Project: AirBnB Pricing Prediction

**Team:**

James R Gearheart  
Danny Zhuang  
Bob Saludo  
Ryan, Wallace

**Final Report**

**Overview:**

Airbnb is a peer-to-peer online marketplace and homestay network enabling people to list or rent short-term lodging in residential properties, with the cost of such accommodation set by the property owner. With thousands of property owners listing their property for short-term lodging, the marketplace for lodging has become very competitive. The success of Airbnb can be attributed to the information, decision power and functionality that it places in the hands of property owners. In order to be competitive in the marketplace, it is essential that property owners have the most up to date information about the housing market. This information must be catered to the specific location, property type and lease date for their property.

The decision of pricing a property on specific dates is a critical decision that every Airbnb client must address. The housing market is volatile. Per Diem rates can have extreme fluctuations from day to day and these fluctuations can be based on several factors such as day of week, month, holidays, local events, competitor’s prices along with hundreds of other possible factors.

The goal of our research is to develop a predictive model that will provide the property owner with a competitive target price on specific dates for their property. This model will empower Airbnb property owners to competitively price their property and provide renters a fair market price.

**Literature Review:**

Since the inception of “shared economy”, there have been dozens of articles and scholarly papers written about Airbnb along with other peer-to-peer homestay networking sites. Although the researchers of this project reviewed several of these papers, we present the detailed overview of two of the most relevant papers for this literature review:

*Predicting Airbnb Listing Prices with Scikit-Learn and Apache Spark*

The online article titled “Predicting Airbnb Listing Prices with Scikit-Learn and Apache Spark” by Nick Amato was published on April 20th, 2016 on the Mapr blog site. This article details the construction of a predictive model using python’s Scikit-Learn package in combination with Apache Spark for performance enhancement. This model leverages the publically available dataset titled “Inside Airbnb” published by Airbnb and focuses on the San Francisco area.

The article walks the reader through the step-by-step process of building and evaluating a predictive model using publically available data. In the first section, the author explains the available data sources and provides python code for accessing and evaluating some core features of the data. This initial analysis demonstrates the raw nature of the data and the need to clean the data. In order to create a concise model, the author limits the data to Airbnb listings in the San Francisco area with one bedroom. This filter captures the majority of the available listings and limits the possible complex interactions involved in a broad study of all listing types.

Once the author limits the data to a select number of features, he cleans the data by replacing missing values for the ‘reviews\_per\_month’ field from NaN to 0. He also cleans the data by filtering out any listings where there are missing values for price, bedrooms and beds. The next step is to recode the categorical variables into dummy variables (values of 1 or 0) by using the ‘get\_dummies’ feature of Scikit-Learn. This technique allows the model to incorporate the categorical data into the model as unique numeric features. An analysis of the scatter plot matrix shows that there is not an issue with correlation that would need to be addressed.

The author uses several predictive models from the Scikit-Learn package to demonstrate the differences in predictive performance by using the median squared error metric to evaluate model fit. The models that were initially developed were: linear regression, ridge and lasso regressions, ElasticNet, Bayesian ridge and Orthogonal Matching Pursuit. A comparison of the model outputs shows that the Bayesian ridge model provides the lowest error from the six candidate models. The error is translated into dollars in order to be consistent with the project goal of predicting prices. The Bayesian ridge model results in an error of $30 - $35 dollars.

The next step is to implement a model tuning algorithm. For this process, the author uses the GridSearchCV function from the Scikit-Learn package. The GridSearchCV function is an ensemble model that explores a large variety of hyperparameters through an exhaustive search process. This process tries all the supplied parameter combinations and uses cross-validation to find the best one. The concern with this approach is that it requires a lot of CPU processing power and time. In order to process the exhaustive grid search efficiently, the author uses Apache Spark to distribute the process across several machines and cores. The author uses the GradientBosstingRegressor predictive model in combination with the GridSearchCV cross-validation technique on the Apache Spark system to achieve an error rate of $21.43.

Conclusion:

This article provided an excellent step-by-step look at developing several predictive models and assessing the accuracy of those models. The author also demonstrated how Apache Spark can be leveraged to increase processing efficiency and allow the full range of exhaustive search methods to be employed in model development.

*Neighborhood and Price Prediction for San Francisco Airbnb Listings*

In ``Neighborhood and Price Prediction for San Francisco Airbnb Listings'', Tang and Sangani explore data from the Inside Airbnb project, containing a complete set of 7,029 listings of properties for rent on Airbnb in San Francisco as of November 2, 2015. For each listing, the data set contains the price per night, text information, such as the name of the listing and a description of the property and host, an image of the property, and a number fields describing quantifiable attributes of the property, such as amenities offered and square-footage. With these data, the authors seek to predict two attributes of a listing, the price per night, and the neighborhood in which the property is located.

The authors choose these metrics for their potential usefulness to Airbnb's platform. In particular, predicting price may allow Airbnb to offer improved suggested pricing for hosts ready to list while predicting neighborhood could provide a first step towards recommending new properties to clients based on where they have stayed in the past. The authors make several interesting decisions about how to approach the tasks of predicting price and neighborhood.

Rather than attempting to predict price as a continuous variable, a binary response variable for price is created that indicates whether the predicted price is above or below the median price in the data set. This method has the advantage of simplifying both the price prediction task and the evaluation of the goodness of the model. However, disadvantages of this approach include that simple linear regression is no longer a suitable option to model the price response variable, and that the results of the prediction are of limited usefulness, especially for the application of suggesting pricing to hosts.

Additionally, in the data cleaning phase, all observations in neighborhoods containing 70 or fewer listings are removed. While this reduces the burden on the neighborhood classifier, it also restricts the model's usefulness to neighborhoods with a relatively high number of listings.

The majority of the work is done in extracting features from the data set in order to prepare the data for modeling. A total of five features are chosen: listing information features, bag of words features, word class features, text sentiment features, and visual features. The authors employ a variety of techniques in order to extract these features. Interestingly, pre-developed packages are used to do the majority of processing for all but the listing information feature. In particular, packages from the Natural Language Toolkit such as Porter-Stemmer seem to be useful for the processing of hand-written data into quantifiable features. Similarly, OpenCV's libraries for extracting SURF features from images appear useful for the processing of the images associated with the listings.

After creating a feature vector for each listing, two support vector machine models are trained to predict each of the response variables. The choice of SVM classifier is not explained; although, given the large number of features, the choice seems reasonable. Nevertheless, alternative methods may be worth investigation. Two major adjustments are needed to improve the models. First, the optimal value of the regularization parameter is experimentally determined by grid search. Second, high variance in the initial model indicates that overfitting is likely. In order to combat this, the authors use the Recursive Feature Elimination functionality in sklearn to reduce the degrees of freedom in the SVM.

Conclusion:

The final models for both price and neighborhood have impressive predictive power. In the test set, prices are categorized to the correct group (either above or below the median) with approximately 81% accuracy, while neighborhoods are categorized with 42% accuracy. While this performance seems good, the authors note that discretization of the predicted price into smaller bins is likely necessary for most applications, and is a promising direction for future work.

**Data Sources:**

The raw data for this project was developed by Murry Cox and made publically available on the [Inside Airbnb](http://data.beta.nyc/dataset/inside-airbnb-data) site.

<http://data.beta.nyc/dataset/inside-airbnb-data>

The data that was employed for this project was contained in three separate files:

Calendar – Provides listing\_id, date, availability, and price. This dataset contains historic information for January 1, 2015, to January 1, 2016.

Reviews – Provides listing\_id, date, reviewer\_id, reviewer\_name, and comments.

Listings – Provides detailed information about the property:

|  |  |  |
| --- | --- | --- |
| id | country | calendar\_updated |
| scrape\_id | latitude | availability\_30 |
| last\_scraped | longitude | availability\_60 |
| name | is\_location\_exact | availability\_90 |
| picture\_url | property\_type | availability\_365 |
| host\_id | room\_type | calendar\_last\_scraped |
| host\_name | accommodates | number\_of\_reviews |
| host\_since | bathrooms | first\_review |
| host\_picture\_url | bedrooms | last\_review |
| street | beds | review\_scores\_rating |
| neighbourhood | bed\_type | review\_scores\_accuracy |
| neighbourhood\_cleansed | square\_feet | review\_scores\_cleanliness |
| city | price | review\_scores\_checkin |
| state | weekly\_price | review\_scores\_communication |
| zipcode | monthly\_price | review\_scores\_location |
| market | minimum\_nights | review\_scores\_value |
| extra\_people | maximum\_nights | host\_listing\_count |
|  |  | guests\_included |

**Formatting**

The data was retrieved from the data.beta.nyc website by zip file download. In each of the three datasets, (listings, calendar, and reviews), the data are well formatted overall in CSV format with one record (listing, listing-date, or review, respectively) per row. Given the varied nature of the data, python’s dataframe tool “Pandas” is a good choice for initial processing, and each of the three datasets is read into its own Pandas data frame. Manual examination of the data shows that the 'id' field in the listings data set corresponds to the 'listing\_id' fields in the reviews and calendar data, so joining on these fields is feasible.

There are occasional formatting irregularities throughout the data that have been fixed. For example, some fields contained excess parentheses or string data types where floats or booleans are more appropriate. The price fields were originally encoded as strings with dollar signs and comma values. These were converted to floats for use in learning models.

**Missing Values**

While the datasets are nearly complete, there are a significant number of missing values. In the listings dataset, roughly 30% of all review scores, the vast majority of the square footage values, and low percentages of other listing characteristics are missing. In the calendar dataset, 28% of the listing dates do not have price information. In the reviews dataset, an insignificant number of comments are missing.

The missing values were handled in various ways to suit the nature of the data. Missing price data is a roadblock because it is inadvisable to attempt to impute missing response variables. Thus, records with missing price data are dropped. For categorical variables, it is imaginable that knowing which listings have descriptors absent is associated with the price (likely in a negative fashion). Therefore, missing categorical values were replaced with a new category 'unspecified'. Missing review values were imputed with the global mean, as the correlation between price and the majority of review metrics is so low that attempting to use regression for imputation would be useless. Likewise, the data are too noisy for KNN to be of use. For other continuous feature variables such as bedrooms, bathrooms, and beds with relatively high correlation with price, missing values were imputed using linear regression.

After all imputation was finished, the only missing values were those of weekly and monthly prices. These missing values are potential response variables and will be dealt with on an individual model basis.

**Size of Data**

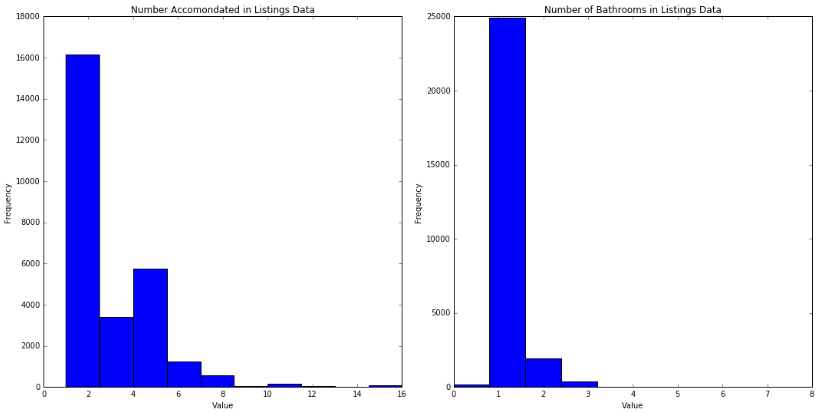
There is a total of about 27,000 listings in the listings data set, each with 52 features. The calendar dataset contains one entry for each day of the year for each of the approximately 27,000 listings. The reviews dataset contains about 28,000 reviews for about 19,000 distinct listings.

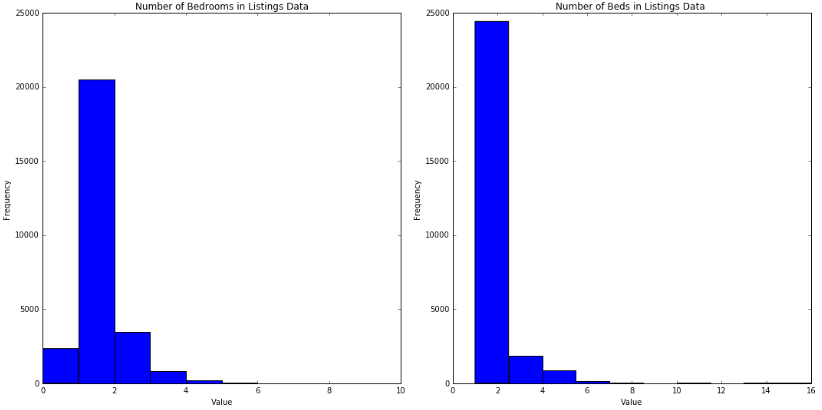
After cleaning, the majority of the data remains. All of the original listings remain in the data; over 70% of the calendar records are useful, and the vast majority of the reviews have complete comments.

**Features and their Distributions**

There are three potential response variables in the data, nightly price, weekly price, and monthly price. Since over half of the weekly and monthly price data are not present, it seems reasonable to focus on predicting nightly price. Nightly prices are right-skewed, with a few very high outliers. The majority of the prices are below 300 dollars per night. The price distribution looks approximately lognormal, so a log transformation of the price variable resulting in a normal distribution of prices may play better with regression techniques.

There are over fifty predictors in the listings dataset, both continuous and categorical. Histograms of the continuous variables of interest and lists of the most frequent categorical variables of interest appear above. Interesting factors to note are that the majority of listings have one bedroom and one bathroom and that reviews are extremely left-skewed.

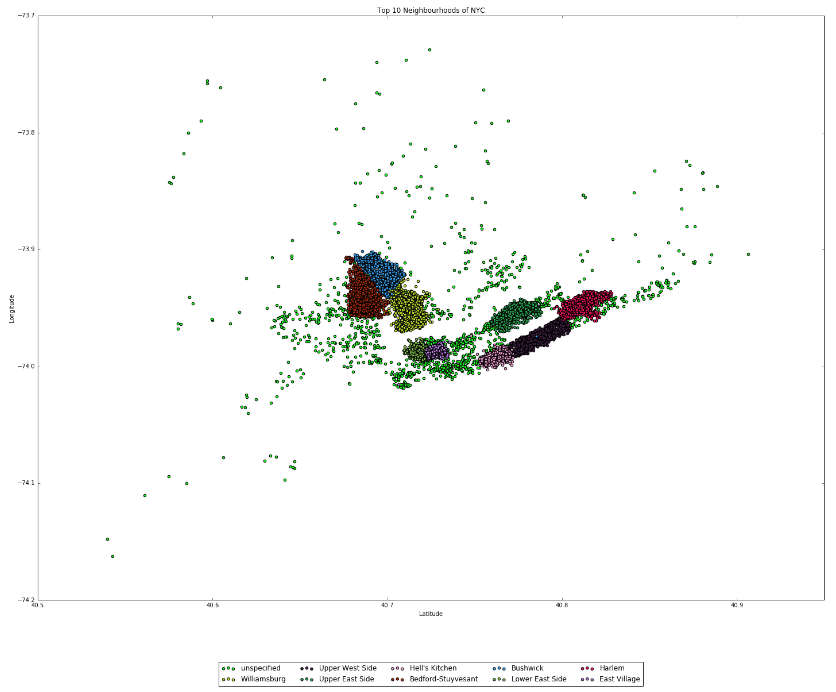




**Visualize the supply/price of Airbnb homes by location**

We notice that in general, the overwhelming majority of rental properties are concentrated in the center of the grid (between 40.6 - 40.9 Latitude, and -73.8 - -74.1 Longitude. Outside of this dense range, we have sparse units in the outskirts of NYC which are predominantly listed under the 'unspecified' category for “neighborhood” and are in the lower end of the pricing spectrum.

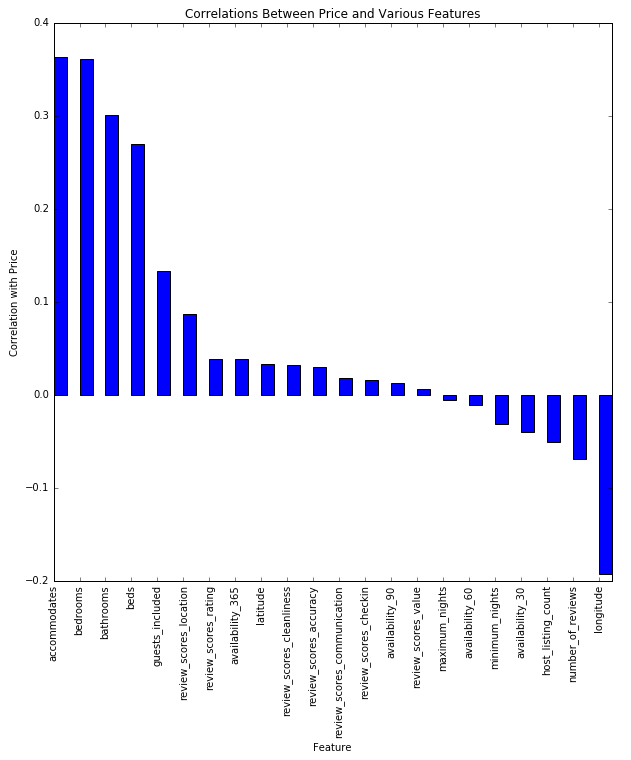
On the whole, the most expensive rentals are in the area around 40.7 Latitude and -74.0 Longitude. As we move outwards in all directions, prices generally become less expensive. One possible way that we could use this observation is to determine the epicenter in terms of high-priced housing, and then calculate the euclidean distance column for all points in the dataset and adding this distance into our model as a predictor. Additionally, we could use the neighborhood predictor variable in our model because the top 10 neighborhoods are each separate and extremely concentrated. We think it is logical to suspect that there will be clear differences between the neighborhoods of NYC, with clustering within the neighborhoods and at least some pricing separation between the neighborhoods. We see this in the average rental prices for each of the 191 neighborhoods in our dataset.



**Correlations between Price and Features**

The Pearson correlations between price and each feature variable of interest are plotted below in decreasing order of correlation. From the chart, we see that the basic physical features of the listings are most strongly linearly related with price: number of individuals accommodated, number of beds and bedrooms, and number of bathrooms have the highest correlation values. The strongest negative linear relationships with price are for the number of listings by the given host, number of reviews, and the longitude of the listing. These results all make intuitive sense. Hosts with many listings may be less attentive to their tenants, tenants may be more inclined to leave a negative review, and knowledge of the demographics of New York supports the observed negative relationship between price and longitude.

While studying these correlation values are useful, they only describe the magnitude of the linear relationship between price and the predictors, so the additional visual analysis is necessary.

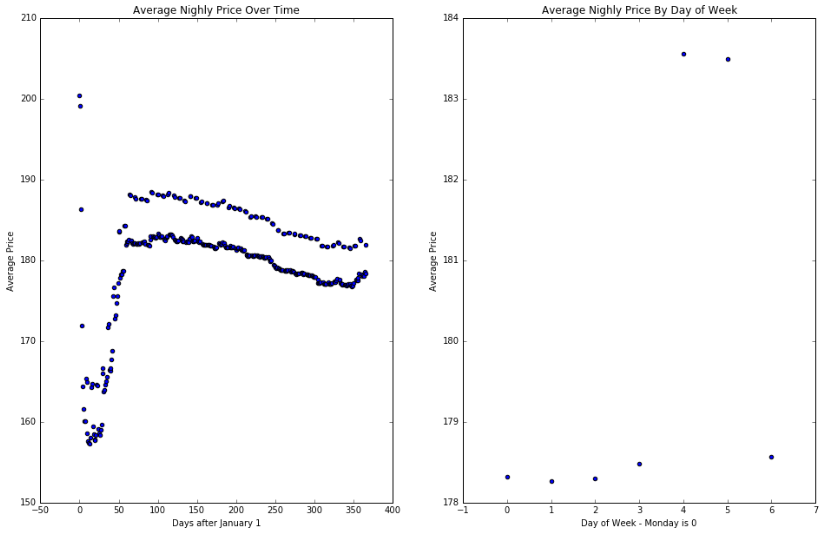


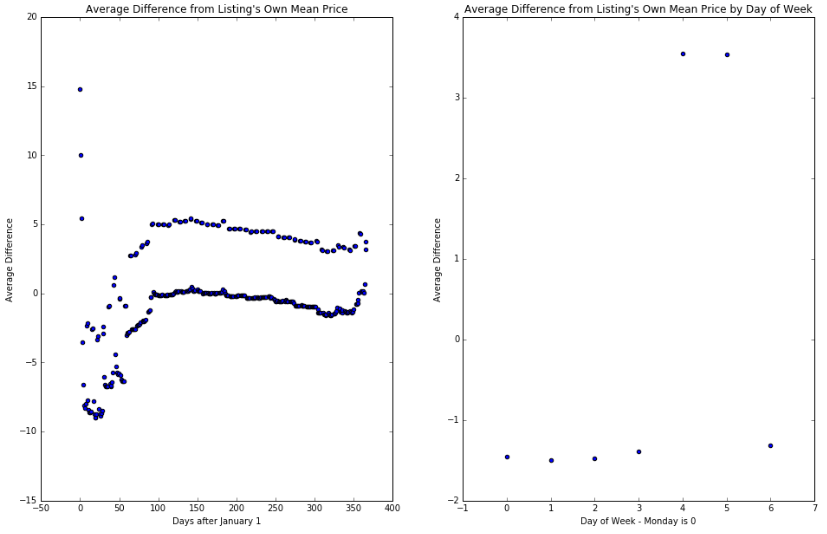
**Relationship between Price and Time**

There are strong and interesting relationships between the prices of listings and both the time of year and the day of the week. Two different sets of statistics were computed to investigate these relationships. First, the simple average price over all the listings was calculated for each day of the year, and for each day of the week. Since this may overlook certain patterns, the differences between the price of a property on each day of the year and its average price throughout the year was also calculated, and averaged across all the properties for each day of the year.

Both sets of metrics show very similar trends. Late in the month of January is very cheap: all of the fifteen dates with the largest negative average difference from mean listing price occurred in the month of January. Most dates are up to $8 cheaper on average. In striking contrast, early January (presumably because of New Years celebrations) are the most expensive, with the first few days averaging over $10 more expensive than usual. The spring and early summer months are also more expensive by about 5 dollars on average for peak times.

The relationship between the day of the week and the average difference in listing price from its mean is also very strong. Weekdays and Sunday are all similarly priced. However, Fridays and Saturdays average $3.50 more expensive than weeknights.





In the modeling stage, it is clear that time (in terms of both date and day of the week) will be an important feature. The interaction between the date and day of the week may also prove helpful.

**Additional Categorical Variables Created**

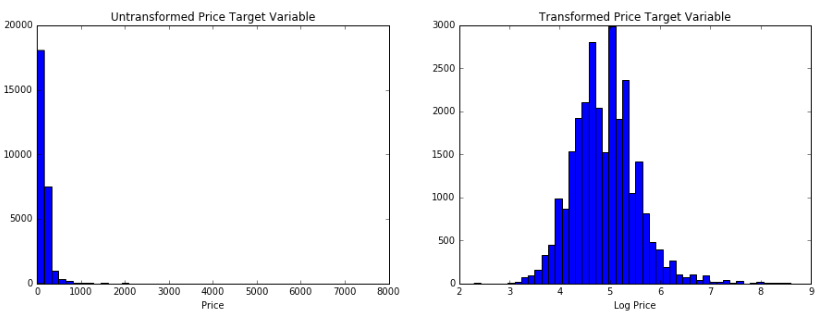
It is clear from the visualization of average price against the day of the year that first three months of the year see lower average prices than the remainder of the year (with the exception of around New Years). Previously, this was incorporated into the model by including a categorical variable indicating whether each observation fell in this slump period. However, it can also be seen that even during the slump, it appears that some months (particularly January) are worse than others. Thus, we also create three new categorical variables for each of January, February, and March to incorporate more time-related nuance into the model.

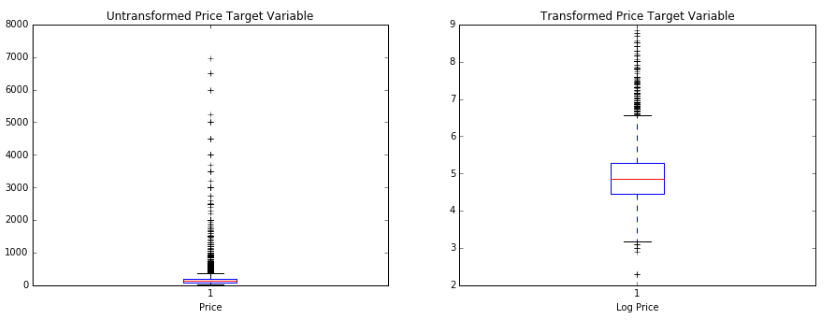
We chose to preserve the quartile representation of zip codes and neighborhoods since we fear that including more specific indicators (such as for each zip code) would lead to drastic overfitting. On examining other approaches for the encoding of these variables, we come to realize that other clustering approaches such as K-means are not suited to this task since we are dealing with categorical data that can not be grouped well in a continuous fashion.

Indicators were also created for each of the property types, each of the room types, and each of the bed types. Since each of these categories has relatively few distinct values, it seems that creating one indicator for each distinct value is less likely to lead to severe overfitting or over-complication of the model. Nevertheless, it will be important to perform regularization to avoid potential overfitting.

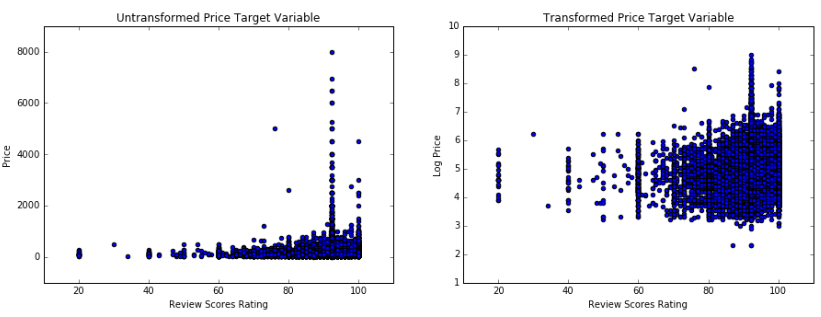
**Analyze Target Variable and need for Transformation**

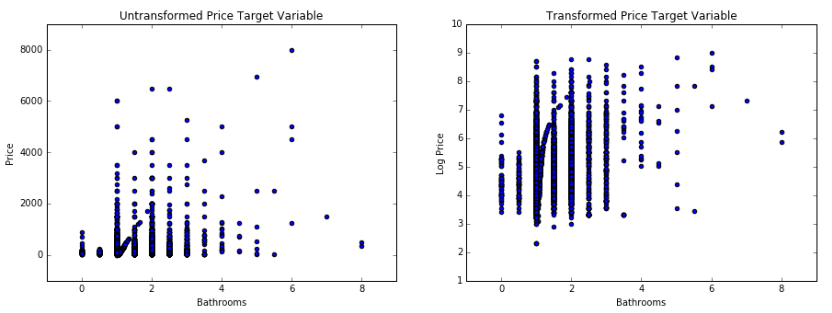
The target variable of "price" was analyzed to determine the variable's skewness. Histograms and boxplots were created to visualize the distribution of the target variable. These visualizations confirmed that price is highly skewed to the right. This indicates that the price variable contains several extreme outliers and the need to transform this variable to ensure that it is normally distributed and prepared for predictive modeling.

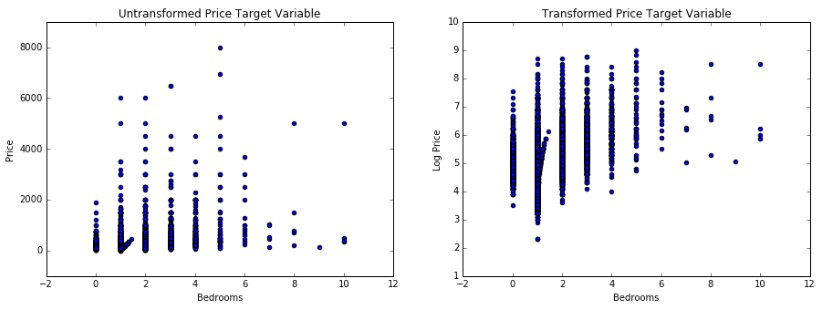




The "price" variable was log transformed and new visualizations were created to demonstrate the normal distribution of the log-transformed target variable. Boxplots were also created to demonstrate that the transformation contained the outliers. Several additional visualizations were created that plotted the log transformed target variable against several of the possible predictors.







**Transformations and Interactions**

Each of the continuous variables is plotted against the response variable, the log of price, in order to determine if there are significant non-linear relationships between the variables. From the plots, we see that a quadratic term for longitude, host listing count, and the value rating produce a more linear relationship with log price, so these transformations are generated.

Since an exhaustive search of possible interaction terms to include would roughly involve  interactions to examine, this approach is not feasible computationally. Instead, we rely on our intuitive understanding of the relationships between the variables to anticipate which interaction terms would be meaningful. Using this approach, we include interactions for both weekend and holiday with both number of bedrooms and number of guests accommodated. This interaction seems potentially reasonable because one could imagine that in particularly high demand times, larger or more accommodating properties are especially valuable.

**General Modeling Approach and Assumptions Possibly Violated**

We choose to model our data by including a single observation for each listing for each day it is offered. This is potentially problematic for two reasons. Firstly, it violates OLS assumption that each observation is independent, because it is obvious that the listings for a given property on different days are highly dependent. Secondly, this may bias our regressions such that the relationships present in properties that are offered more regularly are more highly weighted.

Despite these potential violations, we maintain this data representation because it fully encodes the all of the information we have, both related to the properties, the time of year, and the relationship between properties and the time of year (which we exploit in creating some interaction variables, as described above). We feel that it is reasonable to potentially violate the i.i.d. assumption since the size of our data set is so large (millions of listings) compared to the size of the sets of dependent observations (maximum size of 365). Additionally, we feel that biasing the results towards properties that are more frequently offered is not necessarily detrimental, since we expect these types of properties to be offered more frequently in the future as well, leading to a potentially better performance on future test sets.

**Methodology:**

For our baseline predictive model, we fit a linear regression model using the relevant features to predict price. However, the main hurdle in this task was massaging/manipulating each of the raw features so that we could fit our model while achieving interpretability and computational efficiency.

To incorporate the days of the week and holidays, we used the information that we gained from analyzing the average difference from the listing’s own mean price in visualizing how prices changed throughout the year. Ultimately, we found that the real increase in per-night rental costs came on Friday, Saturday, and around the New Years’ holidays. Additionally, we found that non-holiday dates in January and February showed the lowest prices, which we deem as “slump” dates. Thus, we made categorical variables to denote the day of the week (weekend or no weekend), holiday (3 days around New Years’), and slump dates (January and February dates that aren’t around New Years’).

Because the categorical variables of neighborhood and zip code have over 200 distinct values each, one-hot encoding would produce to far too many variables for a linear regression model – leading to long computational time and a small chance of over-fitting. To solve this, we created four categorical variables for each of these features that separate the neighborhoods and zip codes by price into quartiles. Each quartile is its own categorical variables (e.g. most expensive 25% of neighborhoods, least expensive 25% of zip codes). Thus, we move away from trying to account for individual neighborhoods such as “Tribeca” separately and instead choose to analyze the most expensive neighborhoods together. While we lose some degree of granularity, we believe that what is gained in computational efficiency and streamlined interpretability is well worth it.

The final modeling dataset consisted of the following continuous and categorical variables:

**Continuous variables:**

['latitude', 'longitude', 'accommodates', 'bathrooms', 'bedrooms', 'beds', 'guests\_included', 'minimum\_nights', 'maximum\_nights', 'availability\_30', 'availability\_60', 'availability\_90', 'availability\_365', 'number\_of\_reviews', 'review\_scores\_rating', 'review\_scores\_accuracy', 'review\_scores\_cleanliness', 'review\_scores\_checkin', 'review\_scores\_communication', 'review\_scores\_location', 'review\_scores\_value', 'host\_listing\_count', 'weekend\_x\_bedrooms', 'weekend\_x\_accommodates', 'holiday\_x\_bedrooms', 'holiday\_x\_accommodates', 'longitude\_sq', 'review\_scores\_value\_sq', 'host\_listing\_count\_sq']

**Categorical variables:**

['zipcode\_q1', 'zipcode\_q2', 'zipcode\_q3', 'zipcode\_q4', 'neighborhood\_q1', 'neighborhood\_q2', 'neighborhood\_q3', 'neighborhood\_q4', 'weekend', 'holiday', 'slump', 'jan', 'feb', 'march', 'Apartment', 'Bed & Breakfast', 'Boat', 'Cabin', 'Camper/RV', 'Castle', 'Cave', 'Chalet', 'Dorm', 'Earth House', 'House', 'Hut', 'Lighthouse', 'Loft', 'Other', 'Tent', 'Treehouse', 'Villa', 'f', 't', 'Entire home/apt', 'Private room', 'Shared room', 'Airbed', 'Couch', 'Futon', 'Pull-out Sofa', 'Real Bed']

**Results:**

Baseline Model

The modeling dataset was split into train and test datasets where 75% of the data was used for training. The baseline simple linear regression model was developed on the training dataset consisting of 7.2 million records with 72 predictors. This model resulted in an R^2 of 0.63 in the test dataset. These initial baseline results supported our assumptions that the Airbnb prices can be effectively modeled.

Competitive Models

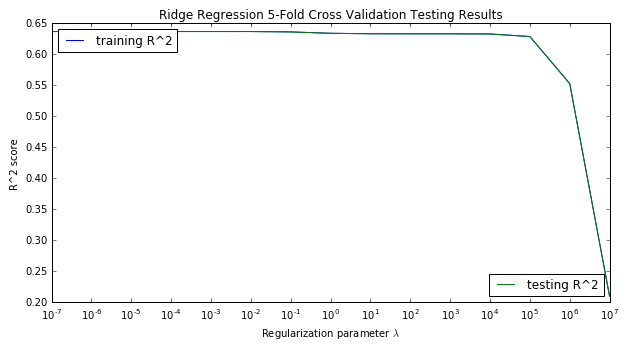
Due to the size of the dataset, the simple linear regression model required a significant amount of computational resources. In order to produce competitive models that would be even more computationally expensive, we decided to reduce the size of the modeling dataset. We created a random sample of one million records (~14% of the total modeling dataset). In order to tune the parameters, we preformed a 5-fold cross validation methodology on this sampled dataset.

Ridge Regression

Due to the number of predictors that the modeling dataset contains, it is vital that we develop regularization models to reduce the number of predictors. A list of tuning parameters for the Ridge and LASSO regression models were selected:

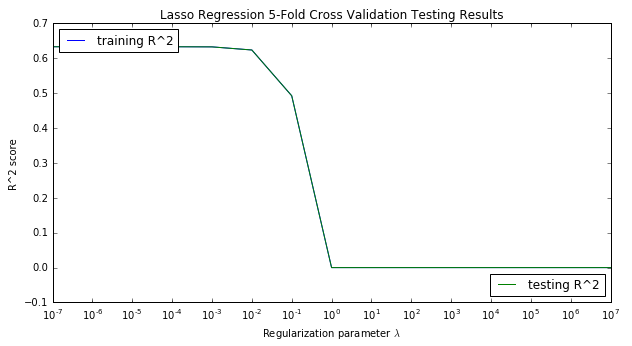
lambda\_lst = [1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1e0, 1e1, 1e2, 1e3, 1e4, 1e5, 1e6, 1e7]

The Ridge regression methodology was applied to the 5-fold cross validation datasets with the tuning parameters shown above. This methodology resulted in an average R^2 of 0.64 on the test datasets. Although this was a slight improvement on the baseline model, further modeling efforts were deemed necessary.



LASSO Regression

The LASSO regression method was applied to the 5-fold cross validation datasets with the same tuning parameters that we employed for the Ridge regression model. This method was much more sensitive to the effects of the tuning parameters. The tuning parameters 10-7 through 10-2 retained an average R^2 of 0.63 on the test datasets. However, as the tuning parameters increased the model score drastically dropped to an R^2 of 0.0.



The Ridge and the LASSO regularization models did not provide a significant lift over the baseline linear regression model. The final model that was developed for predicting Airbnb prices was a Random Forest Regression model. This model is also the most computationally expensive.

Random Forest Regression

Due to the computational expense of the Random Forest Regression model, the original modeling dataset of 7.2 million records was randomly sampled to create a dataset of 10 thousand records for an initial approach with this method. The Random Forest Regression model was created by applying an AutoKFold Random Forest Regression method to the 10K observation subset of the modeling dataset. This methodology applied several different levels to the input model criteria of “number of trees”, “tree depth” and “number of features”. The final number of possible combinations of the model criteria was 96:

Number of Trees: 10, 40, 70

Tree Max Depth: 2, 3, 4, 5, 6, 7, 8, 9

Number of Features: 5, 25, 45, 65

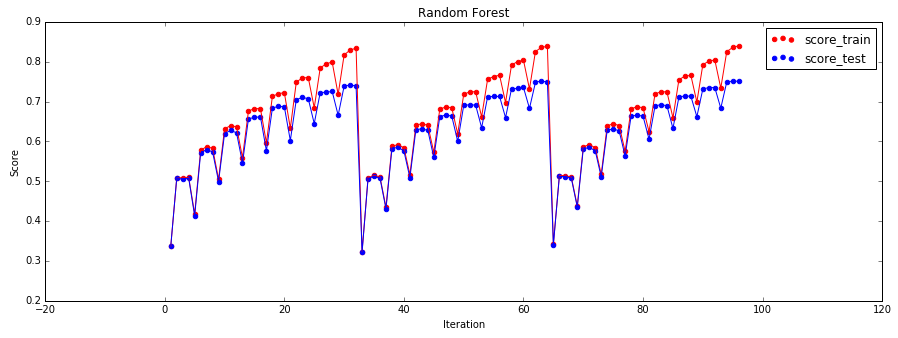
# Random forest regression with 10,000 observations

n\_trees = np.arange(10, 80, 30)

depths = np.arange(2, 10)

features = np.arange(5, 70, 20)

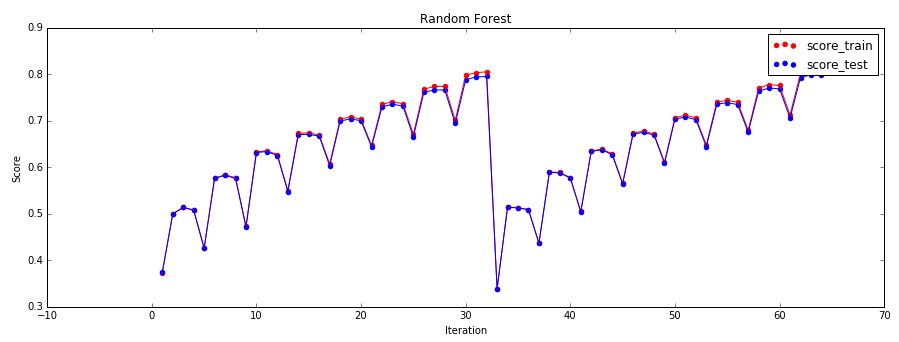
The Random Forest Regression model results in a max R^2 score of 0.80 on the test dataset. This score is the result of applying each of the combinations of possible model criteria to the Random Forest Regression algorithm on a 5-fold cross validation dataset for 100 iterations.



This methodology provided a significant lift over the baseline linear regression model. Due to the positive results of this approach, we increased the number of observations from 10 thousand to 100 thousand in order to determine the impact of the sampling size on the model’s predictive power.

The graphic above demonstrates that the number of trees doesn't significantly change the testing score based on the step-wise nature of the testing scores. It is the maximum depth and maximum number of features that drive increased test scores so we tune exclusively for those and set the number of trees=40. We tune on 100,000 observations for run-time reasons.

The original modeling dataset of 7.2 million records and 72 predictors was randomly sampled to create a dataset of 100 thousand observations. A Random Forest Regression model was applied to this dataset with the same model criteria as the previous Random Forest Regression model. This model also resulted in a max R^2 of 0.80 on the test dataset created by a 5-fold cross validation methodology.



It is recommended that for a random forest model the maximum number of features should be p/3 where p is the number of predictors. Also, random forests cannot overfit the data with respect to the number of trees, so we should first select the best maximum depth and maximum number of features for some "big enough" value for number of trees, and then re-tune with fixed maximum depth and maximum features while varying the number of trees in order to find the minimum value for number of trees that gives us our best testing scores.

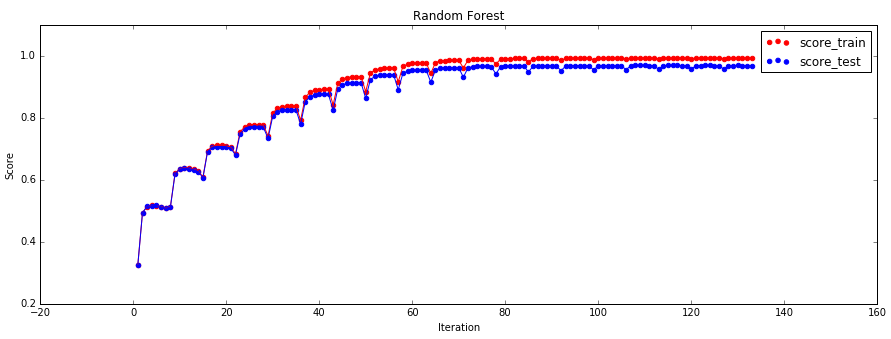
So, we fixed the number of trees to be 40 and we increased the possible maximum tree depth from 10 to 40 (with intervals of 2) and we increased the maximum number of features by reducing the interval level from 20 to 10. This resulted in a 133 total number of possible combinations:

# Random forest regression with 100,000 observations tuning max\_depth and max\_features

n\_trees = np.array([40])

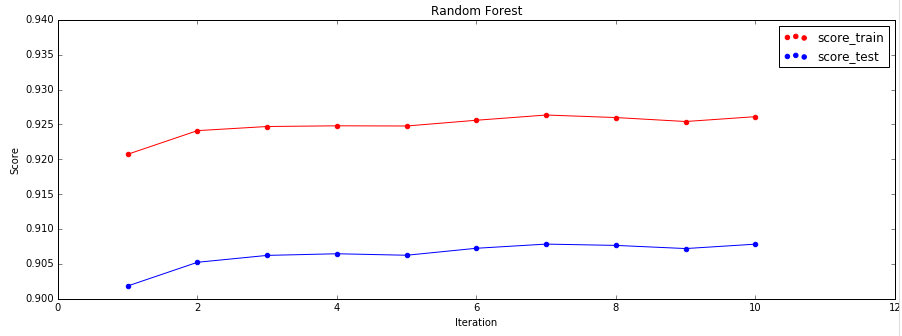
depths = np.arange(2, 40, 2)

features = np.arange(5, 70, 10)



This model resulted in an R^2 value of 0.91 on the test dataset for a 5-fold cross validation dataset.

For the final iteration of the model, we selected a maximum depth of 14 and the maximum number of features to be 25. We want to have the fewest number of trees possible while maintaining model score to decrease training time. However, in order to get an accurate prediction interval with our quantile regression forests method, we need to use more trees (200) as explained by: <http://blog.datadive.net/prediction-intervals-for-random-forests/>



Final Model

This final model results in an R^2 value of 0.92 on the test data created from the 5-fold cross validation method. This tuned model was implemented on a sample of 1 million randomly sampled observations from the 7.2 million observation modeling dataset. The final model was a Random Forest Regression model with 1000 trees, a maximum depth of 14 and maximum features of 25. The dataset was split into 75% for the training dataset and 25% for the test dataset. The results of the application of this model was an R^2 of 0.92.

We use the quantile regression forests method to build a 90% prediction interval around each of our predictions. The idea behind quantile regression forests is relatively straightforward. Treating the 1000 predictions from each of our trees as the distribution of our prediction, we use the 0.05 and 0.95 percentiles as the lower and upper bounds on our prediction interval. This prediction interval contains the true y\_test value for approximate 80% to 90% of the test values.

**Next Steps:**

The Airbnb price prediction model has to opportunity to be developed further by tuning the model to other locations and making the predictions public for Airbnb users. Property owners could use the model results to competitively price their property while customers could use the model results to seek fair priced properties to rent.