CS506 Midterm Writeup
Ethan Machleder

WORKFLOW/CHOICES

When we initially started the competition, we only had the raw data of Id, ProductId, UserId, HelpfulnessNumerator, HelpfulnessDenominator, Time, Summary, and Text and used the KNN model. We created a new feature, Helpfulness, which is simply HelpfulnessNumerator / HelpfulnessDenominator, and this initial model had an accuracy of around 0.4. My first instinct was to tweak the parameters of the KNN model using different neighbors, from the default of 5 to as high as 90.

I noticed that even though the accuracy score reached a high of 0.53, the confusion matrix only indicated that it was making effective guesses for the 5-star rating with a near-non-existent metric for the other ratings. So as a result my next step was to find other models that would give better confusion matrices out of the box. This is because I figured a model with an acceptable score and balanced confusion matrix could be improved by adding more targeted features.

The next model that I came across was RandomForest[1] which creates multiple trees given the Dataframe and decides on the outcome most agreed upon. This sounded ideal as consensus seemed like a good way to manage having multiple possible outcomes. The initial accuracy was 0.5 and the confusion matrix seemed promising so my next decision was to start expanding out the preprocessing and adding features. TextLength and SummaryLength were intended to allow the model to draw correlations between the length of the text and summary and their respective ratings, low amount of text being a low rating and high amounts of text being a high rating. Alongside this, I removed Time as a feature as I didn't see any possible correlation between raw Time and ratings. This is when I used GridSearchCV[2] to find an ideal set of parameters, the range of parameters was generated by ChatGPT, to get the best output from RandomForest. In the end, I had an "ideal" RandomForest model that had an accuracy of around 0.55.

Not satisfied also found several other models and tested them out, including NaiveBayes, Logistic Regression, and GradientBoosting[4]. While NaiveBayes seemed promising, GradientBoosting provided the best "out-of-box" returns. I also added StandardScaler[3] to balance out the new values of TextLength and SummaryLength to make sure all features would be weighted equally.

At this point, I had turned my attention toward the features, which would provide the model with a better footing to make its predictions.

FEATURE ADDITIONS/FINAL CHOICES

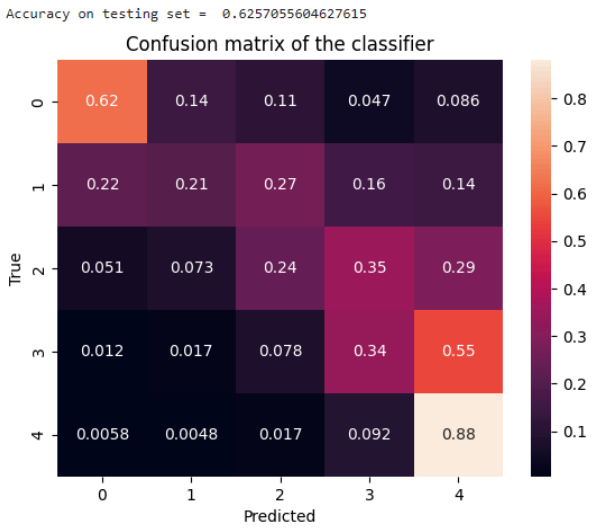
As mentioned before I had discarded the Time column entirely as I didn't see any use for it and created TextLength and SummaryLength. For further text/sentiment analysis, I tried using TF-IDF and CountVectorizer but I didn't have a full understanding of their functionality and saw that they would increase the pre-processing time and execution time of the model significantly.

Instead, I created TextSubjectivity and TextPolarity utilizing TextBlobs[5] sentiment analysis on the Text fields. Further adding to this I added nltk[6] Opinion Lexicon to count "positive" and "negative" words to create a PositiveWordCount and NegativeWordCount as well as a PosNegRatio for further sentiment analysis data points. I also got the idea to add ExclamationCount as another data point to represent emotion/emphasis in Text.

Another avenue that was suggested by ChatGPT was to create correlations between the columns UserID, ProductID, and Score. Following this I created UserAvgRating to link a User and the Ratings they gave and ProductAvgRating for the ratings given to a specific product. On top of that to create a "trust-worthiness"/similarity connection I created UserReviewCount and ProductReviewCount to add more "weight" to the AvgRating features. Lastly in this category, I added UserDeviation created from a UsersAvgRating minus a global rating average, and ProductDeviation which followed the same principle. These 3 sets of features were meant to connect all aspects of a User and Product with their Ratings. I feel like this alongside the sentiment analysis features was the most instrumental in my model's accuracy.

FINAL SCORE/POTENTIAL IMPROVEMENTS

In the end, I utilized an un-optimized GradientBoosting model with 18 features for a final score of 0.62, although there were issues with NaN values so the final score came out to 0.6.



The accuracy on both ends was acceptable with a decent range between the 2,3 and 4-star mark. The issues primarily came in determining between the correct range and the one below it, so for 4 the model had a hard time predicting between a 4 and 3.

If I had to do this project again I would take several steps. Firstly, I would focus on building out relevant and useful features such as sentiment analysis and connecting UserID, ProductID, and Ratings. I would make sure all this preprocessing data would be done first as it takes a significant amount of time to initially generate and is the most important factor in the accuracy of the model. As I did this as my last step I feel like I wasted a large amount of time juggling between models before I had any decent data to give it.

Second, I learned to understand the impact and importance of features and preprocessing. I added that even though I tried using TF-IDF and CountVectorizer they created significant issues in readability, bloat, and interpretation by the models. In the end, they were replaced by TextBlobs sentiment analysis and ntlk's Opinion Lexicon which were significantly easier to interpret and more efficient with space and time.

Another side effect of adding impactful features was not being able to fine-tune a model. Even though I used GridSearchCV on RandomForest, it was with a feature set that gave it very little to work with. If I had a complete feature set at the beginning I think I could have gotten better results with my models as well as explored the idea of using different models for different ratings.

All in all, I learned a large amount about individual models and the way they operate as well as the importance of preprocessing and properly interpreting data.

References:

1. RandomForest
https://en.wikipedia.org/wiki/Random_forest
https://www.datacamp.com/tutorial/random-forests-classifier-python
https://scikit-learn.org/1.5/modules/generated/sklearn.ensemble.RandomForestClassifier.html
2. GridSearchCV
https://scikit-learn.org/dev/modules/generated/sklearn.model_selection.GridSearchCV.html
3. StandardScaler
https://scikit-learn.org/dev/modules/generated/sklearn.preprocessing.StandardScaler.html
https://scikit-learn.org/1.5/modules/preprocessing.html
4. GradientBoosting
https://scikit-learn.org/dev/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
https://en.wikipedia.org/wiki/Gradient_boosting
5. TextBlobs
https://textblob.readthedocs.io/en/dev/
https://guides.library.upenn.edu/penntdm/python/textblob
6. NTLK
https://www.nltk.org/_modules/nltk/corpus/reader/opinion_lexicon.html

MISC:
https://arxiv.org/abs/1303.4402