Final Report

Team 26: Dipak Krishnan, Santiago Roa

May 3rd, 2019

Contents

1 Introduction

The main dataset we are working with in this project has data on Airbnb listings in Seattle. Through analyzing this data, we are trying to fulfill two main objectives. First, we are trying to determine whether several host and housing quality measurements have an effect on the price of a given listing. Second, we are attempting to determine whether these same measurements have an effect on the review score of the given listing. The first task is a regression based task and the second is a classification based solution. Our tasks have two different datasets, of which the price dataset has 5200 observations and 25 variables, including our response variable *price*. Our review dataset has 4043 observations and 26 variables, including our response variable *review_scores_rating*. Initially, there were 48 NA values in the price dataset and 34 NA values in the review dataset. These NA variables were entirely contained in 2 columns, *price* and *cleaning_fee*. We treated the NA values by converting them to 0, meaning that there was no price or cleaning fee for a given listing.

2 Exploration

We begin by taking a look at the different neighborhoods in Seattle. The table below summarizes the top and bottom five average prices of listings in each neighborhood and its proportion over both datasets. We see that about 18% of the listings alone are located in Downtown Seattle. On average, the price per listing there is about 60 dollars more expensive than the overall average of about 147 dollars. Note that in Queen Anne, a notoriously expensive area of Seattle, the average price is about 30 dollars more expensive than the overall average. This difference in price hints at the importance of location for an Airbnb listing.

	mean price	proportion
Downtown	204.23477	17.616961
Queen Anne	184.88353	6.954582
Cascade	173.61775	4.260357
Magnolia	153.17117	1.946791
Capitol Hill	144.67793	11.398992
University District	114.34861	2.839785
Northgate	105.79103	2.220165
Beacon Hill	104.43276	4.169707
Lake City	102.29285	1.678889

	mean price	proportion
Delridge	89.65243	2.940082

Each listing in the dataset included its unique latitude and longitude coordinates. To make the latitude and longitude measurements more useful and to improve the feature set, we tabulated the latitude/longitude coordinates of five Seattle places of interest: downtown, the Space Needle, the Gum Wall, Pike Place Market, and the Great Wheel. We wrote a script that calculated, for each listing, the distance between the listing and each of the five important locations in Seattle. This provided context to our task because it established how well-located certain listings were in comparison to others.

More specifically, for both the regression and classification task, predicting *price* and classifying *review_scores_rating* in the best way required first looking at the data and evaluating the feature set. Both datasets contained the same feature set with the review dataset containing an extra variable - our response *review_scores_rating* for the classification task.

We continued by looking for potentially problematic features in the dataset. In the price dataset, there were two: *amenities* and *room_type*. The *amenities* variable, for each listing, lists the specific amenities that the listing has such as wifi access, TV availability and air conditioning. The *room_type* variable lists the type of the room offered by a given listing. These two features presented probelmatic due to the number of levels of the factor variables.

Each index in the *amenities* column contained a dictionary containing that specific listing's amenities. Logically, we decided to test if the number of amenities for a given listing would affect its price instead of the amenities themselves. We parsed the *amenities* column and produced a new feature *num_amenities* that reported the total number of amenities for each listing. We dropped the original amenities column from our analysis, and dropped the room type column as well from our analysis, as the ordering of its levels presented problems in the model creation process. We kept all other variables that were initially in the dataset in some form in our analysis.

Finally, multiple features in the dataset required recoding. For instance, we refactored the *host_is_superhost*, *host_identity_verified*, and *instant_bookable* variables that were originally true/false variables coded as "t" and "f" into 0/1 variables with 1 coded as true. The *host_response_rate* variable was a percentage in the original dataset, and was changed into a proportion. Two variables: *host_response_time* and *cancellation_policy* were refactored into categorical variables with numbers representing levels of factors instead of words. Our response variable *price* and the *cleaning_fee* variable contained the "$" character before their values - this character was removed and the predictors were converted to numeric variables.

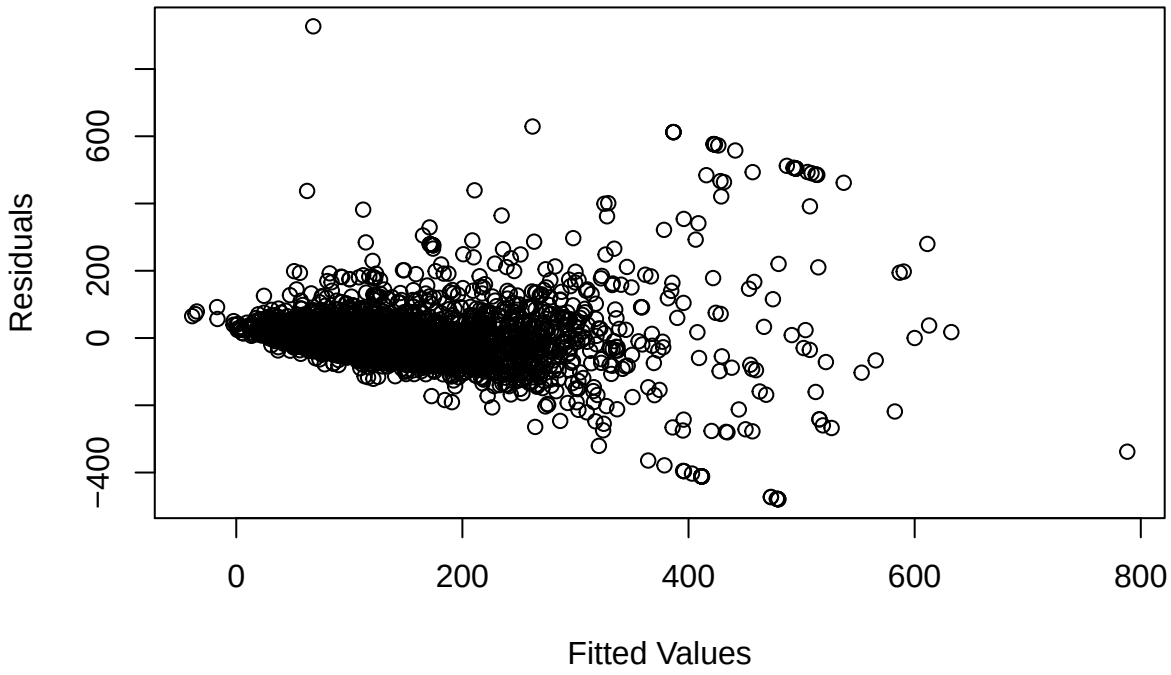
2.1 Regression Model Selection

At this point, after processing and cleaning the data, we had a feature set that we felt good about. To move forward with the regression task, we knew that some variables had linear correlation with our response variable *price*, namely *cleaning_fee* and *beds*. It seemed that fitting a multiple linear regression model was a logical place to start as a baseline technique for prediction.

To begin, we split our price dataset into a training and test set with 75% of data in the price dataset allocated to training and 25% allocated to a validation test set. We fit a multiple linear regression model, regressing price against all the variables in our final, processed training set.

$$lm(price \sim ., data = price.train) \tag{1}$$

The goodness of the fit was evaluated in two ways: against our regression assumptions and with cross-validated mean squared error. The residuals vs fitted values plot of the fitted regression object, found below, showed a cluster of points around 0 with high outliers, potential influential points, and an uneven distribution. Our baseline linear model was not optimal for the dataset, and we inferred another technique could produce a better fit. The cross-validated error for the linear model using k-fold cross validation was approximately 9188.



With our baseline in hand, we logically assumed that it's possible the fit could be improved by improving the bias-variance tradeoff we were dealing with. To make the model more generalizable and get a sense for which predictors were important, we fit a ridge and lasso model on our training set, where equations 2 and 3 represent ridge and lasso respectively.

$$cv.glmnet(model.matrix(price \sim ., data = price.train), price.train.price, alpha = 0) \tag{2}$$

$$cv.glmnet(model.matrix(price \sim ., data = price.train), price.train.price, alpha = 1) \tag{3}$$

We used cross validation techniques to establish the optimal lambda penalty value for both the ridge and lasso regression models above. Our cross-validated mean squared errors for both ridge and lasso were around 8942 and 8940 respectively which was an improvement on our baseline model. We further explored trees and random forests. Given the data had a mix of categorical and numeric predictors, with some outliers in the data, trees and random forests provide a robust measure that does not lose predictive power to high outliers and non-linearity. We also fit a normal decision tree, and a pruned version of the tree, shown below.

$$tree = rpart(price \sim ., data = price.train) \tag{4}$$

$$pruned.tree = prune(tree, cp = 0.019) \tag{5}$$

From the plot of the decision tree we chose a cp parameter that minimized relative error. The cross validated mean squared errors both the normal decision tree and pruned tree improved upon ridge/lasso and was 6572 and 6999 respectively.

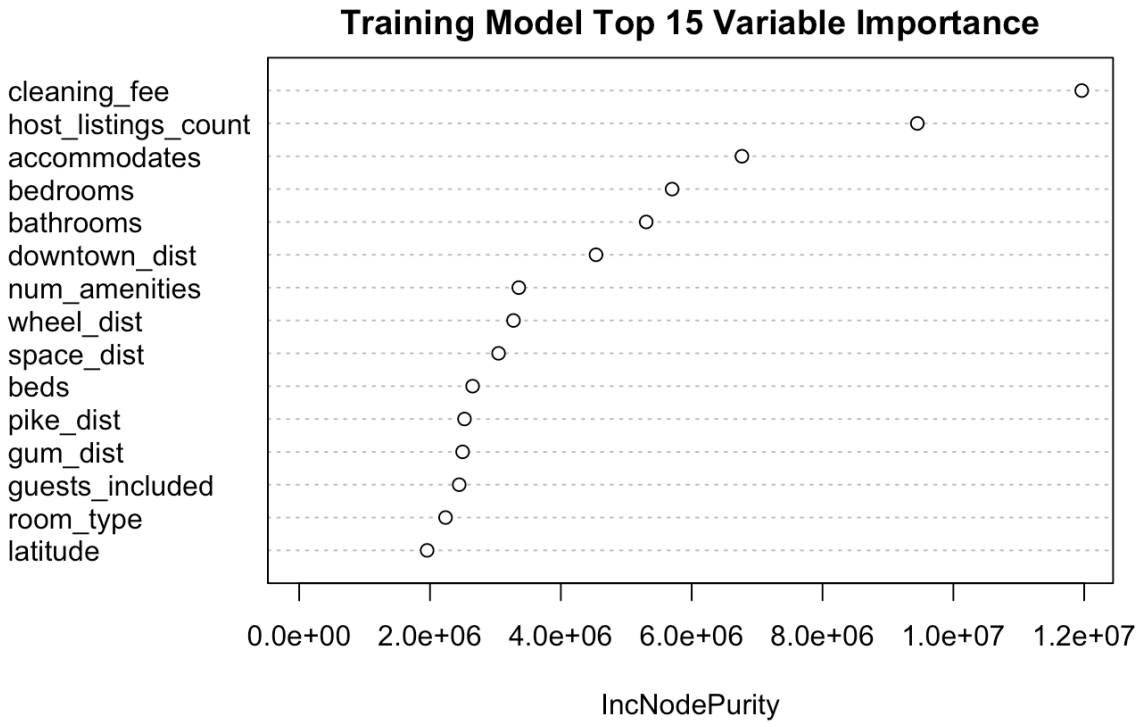
Random forests are an algorithmic improvement on decision trees, due to their greedy search on a subset of features for the best split to make rather than evaluating all features in a decision tree. In addition, random forests retain the strengths of decision trees in working with numeric and categorical predictors and generally produce low bias, moderate variance models. The algorithm averages over mutliple predictions to better predict the response.

We fit a random forest with 500 trees as a baseline on our training set.

$$randomForest(price \sim ., data = price.train, ntree = 500) \tag{6}$$

Random forests had the lowest cross-validated mean squared error around 4005. The predictions that the random forest produced were the closest to the price of the listings in our training set. There are two parameters of interest in the random forest algorithm: the number of trees to grow and the number of variables randomly sampled as candidates for each split. We attempted to tune the random forest to find the best possible number of variables to sample for each split to minimize error, using tuneRF. Tuning the number of trees to grow for a random forest isn't necessary after a certain amount of trees, as the forests stabilize at around 200 trees. Our tuned random forest used the default number of variables to consider for splits. We made our predictions on the test set using the tuned random forest.

After trying various algorithms, we concluded that random forests produced the lowest prediction error without extreme bias or variance, and we chose the random forest model as our final regression model. Regarding the relationship between the predictors in the model and the predictions themselves, an advantage of random forests is that they have a very interpretable variable importance measure that compares the prediction strength of each variable.



The most powerful predictors for price in terms of prediction strength were *cleaning_fee* and *host_listings_count*. Interestingly, the variables we added for location mapping were among the best predictors in the feature set. Our model selection process led us to choose our random forest model as the model that best explains the test data.

2.2 Classification Model Selection

Similar preprocessing was done for the price and review datasets, hence the model selection process was similar for both the regression and classification tasks - albeit with different algorithmic techniques. We split our review dataset into 75% training and 25% test as done in the regression task. The amenities and property_type variables were dropped from the overall dataset as in the previous task. The same location mapping and amenities cleaning added to the regression data feature set was included in the classification data feature set.To get a baseline accuracy, it seemed logical to begin with a naive classifier.

We chose a Naive Bayesian classifier, shown below, as a baseline to learn more about our ability to classify for our response variable review_scores_rating.

$$naiveBayes(review\_scores\_rating \sim ., data = review.train) \tag{7}$$

Our cross-validated misclassification error was 17% as a baseline. Given that our response variable review_scores_rating was split into two classes - 0 and 1 - we thought logistic regression would serve as a good starting point.

The following logistic regression model was fit on the training set, which resulted in a cross-validated misclassification error of 12.8%:

$$glm(review\_scores\_rating \sim ., data = review.train, family = "binomial") \tag{8}$$

The variables that contribute most to the fit were *cancellation_policy*, *num_amenities*, *neighborhood_group*, *host_identity_verified* and *host_is_superhost*. The cleaning of the amenities variable done in preprocessing, on the evidence of the logistic fit, seemed to improve the feature set. However, there seemed to be possible improvements in misclassification accuracy, hence we tried three other classification techniques: support vector machines, boosting, and tree-based random forests.

Support vector machines were deployed in three different ways: utilizing a linear kernel, polynomial kernel and a radial kernel, shown below as equations 9, 10, and 11 respectively. The models were trained on the training set with the same set of predictors used for the naive and logistic models.

$$svm(review\_scores\_rating \sim ., data = review.train, kernel = "linear") \tag{9}$$

$$svm(review\_scores\_rating \sim ., data = review.train, kernel = "polynomial") \tag{10}$$

$$svm(review\_scores\_rating \sim ., data = review.train, kernel = "radial") \tag{11}$$

Without tuning, the three methods all produced cross-validated misclassification error of approximately 12%, improving on the logistic regression fit. Tuning the models for the parameters cost and gamma took an long time to run due to the large feature set - and the best models marginally improved misclassification error.

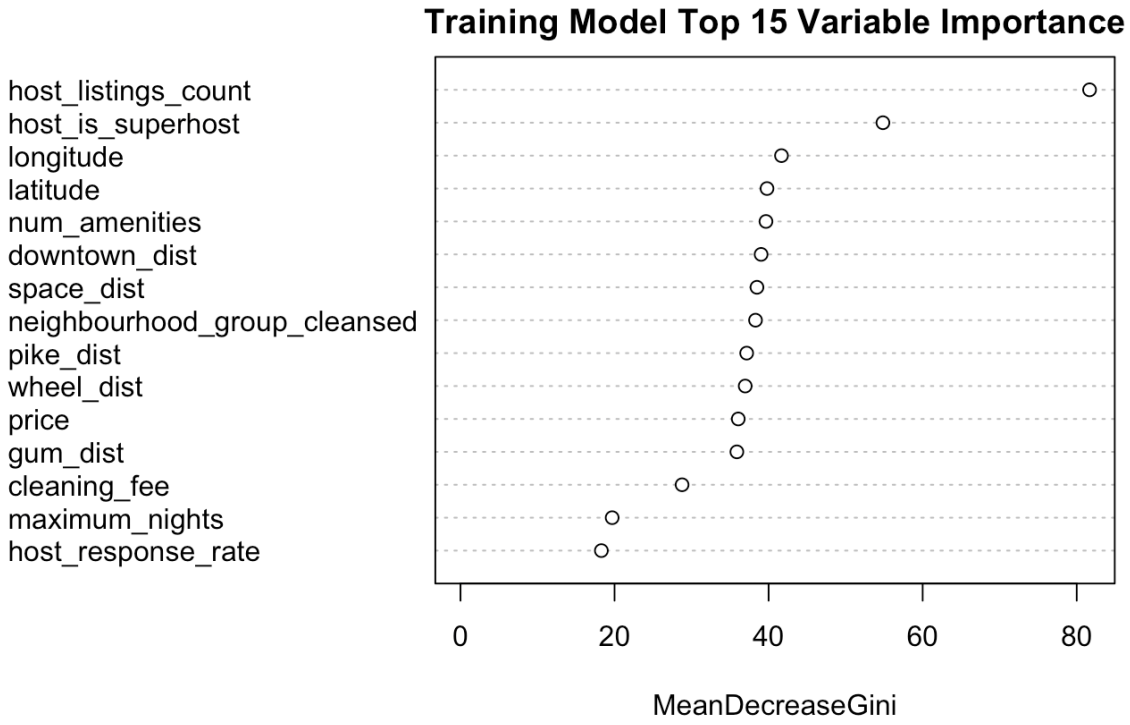
Boosting, like random forests, is an ensemble learner that produces low bias, moderate variance models. We theorized that adaptive boosting would be an algorithmic improvement on all three of the classification techniques used previously due to its additive nature. Utilizing gradient descent to minimize error over a fixed number of trees led us to believe boosting would increase classification accuracy. We fit an adaptive boost model with 100 iterations, using the "Freund" Adaboost algorithm. Cross-validated misclassification error dropped to sub 12% - approximately 11.7%.

$$boosting(review\_scores\_rating \sim ., data = review.train, mfinal = 100, coeflearn = "Freund") \tag{12}$$

Finally, we chose to try random forests for many of the same reasons that we denoted for the regression task and for its flexibility. We fit a random forest with 500 trees on the training data.

$$randomForest(review\_scores\_rating \sim ., data = review.train, ntree = 500, na.action = na.exclude) \tag{13}$$

Initially, the cross-validated misclassification error hovered around where the boosting algorithm was performing, but after tuning the number of variables randomly sampled as candidates for each split, random forests produced the lowest cross-validated misclassification rate of 11.35%. The boosting algorithm and the random forest algorithm produced similar results, but since the random forest classifier was easily interpretable i.e. it was clear to see which variables that were important for improving accuracy, we chose our random forest model as our final model. In a similar fashion to the regression task, we modeled the relationship between the predictors and their relative importance to the fit.



The most impactful features for classifying *review_scores_rating* were *host_listings_count* and *host_is_superhost*. For classifying the desirability of a listing, our engineered features performed very well, but *host_listings_count* surprisingly had twice as much impact on the random forest's model than all the other variables. Hosts with more listings must be more experienced and may have better practices for providing users a more desirable experience. This idea is supported by the fact that *host_is_superhost* is the second most importance variable according to our model.

3 Analysis of Results

The performance, on both the regression and classification task, of our predictor set was better than baseline models on average. That being said, there were some noticable patterns in the missclassified listings that are worth mentioning. To begin with, over 95% of the training points missclassified had "Real Bed" for *bed_type*. Moreover, half of the missclassified points were "Entire home/apt" or "Private room". Based on the importance plots from the random forest models, we saw that the geographical features were heavily relied upon, but characteristics about the perks of the venue itself were not adequately used. This was, in part, a result of how the *amenities* column was preprocessed. Features indicating the presence of desirable ammenities such as a gym, gatekeeper, free breakfast, or even a pool could have been included to provide a more holistic look at the listing.

With more time, there are a number of other steps that can be taken. In particular, the data was skewed and contained high outliers in multiple categories. High outliers existed in several numeric variables in the dataset, specifically *price*, *cleaning_fee*, *maximum_nights* and *accommodates*. An example of a possible outlier that could skew price would be a mansion in an expensive neighborhood that could accommodate a high number of guests, and would have a corresponding high cleaning fee. More time dedicated to filtering the data and making it as granular and useful as possible would surely improve our model building and selection process, and reduce mean squared error and misclassification error respectively.

Another area for improvement is feature engineering and selection. The features that we created to create context for our latitude/longitude coordinates using distance measures, and the data cleaning done on the

amenities column did very well in our final models, and they were some of the largest contributors to the fit and resulting predictions on the test set. To that end, exploring logical correlations could produce a richer feature set. For example, higher priced listings tend to be more desirable, have more amenities and a better location. Exploring the time of the year when listings are most popular could be fruitful. It's also important to note that the data we have is from the host's perspective, including data on the user could further enrich the dataset and possibly provide a more complete picture as to what influences price and review score.