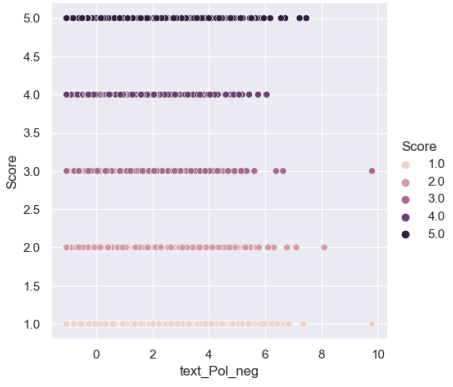
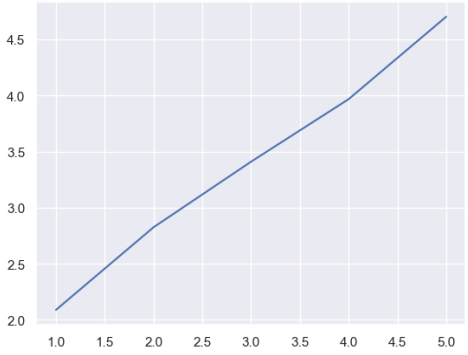
Name: Jialin Yu U41792151
Kaggle UserName: Jolene1 Yu

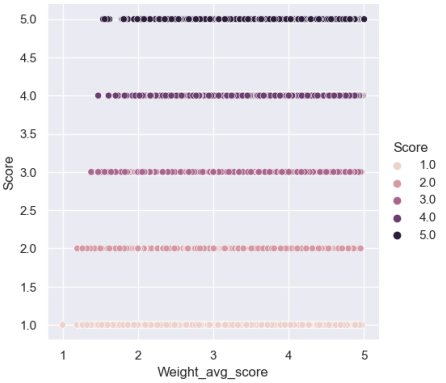
finding key features:



To maximize the usage of Text and Summary information. I analyze the polarity of the review and title using TextBlob, and word frequency with tfidf. Before using all the tools provided, we should first clean the Text and Summary data because there are many stopwords like 'I, They, But, Therefore' , etc. To clean the text data, I apply the Nlkt package. As tfidf returns a sparse matrix of word frequency for each review, It later becomes the most crucial element for classification in my model.

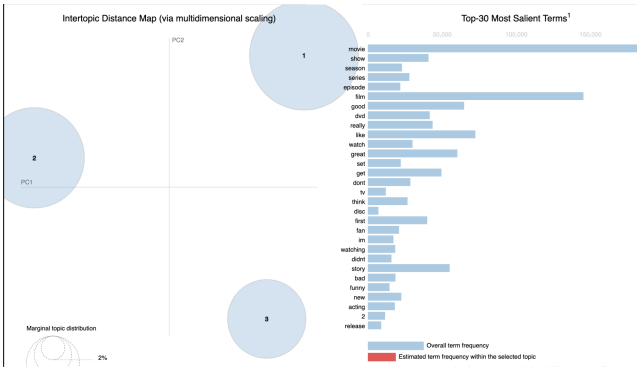
For polarity, I extract not only the compound polarity score, but also the negative, positive, and neu polarity. By plotting the graph against Score, I got graphs like the one on the left. We can see that the polarity mainly focuses on 0 to 2. However, the compound polarity and subjectivity which are also provided in the TextBlob package don't help a lot as the distribution spread out in every score. Analyse from these data, I am able to conclude that people who rate scores 1 and 5 on Amazon has strong emotion compared to those who rate the product with score 2, 3, and 4. So this makes the separation of clusters will score 2,3, and 4 extremely hard.





I compare the average user score to the actual score they provide and find out a linear relationship between them. Which implies that users might prone to score similarly even for different products. However, this feature doesn't work well as I suppose some users might only offer 1 feedback which will lead to overfitting, so my assumption above might not be true.

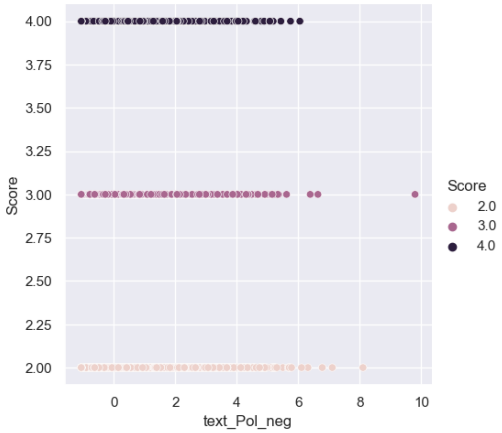
I also create a new feature called weighted product average score. For this feature, I produce average scores for each product regarding the usefulness (HelpfulnessNumerator/HelpfulnessDenominator) of a comment/feedback. In another word, I decided to give more weight to the score the useful comment provided because the more useful it is, the more representative a score the user provided would be. Applying the concept of Beta distribution in Naive Bayes, I use the average HelpfulnessNumerator as alpha and the average HelpfulnessDenominator as beta to weight all the product average score and assignment the score to each product evach row. By looking at this distribution, It's easy to find out that predict products with a higher average score with a high score can make the prediction more precise.



In addition, I also use gensim package to analyze the topic in Text and Summay. Using pyLDAvis package allow me to easily visualize and tun the parameter, and I found out the text feature located in three major clusters. And this feature allow me to improve my prediction accuracy by turning it to a sparse matrix and feed it to the classification model. (And interestingly, Summary feature can also be separated into 3 main clusters. From this feature, I found out that people are talking mainly about three aspects: Actors, Story/Plot, and background information related to the products themselves.

selecting the model

For the prediction model, I select both Logistic Regression and MultinomialNB. I have gotten the word frequency using tfidf, and I want to analyze and classify them. The best way to do it logically is to use Naive Bayse. And MultinomialNB provides this feature for us. One problem I encountered is that after applying tfidf is that it returns sparse matrix. To also feed other features into MultinomialNB model require me to also turn other feature into sparse matrices and hstack them together. However, a problem with MultinomialNBmodel is that it does not allow any negative number. Which is impossible when we try to hastack other features into a huge sparse matrix. So I decide to not using Multinomial NB and instead choose Logistic Regression model because of how the feature behave is closely related to the score. So it will behave well if there are some linear relationship between the result and input features. I also try random forest, Knn, and SVC. But I did not use any of them at the end because first they are more suitable for data with specific patter, and they all take long time to fit the model which is bad for testing. Most importantly, they did not provide a more preferable classification result.

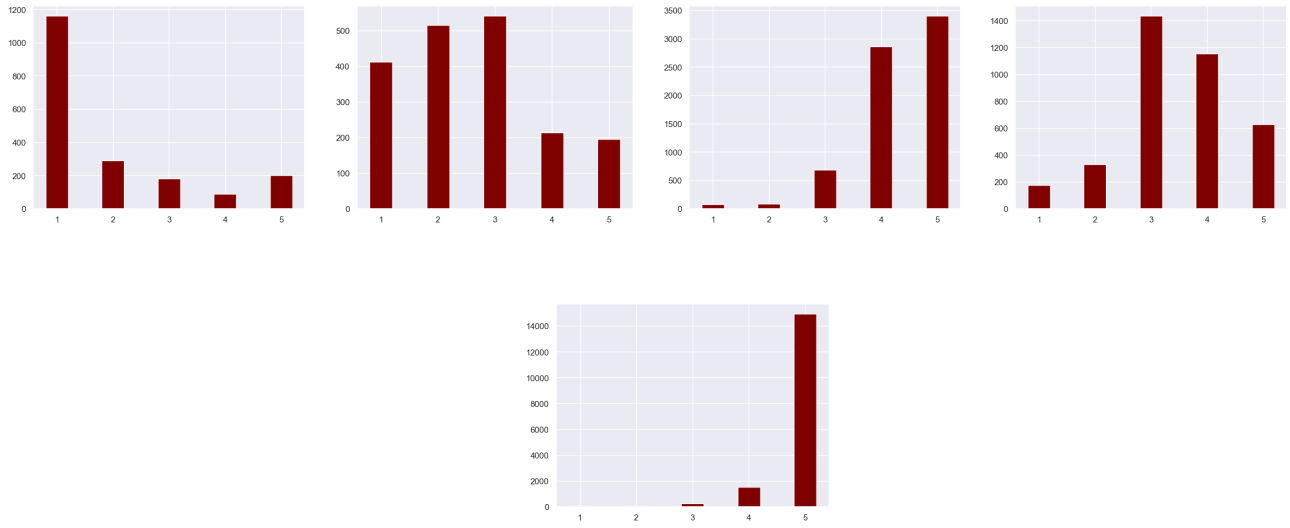


Puning model:

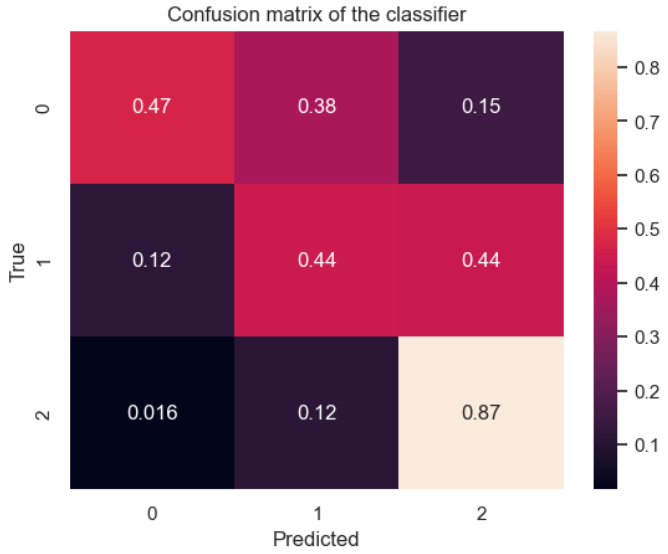
For Parameter tunning, I choose to use cross validation methods like GridSearch CV and Kfold. GridSearch can tune hyperparameters like learning rates, and the number of trees in a random forest model. So I used it when testing with knn and random forest model. It evaluates and optimize the model's performance for combinations of hyperparameters. Later, I also use kfold when I was testing with different features. It provides me a better and more comprehensive insight into the result of my classification/prediction as it Split dataset into k consecutive folds where each fold is used once as a validation.

Validating the model:

After running tfidf on the Naive Bayes model, we can see that the accuracy for predicting score 1 is 60%, the accuracy predicting score 5 is 90%. However, for data of true score value 2, most of them are assigned to scores 1, and 3. And data with true score value 3 are assigned to class of score 4 and 5. (See Graph below for more details, they represent the distribution of classification of each score in order. For example, the fist one on the left is the distribution of which class the score 1 data are assigned.)



So I want to find out if there is any feature that can seperate score 2,3,4. I found out there are minor differences between them regard to the polarity negative score and polarity positive score for Text feature, and helpfulness numerator. I use fit tfidf on dataset with score 2,3,4 and transform the Text and Summary features in training data, I did the same for topic sparse matrix. And using these two feature. I successfully separate score 2,3,4 using logistic regression model and get the following result.



It seems that it's a pretty good model to classify dataset with scores 2, 3, and 4. So then I first change the score value of data with score 2,3,4 to 3 and then use the LogisticRegression model (the first model that can predict data with score 1 and 5 with high accuracy) to classify data into 3 cluster (which is 1, 5, and 3). Then I separate out the data with predicted value 3 from the training dataset and feed it into the second model I just describe above to classify dataset to score 2, 3, and 4 clusters.