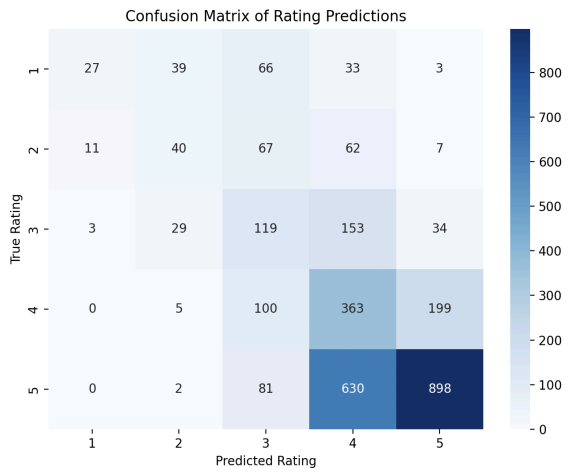
Methodology

I tried approaching the Amazon rating prediction problem by using ridge regression as I believe the linear model will be very beneficial for multidimensional datasets. I know that the train set has 9 features ranging from text to numerical data. That is the reason why I decided to stick with the ridge regression model and tried to make modifications around the model this past week.

The model starts with loading the train data and cleans all the text fields so it only has the information we care about. For the empty fields I just handled it by using an empty string. I noticed that the ratio between string and empty string is big enough for me to not handle those with additional care. I would definitely improve on how I choose to handle empty numerical and text fields. It is also worth mentioning that I declare the random seed to be 42 for repeatability as stated in class.

After the data is processed they will go through the pipeline I defined that includes standardized numerical values, TF-IDF vectorization for text and summary features. I capped the amount of features in text to 3000 for the text, and 1500 for the summary. I was hoping to include all the important information without trying to include everything which will take longer to compete. Lastly the pipeline will include the regressor that has an alpha value of 1, that remained unchanged for all the different variations I tried. By now, I have the model I can fit my processed data with. I will save the result of the prediction and export it as a csv.

Result Breakdown



The mean absolute error is 0.628 for the sample size of 0.01. From the confusion matrix, 5 star ratings and 4 star ratings have the best accuracy of 55.9% and 54.4%. My model had trouble predicating the 1 star and 2 star ratings. There were runs where the model mistakenly considered all ratings to be 4, which is not ideal. I think part of this is because the majority of the reviews are 4 - 5 stars, so there is a bias towards higher ratings. I think I should have processed the different ratings separately.

I have run the same model in three different sample sizes and the chart below is the result. This is the distribution of the reviews as I increase the batch size using the same model. The overall accuracy increased 3% from 0.01 batch size to 0.5 batch size.

	0.01 (1.31 minutes, 0.47 accuracy)	0.1 (4.33 minutes, 0.50 accuracy)	0.5 (121.87 minutes, 0.503 accuracy)
1	2686	2843	2946
2	8702	8639	8693
3	32393	29335	29001
4	91566	94194	94276
5	76845	77181	77276

One thing I noticed is that the ratings are very skewed to the higher side, as rating 4 and 5 have roughly 4 times as much as the rest of the ratings combined. However, even though the data have a similar trend, these predictions only have an accuracy of as high as 50%. I learned that feeding an imperfect model with more data might not be the most efficient use of my time because a better model might be able to achieve the same amount of accuracy with less sample data. I would choose to prioritize improving my model further instead of increasing my sample size multiple times hoping to see a big boost in my performance.

Challenges

Like mentioned above, one challenge I ran into is the amount of times it takes to train my mode. The time for training grows exponentially as the sample size increases. There are a lot of tradeoffs I have to make between spending the time to improve my model or waiting for a model to run and decide if I should take another approach.

Another challenge is obviously the accuracy. To do better, I tried using different models to come up with the ratings along with ridge regression and weight there results. One method I tried is using both a random forest model and the ridge model, each weighing 0.3 and 0.7, and coming up with a prediction. The result was also around 0.47 and there was no significant increase or decline, so I gave up that plan. I also tried to incorporate boosters in my pipeline but the improvement is minimal at the cost of longer runtime. I suspect I need to spend more time making the features more meaningful and try to put weights on different features because they do have various influences on the rating, and the influence is not the same. I also neglected the possibility of fully trying out different models instead of choosing to stick with ridge regression fully. I have built in cross validation tests in some of the test runs, and all of them appear to be higher than the Kaggle score, which means there is overfitting going on. These are the limitations I want to address if I were to continue working on this project.