Airbnb Property Rental Price - Prediction Modeling

Anna Hudgins

Pulin Kulshrestha

Venkata Pavan Tej Deevi

Yiwei Chen

SCMT 650, Section 600, Group 4

Contents

1. Executive Summary

Goals and Objectives:

The goal of our study is to build a predictive model that explains how renters value their properties and set their rental prices in Austin, TX.

Observed Data:

Data was gathered from 'Inside Airbnb'. The data collected was based off Austin rentals and taken from 2017.

Methodology and Approach:

1. Clean and transform the data. Before building the actual model, it was necessary to clean the data to make it more coherent and remove erroneous input.

2. Conduct best subset and forward/backward stepwise to identify the most significant and predictive variables for model reference

3. Conduct and analyze 4 predictive models:

Predict a quantitative outcome and identify significant predictors.

- Linear
- KNN
- Decision Trees
- Random Forest

4. Compare all four models Test RMSE's in order to identify the best model.

5. Validated each model by validation set approach using training and testing data sets.

Conclusion:

The random forest model was selected as the best model to predict the price based on its smallest test RMSE, which is also comparable to the results of other Airbnb predictive models. Preparing

the data was a time-consuming, tough and significant procedure before we could run the model and get reasonable results. Empirical data sets are always not as strong and accurate.

Recommendation:

Results of various predictive models suggested that number of bedrooms in a property is a strong predictor of the price of the property. This is followed by the amount of cleaning fees charged by host per day.

2. Introduction

Airbnb is a growing industry in which many people have heard of or previously used themselves. Are the hosts properly valuing their rentals or are renters paying too much or too little for their chosen rental property? Airbnb is an online marketplace and lodging service that allows people to rent their properties for short term leases or vacation rentals. The rental property cost is determined by the host and the company receives a percentage of the stay as a service fee. Airbnb has a predictive tool for hosts to utilize in order to set a competitive price for their rental property. The tool is based on a description of the property, but the host ultimately determines the price. The goal this study is to create a predictive model based off Airbnb data to better understand how hosts actually price their property rentals.

3. Objectives

The goal of our study is to build a predictive model that explains how renters value their properties and set their rental prices in Austin, TX. In order to reach this goal the necessary objectives include:

- Analyze and understand data for rental prices
- Identify key predictors and drivers for Airbnb rental prices
- Understand the direction and weight of the significant predictors

4. Data Collection and Analysis

In order to predict Airbnb price rentals in Austin, TX. data was collected from Airbnb website. The data obtained required cleaning and transformation before models could be run. The dataset

was eliminated to 22 predictors based off predictors that would impact price. It included both continuous and categorical variables.

To clean the data price outliers and insignificant values (NA) were eliminated so that the results were not skewed. The data also contained observations not located in Austin, TX. so those observations were also eliminated. Predicting price (a quantitative variable) the categorical predictors needed to be transformed. Variables property type, rental type, and bed type were therefore transformed into dummy variables. The categorical variable property type was also grouped into smaller segments. They were further categorized into three categories: Home, Apartment, or Other to reduce the number of dummy variables for that category.

- Property Type: house (dummy), apartment, and other
- Rental Type: Entire home/apartment (dummy), shared room, and private room
- Bed Type: real bed (dummy), couch, futon, air bed, and pullout-sofa

In order to account for specific location in Austin, a visualization was initially made in Tableau to understand prices by zip code. The tableau representation depicted our hypothesis that rental prices would be more expensive the closer they are to the center point of Austin. Using the Latitude and Longitude variables a distance measure was created as a new variable. The distance formula represents the larger the score the farther the rental was from the center of Austin. Distance Formula= 2 * 6371000 * ASIN(SQRT((SIN((LAT2*(3.14159/180)-LAT1*(3.14159/180))/2))^2+COS(LAT2*(3.14159/180))*COS(LAT1*(3.14159/180))*SIN(((LONG2*(3.14159/180)-LONG1*(3.14159/180))/2))^2))

Figure 1 – Variation of average property price with property type

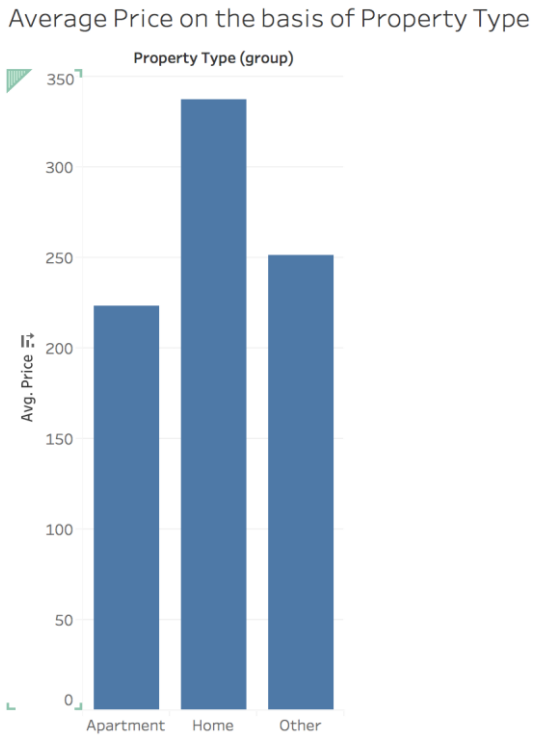


Figure 2 - Variation of average property price with room type

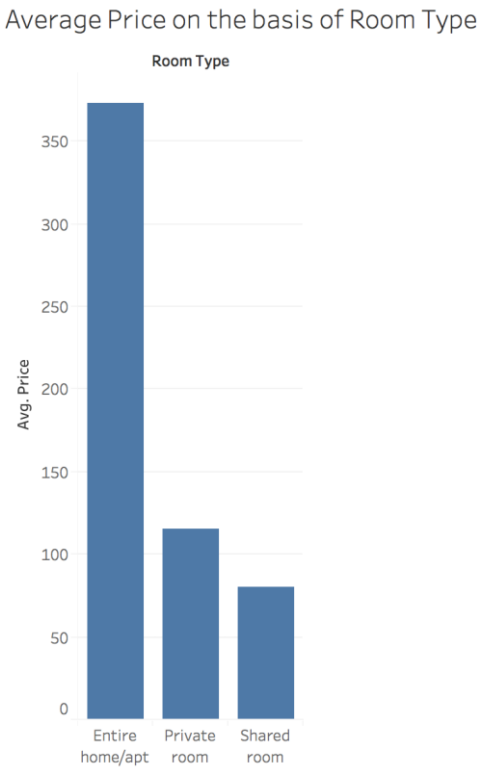


Figure 1 shows the variation of average rental prices among different property types like Apartment, home or other. Price is depicted on y axis and property type on x axis of the plot.

Figure 2 shows the variation of average rental prices among different room types like if the rental is entire home, a private room or shared room. Average price is on y axis and room type is shown on x axis.

Figure 3 - Average rental price by latitude and longitude in Austin

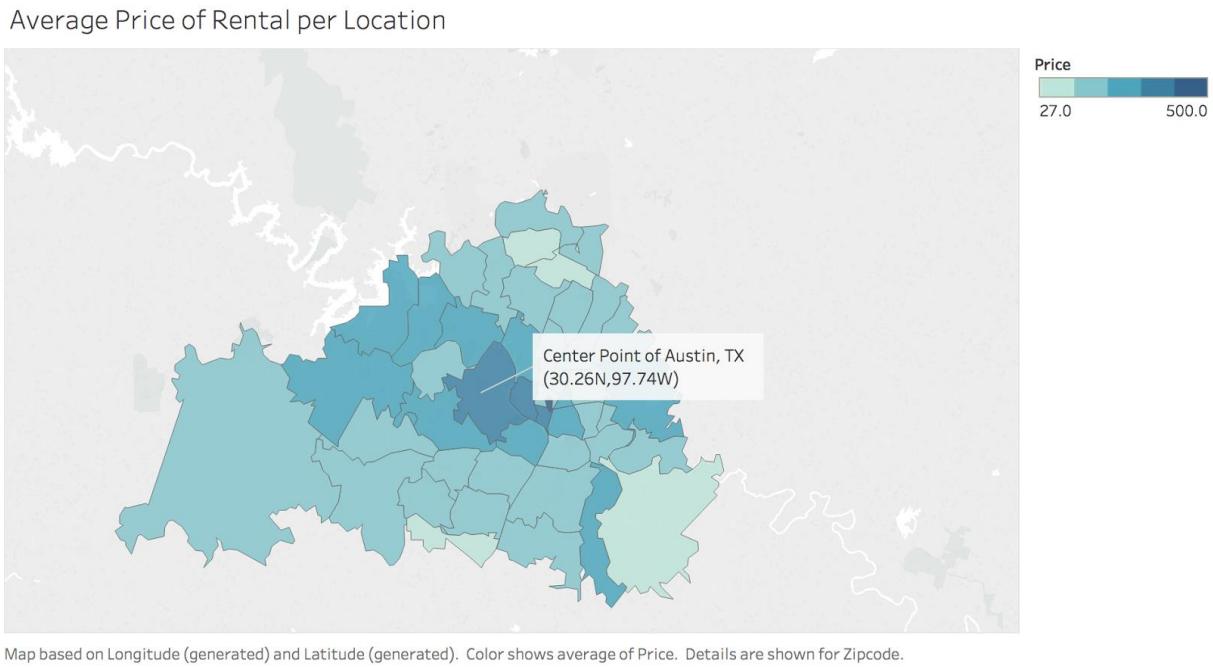


Figure 3 represents the average price of rentals measured by latitude and longitude. The prices are displayed through a heat map with five colors. The darker the shade of blue, the more expensive the average price of the rental property. The center of Austin is (30.26N, 97.74W) which was used at the base measure for all other locations.

5. Literature review

Before we started working on this project, we did a search on existing Airbnb price prediction models. We were able to find few, and in fact, Airbnb themselves invested in such a project. Although we didn't aim at beating the accuracy of Airbnb owned price predicting algorithm, we aimed at creating a model better than or at least close to the other models we found online. Below is the comparison of Test RMSE in one such projects. Figure 4 shows the plot between RMSE (train and test) and different models used in one of the Airbnb price predictive modeling projects.

Figure 4 - Train and test RMSE for different models

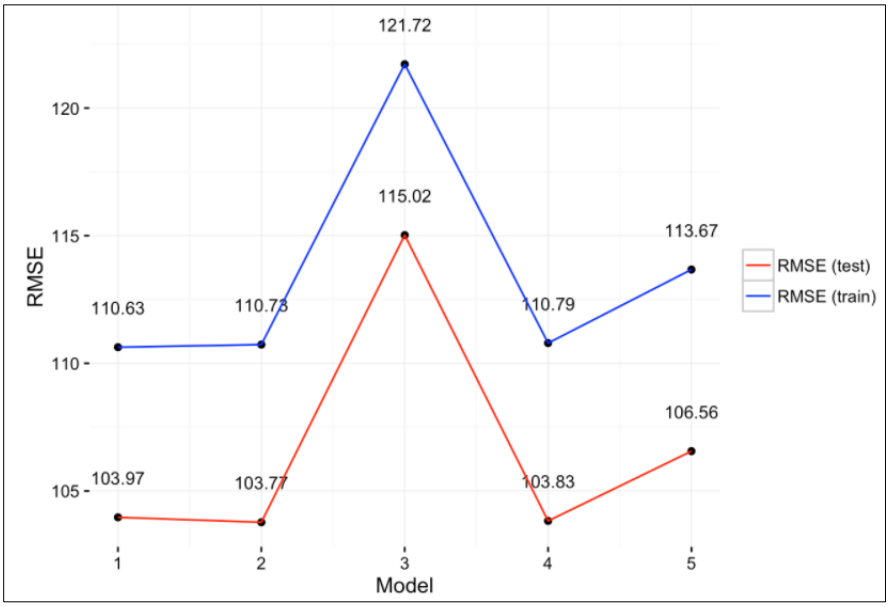


Figure 4 shows the train and test RMSE variation with different models for one of the modelling project we found online.

In our project, we tried to achieve an RMSE close to 100 and also aimed at finding significant predictors that can help predict the rental prices as accurate as possible.

6. Models

a. Variable Selection

Approached the data set using best subset, forward stepwise and backward stepwise in order to get a better understanding of the most important predictors before running the models. Highest adjusted R squared is plotted against model size.

Figure 5 - Adjuster R squared vs Number of variables

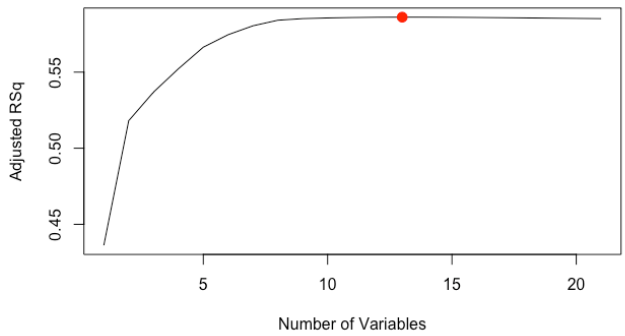


Figure 5 measures adjusted R squared by number of variables. This plot depicts that thirteen~ variables is optimal for best model (red dot). However, to measure use for best subset model adjusted rsquare, Cp, and BIC were all plotted to confirm and measure how many variables is optimal for the models.

Figure 6 - Cp vs Number of variables

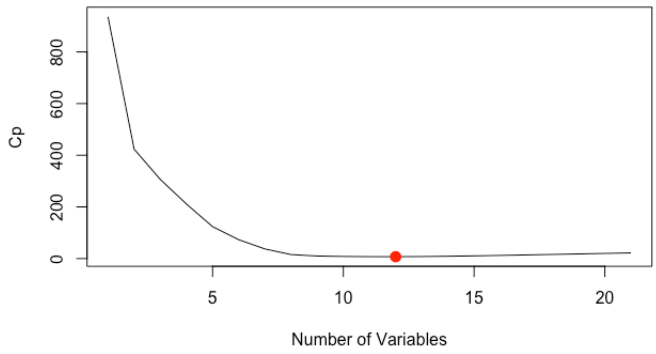


Figure 6 represents the measurement of Cp by number of variables. The plot for Cp was similar to adjusted rsquare measurements and indicated variable count for best model around thirteen (red dot), but the measurements begin to plateau around eight variables.

Figure 7 - BIC vs Number of variables

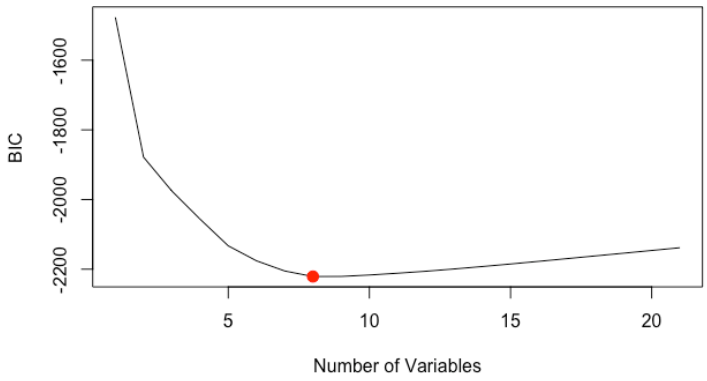


Figure 7 represents the BIC plot against number of variables which indicates that lowest BIC is at eight variables for best model. Therefore, after analyzing all plots it was determined that the variables for best model was eight variables. These eight variables remained the same for best subset, forward, and backward stepwise, which include: distance, accommodates, bathrooms, bedrooms, amenities_count, cleaning_fee, and reviews_per_month.

Figure 8 - Most significant predictors and their coefficients

```
      (Intercept)          Distance      accommodates         bathrooms          bedrooms
      70.26379066       -9.94078558        12.71010263       59.28503977       21.95524670
Amenities_Count  security_deposit      cleaning_fee reviews_per_month
      -3.48013809        0.05259995         0.88758444      -11.29505839
```

Figure 8 represents the output for best subset, forward and backward stepwise variables and coefficients. These variables were then used as a reference to compare the outputs of the four models conducted.

b. Model 1: Linear

Linear model was created measuring price against all 21 variables. Significant predictors were measured at the 0.001 level. Just as the best subset and stepwise models predicted, the linear model also contained eight significant predictors which were also the same variables. Their coefficients include: Distance (-10.7), Accommodates (11.4), Bathrooms (69.7), Bedrooms (24.8), Amenities (-4.2), Security Deposit (.06), Cleaning Fee (.82), and Reviews Per Month (-10.2). Adjusted rsquare measured at 0.60 explaining 60% of variation by only the independent variables that actually affected the price variable. Validation for the linear model was 50/50, training 50% of the data and testing with the other 50% of the data. Test RMSE measured at 128.15 which is higher than ideal. Meaning the predicted price for a rental may vary by $128.15 than the actual rental price.

Figure 9 - Observed price vs predicted price for linear model

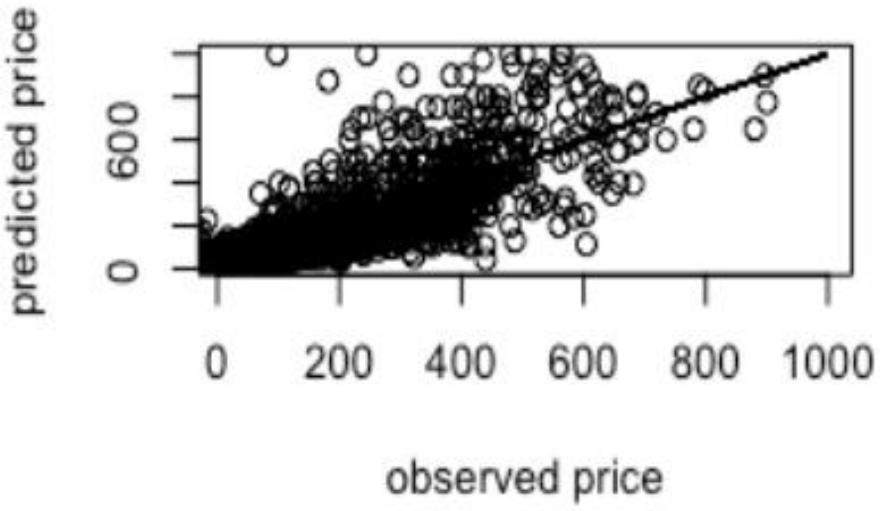


Figure 9 represents the predicted verse observed price.

Linear plots were then conducted to measure for model assumptions.

Figure 10 - Plots for linear model assumption validation

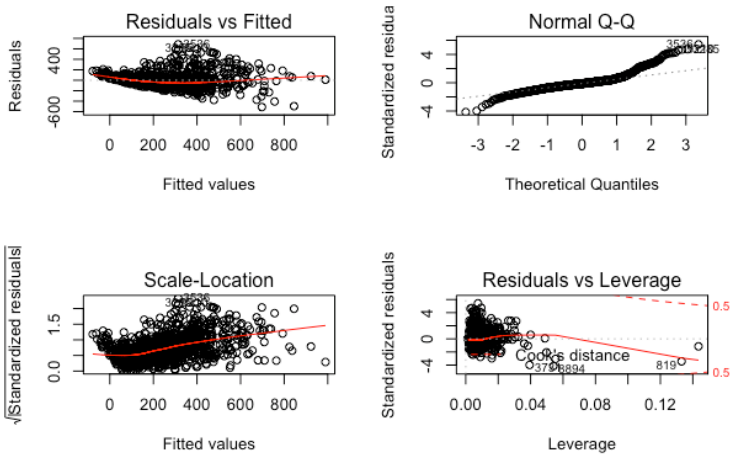


Figure 10 displays the four plots: Residuals vs. Fitted, Normal Q-Q, Scale- Location, and Residuals vs. Leverage. The residual fit model residuals are equally spread around horizontal lines without distinct patterns, indicating that we do not have a non-linear relationship. The Q-Q Residuals are lined well on the straight dashed line. The Standardized Residuals plot shows homoscedasticity with equal points spread around horizontal line. Lastly, the Residuals vs. Leverage plot does not have points that lie outside of Cook's distance. All assumptions are therefore met and the linear model can be used.

Strengths and weaknesses of the model:

Linear model easily identified the most significant predictors with their coefficients impact and direction on price. This allows the results to be easily interpreted. With an adjusted rsquare of 0.60 the model is able to explain 60% of variance in data. A weakness to linear models is that it is limited to numeric data; therefore, categorical variables had to be transformed.

c. Model 2: KNN

KNN stands for K Nearest Neighbors. For a given value of K, this model tries to predict the value of a test observation as the average value of the K nearest train data set neighbors. Nearest neighbors are decided using Euclidean distance between the test observation and train observations.

Reasons for considering KNN model for our project:

1. We aimed at predicting a value (price) for a given set of predictors and KNN can be applied as a regression model.
2. As the definition of KNN model says, the prediction depends on the quality and quantity of train data set. Since we had a considerable amount of data, we were able to classify a decent amount into train data set (close to 2000 observations after KNN specific cleansing) and so KNN stood as good choice.
3. Although KNN model takes longer amounts of time compared to other models for larger data sets and higher K value with more predictors, since we did not have a time constraint, KNN did not pose any drawbacks.

Step 1 – KNN specific data cleansing

After initial data cleansing of creating dummy variables and removing observations that don't belong to the Austin data set, we were left with data that still had blank values in several predictors. Since KNN model cannot compute Euclidean distance when a certain value is N/A, we had to exclude the entire row from data set using 'complete.cases' command.

Step 2 – Creating training and testing data sets

Since KNN model predicts better with larger training data sets, we used 75:25 for diving train and test sets. Seed is used as 25 to sample the data.

Step 3 – Predictor scaling

Since KNN calculates Euclidean distance to find the K nearest neighbors, so it only makes sense to scale all the predictors to one common scale so that the distance value is skewed because of one predictor value range being larger than the others. For example, 'Distance from central Austin' values range from 0.03 up to 18 while 'Dollar amount of security deposit' values range from 95 till 5000. Clearly, security deposit value dominated the distance calculation if not normalized. So, we used command 'Preprocess' to scale the predictors to a common scale.

Step 4 – Predictor subset selection

When KNN is run using k = 10 and with all the predictors in the data set, we got a considerably high Test RMSE (127.9). Hence, to bring down the residual error between predicted and observed values and thereby reducing test RMSE, we used only those predictors which showed as significant predictors when run through forward and backward subset selection. These predictors are –

- Distance from Central Austin
- People accommodates
- Number of bathrooms
- Number of bedrooms
- Number of amenities provided
- Amount of security deposit
- Cleaning fee charges
- Reviews per month received

When KNN model is run using only the above significant predictors, test RMSE is further reduced, but only a slight reduction (127.28).

Step 5 – Selecting the best value for K

As an initial guess, we looped K in the range of 1 to 45 (since 45 approximately is square root of 2000, the number of train observations and we read that optimum K should lay below square root of number of train observations). For every K in this range, we built the model using training observations and calculated Test RMSE on the model. This gave a minimum RMSE of 121 at K = 36. Although this is a considerable reduction in RMSE, we did not see global minima in the K vs Test RMSE plot. Or at least, we wanted to confirm if the minima is global. So, increased the maximum value of K to 100 and re ran the code. Although we ended up with the same observation, this time the K vs Test RMSE plot is a close match to ideal plot where Test RMSE should reduce as we increase K till a certain value of K after which Test RMSE should start increasing with increasing K thus confirming that the minimum Test RMSE obtained is the global minimum.

Figure 11 - Variation of Test RMSE with change in K value

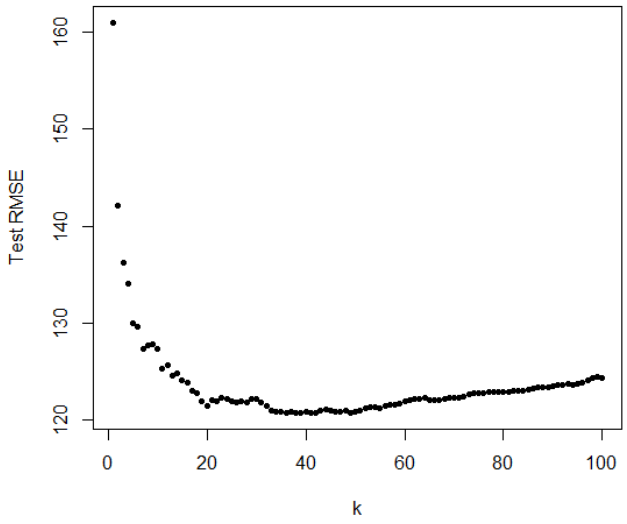


Figure 11 shows variation of Test RMSE with increase in K from 1 till 100.

Thus, with a minimum Test RMSE of 121 at K = 36, we were able to further reduce residual error as compared to linear model. Since optimum K = 36 which is in accordance with bias –

variance balance where K is neither 1 nor equal to number of train observations (or very high like 99 or 100), we can say that KNN model turned out to be a decent success although the Test RMSE value is as low as we expected.

Figure 12 - Observed price vs predicted price for KNN model

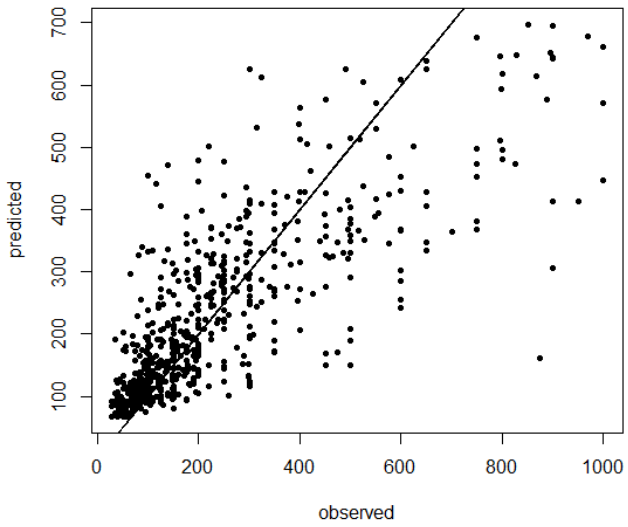


Figure 12 is plotted between observed Airbnb room prices we got from the data set and predicted prices using KNN model.

Strengths and weaknesses of KNN:

- KNN model was very simple to understand and implement. As we were implementing each step mentioned above, we were able to see changes in Test RMSE and were able to understand why the model is behaving the way it is.
- As we increased the training data set size, results improved and, as mentioned conceptually, KNN does work better with larger train sets.
- Along with better results came slowness. With increase in train data set size and maximum range of K, model started responding slowly.
- Test RMSE results improved when subset selection approach was applied. Thus, KNN is prone to noise data.

d. Model 3: Decision Trees

Decision tree is a type of supervised learning algorithm (having a pre-defined target variable) that is mostly used in classification problems but can also be applied on regression problems. It works for both categorical and continuous input and output variables.

Working with Decision trees, we built a decision tree based on the original dataset without any feature engineering to set up a benchmark value. First of all, we will divide the data into two datasets -Training and Testing. Ratio for both dataset will be 50:50. Now we will remove and ID and host_ID become huge burden to R and interferes with the train function in the caret package. Also, ID and host_ID are just for the sake of uniquely identify the observation. Therefore, we created a new dataset, which contains the exactly same information as airbnb (after cleaning) minus ID's.

Decision tree is created using all the predictors and excluding the NA values. First, the model is executed on the training subset. In the below figure, we can see that the variable used in the tree construction are as follows: -

- Bedrooms
- Reviews_per_month
- Cleaning_fee
- Distance
- security_deposit

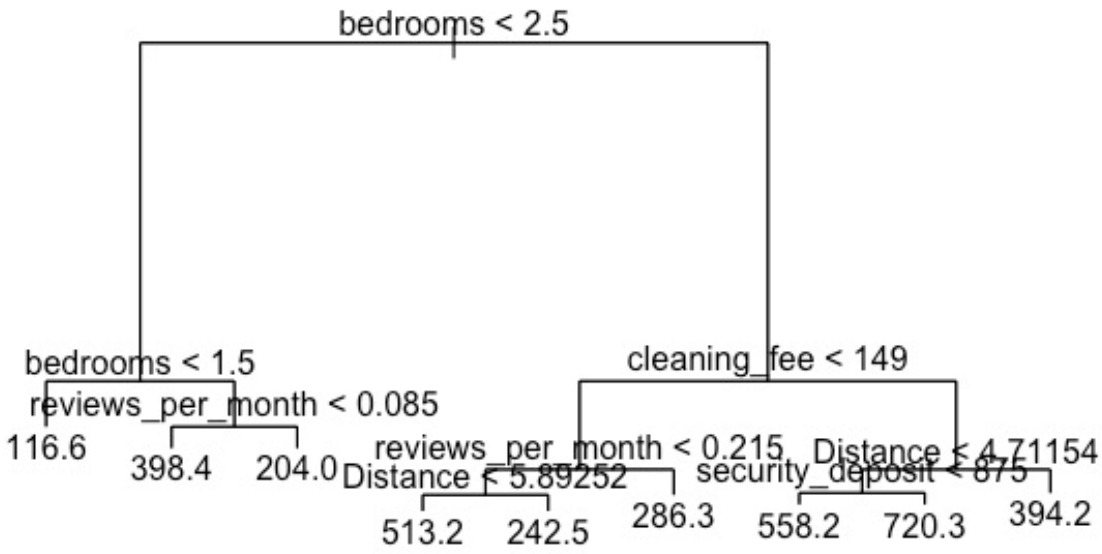
Number of terminal nodes created in the tree is 9. As we can see from the figure below, Residual mean deviance is very high of around 15200. In the context of a regression tree, the deviance is simply the sum of squared errors for the tree.

Figure 13 - Decision tree model parameters

```
Regression tree:
tree(formula = price ~ ., data = airbnb, subset = train)
Variables actually used in tree construction:
[1] "bedrooms"          "reviews_per_month" "cleaning_fee"      "Distance"
[5] "security_deposit"
Number of terminal nodes:  9
Residual mean deviance:  15200 = 19590000 / 1289
Distribution of residuals:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-403.20  -61.34  -17.56    0.00   40.80  758.40
```

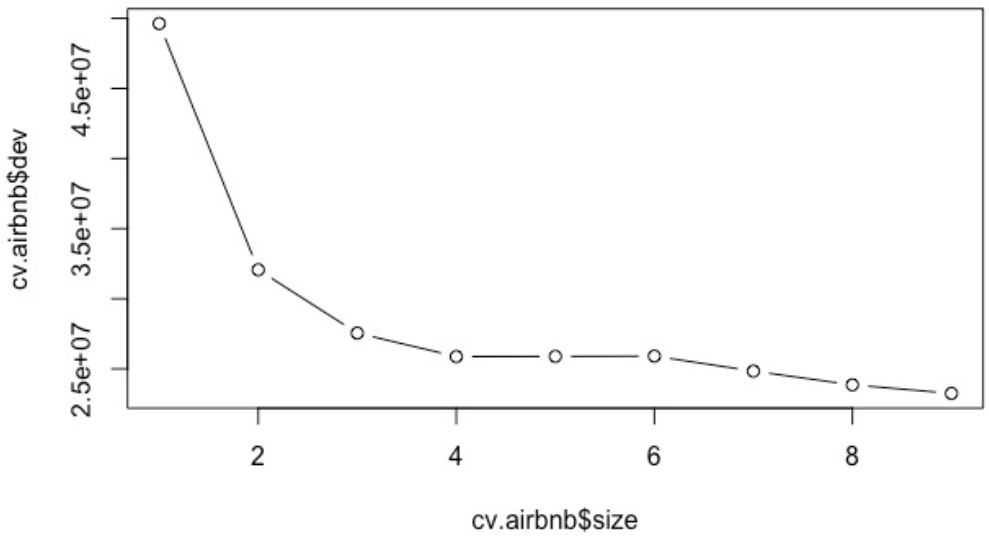
Now after plotting the tree, we can check the price classification based on other predictors. The tree predicts a median house price of $720.3 for larger homes greater security fees and is around the central area of Austin. (Distance <4.71).

Figure 14 - Decision tree



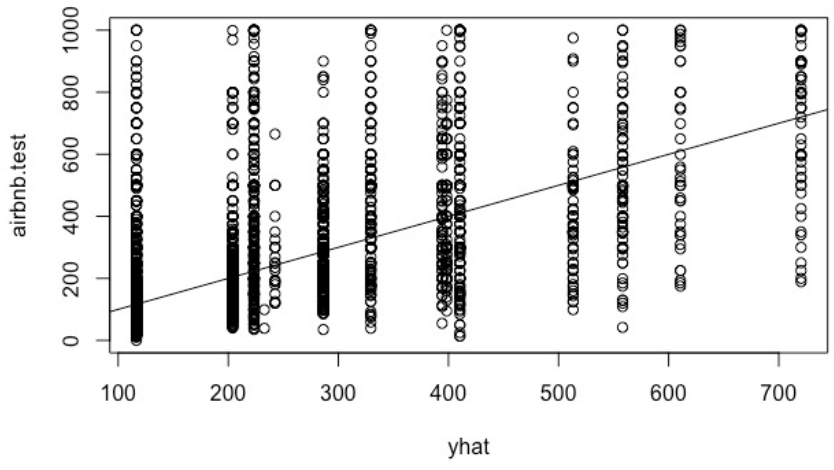
After calculating the test RMSE using the training dataset, we found out the RMSE(test) is around 173.01. This is very high comparing to the other models. So, we opted to prune the model. After plotting the cost complexity pruning (figure 15), we found out that 9 was the best trees size. To try something best out of the model, we ran the code for tree size of 8 and 7 but as expected they turn out to increase the test RMSE.

Figure 15 - Variation of deviance with increase in tree size



We also plotted the graph (figure 16) to visualize the predictions relative to the observations as seen in the figure below. X axis contains predicted prices and y axis the observed prices.

Figure 16 - Predicted vs observed prices using decision tree model

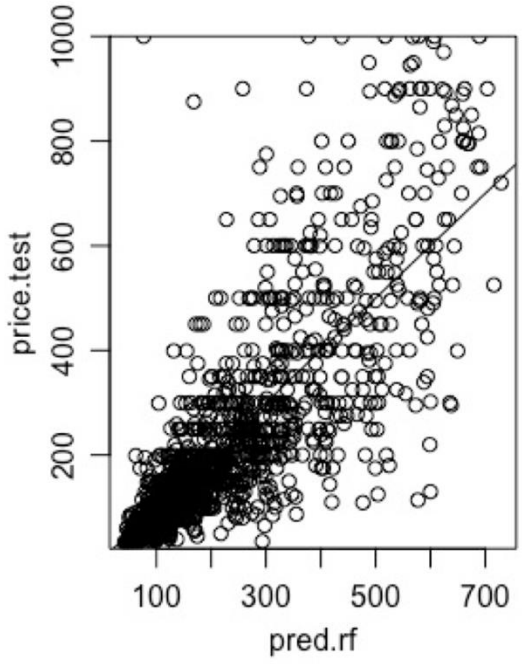


In keeping with the cross-validation results, we can use the unpruned tree to make predictions on the test set. Although the values are not the best till now but we can use it further to check in Random Forest or bagging model techniques.

e. Model 4: Random Forest Model

We used seed 25 here as well to ensure the consistency and used 50% of data as training set and 50% as test set. We first used the training set to train the bagging model, using the random Forest package in R. The bagging model is a special case of a random forest when all predictors are considered. The tree number was set as 1000. Then we used the testing data set to test this bagging model and do the validation, and then got a test RMSE of 122.25, which is pretty high. From the plot, we can see that the model works well especially when predicting lower prices. When it comes to some high observed prices, the model failed to predict them correctly.

Figure 17 - Predicted vs observed prices using random forest model



The second step was to tune the bagging model by setting the number of predictors to be randomly selected and the result of the best number is 12. When we set the mtry as 12, the test RMSE has a slight reduction to 120.66.

Boosting was another approach we used to improve the prediction power from a decision tree. Boosting works similarly to bagging except that the trees are grown sequentially. In a boosting model, each tree is grown using information from previously grown trees.

We used the gbm package and the gbm() function in R to improve the model. To train the model, we set the tree number as 5000 and the lambda as 0.001. Setting the depth as 6,8 and 10, we came up with results as following:

Boosting(trees=5000): test MSE= 13983.05 Test RMSE= 118.25 (Depth =6)

Boosting(trees=5000): test MSE= 13877.20 Test RMSE= 117.80 (Depth =8)

Boosting(trees=5000): test MSE= 13776.11 Test RMSE= 117.37 (Depth =10)

The test RMSEs here are better than the last random forest model as well as the bagging model and the best RMSE is 117.37 when depth is 10.

For a random forest regression model, incremental MSE, incremental node purity and VIMP can be used to identify the important variables easily. For an important variable, if it is replaced with random noise, you would imagine MSE with the faulty data to increase. IncMSE (Incremental MSE) for a particular variable is how much the MSE will increase if the variable is completely randomized. For regression models, and node impurity is usually taken as the variance in a node. The most popular VIMP method uses a prediction error approach involving “noising-up” each variable in turn. VIMP for a variable xv is the difference between prediction error when xv is noised up by randomly permuting its values, compared to prediction error under the observed

values. VIMP measures are shown using bars to compare the scale of the error increase under permutation.

To identify the important features, we applied importance() function, varImpPlot() function as well as plot(gg_vimp()) function in R. To use the ggvimp() function, we should first install the ggRandomForest package. (two images of %IncMse and VIMP)

Figure 18 - Increase in MSE due to predictors

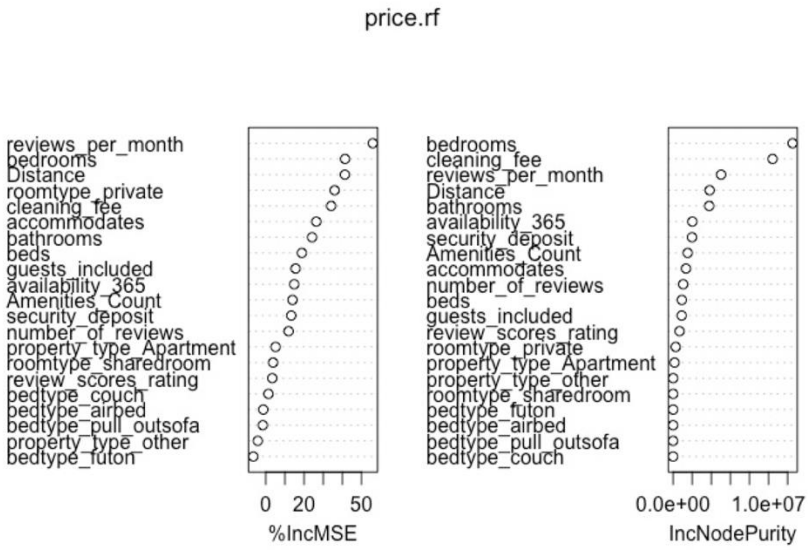
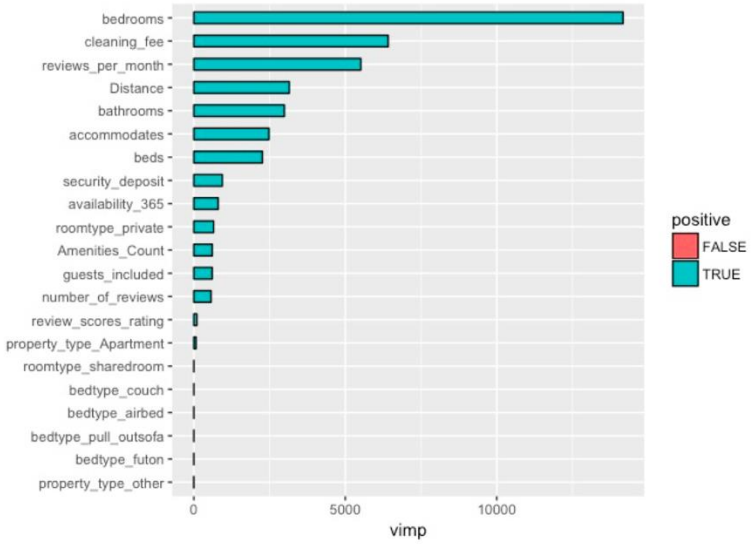


Figure 19 - Importance score for predictors



From the plots above, we came up with seven important variables including: cleaning fee, bedrooms, accommodates, reviews/mo., bathrooms, distance, and security deposit. These important variables are also the same as the significant variables of our linear model, showing that these seven variables are the key predictors to predict the prices. The least important variables are the bed types, which indicates that the bed types have little influence on the prices.

f. Model validation

We followed validation set approach to validate our models. We build the models using training data set and considered the model with least Test RMSE for each model we tried. For linear, decision trees and random forest models, we divided the initial data set into equal sets of training and testing data sets. For KNN model, we used 75% of useful observations (close to 2000 observations) as training data and rest as testing data. For sampling the data sets into training and testing set, we used a seed of 25.

7. Conclusion

Model	RMSE	Strengths	Weaknesses
Linear	128.15	Easily identified the most significant predictors and their coefficients impact and direction on price; able to explain 60% of variance in data	Limited to numeric data and we had many categorical variables that had to be transformed
KNN	121 at K = 36	Less chances of overfitting, with large training dataset and optimum K	Slow in prediction, especially for larger data sets with multiple predictors
Decision Trees	173	It was easy to identify significant variable and predict the median price based on those variables.	High Residual mean deviance and RMSE, no changes after pruning due to overfitting of data. Information biased toward attribute with more levels.
Random Forest	117.37	Easy to identify important predictors Easy to tune the model.	Large number of trees made real time prediction very slow

When we compared the test RMSE of these models, the decision tree model has the highest one which is 173 and the random forest model has the lowest one, which is 117.37. Compared to other models we used, random forest model is an effective model to predict the Airbnb price here, because this model is easier to identify important predictors and easier to be improved, and it works well with high dimensional predictors and large training sets. However, it has a disadvantage that it takes too long to do the prediction when the tree numbers are large. Based on the test RMSE of each model, we selected the random forest model as our best model to predict the price as it has the smallest test RMSE. We also concluded that the most significant variables for the prediction of price are cleaning fee, bedrooms, accommodates, reviews per month, bathrooms, distance and security deposit.

From this project, we learned that preparing the data is difficult but important. We spent a lot of time to clean, transform and prepare the data before we ran the model. To come up with reasonable results, we need to try different methods to transform the data and drop out some irrelevant predictors. Then, we should select appropriate models to do the prediction based on the characteristics of the data set and the predictors as well as the predictive goal we want to achieve. Using the same data set and predictors, we chose the best model, the random forest model, from all the model we used base on the lowest test RMSE. During the prediction, we realized that real empirical data sets are always not as strong and accurate. Lastly, even though it seems that our RMEs are too high, actually they are quite comparable when we compared them to the research of other Airbnb predictive models, which came up with test RMSEs ranging from 103.77 to 115.02.

8. Appendix

a. Data Dictionary

Column name	Description
id	Each Airbnb listing has a unique ID
host_id	Each Airbnb host has a unique host_id value
neighbourhood	Neighborhoods of the listings
zipcode	Zip Code of the listings
latitude	The latitude, as listed in the web page source
longitude	The longitude, as listed in the web page source
property_type	Type of property, listing is registered as Apartment, House, Villa and some other types
room_type	One of the three room types listed
accommodates	The number of people that the listing accommodates,
bathrooms	If listed, the number of bathrooms
bedrooms	If listed, the number of bedrooms
beds	Number of beds in the listing
bed_type	Type of bed present in the particular listing
amenities	Amenities available in the listing

square_feet	Size of the listing in square feet
price	The price listed by the host. This is always the price per night
security_deposit	Minimum amount paid as security for a listing
cleaning_fee	Minimum cleaning fee imposed while booking the listing
guests_included	Number of guests that can be accommodated on that listing
availability_365	Number of days listing is available in a year
number_of_reviews	Total number of reviews for the listing
reviews_per_month	The number of reviews per month for the listing

b. Models in R

Best Subset and Forward/Backward Model

```
#Stepwise
rm(list=ls())
library(readxl)
ATX_V1 <- read_excel("ATX_V1.xlsx")
View(ATX_V1)
sum(is.na(ATX_V1$security_deposit))
ATX_V1=na.omit(ATX_V1)
install.packages("leaps")
library(leaps)
regfit.full=regsubsets(price~.,data=ATX_V1,nvmax=22)

reg.summary=summary(regfit.full)
reg.summary
names(reg.summary)

reg.summary$rsq

# We know this is not a good measure to use for subset selection.
```

```
# Instead, let's plot all adjusted rsquare, Cp, and BIC for all the models.

par(mfrow=c(1,1))  # Set up for a single plot in a window

# plot the rss as a function of model size

plot(reg.summary$rss,xlab="Number of Variables",ylab="RSS",type="l")

# plot adjusted rsquare as function of model size
plot(reg.summary$adjr2,xlab="Number of Variables",ylab="Adjusted RSq",type="l")
maxadjrsquare<-which.max(reg.summary$adjr2)
points(maxadjrsquare,reg.summary$adjr2[maxadjrsquare], col="red",cex=2,pch=20)

# plot Cp as function of model size
plot(reg.summary$cp,xlab="Number of Variables",ylab="Cp",type='l')
mincp<-which.min(reg.summary$cp)
points(mincp,reg.summary$cp[mincp],col="red",cex=2,pch=20)

# finally, plot bic as a function of model size
plot(reg.summary$bic,xlab="Number of Variables",ylab="BIC",type='l')
minbic<-which.min(reg.summary$bic)

# plot the model with minimum bic as a red circle
points(minbic,reg.summary$bic[minbic],col="red",cex=2,pch=20)

plot(regfit.full,scale="r2")
plot(regfit.full,scale="adjr2")
plot(regfit.full,scale="Cp")
plot(regfit.full,scale="bic")
coef(regfit.full,8)

############ Forward and Backward Stepwise Selection ##############
regfit.fwd=regsubsets(price~.,data=ATX_V1,nvmax=24,method="forward")
summary(regfit.fwd)

regfit.bwd=regsubsets(price~.,data=ATX_V1,nvmax=24,method="backward
summary(regfit.bwd)

coef(regfit.full,8)
coef(regfit.fwd,8)
coef(regfit.bwd,8)

plot(regfit.fwd,scale="r2")
plot(regfit.fwd,scale="adjr2")
plot(regfit.fwd,scale="Cp")
plot(regfit.fwd,scale="bic")
```

```
plot(regfit.bwd,scale="r2")
plot(regfit.bwd,scale="adjr2")
plot(regfit.bwd,scale="Cp")
plot(regfit.bwd,scale="bic")
```

Model 1: Linear Regression

```
rm(list=ls())
library(readxl)
ATX_V1 <- read.csv("Downloads/listings_V1.csv")
View(ATX_V2_1)
ATX_V2_1=na.omit(ATX_V2_1)
attach(ATX_V2)

#linear model with all predictors, price as dependent variable
trainset<-sample(nrow(ATX_V1),(nrow(ATX_V1)/2))
train<-ATX_V1[trainset,]
test<-ATX_V1[-trainset,]
names(ATX_V1)
summary(ATX_V1)

set.seed(25)
linearresults<-lm(price~.,data=ATX_V1)
summary(linearresults)
View(linearresults)
linearresults<-lm(price~.,data=train)

par (mfrow=c(2,2))  # will show 4 plots in one graph as a 2x2 matrix
plot(linearresults) # plots four graphs in a 2x2 matrix
probs<-predict(linearresults,newdata=test,type="response")
plot(test$price,test$price, type='l',xlab="observed price",ylab="predicted price")
points(probs,test$price)

#linear model with all significant predictors, price as dependent variable
linearresults2<-
lm(price~accommodates+bathrooms+bedrooms+Amenities_Count+security_deposit+cleaning_f
ee+guests_included+availability_365+review_scores_rating+zipcode_level1+zipcode_level2,dat
a=ATX_V2_1)
summary(linearresults2)

par (mfrow=c(2,2))  # will show 4 plots in one graph as a 2x2 matrix
plot(linearresults2) # plots four graphs in a 2x2 matrix
probs<-predict(linearresults2,newdata=test,type="response")
plot(test$price,test$price, type='l',xlab="observed price",ylab="predicted price")
points(probs,test$price)
```

Model 2: KNN

```
library(xlsx)
temp = read.csv("newData/listings_V1.csv")

### KNN regression modelling ###

AirbnbAustin <- temp[complete.cases(temp),]

# creating train and test sets
set.seed(25)
smp_size <- floor(0.75 * nrow(AirbnbAustin))
trainindex <- sample(nrow(AirbnbAustin),size=smp_size)
train <- AirbnbAustin[trainindex,]
test <- AirbnbAustin[-trainindex,]

# KNN model using one predictor (number_of_reviews)
library(caret)

# Scaling data for using in KNN
#scaled <- preProcess(AirbnbAustin[,c(10:13,18:25)],method =c("scale"))

# model using all the predictors
scaled <- preProcess(AirbnbAustin[,c(1,3:22)],method =c("scale"))
train.x = predict(scaled,train[,c(1,3:22)])
train.y = AirbnbAustin[trainindex,2]
test.x = predict(scaled,test[,c(1,3:22)])
test.y = AirbnbAustin[-trainindex,2]

#model using significant predictors from linear
scaled <- preProcess(AirbnbAustin[,c(1,7,8,9,15,16,17,22)],method =c("scale"))
train.x = predict(scaled,train[,c(1,7,8,9,15,16,17,22)])
train.y = AirbnbAustin[trainindex,2]
test.x = predict(scaled,test[,c(1,7,8,9,15,16,17,22)])
test.y = AirbnbAustin[-trainindex,2]

# KNN modelling

# Tuning the KNN model for finding best K with least Test RMSE
## train set has 1000 observations, so k is (sqrt 1300) ~ 40
rmse<-rep(0,100)
for (knum in 1:100) {
  knn.fit<-knnreg(train.x,train.y,k=knum)
  rmse[knum]<-sqrt(mean((test.y-predict(knn.fit,test.x))^2,na.rm=TRUE))
}
```

```
print(paste("Miniminum RMSE of",as.character(round(rmse[which.min(rmse)])),"at k
=",as.character(which.min(rmse) )))
plot(1:100,rmse,type="p",xlab="k",ylab="Test RMSE")

# plotting the best KNN model results
par(mfrow=c(1,1))
par(pch=20, col="black")
knn.fit<-knnreg(train.x,train.y,k=36)
plot(test.y,predict(knn.fit,test.x),type='p', xlab="observed", ylab="predicted")
lines(test.y,test.y)
```

Model 3: Decision Tree

```
#Decision Trees
####################################

library(tree)
set.seed(25)
rm(list=ls())
#library(readxl)
airbnb <- read.csv("listings_V1-3.csv")
# set up a training set

train = sample(1:nrow(airbnb), nrow(airbnb)/2)

# create the tree using the tree command

tree.airbnb=tree(price~.,airbnb,subset=train)

# check the output via summary, you should see that only three variables are used
# note that deviance is simply the sum of squared errors for the decision tree

summary(tree.airbnb)

# Plot the tree

plot(tree.airbnb)
text(tree.airbnb)

# Now, check for opportunities to improve through cost complexity pruning

cv.airbnb=cv.tree(tree.airbnb)
plot(cv.airbnb$size,cv.airbnb$dev,type='b')

# In this case it seems that the best tree is the most complex tree (most nodes)
```

```
# if we did wish to prune it, we could do something like the following
prune.airbnb=prune.tree(tree.airbnb,best=8)
plot(prune.airbnb)
text(prune.airbnb,pretty=0)

# However, in keeping with the cross-validation results, we should predict with
# the unpruned tree.

yhat=predict(tree.airbnb,newdata=airbnb[-train,])
airbnb.test=airbnb[-train,"price"]

# Now plot the results to visualize the predictions relative to the observations

plot(yhat,airbnb.test)
abline(0,1)

# Compute the test MSE

mean((yhat-airbnb.test)^2) #####31144.96 ### RMSE 176
rmse<-sqrt(mean((yhat-airbnb.test)^2) )

### prune
yhat=predict(prune.airbnb,newdata=airbnb[-train,])
airbnb.test=airbnb[-train,"price"]

rmse_pruned<-sqrt(mean((yhat-airbnb.test)^2) )
```

Model 4: Random Forest

```
rm(list=ls())
library(readr)
listings_V1 <- read_csv("~/Downloads/listings_V1 (1).csv")
View(listings_V1_1_)
set.seed(25)
library(randomForest)
library(ggRandomForests)
airbnb<-listings_V1[complete.cases(listings_V1),]
colnames(airbnb)[14]<-"bedtype_pull_outsofa"
airbnb$property_type_Apartment=as.factor(airbnb$property_type_Apartment)
airbnb$property_type_other=as.factor(airbnb$property_type_other)
airbnb$roomtype_sharedroom=as.factor(airbnb$roomtype_sharedroom)
airbnb$roomtype_private=as.factor(airbnb$roomtype_private)
airbnb$bedtype_airbed=as.factor(airbnb$bedtype_airbed)
airbnb$bedtype_couch=as.factor(airbnb$bedtype_couch)
airbnb$bedtype_futon=as.factor(airbnb$bedtype_futon)
```

```
airbnb$bedtype_pull_outsofa=as.factor(airbnb$bedtype_pull_outsofa)
trainindex = sample(nrow(airbnb), nrow(airbnb)*0.5)
train<-airbnb[trainindex,]
test<-airbnb[-trainindex,]
#bagging model, using all 21 predictors#
price.rf=randomForest(price~.,data=train,mtry=21,ntree=1000,importance=TRUE)
price.rf
pred.rf = predict(price.rf,newdata=test)
price.test=test$price
plot(pred.rf, price.test)
abline(0,1)
mean((pred.rf-price.test)^2)
#identify the important predictors#
plot(gg_vimp(price.rf))
importance(price.rf)
varImpPlot(price.rf)
# random forest model with the best number of mtry#
price.rf=randomForest(price~.,data=train,mtry=12,ntree=1000,importance=TRUE)
price.rf
pred.rf = predict(price.rf,newdata=test)
price.test=test$price
mean((pred.rf-price.test)^2)

#boosting the model#
library(gbm)
set.seed(25)
boost.price=gbm(price~.,data=train,distribution="gaussian",n.trees=5000,interaction.depth=10)

summary(boost.price)

par(mfrow=c(1,2))
plot(boost.price,i="cleaning_fee")
plot(boost.price,i="bedrooms")

# Now use the boosted model to predict price on the test set:

yhat.boost=predict(boost.price,newdata=test,n.trees=5000)
mean((yhat.boost-test$price)^2)
```