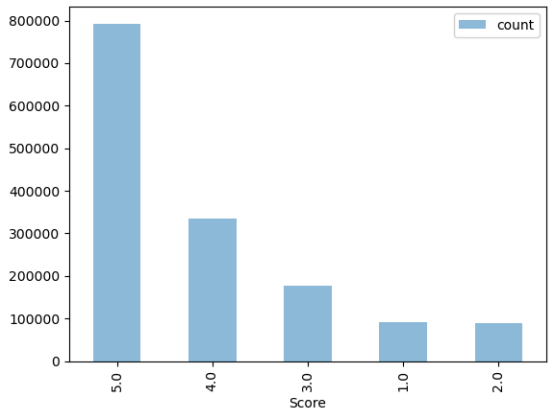
Christopher Min

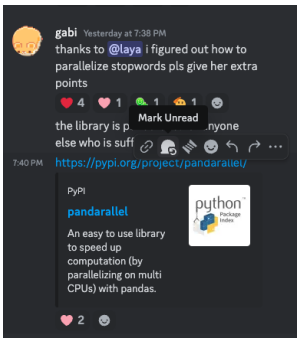
CS506 Midterm Report

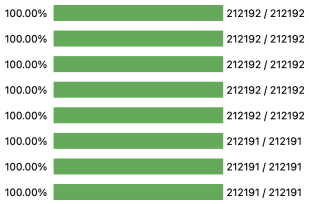


To begin the midterm, I thought it was very important to identify the traits of the data sets that were provided. Immediately, based on the output of the training data, it was evident that there was a skew in the Score distribution, with the majority of reviews clustered around high ratings, particularly 5 stars. This positive bias suggested that users were more likely to leave favorable reviews, which could impact the model's ability to accurately predict lower scores. Thus, I had to focus on making sure that I kept this data skew in mind when creating my model.

To capture multiple dimensions of the data and improve model accuracy, I included various features to represent aspects of helpfulness, user behavior, product characteristics, time patterns, and text sentiment. Each feature was designed with a specific purpose: Helpfulness Metrics: These metrics, including the helpfulness ratio, a binary helpfulness indicator, and weighted helpfulness, quantify how valuable a review might be to readers. By calculating the ratio of helpful votes to total votes, the binary helpfulness indicator (flagging helpful vs. unhelpful reviews), and a weighted measure, we gain a nuanced understanding of reader perception and review influence, which can correlate with higher or lower scores. Time-based Features: Including year, month, day of the week, and quarter captures potential seasonal and weekly patterns in reviews. This is helpful for understanding if certain periods (e.g., holiday seasons) have a correlation with higher or lower ratings. User and Product History: Features like user and product mean scores, user median scores, and user review count add historical context. For instance, users who habitually leave higher ratings or engage heavily might have different scoring tendencies, while product scores reveal how well-regarded certain items are over time. The interaction term (UserProductInteraction) multiplies user and product averages to account for the combined influence of

specific user-product pairs, potentially capturing unique review patterns.

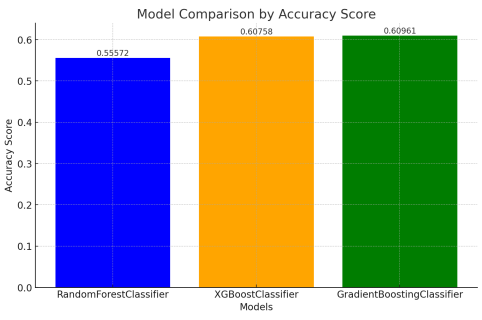




One challenge I encountered with the add_features function was its initial runtime, which ranged from 15 to 20 minutes. Fortunately, after a helpful suggestion from this message, I was able to implement the pandarallel library, reducing the runtime to around 4 to 5 minutes. This improvement in efficiency allowed me to more easily include additional text-based features such as word count, unique words, average word length, and sentiment polarity. By integrating these diverse features, the model gains a more comprehensive perspective on each review, which in turn can enhance predictive accuracy.

Now post adding the features, the model training and evaluation process utilizes a Gradient Boosting Classifier. I used feature selection to streamline the datasets where only the most relevant attributes are chosen and assigned to X_train_select, X_test_select, and X_submission_select for training, testing, and submission, respectively. This targeted feature selection improves efficiency and can potentially enhance the model's prediction accuracy.

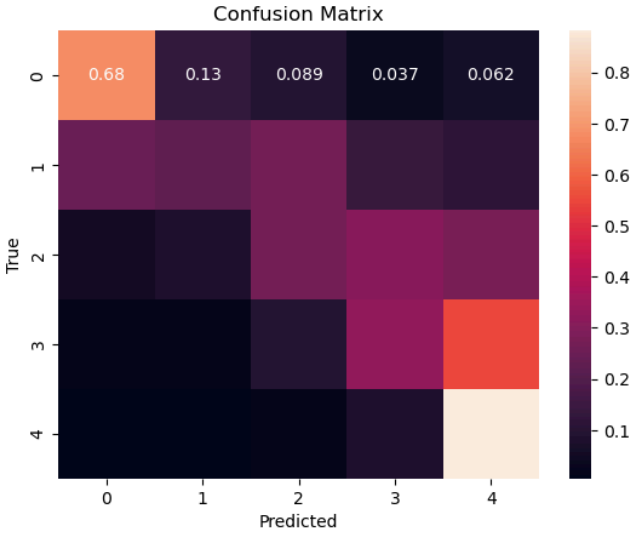
I then utilized GradientBoostingClassifier, which is an ensemble method that combines multiple weak models, typically decision trees, to build a more accurate predictor. Key parameters are set for this model: n_estimators=200 defines the number of boosting stages or trees, helping balance accuracy and computation time, while learning_rate=0.3 adjusts the weight of each tree in the final prediction. Setting random_state=42 ensures consistency and reproducibility in results.



I actually implemented different models beforehand, such as RandomForestClassifier and XGBoostClassifier, to evaluate different ensemble approaches and optimize prediction accuracy. The RandomForestClassifier yielded an accuracy score of 0.55572, while the XGBoostClassifier, a model known for its computational efficiency and predictive power, achieved a higher score of 0.60758. Although these models showed

reasonable performance, the GradientBoostingClassifier provided the most balanced results with an accuracy score that outperformed the alternatives.

Upon completing the model and examining the confusion matrix, we observe that the model generally performs well in classifying reviews according to their true ratings. For instance, for reviews classified in the 0 class, the model correctly predicts the true rating approximately 70% of the time, indicating reasonable accuracy for this rating category.



For classes 1, 2, and 3, we can see that our model does struggle a bit, where it hovers around .3, .4 and around .45 respectively. But for reviews classified in the 4 class, it does an exceptionally good job, where it predicts the true rating approximately 80% of the time. Considering this, we can see that the model does a great job identifying extreme reviews, in both the positive and negative lights as they most likely have easier to recognize patterns in their reviews. However, it could be more difficult to identify accuracy for the middle-range ratings as the reviews may contain data points that aren't as polarizing.

Upon listening to the top 5 finishers, it could have helped me to try vectorization on text data which could have helped in enhancing how the model performed when understanding the reviews. Stacy mentioned the use of lightgbm, which could have sped up the time it took my model to process the data. Integrating TF-IDF vectors with features such as user and product metrics could provide a comprehensive feature set. Vectorizing the text allows the model to detect subtle differences in phrasing that could signal rating nuances. For instance, "not good" and "great" could carry vastly different sentiment weights in the vector space. Ryan mentioned LDA/LSA and combining TF-IDF with a dimensionality reduction technique like TruncatedSVD (LSA) could have been used to reduce the vector dimensions, retaining only the most informative aspects of the text data while reducing the risk of overfitting.

Citations

- https://scikit-learn.org/dev/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
- https://xgboost.readthedocs.io/en/stable/
- CS506 Discord
- CS506 Piazza