Feature Engineering Analysis
Byron Becker

I would first off like to state that I did not get the opportunity to implement all of the features that I would have liked to implement. I had many in the works, but due to a lack of "python skill" and familiarity with scikit-learn, I got hung up and spent a lot more time debugging issues than actually implementing and testing features. In my classify.py file, you will find many of these "in the works" features commented out, although there are many helper files that I did not include in the final submission. That being said, I learned a lot in this assignment about scikit learn, data wrangling, and feature engineering using outside sources in this assignment, and will look into more kaggle competitions in my spare time going forward.

Results

Having started with a baseline of ~60% using the original code, I was able to get my spoiler prediction score up to 65.7% using the features specified below, but am pretty unsatisfied with that result as I had many ideas that I wanted to implement or was in the process of implementing that I feel would have pushed me up over at least 70%.

Features

My features that were implemented (i.e. changed from the baseline) were as follows:

1. I changed the countvectorizer to a tfidfvectorizer with the following params:
    a. Analyzer = 'word'
    b. Strip_accents = 'ascii'
    c. Stop_words = 'english'
    d. Ngram_range(1,3)
    e. Smooth_idf=True

The reason that I used a tfidf vectorizer was to penalize frequently and most likely unimportant words from influencing the classifier. I additionally did some preprocessing with the text eliminating accents, and putting in the stop_words parameter to help detect certain unimportant words. I experimented with the Ngram_range, but found after several submissions that looking at single grams, bigrams, and trigrams was most effective. One can reason that phrases like "I can't believe" or "I was shocked" are counted whereas any additional words are unimportant in the spoiler determination.

2. I additionally appended several phrases including words that I thought may signal spoilers to the end of my train.csv file. I figured that doing so with the spoiler column set to 'True' might help weight some of these words such as 'died, killed, baby, shocked, finale' more heavily. To come up with some of these words, I used intuition, as well as \ looking at the top 10 print out from each run of classify.py

There were also several features I was in the process of implementing, but did not finish in doing so. Hopefully these give a further insight into my intuitions of going about the assignment. These to-be-implemented features were.

1. Creating a pipeline of fits and transforms corresponding to different features. For example, tfidfvectorizing the 'sentence' column and assigning it a certain weight, while countvectorizing the tropes column and assigning that a certain weight, and then putting the transformed combination into the SGDClassifier and fitting it. I had several ideas of doing this using FeatureUnion and Pipeline in sklearn, but couldn't get either to work properly.
2. Pulling data from imdb and adding it to this pipelined list of features
    a. Note: I was very close to implementing this as I had a script pulling data from imdb, but errors with creating the pipeline were limiting me from adding this to my list of feature inputs. My goal was to, given the page(title of a show), pull cast data about that show and then if the page matched, look to see if in the 'sentence' column, a word matched to a character name.