Andrew Tuckman

Crovella

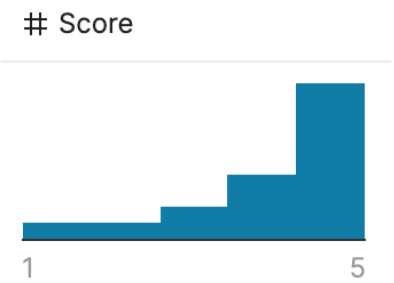
CS506 A1

27 October 2021

C506 Midterm Report

**Preliminary Analysis:**

The competition goal was to predict the star rating associated with user reviews from Amazon Movie Reviews using the available features. My initial thoughts for this would be to clean up the reviews of the training data, create a model using an available classifier based off of these reviews, and predict the new star ratings using this model. After taking a look at the graphs made by Kaggle on the training data star ratings (as shown to the right), I noticed an overwhelming amount of 5 star reviews. Therefore, I assumed about the same would occur for the final predicted ratings as well.




**Feature Extraction:**

I chose to look solely at the text reviews of the training data in order to create a fitted model for the test data. The only other column of the training data that I considered was the summary column. However, I went off the assumption that the reviews in column ‘text’ would be sufficient for a decently accurate model since they are longer and, from a glance of the first 100, seem to be more descriptive. Additionally, inconsistencies in the positivity between the summary and the review could cause an inaccurate model. These reviews would be pre-processed and vectorized before being fitted. The actual features extracted from these reviews are done through the scikit learn function TfidfVectorizer().

**Flow, Decisions, and Techniques Tried:**

The flow of my code begins with the training data. It reads it from the .csv file, extracts the two features I am going to use, and drops any samples with null values. Next is the text pre-processing. This is the first hefty part of the code. The beginning of the pre-processing removes any special characters from the text. The text is then split into lists of single word strings and made lowercase. It is split into lists in order to parse through all of the text word by word and remove any stopwords, defined by a list I made going off of lists from online. I decided to go about it this way instead of using a built-in one like in Natural Language Toolkit because I wanted to easily add my own. I felt the need to add my own after noticing that the reviews contained stopwords but without proper punctuation, so a premade list would fail to remove them. The code then implodes the list of words back into strings and they are ready to be fitted.

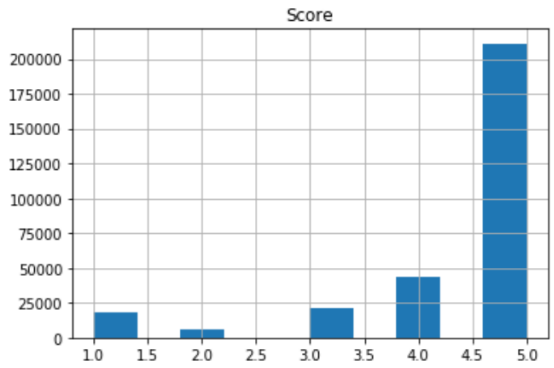
The training data is further split into its two columns, with ‘Text’ being assigned to X_train and ‘Score’ being assigned to y_train. These are passed through a pipeline, the second hefty part of my code, which contains the vectorizer which converts the pre-processed text to a matrix of TF-IDF features. Additionally, it trains the Linear Support Vector Classification. Through testing, I found the best score with a Linear Support Vector Classification over other classifiers such as Naive Bayes. For LinearSVC specifically, I tested multiple C values and found 0.2 yielded the best score. I stumbled upon the Pipeline package from sklearn when looking at the LinearSVC page on scikit learn’s website. I found it to be an efficient way to perform both steps in the overall algorithm together.

The final part of the code is pre-processing the test data in the same way that the training data was, and making the predictions on the star ratings. The ease of using a pipeline is revealed at this step as X_test2, the ‘Text’ column of the test data, is run through the vectorizer and

trained classifier using predict(). The rest of the code is simply creating the .csv file for submission.

**Model Validation/Testing:**

Besides testing my implementation of the predictions by uploading to Kaggle or running independent error checking, I wanted to return to a point I made in my preliminary analysis. I assumed that with the training data having a majority of five star ratings, the test data would as well, in addition with a similar trend for the other number ratings. As one can see in the histogram on the right, made from the test data of my best submission, it follows this trend. With a Kaggle score of 0.99955, I feel satisfied with my algorithm and the predictions it makes.




**Citation:**

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.