Kaggle id: davidyh3

Yuhan Peng

Professor Lance Galletti

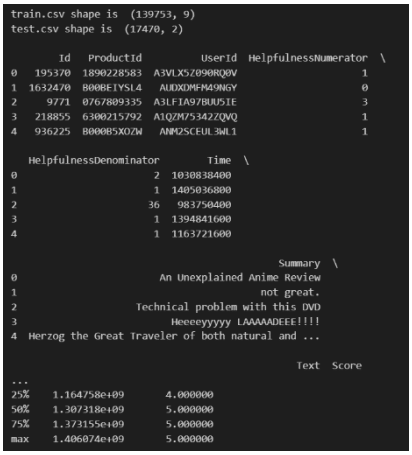
CAS CS 506

7 November 2023

Report on CS506 Kaggle Competition

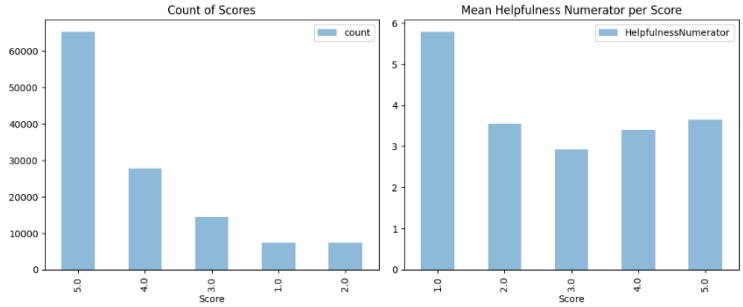
Data Analysis

Before we dive into the model and the library I am using for my project, we should see explore and have look at the data structures first. Here is the output of the structure.



From the result in the right picture we could see that "train.csv shape is (139753, 9)," which means that we have a substantial training set with 139,753 records but missing scores for a significant portion (122,283 scores present out of 139,753 rows). This indicates that some data cleaning or imputation might be necessary.

Then, with the following figures, we could have a conclusion that the greatest portion of the score is 5.0, followed by 4.0, 3.0, 1.0, and 2.0, which do relates to what the distribution looks like in the real world, either a P shape (a great number of 4.0 or 5.0 followed by 5.0 and 3.0), or a E shape (a great number of 5.0 and



1.0). This also give us the signal that we should put more emphasis on predicting scores that might have a 5.0 compared to scores that might have a 2.0.

After we have explored the data, what we need to do next is to do the data cleaning, the feature engineering (create new features, like the ratio of “HelpfulnessNumerator” to “HelpfulnessDenominator.” Or create the number of reviews per user and product could also be informative), the modeling, and the validation and tuning.

Model Choosing

For the model I choose, I use the K-Nearest Neighbors (KNN) model. Since "KNN relies on observable data similarities and sophisticated distance metrics to generate accurate predictions(Aggarwal)." It is valuable because it is non-parametric, meaning it makes no underlying assumptions about the distribution of the data so I could start at the very beginning.

The drawback of the KNN model is that it may not be the best model for high-dimensional data, particularly after adding TF-IDF features which tend to be high-dimensional and sparse.

I changed the value of random_state from 0 to 3. Even though the result after I changed the randome_state does not change my prediction a lot, it still makes my prediction better.This is because the KNN algorithm can be quite sensitive to the specific instances in the training dataset. Small changes in the training set can lead to different nearest neighbors being selected during prediction.

I also tried to use the Decision Tree Model for my data predictions. However, since the decision trees tend to memorize the training data if they are allowed to grow without constraints,

leading to complex models that do not generalize well to unseen data, it crashed a lot. Thus, I stop using it and decided to modify the KNN model.

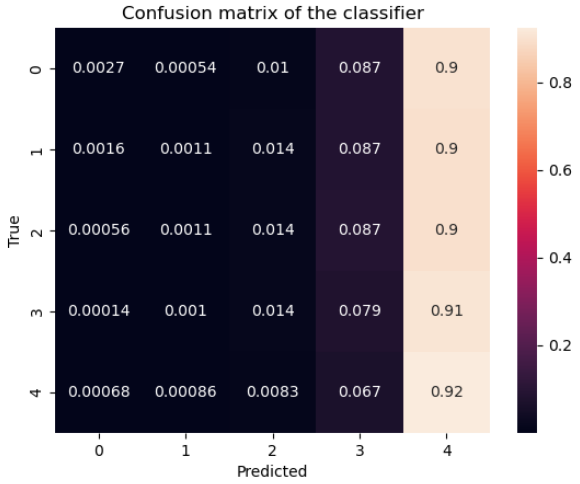
Features Evaluation

In my code, I choose Helpfulness and ReviewLength as my features. For Helpfulness, it shows that the larger this value could be, the better the score could be predicted. While for ReviewLength, the longer the ReviewLength could be, the accurate the predictions could be.

Other than these two features, I was planning to add TF-IDF (Term Frequency-Inverse Document Frequency) features to my model. This feature is a numerical statistic used to reflect how important a word is to a document in a collection or corpus. If it could compile, it could produce the ratio of certain words (for example, negative words or positive words), which will make the prediction more accurate. KNN can perform poorly in high dimensions due to the curse of dimensionality, that's why I have a crushed version of KNN models because the TF-IDF features I implemented seems to have bugs.

Confusion Matrix

For my matrix, my program is spending more than 90% of the time analyzing and predicting for group four, but the accuracy for the other groups are not that good as group 4.



Work Cited

Aggarwal, Sangeet. "K-Nearest Neighbors." Medium, Towards Data Science, 8 June 2020, towardsdatascience.com/k-nearest-neighbors-94395f445221.