CS506 Midterm

Andrew King

CS506

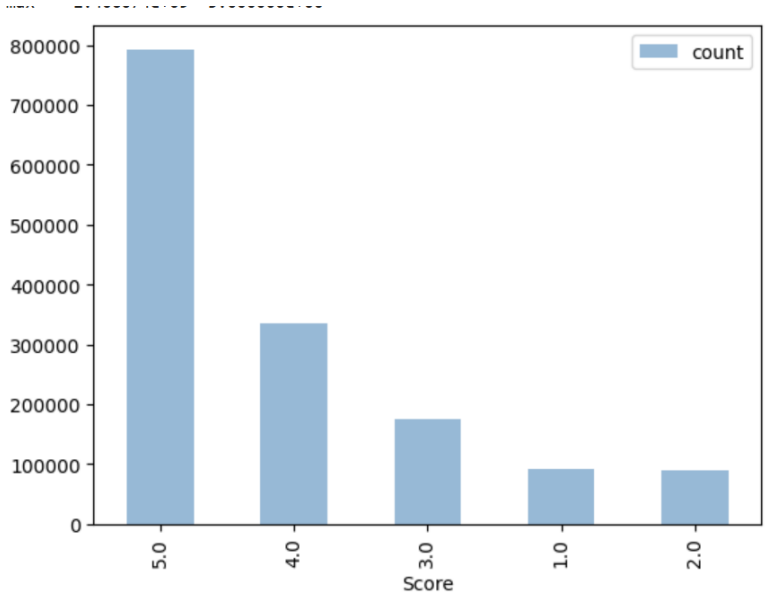
Midterm Report

CS 506 Midterm report

Predicting amazon product ratings given reviews.

1. Approach

I approached this project with an initial focus on feature extraction and text pre-processing, followed by training utilizing a K nearest neighbors model. I also utilized Sentiment analysis through the NLTK vader library to further enhance my score.



The initial training set contained a distribution of score ratings of amazon products ranging from 1-5 with key features on each review. Initially, I believed that

normalizing the amount of positive and negative scores would counter overfitting and allow for a better accuracy score. After normalizing, however, the training accuracy decreased. Therefore I used the initial trainingset as given.

2. Text Preprocessing

I first began preprocessing the text by taking various approaches to visualize the data, including utilizing a wordcloud with my list of stopwords to identify the keywords of the summary and text columns. This coincided with the prevalance of far more high ratings in the dataset.

To preprocess the text, I began by dropping any low quality rows. This includes NA rows or rows with less than 5 words. This was an arbitrary amount that I chose, but it yielded a better accuracy. I then began finetuning by utlizing the NLTK stopwords library along with a tokenizer to remove commonly used and sentimentless words, [a, an, the, not, or, will to further reduce the trainingset. I then tokenized the words to extract the key vectors I would then use for training.

3. Sentiment Analysis

I performed sentiment analysis using the NLTK SentimentIntensityAnalyzer package. Once I had removed all stopwords from the text column, I merged text with the summary to allow for greater penetration. I attempted to run this on the merged cells, but the processing time was far too long for the timeframe of the project. Instead, I analyzed the sentiment of the summaries, believing that people would indicate whether their reviews were positive or negative through the titles. This had a far quicker processing time and only took a few hours. I then saved a dataframe with the sentiment scores and rating ID to a seperate CSV file to avoid having to process it again in the future.

4. Training

I utilized the provided K nearest neighbors code in the end to train the final model. I experimented with using logistic regression and SVM, but I was unable to achieve a greater accuracy than through just K nearest neighbors in a suitable amount of time. SVM showed higher promise, but unfortunately the training time was far longer than K nearest neighbors was not complete before the final deadline. In theory, however, SVM would be more suitable as the features list was sparse, and the SVM model would have greater effect with features of this size. I later determined that, although I was unable to achieve a great accuracy through hyperparameter tuning along, removing median scores was able to achieve a much better result. This is, in effect, “cheating” the original vision of the project, as my model would only be classifying reviews as 1 or 5, but this ended up being more accurate on the kaggle submission, so I persisted with it.

5. Saving, final submission

My final submission ended up being a hybrid of sentiment analysis conjoined with preprocessed text, modeled with a K nearest neighbors classifier to identify reviews as 1 or 5 stars. The sentiment analysis increased accuracy by around 3%, and removing the middle star ratings from the set improved accuracy by around 11%, leaving my final accuracy at 54%. I am satisfied with the result, but I am interested in pursuing other approaches through bag of words or through greater fine-tuning the sentiment analysis by preprocessing with LSA.