Name: Shangwei Liu
Class: CS506
Date: Oct 28, 2024

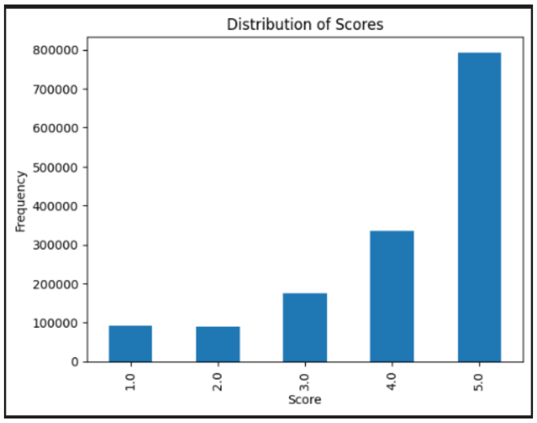
Introduction and Objective

In this project, the objective was to predict the star rating associated with Amazon Movie Reviews based on multiple data types, including numerical, textual, and categorical data. The challenge required managing class imbalance and engineering features that would improve model performance. I leveraged Logistic Regression, a traditional machine learning classifier, along with TFIDF encoding for text features to enhance interpretability and generalization. Each decision in the preprocessing and feature engineering phases aimed to capture informative patterns within the data, helping the model to identify rating-related characteristics more effectively.

Data Preprocessing and Handling Class Imbalance

Something that I noticed from the project is that a common issue in star rating prediction tasks is the overrepresentation of high ratings, particularly 5-star reviews. This imbalance can bias the model, resulting in poor generalization to other classes. To address this, I used downsampling on the majority class (5-star ratings), reducing its count to approximately 21% of the dataset. This step effectively balanced the classes, enabling the Logistic Regression model to learn and generalize across different rating levels without bias toward higher ratings. Before making the downsize, the overall accuracy of the model prediction was around 55%, after the downsize, the average prediction rate was around 62% - 64%, And based on my experiment the best range should be around 40 to 45 % downsize range of score 5. Hence, we can tell that the importance of downsizing significantly impacts the result of the model.

Fig.1 – The image of the Distribution of 5 rating Scores



Categorical Encoding with OneHotEncoder

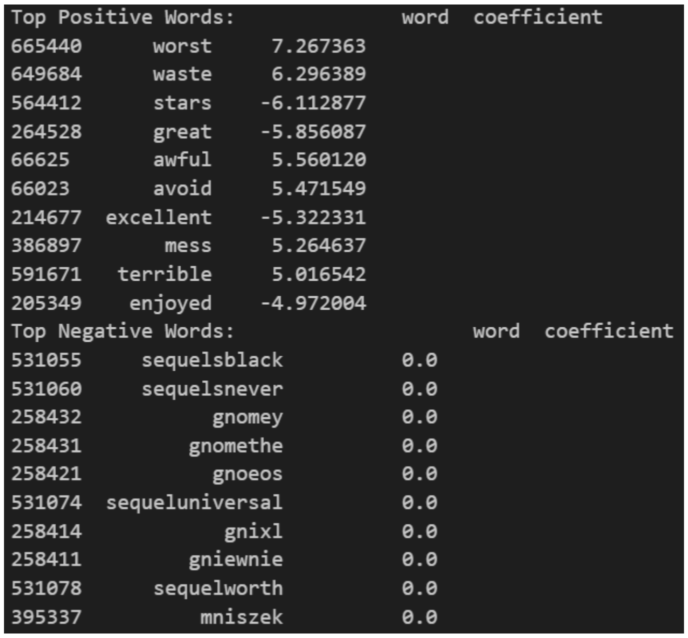
The `ProductId` and `UserId` fields are identifiers with unique but nonordinal values. Using `OneHotEncoder` transformed these categorical variables into a sparse matrix, where each unique identifier was represented as a separate feature without implying any order among them. This method is beneficial because it retains information about each unique product and user without creating misleading relationships. The resulting sparse matrix allowed the model to incorporate categorical information efficiently while avoiding the pitfalls of direct label encoding, which could introduce unintended ordinal relationships. In addition, the other method I tried can take advantage of larget amount of computing due to such a large amount of training set, thereby, the OneHotEncoder is considered as the most effective among other steps such as pandas get_dummies to encode features, will internally lead the python kernel to crashes.

Feature Extraction

The `HelpfulnessNumerator` and `HelpfulnessDenominator` fields were transformed into two features: `Good` (number of people who found the review helpful) and `Bad` (those who did not find it helpful). By using `StandardScaler` to standardize these features, I ensured that they had a comparable scale to other features, preventing any one feature from disproportionately influencing the model. These transformed helpfulness features provided a way to incorporate sentiment indirectly, as reviews with high helpfulness scores often correspond to higher or lower ratings based on the content.

TFIDF Vectorization of Text Features

Textual data, including the `Text` and `Summary` fields, was transformed using TFIDF (Term FrequencyInverse Document Frequency). This method assigns higher weights to words that are unique within a review, making it easier for the model to identify meaningful, sentiment-rich words. TFIDF emphasizes terms that distinguish one review from others, which is particularly useful for sentimentbased ratings prediction. For example, rare but impactful words, like "terrible" or "excellent," can heavily influence the review rating prediction. Using separate TFIDF vectorizers for `Text` and `Summary` allowed us to capture nuanced differences in both fields. One of the advantages of using Logistic Regression with TF-IDF is the ability to analyze word coefficients to understand which terms most significantly impact ratings. (**Figure 2 – The word extracted from the summary and Text**) By examining these coefficients,



```
Top Positive Words:              word  coefficient
665440      worst     7.267363
649684      waste     6.296389
564412      stars    -6.112877
264528      great    -5.856087
66625       awful     5.560120
66023       avoid     5.471549
214677  excellent    -5.322331
386897       mess     5.264637
591671   terrible     5.016542
205349    enjoyed    -4.972004
Top Negative Words:                    word  coefficient
531055     sequelsblack          0.0
531060     sequelsnever          0.0
258432           gnomey          0.0
258431         gnomethe          0.0
258421           gnoeos          0.0
531074  sequeluniversal          0.0
258414            gnixl          0.0
258411          gniewnie         0.0
531078      sequelworth          0.0
395337           mniszek         0.0
```

we could identify the top words associated with high and low ratings, providing insights into sentiment. For example, positive words like "amazing" and negative words like "disappointing" were strong indicators of high and low ratings, respectively. This interpretability allowed us to validate that the model was learning relevant sentiment cues from the textual data.

Model Selection and Training: Logistic Regression with Stratified KFold CrossValidation
Why Choose Logistic Regression? Logistic Regression was selected due to its effectiveness in classification tasks with large feature sets and its interpretability. Unlike neural networks, Logistic Regression allows direct analysis of feature weights, making it easier to interpret the influence of each word or feature on the final prediction. Logistic Regression is computationally efficient with sparse matrices, making it suitable for the extensive feature set generated by TFIDF and onehot encoding.

Stratified KFold CrossValidation
To ensure robust evaluation, One method is to apply Stratified KFold cross-validation, which preserves the class distribution in each fold. I applied this approach to give a more stable accuracy estimate by reducing variance across the folds. According to Geeks for geeks, the validation method can help in verifying that our model generalizes well and that no class, especially minority classes, was overlooked during training. I consider this as a suitable method due to the imbalanced class from the dataset. Additionally**,** the advantage of using Logistic Regression with TF-IDF is the ability to analyze word coefficients to understand which terms most significantly impact ratings. After examining these coefficients, I could identify the top words associated with high and low ratings, providing insights into sentiment. For example, positive words like "amazing" and negative words like "disappointing" were strong indicators of high and low ratings, respectively. This interpretability allowed us to validate that the model was learning relevant sentiment cues from the textual data.

Conclusion & Some potentially reducing calculation methods Or tricks:
Sparse matrices, a compact representation of data with many zero values, were used to store the one-hot encoded categorical features and TF-IDF vectors. This saved computational resources and optimized memory usage, especially crucial given the large number of features generated by encoding and vectorization. SciPy's sparse matrix functions allowed efficient storage and manipulation of this high-dimensional data, improving the model's performance without overloading memory resources. The **scipy.sparse.hstack** function was particularly useful for combining different feature types (TF-IDF, one-hot, numerical) into a single dataset without duplicating data in memory. By addressing the class imbalance, applying feature extraction, and using Logistic Regression, The model is capable of predicting star ratings with a valid accuracy range of around 65 to 67%. The ability to analyze the model's coefficients provided additional insights into how different words and features influenced predictions, contributing to a deeper understanding of review sentiment.

References

Shah, K., Patel, H., Sanghvi, D. *et al.* A Comparative Analysis of Logistic Regression, Random Forest and KNN Models for the Text Classification. *Augment Hum Res* **5**, 12 (2020). https://doi.org/10.1007/s41133-020-00032-0

J. Brownlee, "Why One-Hot Encode Data in Machine Learning?" Machine Learning Mastery. [Online]. Available: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/

GeeksforGeeks, "Text Classification using Logistic Regression." [Online]. Available: https://www.geeksforgeeks.org/text-classification-using-logistic-regression/

GeeksforGeeks, "Stratified K-Fold Cross Validation." [Online]. Available: https://www.geeksforgeeks.org/stratified-k-fold-cross-validation/

M. Ibrahim, "One-Hot Encoding: Creating a NumPy Array Using Weights & Biases," Weights & Biases. [Online]. Available: https://wandb.ai/mostafaibrahim17/ml-articles/reports/One-Hot-Encoding-Creating-a-NumPy-Array-Using-Weights-Biases--Vmlldzo2MzQzNTQ5

W3Schools, "SciPy Sparse Data." [Online]. Available: https://www.w3schools.com/python/scipy/scipy_sparse_data.php