Stephen Wong
Professor Galletti
CS 506
03/26/23

Midterm Analysis on Amazon Movie Reviews

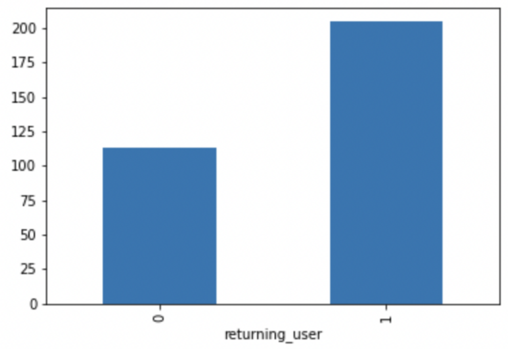
Introduction

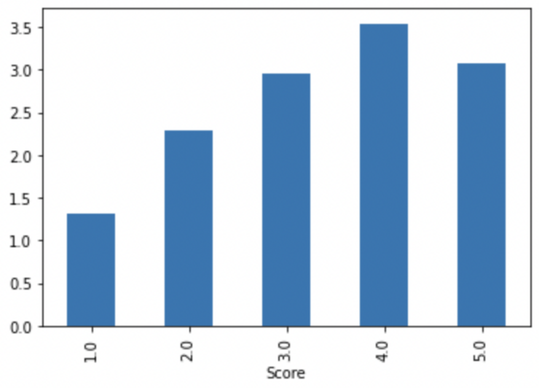
For this project, I was tasked with reviewing a data set with Amazon movie reviews and to make a prediction model for ratings based on the data available.

Key Features

Besides the features already available in the data set and in the starter code, I decided to get more values that would be good to make predictions for the data set. I first decided on going for a positive/bad word ratio which would be positive if there were “nicer” words being used in the Text section and negative if there were more “bad” words used in the text. Of course for this value, I also added columns in the data set with the amount of good and bad words per comment. The thought process was that reviews rated 4-5 stars would generally use more positive words when describing the movies. As I went through the data set, a very common type of 5 star rating review would be one where helpfulness was 0 and only positive words were included which was perfect for how I was doing. On the other hand, I started to notice that with the word extraction from the text, negations were a problem as in several cases, a bad review would include a lot of positive words with a negation stuck to them so the algorithm would think of them as positive. This led me to create another counter where negations would also be counted per comment. I also tried implementing TF-IDF but I’ll talk about that later in the report.

Besides features for Text, I also implemented two features as dummy variables “returning_user” and “movie_repeated” that consist of if a movie or user was repeated, 1 if there were more in the data set and 0 if there wasn't. For these, we check if there is a duplicate of them in the data set. From here I also added values ‘goodbad_movie’ and ‘goodbad_user”, that would check if the user/movie had other reviews and if those were usually good or bad. After I got the new features, I explored how they could be related to other features from the data. Below I have some examples.





The left graph explains how returning users are more likely to do lengthier comments in Text and the right graph shows the average ratio of good words in each score.

Basically I've tried to clean the variables as much as I can so that the .predict function has as much information it can to do the predictions.

Methods Used

For this kind of process, I based most of my model on decision trees and bayes algorithm as I thought with the data available, it would be the best way to get a prediction made. It had to do mostly with some of the key features I extracted. The main questions I made from a bayes stand point were: Given there are more "good words" in df['Text'], what is the probability it has a good rating(4,5)? Given there are more "bad words" in df['Text'], what is the probability it has a bad rating(1,2)?These would obviously vary a lot as the data set does have more of the higher ratings but I found examples where the Text was obviously negative, yet there were more "good" words in it. After these, I've been trying to implement TF-IDF as not all words should weigh the same as some should carry more weight than others. My computer wasn't able to handle getting an TF-IDF score for the words, but it would've been done using the TfidVectorizer, parsing all words from text and getting a score for each which will be useful in detecting which words carry more weight than others because of their exclusive use on a certain type of comments.

For other predictions such as in goodbad_movie and in good_baduser, I based the values for these using the decision tree method from class but in a non-recursive way. Here I tried checking first if they were repeated within the train and what type of rating the user/movie used to have in the train data set. This was also done just using the training part of the data set to not "cheat" on the data set.

The rest of the extra methods for extracting features were based on using pandas and the info already in the csv to make dummy variables(set to either 1 or 0) for a type of classification in the data. Methods used for the model are talked about in Validating Model.

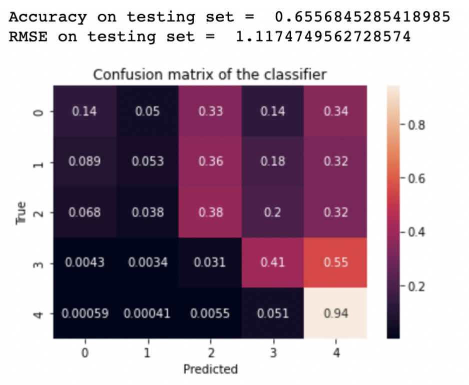
Tuning Model Parameters

From the model parameters, I dropped all the parameters that I felt didn't have anything to do with what goes on the score prediction. These were helpfulness denominator, helpfulness numerator, time from the features that came in the starter code. I also dropped all features that were only present to help implement others. I tuned the Text by performing the TF-IDF to set a weight on the words.

Validating Model

To validate the model, I based it on the ones that came with the sample code. The .predict function was in charge of taking all this new features that were appended to the test and trying to predict what the score for the test would be like considering all the differences in the data.

After trying and results not getting better, I opted for using linear regression, using the sklearn library. This method proved to work much better and would've worked even better if the TF-IDF would've run.
For the validation of the model, I started by getting different results between the Kaggle submission and the mean_sqaured error from the dummy code as one increased when the other got worse. I noticed this was usually happening when I used features that included using the score. This is the confusion matrix for the most accurate try, but not in Kaggle.



```
Accuracy on testing set =  0.6556845285418985
RMSE on testing set =  1.1174749562728574
```

Challenges and Lessons

There were a lot of difficulties making the data set be able to predict better. I found the .predict function a little difficult to use with the new features I added, as 0s would for some reason be perceived as Nans in the code. Also several reviews I noticed could be found to be very difficult to predict. Some positive reviews didn't have any praise for the movie, but provided more of a summary of it, which was hard to predict as they didn't have "good words" , weren't even marked as helpful and without the TF-IDF were very difficult to predict. The best that was done for these was the assumption that not many people are going to write a large review over a movie they were not very positively or negatively passionate about. The main difficulty I found, which I have been expressing in all the paper, was the TF-IDF implementation not working, as I believe it would've been the key feature that would've taken the model to the next level. I tried doing a manual version of it with good and bad words but without the specific weights or a type of normalization, it proved to be very difficult.

Conclusion

Even though my model didn't work as well as others, I believe I had a good understanding of the concepts but encountered a lot of difficulties when trying to enable them. With tools like TF-IDF added to my work, I'm sure the model would work very well.