Logistic Regression with Feature Engineering and Grid Search

Daian Liu, tigerliu@bu.edu
Kaggle: DaianLiuTiger

Data Preprocessing

The train.csv file, containing Id, HelpfulnessNumerator, HelpfulnessDenominator, Summary, Text, and Score columns, was used to build the training data. The test.csv file was loaded to generate predictions for unseen data and construct the submission file.

Feature Engineering:

I created a new **Helpfulness** feature as the ratio of HelpfulnessNumerator to HelpfulnessDenominator, filling any NaN values with 0 to avoid issues during model training.

HelpfulnessNumerator, HelpfulnessDenominator and Helpfulness (Derived Feature)

Reviews that are rated highly by other users might correlate with a positive Score, as more helpful reviews tend to have better content or quality, which could influence the overall rating.

Higher values in HelpfulnessNumerator or Helpfulness might indicate reviews that are more favorable and therefore receive higher scores. Similarly, low helpfulness ratings could correspond to lower scores.

Helpfulness is the ratio of HelpfulnessNumerator to HelpfulnessDenominator, representing the fraction of users who found the review helpful.

The idea is that the proportion of users who find a review helpful is more informative than the raw counts, especially when the total number of users rating helpfulness varies widely across reviews. A high Helpfulness ratio might suggest that the review is well-regarded, which could be associated with a higher Score.

Time

Although it is provided as a timestamp, I think Time can still capture some sort of temporal trends.

Review scores could be affected by temporal factors. For example, older reviews might reflect different user expectations or product quality standards compared to more recent reviews. By including Time, the model can learn any existing patterns based on the review's age.

There might be underlying trends in scores over time, such as product improvement or shifts in user sentiment, that could impact the Score.

One feature I should not have ignored until last: Text

Extracting useful features from text is more complex than numerical data. It requires text preprocessing, vectorization, and often, a larger model or additional memory resources to handle the high dimensionality. I think that text data, once processed using TF-IDF or sentiment analysis, would provide better insights directly correlated with the review score. However, because of the additional processing overhead, I decided to first evaluate the effectiveness of more straightforward features. Which now I realize I should've made it my sole focus.

Model Selection

Given the classification nature of the midterm data set (and how large it is really), logistic regression was my choice due to its interpretability, efficiency with large datasets, and compatibility with hyperparameter tuning through GridSearchCV.

Hyperparameter Tuning and Cross-Validation

To maximize model accuracy, I applied GridSearchCV with cross-validation and tested various hyperparameters:

Regularization Strength (C): A range of values was tested, from 0.01 (strong regularization) to 100 (minimal regularization). Adjusting C allowed us to control model complexity and avoid overfitting.

Penalty Type: Both L1 (lasso) and L2 (ridge) regularizations were explored. L1 can be beneficial for feature selection, while L2 works well for minimizing overfitting with high-dimensional data.

Solver: For optimization, the liblinear and saga solvers were chosen. liblinear is well-suited for smaller datasets and regularized logistic regression, while saga is efficient for large datasets and supports both penalties.

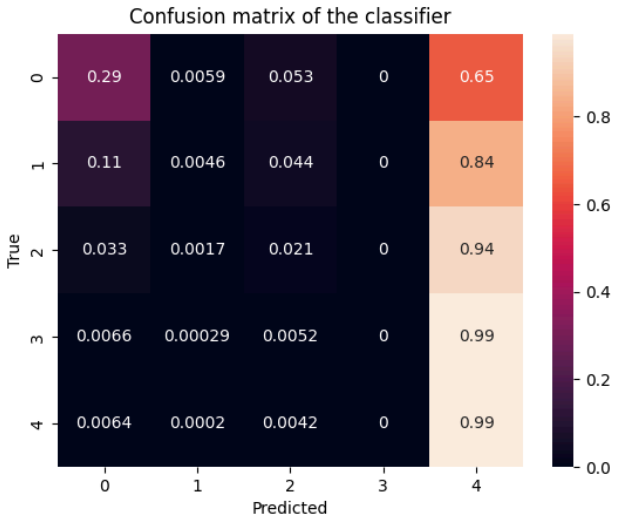
Special Trick: n_jobs=-1 was set in GridSearchCV to leverage multithreading, fully utilizing CPU cores and significantly reducing runtime. (There goes my electricity bills for the month).

Model Training and Evaluation

After selecting the best hyperparameters, the model was trained on the entire training dataset, and performance was evaluated using accuracy and a confusion matrix:

Accuracy: I evaluated the model using the accuracy_score metric on a test split, aiming to maximize this metric as a reflection of correct score prediction.

Confusion Matrix: The model is heavily biased towards predicting higher scores, particularly 4 (the



rightmost column). The model has high accuracy for true labels 3 and 4 (99% correct for each). However, this could mean that it's overfitting to these classes and struggling to differentiate other scores.

Observations: The model initially showed bias towards predicting 5.0 for most entries. This could be due to the inherent class imbalance in the training data or a lack of diversity in feature information. However, further analysis revealed that adding text features could help reduce this bias.

Patterns and Observations

High Bias Towards Positive Scores: Logistic regression displayed a tendency to predict 5.0 frequently, likely influenced by the distribution of scores in train.csv.

Importance of Textual Features: Based on observations, Text features are likely correlated with score predictions, as text sentiment directly affects user perception. Future iterations should incorporate a text-based model, such as using TfidfVectorizer for feature extraction, to improve score differentiation.

Performance Impact of Regularization: Tuning C significantly affected performance (Ranging from 15 minutes to over 1 hour of execution time). Lower C values increased generalization, while higher C values improved fit on the training data, showing logistic regression's sensitivity to regularization.

Future Improvements

Text Feature Engineering: Implement TF-IDF or semantic other methods to capture semantic information from the Text column, potentially improving score prediction by reflecting review sentiment.

Addressing Class Imbalance: Techniques like oversampling minority classes or using class-weight adjustments within logistic regression could help balance the predictions across scores.

Alternative Models: Experiment with other classifiers, such as random forests or support vector machines, which might perform better with non-linear relationships present in the data.

Conclusion

In this midterm, I attempted to showcase the iterative process of feature engineering, model selection, hyperparameter tuning, and evaluation, resulting in a predictive model based on logistic regression. Key takeaways include the model's reliance on sufficient feature variety (including text data) and careful handling of regularization to avoid overfitting. Future enhancements could leverage more complex features to improve prediction accuracy and address biases observed during evaluation.

Conclusion

GRIDSEARCHCV. scikit. (n.d.).
https://scikit-learn.org/dev/modules/generated/sklearn.model_selection.GridSearchCV.html

W3schools.com. Python Machine Learning Linear Regression. (n.d.).
https://www.w3schools.com/python/python_ml_linear_regression.asp

YouTube. (n.d.). *Learn How to Boost Your Python Sklearn Models with GridsearchCV!* . YouTube.
https://www.youtube.com/watch?v=YPQcvFSRKww