**Jessica Garson**

**Movie Sentiment In Reviews - Kaggle Competition**

**The Problem**

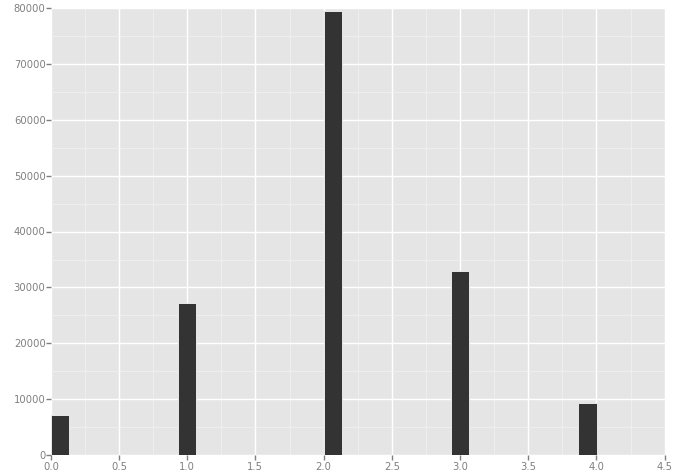
I entered a [Kaggle Competition](Kaggle%20Comeptition) that had you categorize sentences from movie reviews. Each review was broken out into sentences that could be confusing to categorize. The training dataset had scores assigned to each sentiment which were as follows:

0 - negative  
1 - somewhat negative  
2 - neutral  
3 - somewhat positive  
4 - positive

The training dataset lacked any scores that assigned sentiment to the reviews. The role of the model is to assign a sentiment score to each review. The overall goal of the competition was to create a csv that had the phrase id of each review and a corresponding sentiment score. The overall evaluation was based on the accuracy of the csv.

**Hypothesis**

When I viewed the from the sentiment scores broken out, it was clear that the majority of the scores were score 2. Here is a break down of the scores from the training dataset:



My first instinct was on Random Forests but I never got the model to work for this purpose. When that did not work as well as I would have hoped I turned to a logistic method.

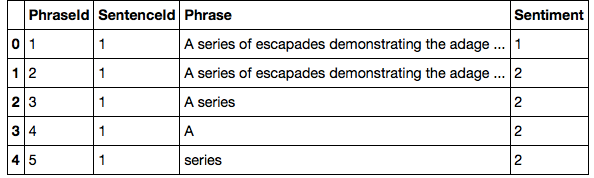
If I created a model that only predicted a score of 2 it would fair pretty well, so I assumed that a logistic method that I’d heard works fairly well with a lot of data in the middle would be adequate in the competition. However, I assumed using the ensemble method would be a better fit since it would push the scores closer to the middle.

**The Dataset**

Since I participated in a Kaggle competition, I had the luxury of having my data already provided to me and broken out into a training dataset and a test dataset. Doing this competition was actually a plan b when my original plan proved to be too ambitious for the short timeframe of the class.

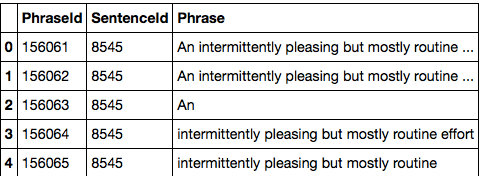
The training dataset consisted of 156,060 rows and the test dataset had 66,294 rows.

A sample of the training dataset can be seen here:



As you can see it composed of a PhraseId, SentanceId, Phrase, and Sentiment score.

A sample of test dataset can be seen here:



This dataset is the same except it is lacking the sentiment score.

The fact that this dataset was clean and had a few features made it easy to work with. The size of the dataset was also not too large, which made a good size to play around with a few different approaches.

One challenge with the dataset was that the main feature I had to work consisted mainly of text. This proved to be a bit tricky at times, and I had to use the count vectorizer function in sckitlearn to get around this key challenge. Another challenge was for the most part the reviews did not use overly positive or negative language, which made the reviews a bit harder to classify into scores.

**Issues with Random Forests Models**

As I mentioned my first instinct was to use random forests to tackle this issue and due to the amount of features I never quite got it to work. I attempted to use PCA, to solve this issue which seemed like the best option but than I got this following error:

TypeError: A sparse matrix was passed, but dense data is required. Use X.toarray() to convert to dense.

When I attempt to use this line of code: X\_train\_array = X\_train.toarray() my kernel dies. I think I can dummy out the features, to solve this but I started playing with logistics methods and got those to work quite successfully.

**Why Logistic?**

Logistic might seem like a strange choice for this type of problem since I’m dealing with non-binary data. Early on I was trying random forests and I could not get it to work in the ways, I was envisioning, it kept

While other methods might be better for this type of problem, I was interested in finding the cut points for the data. I figured I could

I was thinking that I could adjust the scores to be as follows to better fit a

0 = 0

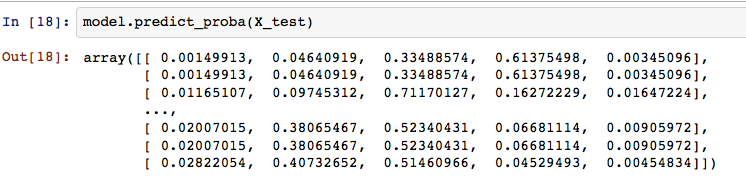
1 = 0.25

2 = .50

3 = .75

4 = 1

What I found while running the logistic model was that it automatically adjusts it to be multinomial. I discovered this while running this line of code:

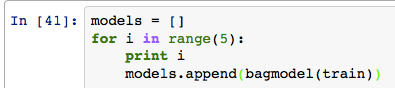


The straight logistic approach produced better results than I was expecting and even outperformed my first attempt at bagging. I was expecting this approach to work as a starting point, but it wound up being easier to work with and less clunky than other things I tried before that that point. This method produced a score of 0.60679 on Kaggle.

When I attempted to use the ensemble method bagging to split up my training set and combine them together. In my second attempt I also converted a lot of the code over to functions. This approach combined it into one big training set and from there I used the most common method to choose number returned. This might be a place where I could improve upon my code and choose a different method in the future.

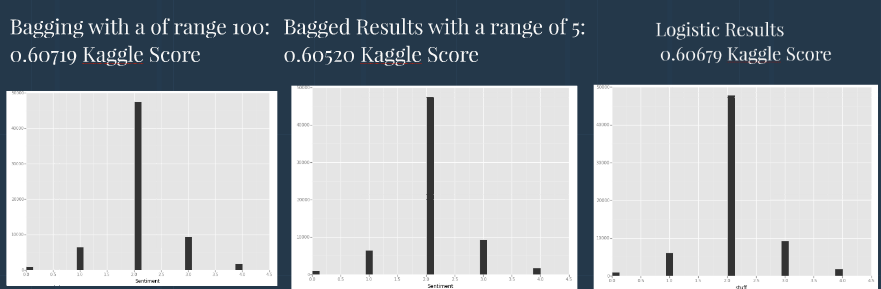
My first attempt at bagging did this method did not fair as well as the straight logistic approach. However, when I adjusted the range or rather the amount of training sets created from 5 to 100, the results actually were as I first expected the best of the all the methods I attempted.

Here is the line of code I adjusted that helped improve the results of the model, while taking much longer to run:



In Kaggle, he first bagging approach yielded a result of 0.60520 and the adjusted range to 100 yielded a 0.60719 this was a change of + 0.00041.

Here is a comparison of the three methods:



**Next Steps**

I’m lucky in the fact that I still have a few months to continue submitting code to kaggle. I’m planning on using this as a way to continue learning and playing around after this class ends. On my list of things to try are to try a voting scheme, which would have non 2s be weighted more. Also trying on combining random forests with the logistic methods I’ve been playing around with may also be a good next place to go. I also wish I had more time to work with a natural language tool kit. Building a more robust random forest model may also prove to be more effective as well.

**Business Applications**

Since this dataset came from rotten tomatoes reviews, a few similar businesses such as Amazon, Netflix, Hulu, IMDB, and other such places may be interested in using similar models. For my own uses it was helpful in learning how to break down text and apply in the ways that I did in these models, and I hope to use for classifying resumes and other such tasks in the future.