Prediction on the rating of fine food and evaluation for the helpfulness of review based on the Amazon Fine Food Review

Capstone Project by Liyan Cen

1 INTRODUCTION

Customer reviews is an important part of the Amazon shopping experience. It is so far the major reference and indicator for Amazon to evaluate the quality of their products, and the performance of their sellers.

Additionally, it is a great tool for customers to compare between products. Amazon always tend to place the most helpful reviews at the first place of the list as a reference for customers, and also as a standard for reviewers to follow. The company carefully considers any changes to the ratings and review system to ensure customers and sellers continue to trust it.

In order to improve the quality of the review system, Amazon started to distribute free products to quality reviewers years ago. To be more specific, Amazon sends free products to their top reviewers. Once received, all they need to do is to review the product on the website, and then keep the product themselves. Amazon receives a ranking that says how helpful review was to potential consumers. The more helpful the review is, the more likely the reviewer could become a top reviewer.

In our project, we divided up the analysis and prediction into two parts with two iPython notebooks, they are “Ratings vs. Reviews”, and “Helpfulness vs. Reviews”. For “Ratings vs. Reviews” section, we use classification models to examine ratings based on the reviews, and to predict the pattern of how the reviewers rate products based on written comments. This model could help amazon to evaluate the reasonableness of the ratings. For “Helpful vs. Reviews” section, we use classification models as well to predict which review is more helpful.

2 Data Acquisition and Cleaning

This dataset consists of reviews of fine foods from amazon. The data span a period of more than 10 years, including all ~500,000 reviews up to October 2012. Reviews include product and user information, ratings, and a plain text review.

Because there is memory limitation of our CPU, we first cross out the “HelpfulnessDenominator” value that is smaller than 10. Also, we cross out all rows which has null values, and make sure there is no duplicate contents. As a result, the length of data changes from 568,454 to 21,463.

In order to make prediction models with better performance along with less noises or outliers, we first reprocess the text to make our feature set more correlated with the target variable. By doing that, we create a function called “normalize_corpus” to remove punctuations, digits, html tags, accented characters, special characters,and stop words. At the same time, we lower all strings of each text.

For both the classification and regression models, we use text as our feature set by converting each word from each text into numerators by “TfidfVectorizer”. Frequent words across all documents may tend to overshadow other terms in the feature set. The TF-IDF model is functioned to solve this issue by using a scaling or normalizing factor in its computation.

2.1 Data Wrangling and Cleaning - “Ratings vs. Reviews” section

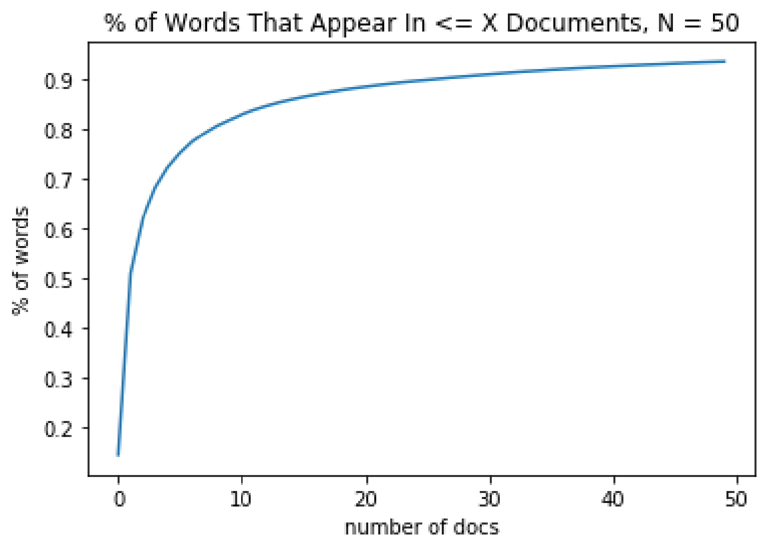
For this section, we use text feature as our feature set, and score as our target variable. Classification models are used because our model predicts and classifies reviews into a score ranging from 1 to 5.

2.2 Data Wrangling and Cleaning - “Helpfulness vs. Reviews” section

For this section, we use text feature as our feature set, and “Helpful” as our target variable. We get “Helpfulness ratio” by having “HelpfulnessNumerator” divided by “HelpfulnessDenominator”. Classification models are used because our model predicts and classifies reviews into either “0” or “1”, while ratio greater than the mean will be equal to “1”, and less than the mean will be equal to “0”.

More details on the data cleaning process can be found from the IPython notebook. The cleaned dataset is then ready for explorations.

3 Data Exploration



Based on the graph, we can see that around 70% of the text features appeared in about 4 or 5 documents, so we set our min_df to be “5”, which means we cut off the words that occur in 5 documents or less.