AMA563 Group Project - Airbnb Price Prediction

Group 10 - 20000963g, 20041325g, 20071573g, 20101492g, 20088944g

Lui Man Hon, Au Hing Yu, Chan Yuet Fai Novena, Liu Chun Sang, Wong Cheuk Naam

2020/11/20

Data Exploration

Summary of Dataset

ABPrices.xlsx is a summarized dataset in Airbnb. It captures information on hosts, geographical locations, rental prices and reviews etc. The dataset contains 48,895 rows and 16 attributes. Below shows all the attributes in the dataset with their corresponding data type and a few examples.

```r
library(HistData)
library(ggplot2)
library(rapportools)
library(randomForest)
library(class)
library(expss)
library(fitdistrplus)
library(rpart)
library(rpart.plot)
require(caTools)
library(xgboost)
library(dplyr)
library("corrplot")
source("http://www.sthda.com/upload/rquery_cormat.r")
ABP <- read.csv("https://brendanlui.github.io/data/ABPrices.csv")
str(ABP)
```

```
## 'data.frame':    48895 obs. of  16 variables:
##  $ id                            : int  2539 2595 3647 3831 5022 5099 5121
5178 5203 5238 ...
##  $ name                          : chr  "Clean & quiet apt home by the
park" "Skylit Midtown Castle" "THE VILLAGE OF HARLEM....NEW YORK !" "Cozy
Entire Floor of Brownstone" ...
##  $ host_id                       : int  2787 2845 4632 4869 7192 7322 7356
8967 7490 7549 ...
##  $ host_name                     : chr  "John" "Jennifer" "Elisabeth"
"LisaRoxanne" ...
##  $ neighbourhood_group           : chr  "Brooklyn" "Manhattan" "Manhattan"
"Brooklyn" ...
```

```
##  $ neighbourhood                 : chr  "Kensington" "Midtown" "Harlem"
"Clinton Hill" ...
##  $ latitude                      : num  40.6 40.8 40.8 40.7 40.8 ...
##  $ longitude                     : num  -74 -74 -73.9 -74 -73.9 ...
##  $ room_type                     : chr  "Private room" "Entire home/apt"
"Private room" "Entire home/apt" ...
##  $ price                         : int  149 225 150 89 80 200 60 79 79 150
...
##  $ minimum_nights                : int  1 1 3 1 10 3 45 2 2 1 ...
##  $ number_of_reviews             : int  9 45 0 270 9 74 49 430 118 160 ...
##  $ last_review                   : chr  "2018-10-19" "2019-05-21" ""
"2019-07-05" ...
##  $ reviews_per_month             : num  0.21 0.38 NA 4.64 0.1 0.59 0.4
3.47 0.99 1.33 ...
##  $ calculated_host_listings_count: int  6 2 1 1 1 1 1 1 1 4 ...
##  $ availability_365              : int  365 355 365 194 0 129 0 220 0 188
...
```

Data Cleansing

Ignore irrelevant fields

Since emphasis is placed on price prediction in this report, the irrelevant and meaningless fields - ID, NAME, HOST_ID and HOST_NAME are being ignored.

```
ABPrices <- ABP[!duplicated(ABP$id), c(5:9,11:16,10)]
head(ABPrices)
```

```
##   neighbourhood_group neighbourhood latitude longitude       room_type
## 1            Brooklyn    Kensington 40.64749 -73.97237    Private room
## 2           Manhattan       Midtown 40.75362 -73.98377 Entire home/apt
## 3           Manhattan        Harlem 40.80902 -73.94190    Private room
## 4            Brooklyn  Clinton Hill 40.68514 -73.95976 Entire home/apt
## 5           Manhattan   East Harlem 40.79851 -73.94399 Entire home/apt
## 6           Manhattan   Murray Hill 40.74767 -73.97500 Entire home/apt
##   minimum_nights number_of_reviews last_review reviews_per_month
## 1              1                 9  2018-10-19              0.21
## 2              1                45  2019-05-21              0.38
## 3              3                 0                            NA
## 4              1               270  2019-07-05              4.64
## 5             10                 9  2018-11-19              0.10
## 6              3                74  2019-06-22              0.59
##   calculated_host_listings_count availability_365 price
## 1                              6              365   149
## 2                              2              355   225
## 3                              1              365   150
## 4                              1              194    89
## 5                              1                0    80
## 6                              1              129   200
```

Categorical data handling

Some attributes appear as ‘string’ are converted to ‘factor’. To ease analysis, ‘factor’ is then transferred to ‘numeric’. Those fields are NEIGHBOURHOOD_GROUP, NEIGHBOURHOOD and ROOM_TYPE.

```
head(ABPrices[c(1:2, 5)])

##   neighbourhood_group neighbourhood       room_type
## 1            Brooklyn    Kensington    Private room
## 2           Manhattan       Midtown Entire home/apt
## 3           Manhattan        Harlem    Private room
## 4            Brooklyn  Clinton Hill Entire home/apt
## 5           Manhattan   East Harlem Entire home/apt
## 6           Manhattan   Murray Hill Entire home/apt

ABPrices$room_type <- as.numeric(factor(ABPrices$room_type, levels =
unique(ABPrices$room_type)))
ABPrices$neighbourhood_group <-
as.numeric(factor(ABPrices$neighbourhood_group,levels =
unique(ABPrices$neighbourhood_group)))
ABPrices$neighbourhood <- as.numeric(factor(ABPrices$neighbourhood),levels =
unique(ABPrices$neighbourhood))
head(ABPrices[c(1:2, 5)])

##   neighbourhood_group neighbourhood room_type
## 1                   1           109         1
## 2                   2           128         2
## 3                   2            95         1
## 4                   1            42         2
## 5                   2            62         2
## 6                   2           138         2
```

Missing value handling

It is found that there are in total 20,144 missing values in the dataset. Missing values appear in 4 attributes: NAME, HOST_NAME, LAST_REVIEW and REVIEWS_PER_MONTH in 10,076 records.

```
ABP[ABP == ""] = NA
sum(is.na(ABP))

## [1] 20144

sum(!complete.cases(ABP))

## [1] 10076
```

```
na_count <-sapply(ABP, function(y) sum(length(which(is.na(y)))))
data.frame(na_count)
```

```
##                                na_count
## id                                    0
## name                                 19
## host_id                               0
## host_name                            21
## neighbourhood_group                   0
## neighbourhood                         0
## latitude                              0
## longitude                             0
## room_type                             0
## price                                 0
## minimum_nights                        0
## number_of_reviews                     0
## last_review                       10052
## reviews_per_month                 10052
## calculated_host_listings_count        0
## availability_365                      0
```

Both LAST_REVIEW and REVIEWS_PER_MONTH have 10,052 missing values. Having no LAST_REVIEW naturally means no reviews are received. In other words, REVIEWS_PER_MONTH equals to 0. Special handling of 'Null-to-Zero' is performed for REVIEWS_PER_MONTH.

```
head(ABPrices[c(9)])
```

```
##   reviews_per_month
## 1              0.21
## 2              0.38
## 3                NA
## 4              4.64
## 5              0.10
## 6              0.59
```

```
ABPrices$reviews_per_month <- ifelse(is.na(ABPrices$reviews_per_month), 0,
ABPrices$reviews_per_month)
head(ABPrices[c(9)])
```

```
##   reviews_per_month
## 1              0.21
## 2              0.38
## 3              0.00
## 4              4.64
## 5              0.10
## 6              0.59
```

LAST_REVIEW with NULL value is converted into the earliest date among LAST_REVIEW data, and all LAST_REVIEW are converted into a yearly basis for easy analysis.

```r
head(ABPrices[c(8)])
```

```
##   last_review
## 1  2018-10-19
## 2  2019-05-21
## 3
## 4  2019-07-05
## 5  2018-11-19
## 6  2019-06-22
```

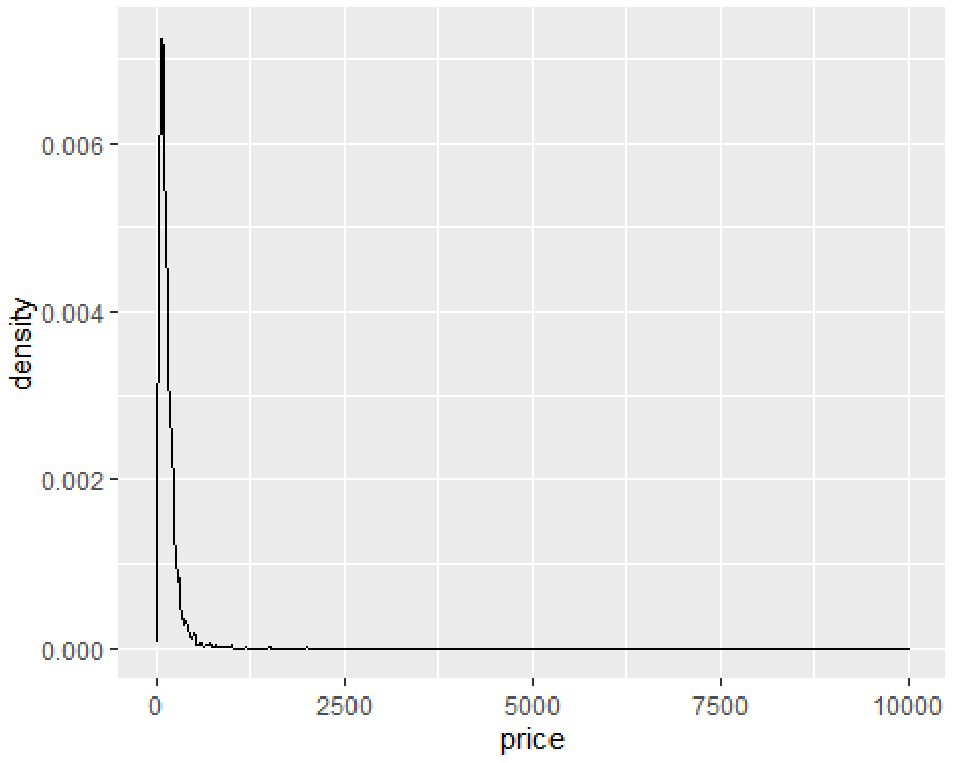
```r
ABPrices$last_review <- sub( "^$", "2099-12-31", ABPrices$last_review)
minLast_review <- min(ABPrices$last_review)
ABPrices$last_review <- ifelse(ABPrices$last_review == "2099-12-31",
minLast_review, ABPrices$last_review)
ABPrices$last_review <- as.Date(ABPrices$last_review)
ABPrices$last_review <- cut(ABPrices$last_review, breaks = "year")
ABPrices$last_review <- gsub("-","" , ABPrices$last_review ,ignore.case =
TRUE)
ABPrices$last_review <- as.integer(ABPrices$last_review)
head(ABPrices[c(8)])
```

```
##   last_review
## 1    20180101
## 2    20190101
## 3    20110101
## 4    20190101
## 5    20180101
## 6    20190101
```

Outlier detection and filtering

There are some extreme prices in the dataset which lead to significant effects on mean and standard derivation, and thus the prediction result. For example, 3 listings are rented at $10,000 while 11 listings are rented for free (i.e. $0).

```r
ggplot(ABPrices, aes(x=price)) + geom_density()
```



```
mean(ABPrices$price)

## [1] 152.7207

sd(ABPrices$price)

## [1] 240.1542

extremePrice <- filter(ABPrices, ABPrices$price == min(ABPrices$price) |
ABPrices$price == max(ABPrices$price))
extremePrice$price

##  [1] 10000 10000     0     0     0     0     0     0     0     0     0
0
## [13]     0 10000
```

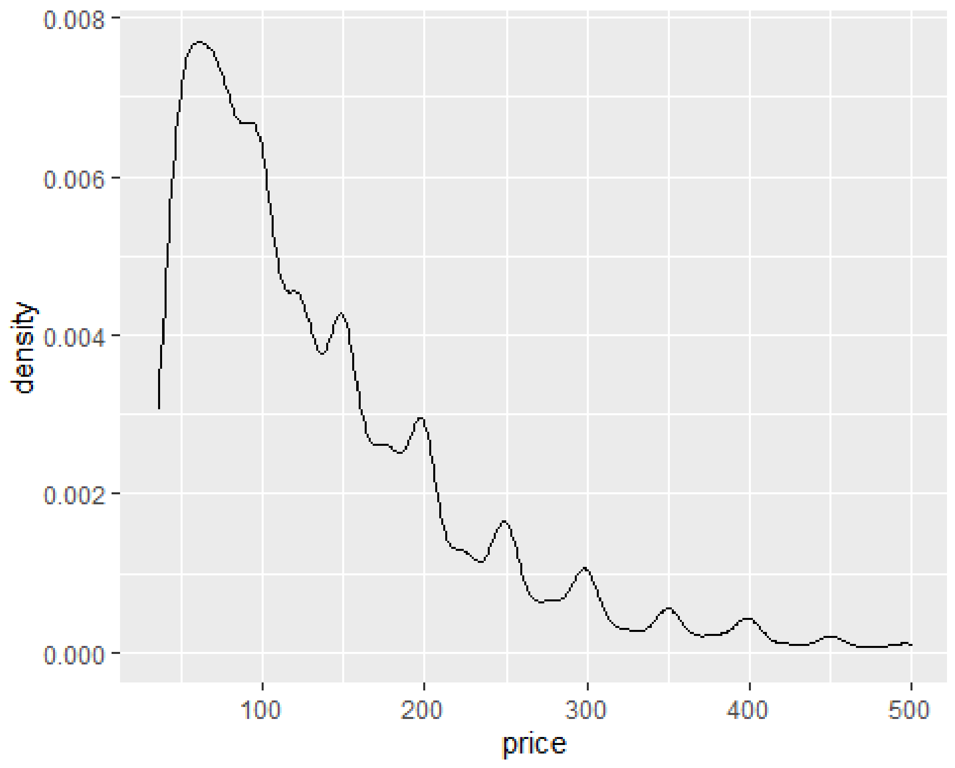
To eliminate outliers, the lower and upper bound are set to filter out data lying outside the 5% confidence interval. In other words, the lowest 2.5% and highest 2.5% prices are removed for subsequent analysis in this report.

```
lower_bound <- quantile(ABPrices$price, 0.025)
upper_bound <- quantile(ABPrices$price, 0.975)

outlier_ind <- which(ABPrices$price < lower_bound | ABPrices$price >
upper_bound)
```

```
ABPrices <- subset(ABPrices, ABPrices$price > lower_bound & ABPrices$price <
upper_bound)
```

After filtering,

```
ggplot(ABPrices, aes(x=price)) + geom_density()
```



```
mean(ABPrices$price)

## [1] 133.1656

sd(ABPrices$price)

## [1] 84.53133
```

Data Visualization

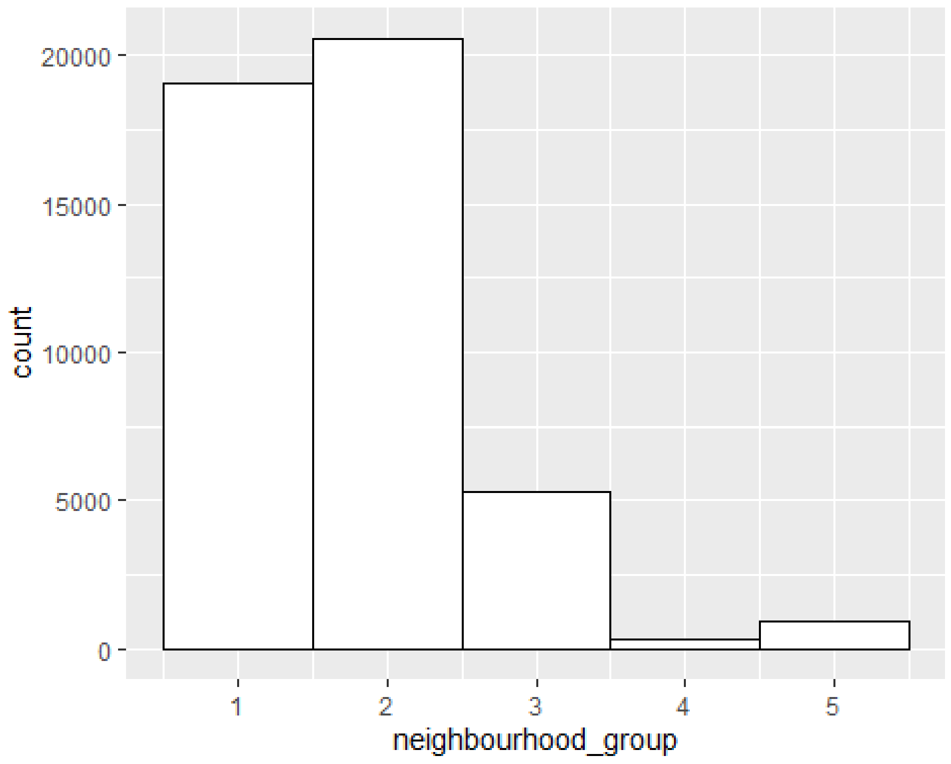
Key attributes

Neighbourhood_group

1: Brooklyn 2: Manhattan 3: Queens 4: Staten Island 5: Bronx

Manhattan has the most listings, followed by Brooklyn and Queens. Over 80% of listings are located in Manhattan and Brooklyn.

```r
ggplot(ABPrices, aes(x=neighbourhood_group)) +
  geom_histogram(binwidth=1, color="black", fill="white")
```

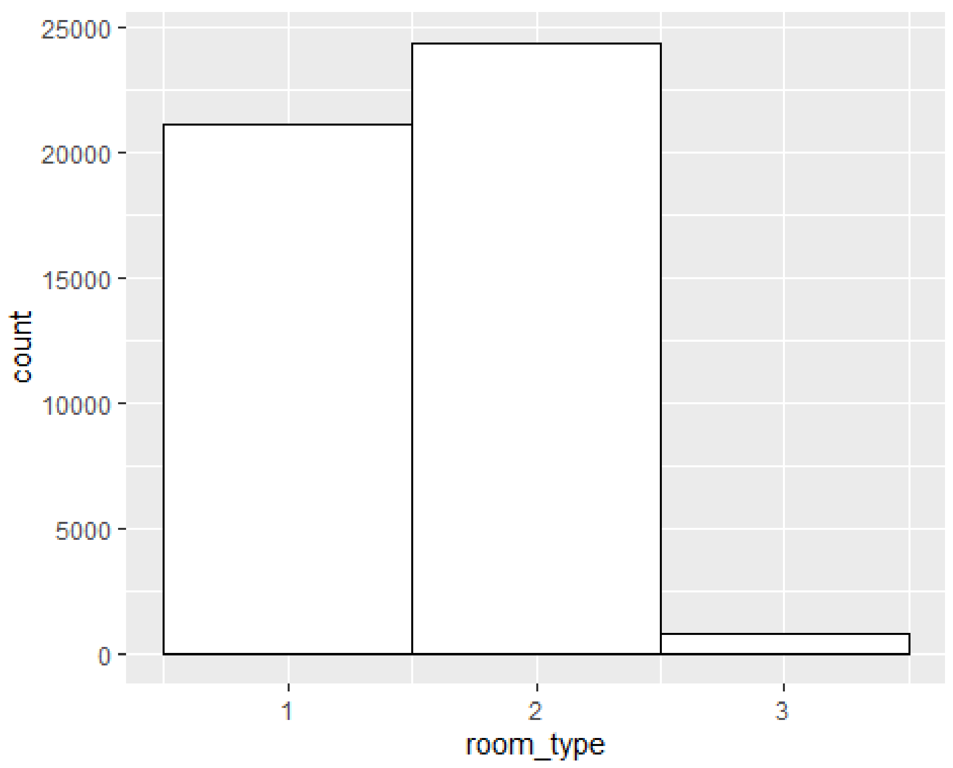


Room type

1: Private room 2: Entire home/apt 3: Shared room

The most common room type is Entire home/apt. Private room comes the second and shared room is extremely limited.

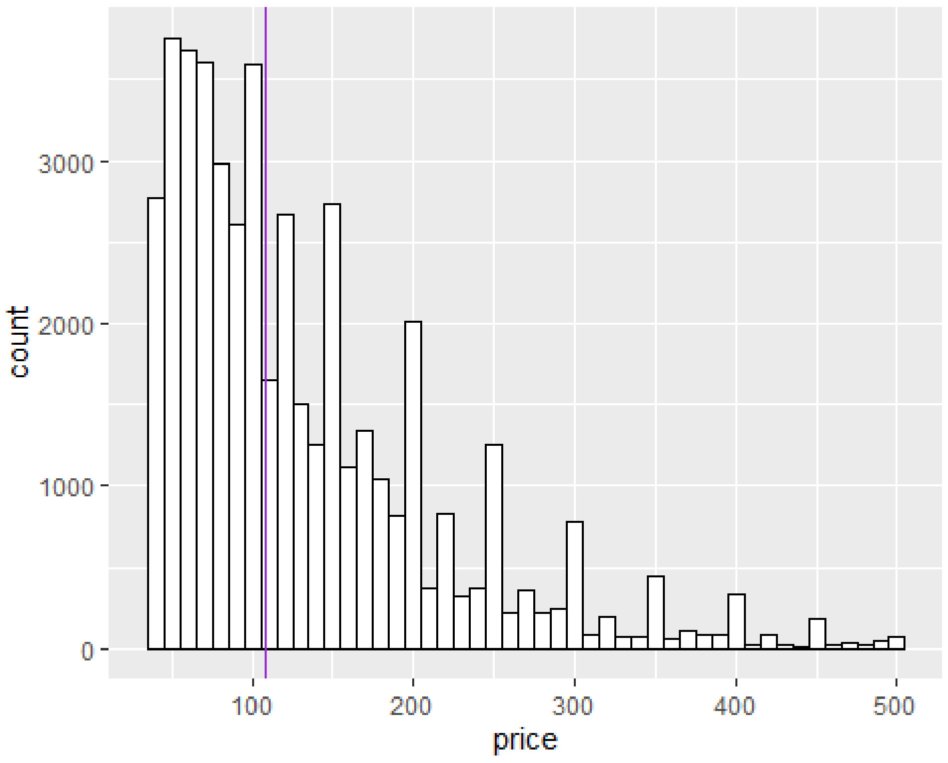
```r
ggplot(ABPrices, aes(x=room_type)) +
  geom_histogram(binwidth=1, color="black", fill="white")
```



Price

Price ranges from $36 to $499 with a mean of $133. But nearly 50% of listings are rented at $100 or lower. The purple line plots the median of price. Roughly speaking, the higher the price, the fewer the count of listings.

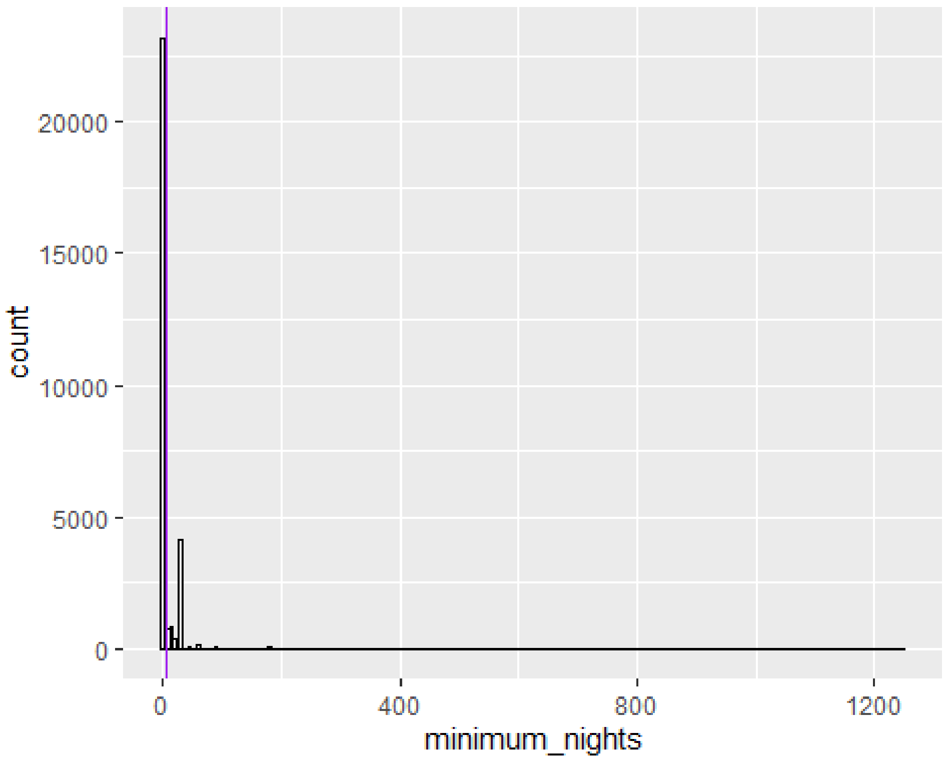
```
summary(ABPrices$price)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    36.0    70.0   108.0   133.2   175.0   499.0

ggplot(ABPrices, aes(x=price)) +
  geom_histogram(binwidth=10, color="black", fill="white") +
  geom_vline(xintercept = median(ABPrices$price), color = "purple")
```



Minimum night

The shortest minimum night is 1 day while the longest minimum night is 1250 days. More than 75% of listings requires minimum night of bookings between 1 to 5 days, while there are 12 extreme cases which requires at least 1 year booking. The purple line indicates the third quartile of MINIMUM_NIGHT.

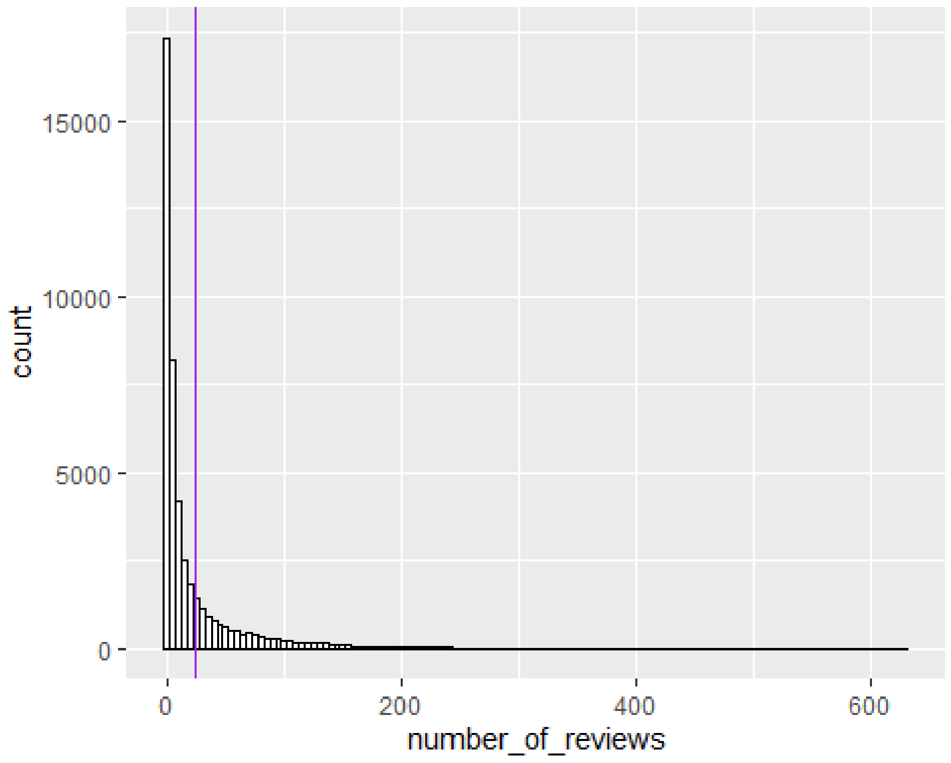
```
summary(ABPrices$minimum_nights)

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##    1.000    1.000    2.000    6.912    5.000 1250.000

ggplot(ABPrices, aes(x=minimum_nights)) +
  geom_histogram(binwidth=5, color="black", fill="white") +
  geom_vline(xintercept = quantile(ABPrices$minimum_nights,0.75), color =
"purple")
```



Number of reviews

More than 75% listings has at least one review. The majority of listings have less than 24 reviews, while a few hot listings have over 500 reviews. The purple line reflects the upper quartile of NUMBER_OF_REVIEW.

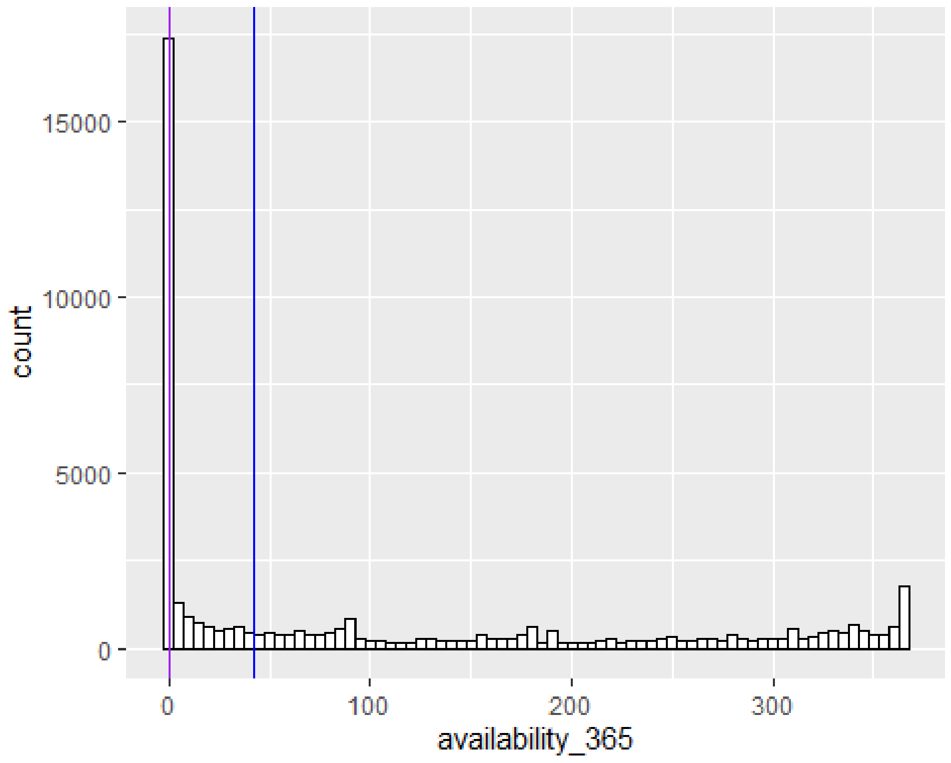
```
summary(ABPrices$number_of_reviews)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    1.00    5.00   23.82   24.00  629.00

ggplot(ABPrices, aes(x=number_of_reviews)) +
  geom_histogram(binwidth=5, color="black", fill="white") +
  geom_vline(xintercept = quantile(ABPrices$number_of_reviews,0.75), color =
"purple")
```



Availability

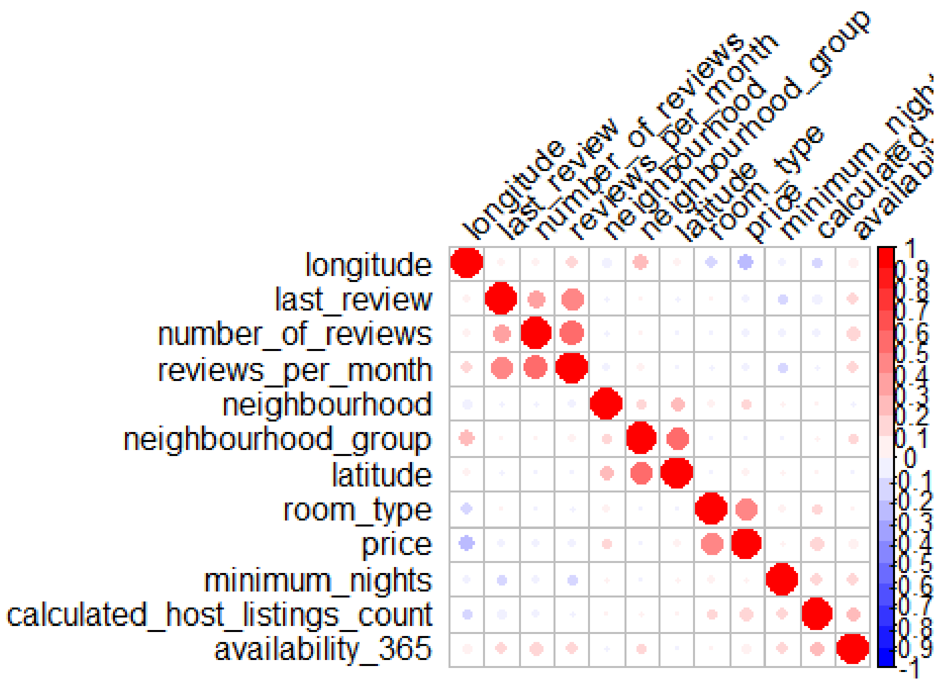
Over 25% of listings are not available all over the year (the purple line) and half of the listings are available less than 50 days (the blue line). Listings with availability between 100 and 300 days are almost equally distributed.

```
summary(ABPrices$availability_365)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0     0.0    42.0   110.9   220.0   365.0

ggplot(ABPrices, aes(x=availability_365)) +
  geom_histogram(binwidth=5, color="black", fill="white") +
  geom_vline(xintercept = quantile(ABPrices$availability_365,0.25), color =
"purple") +
  geom_vline(xintercept = quantile(ABPrices$availability_365,0.5), color =
"blue")
```



Correlation

Since the correlation between the fields only implies the linear relationship with each other, after we tried to use only 4 fields (LONGITUDE, NEIGHBOURHOOD, ROOM_TYPE and CALCULATED_HOST_LISTINGS_COUNT) to train the models and have a test, there is only K-NN model having a very slight improvement to predict the price. Thus, we are going to using all meaningful columns to train the models, which is much easier in comparison.



After showing the correlation for each attribute, it shows NUMBER_OF_REVIEWS vs REVIEWS_PER_MONTH and LATITUDE vs NEIGHBOURHOOD_GROUP having a relatively higher correlation, which are explainable that the items in these 2 pairs are similar. Besides, it is remarkable that room type has the highest correlation in terms of price.

Modeling in Supervised Learning

Set up RMSE and R-square functions

```r
RMSE <- function(actual ,predict){
  return(sqrt( mean( (predict - actual)^2, na.rm = TRUE) ))
}
R2 <- function (actual, predict){
  return(cor(actual, predict) ^ 2)
}
```

Split the data into two sets - training data and testing data

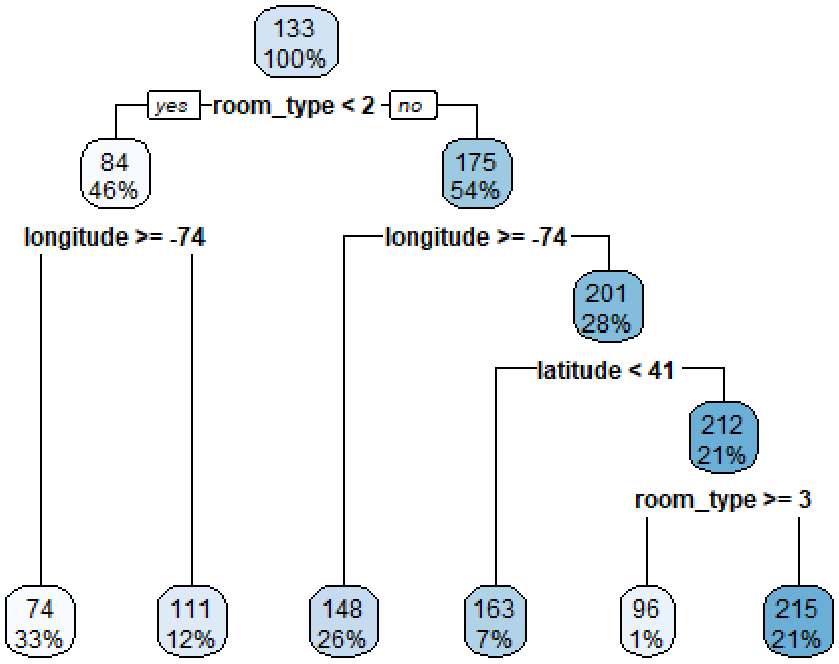
```r
set.seed(runif(1) * 100)
sample = sample.split(ABPrices$price, SplitRatio = .75)
training_set = subset(ABPrices, sample == TRUE)
test_set  = subset(ABPrices, sample == FALSE)
rmse_result <- c()
```

```
r2_result <- c()
num_of_models <- 0
```

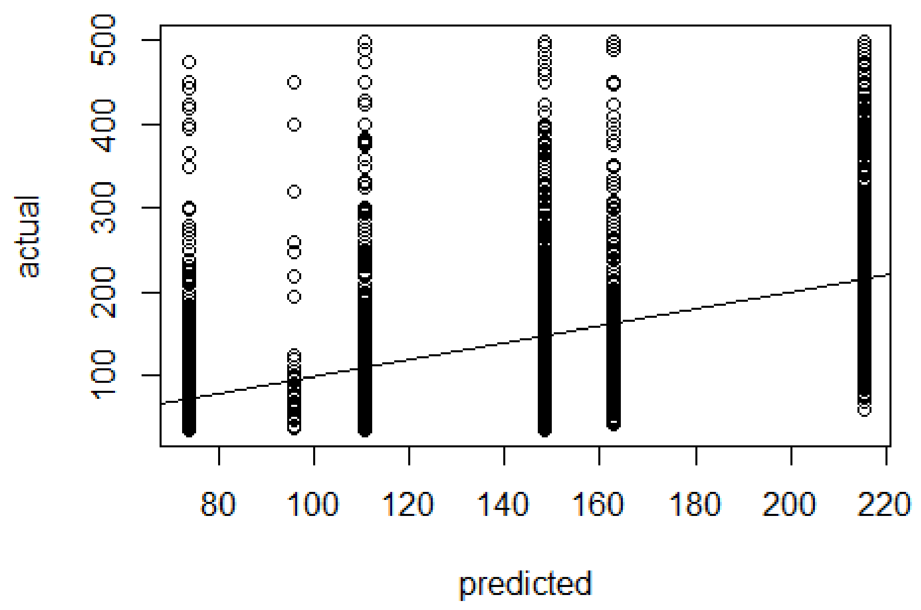
Decision Tree

Build a decision tree model to predict price

```
dt <- rpart(
    price ~ .,
    data=training_set
  )
rpart.plot(dt)
```



```
pred = predict(dt, newdata=test_set[-12])

pred_ensemble = pred
num_of_models <- num_of_models + 1

# difference between predicted and actual value
plot(pred,test_set$price,
     xlab="predicted",ylab="actual")
abline(a=0,b=1)
```



```r
# rmse and r2
rmse_result <- cbind(rmse_result, sqrt( mean( (pred - test_set[, 12])^2,
na.rm = TRUE) ))
colnames(rmse_result) <- c(colnames(rmse_result)[colnames(rmse_result) !=
""], 'Decision Tree')
r2_result <- cbind(r2_result,R2(test_set[, 12],pred))
colnames(r2_result) <- c(colnames(r2_result)[colnames(r2_result) != ""],
'Decision Tree')
RMSE(test_set[, 12],pred)
```

```
## [1] 65.80048
```

```r
R2(test_set[, 12],pred)
```

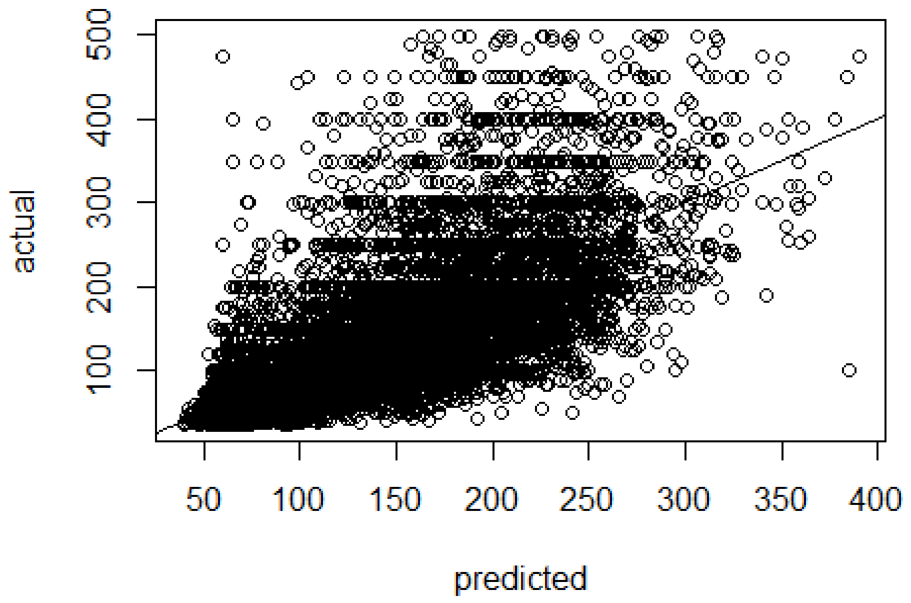
```
## [1] 0.3797508
```

Random Forest

Build a random forest model to predict price

```r
rf <- randomForest(
  price ~ .,
  data=training_set
)
pred = predict(rf, newdata=test_set[-12])
```

```
pred_ensemble = pred_ensemble + pred
num_of_models <- num_of_models + 1

# difference between predicted and actual value
plot(pred,test_set$price,
     xlab="predicted",ylab="actual")
abline(a=0,b=1)
```



```
# rmse and r2
rmse_result <- cbind(rmse_result, sqrt( mean( (pred - test_set[, 12])^2,
na.rm = TRUE) ))
colnames(rmse_result) <- c(colnames(rmse_result)[colnames(rmse_result) !=
""], 'Random Forest')
r2_result <- cbind(r2_result,R2(test_set[, 12],pred))
colnames(r2_result) <- c(colnames(r2_result)[colnames(r2_result) != ""],
'Random Forest')
RMSE(test_set[, 12],pred)

## [1] 57.35425

R2(test_set[, 12],pred)

## [1] 0.5297347
```

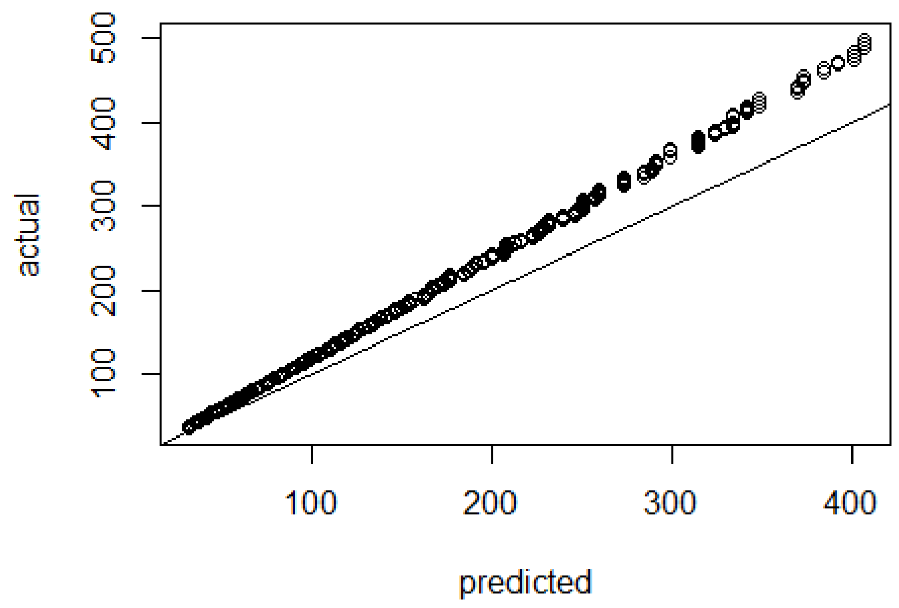
XGBoost

Build a XGBoost model to predict price

```
train <- xgb.DMatrix(data = data.matrix(training_set), label=
training_set[,12])
test <- xgb.DMatrix(data = data.matrix(test_set), label= test_set[,12])

model <- xgboost(data = train, nround = 5)
```

```
## [1]  train-rmse:110.468872
## [2]  train-rmse:77.398857
## [3]  train-rmse:54.232010
## [4]  train-rmse:38.001606
## [5]  train-rmse:26.629797
```

```
pred <- predict(model, test)

pred_ensemble = pred_ensemble + pred
num_of_models <- num_of_models + 1

# difference between predicted and actual value
plot(pred,test_set$price,
     xlab="predicted",ylab="actual")
abline(a=0,b=1)
```



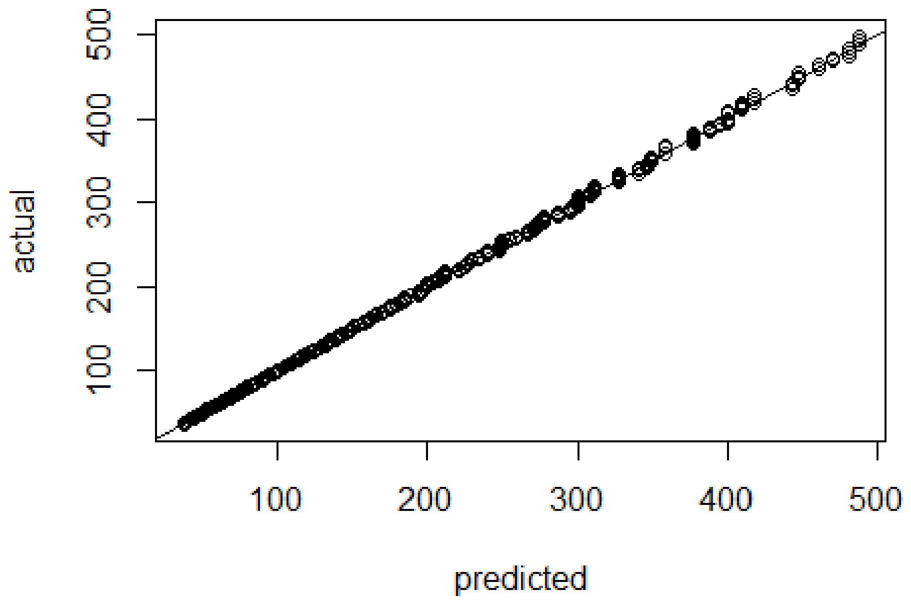
```
RMSE(test_set[, 12],pred)
```

```
## [1] 26.38511
```

```
R2(test_set[, 12],pred)
```

```
## [1] 0.9998501
```

```
# transformation for performance tuning based on the above observation
pred = pred *1.2
plot(pred,test_set$price,
     xlab="predicted",ylab="actual")
abline(a=0,b=1)
```



```
# rmse and r2
rmse_result <- cbind(rmse_result, sqrt(mean( (pred - test_set[,12])^2, na.rm
= TRUE) ))
colnames(rmse_result) <- c(colnames(rmse_result)[colnames(rmse_result) !=
""], 'XGBoost')
r2_result <- cbind(r2_result,R2(test_set[, 12],pred))
colnames(r2_result) <- c(colnames(r2_result)[colnames(r2_result) != ""],
'XGBoost')
RMSE(test_set[, 12],pred)
```

```
## [1] 1.079112
```

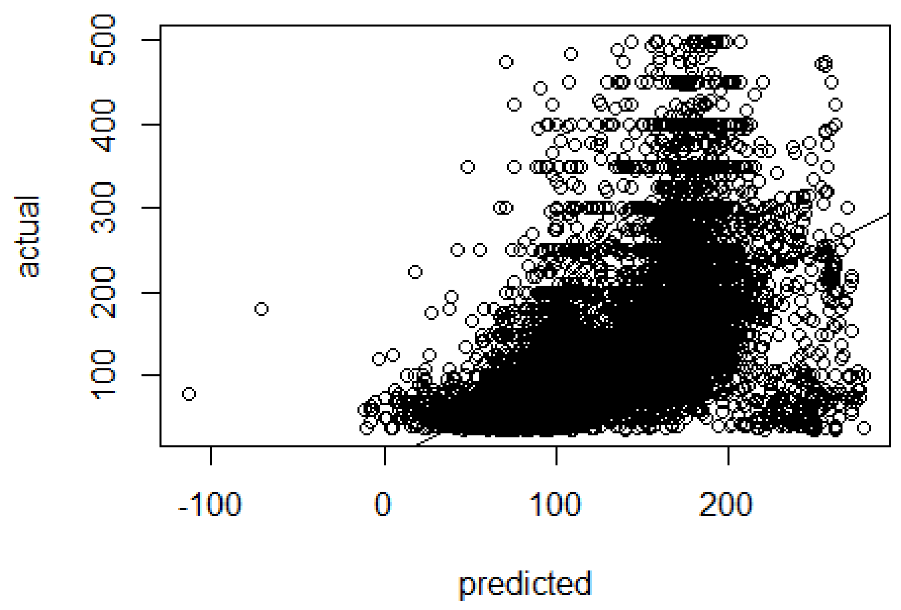
```
R2(test_set[, 12],pred)
```

```
## [1] 0.9998501
```

Linear Regression

Build a standard linear regression model to predict price

```
gen.lm <- lm(price ~ ., data = training_set)
summary(gen.lm)

## 
## Call:
## lm(formula = price ~ ., data = training_set)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -236.07  -38.38  -13.19   22.28  411.07 
## 
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)    
## (Intercept)                    -3.014e+04  7.817e+02 -38.553  < 2e-16 ***
## neighbourhood_group             6.476e-01  5.622e-01   1.152    0.249    
## neighbourhood                   7.485e-02  5.665e-03  13.213  < 2e-16 ***
## latitude                        9.438e+01  8.389e+00  11.250  < 2e-16 ***
## longitude                      -4.032e+02  8.918e+00 -45.217  < 2e-16 ***
## room_type                       7.166e+01  7.268e-01  98.586  < 2e-16 ***
## minimum_nights                 -2.228e-01  1.952e-02 -11.416  < 2e-16 ***
## number_of_reviews              -7.241e-02  1.033e-02  -7.007 2.48e-12 ***
## last_review                    -1.747e-04  1.385e-05 -12.620  < 2e-16 ***
## reviews_per_month               1.233e+00  3.091e-01   3.990 6.61e-05 ***
## calculated_host_listings_count  1.885e-01  1.179e-02  15.991  < 2e-16 ***
## availability_365                7.640e-02  3.113e-03  24.544  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 70.02 on 34673 degrees of freedom
## Multiple R-squared:  0.3193, Adjusted R-squared:  0.3191 
## F-statistic:  1479 on 11 and 34673 DF,  p-value: < 2.2e-16

y_pred = predict(gen.lm, test_set, allow.new.levels = TRUE)

pred_ensemble = pred_ensemble + y_pred
num_of_models <- num_of_models + 1

# difference between predicted and actual value
plot(y_pred,test_set$price,
     xlab="predicted",ylab="actual")
abline(a=0,b=1)
```



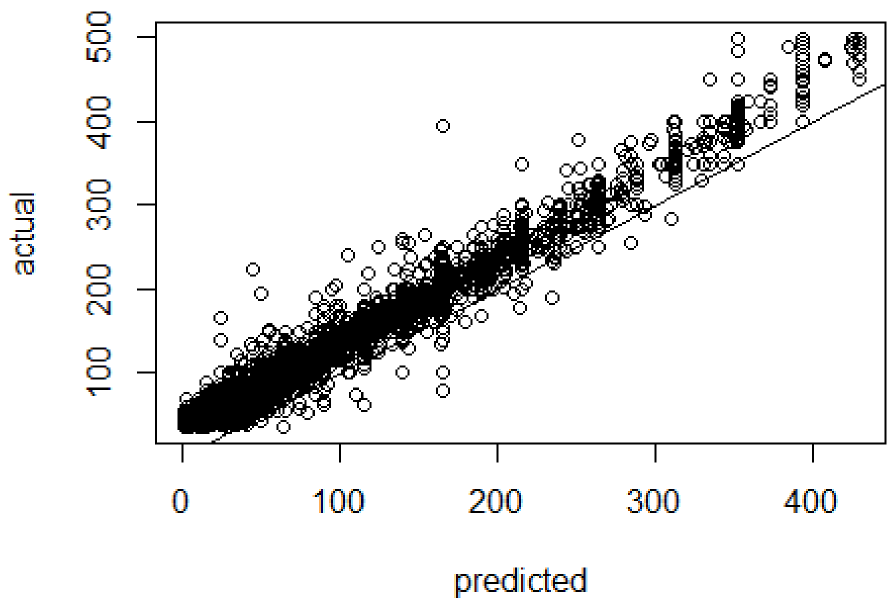
```
# rmse and r2
rmse_result <- cbind(rmse_result, sqrt( mean( (y_pred - test_set[, 12])^2,
na.rm = TRUE) ))
colnames(rmse_result) <- c(colnames(rmse_result)[colnames(rmse_result) !=
""], 'Linear Regression')
r2_result <- cbind(r2_result,R2(test_set[, 12],y_pred))
colnames(r2_result) <- c(colnames(r2_result)[colnames(r2_result) != ""],
'Linear Regression')
RMSE(test_set[, 12],y_pred)

## [1] 69.03363

R2(test_set[, 12],y_pred)

## [1] 0.3171391
```

K-nearest Neighbors

Build a K-nearest neighbors model to predict price

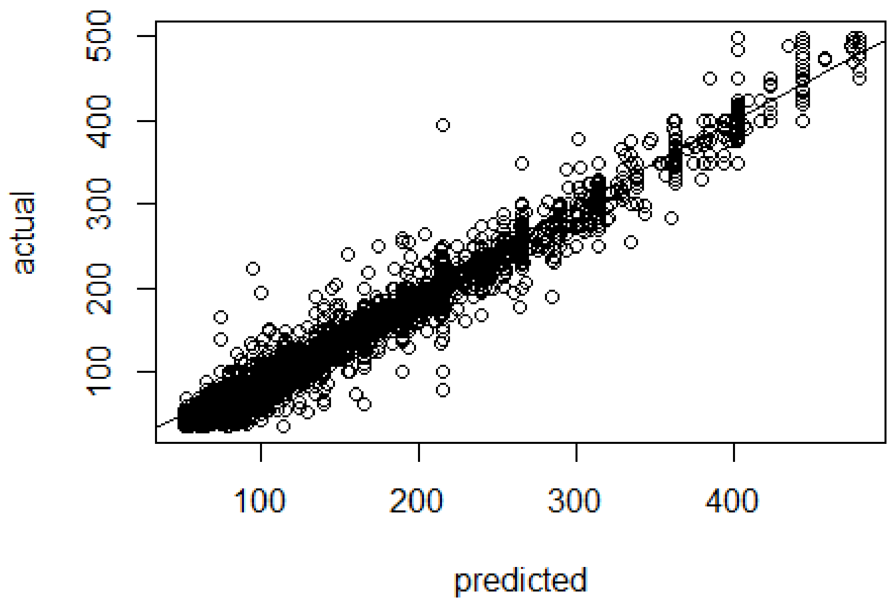
```
kMin = 5
kMax = 31
save_k = 0
rmse_save = 99999999
y_pred = 0
y_pred_save <- c()
```

```r
for(i in kMin:kMax){
  if (i %% 2 == 1) {
    y_pred <- knn(train = training_set, test_set, cl=training_set[, 12], k=i)
    rmse = RMSE(test_set[, 12],as.numeric(y_pred))

    if(rmse < rmse_save){
      rmse_save = rmse
      save_k = i
      y_pred_save <- as.numeric(y_pred)
    }
  }
}

pred_ensemble = pred_ensemble + y_pred_save
num_of_models <- num_of_models + 1

# difference between predicted and actual value
plot(y_pred_save,test_set$price,
     xlab="predicted",ylab="actual")
abline(a=0,b=1)
```



```r
RMSE(test_set[, 12],y_pred_save)

## [1] 37.53984

R2(test_set[, 12],y_pred_save)
```

```
## [1] 0.9821172

# transformation for performance tuning based on the above observation
y_pred_save = y_pred_save + 50
plot(y_pred_save,test_set$price,
     xlab="predicted",ylab="actual")
abline(a=0,b=1)
```



```
# rmse and r2
rmse_result <- cbind(rmse_result, RMSE(test_set[, 12],y_pred_save))
colnames(rmse_result) <- c(colnames(rmse_result)[colnames(rmse_result) !=
""], 'K-nearest Neighbors')
r2_result <- cbind(r2_result,R2(test_set[, 12],y_pred_save))
colnames(r2_result) <- c(colnames(r2_result)[colnames(r2_result) != ""], 'K-
nearest Neighbors')
RMSE(test_set[, 12],y_pred_save)

## [1] 18.15972

R2(test_set[, 12],y_pred_save)

## [1] 0.9821172
```

Comparison

```
t(rmse_result)
```

```
##                           [,1]
## Decision Tree       65.800484
## Random Forest       57.354252
## XGBoost              1.079112
## Linear Regression   69.033632
## K-nearest Neighbors 18.159724
```

```
t(r2_result)
```

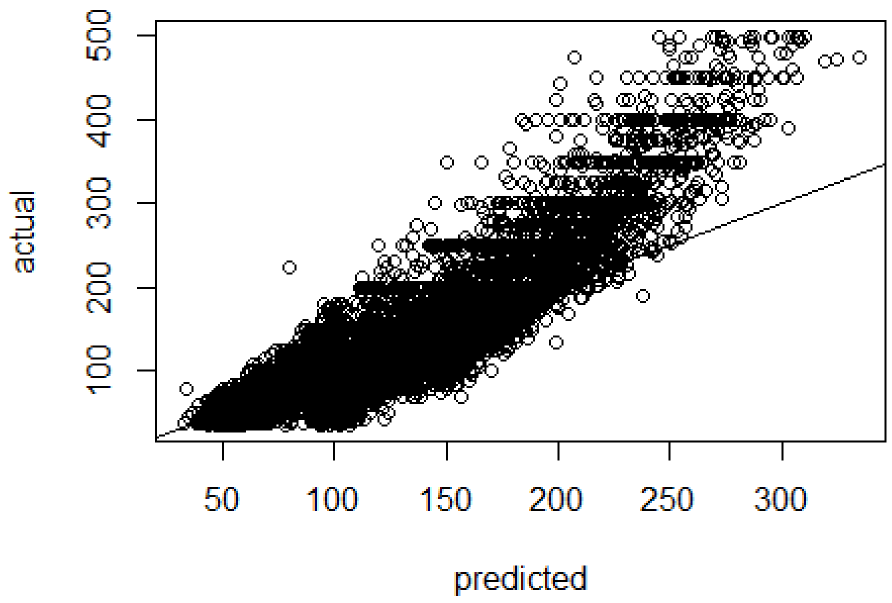
```
##                          [,1]
## Decision Tree       0.3797508
## Random Forest       0.5297347
## XGBoost             0.9998501
## Linear Regression   0.3171391
## K-nearest Neighbors 0.9821172
```

Based on the above modeling results, the root-mean-square error in the XGBoost model is always smaller than other models. Thus, XGBoost model is suitable for predicting the price by using this dataset since its prediction performance is the best among the above models.

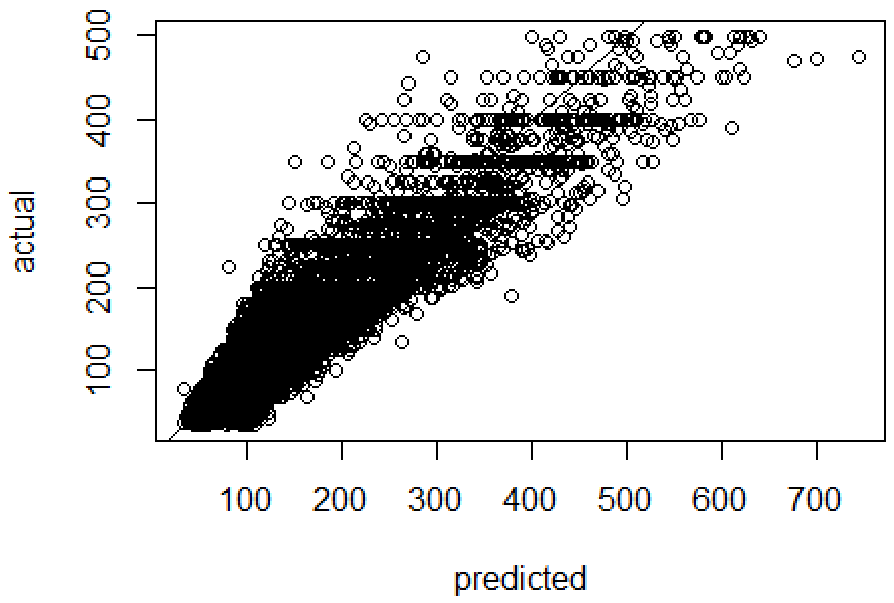
Stacking Ensemble Machine Learning

With a view to preventing from overfitting, we tried to combine the above 5 models' prediction results and then got the final prediction value with their mean as below.

```
pred_ensemble = pred_ensemble / num_of_models

# difference between predicted and actual value
plot(pred_ensemble,test_set$price,
     xlab="predicted",ylab="actual")
abline(a=0,b=1)
```



```
RMSE(test_set[, 12],pred_ensemble)

## [1] 41.15305

R2(test_set[, 12],pred_ensemble)

## [1] 0.8353912

# transformation for performance tuning based on the above observation
pred_ensemble[pred_ensemble > 150] <- (pred_ensemble[pred_ensemble > 150] ^
2) /150
plot(pred_ensemble,test_set$price,
     xlab="predicted",ylab="actual")
abline(a=0,b=1)
```



```r
# rmse and r2
RMSE(test_set[, 12],pred_ensemble)

## [1] 33.90134

R2(test_set[, 12],pred_ensemble)

## [1] 0.8726239
```

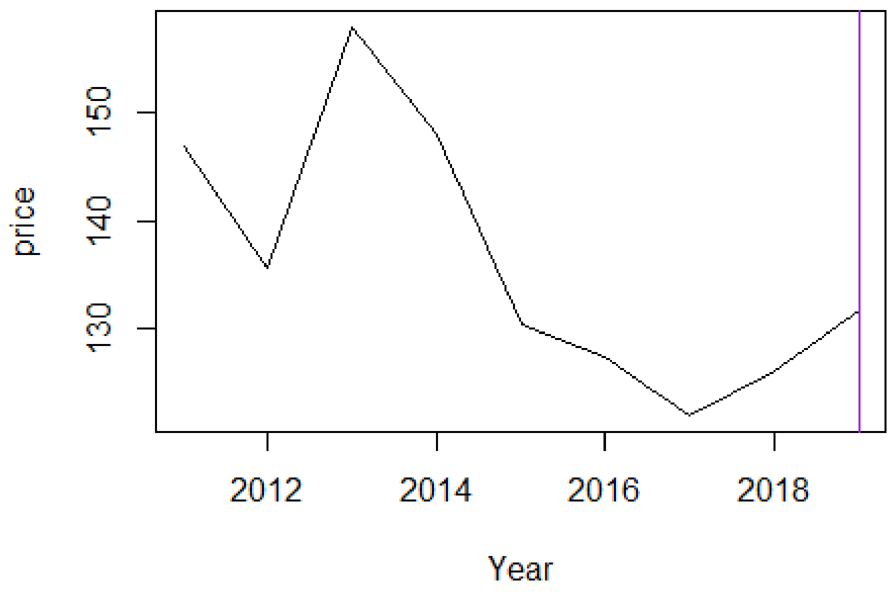
Application

Our goal is to build an application to determine whether the marked price in Airbnb website is higher or lower than the normal market price which is predicted by our model. Also, since this dataset is in 2019 and we are trying to predict the price at the end of 2020, in order to predict the price more accurately, we use Monte Carlo Simulation to forecast the future price trend in years and then apply to model for sight tuning.

Monte Carlo Simulation using GBM

Monte Carlo Simulation with Geometric Brownian Motion is performed for predicting rental price in the coming 3 years. Average price of each year is taken as reference in this simulation. Overall speaking, there is a decreasing trend, especially between year 2013 and 2017.

```
sim_base <- c(ABPrices %>% dplyr::select(room_type,last_review,price) %>%
group_by(last_review) %>% summarise(mean=mean(price), sd=sd(price)))

## `summarise()` ungrouping output (override with `.groups` argument)

sim_base <- as.data.frame(sim_base)
price <- c(sim_base$mean)
price <- cbind(Year=c(2011:2019),price)
plot(price, type='l', xlab = 'Year')
abline(v=2019, col="purple")
```

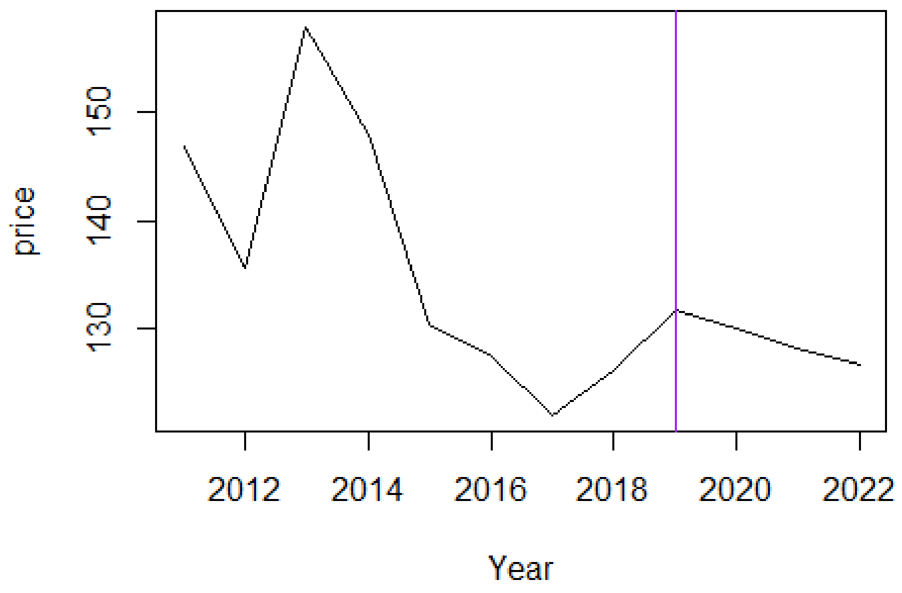


Log returns is used for calculating the yearly price change in average.

```
log_returns <- diff(log(sim_base$mean), lag=1)
mean_returns <- mean(log_returns)
sd_returns <- sd(log_returns)
var_returns <- var(log_returns)

drift <- mean_returns - var_returns/2
sd <- sd_returns
```

Standard normal random variables are generated with reference to the concept of Brownian Motion. A simulation of 50000 iterations is performed to predict the average price between 2020 and 2022. From the predicted result, it seems that there is a decreasing price trend for the coming three years. Remarks: Years on the right of the purple line are the simulated result.

```r
n <- 4
iteration <- 50000

rn <- replicate(iteration,rnorm(n,0,1))
changes <- exp(drift + sd * rn)

set <- matrix(
  c(rep(last(sim_base$mean),iteration),rep(0,n*iteration-iteration)),
  nrow=n,
  ncol=iteration,
  byrow = TRUE
)

for (i in 2:n){
  set[i,] <- set[i-1,] * changes[i,]
}

set <- cbind(1:n,set)
colnames(set) <- c('Year',paste("Iteration",seq(1:iteration), sep=""))

pred_result <- as.data.frame(set)
pred_result <- melt(pred_result, id.vars = 'Year', variable.name =
'iteration')

predict_mean <- c(rowMeans(set[,-1]))
price <- c(sim_base$mean,predict_mean[2:n])
price <- cbind(Year=c(2011:(2019+n-1)),price)
plot(price, type='l', xlab = 'Year')
abline(v=2019, col="purple")
```



```
#the percentage change of inflation from 2019 to 2020
(price[10, 2] - price[9, 2]) / price[9, 2]

##       price
## -0.01363846
```

For example, in 2020, we can apply the percentage change of inflation from 2019 to 2020 to the predicted price from model in order to get a more precise value.