Introduction

Kaggle username: shivangidhiman

The goal of the competition was to predict the star rating associated with user reviews from Amazon Movie Reviews using the given features. To accomplish this goal, I used techniques like feature extraction/creation, model selection, tuning of hyperparameters, K-fold cross-validation, and model evaluation.

Exploratory Data Analysis

As the first step to performing EDA, I generated a correlation matrix to investigate the relationship between variables. This was done before any feature engineering or preprocessing to understand the relationship between raw data variables. From the above correlation matrix, it can be seen that 'Time', 'Helpfulness', and 'Score' has a positive correlation. This means that 'Time' and 'Helpfulness' increases with the 'Score'. Next, I plotted a Distribution plot for 'Score' to understand its distribution and to identify any patterns or trends in the data, which showed that the Data is skewed toward the higher rating (Majorly 5).

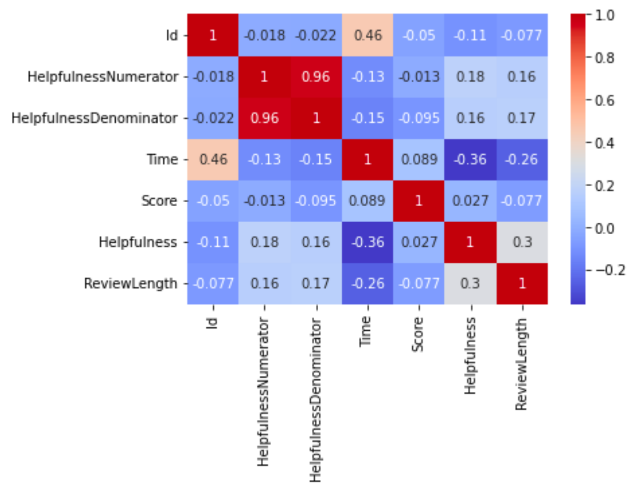

Figure1: Correlation Matrix for Raw Data

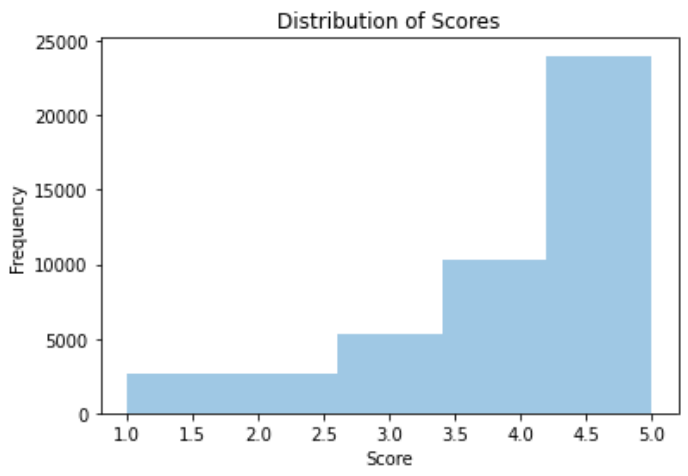

Figure2: Distribution plot for 'Score'

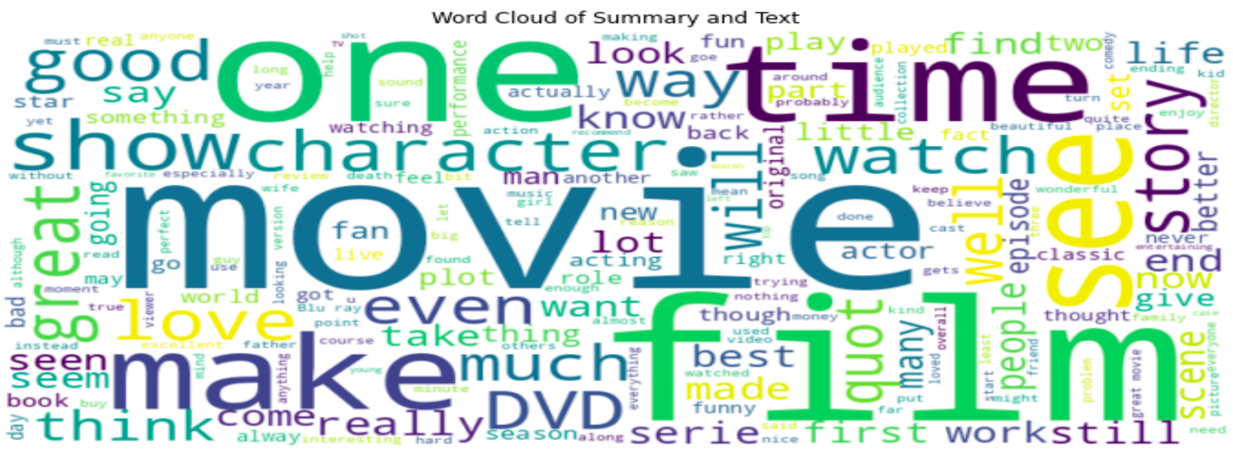

Figure3: Word Cloud of Summary and Text Columns

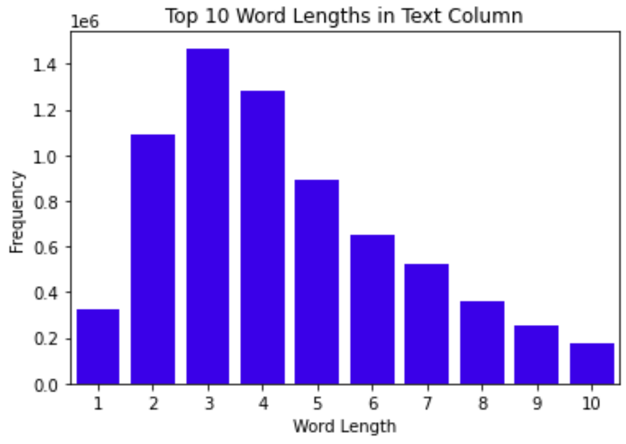

Figure4: Top 10 Words lengths in Text Column

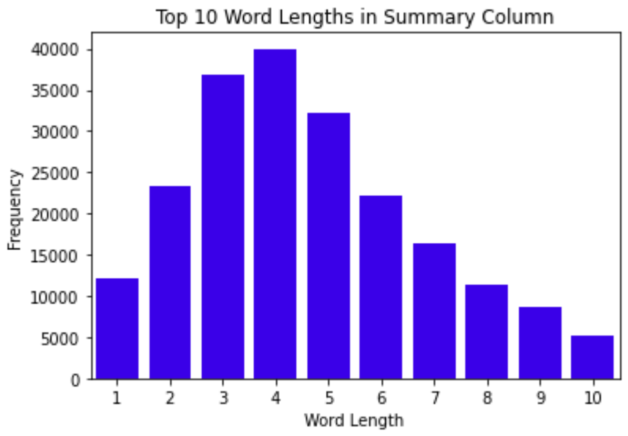

Figure5: Top 10 Word Lengths in Summary Column

I also wanted to understand which words are used the most in the reviews and therefore, I plotted a word cloud to visualize that. From this graph, we can see that words like 'movie', 'one', 'film' etc are used the most. I noticed that the top most used words were of 3 or 4 characters and therefore, I plotted the bar graphs of the top 10 word lengths for the 'Text' and 'Summary' columns to find any underlying trend and I can conclude that most of the words in either 'Summary' or 'Text' are of smaller length (<5).

Feature Engineering/Extraction

After doing the exploratory data analysis, I decided to create a few new features that I believed would be correlated with the 'Score'.

1. Sentiment Scores: As we have customer feedback data, therefore, my first choice was to go for Sentiment Analysis. I calculated sentiment scores for both 'Text' and 'Summary'. I used NLTK's SentimentIntensityAnalyzer (VADER) for this. Out of all the scores, I chose 'compound' score returned by the SentimentIntensityAnalyzer as it is a normalized and weighted score that takes into account the positive, negative, and neutral sentiment scores of the text.
2. TF-IDF vectors: As I had two text columns which is unstructured, and machine learning models require structured data as input. Creating TF-IDF vectors of the text data can help me transform unstructured text data into a structured numerical format. I used two TF-IDF vectorizers: a word analyzer and a character analyzer. I used the word analyzer one with ngram range (1, 3) on my 'Text' column as the this column contains words and phrases and I wanted to capture that in my TF-IDF vector. At the same time, the 'Summary' column just contains the important words from the 'Text'column therefore, I used character vectorizer with ngram range (2, 4) so that I capture the important words of length between (2, 4). The reason behind this is the insights I derived from my word length plot.
3. Helpfulness: From the correlation matrix, I could see a minor correlation between 'HelpfulnessNumerator', 'HelpfulnessDenominator' and the 'Score'. This led me to create a obvious metric by dividing these two and create 'Helpfulness'. And when I again created a correlation matrix, I found out that 'Helpfulness' has a significant correlation with 'Score'.
4. Encoded Product ID: I wanted to check the correlation between product IDs, user IDs and 'Score', that is why I encoded both product ID and user ID using the label encoder. In the new correlation matrix, I noticed that product IDs have a significant correlation with 'Score', whereas user IDs don't. That is why I just went ahead with encoded product IDs.

After creating all these features, I combined them into a feature matrix and then scaled them using MaxAbsScaler. I knew that some of my data contains negative values as well and I wanted to preserve the sign of the data that is why I went ahead with MaxAbsScaler rather than others. Also, as the data was skewed, I wanted to scale the feature matrix so that my model is not biased.

Model Selection

After creating the new features, I trained several models to predict the star rating. I began with a simple logistic regression model and gradually moved on to more complex models such as Xgboost and Lightgbm. I found that the Lightgbm model performed the best on the validation set and decided to use it for further tuning.

I tried multiple models one after the other in a pipeline on a sampled dataset, to see which one is giving me best RMSE score. After trying some more basic models like logistic regression. I moved to ensembled and boosting models like Gradient Boosting and Random forest. Out of these two Gradient Boosting did perform the best. But, the RMSE score was still not good enough. I tried stacking multiple models like RandomForest, SVM and Gradient Boosting to improve the RMSE of my model. But this started overfitting, So, I moved to more complex models like XgBoost and Lightgbm and out of these two Lightgbm gave me the best accuracy.

My reason for going with LightGBM other than RMSE was its high accuracy and speed. As it used tree-ensembled approach, it results in higher accuracy compared to other linear models. Also, LightGBM provides feature importance score, which helped me identigy the most important features. Also, I used regularization techniques which prevented overfitting and improved generalization.

Hyperparameter Tuning

To improve the performance of the my LightGBM model, I first tried performing hyperparameter tuning using grid search. It crashed my system multiple times. So, I moved to random search but that didn't work too because of CPU and memory issues. Therefore, I played arround with the hyperparameters of LightGBM to find the optimal one. I reduced the learning rate to 0.06 and changed the num_leaves to 30. After eighth or ninth try, I got the best RMSE.

Model Evaluation

Finally, I evaluated the performance of the model on the test set. The model achieved RMSE of 0.82 on the test set, which was a significant improvement over the baseline RMSE of 1.44..

Challenges and Findings

Throughout the process, I encountered several challenges. One of the main challenges was dealing with the class imbalance in the dataset. The majority of the ratings were 5 stars, which made it difficult to predict the lower ratings. To address this, I experimented with different sampling techniques such as oversampling and undersampling. Ultimately, I found that oversampling the lower ratings resulted in the best performance on the validation set.

Additionally, I had major system issues, as non of the grid searched worked. I tried a few stacking models but they also crashed my system. Also, it wasn't a challenge but finding a model which gave me better RMSE was time taking as I had to test almost ten models to find and research about lightGBM and then try it.