UNIVERSITY OF CALIFORNIA

Los Angeles

California Rental Price Prediction Using Machine Learning Algorithms

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Statistics

by

Yue Fei

2020

ABSTRACT OF THE THESIS

California Rental Price Prediction
Using Machine Learning Algorithms

by

Yue Fei
Master of Science in Statistics
University of California, Los Angeles, 2020
Professor Yingnian Wu, Chair

Rental price prediction (price recommendation) is a practical topic in the current online marketplace. In order to support hosts with less experience to set up the competitive rental prices, we utilize the techniques, such as feature engineering and machine learning algorithm, to select useful features and conduct models to predict possible rental prices based on the property information provided by the hosts. In this paper, machine learning algorithms are implemented on the same dataset which contains all the properties in California listed on Airbnb. After feature analysis, we notice the number of bedrooms and property types are the most important features that are highly associated with the rental prices. Among all the methods, XGBoost gives the most satisfying prediction results of rental prices based on RMSE, MAE, and R^2.

The thesis of Yue Fei is approved.

University of California, Los Angeles

2020

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

Nowadays, traveling becomes much easier and more diversified. People are given a lot of options for accommodations during traveling. Where and what property type do you want to stay in when you are planning for your trip? This may be the first concern for most people when they are making plans for the trips. Years ago, while the hotel used to be the first choice for many of us, now, people have more choices on online marketplaces, such as Airbnb, which is a platform connecting guests who need accommodations with hosts who want to rent their properties. By 2020, Airbnb has more than 150 million users, 0.65 million hosts, and 7 million listings globally [1]. According to the current growth, we may expect that the total number will continually increase in the next few years. Given a large number of hosts and listings, reasonable and competitive rental prices are important to maximize the benefits for both hosts and guests.

On the Airbnb website, it is very considerable to provide the estimated earnings based on the general information, such as location (city), room type, and the number of guests with the assumption that at least 15 nights are booked every month. However, the estimation may not be appropriate because the information used is too general. Even if for the properties in the same area, rental prices can be various based on property types, zip code, and so on. Therefore, the task in our case is to find a more reliable way to estimate the rental prices. I notice that the commonly used ways for most hosts to estimate the prices are based on their own experiences, or they can use other properties with similar conditions as references. However, these ways may not be sufficient since the number of properties the hosts can use as references are relatively small. Also, it can be even harder for the first-time host who has no experience in setting up rental prices at all, so we want to use Airbnb data to provide more reasonable prices as references for the hosts.

In this project, we collect all California data from Inside Airbnb website to estimate the rental prices. As the preparation step for analysis, data cleaning, exploratory data analysis, and feature engineering will be conducted. After the finalized dataset is well-formatted, we will implement five machine learning methods, which are Ridge Regression, Ridge Regression with K means, Support Vector Machine - Regression (SVR), Random Forest, and XGBoost. Their performances will be evaluated based on three indicators, R^2, Root Mean Square Error (RMSE), and Mean Absolute Error (MAE). Among all five methods, the best-preformed one will be eventually selected, and further discussion will be given on later chapters.

CHAPTER 2

Exploratory Data Analysis

2.1 Data Overview

The data we used is collected from Airbnb by inside the Airbnb website, and it was last updated on February 13, 2020. The data contains all datasets from eight regions in California, which are Los Angeles, San Diego, San Francisco, Santa Cruz, San Mateo, Oakland, Pacific Grove, and Santa Clara. We have 8 datasets (75250 observations in total), and each dataset contains 106 variables. Table 2.1 only includes the first six rows of our raw data. Due to a large number of variables and messiness of data, we will talk about more details of data cleaning and feature engineering for analysis in the next chapter.

Table 2.1: Data Observation

id	listing_url	host_id	host-since	host_location	host_repsonse_time	...	price	...	reviews_per_month
109	http://www.air...	521	2008-06-27	San Francisco,CA,US	NA	...	$122	...	0.02
344	http://www.air...	767	2008-07-11	Burbank,CA,US	within a few hours	...	$168	...	0.19
2708	http://www.air...	3008	2008-09-16	Los Angeles,CA,US	within a few hours	...	$79	...	0.33
2732	http://www.air...	3041	2008-09-17	Santa Monica,CA,US	within a few hours	...	$140	...	0.19
2864	http://www.air...	3207	2008-09-25	Bellflower,CA,US	within a few hours	...	$80	...	NA
5728	http://www.air...	9171	2009-03-05	Los Angeles,CA,US	NA	...	$75	...	2.41

2.2 Data Cleaning

On the inside Airbnb website, each region has its own dataset. For example, there are 8 datasets in total for California. As a result, I cleaned each dataset separately before we combine all datasets together. Each dataset has 106 variables that include many irrelative features, such as different types of URL, user names, and state information, so they are eliminated in the first step. Since we only work on the data of California, the physical locations of the properties, which are not in California, will be also excluded in this step. Before exploratory data analysis, we will look at the summary table of the response variable, price [1].

Table 2.2: Summary of Price

Min	1st Qu	Median	Mean	3rd Qu	Max
0.0	75.0	120.0	2217.1	200.0	25000.0

Based on the summary in Table 4.2, we may notice it is not reasonable to have rental prices equal to \$0 from a profit perspective, and there are only 12 observations with rental prices equal to \$0, so the observations with \$0 as their prices will be omitted. Also, the maximum price is \$25000 which is extremely larger than most of our prices, so we will only include the prices lower than \$5000. The next task is to deal with missing values. For the numerical variables with a large proportion of missing values, instead of filling missing values, they will be directly excluded to limit noise caused by imputation. For the rest numerical variables and binary variables, the missing values will be replaced by 0 or ”False”. For example, the missing values in *security_deposit* will be directly replaced by 0, and those in *is_host_superhost* will be replaced by ”False”. For the categorical variable with multiple levels, such as *host_response_time*. The missing values will be replaced by the base level, ”Never”, and the variable, *bedrooms* which is the total number of bedrooms in the property, will be imputed by group mean based on *property_type* and *room_type*. The last variable

[1]Daily rental price

with missing values is *zip_code*. We can use the external sources of US zip code to fill the missing values based on their latitudes and longitudes [12].

Besides missing value imputation, data transformations may also be necessary for some features. For example, the variable, *summary*, contains descriptive text information. For simplicity, a new binary variable will be created to determine if the summary is provided or not. This is because we expect that more property information provided, more popular the properties will be, and rental prices can be higher. Also, the variable, *amenities*, includes all details of amenities offered by hosts. In this case, we only count the number of amenities offered in each property. Moreover, there are two time-related variables, $first\_review$[2] and *last_review*[3], so the difference between the two variables will be used to keep track of how long the hosts continue their businesses on Airbnb. As the last step of data cleaning, we will create dummies for multi-level categorical variables for feature selection. After the data cleaning steps, all eight datasets will be appended together as our final dataset. The exploratory analysis will be conducted in the next chapter, so we can have a better understanding of the data, and the variable description will be given after feature selection.

2.3 Exploratory Analysis

In this section, we can take a further look at the relations between all the features and rental prices using data visualization. Since the finalized data includes all property information crossed California, let us first look at the proportion of each region in the final dataset as showed in Figure2.1. We can see that 53.36% of observations are from Los Angeles, and the data from San Mateo and Pacific Grove is relatively small since both of them are less than 1%. Based on this information, we may expect the rental prices in these two regions have large fluctuations.

[2]The date on which the host is reviewed at the first time

[3]The date on which the host is reviewed at the last time

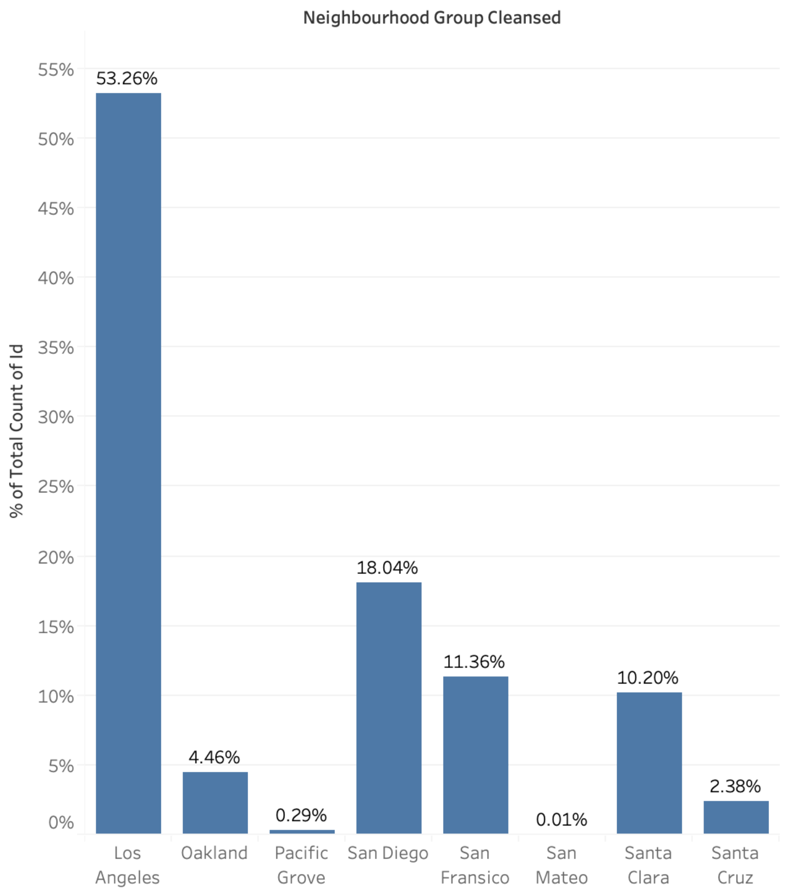


Figure 2.1: Sample Proportion of Neighborhood Groups

Figure 2.2, the boxplot of rental price based on neighbor groups shows that there are many extreme values in each group, which may be caused by different reasons, such as property types or location. We can expect that the rental price of a villa will be much higher than it for a shared room. This issue will be solved in the next section by data transformation. Other than the extreme values, we may also notice that there are some overlaps in the interquartile ranges for some groups. For example, the IQRs and means for San Diego and San Francisco are almost the same, so we may assume there is no significant difference in rental price between these two groups. The boxplot also indicates that there are possible significant differences in average rental prices for other groups, so some values in the variable, *neighbor_groups*, can be important for our prediction.

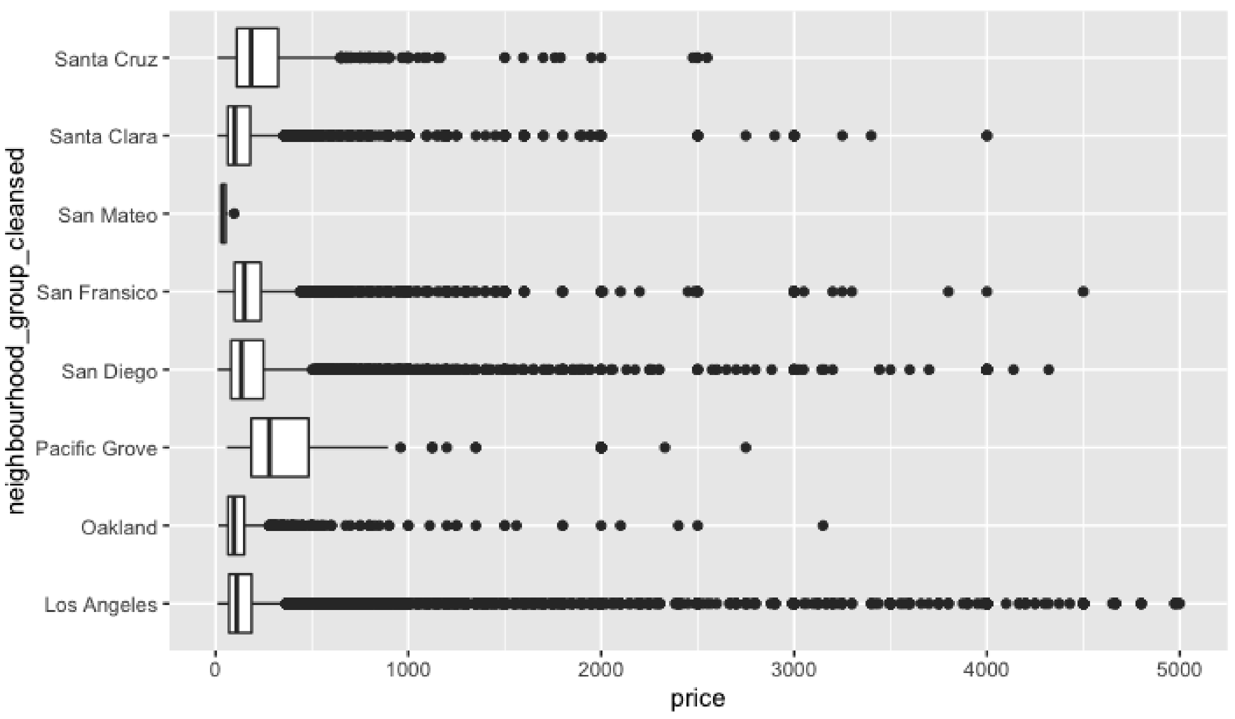


Figure 2.2: Rental Price by Neighborhood Groups

When we think about factors that highly affect rental prices, what will be the first feature that comes into our mind? For me, it is the location of properties. We know that even with the same quality and conditions, the property rental prices can be various if they are not in the same location. Figure 2.3 shows the relations among the number of properties, average prices, and locations. The two plots on the left show the number of properties in each zip code. We notice that Southern California has more cities in which the total number of properties is greater than 1000 compared with North California. That can be reasonable since we have more than half of observations are from Los Angeles. Also, the right plots show the average rental prices by zip codes. Similar to the plots on the left, there are more local areas in southern California with average rental prices higher than $500. If we conduct further analysis of those cities, we can find there are some cities significantly affects the rental prices. Considered the case in Los Angeles, the areas with high average prices are either close to the coast, such as Malibu, or they are near to Beverly hills. Because of that, we can assume some cities (neighborhoods) will have significant impacts on rental prices.

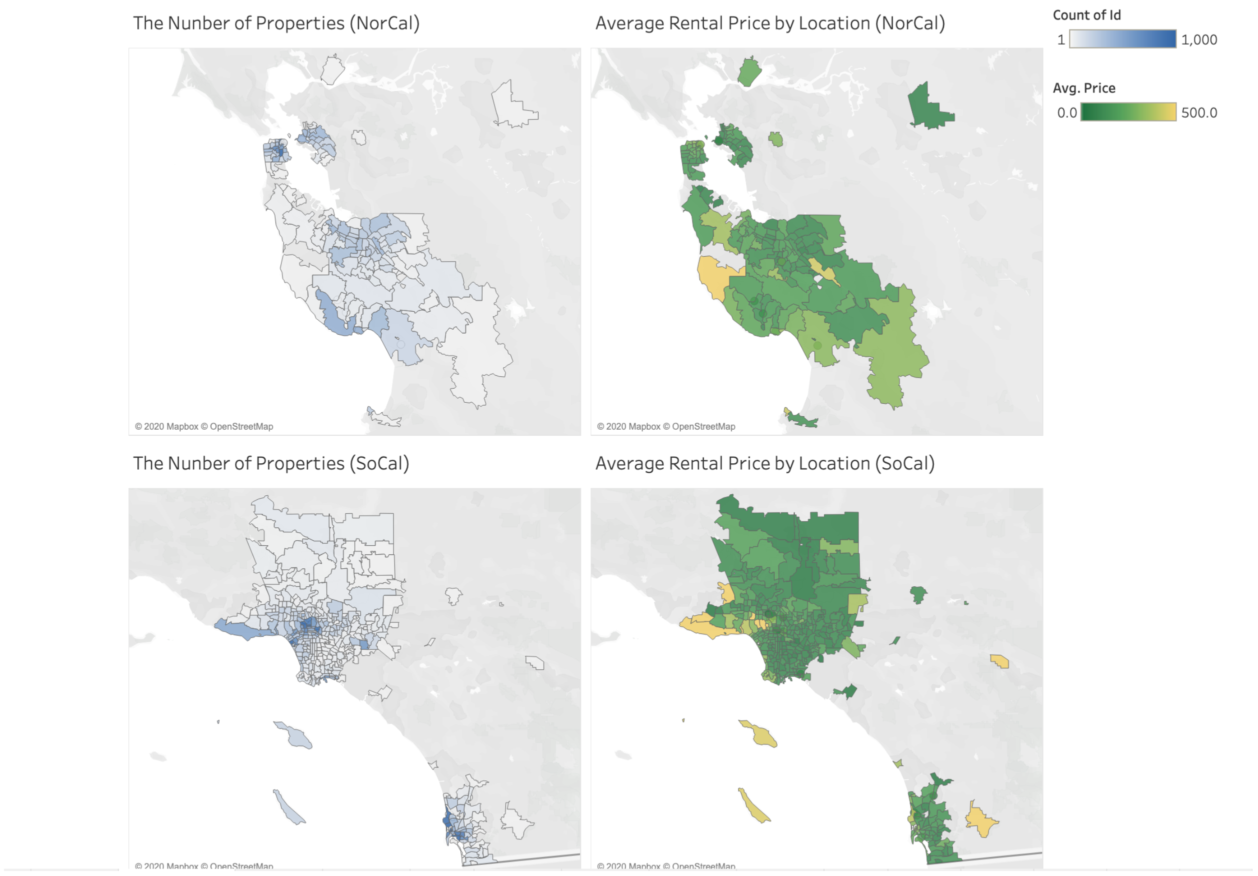

The Nunber of Properties (NorCal)
Average Rental Price by Location (NorCal)
Count of Id
1
1,000
Avg. Price
0.0
500.0
The Nunber of Properties (SoCal)
Average Rental Price by Location (SoCal)
© 2020 Mapbox © OpenStreetMap

Figure 2.3: Maps

Other than the location, the duration of being a host on Airbnb can be also highly related to the rental price. Considered the case that the hosts actively run their rental businesses on Airbnb for many years, then we can expect these hosts have more experience in setting up the rental prices, and we can assume their prices should be more stable, so we conduct the line plot of *cont*[4] versus average prices to check the association between average prices and duration of being hosts on Airbnb. In Figure 2.4, we may find that the fluctuations of the average prices become larger as *cont* increases. The possible reason can be the hosts with less experience tend to be more conservative when setting up the prices, so we can see

[4]cont = last_review − first_review (the duration of being a host on Airbnb)

that for the hosts run their business on Airbnb less than 500 days, the fluctuations are mild compared with those are Airbnb longer than 2000 days.

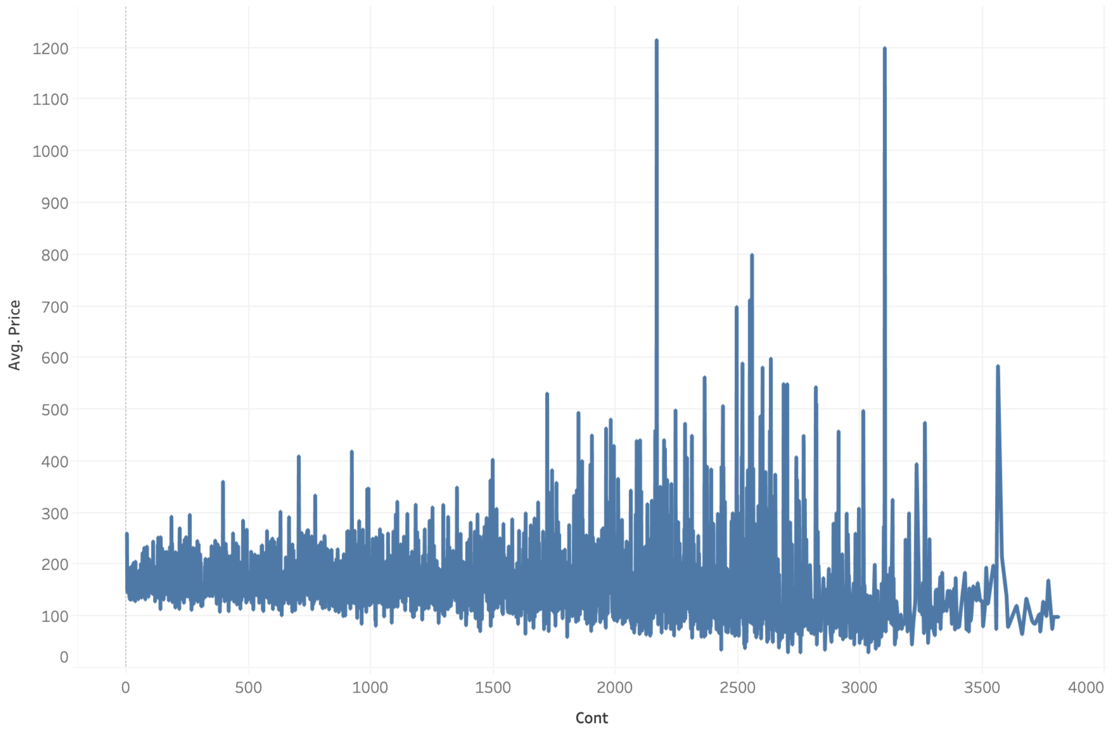


Figure 2.4: Average Rental Price by the Days Stayed in Airbnb as Hosts

The cancellation policy is also likely to affect rental prices. Considered the case if the cancellation policy is extremely strict, then we can expect the rental price of such properties should be high. Based on the information from Airbnb, we have five levels for cancellation in general, flexible, moderate, strict, super strict 30 days, and super strict 60 days [4]. Figure 2.5 gives a straight forward view of the relation between cancellation policy and average rental prices. For all the policies contained "luxury", the average prices are significantly higher than others, which can be explained by the fact that costs of luxury properties are much higher than any other types, so it may cause a big loss for hosts without the relatively strict cancellation policy. Besides the cancellation policy of luxury properties, we also can observe the trend that as the policy becomes stricter, the average rental prices get higher.

We can expect some values from the cancellation policy will stand out for modeling.

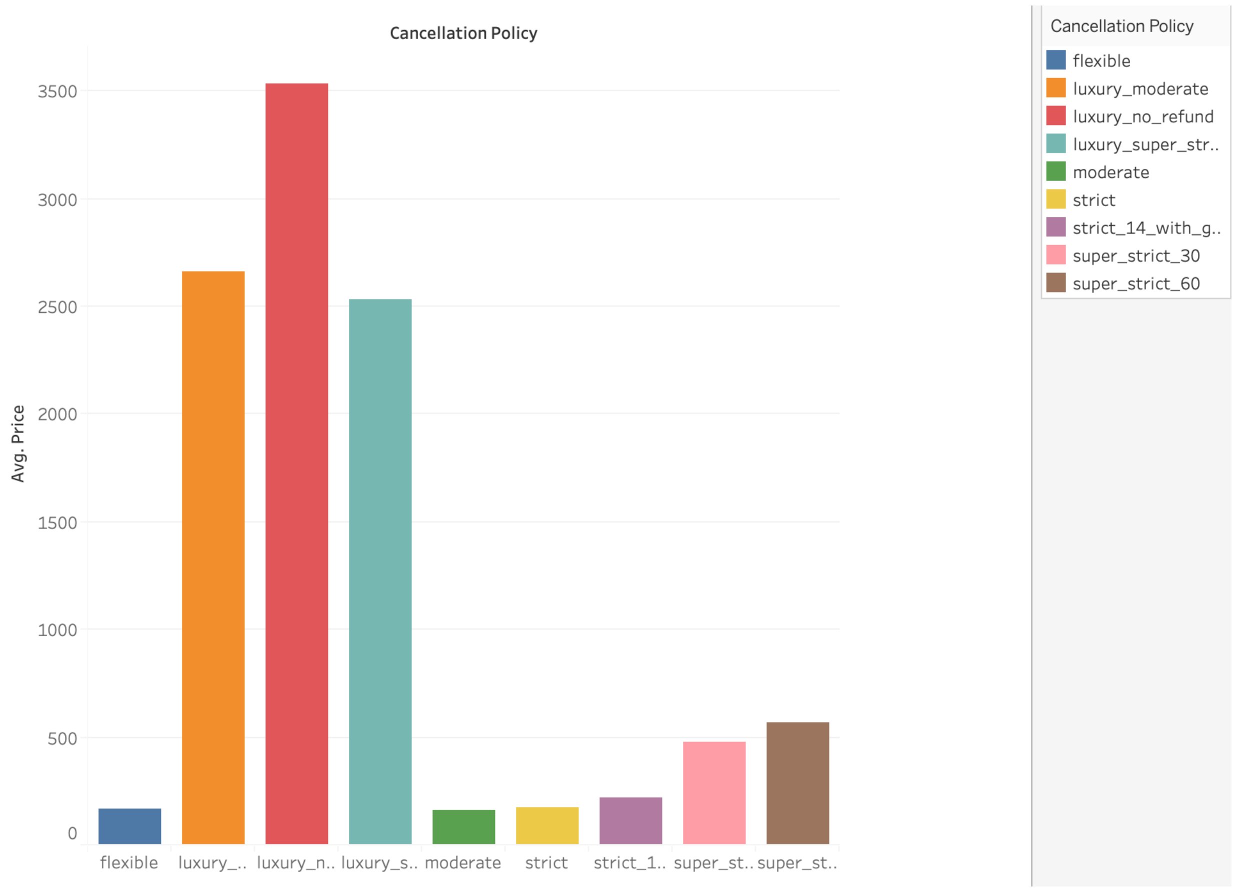


Figure 2.5: Average Rental Price by Cancellation Policy

The total number of bedrooms and bathrooms can also affect the rental prices. Based on common sense, we know the number of bedrooms is proportion to the number of bathrooms, so these two features should be highly correlated. Also, we should expect an increase in the rental price as the total number of bedrooms and bathrooms increase. In Figure 2.6, we can see the general trends of the subplots are similar. When the values of variables on the x-axis are relatively small (less than 8), the average prices go up as these variables increase. This information implies that these two variables may be highly correlated and impact the rental price, so we want to keep one of the variables, *bedrooms* or *bathrooms*, in the models. In the later section, we will conduct a correlation matrix to determine which variable to keep for modeling.

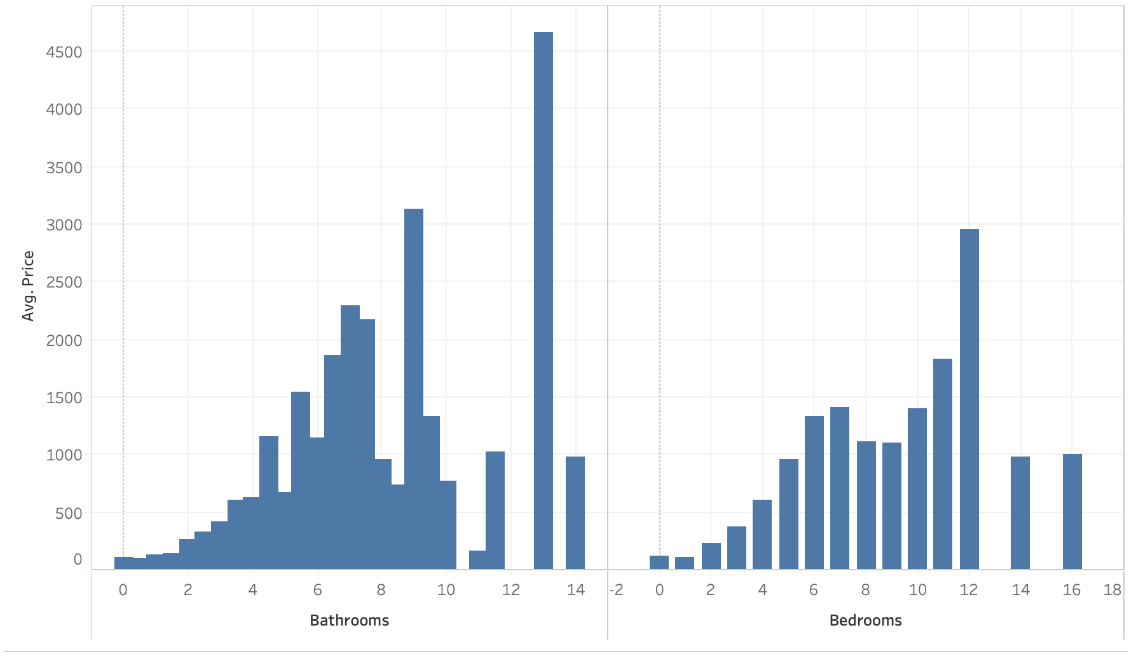


Figure 2.6: Average Rental Price by the Number of Beds and Bathrooms

Now, let us take a look at the relations among room types, property types, and average rental prices. As we mentioned earlier, the average rental prices should be much higher for luxury properties, such as the castle and villa. Figure 2.7 indicates that the average rental prices of the entire home or apartment associated with some specific property types are significantly higher than others. We can see that the average rental price of an entire villa is almost \$1500 while the average price of a shared room in an apartment is only 68. As a result, we have a reason to believe that some components in these two variables are important for rental price prediction.

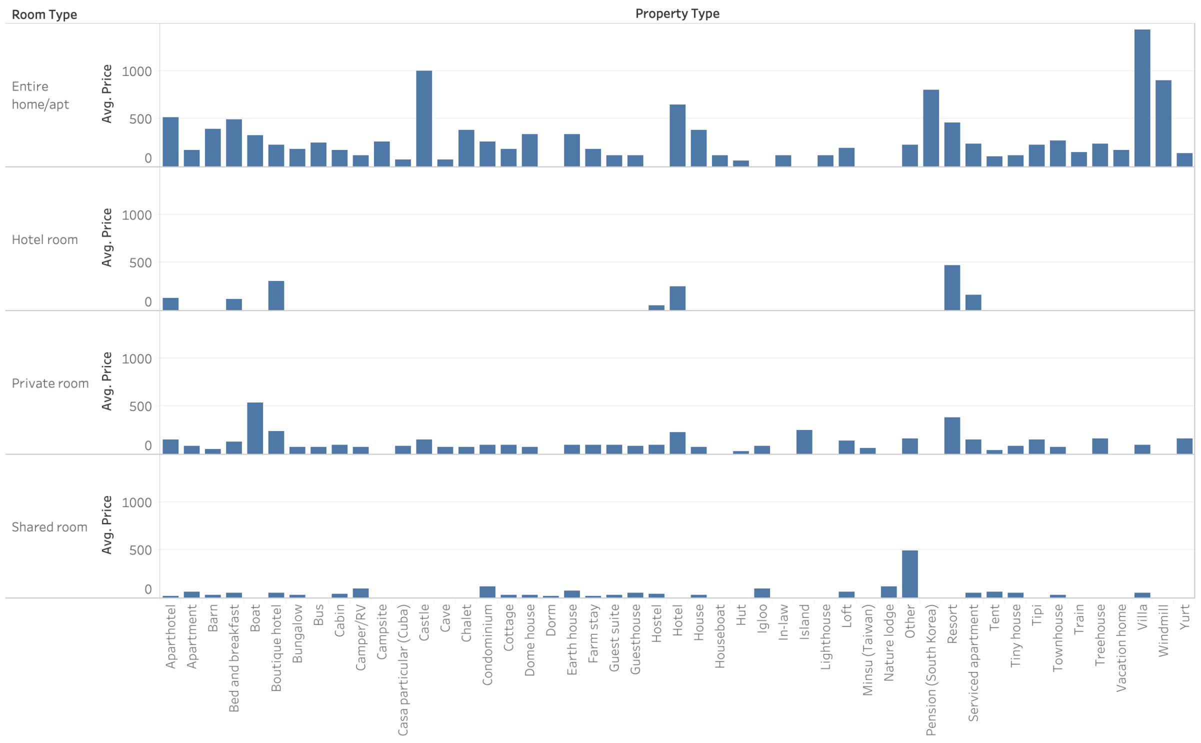


Figure 2.7: Average Rental Price by Room Types and Property Types

2.4 Feature Transformation and Selection

After the exploratory data analysis, we have a brief understanding of our data, also, we may notice there are some drawbacks in the original dataset. For example, most variables have different units, and the response variable, *price*, is highly skewed as it indicates in Figure 2.9. This may cause the result that the variables with larger scales may have bigger impacts on prediction prices. Therefore, data scaling and transformation are necessary before modeling. In order to adjust all the variables to the same scale, standardization of numeric variables will be performed, which is that each numeric variable will minus its mean and then divide by its standards deviation.

$$Z_i = \frac{V_i - \bar{V_i}}{S} \tag{2.1}$$

where Z_i is the i^{th} variable after re-scaling, V_i represents the i^{th} variable,$\bar{V_i}$ is the mean of i^{th} variable, and S is the sample standard deviation.

For the response variable, *price*, we may notice the density of the original price is highly skewed, which is caused by the fact that there are several types of properties with extremely high prices, such as the entire castle or villa. In order to reduce the effects of the extreme values on our models, the log transformation will be provided. In the Figure 2.9, we can notice that after the log transformation, the variable, *price*, is not highly skewed, also, the qqnorm plot (Figure 2.8) indicates that the response variable becomes normally distributed after the log transformation, which will be extremely helpful for model implementations. The final step in this section is to split our dataset into training and testing sets. We will randomly select 75% of observation as the training set to fit models, and the rest 25% of the data will be treated as testing set to check the performance of models. The feature selection step will be conducted based on the training set.

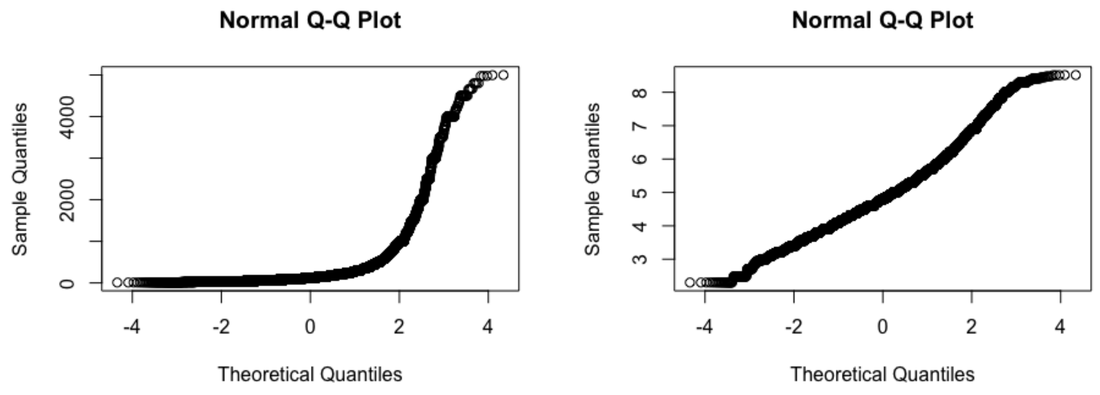


Figure 2.8: QQ plot

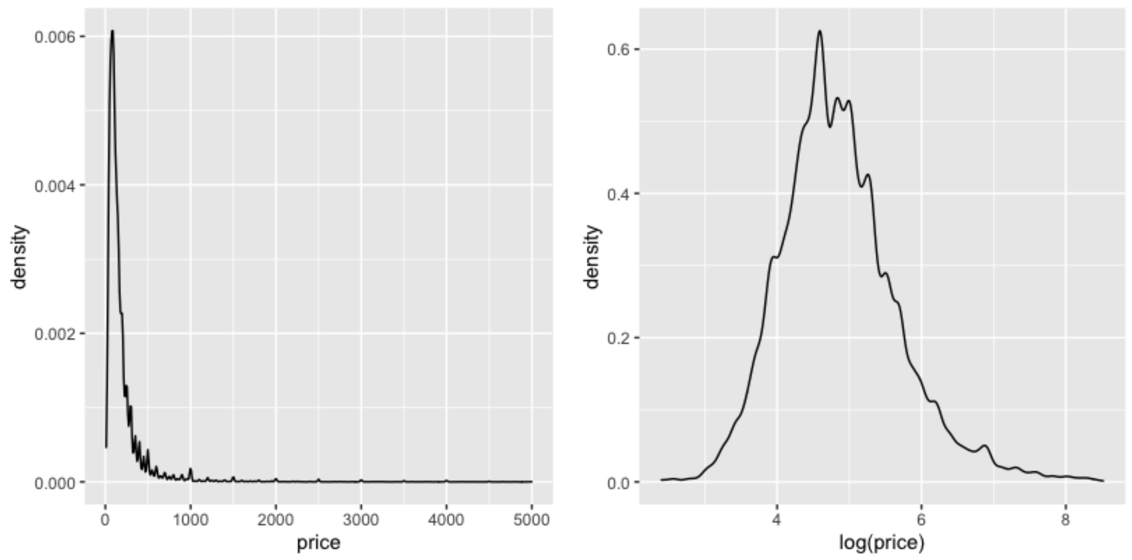
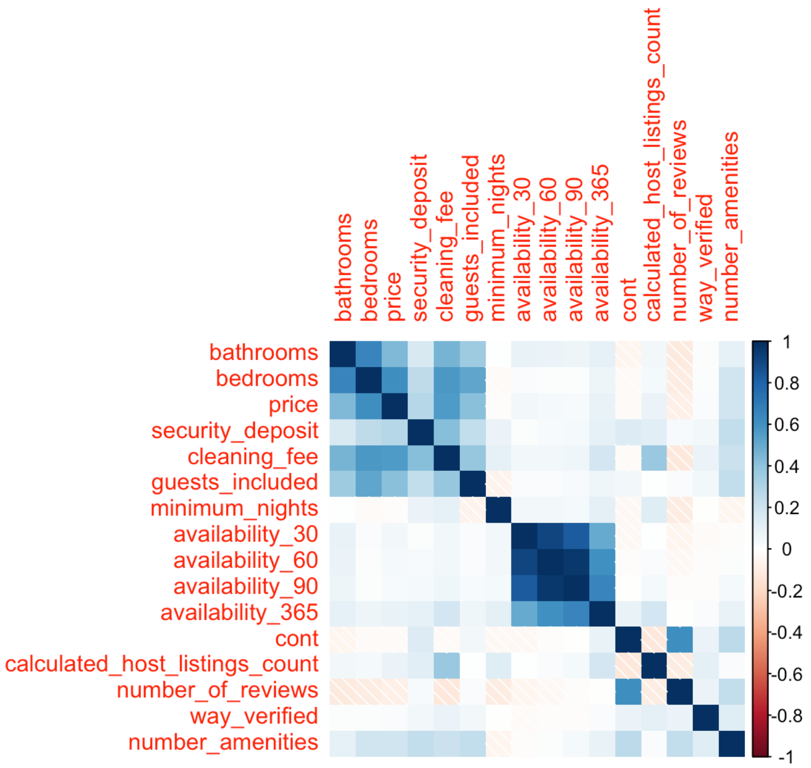


Figure 2.9: Log Transformation of Response

After the data cleaning step, we keep 276 variables including dummy variables. In order to prepare the data for modeling, the further feature selection will be performed in this section since not all variables are informative for prediction. We first check the correlations among all numeric variables. In Figure 2.10, we can see that the variables, *bedrooms* and *bathrooms* are highly correlated, and as we assume, the number of bedrooms is proportional to the number of bathrooms. Besides that, *bedrooms* is also highly correlated with *cleaning_fee* and "*guest_included*. Since the number of bedrooms has the highest correlation with price, we will only keep *bedrooms* to avoid the multi col-linearity issue. Another big dark blue chunk in the plot includes all four variables related to availability, so we decide to only keep *availability_*30. The last two highly correlated variables are *number_of_reviews* and *cont*. This is because the longer the hosts run their businesses on Airbnb, the more reviews they will get. Then, the highly correlated variables will be eliminated from our final dataset, and the new correlation plot is conducted on the bottom.



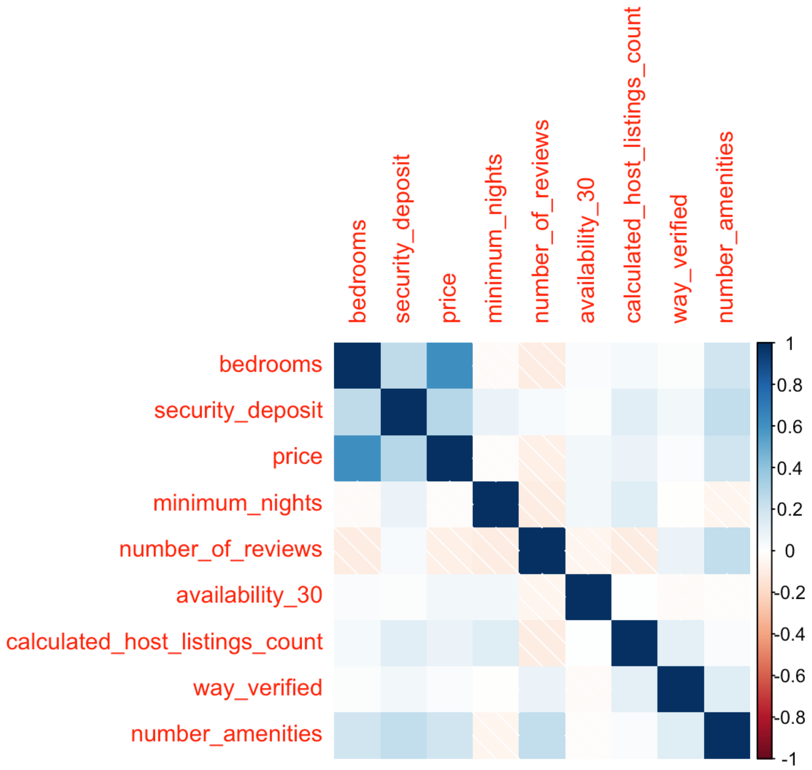
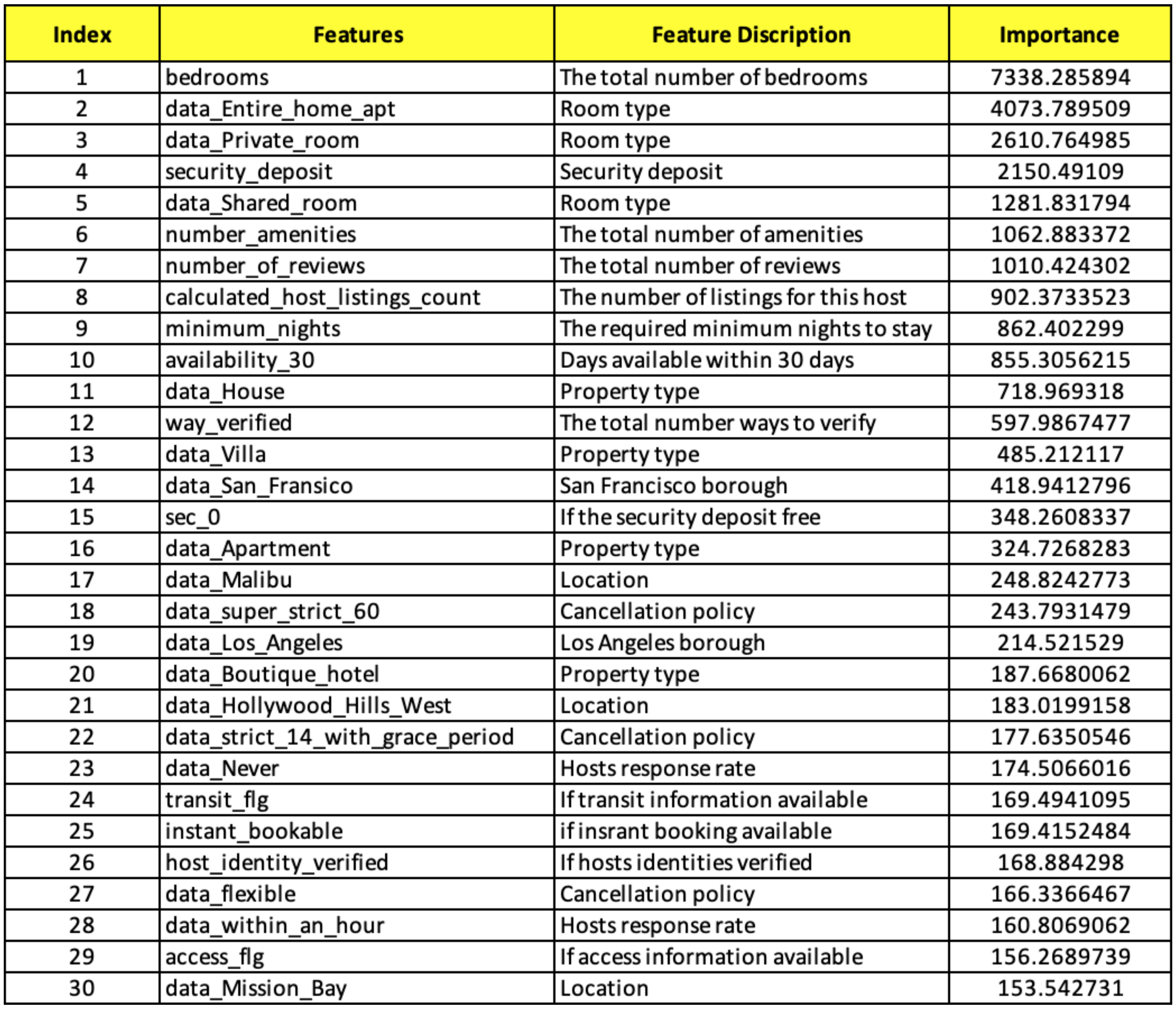


Figure 2.10: Correlation Maps

The next step is a formal feature selection. We will pick the important features from all 276 variables so that it will be more efficient for modeling and prevent from overfitting issue. Here, we use Random Forest for feature selection. Since too many variables in the dataset, we finally decide to include the top 30 variables, and their descriptions are also listed below.



Index	Features	Feature Discription	Importance
1	bedrooms	The total number of bedrooms	7338.285894
2	data_Entire_home_apt	Room type	4073.789509
3	data_Private_room	Room type	2610.764985
4	security_deposit	Security deposit	2150.49109
5	data_Shared_room	Room type	1281.831794
6	number_amenities	The total number of amenities	1062.883372
7	number_of_reviews	The total number of reviews	1010.424302
8	calculated_host_listings_count	The number of listings for this host	902.3733523
9	minimum_nights	The required minimum nights to stay	862.402299
10	availability_30	Days available within 30 days	855.3056215
11	data_House	Property type	718.969318
12	way_verified	The total number ways to verify	597.9867477
13	data_Villa	Property type	485.212117
14	data_San_Fransico	San Francisco borough	418.9412796
15	sec_0	If the security deposit free	348.2608337
16	data_Apartment	Property type	324.7268283
17	data_Malibu	Location	248.8242773
18	data_super_strict_60	Cancellation policy	243.7931479
19	data_Los_Angeles	Los Angeles borough	214.521529
20	data_Boutique_hotel	Property type	187.6680062
21	data_Hollywood_Hills_West	Location	183.0199158
22	data_strict_14_with_grace_period	Cancellation policy	177.6350546
23	data_Never	Hosts response rate	174.5066016
24	transit_flg	If transit information available	169.4941095
25	instant_bookable	if insrant booking available	169.4152484
26	host_identity_verified	If hosts identities verified	168.884298
27	data_flexible	Cancellation policy	166.3366467
28	data_within_an_hour	Hosts response rate	160.8069062
29	access_flg	If access information available	156.2689739
30	data_Mission_Bay	Location	153.542731

Figure 2.11: Average Rental Price by Room Types and Property Types

CHAPTER 3

Methods

In this chapter, we will go through more details about the application of machine learning algorithms in rental price prediction. We want to use the machine learning algorithms to build up our own recommendation system to suggest reasonable prices based on the property information provided by hosts, and they can make any adjustments based on the predicted results as they need. We have continued response in this case, so the implemented regression models are Ridge Regression, Ridge Regression with K means, SVR (support vector machine regression), Random Forest, and XGBoost. Their performances will be evaluated by R^2, RMSE (the root of mean square error), and MAE (mean absolute error). The detailed implementation strategies used in these methods will also be discussed in this chapter.

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} \tag{3.1}$$

$$RMSE = \sqrt{\frac{\sum(y_i - \hat{y}_i)^2}{n}} \tag{3.2}$$

$$MAE = \frac{1}{n}\sum |y_i - \hat{y}_i| \tag{3.3}$$

3.1 Ridge Regression

The first method we considered is ridge regression which consists of linear regression with a weighted L2 regularization term to prevent the parameters from overfitting. From a mathematical perspective, the ridge regression finds the optimal $\hat{\beta}$ that minimizes the residual sum of square error plus a regularization term. Shown as below [5]

$$\hat{\beta}_\lambda = \underset{\beta}{argmin}[|Y - X\beta|^2 + \lambda|\beta|^2] \tag{3.4}$$

Where β is a vector of coefficients and λ is the weight of regularization. Suppose $f(\beta) = |Y - X\beta|^2 + \lambda|\beta|^2$, then $f'(\beta) = -2X^T(Y - X\beta) + 2\lambda\beta$ and set $f'(\beta) = 0$, we can get $\hat{\beta} = (X^TX + \lambda I_p)^{-1}X^TY$. Choosing the propriety lambda value is important in Ridge Regression. Considering the case when $\lambda = 0$, it becomes to the linear regression, which may cause overfitting, or if lambda is extremely large, then the model becomes extremely conservative, which leads to the result that all coefficients are close to zero. Figure 3.2 shows the coefficients shrink towards zero as the $\log(\lambda)$ increases. The red dot in Figure 3.1 is the ridge estimation [5].

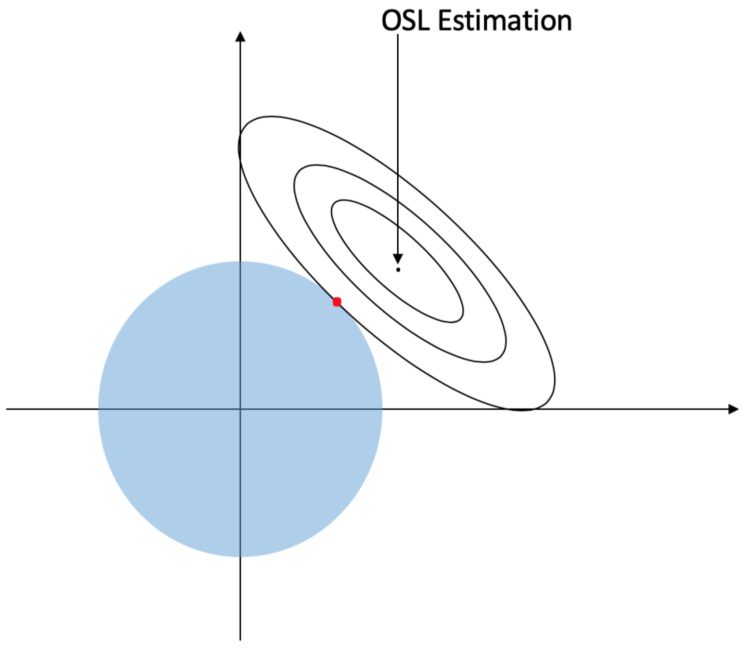


Figure 3.1: Average Rental Price by Room Types and Property Types

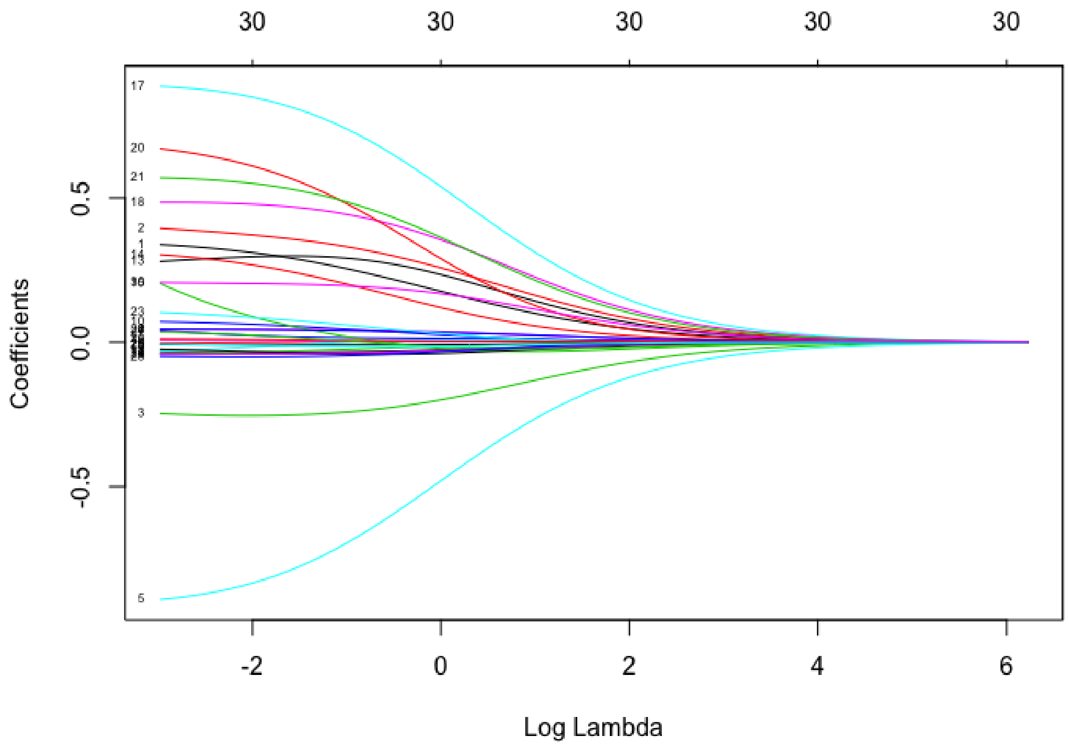


Figure 3.2: Average Rental Price by Room Types and Property Types

In order to find the best λ, we will fit the ridge regression with a sequence (100 values) of λ from 1e-10 to 10. Then, the 10 folds cross-validation will be used to find the optimal λ, which minimizes the mean cross-validated error. In Figure 3.3, the plot on the bottom shows the relation between mean square error and log(λ). The first dash line form left is the best lambda which is 1.668101e-06, and it will be used to fit ridge regression.

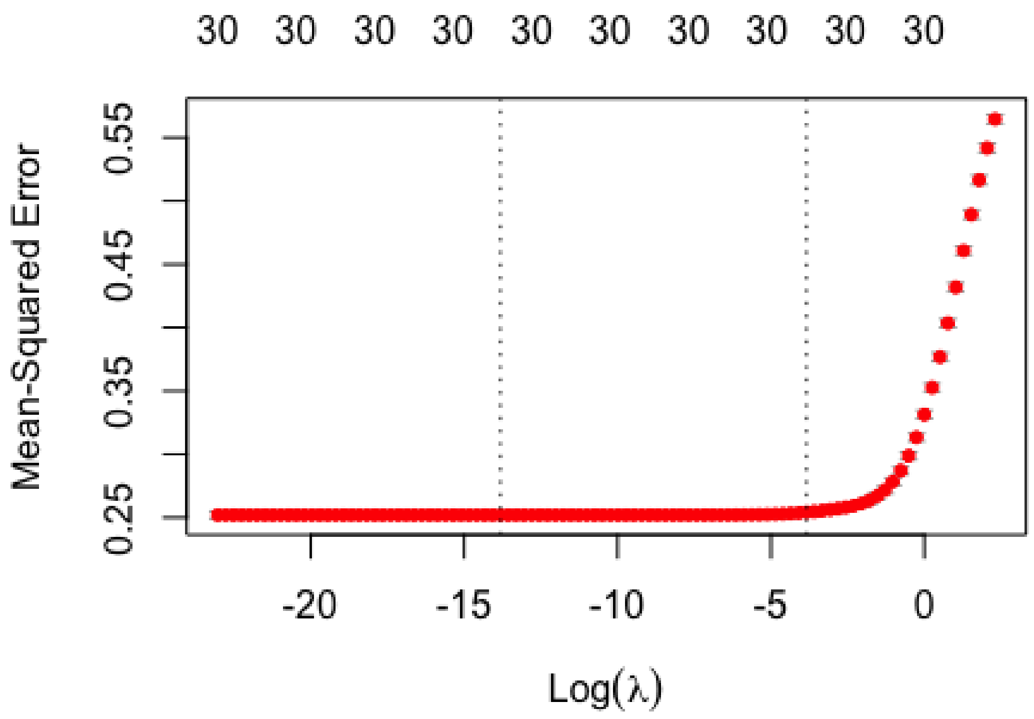


Figure 3.3: Average Rental Price by Room Types and Property Types

3.2 Ridge Regression with K Means

Ridge regression itself may not able to capture the effect of segmentation on our predictions, so to improve the prediction, we can consider first putting the data into different groups, and each group has training and testing set. Then, we will build ridge regression for each group based on training data and evaluate the performance using the testing data [10]. Since we do not have clear labels, the unsupervised clustering algorithm, K means will be used. In K means algorithm, the centroid, k, indicates the number of clusters that we want to divide data into. The algorithm starts from randomly choosing k centroid as we specified. Then, data will be assigned to different clusters according to the centroids, meanwhile, centroids will be re-calculated based on the data points in each cluster. Those steps will be performed iteratively until convergence.

After we know the algorithm, choosing k would be the most important task in this combined method. Let us first visualize the data in a 3D space using a dimension reduction technique. In Figure 3.4, we notice that the data is clearly separate into two groups in 3D space, so the initial approach is chosen k using the Elbow method, which is that we will set k from 2 to 20, and the total within clusters sum of square will be calculated for each k. We want to find the elbow point from the plot. That will be our choice of k. In it is shown in Figure 3.5, we can see that the within clustering total sum of square gradually decreases as k increases, and we may notice the elbow point is 10.

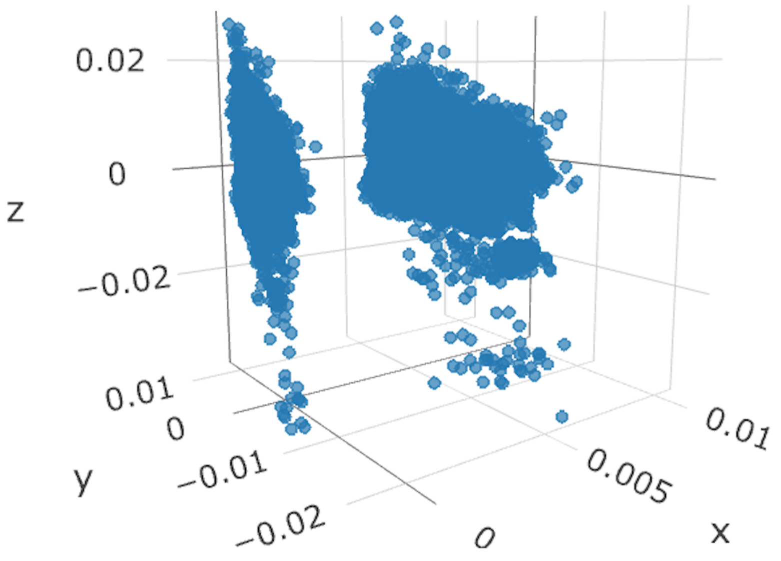


Figure 3.4: PCA without labels

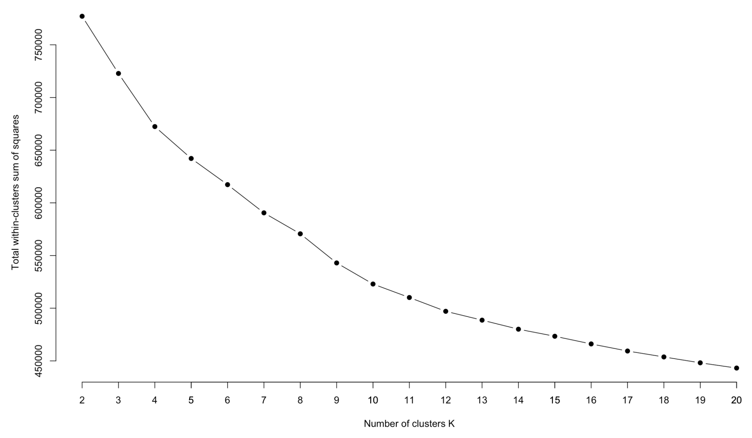


Figure 3.5: K means

3.3 Random Forest

Random forest has been a wildly used machine learning algorithm in industries because it not only provides satisfying results but also has no distribution assumption about the data. Random forest works pretty well on high dimensional data, which is the case for our dataset. Before we talk about random forest algorithm, we should mention the decision tree a little bit. There are two types of decision trees, classification, and regression. Since we have continued numeric response, so the regression tree is appropriate for our case. As shown in Figure 3.6, the regression tree will go through each node to determine which branch the observation will be assigned, and the group average will be used as predicted results when we reach the bottom of the tree. We may have a question about how the decision tree arranges the conditions? For example, if we have only two variables associate with rental prices, which are bedrooms and security deposit. How do we know which one we will determine first? The sum of squared errors (SSE) for those conditions will be compared. The conditions with the smallest SSE will be the first node, and so on.

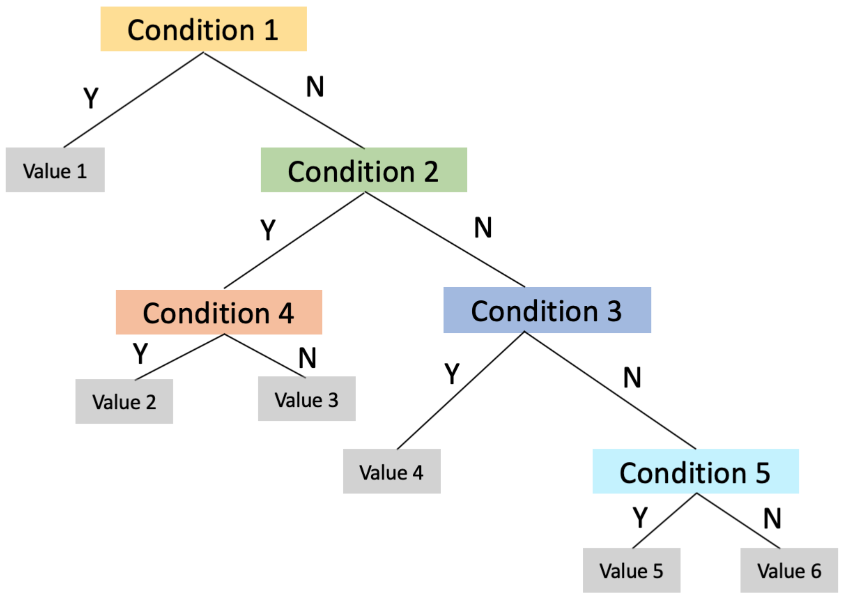


Figure 3.6: Decision Tree

Now, we can consider random forest as a combination of multiple independent decision trees, which showed in Figure 3.7 [9]. Each individual tree will go through the nodes independently with the same weight and come up with its own predictions. For the regression, the average of predictions from all decision trees will be used as our final predictions. Also, there are some unique characteristics of random forest. Instead of using all observations, we can randomly choose n bootstrap samples from the whole observations, and for each bootstrap sample, only m out of the total number of features (specified as mtry in R) will be used in each node, which helps to prevent from overfitting [8]. In our case, the different combination will be used. We will use mtry as 10 and 15, ntree as 500, sample size as 10%, 50%, and 100 % of training data with replacement. The results of random forest will be discussed and compared with other methods in the later chapters.

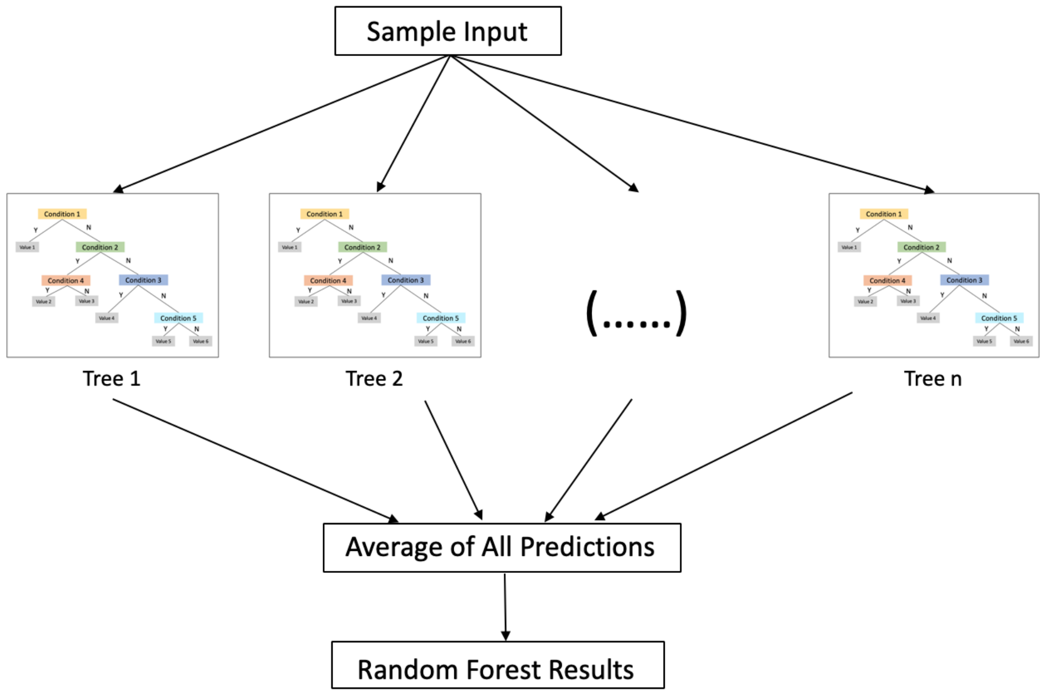


Figure 3.7: Random Forest

3.4 Support Vector Machine Regression

Support vector machine invented by Vladimir Vapnik [13] has become a widely used non-parametric machine learning algorithm for not only classification but also regression. The idea behind it is that we want to find the optimal solution of the boundary for the regression in the high dimensional space. That means if we consider X as our predictor matrix, and Y as response vector in the training set, we want to find the function $f(X)$ such that the difference between $f(X)$ and true response Y are less than or equal to ϵ which can be considered as a threshold we define. It has a similar purpose as regularization since ϵ is used to penalize the estimations when the differences between prediction and true response are larger than ϵ. Therefore, as Alex mentioned in his tutorial [11], the optimization question that we need to solve is

$$\begin{aligned} min \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi^*) & (3.5) \\ \text{subject to} \quad & y_i - \langle w, x_i \rangle - b \le \epsilon + \xi_i & (3.6) \\ & \langle w, x_i \rangle + b - y_i \le \epsilon + \xi^* & (3.7) \\ & \xi_i, \xi^* \ge 0 & (3.8) \end{aligned}$$

where w represents the classes that can be separated linearly. ξ_i, ξ^* measure the differences larger than ϵ. Parameter C indicates the trade-off between the complexity of the model and the level of deviations allowed. Figure 3.8 gives a better illustration about each term [6]. When we implement support vector machine regression in R, there are some important parameters in SVM function. The kernel is the function that is used to map the features into high dimensional space, here we use the Radial basis function. Type is the indicates the types of output, and we use eps-regression in this case since we do not want the model to be too complex. Gamma indicates the range of the effects of the training data points, and the default value is 1/(ncols of data). We will try to tune those parameters to find the best combinations.

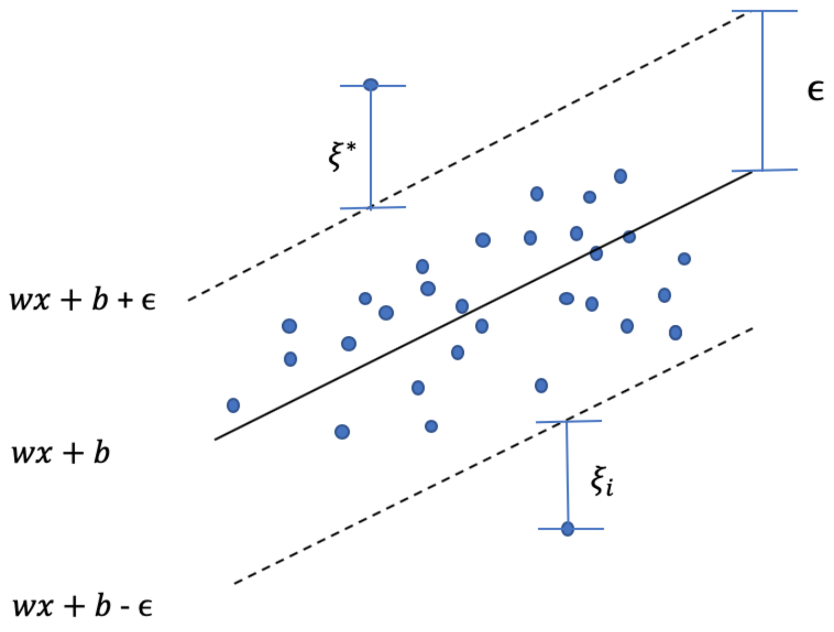


Figure 3.8: Support Vector Machine Regression

3.5 XGBoost

XGBoost is another widely used machine learning algorithm for the real-world problem, and it gives pretty satisfying performances in most cases. Based on the example given in Tianqi's paper [2], XGBoost works pretty well on large datasets and takes a shorter time to implement with higher accuracy. XGBoost algorithm uses weak learners to form a strong learner in an iterative way, which means that each model will depend on the error of previous predictions. This method fits the XGBoost trees to the residuals, and then the splitting technique will be used to determine how much better the clusters performed and pick the best threshold for each feature. By conducting this step sequentially, we can make sure that we are on the right track to the true values. The loss function we used for regression is

$$L(\Theta) = \sum_{i=1}^{n} l_i(y_i, \hat{y}_i) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{3.9}$$

In order to build trees, we try to minimize the objective function listed as Equation 3.10. Since XGBoost works as additive training, the objective function can be rewrote as Equation 3.11 or Equation 3.12 mentioned in Tianqi's paper [2].

$$Obj = \sum_{i=1}^{n} l_i(y_i, \hat{y_i}^{(t)}) + \sum_{i=1}^{t} \Omega(f_t) \tag{3.10}$$

$$= \sum_{i=1}^{n} l_i(y_i, \hat{y_i}^{(t-1)} + f_t(x_i)) + \Omega(f_t) \tag{3.11}$$

$$= \sum_{i=1}^{n} [2(\hat{y_i}^{(t-1)} - y_i) f_t(x_i) + f_t(x_i)^2] + \Omega(f_t) \tag{3.12}$$

After taking the second order of Taylor expansion and re-formulating [1], the final objective function is

$$Obj = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T \tag{3.13}$$

where

$G_j = \sum_{i \in I_j} g_i = \sum_{i \in I_j} \partial_{\hat{y_i}^{(t-1)}} l(y_i, \hat{y_i}^{(t-1)})$, $H_j = \sum_{i \in I_j} h_i = \sum_{i \in I_j} \partial^2_{\hat{y_i}^{(t-1)}} l(y_i, \hat{y_i}^{(t-1)})$, and γ is the regularization. Now, after we get the finalized objective function, we can use the approximate greedy algorithm for the best splitting because of the efficiency.

To implement the XGBoost algorithm in R , the most important step is to tune the parameters. In our case, the tuning parameters included as below [3]:

- eta: learning rate ($0 \leq \text{eta} \leq 1$)
- gamma: minimum loss reduction required. The larger gamma indicates
  the more conservative algorithm
- nrounds: number of iteration
- max_depth: maximum depth of a tree
- early_stopping_rounds: the algorithm will stop after k rounds if no improvement
- subsample: the ratio of the training observation will be used

[1] Please refer to Tianqi's paper for the detailed derivation process [2]

CHAPTER 4

Results

In this chapter, the predictions from all methods will be compared. Also, most of the methods require tuning parameters, so the values of parameters inside of each algorithm will also be given. The final results are shown in the table below.

Table 4.1: Ridge Regression Results

Method	R^2 train	rmse train	mae train	R^2 test	rmse test	mae test
Ridge	0.627	0.5035	0.3723	0.6178	0.5128	0.3762
Ridge with Kmeans	0.6770	0.4668	0.3452	0.6731	0.4743	0.3513
Random Forest	0.9346	0.211	0.1475	0.7255	0.4285	0.3075
SVR	0.7626	0.4001	0.2617	0.6976	0.4561	0.3248
XGB	0.8611	0.3062	0.2265	0.7304	0.4306	0.3087

For Ridge regression, we already know that the optimal λ is 0.668101e-06. As the result shown in Table 4.1, both training and testing R^2 are relatively low. RMSE and MAE may also be unsatisfying. The reason for this might be that the ridge regression did not capture the non-linearity of our dataset. For ridge regression with K mean, we expect it can help to group the observations with high similarity to reduce the errors in our prediction. After tuning parameters, we notice that the best result given when k is 10. Also, Table 4.1 shows the results of ridge regression with k means indeed improves the model performance compared with ridge regression. The best predictions of random forest are obtained when "mtry" is 10, and building model uses all observations in the training set. For support vector machine regression, we get the best predictions when γ is 0.06, and ϵ is 0.1. As we mentioned in the previous chapter, XGBoost gives satisfying predictions in most cases. This method involves tuning process for 4 parameters, which are “eta”, “gamma”, “nrounds”, and “subsample”. We find the best result when eta is 0.03, gamma is 0.36, nrounds is 500, and subsample is 0.6.

When we compare all three indicators, R^2, RMSE, and MAE. We may notice that random forest has the best performance on the training dataset because R^2 is 0.9346, which means the model can explain 93.46% of our training data. Also, it gives the lowest RMSE and MAE among all five models. However, for the testing set, XGBoost can explain more data than random forest, and both of them have small RMSE and MAE. I will choose XGBoost as the model with the best performance for two reasons. The first reason is that based on all three indicators, XGBoost has the highest R^2 and small RMSE and MAE. The second reason is that XGBoost may require a shorter time than random forest for implementation. If we want to use the algorithm to help hosts to set up prices, we would never want them to wait for too long to get the result. Overall, XGBoost gives satisfying predictions.

In Figure 4.1, the histogram shows the difference between predictions and true value in the testing set. We can see that our predictions are more centralized compared with the true values. More than that, we have more prediction values lays in the interval between 4.2 to 4.4 for ridge regression while more predictions are between 4 to 5 for ridge regression with k means. Also, for both methods, there are fewer predictions in the interval from 2 to 4, so these methods might be improved in predicting low responses. Now, for the histogram of the random forest and XGBoost, we may notice they have better performance when the responses are small compared with other methods, and they have more predictions around 5.

Also, the densities of predictions and true responses are shown in Figure 4.2. We can see the density of true price is like a bell shape while all the densities of predictions have more than one peak. The interesting fact is that we can see the first peak becomes lower as the model performance becomes better.

After choosing XGBoost as the model with the best performance, we would like to further investigate the difference between our predictions and true rental prices. Since the purpose of this project is to provide the reasonable suggestions of rental prices to hosts, we want to analyze if the model could identify the observations either overpriced or underpriced. The summary statistics of the difference between prediction and true prices listed below.

Table 4.2: Summary of difference between predictions and true prices

Min	1st Qu	Median	Mean	3rd Qu	Max
-3.49885	-0.21704	0.01674	-0.00401	0.24139	2.66470

We may notice that the minimum difference is -3.49885, and the maximum difference is 2.66470. We first look at the overpriced observations. For example, observation No. 16460 has the largest absolute difference. We first select the observations in the testing dataset which have the similar condition as the observation No.16460, and the result shows that the log of rental prices of those observations are between 4 and 5 while the log of the price of No.16460 is 8.022897. The same situation happens to other overpriced observations. Based on the information of No.16460 we obtained, the property is a studio located in San Francisco with a rental price $3050. There is nothing special about this property, so the price should not be so high. Unfortunately, we do not have enough information conclude the reason why the price is overpriced, but there are some possibilities. For example, the property is close to landmarks, or the property might be spacious. The price can be reasonable if we want to rent a house next to Disneyland, or it has a great view. The additional information is required to find the root causes.

For underpriced observations, we first look at the observation No.5379. The log of the true price is 2.3035. Similarly, we select the observations with the similar condition in the testing set. The log of prices for other observations fluctuates around 4.967. In the raw data, it indicates observation No.5379 has $150 for cleaning fee which is much higher than other properties with the same condition, so we may guess that host use this as a strategy to attract more guests. Even though the price is extremely low, a high cleaning fee makes sure that it is still profitable. There are also other possible reasons why this property underpriced. For example, the neighborhood is not safe. To find the root causes, we may need additional information.

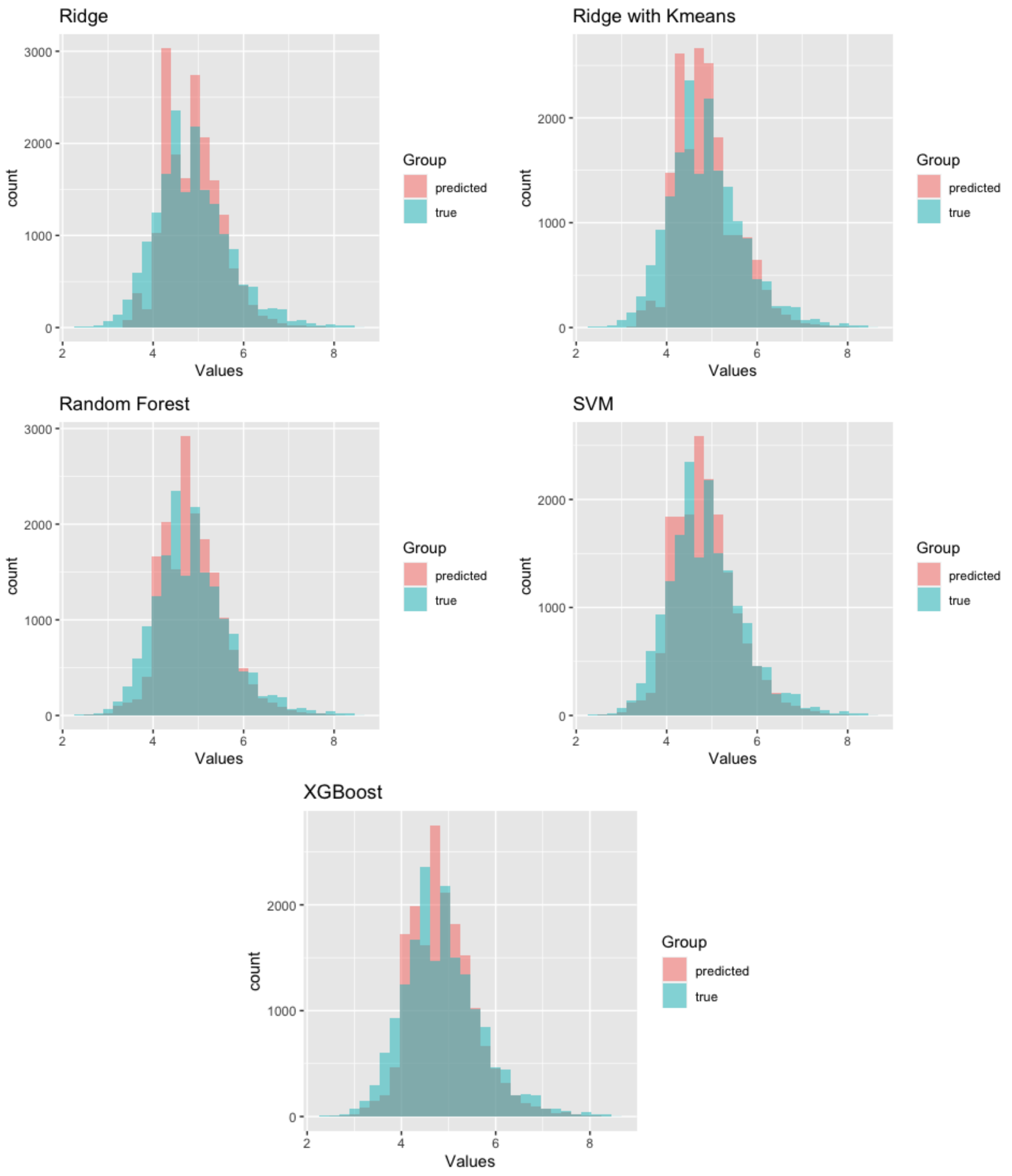


Figure 4.1: Predicted log price distribution

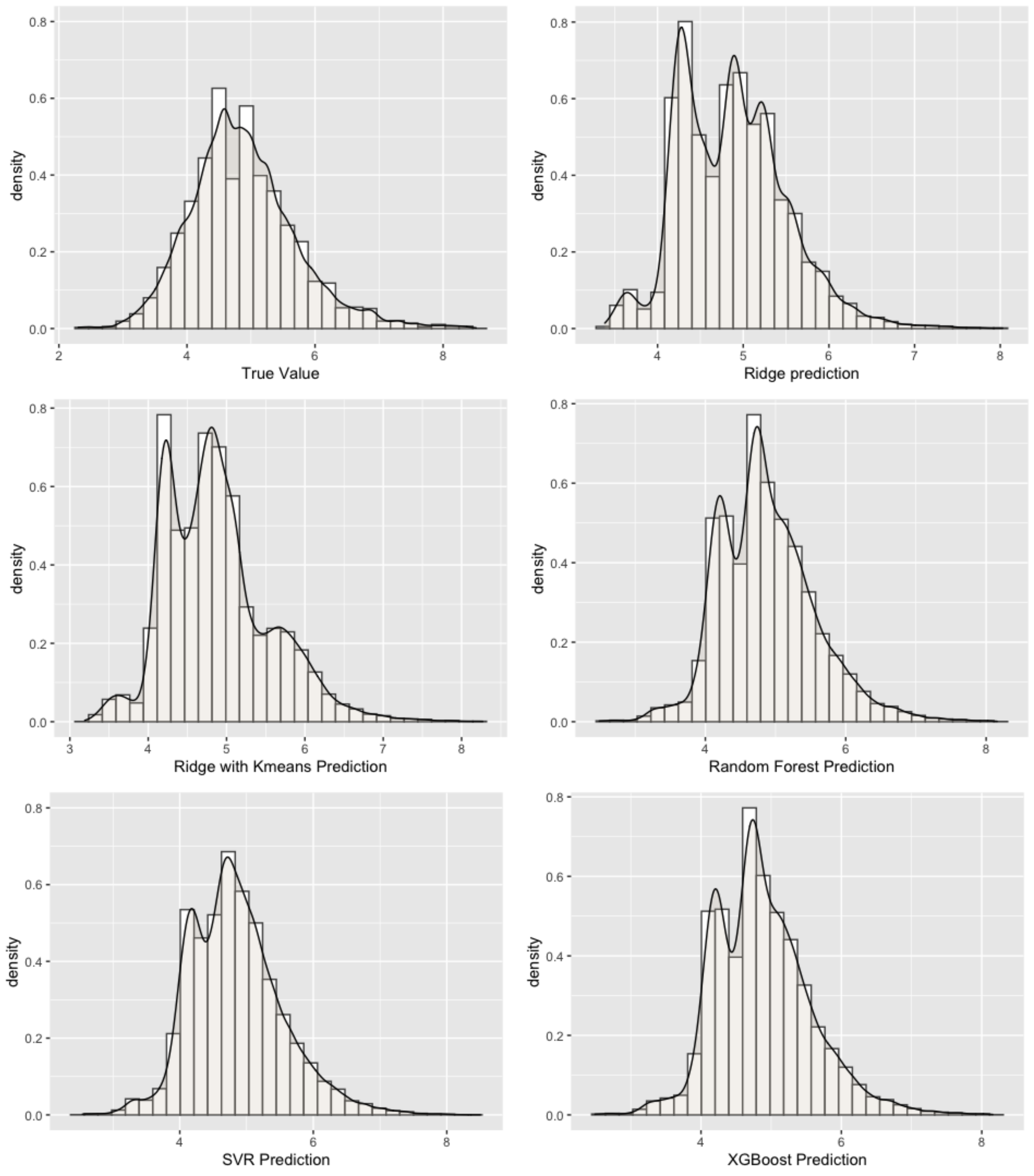


Figure 4.2: Predicted log price distribution

CHAPTER 5

Conclusion and Enhancement

5.1 Conclusion

As Airbnb becomes a popular option, more people consider renting properties on Airbnb as their businesses. It might bring a question if there any suggestion for the hosts, especially, the host with less experience to set up rental prices. If we can come up with some strategies to give them suggestions, it not only benefits the hosts but also attracts potential hosts for Airbnb.

Because of this motivation, we start this project. Since we include all datasets from California, the data cleaning step becomes extremely tedious because it involves a great deal of data formatting and imputation. Also, for this practical project, exploratory analysis and data visualization play important roles. Because of these two steps, we can have a better understanding of features and finish the initial feature selection. During the algorithm implementation step, we try to use different indicators to evaluate model performances. As we expected, XGBoost gives the most satisfying predictions. Even though the rental prices are totally set up based on hosts' wills, we try to get the general trend of rental prices on the marketplace to give some reasonable suggestions about rental price, and then the host can make any adjustments based on the suggestions as they need.

5.2 Enhancement

At the end of this project, we may also notice there are some enhancements we can make in the future. As it is mentioned in Yang Li's paper [7], we can include the information of landmarks for prediction. For example, the distance between nearby landmarks and property can be used as a new feature for modeling. The seasonality can be another important feature since the rental price will be higher in some specified date, such as holiday seasons. In addition, the supply and demand are also affected the rental price, and we may need to get this information from external sources. By including more relative features, I believe model performance will be improved.

REFERENCES

[1] Airbnb statistics [2020]: User & market growth data. Retrieved March 25, 2020, from `ipropertymanagement.com/research/airbnb-statistics`.

[2] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA, 2016. Association for Computing Machinery.

[3] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, Rory Mitchell, Ignacio Cano, Tianyi Zhou, Mu Li, Junyuan Xie, Min Lin, Yifeng Geng, and Yutian Li. *xgboost: Extreme Gradient Boosting*, 2019. R package version 0.90.0.2.

[4] How cancellations work for stays, 2020. from `https://www.airbnb.com/home/cancellation_policies`.

[5] K. Kim. Ridge regression for better usage, 2019. `https://towardsdatascience.com/ridge-regression-for-better-usage-2f19b3a202db`.

[6] Tania Kleynhans, Matthew Montanaro, Aaron Gerace, and Christopher Kanan. Predicting top-of-atmosphere thermal radiance using merra-2 atmospheric data with deep learning. *Remote Sensing*, 9:1133, 11 2017.

[7] Y. Li, Q. Pan, T. Yang, and L. Guo. Reasonable price recommendation on airbnb using multi-scale clustering. In *2016 35th Chinese Control Conference (CCC)*, pages 7038–7041, 2016.

[8] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[9] Gemma Nedjati-Gilani, Torben Schneider, Matt Hall, Niamh Cawley, Ioana Hill, Olga Ciccarelli, Ivana Drobnjak, Claudia Gandini Wheeler-Kingshott, and Daniel Alexander. Machine learning based compartment models with permeability for white matter microstructure imaging. *NeuroImage*, 150, 02 2017.

[10] P. Rezazadeh, L. Nikolenko, and H. Rezaei. Airbnb Price Prediction Using Machine Learning and Sentiment Analysis. 2019.

[11] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[12] Us zip code latitude and longitude (2018, february 9). Retrieved February 27, 2020, from `https://public.opendatasoft.com/explore/dataset/us-zip-code-latitude-and-longitude/export/?refine.state=CA`.

[13] Vladimir Vapnik. *The nature of statistical learning theory.* Springer science & business media, 2013.