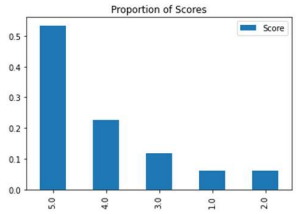
CS506 Midterm Report (Bingquan Cai)

1. Preliminary analysis

As you can see from the original dataset, there are some missing values in the 'Score' column which are the ones that we need to predict. Besides the 'Score' column, there are other features that we may be able to exploit. They are *'Id'*, *'ProductId'*, *'UserId'*, *'HelpfulnessNumerator'*, *'HelpfulnessDenominator'*, *'Time'*, *'Summary'* and *'Text'*. Now let's go through them one by one.

1.1 Score

First let's look at the distribution of score. The score 5 is over 50 percent comparing to other scores are at most nearly 20 percent. Score that bigger than 3 takes the majority which we need to pay more attention to in the following analysis.

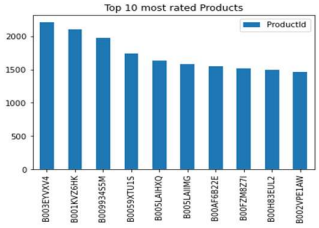


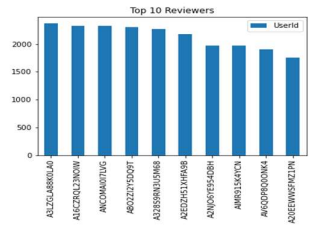
1.2 Id

'Id' is like an index for different samples and is not useful for prediction which we can be discarded.

1.3 ProductId & UserId

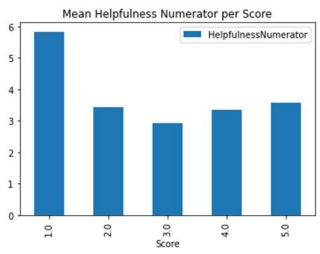
As you can see, the highest numbers of rating from the most rated product and the most rated user are at about 2000, which is a small amount compared to over 1 million samples. Therefore, we will not have one single product or on single user to have large impact on the prediction of score. Also, as the score 5 takes the majority, the difference in score between products and between users will not be particularly large. If we use these two features, the score we predict may overfit into the score distribution that belongs to its corresponding product id group or user id group. So, I decide to discard these two features.

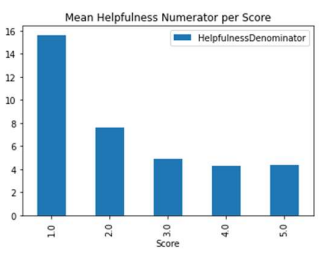


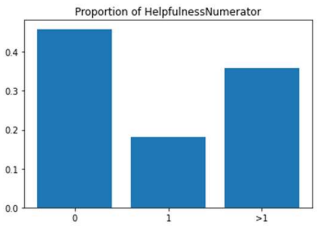


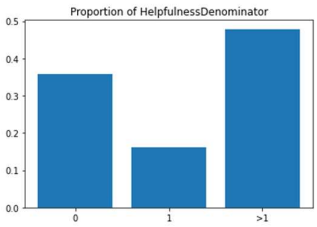
1.4 HelpfulnessNumerator & HelpfulnessDenominator

As you can see, the score 1 seems to have more helpful numerator and denominator than others, but it is hard to distinguish between score 3, 4 and 5 which have higher proportion among all the scores. Also, there are over 50 percent of the reviews only have 1 or even 0 helpful numerator and denominator. Therefore, these two features are not so informative and too many zeros may be bad for prediction that I decide to discard these two features.



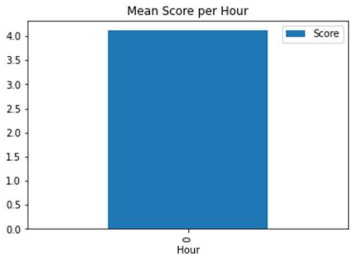


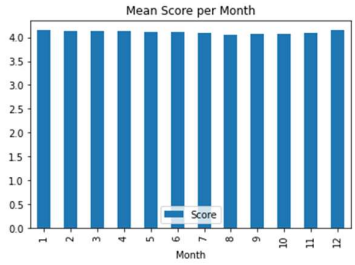


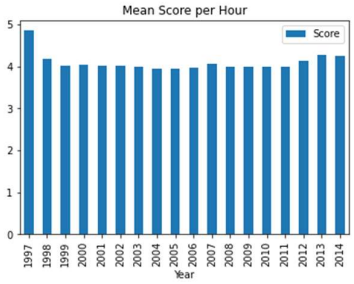


1.5 Time

Time is also what I want to discard because all reviews are from the same hour and the average scores of each month and of each year (except 1997 to be a little bit higher) are almost the same, which is not discriminative.

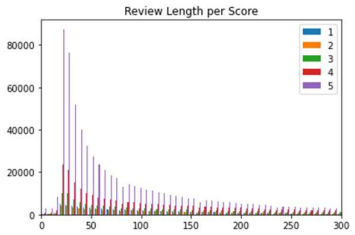


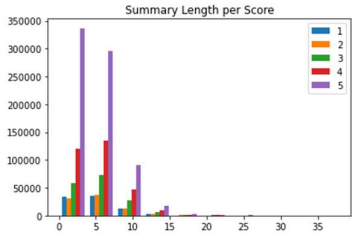




1.6 Summary & Text

In my opinion, these two features are mostly related to the score. For the length can't distinguish the different scores, I think the most important part is the content of reviews. It has a huge relation between the reviews and the score. People may write highly appreciative comments for a 5 score or write something to express their huge disappointment for a 1 score. It is a rare situation where people express their praise but then rate a low score. Therefore, I would like to consider the reviews most.





2. Data preprocess

As showing in the original dataset, the *'Text'* part is more informative than the *'Summary'* part. Sometimes it may be hard to get a lot of information out of a short summary. So I choose *'Text'*,which is more informative, to be the only feature for my prediction.

2.1 the empty text

There are reviews that only have *'Summary'* part but no *'Text'* part, so for each missing *'Text'* part, I fill it in with the corresponding *'Summary'* part.

2.2 vectorization

I use ***TfidfVectorizer*** for the text vectorization. For the parameter ***ngram_range***, I set it to be (1, 2) so that *TfidfVectorizer* may distinguish between, for example, 'good' and 'not good'.

2.3 split

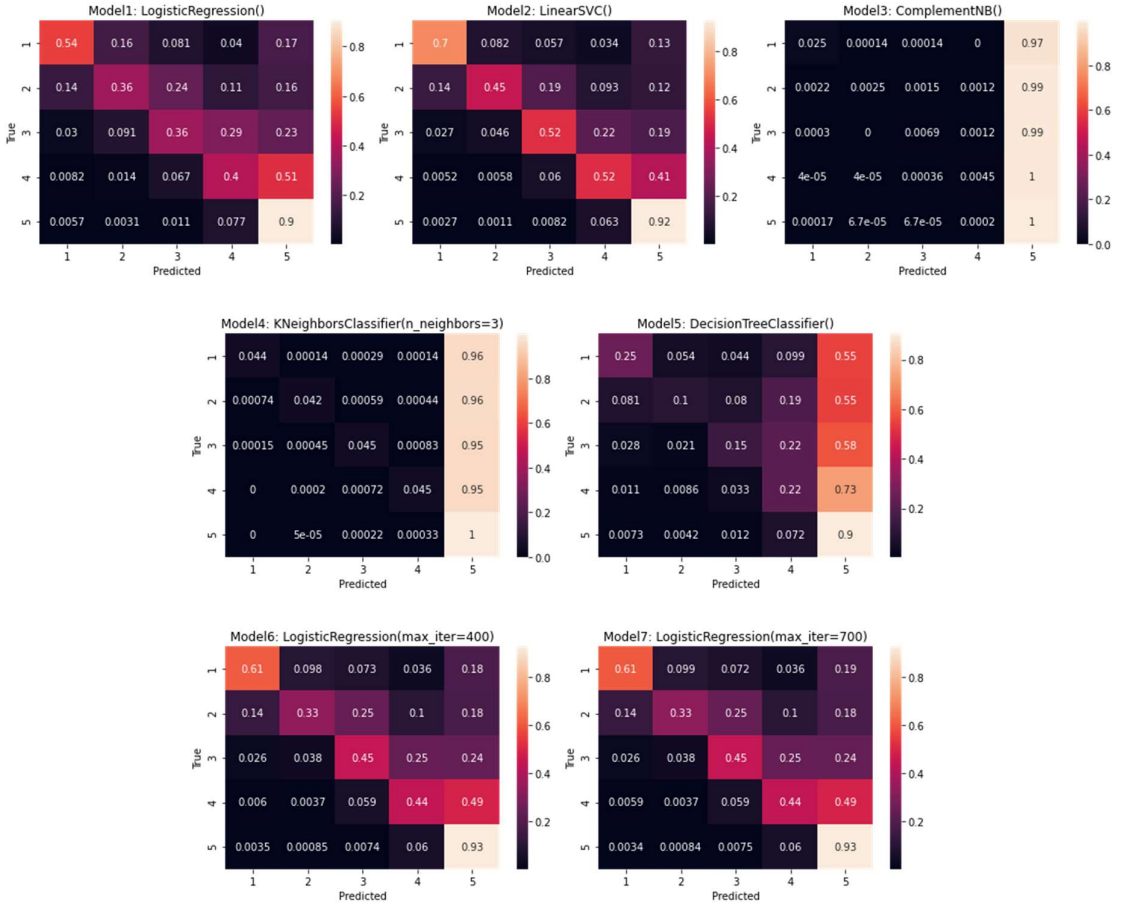
Since the original data is too large, I randomly take 40% of the original data and use ***train_test_split*** to set 80% for training and 20% for testing. I do a simple train-test-split because I think that even I use part of the original date and choose 20% of the data for testing, it is still large enough for us to judge on the performance.

3. Models select

I choose Logistic Regression Classifier, Linear Support Vector Classifier, Complement Naive Bayes Classifier, KNN Classifier and Decision Tree Classifier. Complement Naive Bayes Classifier is what I learn from online which is particularly suited for imbalanced data sets and the rest four classifiers are learned from class.

4. Training and testing

Model	Accuracy
Logistic Regression Classifier (max_iter: default=100)	0.6705904451818921
Linear Support Vector Classifier	0.7422918325178672
Complement Naive Bayes Classifier	0.5373400001788957
KNN Classifier (n_neighbors=3)	0.5544513716826033
Decision Tree Classifier(max_depth=20)	0.5717416388632969
Logistic Regression Classifier (max_iter =400)	0.7061191266313049
Logistic Regression Classifier (max_iter =700)	0.706029678792812



For the Logistic Regression Classifier of model 1, it turns out that the number of iterations reach the limit. So, I increase the ***max_iter*** parameter and the accuracy seems to be no difference after 400 showing from model 6. As showing in the model accuracy table and confusion matrix graphs, the Logistic Regression Classifier and the Linear Support Vector Classifier seem to be better than the others which have accuracy over 0.7. So, I will choose these two model for my final prediction.

5. Prediction

In the competition, I got Public Score of 0.65893 from the Linear Support Vector Classifier and 0.67674, which is even higher, from the Logistic Regression Classifier.

6. Conclusion

The result shows that a classifier with a better training accuracy does not always have a better testing accuracy. Also, there are some spaces where I can improve. For example, I could discover more features that are related to the score. Moreover, the helpful part may be useful if I could figure out how to deal with the large amounts of zeros. But overall, I still get a good performance on the Kaggle competition.