Dawei Wang

October 27, 2021

U65522424

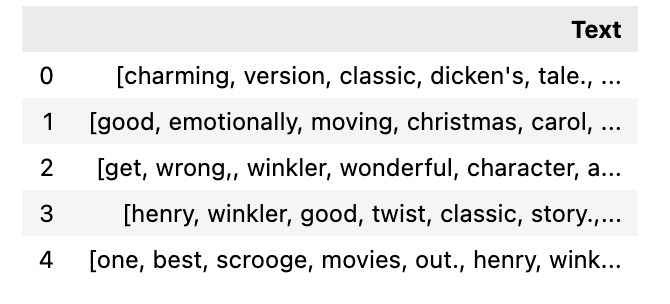
CS506 Midterm

Preliminary analysis:

There are 9 columns in the dataset, but I think the most important column is the “text” column. At the beginning, I thought we might use HelpfulnessNumerator and HelpfulnessDenominator to predict our data score, but I realized that HelpfulnessNumerator and HelpfulnessDenominator can only predict the credibility of the review. But every review can be either positive or negative, we can not distinguish negative reviews and positive reviews by HelpfulnessNumerator and HelpfulnessDenominator. So if we want to predict the “score” of the review, the only way is to analyze the text. If there are more positive words in the review, the score score of the product might be higher. In contrast, If there are more negative words in the review, the score score of the product might be lower.

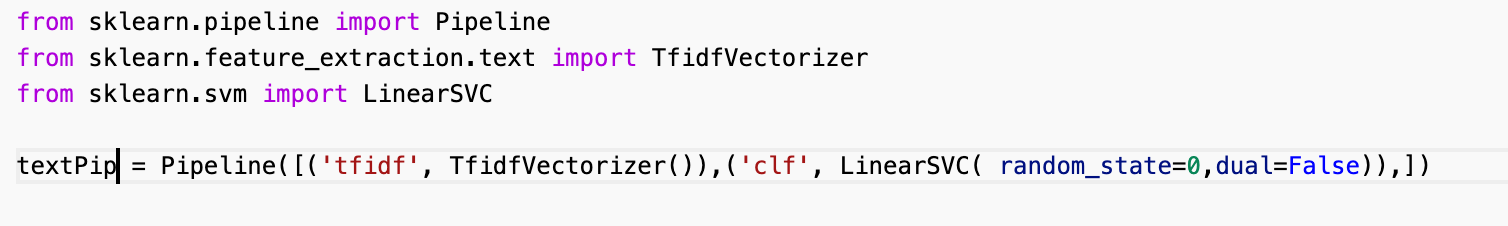
Feature extraction:

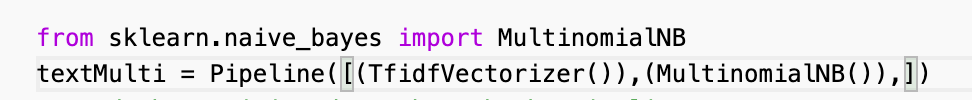
CleanText column: Review in “text” column contains a lot of useless information. First I try to change every character into lowercase, then I remove all the punctuation and stop words. The library I used was nltk.corpus.stopwords. One problem with using this dataset was it removed the stop word “Not”. In some reviews, the stop word “Not” changes the review's meaning from positive to negative. But after removing all stop words “not good” and “good” have the same meaning. So I kicked some words out of the filter list. Finally I reorganize the data as clean as possible.



countText: Word count for every review. People who write more words might tend to give a higher/lower score.

Flow, decisions, and techniques tried:

After I got clean “text”, I tried to use the algorithm SVM(Support Vector Machines) with TF-IDF(term frequency-inverse document frequency) . The reason why I chose SVM was because it is a "best fit" hyperplane that categorizes data. It’s the most efficient way to classify data.TF-IDF to the data you provideI just iterate over all the data, and get the frequency of each unique word. And TF-IDF used to encode the value of a term in a document vector. In programming, I also use Pipelining to do the TF-IDF. Pipeline is a good method for text data analysis and use it later for grid search. However, I also pipelined the Multinomial Naive Bayes Classifier for training with TF-IDF Vectorizer. 



Model validation/testing:

The SVM algorithm creates the best line or decision boundary that can segregate space into classes so that we can easily put the new data point in the correct category in the future. I also tried to create a heat map to compare the sample result with my actual output. But the result was that my model didn't work well. However, I work on the Jupyter NoteBook so I can test every step of my work. To prevent data lost or messed up, I also used DeepCopy to prevent changes to original data.

creativity/challenges/effort:

It’s hard to find the correct path. At the beginning, I tried to predict scores by HelpfulnessNumerator and HelpfulnessDenominator, and it wasted a lot of time. After I tried to work on the review, it’s hard to find a way to extract the useful information we want. However, it’s also really hard to convert the concepts that we learned in class into the actual project. Another changeling point is we have to be familiar with the python library.