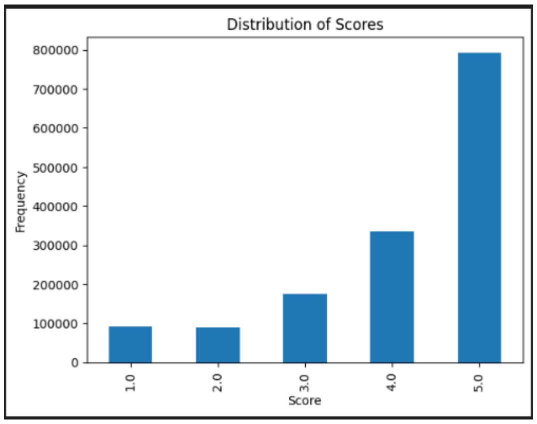
Name: Shangwei Liu
Class: CS506
Date: Oct 28, 2024

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.



Categorical Encoding with OneHotEncoder

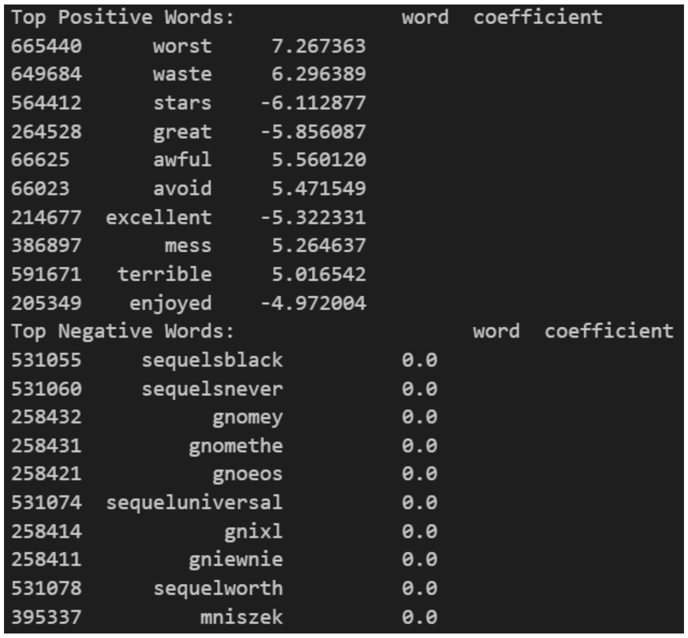
The `ProductId` and `UserId` fields are identifiers with unique but nonordinal values. Using `OneHotEncoder` transformed these categorical variables into a sparse matrix, where each unique identifier was represented as a separate feature without implying any order among them. This method is beneficial 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 in the `Text` and `Summary` fields was transformed using TF-IDF (Term Frequency-Inverse Document Frequency) to assign higher weights to unique or frequent words within a review, helping the model capture sentiment-rich terms. This transformation highlights words that distinguish one review from another, enhancing the model's ability to predict ratings based on sentiment cues like "terrible" or "excellent." Using separate TF-IDF vectorizers for `Text` and `Summary` allowed us to capture nuanced differences in these fields. With Logistic Regression, we could also examine word coefficients to understand which terms most influenced each rating. However, as shown in Figure 2 **(Image on the Left side)**, not all influential words align strictly with positive or negative sentiment; for instance, words like "avoid" or "awful" may appear in "positive" classifications due to association with specific ratings rather than sentiment. Similarly, nonsensical words like "sequels-black" show up with zero coefficients, reflecting no impact despite their presence in the



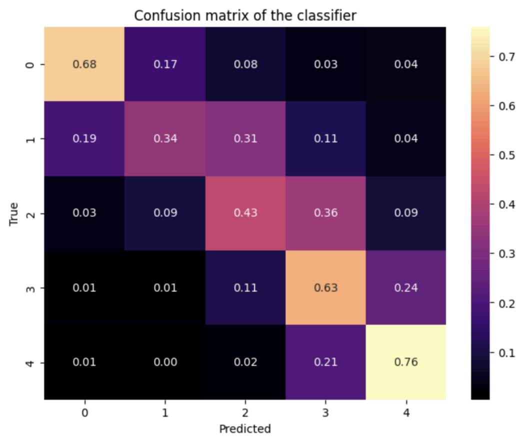
```
Top Positive Words:              word  coefficient
665440      worst     7.267363
649684      waste     6.296389
564412      stars    -6.112877
264528      great    -5.856087
66625       awful     5.560120
66023       avoid     5.471549
214677  excellent    -5.322331
386897       mess     5.264637
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
```

dataset. This examination of word coefficients highlighted areas where further vocabulary cleaning could improve results.

Model Selection and Training: Logistic Regression with Stratified KFold CrossValidation
Why Choose Logistic Regression? Logistic Regression was selected for its effectiveness in handling classification tasks with large feature sets and its interpretability. Unlike other algorithms, Logistic Regression allows direct analysis of feature weights, making it easier to interpret the influence of each word or feature on predictions. It is also computationally efficient with sparse matrices, making it well-suited for the extensive feature set generated by TF-IDF and one-hot encoding. To ensure robust evaluation, we used Stratified K-Fold cross-validation, which preserves class distribution across folds and provides a more stable accuracy estimate by reducing variance. This method helps verify that the model generalizes well, particularly with imbalanced data. Another advantage of Logistic Regression with TF-IDF is the ability to analyze word coefficients to understand the terms that most impact ratings. However, as shown in Figure 2, some of the top "positive" words include terms like "avoid" and "awful," and the "negative" words list contains nonsensical terms, indicating that not all influential words align intuitively with sentiment. This suggests that while the model captures some sentiment cues, additional data cleaning and vocabulary filtering could improve interpretability and relevance.

Conclusion & Some potentially reducing calculation methods Or tricks:

Using sparse matrices to store high-dimensional features from one-hot encoding and TF-IDF vectorization was essential for managing memory and computational resources efficiently. SciPy's sparse matrix functions enabled effective storage and manipulation, while scipy.sparse.hstack allowed us to combine TF-IDF, one-hot, and numerical features without duplicating data. With class imbalance addressed, feature extraction, and Logistic Regression, the model achieved an accuracy range of 65 to 67%.



However, the coefficient analysis showed that certain words' influences did not align with the expected sentiment, with positive and negative terms appearing in "positive" coefficients and nonsensical terms having zero influence. While the model captures some sentiment patterns, improvements could involve refining vocabulary by removing irrelevant terms and trying text representations like word embeddings to better capture context. Additionally, the confusion matrix reveals good accuracy for extreme ratings (0 and 4), but challenges with intermediate ratings, especially rating 2, which is often misclassified as 1 or 3. Future refinements could focus on capturing subtler distinctions in sentiment for moderate ratings to improve overall performance.

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