CS506 Kaggle Competition Project: Joshua Leeds

Introduction:

My plan for this project was to design new features that would be beneficial in a machine-learning model. To identify possible features, I spend time in exploratory data analysis on the provided features from the dataset.

Features:

The first feature that I wanted to add was a feature that had the average score for each user and each product. The point of this feature was to use each product's score and each user's score to predict new scores. After working with the creation of the average score for each user and each product, I found that it could be more beneficial to create a lower and upper bound using a 95% confidence interval as features, instead of just simply using the mean. I did this because the confidence interval takes into account standard deviation and the number of reviews left for each user and product. I found that using the lower and upper bounds from a confidence interval was more effective for my machine-learning model.

In addition, as part of these features, I also performed a Kruskal Wallis test on the mean for each user and the mean for each product. [I learned about the Kruskal Wallis test in the statistics class I am taking this semester]. The purpose of this test was to find if there was a significant difference in the means of each product and each user between every other product and user, respectively. The tests yielded the following results.

```
Kruskal-Wallis H-test result: KruskalResult(statistic=482828.0207144043, pvalue=0.0)
Kruskal-Wallis H-test result: KruskalResult(statistic=315239.4720873702, pvalue=0.0)
```

These results show a p-value of 0.0 for both user and product means, meaning that there is a significant difference between each user and each mean. Due to the p-values being 0, I decided that it would be beneficial to use the mean in the representation of lower and upper confidence interval bounds.

Next, I worked on sentiment analysis features. I used SentimentIntensityAnalyzer from the nltk.sentiment library. I created functions that would get the sentiment score for the text and summary and then took the compound_score for each. After doing this, while both of these features were beneficial for my machine-learning model, that summary took significantly less time and also added much more benefit to my accuracy. Because of this I also decided to add both a positive and negative sentiment score for summary. Next, I did data discovery on the sentiment scores to see if I could draw any conclusions.

Text

	SentimentCategory	Percentage
0	Strong Positive	70.951326
1	Strong Negative	10.612669
2	Moderate Positive	9.207399
3	Moderate Negative	4.644195
4	Slightly Negative	2.623645
5	Neutral to Slightly Positive	1.960766

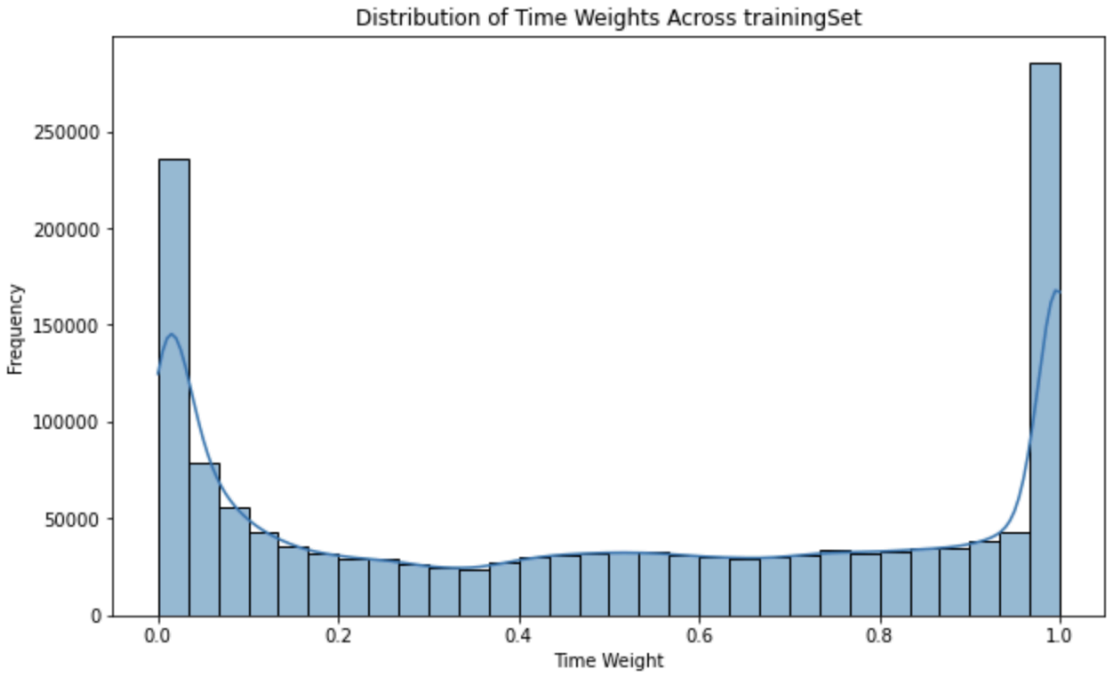
Summary

	SentimentCategory	Percentage
0	Slightly Negative	37.983264
1	Moderate Positive	26.187926
2	Strong Positive	21.388691
3	Moderate Negative	9.765232
4	Strong Negative	2.512285
5	Neutral to Slightly Positive	2.162601

The following above charts show the breakdown of the sentiment scores for summary and text respectively. It was surprising that these differed so much, with text sentiment having strong positive as the highest at 70% and summary having slightly negative being the highest at almost 38%. It was particularly interesting to see that the text sentiments were strongly positive most of the time, however, this also relates to the fact that most of the scores in the data set were also 5's, meaning that most of these 5's may have a strong positive summary sentiment rating.

Throughout the data discovery process for sentiment scores, I also looked into the breakdown of each sentiment category for each specific movie score, to see if any specific movie score had a higher breakdown of any sentiment category. For this, I looked specifically at the summary scores since they had more variance compared to the test scores. However, I found that the highest category for each score was still slightly negative, meaning that there was not much of a significant difference between the sentiment categories and each movie rating.

Finally, I added three features related to time. I decided to do this after I looked at the abstract of the research paper connected to the Kaggle, which said that it could be effective to look at how time is relevant when looking at movie ratings. The three features that I added were experience_level - which is the amount of times a UserId has appeared so far -, days_since_last - which is the amount of time between reviews for a specific UserId, and time_weight -which uses a decay function to calculate the relevance of a review by giving priority to newer reviews. Each of these features slightly helped my results, however, the decay function helped the most and was the feature I played with the most. For the decay function at first, I had a decay value that was too high, which made almost all of my time_weight to be 0. I changed the value over time and found that 0.00175 was the value that resulted in the most variance.



This graph shows the time_weight distribution and shows that 0 and 1 are the most common values. This was beneficial as it could put more emphasis on the most extremes, which can be helpful in distinguishing differences between each ID.

Methods:

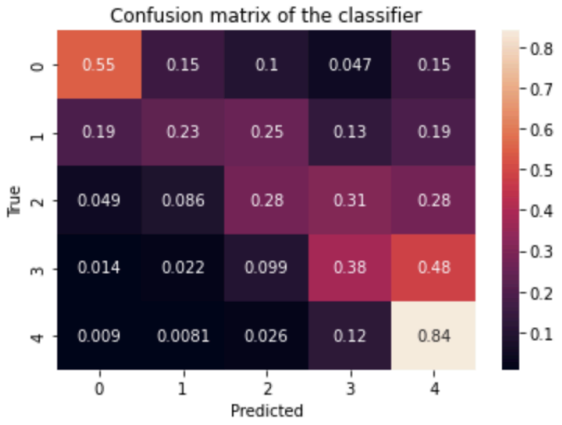
Throughout the creation of features, I wanted to ensure to avoid leakage and nan values if possible. For the average values for userId and productId to make the confidence interval bounds, I needed to make sure that I avoided “cheating” my calculated values after a train test split. Because of this some of the confidence interval bounds in the testing data from the train test split were null. To fix this I used a SimpleImputer from sklearn to take the mean value of the column to calculate the upper and lower bound confidence intervals for the nan values. I found this effective in reducing null values in the data set.

Feature Selection and Machine Learning Model:

Next, I worked on feature selection. To select features I made a correlation matrix to see if any of the features were related. I found that some of the features did have a direct correlation, such as some of the time-related features, however, I found that removing them hurt my accuracy, so I decided to keep all numeric features for my final model.

Finally, for my machine learning model, I worked with both an XGBClassifier and a RandomForestClassifier. I found that the XGBClassifier gave me a higher accuracy and took a shorter amount of time to run, so I decided to use the XG boost Classifier throughout my testing process.

```
Accuracy on testing set =  0.6153295128939829
```



Above is the confusion matrix for the XGBClassier for my highest score. In this model, I used all of the numeric features, which included the new features that I created and then helpfulness and time. The newly created features that had the highest impacts throughout the process were the confidence intervals, and the summary sentiment scores, with each of them

adding around 5% when I initially implemented them. The other added features, such as the time-related features only added around 1% accuracy improvement. Throughout the testing process, I did a new XGBClassier model each time I made a new change to the training set to try and understand each change's individual impact. Overall, I felt that this method was effective in reaching a good model.