Amazon Movie Review Score Prediction Model

Analysis by Ilay Guler
(ilay2004@bu.edu)

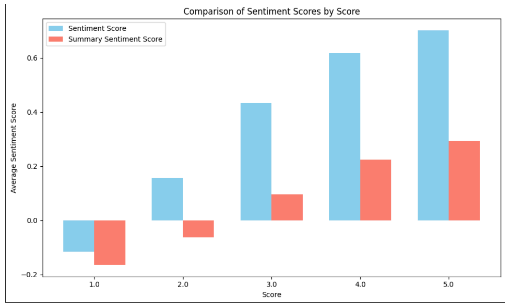
1. Given Data

The dataset given in the Kaggle competition contains 1,697,533 rows, each containing 9 columns corresponding to the given values associated with these reviews; Id, Product Id, User Id, Helpfulness Numerator, Helpfulness Denominator, Time, Summary, Text, and Score.
Given these attributes, we can extract features from them to train a model to better predict the score of a review given.
It is also worth noting that the dataset is incredibly skewed – there are about 800,000 5-star reviews, 330,000 4-star reviews, 176,000 3-star reviews, 91,000 1-star reviews, and 89,500 2-star reviews.

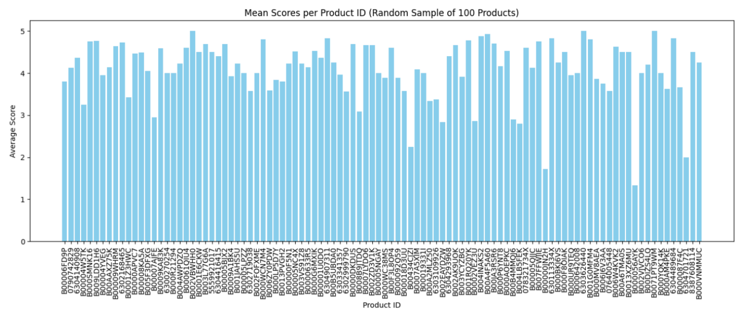
2. Feature Selection

A model is not useful if its features are not telling for prediction. To make accurate predictions, time was taken to select features that would best represent the review scores. I found the most success in examining very specific features.
Below is my thought process and data visualization while finding features.

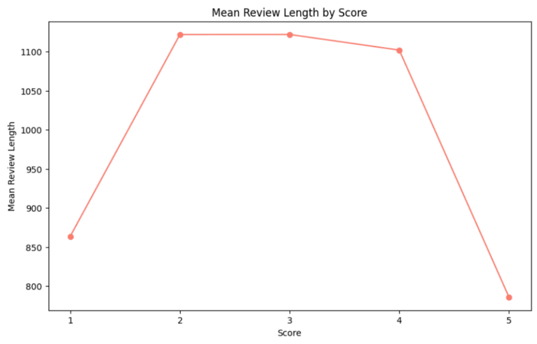
- **Sentiment Score by Text & by Summary**: Using nltk metrics' Vader lexicon sentiment analyzer I was able to calculate the sentiment score for each review's text. Graphing this showed a clear increase in positive sentiment for higher-rated reviews. The relationship was not as strong in summary sentiment score as it was in the text review sentiment scores. This could be because the summaries were far shorter in length – containing fewer words to sway the sentimentscore one way or the other.



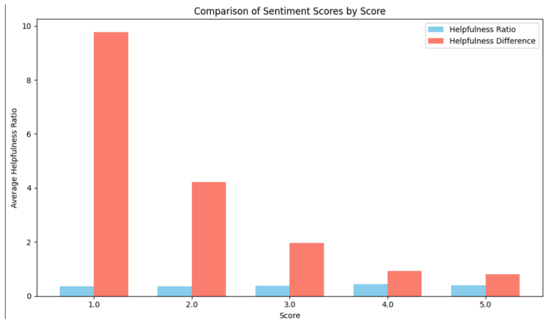
- **Product Average Score**: Products may be of different quality than others – lending themselves to different scores. For each specific product in the training data, I extracted its average score across all reviews for it. The graph below shows how out of 100 products, the average score for each product varies – suggesting model accuracy can improve if we hone in on specific product review trends.

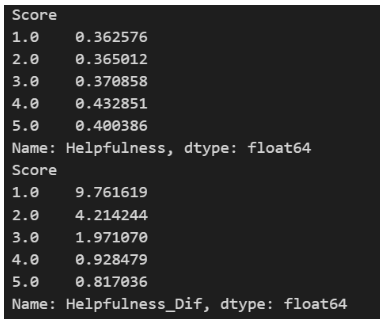


- **Review Length**: Although the bar chart for review length resembled a normal distribution curve, I reasoned that a feature such as this would be an important check for our model, since the average length of 1 or 5 star reviews varies wildly from each other and from the length 2 through 4 star reviews.



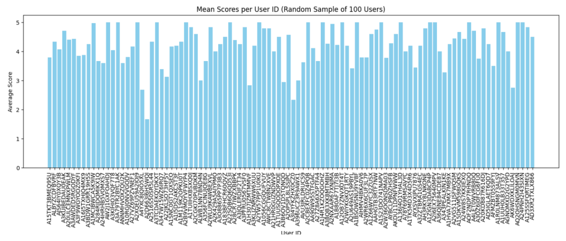
- **Helpfulness Features**
  - **Helpfulness:** The ratio found by helpfulness numerator/helpfulness denominator
  - **Helpfulness Difference:** The ration found by helpfulness denominator-helpfulness numerator (A different kind of ratio measuring how unhelpful a review is. The higher the number, the less helpful a review was). The graph shows that lower rated reviews were on average less helpful.
  - **Helpfulness Numerator & Denominator**: Used as features because the pattern in Helpfulness is very small. Could be better for more nuanced reviews





```
Score
1.0    0.362576
2.0    0.365012
3.0    0.370858
4.0    0.432851
5.0    0.400386
Name: Helpfulness, dtype: float64
Score
1.0    9.761619
2.0    4.214244
3.0    1.971070
4.0    0.928479
5.0    0.817036
Name: Helpfulness_Dif, dtype: float64
```

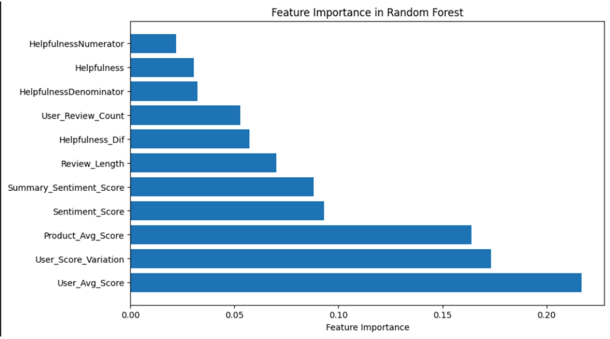
- **User Features:** After searching through the train.csv file it became clear that certain user Id's were repeated, showing that certain users have left multiple reviews. The mean scores they assigned to several movies, along with the standard deviation between these scores, and the number of reviews written was accounted for. The graph above shows how out of 100 users, the average score out of all the reviews each user wrote varies. My



intuition was that by identifying certain users in a review and understanding their behavior, we can better predict the score given.

Feature Conclusion:

User features as well as average product score carry the most importance in our model – suggesting that user review habits and the overall quality of a product lend itself to its review. Sentiment scores also played a strong role in predicting scores, as the heavy use of positive or negative language used can lead the model to a better conclusion. Analyzing the minute details of a review's data in combination with the overarching trends across all reviews led to a more accurate model.



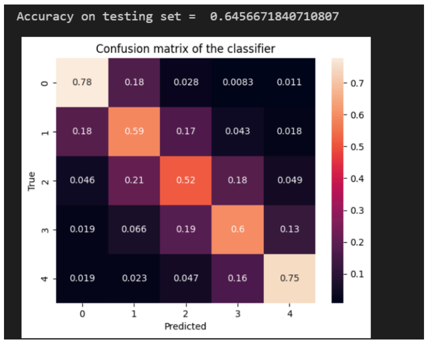
3. Model Selection

Given our data is skewed, we need to select a model which reduces bias toward the skewed data. Given random forests' resistance to skewed data and overfitting, I began to explore how it may benefit me as my selected classification model.
The more decision trees used with different criteria, the better random forest will perform, increasing its prediction accuracy. This is because random forest combines the predictions of multiple trees (which also reduces overfitting). This combination of positive attributes alongside my familiarity with this algorithm from class led me to settle for this model. I found that a good data sample, as well as a good selection of features made its accuracy comparable to that of more complicated models, while still maintaining a lower run time than others.

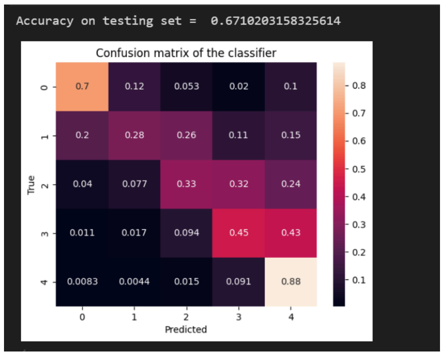
4. After Thoughts

In order to improve the prediction accuracy of our model for each score, I sought to train my data with an equal amount of 1 through 5 star reviews. I achieved this by undersampling everything. For each class, I randomly selected the same number of rows as the minority class to be part of the training data. This approach produced far more accurate results overall.
Unfortunately due to an 'nan' error during submission file generation I could not produce a submission file when I undersampled the reviews to all contain an equal amount of the smallest class. Although I made sure to handle nan entries in my feature extraction, this issue persisted unless I used the whole dataset. Below is a comparison of the two confusion matrices:

- The matrix on the left was produced using the same features and model but with even data.
- The matrix on the right was produced using the same features and model, but with the full skewed data.



```
Accuracy on testing set =  0.6456671840710807
```



```
Accuracy on testing set =  0.6710203158325614
```

Citations

"Python: Sentiment Analysis Using Vader." *GeeksforGeeks*, GeeksforGeeks, 7 Oct. 2021, www.geeksforgeeks.org/python-sentiment-analysis-using-vader/.

"Randomforestclassifier." *Scikit*, scikit-learn.org/1.5/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

Singh, Harshdeep. "Understanding Random Forests." *Medium*, Medium, 24 Mar. 2019, medium.com/@harshdeepsingh_35448/understanding-random-forests-aa0ccecdbbbb.