Task 6 – Health Inspection Prediction

Derek Chapman (derek4)

Task 6

For this task we are asked to predict the outcome of a restaurant's health inspection. We are given specified training and test data consisting of basic metrics (e.g. avg star rating) as well as reviews tied to an individual restaurant.

Setup

The training and testing data split is decidedly unbalanced. There are almost 23 testing points for every training point: 546 training points and 12,753 testing points. In the training data the labels themselves are split evenly between the two classes (0 = 273, 1 = 273). For our data a 1 indicates that a restaurant failed their health inspection while a zero indicates that they passed. We are warned that the testing data is not balanced between the classes. However in general the data itself seems to be fairly well balanced. We see in figure 1 that the distribution of the negative polarity scores looks very similar between the training and testing data (with the notable exception of -2 on the training data).

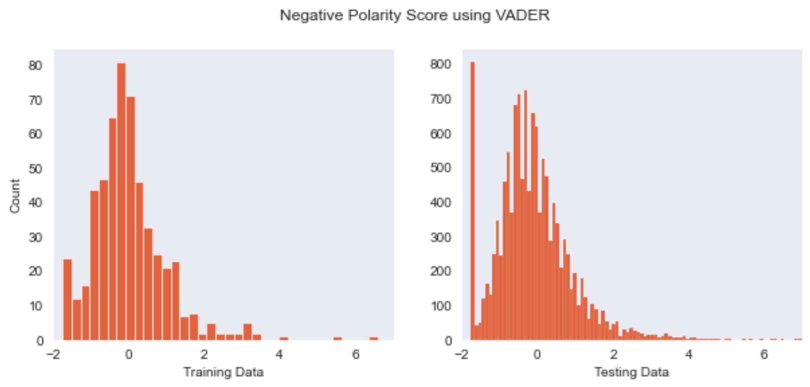

Figure 1

The average review length shows a similar picture [figure 2]. Between these two quick inspections of the data we can feel comfortable saying that the training and testing data itself is approximately similar.

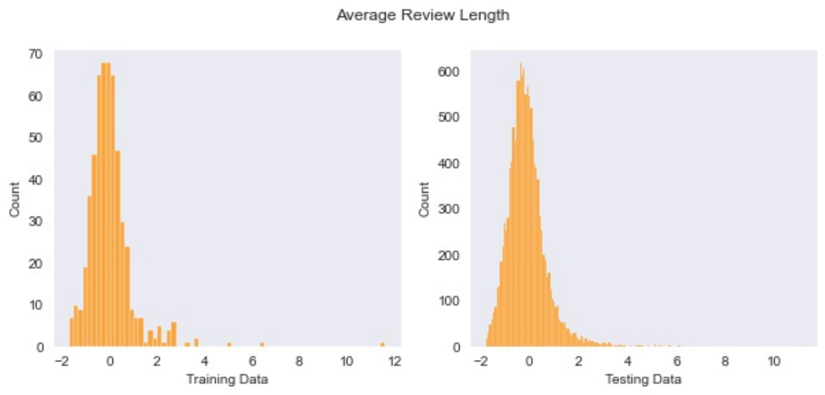

Figure 2

Features

I create three additional features to the ones provided to us. First is the VADER Sentiment Analysis score. This gives us 4 individual features to use later with the classifiers: negative, neutral, positive, and compound scores. I also create a simple Average Review Length feature based on the given number of records and the overall length of the concatenated reviews.

The third feature I create is Doc2Vec. I begin by tokenizing the concatenated reviews and removing stop words. I also drop any word or token that is less than 2 characters long. After tokenizing the list of reviews they are turned into 'tagged documents' in preparation for the model building. My Doc2Vec model uses the default parameters except for a few. Minimum count is set to 2 and epochs (training iterations) is increased to 50. Additional training iterations were needed to get the model to converge to a decent representation. I tried to tune the vector size over a number of runs. A vector size of 25 seemed to work the best. A larger vector (such as 50) actually reduced the overall accuracy of the model. I think with only ~550 training instances having a 50-dimension space in which to put the restaurants allowed for too much space between items. With a mostly empty hyperspace the distance between individual data points becomes increasingly similar, negating the 'discriminating power' of the Doc2Vec embedding. Additional exploration is needed to determine if tuning the various parameters would be helpful, but unfortunately the processing time for a single end-to-end run of Doc2Vec takes an extremely long time limiting the amount of experimentation possible in my limited time.

Data Preparation

This dataset needed some preparation to get it 'ML' ready. First, I use one-hot encoding on the zipcodes. My initial run using raw zipcodes gave a valuable reminder in how much a ML model relies on properly prepared inputs. The classifier treated the zipcodes as an ordered discrete value meaning a higher zipcode was interpreted as being 'larger'. Properly prepping the feature into a one-hot encoded representations solved this issue. I then turned each restaurant's individual Cuisine list into a multi-hot encoded representation. After this I drop the no longer needed columns (Zipcodes, Cuisines, and Restaurants). Finally, I aligned the columns between the train and test data sets and created missing columns in the testing dataset filling them with zeros.

My last step in prepping the data was standardizing specific columns. I scaled the given numerical columns (number of reviews, average ratings, etc.) and also scaled the VADER scores. Although the VADER scores were already in tight ranges of [0, 1] or [-1, 1] I choose to scale them as they weren't very spread out. By scaling them it gives a larger deviation for that feature making it more useful from a model building perspective.

Classifiers

I trained two classifiers, a simple Linear Regression classifier and a Random Forest Classifier. The LR classifier uses all of the defaults except that I again increase the training iterations, this time to 300. Without increasing this the model failed to converge. This model seems to strike a good balance of bias and variance. My 5th full run achieved an overall F1 score of 0.738.

The Random Forest Classifier was built with all defaults except that the number of trees was increased to 250. Unfortunately the RF seems to have overfit to the data leading to a perfect F1 score. However it did provide the added benefit of giving us a feature importance calculation.

Feature Importance

Of the resulting 159 features the table below shows the top 5 overall features and then the top 5 features that are not one of the Doc2Vec components. Interestingly the Doc2Vec components were by far the most important features. You have to go all the way to 17th place to find a feature that wasn't one of the Doc2Vec dimensions. The paper I reviewed for the paper presentation also noted that Doc2Vec seems to be able to capture 'latent' information in unstructured text in a very useful way.

Top 5 Features Overall
Doc2Vec 6
Doc2Vec 22
Doc2Vec 18
Doc2Vec 10
Doc2Vec 3

Top 5 Features other than Doc2Vec
Number of Reviews
Negative Polarity Score
Average Review Length
Positive Polarity Score
Compound Sentiment Score

Results

So on to the million dollar question: what kind of restaurants are least likely to pass their health inspection? (Major caveat that this is based solely on the limited training data as we do not have labels for the testing data to confirm any suspicions.). The cuisines types most associated with failing their health inspection are: Fast Food, Chinese, and Vietnamese restaurants.

F1 Scores for Complete End-to-End Runs

	F1 Score
1st run: no cuisines, basic features, no standardization	0.536
2nd run: one hot encoding zipcodes, multi-hot encoding cuisines, scaled all other features	0.699
3rd run: 50 vector Doc2Vec	0.696
4th run: 25 Vector Doc2Vec	Failed to Converge
5th run: with 300 iterations and warm start	0.738

Scores use average="macro"