SMOTE (Synthetic Minority Over-sampling Technique), or adjusting the decision threshold could potentially improve the model's ability to predict minority classes.

Also, the reliance on TF-IDF and Truncated SVD, while effective in reducing time complexity, might have oversimplified the textual data. Important nuances and contextual information could have been lost in the dimensionality reduction process. Exploring alternative text representation methods, such as word embeddings with models like Word2Vec or GloVe, might capture semantic relationships better, but they also come with increased computational costs. Additionally, feature interactions were not deeply explored in my model. Random Forests can capture some interactions, but explicitly engineering or selecting interaction features might have uncovered valuable patterns. However, this would require additional time and careful consideration to avoid overfitting.

In conclusion, developing this model was a balancing act between enhancing predictive accuracy and managing time complexity. By consistently redefining the feature set, optimizing model parameters, and employing dimensionality reduction techniques, I was able to improve the model's performance without exceeding computational constraints. While the final accuracy of 58% indicated that there was still room for improvement, the journey provided valuable insights into the challenges of working with imbalanced datasets and high-dimensional textual data. The experience showed the importance of thoughtful feature engineering and the need to tailor modeling approaches to the specific characteristics of the data. It also highlighted that sometimes practical limitations, such as computational resources and time constraints, call for trade-offs that can influence the choice of methods and the extent of achievable accuracy. Overall, the project was a rewarding experience that deepened my understanding of machine learning techniques, data preprocessing, and the intricacies of natural language processing. It reinforced the notion that iterative experimentation and adaptation are key components of successful model development.