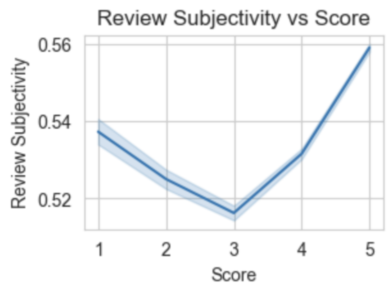
Doruk Savasan (U21084366)
CS506 Midterm

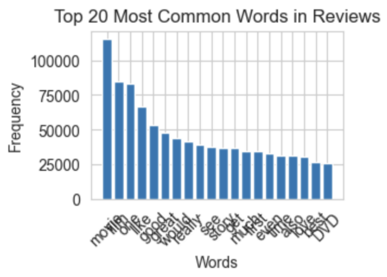
CS 506 MIDTERM REPORT

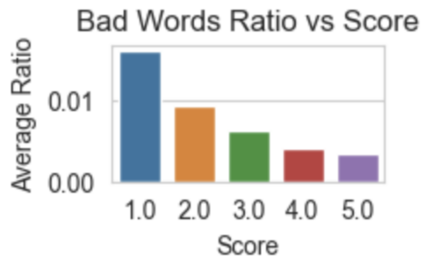
The goal of this kaggle competition is to implement a predictive model that estimates the star rating (score) of a movie based on user reviews by using the initial features:‘Id’, ‘UserId’, ‘Time’, ‘ProductId’, ‘HelpfulnessNumerator’, ‘HelpfulnessDenominator’, ‘Summary’, ‘Text, and ’’Score’.

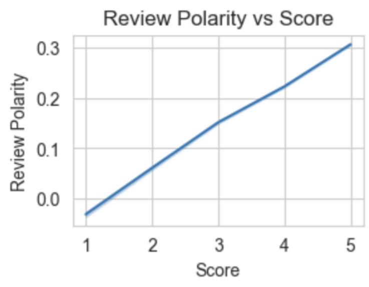
Preliminary Analysis:

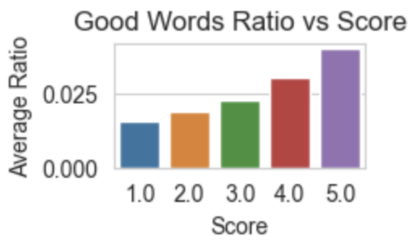
1. Analyzed numerical data features using the provided starter code in the notebook.
2. Checked for duplicate rows within the dataset to ensure data integrity
3. Plotted the 'HelpfulnessRatio', 'NotHelpful', and the distribution of both 'HelpfulnessNumerator' and 'HelpfulnessDenominator' to understand user engagement metrics.
4. Calculated and visualized the average length of reviews to measure the depth of feedback.
5. Generated visual representations of the most frequently used words in the dataset, comparing those commonly associated with the highest (5) and lowest (1) scores to discern potential correlations.
6. Investigated a variety of variables including 'UserAvgReviewLength', 'ProductReviewVolume', 'UserReviewCount', 'TimeSinceFirstReview', and 'RepeatPurchases' in search of high-correlation features; however, the correlations observed were notably weak.
7. Tried to identify any correlation between 'User Average Score' and 'Product Average Score' with the 'Score', aiming to understand the influence of user perception on product ratings.
8. Used the 'missingno' library to visualize the number of missing values, confirming the absence of NaN values in the dataset, with the exception of the 'Score' column.
9. Plotted 'Subjectivity' and observed a linear relationship with 'Score’.
10. Visualized that 'Subjectivity' decreases from scores of 1 to 3, reaching a global minimum at 3, before increasing again towards the score of 5.

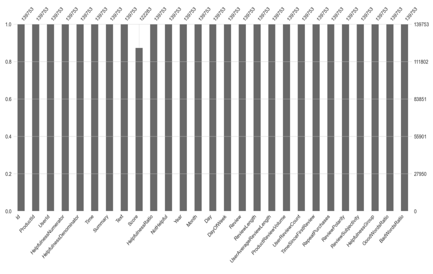








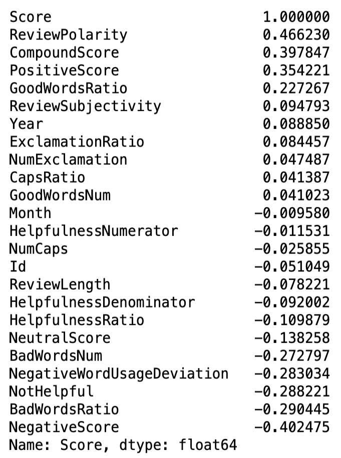




Preprocessing and Feature Extraction:

Feature Extraction:

- HelpfulnessRatio: Indicates the proportion of positive votes out of all votes.
- NotHelpful: Counts the number of unhelpful votes a review has received.
- Time: Extracts the year, month, day, and day of the week from UNIX timestamp, which can be used to analyze temporal trends in review scores.
- ReviewLength: Measures the number of words in a review.
- ReviewPolarity: Uses sentiment analysis to rate the positivity or negativity of the review.
- ReviewSubjectivity: Measures how subjective the review text is.
- NumExclamation: Counts the number of exclamation marks, which can indicate strong emotions.
- NumCaps: Counts the number of capitalized letters.
- CapsRatio: Provides the ratio of capitalized letters to the total length of the review, indicating emphasis or strong feelings.
- ExclamationRatio: Gives the ratio of exclamation marks to the total length of the review, another indicator of strong emotions.
- Good/BadWordsRatio: Measures the frequency of positive/negative words in a review, which can correlate with higher/lower scores.
- Good/BadWordsNum: Number of positive/negative words in each row.
- NegativeWordUsageDeviaton: The deviation in the frequency of negative word usage in each review from the average across all reviews.
- SentimentIntensityAnalyzer: Measures 4 different categories for the review column: positive, negative, compound, and neutral.



```
Score                          1.000000
ReviewPolarity                 0.466230
CompoundScore                  0.397847
PositiveScore                  0.354221
GoodWordsRatio                 0.227267
ReviewSubjectivity             0.094793
Year                           0.088850
ExclamationRatio               0.084457
NumExclamation                 0.047487
CapsRatio                      0.041387
GoodWordsNum                   0.041023
Month                         -0.009580
HelpfulnessNumerator          -0.011531
NumCaps                       -0.025855
Id                            -0.051049
ReviewLength                  -0.078221
HelpfulnessDenominator        -0.092002
HelpfulnessRatio              -0.109879
NeutralScore                  -0.138258
BadWordsNum                   -0.272797
NegativeWordUsageDeviation    -0.283034
NotHelpful                    -0.288221
BadWordsRatio                 -0.290445
NegativeScore                 -0.402475
Name: Score, dtype: float64
```

Scaling and Transformations:

- StandardScaler: Normalizes numerical features to have a mean of zero and a standard deviation of one, which is important for models that are sensitive to the scale of data.
- TfidfVectorizer: Transforms text into a sparse matrix of TF-IDF features, which reflect the importance of words in the context of the document and the entire dataset. Tfidf had the greatest impact on my accuracy.
- OneHotEncoder: Converts categorical variables ('ProductId' and 'UserId') into a binary matrix representation, which is necessary for models that require numerical input.

Combining Features for the Model:

- The final training and test datasets are assembled by horizontally stacking the numerical features, one-hot encoded categorical features, and TF-IDF features to create a feature space for model training.

Model Selection:

Model Selection:

1. In my model development's initial phase, I evaluated several machine learning models to establish a baseline for performance. In order for the model to run quickly I trained all the models solely on the TF-IDF data I extracted from the Review column I generated. The lineup included Logistic Regression, Ridge Classifier, Ridge Regression, Decision Trees, Random Forest, Gradient Boosting, Support Vector Machines, K-Nearest Neighbors, and Naive Bayes. I trained each model on the dataset and assessed their performance using

accuracy and RMSE as metrics. This comparison revealed that the Ridge regression model delivered the best results, particularly in terms of RMSE, which is a critical metric for regression tasks. Given the strong performance of the Ridge regression model, I decided to advance with it into the refinement stage.

2. To refine the model further, I constructed a pipeline that integrated the preprocessing steps such as scaling numerical features and applying TF-IDF vectorization to the review text with the Ridge regression model. This integration was key to simplifying the workflow and ensuring consistent preprocessing during model training and evaluation. Upon training and evaluating the Ridge regression model within this pipeline, it reaffirmed its performance by yielding the lowest RMSE again. This consistent result across different evaluation stages made it clear that Ridge regression was the optimal choice for my predictive model, leading me to concentrate on tuning it to achieve the best possible performance.

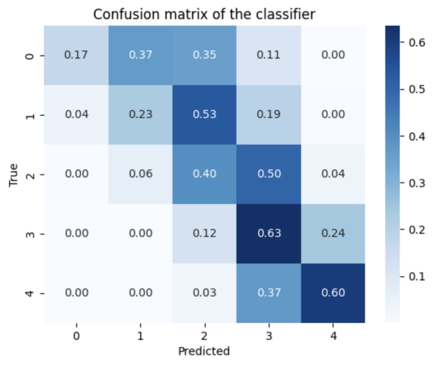
Parameter Tuning:

1. With the Ridge regression model selected for its performance, I proceeded to tune its hyperparameters. I employed GridSearchCV to conduct a search over a specified parameter grid — in this case, a range of alpha values. The alpha parameter in Ridge regression manages the strength of regularization, which is essential for preventing model overfitting. The GridSearchCV process utilized 5-fold cross-validation to assess the performance of each alpha value based on the negative mean squared error. This approach was crucial in identifying the most effective regularization strength for the model. After determining the optimal alpha value through grid search (Alpha=10.0), I also tried Bayesian Optimization to make sure that Alpha=10.0 is the correct value. After many tries I settled on Alpha=4.0. Then, I fitted the Ridge regression model using this parameter on the full training set. The refined model was then used to predict scores on the test dataset. I applied clipping to the predicted scores to ensure they remained within the expected range of 1 to 5, aligning with the known rating scale and maintaining the integrity of the model's outputs.

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
Best alpha parameter: 4.0
```

Challenges Faced:

1. I extracted 2 features namely:
UserAvgScore and ProductAvgScore that calculated the average score of each user and product. Even though it gave a very high correlation with the Score when I trained the model with both features added the RMSE decreased.
2. Before applying TF-IDF on my Review column I tried cleaning the text data from stopwords. After carefully analyzing the resulting column I realized that removing stop words transformed some negative text into positive. Ex: 'not good' to 'good'.
3. I also tried oversampling using SMOTE since the dataset is very biased towards the score 5. Unfortunately the model once again didn't perform well after oversampling. Then. I decided to see if undersampling would help increase my RMSE. When I undersampled the data, each score amount ended up around 7000. Meaning it removed more than %90 of the Score 5 column. Even without training the model I could tell that my RMSE would not get better.



```
RMSE on testing set =  0.5960250612885891
```