CS506 Midterm report

Kaggle: Ruiding Feng

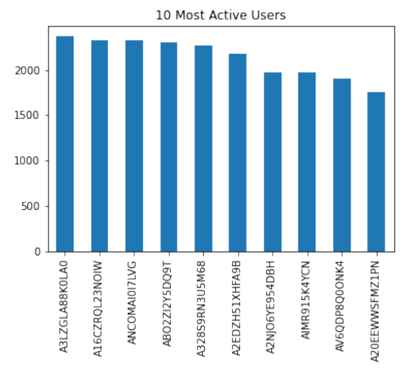
Goal and Introduction

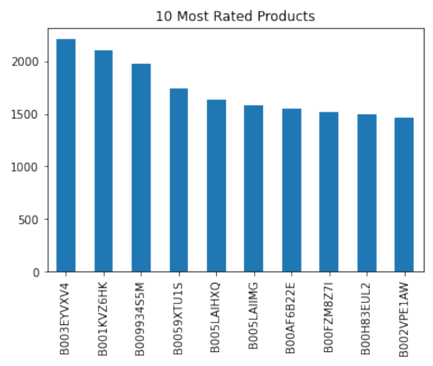
This project examines the relationship between the star rating score between user reviews from Amazon Movie Reviews using the available features. The features include unique identifier for the product/user, the number of users who found the review helpful, the number of users who indicated whether they found the review helpful, the timestamp for the review, the brief summary of the review and the text of the review. In the rest of this notebook, I'm going to utilize these features on the prediction of the star rating score. This project, if successful, is beneficial to estimate any unlisted movie's popularity or reputation, which could be further used in the recommendation system in this field.

Preliminary Analysis

This dataset contains a total of 9 columns. As each record row's identifier, "Id" is presumably pertinent to our modeling or prediction.
In contrast to the other features, "ProductId" and "UserId" are categorical features. I must thus process things in various ways. I'll then examine each one of them in further detail one by one.

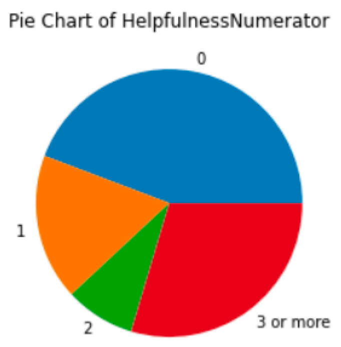
User ID & Product ID
The collection contains more than a million records. The most popular items and users each have less than 3000 entries. In other words, they hardly affect our models in any way. I'll just leave them the way they are.

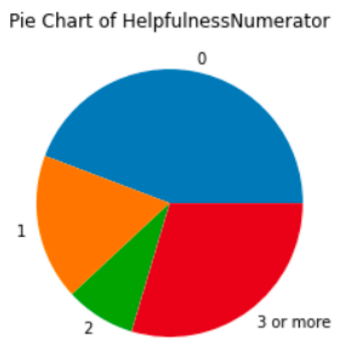




Helpfulness Numerator & Denominator

The following visualization shows that neither of these two attributes makes up more than about half of a certain value for each of them. It is therefore safe to leave them in their current state, barring any necessary standardization.





Score

In contrast to prior features, the rating of 5/5 is too high because people were probably going to score just one movie by default. This could affect our forecast because a predictor tends to establish an unrecorded value as the most frequently occurring value (according to Maximum Likelihood Theorem). We most definitely do not want to see this. In the remainder of my project, I'll make sure that doesn't happen.

Summary & Text

Ideally, I'd like to use every column the dataset has to offer. However, if any two of them have an excessively high correlation with one another, it is not practical. I therefore print out a word cloud of the text features: Summary and Text, and I make a basic assessment.

As can be observed, they don't share many keywords, even if this is partially due to the Summary's greater concision and lack of information. They must be sufficiently independent of one another, I suppose.

Missing values

Most of the empty values are present in the Score column. This is normal as the score is the output and the training/test set is mixed.

Feature Extraction

1. **The main elements used for prediction are HelpfulnessNumerator and HelpfulnessDenominator. I use it by defining a new column called "Helpfulness", which is simply defined as the HelpfulnessNumerator divided by the Helpfulness Index (HelpfulnessDenominator). The more helpfulness, the more helpful your evaluation will be.**
2. To deal with missing values, use the Fillna function. We add it to NULL for the empty part of the data given to us by the user, so that our data can be complete.
3. **Convert the contents of "Text" and "Summary" into vectors, and then integrate the vectors into matrices, i.e. text_matrix and summary_matrix.**
4. **Convert the useful information other than "Text" and "Summary" into a matrix, i.e. numerical.**
5. **Stack all the matrices (the three above) into one matrix, using hstack.**
6. **Remove the redundant information, using df.drop.**
7. **Store the information in the matrix into df for subsequent training.**

Techniques and Models

We mainly used two classification methods to make predictions. k-nearest neighbors and decision trees. First, we used the k-nearest neighbor method. The ANN algorithm looks for similarities in nearby features. In the context of our project, this applies to the fact that similar scores can have a HelpfulnessNumerator and a HelpfulnessDenominator. After trying

several k values at 20 and looking at the error plots, I found that the k value at 20 gave less error, so I left k=20. Then we used a decision tree with a maximum depth of 20. With a large amount of data and very few categories used for prediction, the decision tree algorithm should perform optimally.

Challenges and Further Improvements

I think the biggest thing that needs improvement is my computer configuration, I spent a lot of time trying to put in maximized intervals to get better predictions. However, my device often exploded in memory, making it take a lot of time and not get the data. So, I added an interval limit of 0.1 to allow my device to run the results. However, at the same time, this made the predictions less accurate again. So, my device would run more accurate data if it had more memory.