Predicting AirBnB Rental Rates

Trevor Isaacson, Jonathan Olavarria, Jasmine DeMeyer

12/10/2021

Introduction

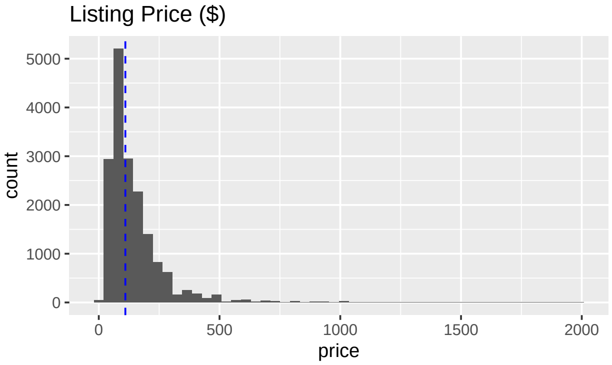
The Data

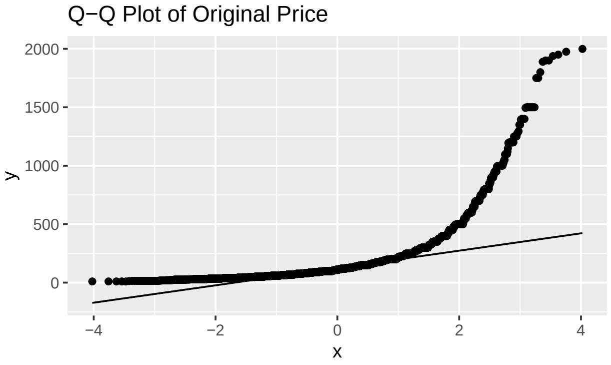
As a company, AirBnB is very open and transparent with the data they collect about their rental properties. They provide data about rental spaces in their system for cities and countries all over the world. Because of this, we were able to find a large dataset on Kaggle with AirBnB listings in major US cities including New York City, Los Angeles, San Francisco and others. The dataset available on Kaggle has over 74,000 entries and was used as a competition a few years ago. For the sake of time and processing, we trimmed our training data to about 17,500 entries and our test data to about 5,000 entries. We did this by taking a random sample of the provided training data. This allowed for easier access and faster processing while maintaining a large amount of data and individual AirBnB listings.

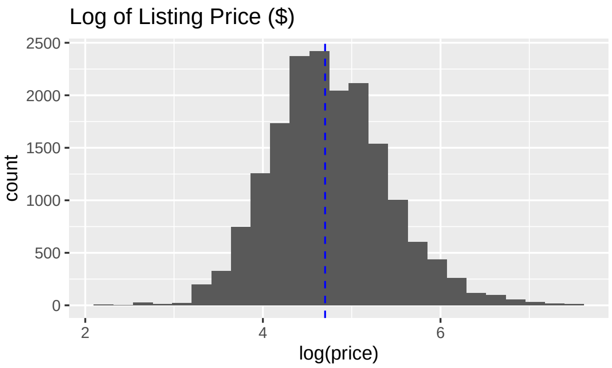
The original dataset contained 30 variables about each listing. Due to high correlations and lack of relevancy, our final dataset consisted of twenty-two variables. Those twenty-two variables can be split into four categories: property, location, host and host reviews.

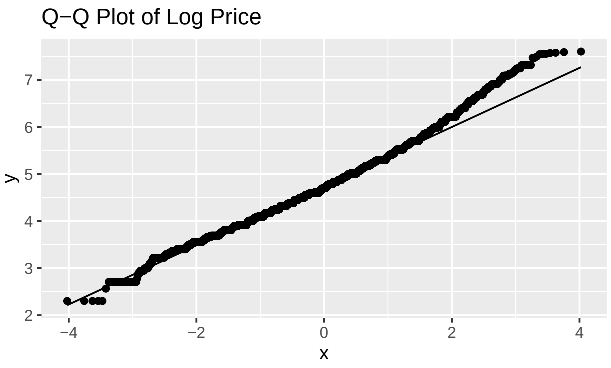
Property includes:

- price: listing price
  - Because the original price data is very heavily skewed, we needed to log transform the prices. As shown in the histogram, we have a very heavy right tail because there are a some listings with very high prices compared to the median price of $110 (blue line). This non-normal shape and distribution is clearly evident in the Q-Q plot. The observations clearly curve away from the line depicting how skewed the distribution is.



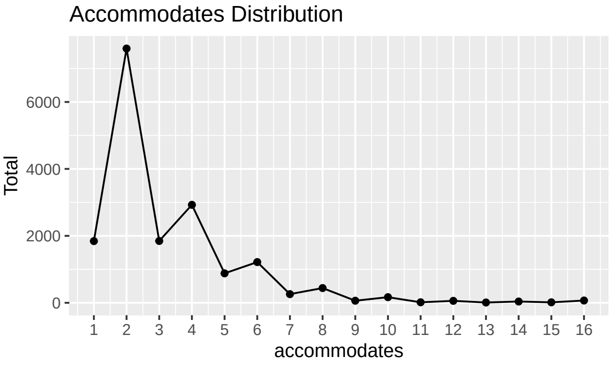


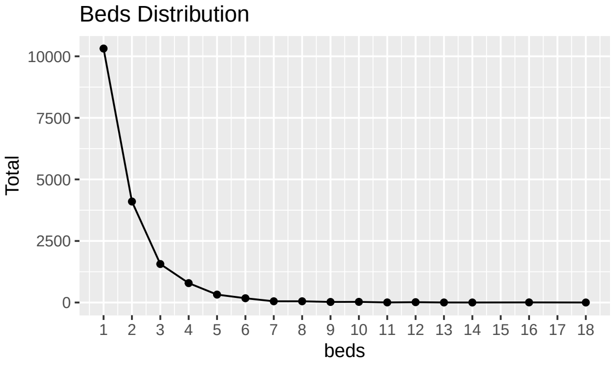


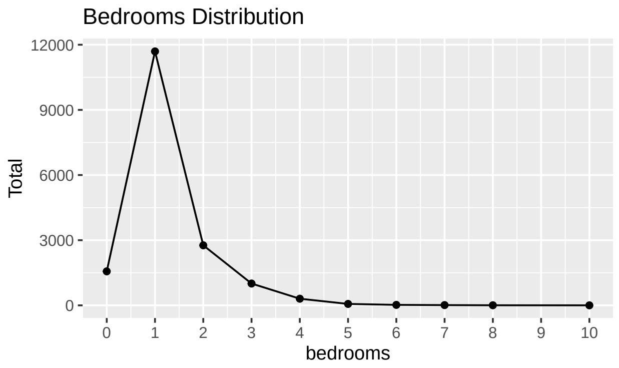


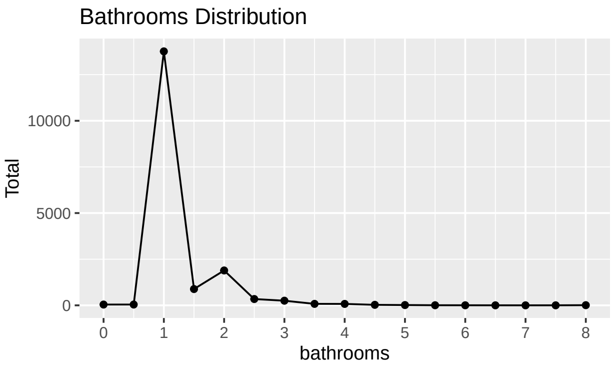
By applying a log transformation, we now have a more normal shaped distribution. Our Q-Q plot shows some evidence of a right tail but this can be expected since the original distribution is very right tailed.

- property_type: defines the type of property listed
  - There are 21 different types ranging from apartments, houses, and condos to boats, cabins, hostels and even castles
- room_type: defines type of rental within the property
  - Includes entire home/apt, private room and shared room
- accommodates: number of people the property can comfortably accommodate
- bedrooms: number of bedrooms within the property
- beds: number of beds within the property
- bed_type: type of bed available
  - This includes a Real Bed, Futon, Pull-out Sofa, Airbed or Couch
  - Only 463 listings have something other than a Real Bed
- bathrooms: number of bathrooms within the property



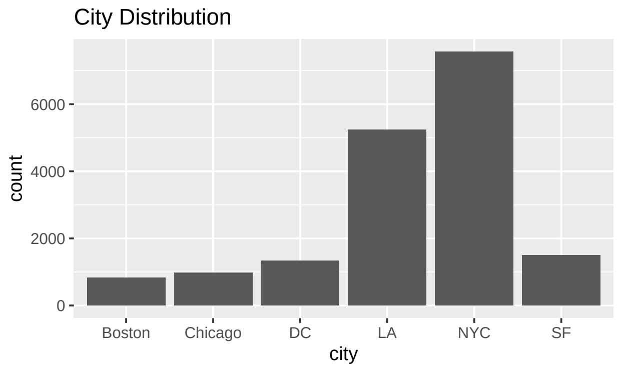


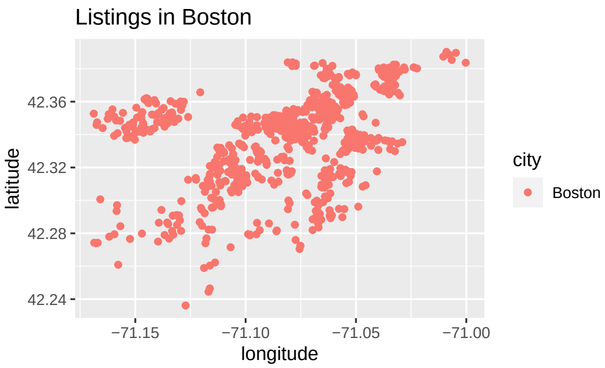




Location includes:

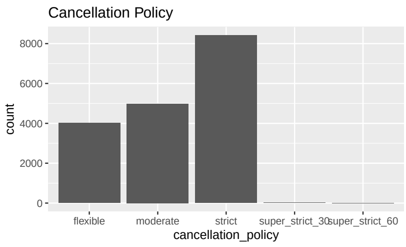
- city: Location of listing
- latitude and longitude: latitude and longitude coordinates of the listing

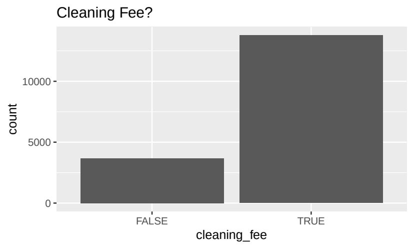


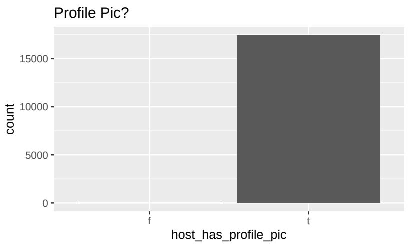


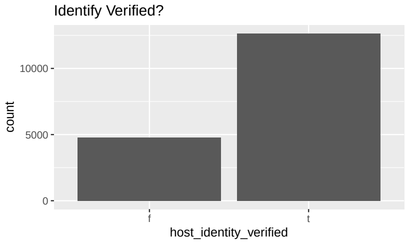
Host includes:

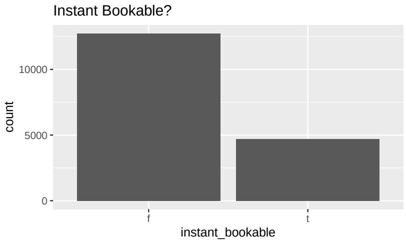
- cancellation_policy: strictness of cancellation policy set by the host
  - Levels include strict, moderate, flexible, super_strict_30 and super_strict_60
- cleaning_fee: TRUE/FALSE determines if host charges a cleaning fee
- host_has_profile_pic: TRUE/FALSE determines if the host has uploaded a picture to their profile
- host_identify_verified: TRUE/FALSE determines if the host's identity has been verified by AirBnB
- instant_bookable: TRUE/FALSE determines if the property can be booked in short notice
- host_response_rate: how often does the host reply to potential clients?











Host Reviews:

- number_of_reviews: Number of reviews the host has received
- review_scores_rating: average review rating for the host and property
- first_review_year: year of the first review
- last_review_year: year of most recent review
- host_since_year: year the property was first listed on AirBnB

As a group, we felt these twenty-two predictors were all relevant and important in helping predict price.

Models

There are have a lot of machine learning methods discussed this past semester and we wanted to incorporate some of our favorites into our research. Thus, we have included linear regression, splines, general additive models, PCR, PLS, trees, bagging, random forests and bagging.

In order to both train and test using the training data file, we needed to split the 17,500 total observations into roughly a 50/50 split. To do this, we took a random sample of 9000 observations and made that the training set. From now on, this random sample with be referred to as the training set. The remaining 7500 observations became our testing set for determining the the performance of each method.

Regression

To begin, we started with a simple multiple linear regression model. We wanted to give ourselves a baseline mean squared error value and because linear regression is the easiest to apply and interpret, we determined this was the best place to start. The model was fit using all twenty-two variables and the training set.

```
## [1] "R^2: 0.632 , Adjusted R^2:  0.63"
```

```
## [1] "MSE of Testing Set:  0.1652"
```

```
## [1] "Leave One Out Cross Validation:  0.1829"
```

```
## [1] "k-fold Cross Validation:  0.3063"
```

As shown in the results above, our linear regression model was able to set a good baseline for future methods with a mean squared error of 0.1652. With our linear regression model, we also applied some cross validation. For Leave One Out Cross Validation, we achieved a mean squared error 0.1829 and applying k-fold cross validation with k = 10, we achieved a mean square error of 0.3063. Clearly, Leave One Out Cross Validation performed better than the k-fold cross validation.

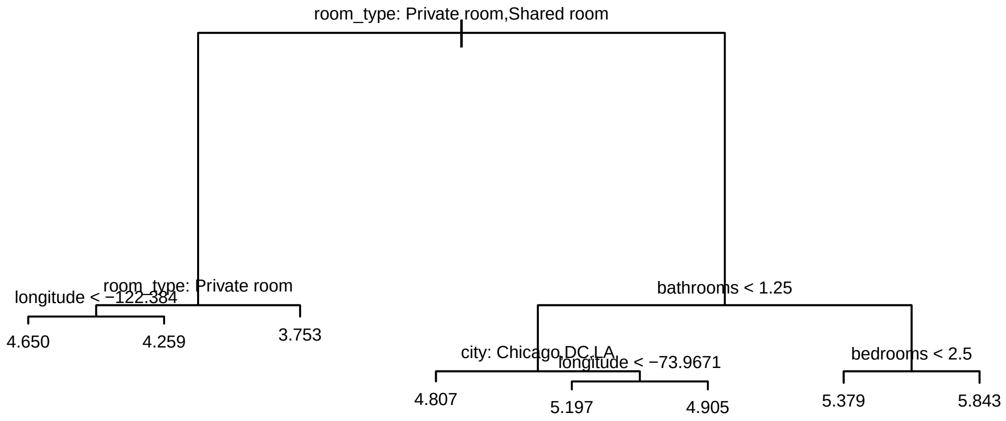
Regression Splines/Generalized Additive Models

PCR and PLS

Trees

The next method we used was decision trees. Fitting the fit using all predictors, we obtained a tree where five variables were actually used in the tree construction. Those variables were room_type, longitude, bathrooms, city and bedrooms. The tree has eight terminal nodes and obtained a test mean squared error of 0.1926. The cv.tree function was used to perform cross-validation to determine the optimal level of tree complexity. The optimal level was 8 and then using the prune.tree function, we attempted to prune our tree to the chosen complexity but found that our original tree was the same as the pruned tree. The table below shows the log prices converted to dollars to help explain the terminal node values. Node 1 is the farthest node on the left.

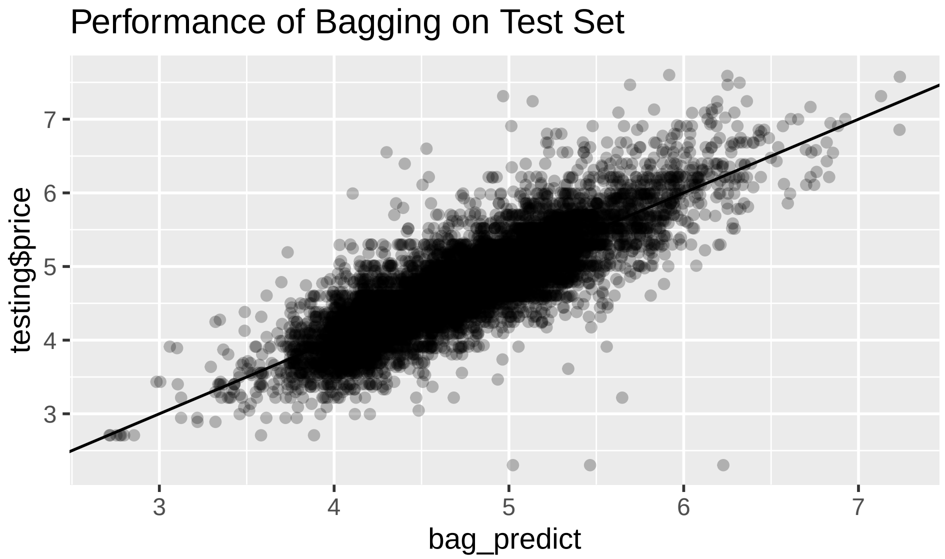
```
## [1] "Test MSE of Tree:  0.1926"
```

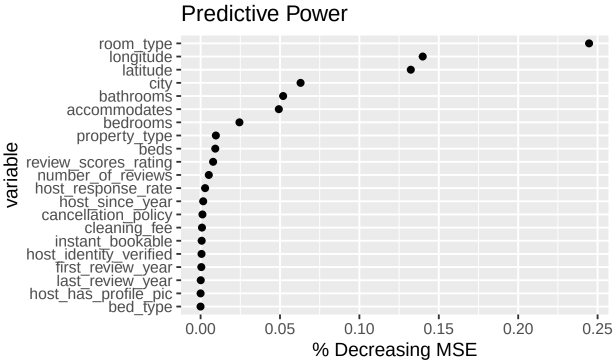


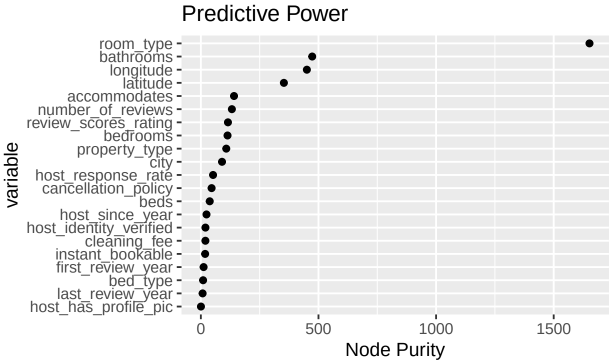
TerminalNode	Log_Price	Price
1	4.650	104.58
2	4.259	70.74
3	3.753	42.65
4	4.807	122.36
5	5.197	180.73
6	4.905	134.96
7	5.379	216.81
8	5.843	344.81

Bagging

```
## [1] "Test MSE of Bagging:  0.1293"
```


Performance of Bagging on Test Set


Predictive Power


Predictive Power

Random Forests

Boosting

Results

City Prediction

References: