Stat 154 Final Project - Yelp! Reviews

Alexander Lee - 24236094, Regina Chan - 24752146, Graham Denevan - 24396733

2017-05-05

1 The Problem and the Data

Yelp! provided a large dataset of information about particular businesses, and asked to predict the average star rating of a business based on its reviews and other attributes of the business. We worked with a reduced dataset, which only included 2510 businesses from ten different cities in the United States. We were provided information about user reviews, business information, users, business check-ins, and tips. User reviews were reviews of the different restaurants posted by users, and included the review itself, the date the review was posted, and the number of each of the responses to the review ("useful", "funny", or "cool"). The business dataset provided data about the restaurants, including its location, categories, hours of operation, and the number of reviews it had. The dataset of users provided data about the user profile, particularly the number of "useful"/"funny"/"cool" votes the user cast, how many compliments the user received on his or her various posts, and the average of the star ratings the user gave to businesses. The business check-in dataset provided data on times when users checked-in (via Yelp!) to each of the businesses. The tips dataset provided data on tips posted by users regarding each business.

The datasets we focused on were the data about user reviews and business information, as these seemed to have the most obvious link to a restaurant's final star rating; reviews are direct because it is a user's direct rating of a restaurant, and business attributes are good predictors of a restaurant's rating because these are general attributes that are often common across different restaurants, and are independent of a restaurant's food, theme, and traits that differentiate it from other businesses. The dataset of users did not seem useful for the analysis we wanted to conduct, as we would have to reconstruct average business ratings based on reviews of individual users, and would thus have to predict ratings based on users who posted reviews of similar restaurants. The check-in data also seemed unhelpful, as check-ins are not only a high-variance measure of a restaurant's rating (since even lower-rated restaurants need foot traffic) but are also difficult to parse through. The tips dataset seemed useful, as tips are similar to reviews at first glance; upon further inspection of the tips data, however, we discovered most of the tips were inconsistently useful, some providing some commentary on the business but many were just descriptions of their orders or short and vague descriptions of their experience at the restaurant.

From these datasets we wanted to find out: what words were correlated to higher or lower reviews from users? Could the given business attributes be predictive of the average rating of a business?

2 Model Selection

The Kaggle competition measures its entries by their root mean squared error. Thus, regressive models are expected to perform better than classification models. Regressive models can generate predictions over a continuous interval; over 1 to 5 stars in this case, along with all fractions between them. However, classification models do not do this: they only predict integer values, because the star response is treated as a factor. This means that for any misclassification the error of that prediction is larger, leading to a higher RMSE; when a regressive model "misclassifies" a prediction, the error of the response is going to be lower because the regressive model does not rigidly predict integer values, leading to a lower RMSE overall. Classification models have potential, but are likely to not perform as well as regression models; but perhaps by combining these classification predictions with other models a more accurate prediction can be attained.

2.1 Attributes

As a preliminary exploration of the data, we looked at how well business attributes (the "attributes" column of the business datasets) predicted a restaurant's star rating. Parsing out various True/False attributes, we ran a simple linear model using these features and some of the general features provided in the business dataset: city, number of reviews, longitude/latitude, and state. This submission had a 0.8007 training RMSE, and garnered a 0.75829 RMSE score on Kaggle. Such a high RMSE is to be expected, as there are many more factors than a business's general attributes that determine how much people like or dislike it. We chose to use a simple linear model on this small feature space just to see how well it worked; if there were a strong linear relationship between these factors and a restaurants star rating we would not have to look at a more complex model. Because the RMSE was so high, we decided to look at other models that would theoretically have better predictive results. We looked to the review data because it would provide many more features from which to predict the star rating.

2.2 Natural Language Processing

From there we worked with the review text and attempted to use NLP techniques to predict a review's rating based on its content. We used the base code Raj provided to clean the reviews and create a feature matrix of the most common words used in reviews. The process gave us a matrix of a little over 800 counts of the most common words in all of the reviews, which would become predictors of a reviews star rating. The linear model fit by Rajs code had a training error of 0.7937; however, due to the nature of the corpus development we could not fit the model to the test set.

We then decided to try boosting, since our model had a lot of residual error. Therefore, a boosted model seemed to be the first step to better fitting the

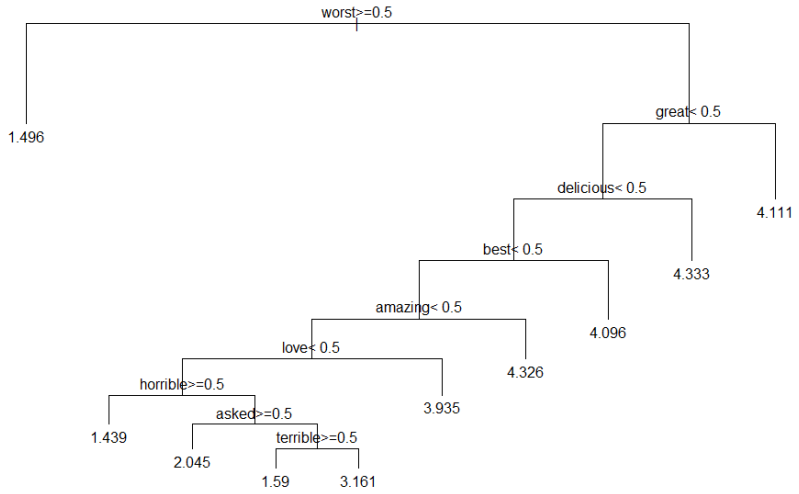

Figure 1: The CART model tree.

response. We also figured XGBoost has been performing well in other Kaggle competitions, and it should perform better than the linear model because it would build it up from weaker models. We ran an XGBoost with various max depths and rounds; they all ended up giving similar training RMSE values, and the final model we ran and submitted was an XGBoost model with a max depth of 10 (for the sake of runtime) and 20 rounds. This only had a 0.9434 training RMSE, and had a 0.81 RMSE score on Kaggle.

During this time we also tried to fit an SVM to the data, but it also performed poorly, as we expected from a linear classifier. We gave more thought to classification methods, and decided that a multi-class classification model would be the most appropriate for star-rating data, but did not know how to implement it and could not find any helpful resources online.

We concluded that our generally poor models were likely the result of having too large a feature space, so we tried to reduce the dimensionality of our predictors. We tried to run a LASSO and PCA to hopefully reduce our feature space to only the most predictive words, but neither yielded helpful results; the LASSO performed poorly, and the PCA did not reduce the feature space very meaningfully so we didnt continue with it.

We finally decided to test tree-based methods, since random forest generally performs fairly well; however, as we were heading into the last days of the competition we first fit a CART model to get an idea of the potential runtime. It had a training RMSE of 1.2068, which was the worst model we had composed

so far; however it did restrict our feature space quite a bit (as shown in Figure 1), and it would be interesting to pursue a simple model based only on the features described here, but we did not think of it at the time.

We attributed the poor RMSE to the simplicity of the model and the fact that the CART model only gives a limited number of response predictions (which would set a standard for the potential RMSE the model would have), and figured a random forest would perform better because it would ensemble many trees together, and so we ran the model of 100 trees with a max tree depth of 25 and ended up with a final Kaggle RMSE of 0.50527, which was the best value we could come up with.

From there we wanted to try to combine several of these models, averaging the predicted stars across all the models; however, we immediately ran into issues. For the CART model and the simple linear model we found that the test set we had parsed using Raj's code was not in the same format and therefore the predict() function could not come up with a prediction; we were unable to figure out how to parse the test data properly so it had the same columns as the training data, so we had to abandon them for the ensemble. With only two proper predictors we felt it would not be significantly advantageous to combine them, so we abandoned the ensembling altogether. Another point of interest was that all of the training data was supposedly rounded to half a star, but all of the review ratings were rated in full-stars; if the Kaggle test set contained any reviews which gave halves of a star in its rating, any model's accuracy would be thrown off.

3 Conclusion

Our best model ultimately was our random forest model. A random forest is an ensemble method which aims to combine decision trees, which are weak classifiers, into an ensemble of trees, which is a strong classifier. We were able to specify parameters such as the number of trees, the number of features to include, etc. The random forest chooses some random subset of m variables at each split with the goal of finding a variable which optimizes the split. The predictor variable that provides the best split is used to do the binary split on that node. At the next node, choose another subset of m variables. The process continues until we have the number of trees specified.

Some of the strengths of random forest are the fact that it decorrelates the trees and implicitly performs feature selection and gives us values of feature importance. In terms of accuracy, it is the most effective method and outperforms all other models we have tried, as expected.

The primary improvement we could make to our base random forest model is to better tune the depth of the trees and to increase the number of trees, since having more nodes would increase the potential variance of the responses (since decision trees can only take on a discrete number of values and is thus susceptible to larger systematic error) and having more trees would allow for better ensembling and thus better predictors.

To better improve the model further we could continue the ensembling we were trying to do by combining multiple models. The primary limitation of this research was essentially learning NLP techniques, which all of us had no prior experience with; if we were able to spend less time working on learning and understanding the NLP techniques (even the "bag of words" provided to us) we could have spent more time trying to improve our feature space, tuning the hyperparameters of our models, or even just running more models (especially those that take a long time, like random forest or even a basic neural net) to better narrow down our predictive results.

As it is, our random forest model does not tell us anything about the features that best predict a restaurant's average star rating: we only have a model which, given a review, will output a predicted rating for that review. We were not able to develop an importance graph, so we have no idea what words or features are most important for predicting the rating of a review. Thus, while we are able to somewhat decently predict a review's rating to within half a star we do not know what factors make a prediction this accurate.

4 GitHub Link and Kaggle Accounts

https://github.com/alexgeraldlee/stat154-finalproj

(Note: only Alexander Lee's code is provided here. Graham Denevan did not send his code before observing the Shabbat.)

Kaggle Accounts used:
Alexander Lee - ALEXANDERGERALDLEE
Regina Chan - REGINACHAN
Graham Denevan - gdenevan

5 Afterthoughts

Alexander worked on most of the side models and wrote up most of the report. Graham made the random forest but didn't send his code; he also did not contribute to any of the report because he had to observe Shabbat, and did not provide his training error for the model nor the importance of the variables in the model. Regina wrote most of the conclusion.