STAT 154 Final Project Report

Yakub Akhmerov: 26882680, Sho Kawano: 23475352, Lydia Maher: 24774122

May 5, 2017

1 Introduction

This report illustrates the analysis of Yelp reviews and using them to predict the star rating of businesses. The idea behind this is to use one data set and learning from it to create a model. This model will be used to predict values, the equation for one of these business is a linear model which conventionally looks like the following, for p variables and n businesses:

$$\hat{y}_i = \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + ... + \hat{\beta}_p x_p, \forall i = 1, ..., n \tag{1}$$

For a bit of a description of what is going on here, the left side of the equation is the output, dependent variable. It is the response of a collection of attributes known as variables. In our case, this value is the predicted star rating of a business. These variables which help predict this star rating are all held together by the addition operator, which is shown on the right side of the equation is the addition of several unknown variables, precisely n of them. In our context, every x represents a variable from our data, examples of this in our context could be average amount of time a business is open a week, whether they accept credit cards, how many times a person says "disgusting" in their review, etc. These variables could be ones that we were given in our uncleaned data-set, or they could be ones that were created through analysis. Many of the variables were created through splitting of existing variables, the specifics of which will be discussed later. The reason for the hat on y and on the coefficients in front of the x's, the betas, is because this is a fitted model. The coefficients, betas, are estimated through calculus to create multiples of the variable values and help predict the response variable. The premise of which is obviously having the data split into training and testing and use the training data to predict what the output variable will be for the testing data. The training data was given to us in a few different tables. The most efficient option for us was to combine them all into one matrix to get an X matrix. Pre-processing was motivated since to be good and ready for analysis, the values in the data frame should be quantitative values.

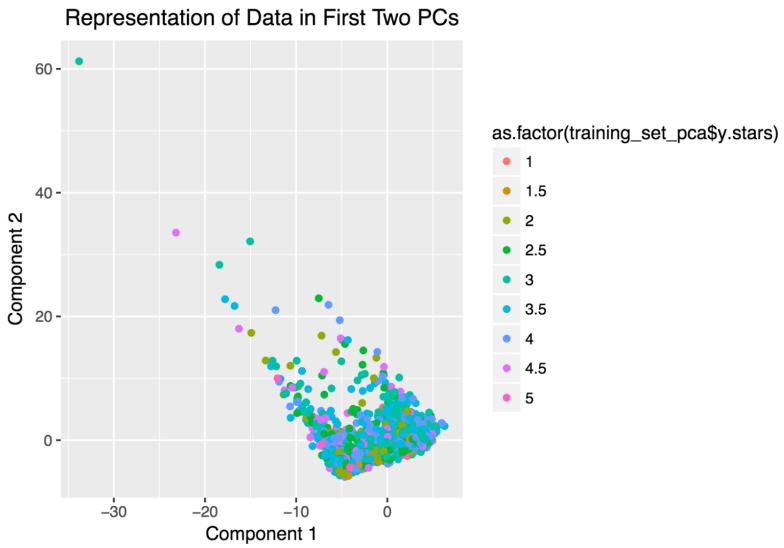
2 Pre-processing

The initial step of the analysis was the pre-processing. Regarding the training set of the reviews, each row was a review by a different user, with a variable called business_id specifying which business had the review written for it, so the specific business was specified in that way. We started by examining the testing data sets and the values of their variables. In data analysis, it is crucial to have values in a quantitative format in order to fit models since the estimation is through calculus so values are required. Therefore, character strings would not be an adequate when fitting models. Clearly since one of the main components of our analysis were actual reviews which are written in text, so we had to do some playing around with the data to get some proper analysis on the reviews done. First off, many texts reviews had stop words such as: it, the, but, that, etc. which don't provide much meaningful information for our purposes and we looked to take them out. It was done by parsing through the vocabulary of all the words and see which ones occurred the most. Originally, we sought to use 100 common phrases for the dividing up of words. Using python script of natural language processing, seen in github, we could find the most common words in the texts. Those 100 words were then changed to variables and the values each row took was ——how many times the review had that word—-, doing NLP takes a set of text and creates them as variables where each row takes values of how often that value appears in the review. This procedure is known as one-hot-encoding.As most common phrases can be a bit of a simple metric, we decided to also look at term frequency inverse document frequency (tfidf) scores. Tfidf looks at the frequency of a word, but in relation to the importance of the term to the entire corpus. This allows us to look at words which are likely to be more influential in reviews. Each entry for those values was a list. The objective was to enlist all the lists of all these entries and create them into variables to analyze them. The business reviews data-set had for starters, the hours variables were difficult because every company had their unique set of hours that were unlike any of the other ones so one-hot-encoding would cause extremely sparse data since the likelihood of different business having the exact same hours is rather low. This motivated us to split the data into time frames, with the times being those which were most popular, i.e. occurred the most through text vectorization. Upon completion of dividing up the values in each row, it was time to combine the data-sets. Our goal was to design an X matrix which had all our features in it for predictions. Our original matrix had 549 features in it. The criteria to doing this was with threshold of how often strings (variables) would appear in whichever context we were using it in. For the review data-sets, the threshold was whether the word appeared in at least 11646 cases (at least 10%). For the attributes variable in the business training data-set, the cutoff was whether it appeared in at least 250 cases (again 10%), and for the dates, whether the hours open was in 100 business (5%). This shrunk the features rather well and brought us down to 110 variables. Which was still a rather large number of features. Though the actual fitting of the model will be discussed later, the specific amount of variables pertains to it so it will be

mentioned briefly. Naturally, what made the most sense was that there were some values with NA in them and the fitting was tossing it out. A quick eye test of the R function complete cases showed there were some cases where the rows values were not complete and had missing values. Naturally the most effective way to preserve the true values of the cases was to remove the variables which had such values. The process by which we filtered out the missing values started with using the function sapply on the columns of the testing dataset (testing has the NAs) to create a dataset with columns that had missing values. The colnames of this dataset was set in a character string. The columns were then extracted from the original testing dataset and removed to bring the variables count to 228 for testing and 229 for the training since the training obviously includes the response variable to be trained on. Finally, the tedious yet highly necessary process of preprocessing the data was finished and we had our dataset ready for analysis.

3 EDA

The next step was exploratory data analysis. We began by performing PCA on the dataset, to see how sparse it was and thus what models would be most helpful in prediction. Although the scree plot had a fairly sharp drop-off (which normally indicates that PCA is a useful technique for the given dataset), 10 principal components were still required to explain even 50% of the variation of the data. Therefore, although our dataset was somewhat sparse and the number of variables could be reduced, it does not seem that PCA regression would be the best model to use for our prediction.



Nevertheless, it is interesting to look at the projection of the original dataset

in the first two Principal Components as a means of further understanding relationships in the data. Doing this allows us to see that there is a concentrated cluster of reviews to the bottom right, with two sprawling streams of outliers to the right and left of this cluster. This suggests that there are a lot of fairly similar reviews, but that in the not-so-similar reviews, there is a lot of variance. Particularly for higher star ratings, there tends to be a lot of variation (we such much less pink values in the center of the cluster and these colors correlate to star ratings of 4.5 and 5). This makes sense as we would expect businesses which are either really good or really bad to have quite different attributes -it wouldnt make sense if ALL restaurants described as fantastic were also Mexican restaurants. Examining these patterns in a 3D scatterplot of the first three components led to similar results. We also looked at what attributes contribute most to the variation in these components. We found that restaurants not labelled as romantic, intimate, upscale, touristy, classy or hipster lead to the most variation in reviews. On the other hand, restaurants described as pretty, great, with i love/more/out or reviewed with similar amounts of stars, tended to be more similar to each other (in other words, they account for less variation).

Finally, we briefly explored our response variable of star ratings across businesses. The star ratings have a mode of 3.5, with the majority of star ratings being between 3 and 4. Generally, the star ratings are skewed slightly towards the upper values of y. Looking at this same information in a boxplot, we can see that the whiskers do not extend below 1.5 stars. Moreover, reviews of 1 star are marked as outliers.

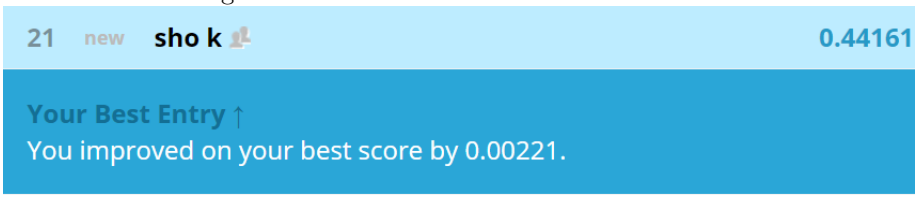
4 Model Fitting

The following step was the model fitting. Through the help of EDA, we decided to use ordinary least squares regression, lasso regression, ridge regression, neural nets, random forests, and PCA regression. Given the magnitude of this data, it was thought necessary to run the fitting on other servers which proved to be efficient. For OLS regression, we fit a model with the lm() function in R and saved the output as a variable and predicted it against the testing data and saved those as variables to get the predictions. It was rather simple saving it due to the simplistic nature of the functions calls in R. Ridge regression was then done and we took a different approach for Ridge regression since it introduces a penalty on the fitting of the coefficients. Essentially, we wanted to train the models by cross validation to find the optimal penalty by examining the best_tune coefficient The value was then saved and using the output coefficients to find the best alpha and lambda parameters. The model was then saved and used to predict the output for the testing data. The next step was a similar technique, lasso. The technique was obviously pretty much identical and we used the optimal model to predict on the testing data. Following our 3 different regression techniques was a different type of model fitting procedure, random forests. The create of random forest was conveniently another R function called randomForest which we used make a random forest, with the typical procedure

of training the models and find the optimal cutoff points. Neural nets was the next step in fitting a model and the procedure was similar to the other ones in finding optimal parameters, etc. Ultimately the models successly fit every several times attempting to run, with the final values being as follows: the RMSE was 0.7536913, 0.4652794, 0.493723, 0.4773863, 0.4577699, and 0.4664328 respectively for PCA, Ridge, Lasso, OLS, Random forests, and Neural Nets and for R-squared is was 0.01394737, 0.6223548, 0.5890142, 0.6059391, 0.3173587, and .5588307, also respectively.

5 Conclusion

In summary, the report showed some interesting findings. The analysis was certainly a good looking glass into industry. It taught about the importance of using data and working with it. Pre-processing data turned out to be the most important part of the report, since the presentation of the data is essentially how it communicates with the procedures of analysis. If a data-set is not in correct format to process, it will mess everything up, especially in our case since we are using the data to predict values on another data-set. It was a lot of trial and error to get a correct data-set with all the values in correct order. The next part, the EDA, helped give insight into which procedures were best to use for creating models, examining outliers, comparing variances, etc. In the actual fitting of the model, it was a lot of run time due to the size of the training data-set. There was a lot of learning to be done to take advantage of UC Berkleley's statistics department's wireless servers. Creating the models helped also in the pre-processing part, to see where the fitting failed, and having that knowledge helps go back and pick up the fallen pieces. To the actual predictions which were calculated, the best indicator was obviously kaggle. This proved to be the most difficult part of the project, since there was no true y's to compare the predictions to. This is a checker of how good your model is so you can compare to the true values of the variables. In addition, only 2 submission were allowed a day, so one had to be make sure the submission they were doing was their true one. That gave extra pressure to make sure the submission made was the right one. Ultimately, submitting was a big help in choosing the correct model. We decided upon creating a combination model of the 2 best predictors, Ridge regression and Neural nets and average the values of them. It ended up being the best model which gave a misclassification of .44161 :



Nonetheless, the report was very enlightening and informing and will be of large assistance for those who decide to end up going into industry.