

Write-Up

I don't have extensive knowledge about NLP, so I essentially approached this competition as a general classification challenge.

- Name: Junhui Huang
- Kaggle and BU Username: hjh604

Data Analysis

I explore the dataset by using `df.dtypes` to see the data type of each column, using `df.describe` to get the information about the distribution of the data, and then using `df.head` to get a few examples of the samples so I could have an initial thought about how to make each column into usable features.

```
Id    ProductId    UserId    HelpfulnessNumerator \
0    914403    B0009W5KHM    AV6QDP8Q0ONK4    2
1    354887    6303079709    A2I8RXJN80A2D2    0
2    1407653    B004H0M2XC    A3FHV3RV8Z12E6    0
3    1377458    B003ZJ9536    A12VLTA3ZHVPUY    1
4    475323    630574453X    A13NM1PES90XVN    2

    HelpfulnessDenominator    Time \
0    2    1341014400
1    0    1168819200
2    0    1386201600
3    1    1348704000
4    3    970012800

                                Summary \
0                                GOOD FUN FILM
1                                Movie Review
2    When is it a good time to Consent?
3                                TRUTH
4    Intelligent and bittersweet -- stays with you

                                Text    Score
0    while most straight to DVD films are not worth...    5.0
1    I have wanted this one for sometime, also. I ...    5.0
2    Actually this was a pretty darn good indie fil...    4.0
3    Episodes 37 to 72 of the series press on in a ...    5.0
4    I was really impressed with this movie, but wa...    3.0

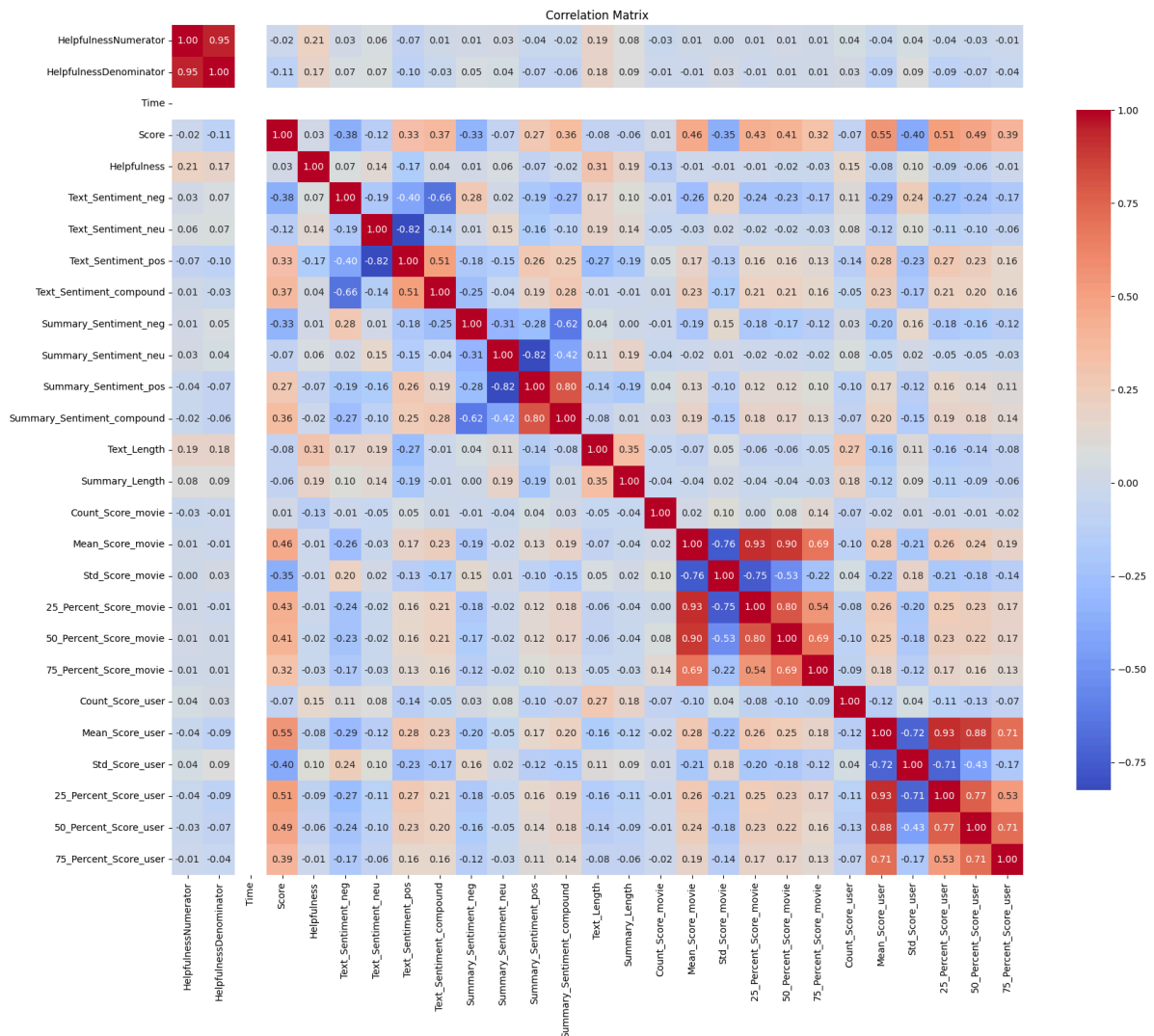
    Id    Score
0    1323432    NaN
1    1137299    NaN
2    1459366    NaN
3    931601    NaN
4    1311995    NaN
```

```
Id    HelpfulnessNumerator    HelpfulnessDenominator \
```

count	1.697533e+06	1.697533e+06	1.697533e+06
mean	8.487660e+05	3.569048e+00	5.301422e+00
std	4.900357e+05	1.727883e+01	2.024445e+01
min	0.000000e+00	0.000000e+00	0.000000e+00
25%	4.243830e+05	0.000000e+00	0.000000e+00
50%	8.487660e+05	1.000000e+00	1.000000e+00
75%	1.273149e+06	3.000000e+00	5.000000e+00
max	1.697532e+06	6.084000e+03	6.510000e+03

	Time	Score
count	1.697533e+06	1.485341e+06
mean	1.262422e+09	4.110517e+00
std	1.289277e+08	1.197651e+00
min	8.793792e+08	1.000000e+00
25%	1.164413e+09	4.000000e+00
50%	1.307491e+09	5.000000e+00
75%	1.373242e+09	5.000000e+00
max	1.406074e+09	5.000000e+00

I also draw a correlation matrix after feature processing to see if any of the features are strongly related.



Feature Engineering

Helpfulness

The first feature I added was "Helpfulness" since it's easy to implement and free!

User Preference and Movie Averages

One of the most important tasks I undertook was grouping the ratings by `userID` and `movieID`. This allowed me to derive several features:

- **Movie Features:**
 - **Average Rating:** This reflects the general quality of the movie.
 - **Count of Ratings:** This indicates the popularity of the movie.
 - **Standard Deviation (STD):** This measures the variability in ratings, helping the model to discern whether to weight the semantic analysis of individual reviews more heavily or to rely on the average rating.
- **User Features:**
 - **Average Rating:** This reflects the user's personal rating tendencies.
 - **Count of Ratings:** This provides insight into the user's experience and qualifications.
 - **Standard Deviation (STD):** This indicates the range of ratings a user typically gives. For instance, some users, like myself, tend to score movies between 2 and 4, categorizing films as either masterpieces or total failures.

Label Encoding

I label-encoded `ProductId` and `UserId` since they were originally strings. I hoped this would aid in classification, although ultimately, it didn't significantly impact the results.

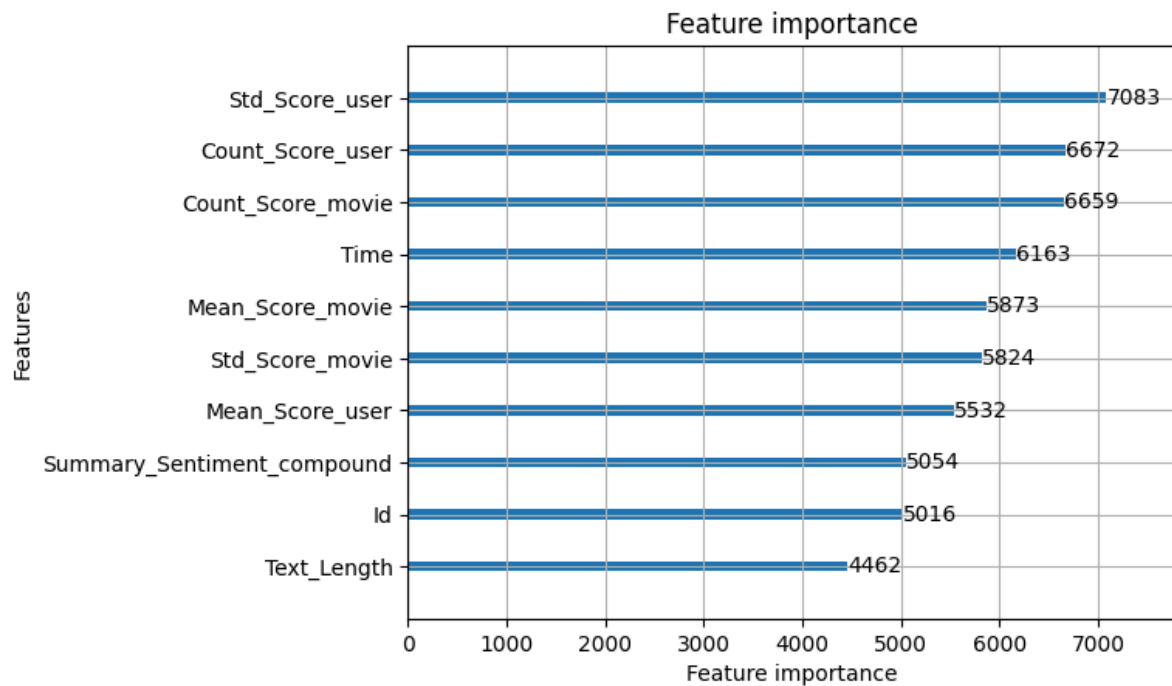
Time as a Feature

By analyzing feature importance plots and correlation matrices, I found that `Time` emerged as an important feature. My current hypothesis is that it may reflect the economic conditions when the movie was released, which could correlate with the quality of the movies.

NLP Features

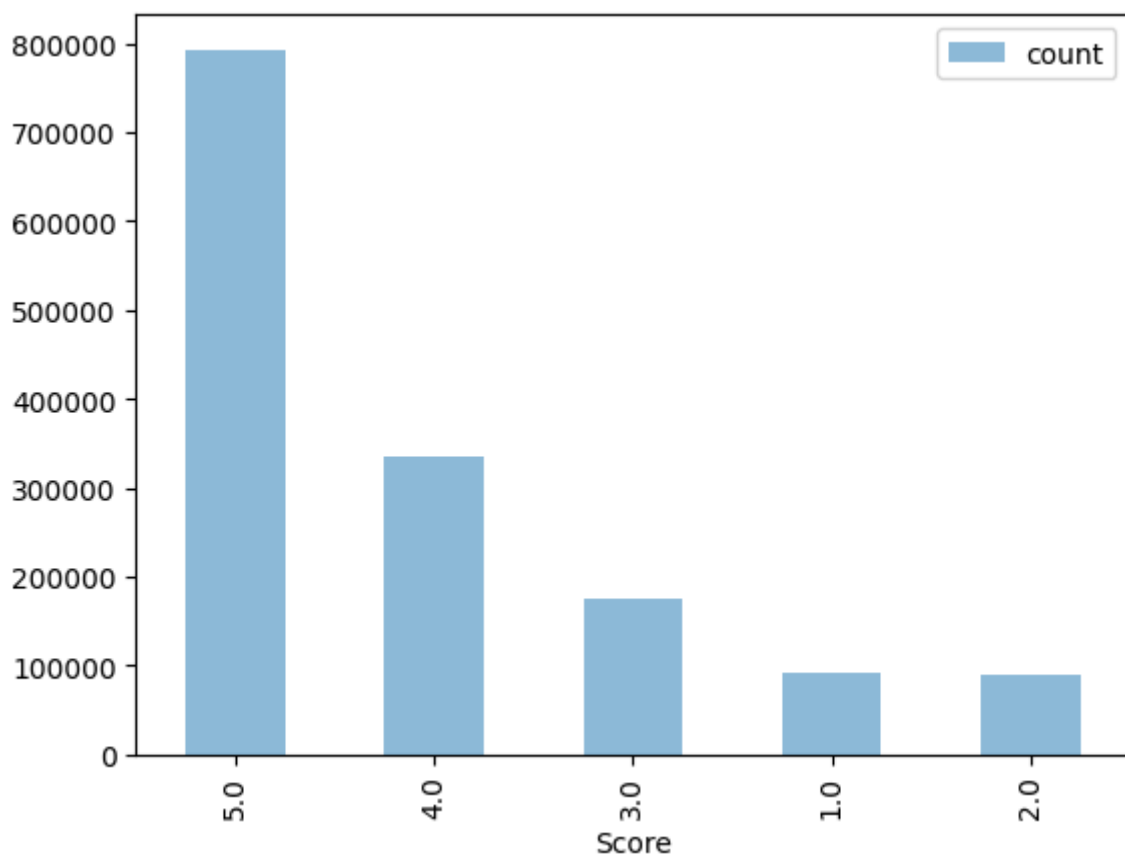
The only NLP features I included were the sentiment scores of the `Summary` and `Text` calculated by Vader Sentiment, as well as their lengths. I believe that longer reviews are often more thoughtful and serious.

Here is a plot showing the importance of each feature in my XGBoost model.



Addressing Class Imbalance

Oversample



Upon observing the distribution plot, I discovered that the dataset was extremely imbalanced. To address this, I employed SMOTE to oversample the features mentioned earlier in the 1 to 4 star comments, aligning their numbers with those of the 5-star ratings. Fortunately, this strategy improved model performance.

Training the model

Models Used

I experimented with XGBoost, LightGBM, and Extra Trees, applying aggressive hyperparameter tuning throughout the competition. The best results came from XGBoost. While I generally prefer LightGBM due to its advantages, it is also prone to overfitting. Given the time constraints, I focused more on feature engineering rather than model selection.

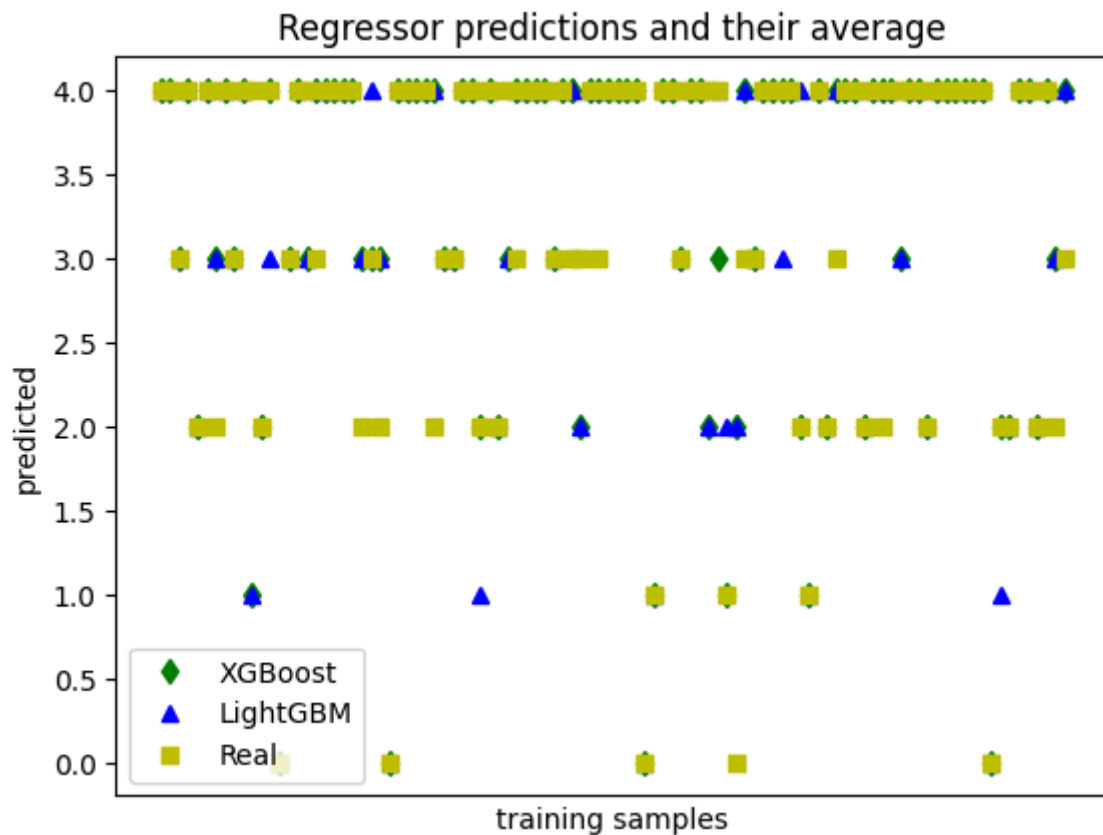
The benefit of XGBoost lies in its layer-by-layer development, which can help mitigate overfitting compared to other methods, and that's important because I had already overfitting my model with only 10% of the dataset or many time.

Classification vs. Regression

I typically prefer regression techniques, clamping scores into integers since discrete ratings often maintain a meaningful relationship. However, for this competition, I opted for classification. My previous models tended to categorize 4-star ratings as 5 stars, which misrepresented their distinctiveness. By treating them as separate categories, the model learned that predicting an actual 5-star rating as 4 stars wouldn't yield any credit, as there's no concept of partial correctness in this context.

Attempt to Combine Multiple Models

I attempted to use another vote classifier (I don't want to train 3 heavy models in a row), but it resulted in the worst performance. The combined predictions were even less accurate than those made by the individual models, LGBM and XGBoost. I had hoped that these models would balance out each other's biases, but it seems they ended up making the same mistakes together!



VaderSentiment: Vader Sentiment (Valence Aware Dictionary and sEntiment Reasoner) is an NLP library for carrying out sentiment analysis.

VotingClassifier: The Voting Classifier combines predictions from multiple models to improve accuracy by mitigating individual model biases.

GridSearchCV: GridSearchCV uses a brute force approach to search all possible parameter combinations in order to optimize the performance of the model.