Thomas Nguyen

CS 506 Midterm Report

Initial Observations/Assumptions:

1. One of the first things that became apparent is that the training data had the bias of containing about 53% 5 star reviews (fig. 1). I initially thought that this was a trick, and that the test set might follow a more uniform or normal distribution. I also did a bit of research and found the interesting fact that different review sites have very different distributions of ratings (fig. 2). However, after working with the data more and browsing Prime Video for a bit, I eventually concluded that the distribution of the training data did, more or less, reflect the general nature of the ratings.
2. My first impression of the features was that the only useful part would be the text column. I figured that I would have to convert the main text of the review into a sentiment score. I couldn't really imagine any other columns being useful at first. I did realize that a mean score based on userId and productId could be computed, but for some reason, this seemed like cheating to me.

Fig. 1

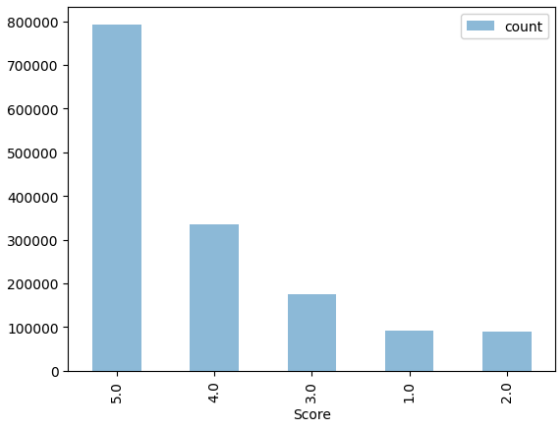
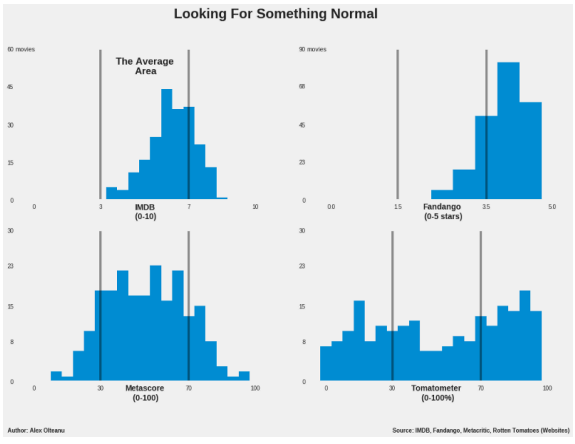


Fig. 2



Methodology:

- Classifier
    - At the beginning, I declared unwavering loyalty to KNN because it was in the starter code and I understood it well. However, after countless hours of consecutive disappointments, I relented and tested out a bunch of other classifiers (fig. 3). I ultimately found that GradientBoostingClassifier performed the best.
- Balancing
    - I initially went with balancing the training data when I was going with the theory that the review scores actually followed a more uniform distribution. This did result in better scores on the diagonal of the confusion matrix, but was overall detrimental to accuracy due to worse prediction of the 5-star majority class.
- Feature Extraction:
    - Sentiment - Separate scores for text and summary with analysis from both TextBlob and Vader
    - Subjectivity - Separate scores for text and summary with analysis from TextBlob
    - ReviewLengthWords - The word count of the review
    - MeanProductScore, MeanUserScore - I thought using these felt sleazy, but you gotta do what you gotta do
- Feature Selection
    - I wish I could have experimented more with selection, but I ended up just throwing every feature into the model besides time. As I expected, sentiment had a high correlation with score (fig. 4). However, MeanProductScore and MeanUserScore had even better correlation and substantially improved my model's performance.

Fig. 3

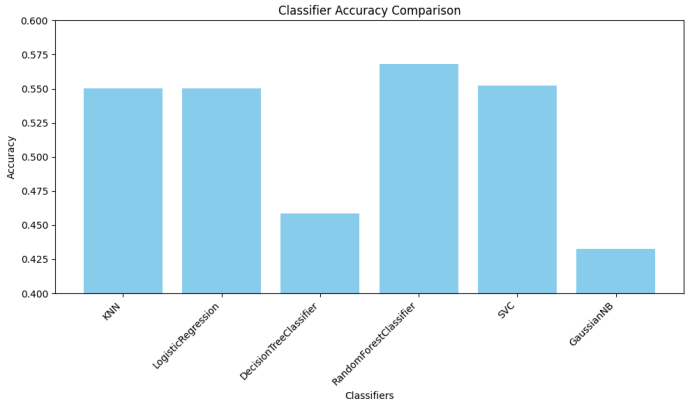
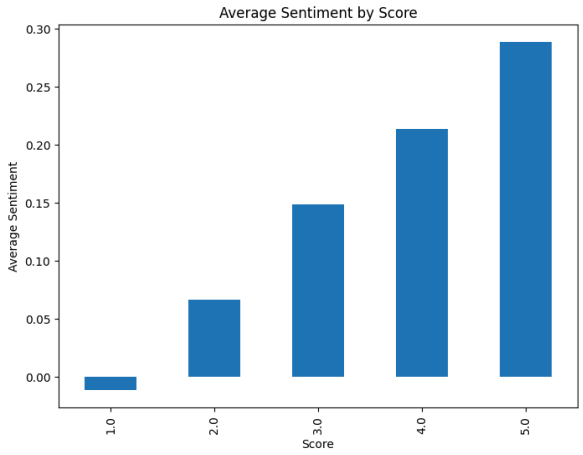
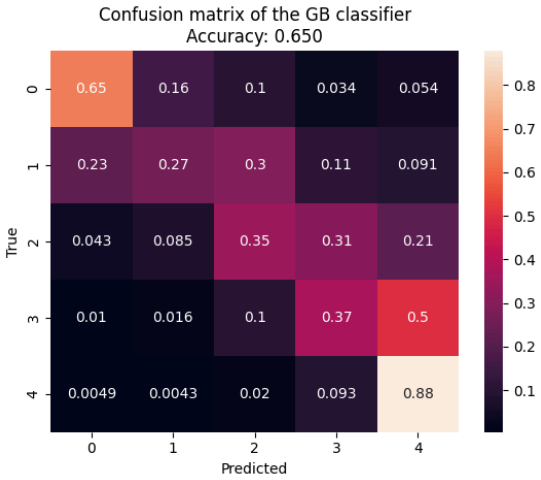


Fig. 4



Result:

My final kaggle private submission got 0.632 accuracy despite getting 0.650 on the local testing set, so I guess I overfitted a bit.



Takeaways/Reflection:

This made data science feel pretty chill on the practical side: most of my time was just spent waiting for model creation. But despite putting a lot of time into the project, I was left wanting more. So in the future, I want to look more into multi-threading and conducting broader experiments earlier in order to make more efficient use of time. I also feel like I should be more open-minded and try to be creative with feature extraction.