Haoxuan Sun U58198360 Kaggle username: Harry HX Sun

1. Finding Key Feature

Firstly, normalize the feature "Review Length" (which is calculated based on the length of the "Text") and "Time" as "ReviewLength_normalized" and "Time_normalized". Then drop "Review Length" and "Time".

Then NLP techniques are used to refining the "Text" and "Summary" features. There are 5 functions used when processing the training data, they are: expand contractions, tokenization, stop words removed, spelling correction and lemmatization. After processed by these 5 functions, the content in "Text" and "Summary" column will be shortened by dropping letters and replacing them by an apostrophe, reducing tokens and stopwords, correcting spelling and grouping together the different inflected forms of a word so they can be analyzed as a single item.

	word	tfidf
0	movie	6983.797656
1	great	6557.028139
2	good	5068.396632
3	best	3282.471388
4	love	2970.510214
5	film	2761.462931
6	classic	2380.913143
7	dvd	2026.240394
8	fun	1881.648178
9	funny	1618.301251

Figure 1 top tfidf

After refining text in the train data, Tfidf is applied on text and summary. TfidfVectorizer completes vectorization and TF-IDF preprocessing and add columns of words into the data. In experiment, I choose 500 most frequent words in summary and 700 in text. Figure 1 shows the words has top tfidf values. Also, I analyze sentiment using TextBlob. It provides polarity and subjectivity for the analyzed text and summary.

```
Score:
Text_sentiment_polarity: 0.42
Summary_sentiment_polarity: 0.36
Time_normalized: 0.09
Time: 0.09
Text_sentiment_subjectivity: 0.08
Summary_sentiment_subjectivity: 0.05
Helpfulness: 0.03
HelpfulnessNumerator: -0.01
ReviewLength: -0.08
ReviewLength_normalized: -0.08
HelpfulnessDenominator: -0.10
```

Figure 2 correlation with "Score"

Many other features are also calculated based on above features. To find the key features in training data, Person correlation is calculated for every of other columns with score (Person_correlation.ipynb). From figure 2, I find the features that have the most probability related with score.

2. Selecting Model

7 normally used machine learning models are tested here with default parameters. The 7 models are: Logistic Regression, Naive Bayes, KNN, Decision Tree, Random Forest, Gradient Boosting and LGBM model.

```
LogisticRegression: accuracy 0.625
MultinomialNB: accuracy 0.579
KNeighborsClassifier: accuracy 0.561
DecisionTreeClassifier: accuracy 0.493
RandomForestClassifier: accuracy 0.6
GradientBoostingClassifier: accuracy 0.612
LGBMClassifier: accuracy 0.637
```

Figure 3 7 Model accuracy

From Figure 3, we can see that the LGBM classifier has the best accuracy result. So, LGBM model is used for project.

3. Tuning Model Parameter

We have decided LGBM as our model. The parameter "n_estimators" is tuning here to get us a well-trained model.

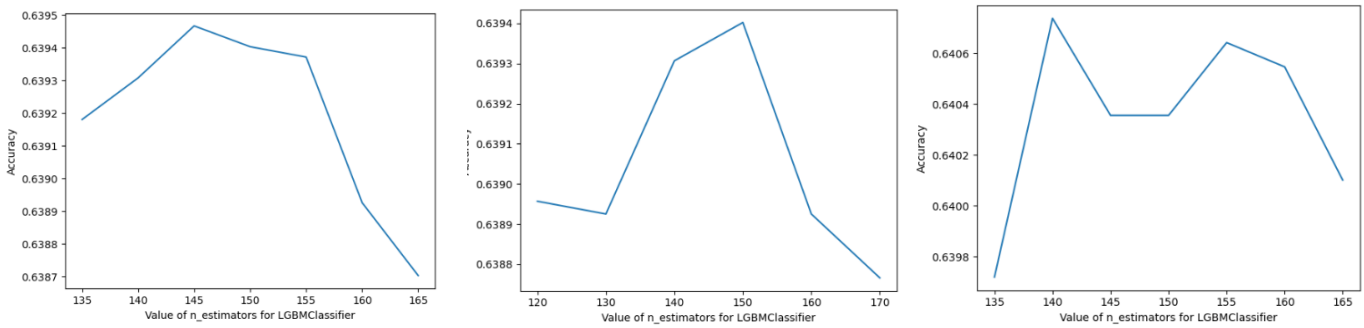


Figure 4 Accuracy VS n_estimators

Lots of n_estimators are tested to a best value to make our model reaches highest accuracy. Figure 4 shows the mean accuracy when n_estimators are ranged from 120 to 170. 155 is selected for our best value for n_estimators in LGBMClassifier().

4. Validating Model

After choosing the best parameter, we train our model with train data and the 5-fold cross-validation is applied to reduce randomness and improve reliability. Then we test the model on our test data and calculate the accuracy and MSE (Figure 5). The accuracy is 0.6406 and MSE is 0.9066. RMSE is 0.94 on Kaggle test set, which means the model is not overfit and underfit. Although it cannot predict score with a very high accuracy.

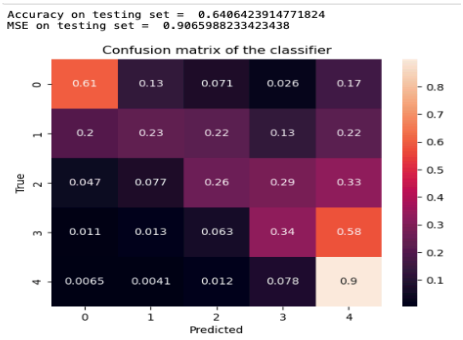


Figure 5 Confusion Matrix

5. Additional Effect to Improve Model

*5.1 Add feature: avg_score * num_review_product or avg_score*

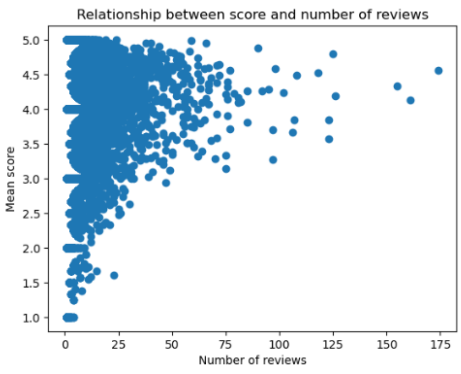


Figure 6 Relationship between score and number of reviews

```
Score: 1.00
avg_score: 0.63
Text_sentiment_polarity: 0.42
Text_sentiment_polarity_normalized: 0.42
Summary_sentiment_polarity_normalized: 0.36
Summary_sentiment_polarity: 0.36
Time_normalized: 0.09
Time: 0.09
Text_sentiment_subjectivity: 0.08
num_product_reviews_X_avg_score: 0.05
Summary_sentiment_subjectivity: 0.05
Helpfulness: 0.03
num_product_reviews: 0.01
HelpfulnessNumerator: -0.01
ReviewLength: -0.08
ReviewLength_normalized: -0.08
HelpfulnessDenominator: -0.10
```

Figure 7 Person correlation on new feature

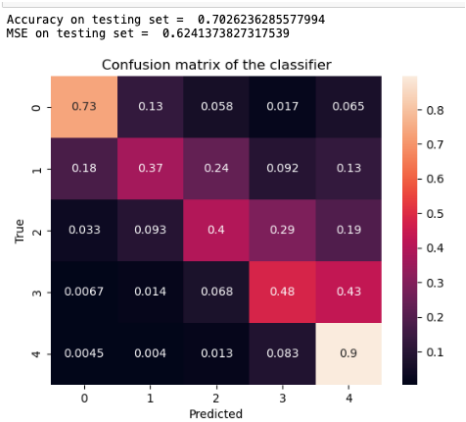


Figure 8 Confusion Matrix on 5.1 model

I find there is a relationship between number of reviews of a product and the product's mean score from reviews (Figure8). The more reviews got of a product, the higher score the product tends to have, but it is correct to reverse the statement. So, I create a new feature equals to avg_score times num_review_product. The Person correlation shows the new feature has some relationship with score (Figure7). And the confusion matrix (Figure6) shows that there is a great improvement on accuracy (0.703) and MSE (0.624). However, RMSE is 1.45 on Kaggle test set. Which means the model is overfitting on my train data. Hence, this is not a good feature.

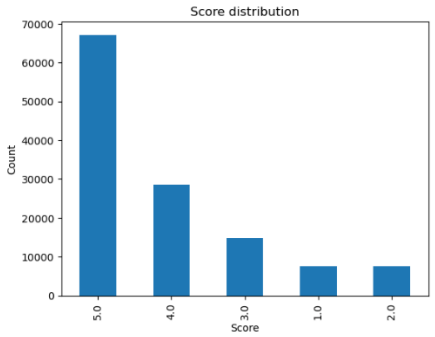
I also tried only adding avg_score feature based on the scores a product gets. The model gets 0.693 for accuracy and 0.625 for MSE. And again, the model gets 3.33 for RMSE on Kaggle test set. The avg_score feature should be very helpful for building a predict model, however, due to time limit of this competition, I haven't come up with a better idea on how to use this feature.

5.2 Test on Balanced Dataset

After analyzing confusion matrix of previous model, I found there are many cases that should be score to 4 but my model predicted to 5. Then I believe this may be caused by the highly imbalanced training dataset (Figure 9). I then resized the train set so that each score has a equal number of cases, which is randomly sampled from original dataset. Score 2 has minimum cases, which is 7567. Then our new dataset has 7567*5 rows (test_imbalance.ipynb).. I repeat the selecting model part on new data, and Random Forest model is optimal. The confusion matrix of new model is shown in figure 10, which looks great as there is a clear diagonal. The accuracy is 0.6634 and MSE is 0.8994. However, RMSE is 1.75 on Kaggle test set. Which means the model is overfitting on my train data. Hence, this is not a good model.

```
5.0 67188
4.0 28572
3.0 14857
1.0 7593
2.0 7567
```

Figure 9 Score VS Count




```
Accuracy on testing set =  0.6633893646262818
MSE on testing set =  0.8993551115339888
```

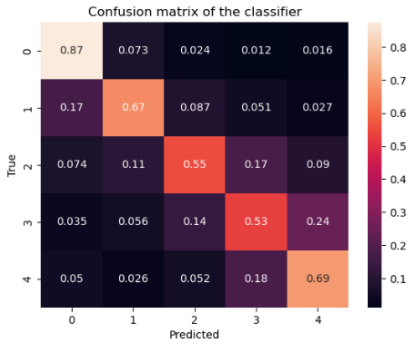


Figure 10 Confusion Matrix on 5.2 model

6. Challenge

The challenge for me during this competition are the usage of Python language, the familiarity of ML models, the understanding of data features and the experience of creating new features and also the poor performance of my laptop, which takes me hours to apply NLP methods on dataset.

My findings about how people rate products is: People tends to rate positive (3 or 4 stars), and the more reviews a product has, the higher score the product tends to have. But we cannot say that the lesser reviews a product has, the lower score the product tends to have. I used this assumption (in Part 5.1); however, it didn't help. But I am sure this can be used to build a better model, just in a suitable method.