Airbnb in Southbank, Melbourne

Written by

Yang Chen, Guorong Liu, Fangwen Wang,

Duosi Zheng and Yuanzhe Zhuang

MET AD 699 A1 Data Mining for Business Analytics

Boston University

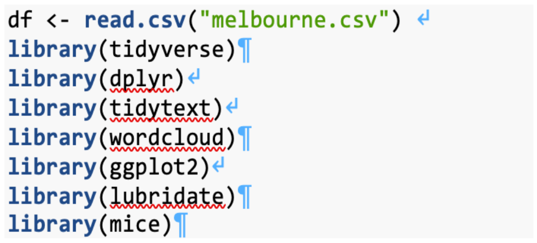
Dec 9, 2019

Table of Contents

Introduction

The neighborhood that our project focuses on is Southbank in Melbourne Australia. Southbank is an inner urban neighborhood of Melbourne, Victoria, Australia. Today, the Southbank is one of the major business centers in the greater Melbourne area and one of the major entertainment districts in Melbourne. By exploring the data, we used different tools to analyze how customers choose their Airbnb in Southbank based on various conditions, such as the house rules, house type, transit, price, cleaning fee, review scores and nights the customers can chose to stay. By cleaning the target data, we used three methods such as k-nearest neighbors, Naive Bayes and Classification Tree to classify different data sets with different outcome values, thus we can group those customers and define the cluster based the characteristic they have. We followed five steps that includes data preparation & exploration, prediction, classification, clustering and conclusion. In the end, we would like to provide some suggestions for the investors, Airbnb host, and Airbnb company in the whole Southbank neighborhood.

Step I: Data Preparation & Exploration



```
df <- read.csv("melbourne.csv")
library(tidyverse)
library(dplyr)
library(tidytext)
library(wordcloud)
library(ggplot2)
library(lubridate)
library(mice)
```

1. Explaining Missing Values

After importing the data, we filter the data to only include the neighborhood of Southbank in Melbourne, Australia.

```
# filter the only neighborhood Southbank
df1 <- filter(df, neighborhood=="Southbank")
dim(df1)
## [1] 1248   84
1248*84
## [1] 104832
#tell us the missing values,
this gives the number of null values:
sum(is.na(df1))
## [1] 5156
```

Then, we look at the data set, which contains 1248 rows, and 84 columns (total of 104832), and they mix with missing values and null values. From the textbook, we choose to sum all the missing values and have a result of 5156. So, our data frame only contains about 1248 rows of data, we are not sure if the missing values are in which columns/rows, or they might exist in the column/row. In order to not lose many useful data for us to analyze the neighborhood as our data frame is already in a small shape with limited data, we have decided to get rid of the missing values based on our data frame for specific model use in the following report. Here are two ways for us to select through the process: 1. Drop the missing values (include blank cells, “N/A” values, NA values): from selecting the variables from the data frame and building a data set for us to run the model, we choose multiple sections to directly drop the missing values. For values are not listed as NA, such as blank cells and "N/A" values they will transform in NA, and they will be dropped then. 2. Another way is to substitute the missing values with average values of the designated data frame for some models. Also, the sections for directly dropping the missing values are as below:

I. Summary Statistics and Visualization: There is a lot of data processing going on in this section, and we are hoping to see more useful data in this section as we have a general overview for the Airbnb business in Southbank neighborhood. The visualization will be clearer since the missing values will be dropped at the beginning of the data frame.

II. Multiple Regression Model for Predication: There are many predicators for regression to select and compare through the process, so dropping the missing values would be easy to process the data set and keep the only useful predicators.

III. K Nearest Neighborhood: In order to have precise k values and classify the features of the data set, it is efficient and accurate have the results for the neighborhood by dropping all missing values.

IV. Naive Bayes: Similar method for this model - after selecting four other variables, dropping all the missing values will be the most efficient way for running the data, and it can provide more precise result from this data model

V. Classification Tree: In order to decide the specific class from the tree model, we decide to drop the missing values and get more precise results from the tree classification.

VI. Clustering: For beds in the selection, we have reviewed the summary (also the histogram graph), and there are too many missing values in the proportion to the whole data set for this model. If we directly drop the missing values, the model will not contain good cluster groups with very less data involved. Then, we have decided to use the mean value of this data to replace the missing values. Same method for the "review scores rating", we replace the mean value for the missing values so the data set can have more data through the whole set, so we can have more data to supplement the clustering model.

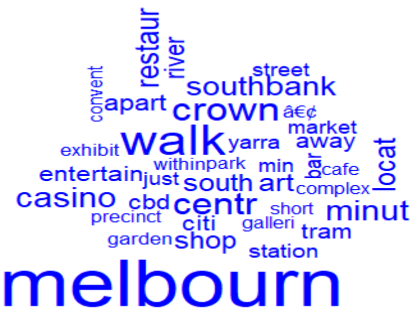
There are two efficient ways for us to detect the missing values through any data frame is "sum(is.na())", which sum up the null values from the data set, and "md.pattern()" functions, which can help us count the missing values (NA or blank cells, etc) in the square shape method, and we will demonstrate it later when we use it. Sum up above methodologies from cleaning data, let's first look at the summary of our Airbnb model in the Southbank Neighborhood of Melbourne, Australia.

II. Summary Statistics & Answer Proposed Questions

Before reviewing actual data, we would like to see the features in this neighborhood overview. Across the columns in our dataset, we find that "neighborhood overview" include the description of the neighborhood, and they also include some summary about the listing, such as "walking distance to the art center", etc. We find this is very helpful for us to know our targeted market.

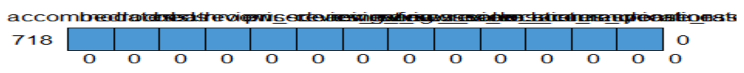
```
#First, what are the features of South Bank, Melbourne?
library(tm)
library(e1071)
clean_corpus <- function(corpus){
  corpus <- tm_map(corpus,stripWhitespace)
  corpus <- tm_map(corpus,removePunctuation)
  corpus <- tm_map(corpus,removeNumbers)
  corpus <- tm_map(corpus,content_transformer(tolower))
  corpus <- tm_map(corpus,removeWords,stopwords("en"))
  corpus <- tm_map(corpus,stemDocument)
  return(corpus)
}
dff <- select(df1,neighborhood_overview)
dff$neighborhood_overview[dff$neighborhood_overview == ""] <- NA
dff <- na.omit(dff)   #drop NA values
sum(is.na(dff))
## [1] 0
```

```
descrip <- VCorpus(VectorSource(dff))
descrip <- clean_corpus(descrip)
tdm1 <- TermDocumentMatrix(descrip)
inspect(tdm1)
## <<TermDocumentMatrix (terms: 2167, documents: 1)>>
## Non-/sparse entries: 2167/0
## Sparsity           : 0%
## Maximal term length: 222
## Weighting          : term frequency (tf)
## Sample             :
##              Docs
## Terms           1
##   art         417
##   casino      497
##   centr       579
##   crown       650
##   locat       465
##   melbourn   1496
##   minut       509
##   restaur     490
##   southbank   490
##   walk        891
mtx1 <- as.matrix(tdm1)
term_frequency1 <- rowSums(mtx1)
term_frequency1 <- sort(term_frequency1,decreasing = TRUE)
word_freqs1 <- data.frame(term = names(term_frequency1), num = term_frequency1)
wordcloud(word_freqs1$term, word_freqs1$num, min.freq = 200, colors = "blue")
```



We choose the words that appear more than 200 times, and we have result in word cloud as above. In the word cloud above, we can see there are a lot of key terms in the graph, such as “art, walk, restaur, casino, locat, shop”, and more. And of course, the biggest font (most frequent) word is “melbourn”, which makes sense since Southbank is part of the Melbourne. Then, we can have an estimated image about this neighborhood, where it involves a lot of art or commercial activities, and it feels more like a neighborhood environment for people to purchase and chill. Next, we will look at more details on the Airbnb housing features.

```
#select variables from the data set and make up a new data frame
df2 <- select(df1, accommodates,bedrooms,beds, bathrooms, price,
              review_scores_rating, cleaning_fee,
              review_scores_value,review_scores_location, review_scores_accuracy,
              review_scores_communication, review_scores_cleanliness, host_response_rate)
df2$host_response_rate[df2$host_response_rate == "N/A"]<-NA  #change the N/A value to NA
df2$host_response_rate[df2$host_response_rate == ""]<-NA     #change blank cells value to NA
#tell us the missing values, this gives the number of null values:
sum(is.na(df2))  #relationship of two or more variables
## [1] 2314
df3 <- na.omit(df2)   #drop NA values
df33 <- droplevels(df3)   #make sure there is no ghost row in the dataset (no ghost)
host_response_rate <- as.numeric(sub("%","", df3$host_response_rate))/100
df3$host_response_rate  <- host_response_rate #now the percentage is in decimal
sum(is.na(df3))  #none missing values anymore
## [1] 0
anyNA(df3)   #double-check...
## [1] FALSE
md.pattern(df3)  #check if there are any missing values involved, 0 represents none
```



As we run the analysis, we need to change the host response rate to numerical values so the summary can run the values, and it can ensure our table for the next part will display successfully. And we filter all the missing values, include blank cells, NA value, and N/A values. In the following table, there are five summary statistics involved from the previous built data frame. Here are some questions proposed based on the data frame:

1. **How does Airbnb housings in Southbank perform?**

```
summary_score <- data.frame(mean <- sapply(df3, mean),
                       sd <- sapply(df3, sd),
                       median <- sapply(df3, median),
                       max <- sapply(df3, max),
                       min <- sapply(df3, min))
summary_score
##                               mean....sapply.df3..mean.
## accommodates                                  4.2451253
## bedrooms                                      1.7103064
## beds                                          2.3649025
## bathrooms                                     1.3495822
## price                                       181.1169916
## review_scores_rating                        94.4206128
## cleaning_fee                                73.7896936
## review_scores_value                          9.4832869
## review_scores_location                       9.8593315
## review_scores_accuracy                       9.6852368
## review_scores_communication                  9.7660167
## review_scores_cleanliness                    9.5041783
## host_response_rate                           0.9762813
##                             sd....sapply.df3..sd.
## accommodates                           1.94443667
## bedrooms                               0.70833229
## beds                                   1.47143681
## bathrooms                              0.50624396
## price                                127.23031847
## review_scores_rating                   7.40848160
## cleaning_fee                          47.15861309
## review_scores_value                    0.84939838
## review_scores_location                 0.46763072
## review_scores_accuracy                 0.74958241
## review_scores_communication            0.58893896
## review_scores_cleanliness              0.80052934
## host_response_rate                     0.08521014
##                             median....sapply.df3..median.
## accommodates                                            4
## bedrooms                                                2
## beds                                                    2
## bathrooms                                               1
## price                                                 150
## review_scores_rating                                   96
## cleaning_fee                                           75
## review_scores_value                                    10
## review_scores_location                                 10
## review_scores_accuracy                                 10
## review_scores_communication                            10
## review_scores_cleanliness                              10
## host_response_rate                                      1
##                             max....sapply.df3..max.
## accommodates                                     12
## bedrooms                                          6
## beds                                              9
## bathrooms                                         4
## price                                           999
## review_scores_rating                            100
## cleaning_fee                                    330
## review_scores_value                              10
## review_scores_location                           10
## review_scores_accuracy                           10
## review_scores_communication                      10
## review_scores_cleanliness                        10
## host_response_rate                                1
##                             min....sapply.df3..min.
## accommodates                                      1
## bedrooms                                          0
## beds                                              0
## bathrooms                                         1
## price                                            30
## review_scores_rating                             20
## cleaning_fee                                      0
## review_scores_value                               2
## review_scores_location                            4
## review_scores_accuracy                            2
## review_scores_communication                     4
## review_scores_cleanliness                       4
## host_response_rate                              0
```

By using the five summary statistics, we have the mean, median, minimum value, maximum value, and standard deviation in the above chart. For the chart, we decide to keep the original data since they are in a small data frame and they are easy to observe, as we do not need to worry about normalization too much. Instead, we can see more direct comparisons between different numbers. For example, we can the mean for location and accuracy review are very high (above 0.5), with a small standard deviation of differences as 0.4 to 0.7, but the minimum score would be 2, which is very low. It seems like most of the people are satisfied about the results, but a few of them may have other concerns. The cleaning fee on average is about 73 AUD. It seems like, normally the accommodations are about 4 people, 2 bedrooms, 2 beds, and 1 bathroom, with an average rating of 94.4 for the overall Airbnb Housing performance.

2. What is the most popular property type?

After we roughly have an idea of about the housing performance, our next question is what are the most popular property types? In the process below, we wonder what would be the most popular room type for people to choose in Southbank area. The chart shows that the most popular type is apartment (at 85% of the whole distribution), and the rest of the property types are similar, which provides very small amount of the type. This could make the investor pay attentions on investing possibly more types of property types to increase the diversity, or they can consider to only invest in apartment since a lot of the hosts have decided to rent their apartment types of housing, which could be easy to clean and customers may attract to the conveniences.

```
#2. What are the most popular property types?
type <- select(df1, host_is_superhost, property_type, room_type, bed_type)
type$host_is_superhost[type$host_is_superhost == ""] <- NA
type<- na.omit(type) #most complete data set rom these types
```

```
pt <- table(type$property_type)
pt2 <- sort(prop.table(pt), decreasing = TRUE)
pt3 <- head(pt2, 10)
pt3
##          Apartment       Condominium Serviced apartment
##       0.8516439455      0.0761828388       0.0609462711
##               Loft         Townhouse              House
##       0.0048115477      0.0024057739       0.0016038492
##              Other        Aparthotel               Barn
##       0.0016038492      0.0008019246       0.0000000000
##   Bed and breakfast
##       0.0000000000
```

3. Knowing most choices are apartment, then, what are the specific room types?

```
#3. what is the most popular room type?
roomtype <- data.frame(table(type$room_type))
names(roomtype)[names(roomtype) == "Var1"] <- "RoomType"
names(roomtype)[names(roomtype) == "Freq"] <- "Numbers"
roomtype
##          RoomType Numbers
## 1 Entire home/apt     915
## 2    Private room     309
## 3     Shared room      23
```

The number for Entire home/apt is the most popular one among all three-room types, the private room is the second with a number of 309, and the shared room includes 23 choices for the customers to choose here. It seems like host care about the customers' privacy and they would like to provide more housing resources on entire home/apt type. [By making the comparison, in the visualization part will provide a clear graph for people who are interested in investing existing or creative room type for the properties in the neighborhood.]

4. What is the most popular bed type?

Then, we can take a quick look at the bed type, which most of Airbnb offers real bed, as the probability of having real bed type at 99.91%. It further provides that hosts are not only caring about privacy for the customers, but they also would like the customers to have a comfortable stay.

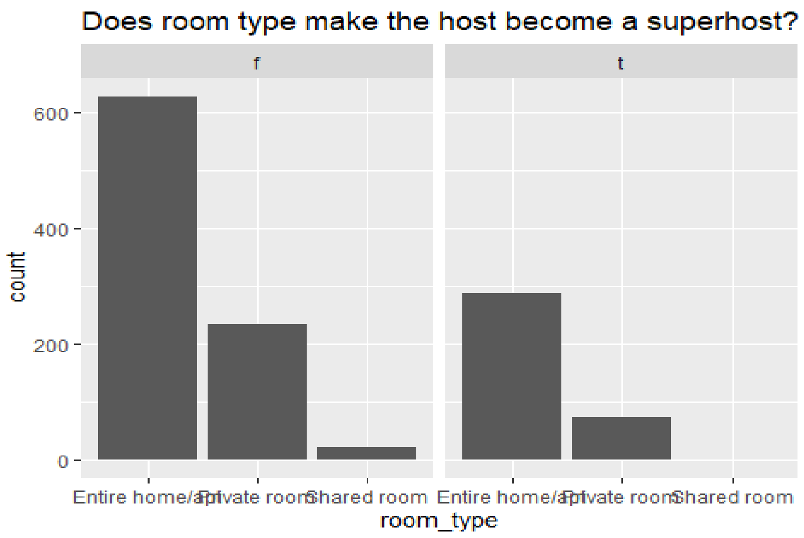
```
bt <- table(type$bed_type)
bt2 <- sort(prop.table(bt), decreasing = TRUE)
bt3 <- head(bt2, 10)
bt3
##      Real Bed         Futon        Airbed         Couch Pull-out Sofa
## 0.9991980754 0.0008019246 0.0000000000 0.0000000000  0.0000000000
```

III. Visualization: Property - Superhost - Price - Review - Listings

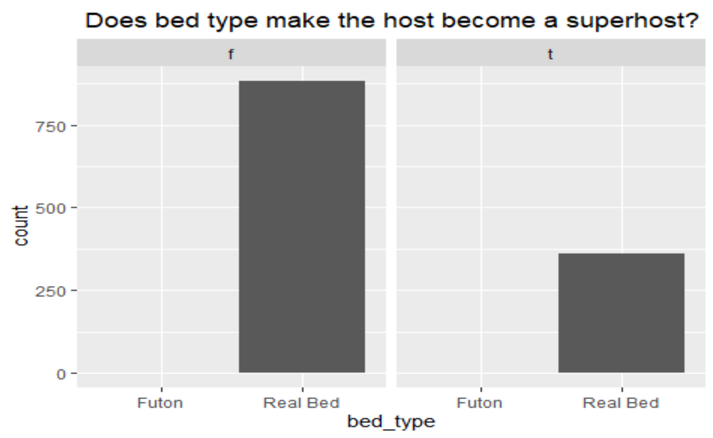
Visualization is easy for the stakeholders, companies, or even customers to know Airbnb's market in Southbank, and how certain information will be portrayed in this part. The easy-to-understanding graph, discerning charts, and the simple labels can provide an overview for the whole neighborhood.

1. <u>Possible reasons for whether the host is a Superhost – facet graph with bar plot</u>

```
a <- table(type$host_is_superhost)
prop.table(a)
                  f         t
## 0.0000000 0.7097033 0.2902967

ggplot(type, aes(x=room_type)) + geom_bar() + facet_grid(~host_is_superhost) +
  ggtitle("Does room type make the host become a superhost?") + theme(plot.title =
element_text(hjust = 0.5))
```



```
ggplot(type, aes(x=bed_type)) + geom_bar() + facet_grid(~host_is_superhost) +
  ggtitle("Does bed type make the host become a superhost?") + theme(plot.title =
element_text(hjust = 0.5))
```

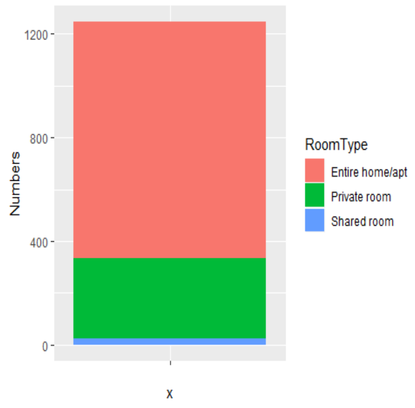


```
#no relation with any of the types above,
#but it shows the proportion of what is the most option that hosts offer
```

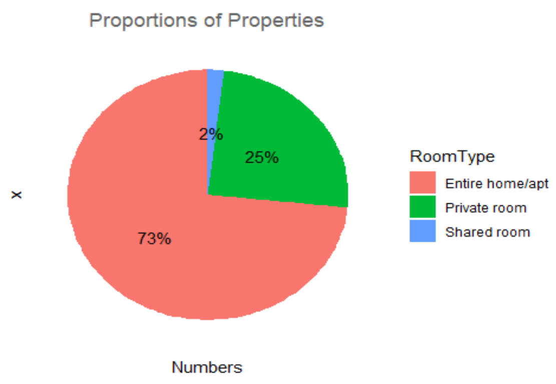
Since Airbnb is the company providing good service and satisfied quality housing resources to the customers, the host will be a very important role for the customers to trust and rely on. By looking at the faceting charts, the relationship between bed type/room type and whether the host is a superhost actually may not impact on each other too much. In the data set, there are 71% ("f=false") of the hosts are not superhost, and only 29% ("t=true") of the hosts are superhost. And among all the superhosts, they offer most of their room type is "Entire home/apt", and real bed is their most choice also. No matter how the proportion of whether the host is a super host or not, the data on regular host is always higher, and the trend for room type/bed type is the same. There are two possible reasons to consider for the results: 1. Hosts are not offering spectacular services for the residents, or 2. Southbank is not very popular in investing good housing resources, which means our company can really help Airbnb in the Southbank to grow with many potential opportunities.

2. Property type - pie chart

```
library(scales)
roomtype <- data.frame(table(type$room_type))
names(roomtype)[names(roomtype) == "Var1"] <- "RoomType"
names(roomtype)[names(roomtype) == "Freq"] <- "Numbers"
roomtype
##          RoomType Numbers
## 1 Entire home/apt     915
## 2    Private room     309
## 3     Shared room      23
b <- ggplot(roomtype, aes(x="", y=Numbers, fill=RoomType)) + geom_bar(width = 1, stat =
"identity")
b
```



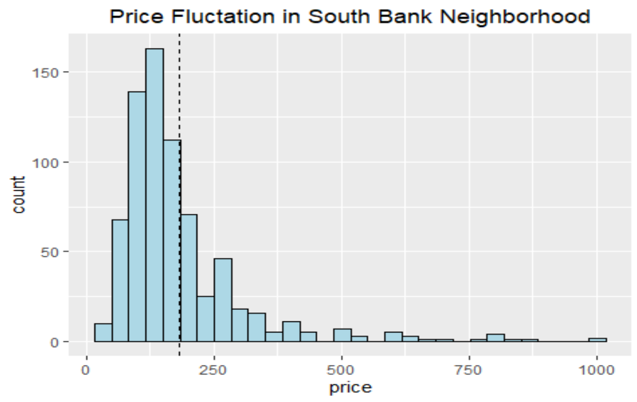
```
pie <- b + coord_polar("y", start=0) +
  geom_text(aes(label=paste0(round(roomtype$Numbers/sum(roomtype$Numbers)*100),"%")),
            position = position_stack(vjust = 0.5)) +
  ggtitle("Proportions of Properties") + theme(plot.title = element_text(hjust = 0.5))
pie = pie + theme_classic() + theme(axis.line = element_blank(),
                                    axis.text = element_blank(),
                                    axis.ticks = element_blank(),
                                    plot.title = element_text(hjust = 0.5, color = "#666666"))
pie
```



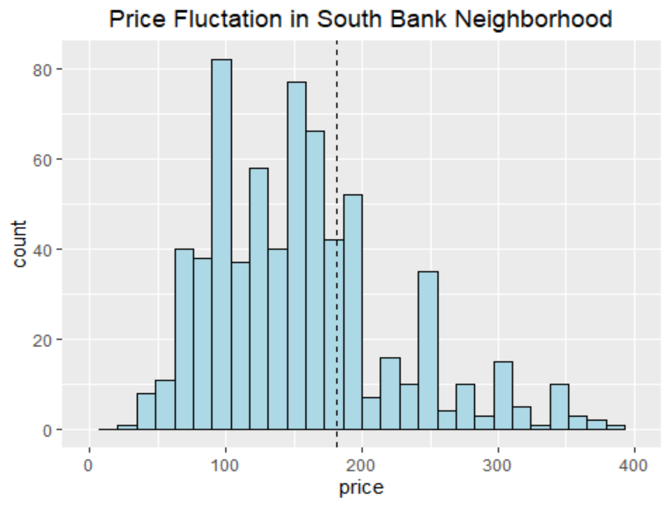
Now, let's look at the pie chart for the property type in this neighborhood: 73% of the properties are "Entire home/apt", 25% are the private rooms, and only 2% is the shared room. This could entail that the host in this neighborhood mostly offering their invested home/apt, or they no longer live in the region, and there is a quite good number of people who share their rooms with the guests.

3. Price fluctuation graph - histogram - since our data is not huge,

```
#I will keep as it looks, instead of normalization....
c1 <- ggplot(df3, aes(x=price))
c1 + geom_histogram(bins=30, color = "black", fill="light blue") +
  geom_vline(aes(xintercept = mean(price)),linetype="dashed",size=0.6) +
  ggtitle("Price Fluctation in South Bank Neighborhood") + theme(plot.title =
element_text(hjust = 0.5))
```



```
c2 <- ggplot(df3, aes(x=price)) + xlim(0,400) #ignore outliners are larger than 400
c2 + geom_histogram(bins=30, color = "black", fill="light blue") +
  geom_vline(aes(xintercept = mean(price)),linetype="dashed",size=0.6) +
  ggtitle("Price Fluctation in South Bank Neighborhood") + theme(plot.title =
element_text(hjust = 0.5))

## Warning: Removed 34 rows containing non-finite values (stat_bin).

## Warning: Removed 2 rows containing missing values (geom_bar).
```

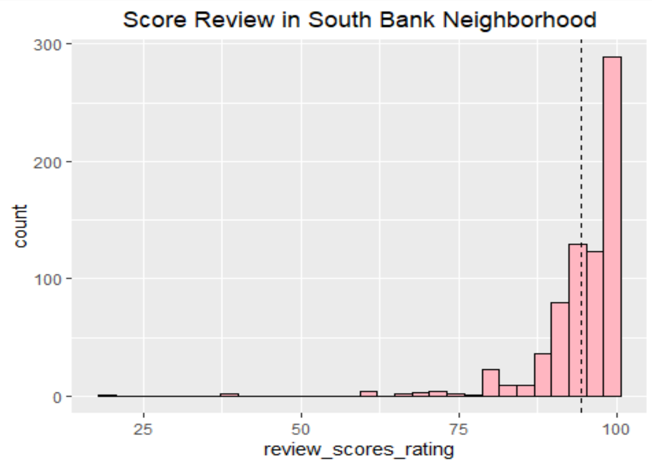


```
mean(df3$price) #approx. 182

## [1] 181.117
```

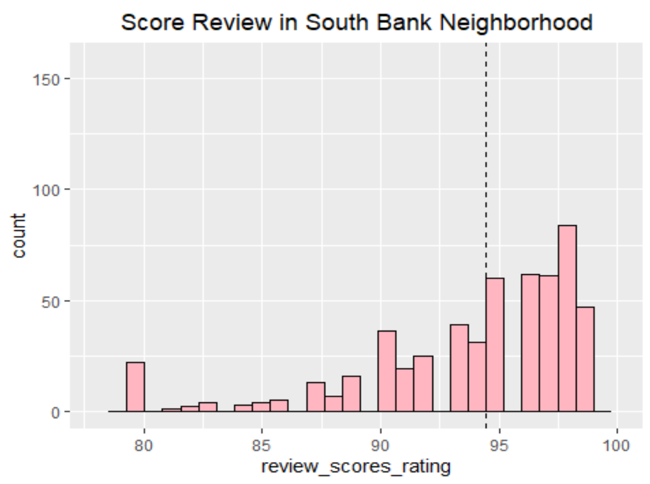
As the first graphs provides an idea of the price's fluctuation for the prices, the second graph provides a mroe clear look at the most distributed part of the histogram. The mean of the price is about 181 AUD, and second graphs show many bars on the left side of the mean, which could indicate the outliners could be very pricy, and they drag the price a little higher than most of the bars. This could help the company search more background for the pricing strategy, which could really help increase the scores review, as shown in the next step.

4. Review score graph - histogram

```
d1 <- ggplot(df3, aes(x=review_scores_rating))
d1 + geom_histogram(bins=30, color = "black", fill="light pink")+
  geom_vline(aes(xintercept = mean(review_scores_rating)),linetype="dashed",size=0.6)+
  ggtitle("Score Review in South Bank Neighborhood") + theme(plot.title = element_text(hjust =
0.5))
```



```
d2 <- ggplot(df3, aes(x=review_scores_rating)) + xlim(78,100) #enlarge the graph and see clear
distribution
d2 + geom_histogram(bins=30, color = "black", fill="light pink")+
  geom_vline(aes(xintercept = mean(review_scores_rating)),linetype="dashed",size=0.6)+
  ggtitle("Score Review in South Bank Neighborhood") + theme(plot.title = element_text(hjust =
0.5))
```

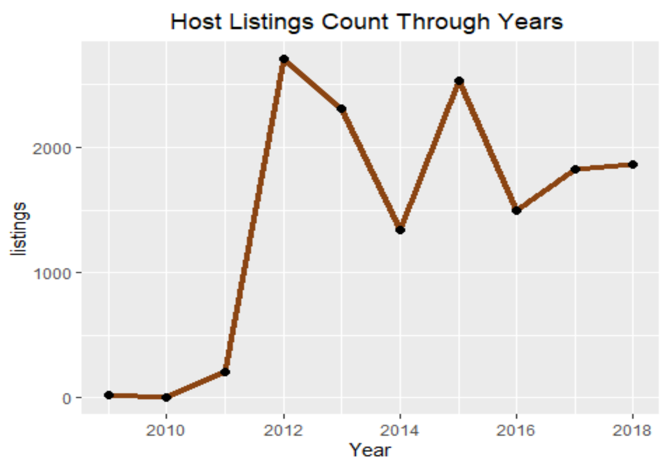


```
mean(df3$review_scores_rating) #94.4

## [1] 94.42061
```

In the above graph, we have the similar graph from the price graph but way more shaped toward the right side. As we look closer, we can see the disparities are wide, and the mean of the review is 94%, which could show the reviews are not very uniform into one direction from the customers' side, and there might have a lot space to improve the scores through later analysis in the report.

5. <u>Listings through years – line graph</u>

```
df4 <- select(df1, host_since, calculated_host_listings_count)
sum(is.na(df4))

## [1] 0

df4$host_since[df4$host_since == ""] <- NA
df4 <- na.omit(df4)
df4$Year <- as.Date(df4$host_since, "%m/%d/%Y")
df4$Year <- as.numeric(format(df4$Year, "%Y"))
str(df4)

df5 <- group_by(df4, Year) %>%
  summarize(listings=sum(calculated_host_listings_count))

ggplot(df5, aes(x=Year, y=listings)) + geom_line(size=1.5, color="chocolate4") +
geom_point(size=2) +
  ggtitle("Host Listings Count Through Years ") + theme(plot.title = element_text(hjust =
0.5))
```

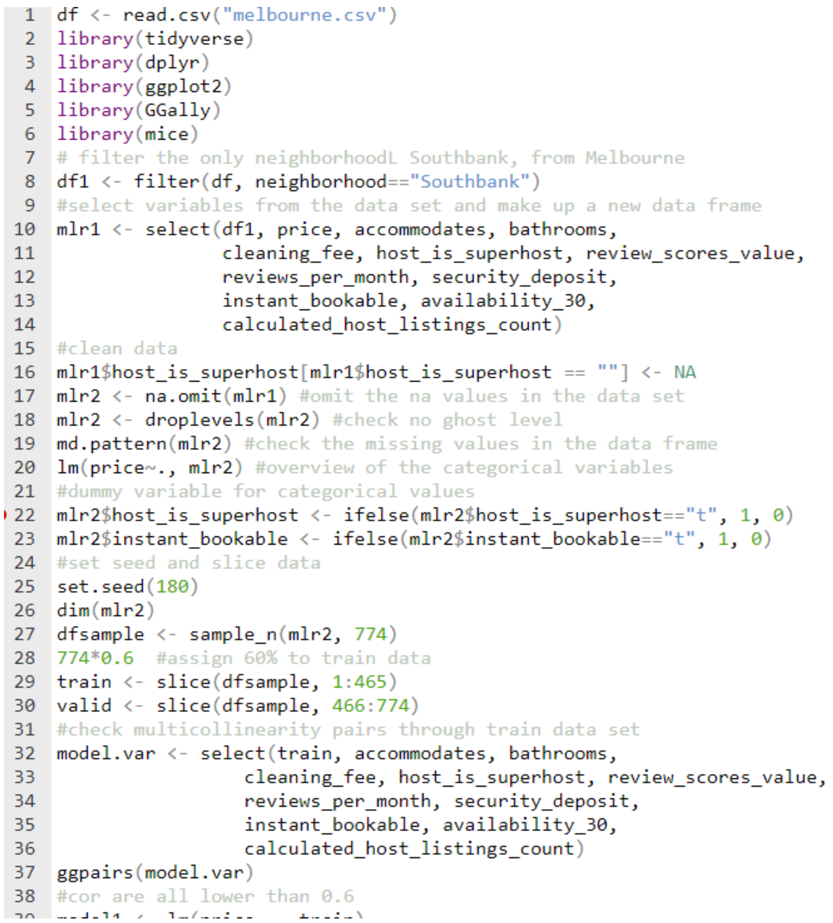


Last but not least, the listings are counted in the line graph in the order of yearly manner. From 2009 to 2018, the listings had dramatically increased in 2011, while it dropped again in 2014, and it increased again. The instability could be a huge concern for Airbnb company, and the market in Southbank. The trend for 2017 and 2018 seems to go up, but the fluctuation may result in uncertainties. There are definitely improvements for Airbnb housing in Southbank to achieve in the next step. From drawing the general overview of the statistics in Southbank from Airbnb data frame, we still need to dive in-depth and dig more specific information, in order to make our strategy happen.

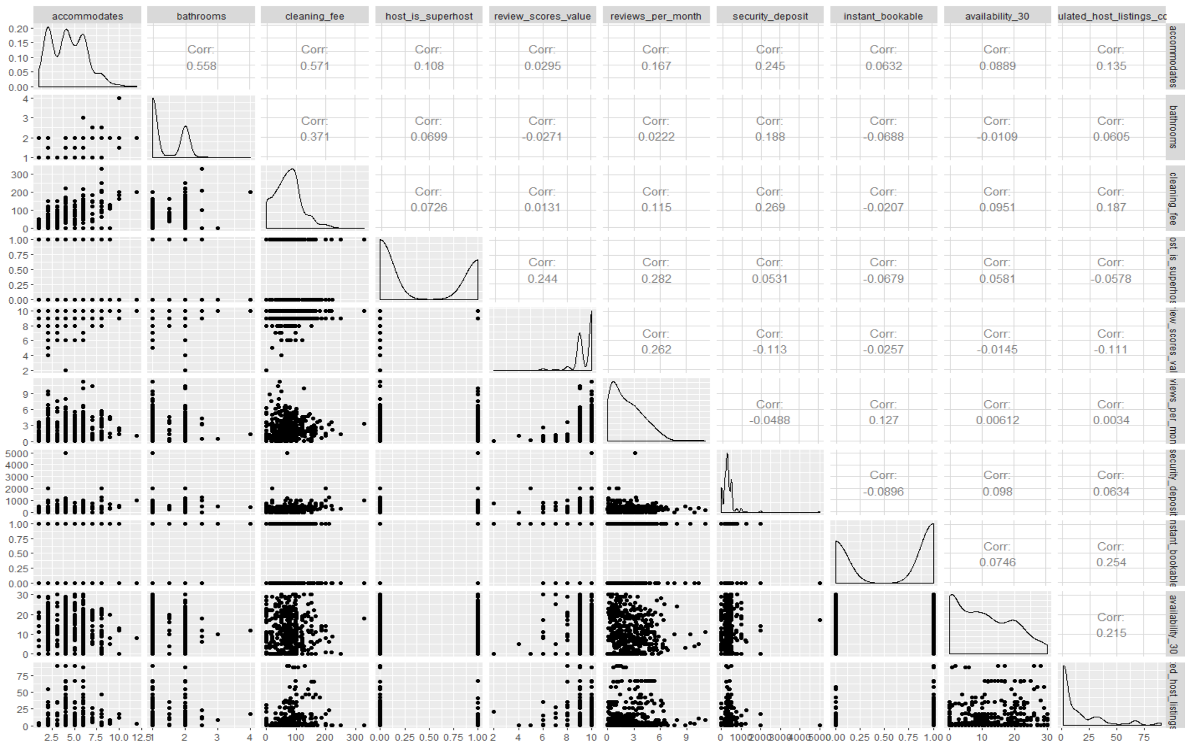
Step II: Prediction: Multiple Regression Model

1. Process Description and Quality of the Model

By importing data and useful packages, we filter the Southbank's data and select the variables may influence the outcome value, price. We consider accommodates, beds and bedroom as the same category, so we only need to take one variable from these three. Then, other variables can all influence the price in its time availability, whether experiences host or not, extra fees for security deposit or cleaning, reviews per month and score for the value. Those are all key performances indicators for building the multiple regression model. Before processing any model further, we need to omit all the missing values (include blank cells and NA values). Having the regression model built in line 20 would help us look at the categorical values' performances, and then they will be transformed into dummy variables for further analysis. For the superhost and instant bookable chance they will be listed as 1, and the other option will be 0. Then, sampling the training set and slicing the data into two data sets: training and validation. Next step, we can select the variables besides the outcome value, and put them into the "model.var", and we would like to see the correlations among these variables, in order to prevent from multicollinearities. Fortunately, all correlation values are lower than 0.6, and we can safely proceed to the next step.

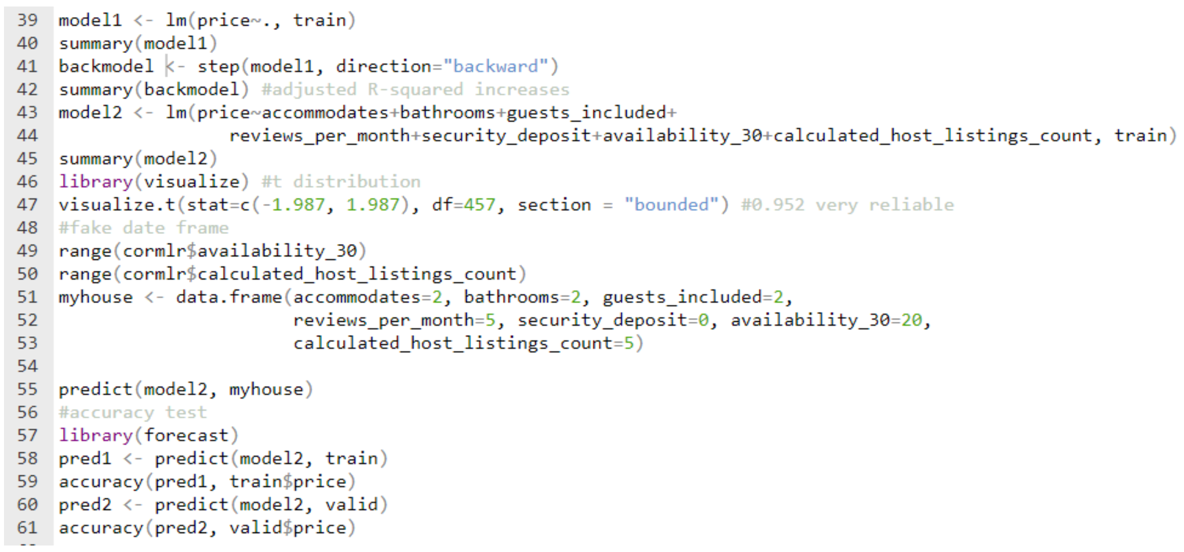


```r
df <- read.csv("melbourne.csv")
library(tidyverse)
library(dplyr)
library(ggplot2)
library(GGally)
library(mice)
# filter the only neighborhoodL Southbank, from Melbourne
df1 <- filter(df, neighborhood=="Southbank")
#select variables from the data set and make up a new data frame
mlr1 <- select(df1, price, accommodates, bathrooms,
               cleaning_fee, host_is_superhost, review_scores_value,
               reviews_per_month, security_deposit,
               instant_bookable, availability_30,
               calculated_host_listings_count)
#clean data
mlr1$host_is_superhost[mlr1$host_is_superhost == ""] <- NA
mlr2 <- na.omit(mlr1) #omit the na values in the data set
mlr2 <- droplevels(mlr2) #check no ghost level
md.pattern(mlr2) #check the missing values in the data frame
lm(price~., mlr2) #overview of the categorical variables
#dummy variable for categorical values
mlr2$host_is_superhost <- ifelse(mlr2$host_is_superhost=="t", 1, 0)
mlr2$instant_bookable <- ifelse(mlr2$instant_bookable=="t", 1, 0)
#set seed and slice data
set.seed(180)
dim(mlr2)
dfsample <- sample_n(mlr2, 774)
774*0.6  #assign 60% to train data
train <- slice(dfsample, 1:465)
valid <- slice(dfsample, 466:774)
#check multicollinearity pairs through train data set
model.var <- select(train, accommodates, bathrooms,
                    cleaning_fee, host_is_superhost, review_scores_value,
                    reviews_per_month, security_deposit,
                    instant_bookable, availability_30,
                    calculated_host_listings_count)
ggpairs(model.var)
#cor are all lower than 0.6
model1 <- lm(price ~., train)
```

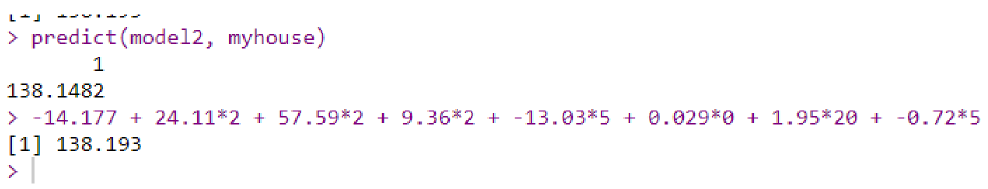


Through summarizing the data from the previous step, we would like to see the model values (in the below graph). For the first model, it includes all possible predictors, and it is easy to say some predictors are not very useful for predicting the outcome variables, based on the significant codes, but the second one eliminate many redundant variables and improve the quality of the model. During the process, we use backward elimination to help us reduce the variables, so we have better predications for the model. By eliminating cleaning fee, superhost, instant bookable, and scores, we have a much better model, whereas the adjusted R-square changes from 0.4213 to 0.4244. Those variables do not seem to impact on the model much, and we follow the elimination and get ride them. We keep the rest of the variables, as shown here: "accommodates+bathrooms+guests_included+reviews_per_month+security_deposit+availability_30+calculated_host_listings_count". For further proof of the quality of the model with keeping these variables, here are some ways to prove: when the p value is greater than others, the t-vale is -1.987 to 1.987 ("security_deposit") for the visualized distribution, which provides the confidence of 0.952 (shaded area), and the model is very reliable. For the relative larger p value for this predictor, the t values shows a wider probability of distribution, which further proves that all the selected predictors should be reliable choices to predict the outcome through the multiple regression model as other predictors' p values are all smaller.

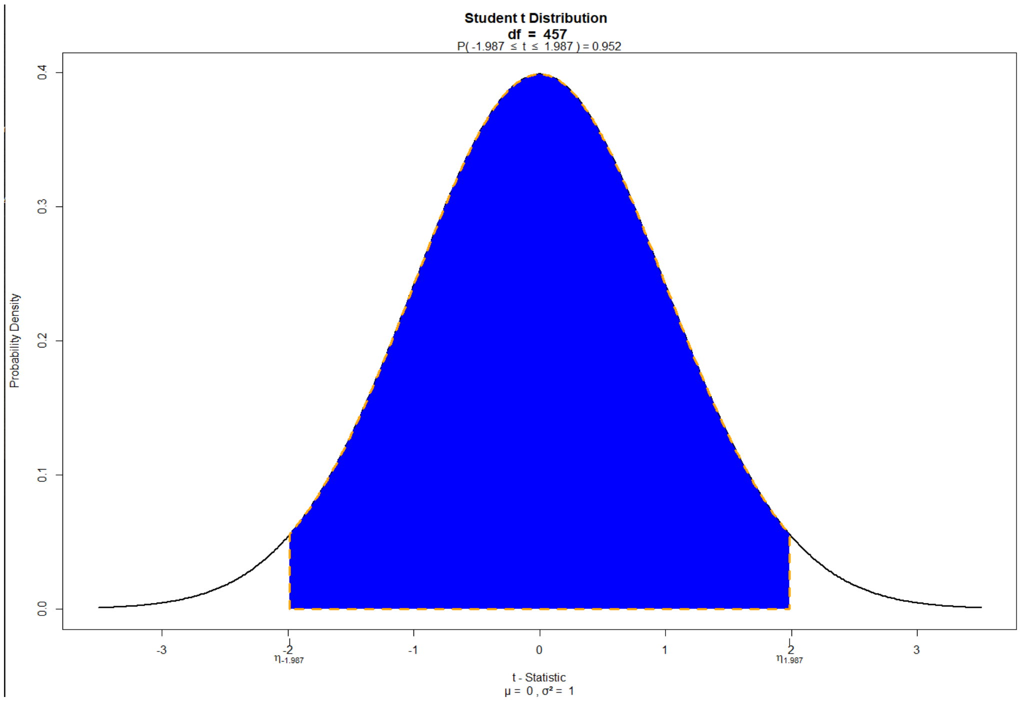
Last but not least, we build "my house" based on the variables of the last model, and we have a predicated value of 138 AUD. The equation that the model generates is: y= -14.177 + 24.11*a + 57.59*b + 9.36*c + -13.03*d + 0.029*e + 1.95*f + -0.72*g, which provides very similar result from the predication. Then we will look at the accuracy of both data sets. Since we build the model based on the training set, the errors in training data set is smaller than the validation set, which further proves the good quality of this model.

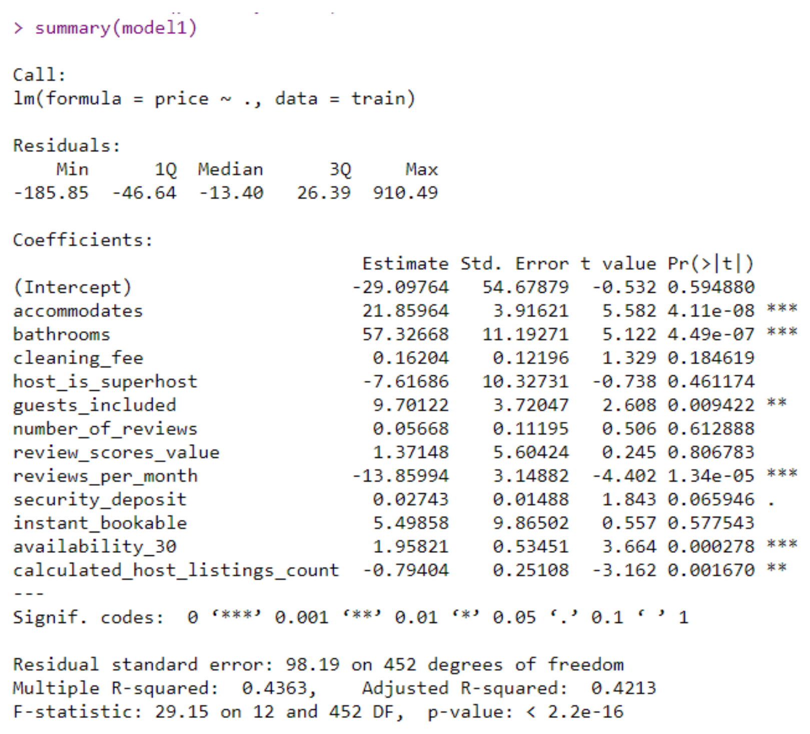


```
model1 <- lm(price~., train)
summary(model1)
backmodel <- step(model1, direction="backward")
summary(backmodel) #adjusted R-squared increases
model2 <- lm(price~accommodates+bathrooms+guests_included+
                reviews_per_month+security_deposit+availability_30+calculated_host_listings_count, train)
summary(model2)
library(visualize) #t distribution
visualize.t(stat=c(-1.987, 1.987), df=457, section = "bounded") #0.952 very reliable
#fake date frame
range(cormlr$availability_30)
range(cormlr$calculated_host_listings_count)
myhouse <- data.frame(accommodates=2, bathrooms=2, guests_included=2,
                      reviews_per_month=5, security_deposit=0, availability_30=20,
                      calculated_host_listings_count=5)

predict(model2, myhouse)
#accuracy test
library(forecast)
pred1 <- predict(model2, train)
accuracy(pred1, train$price)
pred2 <- predict(model2, valid)
accuracy(pred2, valid$price)
```

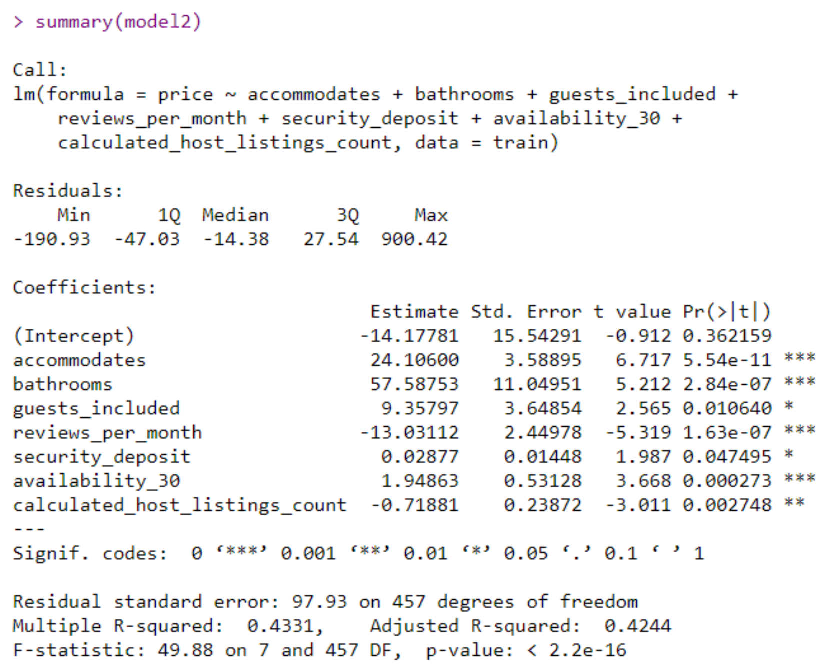


```
> predict(model2, myhouse)
       1
138.1482
> -14.177 + 24.11*2 + 57.59*2 + 9.36*2 + -13.03*5 + 0.029*0 + 1.95*20 + -0.72*5
[1] 138.193
>
```

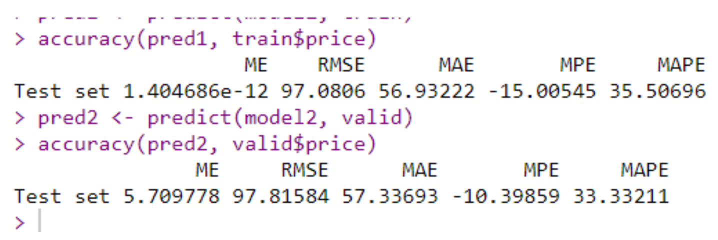




```
> summary(model1)

Call:
lm(formula = price ~ ., data = train)

Residuals:
    Min      1Q  Median      3Q     Max 
-185.85  -46.64  -13.40   26.39  910.49 

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)    
(Intercept)                  -29.09764   54.67879  -0.532 0.594880    
accommodates                  21.85964    3.91621   5.582 4.11e-08 ***
bathrooms                     57.32668   11.19271   5.122 4.49e-07 ***
cleaning_fee                   0.16204    0.12196   1.329 0.184619    
host_is_superhost             -7.61686   10.32731  -0.738 0.461174    
guests_included                9.70122    3.72047   2.608 0.009422 ** 
number_of_reviews              0.05668    0.11195   0.506 0.612888    
review_scores_value            1.37148    5.60424   0.245 0.806783    
reviews_per_month            -13.85994    3.14882  -4.402 1.34e-05 ***
security_deposit               0.02743    0.01488   1.843 0.065946 .  
instant_bookable               5.49858    9.86502   0.557 0.577543    
availability_30                1.95821    0.53451   3.664 0.000278 ***
calculated_host_listings_count -0.79404   0.25108  -3.162 0.001670 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 98.19 on 452 degrees of freedom
Multiple R-squared:  0.4363,	Adjusted R-squared:  0.4213 
F-statistic: 29.15 on 12 and 452 DF,  p-value: < 2.2e-16
```



```
> summary(model2)

Call:
lm(formula = price ~ accommodates + bathrooms + guests_included + 
    reviews_per_month + security_deposit + availability_30 + 
    calculated_host_listings_count, data = train)

Residuals:
    Min      1Q  Median      3Q     Max 
-190.93  -47.03  -14.38   27.54  900.42 

Coefficients:
                               Estimate Std. Error t value Pr(>|t|)    
(Intercept)                   -14.17781   15.54291  -0.912 0.362159    
accommodates                   24.10600    3.58895   6.717 5.54e-11 ***
bathrooms                      57.58753   11.04951   5.212 2.84e-07 ***
guests_included                 9.35797    3.64854   2.565 0.010640 *  
reviews_per_month             -13.03112    2.44978  -5.319 1.63e-07 ***
security_deposit                0.02877    0.01448   1.987 0.047495 *  
availability_30                 1.94863    0.53128   3.668 0.000273 ***
calculated_host_listings_count -0.71881    0.23872  -3.011 0.002748 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 97.93 on 457 degrees of freedom
Multiple R-squared:  0.4331,	Adjusted R-squared:  0.4244 
F-statistic: 49.88 on 7 and 457 DF,  p-value: < 2.2e-16
```



```
> accuracy(pred1, train$price)
                   ME    RMSE      MAE       MPE     MAPE
Test set 1.404686e-12 97.0806 56.93222 -15.00545 35.50696
> pred2 <- predict(model2, valid)
> accuracy(pred2, valid$price)
               ME     RMSE      MAE       MPE     MAPE
Test set 5.709778 97.81584 57.33693 -10.39859 33.33211
> 
```

2. R-Squared and RMSE

R-squared for my first model with all predicators is 0.4363, and adjusted R-squared is 0.4213. The R-squared value for my second model with selected predicators is 0.4331, and adjusted R-squared is 0.4244. R-squared indicates the proportion of variation in the outcome variable, and it SSR/SST, which the sum of squared error divides by the sum of square total. As the R-squared is smaller from the first model, it means that the sum of squared error is smaller, which is caused by the elimination for reducing the variables. In contrary for adjusted R-squared, it modifies the variation in the dependent variable as R-squared only supposes all variations. The adjusted R-squared increases, which means the redundancy of the model is reduced and the model is more efficient.

RMSE is the square root of mean squared error, and it provides the scale of the errors in comparisons to the scale of the targets. Since the model is built on training set, the model's RMSE (97.08) is smaller, comparing to the one (97.82) in the validation set. By selecting variables to the regression, the training set seems to have better accuracy results than the validation, but it is understandable that train model is used to find the better variables to predict a closer model with less errors. Usually the validation set is used to evaluate the model's performance, and it tends to have larger errors than the training set. But we do not want dramatic differences in either errors, or between two data sets, then there would be some overfit situation in the model. So far, our model is very reliable, the errors are small, and it is a good fit to the predications.

Step III: Classification

1. K-nearest Neighbors: Cancellation Policy

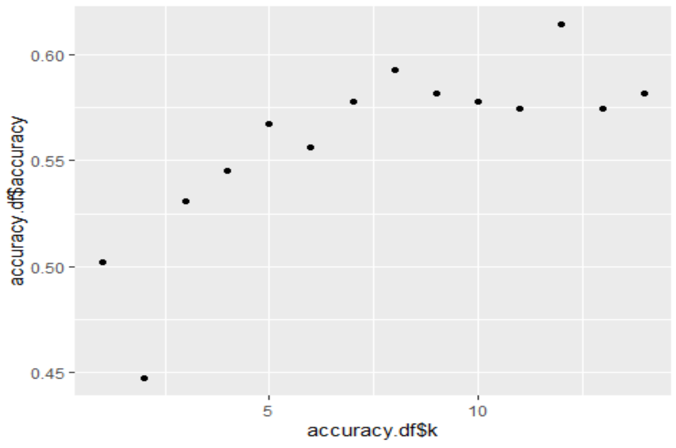
First, we put our useful packages in the top and read our data into our R environment. Then, we filter the data to only our neighborhood, and we select related other variables into the data frame as well. By transforming certain values into numeric, filtering the missing values and NA values, now we have a clean data frame with the most useful data to predict the classification of our cancellation policy. Then, we set our seed values to 180 and slice our data into training set and validation set. We build our new fake rental property, as we have decided the variables: host response rate (whether the host is fast enough to reply to customer's request will directly impact on the cancellation for the booked reservations), price (if the customer find some cheaper housing, then he/she could probably cancel their existing house), security deposit (if the customer has paid the deposit, they may not get the money back due to cancellation rules), and review scores communication (the review on communication's efficiency could also lead to cancellation since the customer may not feel comfortable talking to the host). And our cancellation policy has strict, moderate or flexible options. Then, applying through the normalization process to have new data frame, which shows the rental's certain qualities into smaller values, and they are easier to compare into different classifications.

```r
library(dplyr)
library(FNN)
library(caret)
library(ggplot2)
library(tidyr)
library(e1071)
library(mice)
# get data
survey <- read.csv("melbourne.csv")
neihbor <- filter(survey,survey$neighborhood == "Southbank")
predictor <- select(neihbor, cancellation_policy, host_response_rate, price, security_deposit,
 review_scores_communication)
# process data
predictor$host_response_rate<-as.numeric(sub("%","",predictor$host_response_rate))/100
```

```
predictor$host_response_rate[predictor$host_response_rate == "N/A"]<-NA
predictor$host_response_rate[predictor$host_response_rate == ""]<-NA
predictor$review_scores_communication[predictor$review_scores_communication == "N/A"]<-NA
predictor$review_scores_communication[predictor$review_scores_communication == ""]<-NA
predictor$security_deposit[predictor$security_deposit == "N/A"]<-NA
predictor$security_deposit[predictor$security_deposit == ""]<-NA
predictor1 <- na.omit(predictor)
anyNA(predictor1)
## [1] FALSE

#  divide into training set and validation set
set.seed(180)
predictor2 <- sample_n(predictor1, 689)
train.df <- slice(predictor2, 1:414)
valid.df <- slice(predictor2, 415:689)
# establish a new rental
host_response_rate <- runif(1, min(predictor1$host_response_rate),max(predictor1$host_response
_rate))
review_scores_communication <-runif(1, min(predictor1$review_scores_communication),max(predict
or1$review_scores_communication))
security_deposit <-runif(1, min(predictor1$security_deposit),max(predictor1$security_deposit))
price <-runif(1, min(predictor1$price),max(predictor1$price))
new.rental <- data.frame(host_response_rate, review_scores_communication, security_deposit, pr
ice)
# normalization data
train.norm.df <- train.df
valid.norm.df <- valid.df
norm.values <- preProcess(train.df[, 2:5], method=c("center", "scale"))
train.norm.df[, 2:5] <- predict(norm.values, train.df[, 2:5])
valid.norm.df[, 2:5] <- predict(norm.values, valid.df[, 2:5])
new.rental.norm.df <- predict(norm.values, new.rental)
# build knn model
nn <- knn(train = train.norm.df[, 2:5], test = new.rental.norm.df, cl = train.norm.df[, 1], k
= 7)
row.names(train.df)[attr(nn, "nn.index")]

## [1] "109" "16"  "394" "211" "273" "174" "133"

new.rental.neighbor <- train.df[c(109,16,394,211,273,174,133),]
new.rental.neighbor$cancellation_policy #my policy is flexible

## [1] flexible flexible flexible flexible flexible moderate flexible
## 6 Levels: flexible moderate strict ... super_strict_60
```

Then, we choose 7 as our k values randomly, and we would like to what categories our model will fall into. As applying the values into knn model, the “nn,index” provides the row identity of larger probability of the classification in Cancellation Policy, which is “flexible (109), flexible (16), flexible (394), flexible (211), flexible (273), moderate (174), flexible (133)”. The most outcome is flexible for our model’s cancellation policy. After initializing the data frame, we would like to see how accurate our model would be. Since our data set contains limited data, we would like to see which the most outcome will fall into so we choose 14 as our k values to re-run our model. And we have the best result at 61.45% when the k value is 12. In the graph, it further visualize the result for the values, and most of the data falls into between 55% to 60%, and it exceeds half of the data set, which I think it is reliable since the there are only three options for the cancellation policy. Furthermore, the detailed result for the class predication when k value is 12, it shows that most frequent appearance of the housing falls into **flexible policy** for cancellation, which we think it is very reliable to classify the cancellation policy by k-nearest neighbors' model.

```
accuracy.df <- data.frame(k = seq(1, 14, 1), accuracy = rep(0, 14))
for(i in 1:14) {
  knn.pred <- knn(train.norm.df[, 2:5], valid.norm.df[, 2:5],
                  cl = train.norm.df[, 1], k = i)
  accuracy.df[i, 2] <- confusionMatrix(knn.pred, valid.norm.df[, 1])$overall[1] }
accuracy.df # the best is that k=12

##     k  accuracy
## 1   1 0.5018182
## 2   2 0.4472727
## 3   3 0.5309091
## 4   4 0.5454545
## 5   5 0.5672727
## 6   6 0.5563636
## 7   7 0.5781818
## 8   8 0.5927273
## 9   9 0.5818182
## 10 10 0.5781818
## 11 11 0.5745455
## 12 12 0.6145455
## 13 13 0.5745455
## 14 14 0.5818182

# vislization about K and accuracy
ggplot(accuracy.df, aes(x=accuracy.df$k, y=accuracy.df$accuracy))+geom_point()
```



```
knn12 <-knn(train=train.norm.df[,2:5],test=new.rental.norm.df,cl=train.norm.df[,1],k=12)
knn12
```

```
## [1] flexible
## attr(,"nn.index")
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
## [1,]  109   16  394  211  273  174  133  161  370   223   282   106
## attr(,"nn.dist")
##          [,1]    [,2]     [,3]     [,4]     [,5]     [,6]     [,7]    [,8]
## [1,] 2.648375 2.65362 2.682262 2.801244 2.913901 3.192876 3.367676 3.57006
##          [,9]    [,10]    [,11]    [,12]
## [1,] 3.631248 4.031653 4.041445 4.073051
## Levels: flexible
```

```
row.names(train.df)[attr(knn12,"nn.index")]
```

```
##  [1] "109" "16"  "394" "211" "273" "174" "133" "161" "370" "223" "282"
## [12] "106"
```

```
new.rental.neighbor_valid <- train.df[c(109,16,394,211,174,133,161,370,223,282,106),]
new.rental.neighbor_valid$cancellation_policy #my policy is flexible
```

```
##  [1] flexible                    flexible
##  [3] flexible                    flexible
##  [5] moderate                    flexible
##  [7] flexible                    strict_14_with_grace_period
##  [9] moderate                    flexible
## [11] moderate
## 6 Levels: flexible moderate strict ... super_strict_60
```

2. Naive Bayes: Instantly Bookable Condition

Firstly, preparing data. Based on the actual experience, there are four predictors that are selected to measure whether a particular rental will be instantly bookable: *host response rate, beds, availability_30, and cancellation policy.*

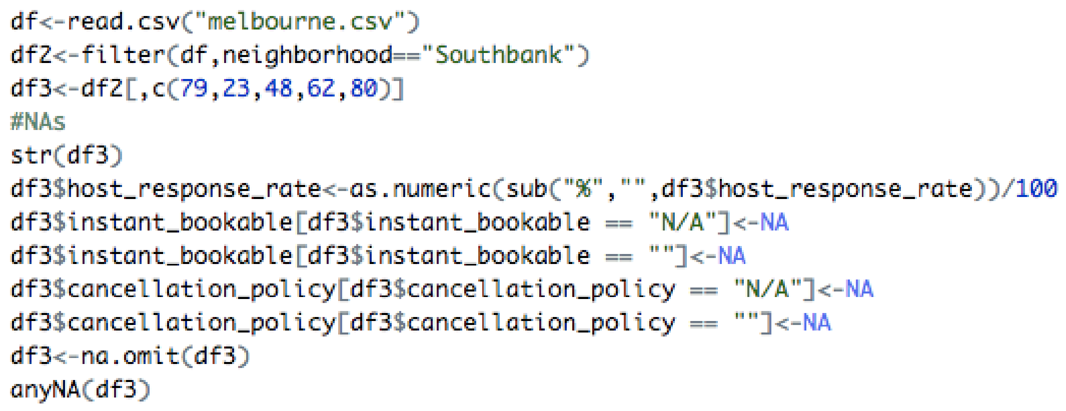
a) Host response rate: if all questions can be responded by the host, then guests will consider the host is very nice, a nice host is an important factor when choosing apartment.

b) The number of beds: most of guests travel with friends or family, so they will consider if the apartment has more than one bed as a condition.

c) Available days in a month: for guests who will stay more than one week, they will care about the available days in a month. It's inconvenient for them to change to another one, so they prefer an apartment for almost whole month is available.

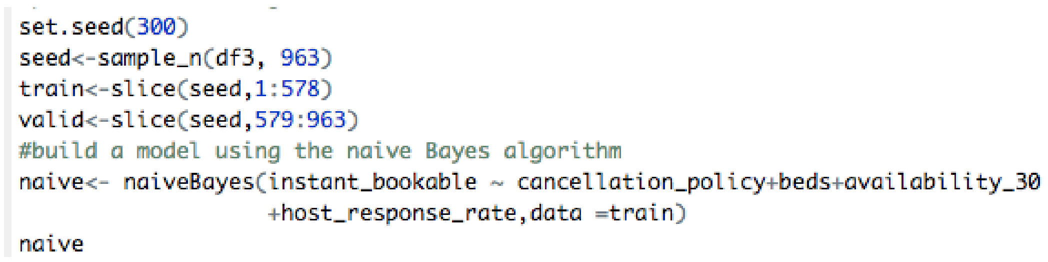
d) Cancellation policy: due to some reasons, guests usually have to cancel the book, but at the same time, they do not want to lose deposit, that's why they prefer an apartment that has a flexible cancellation policy, such as guests can cancel one day before the scheduled time without any deposit.

Therefore, these four variables are considered as predictors of instantly bookable. Select them from data set and then drop missing value and the data that contains NAs.

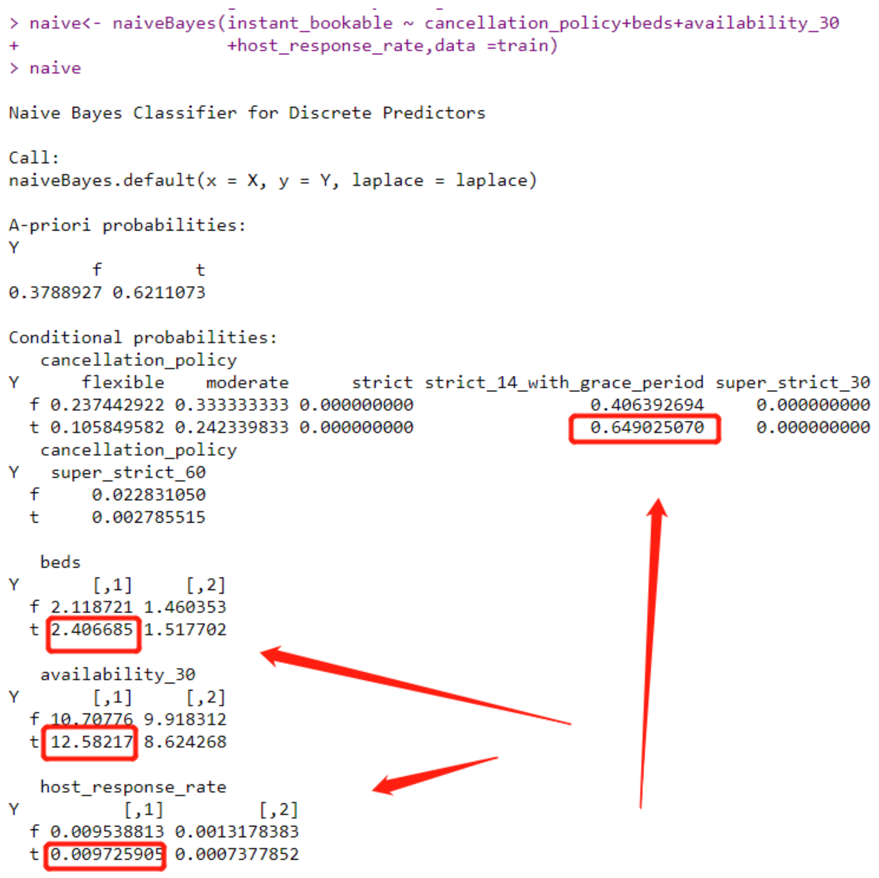


```
df<-read.csv("melbourne.csv")
df2<-filter(df,neighborhood=="Southbank")
df3<-df2[,c(79,23,48,62,80)]
#NAs
str(df3)
df3$host_response_rate<-as.numeric(sub("%","",df3$host_response_rate))/100
df3$instant_bookable[df3$instant_bookable == "N/A"]<-NA
df3$instant_bookable[df3$instant_bookable == ""]<-NA
df3$cancellation_policy[df3$cancellation_policy == "N/A"]<-NA
df3$cancellation_policy[df3$cancellation_policy == ""]<-NA
df3<-na.omit(df3)
anyNA(df3)
```

Secondly, separate data into training and validation parts, use the training part to build a model between instantly bookable and four predictors by using naive Bayes algorithm. In the screenshot below, we see that for the not a chance of the instant bookable housing has 37.89%, and most of the chances would be 62.11%, which would have the particular rental to be instant bookable type. Now, let’s look at the conditional probabilities for the variables’ chances to help or prevent the house from becoming an instant bookable type. The higher probability of impacting the instant bookable condition include “strict 14 with grace periods” for cancellation policy, one bed, more availability of a month, and more responsive host. It seems like customers would like to have a medium level of flexibility of the cancellation, more suitable fore one or two people’s travelling in an advance of 30 days, and they hope to see more responsive hosts. Then, let’s test our model’s accuracy and testing result.

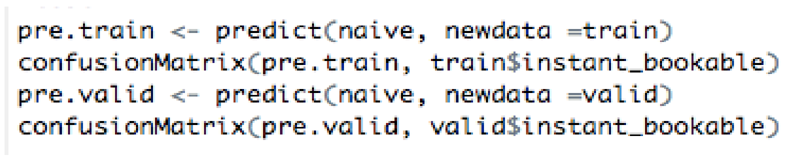


```
set.seed(300)
seed<-sample_n(df3, 963)
train<-slice(seed,1:578)
valid<-slice(seed,579:963)
#build a model using the naive Bayes algorithm
naive<- naiveBayes(instant_bookable ~ cancellation_policy+beds+availability_30
                   +host_response_rate,data =train)
naive
```

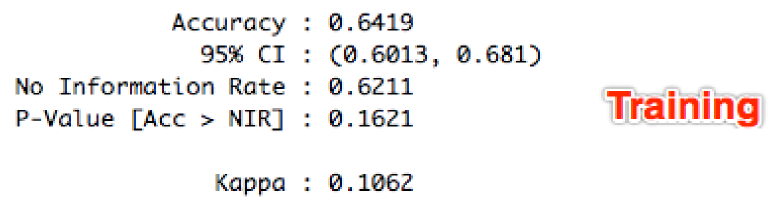


```
> naive<- naiveBayes(instant_bookable ~ cancellation_policy+beds+availability_30
+                    +host_response_rate,data =train)
> naive

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
        f         t 
0.3788927 0.6211073 

Conditional probabilities:
   cancellation_policy
Y      flexible    moderate      strict strict_14_with_grace_period super_strict_30
  f 0.237442922 0.333333333 0.000000000                 0.406392694     0.000000000
  t 0.105849582 0.242339833 0.000000000                 0.649025070     0.000000000
   cancellation_policy
Y   super_strict_60
  f     0.022831050
  t     0.002785515

   beds
Y       [,1]     [,2]
  f 2.118721 1.460353
  t 2.406685 1.517702

   availability_30
Y       [,1]     [,2]
  f 10.70776 9.918312
  t 12.58217 8.624268

   host_response_rate
Y          [,1]         [,2]
  f 0.009538813 0.0013178383
  t 0.009725905 0.0007377852
```

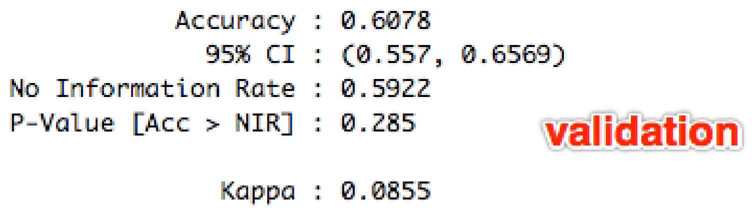
Then, construct confusion matrix for both training data and validation data. The accuracy for training data is 0.6419, and the accuracy for validation data is 0.6078, these two values are close, so the performance for both training data and validation data are close, which means that the model is effective when the new data comes in.



```
pre.train <- predict(naive, newdata =train)
confusionMatrix(pre.train, train$instant_bookable)
pre.valid <- predict(naive, newdata =valid)
confusionMatrix(pre.valid, valid$instant_bookable)
```

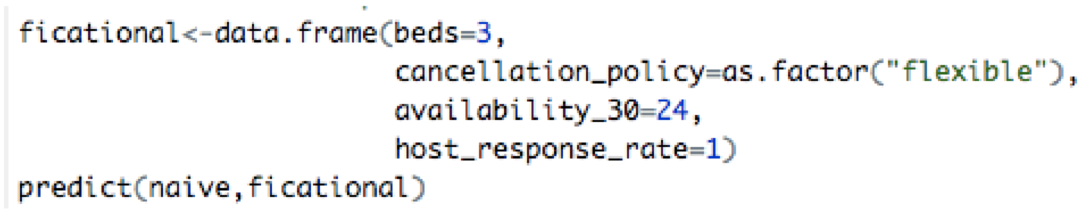


```
              Accuracy : 0.6419
                95% CI : (0.6013, 0.681)
   No Information Rate : 0.6211
   P-Value [Acc > NIR] : 0.1621

                 Kappa : 0.1062
```

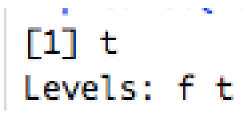


```
              Accuracy : 0.6078
                95% CI : (0.557, 0.6569)
   No Information Rate : 0.5922
   P-Value [Acc > NIR] : 0.285

                 Kappa : 0.0855
```

Finally, create a fictional apartment. Assume there are 3 beds in the apartment, the cancellation policy should be flexible, about 24 days are available in a month, and the host response rate should be 100%. Substitute fictional data into the model, and the result is true. This means that the apartment of what I assume will be instantly bookable.



```
ficational<-data.frame(beds=3,
                       cancellation_policy=as.factor("flexible"),
                       availability_30=24,
                       host_response_rate=1)
predict(naive,ficational)
```

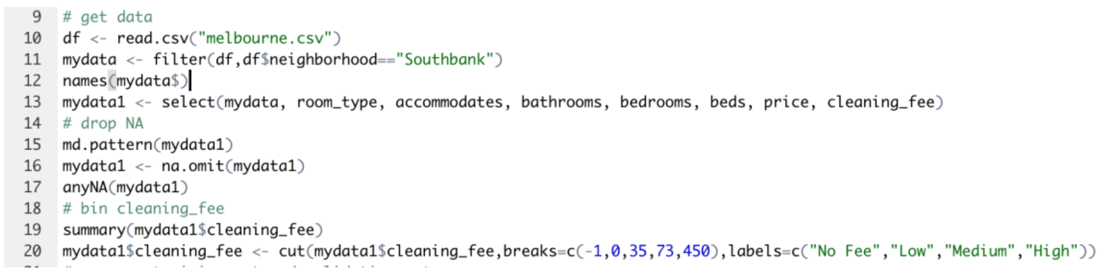


```
[1] t
Levels: f t
```

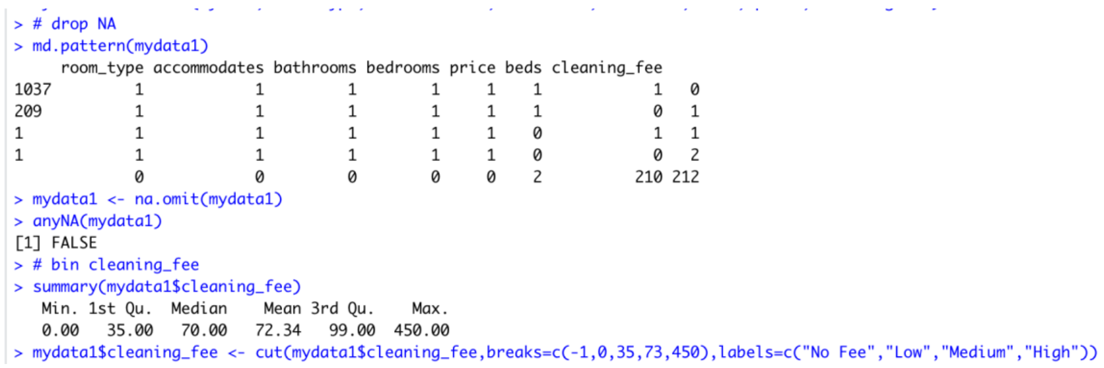
Sum up above model, we understand more about the needs from the customers and what will be the conditions (more available housing, faster host response, somehow flexible, and enough bed for at least one or two people) that the instant bookable type would occur, and it can help Airbnb attracts more customers, as well as assist the host with better options on renting in a fast pace.

3. Classification Tree: Size of the Cleaning Fee

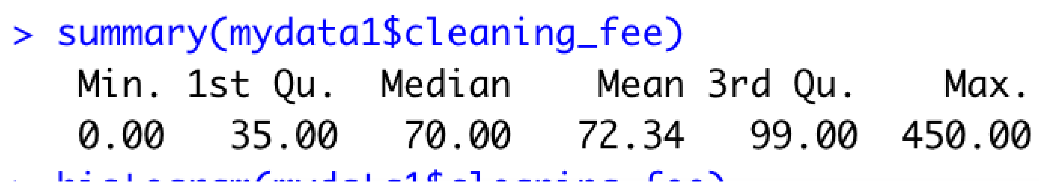
The purpose of this part is to predict the size of the cleaning fee that a particular rental will have through a classification tree. First of all, some specific variables are selected that have an important influence on the cleaning fee. In our team's opinion, the cleaning fee of the private room and the shared room might be higher than that of the entire room or apartment. Also, when there are many accommodate, more stuff is needed to be cleaned, resulting in a higher cleaning fee. Besides, more bathrooms, bedrooms, beds mean that the cleaners are deserved to be paid a higher salary because of more workload. Generally, the higher the house's price is, the higher the cleaning fee is. Secondly, the data selected by us is needed to be processed. The first task is to drop "NA" off because the aim of prediction is to pursue the accuracy and using the original data is most accurate. Based on the summary and histogram of the cleaning fee, the next task is to bin the cleaning fees into four categories: "No Fee" is 0, "Low" is from 1 to 35, "Medium" is from 36 to 73, "High" is from 74 to 450, therefore it is more convenient to predict the size of the cleaning fee for a particular rental. The reason why there are four categories is that we do not ensure the consuming ability of customers, so these four categories could inform customers what level of cleaning fee is among all of the data.



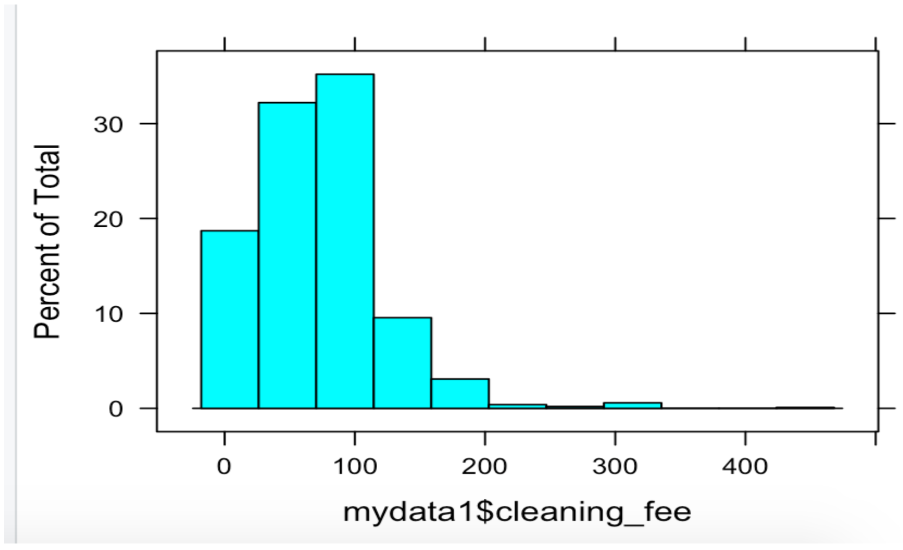
```
# get data
df <- read.csv("melbourne.csv")
mydata <- filter(df,df$neighborhood=="Southbank")
names(mydata$)
mydata1 <- select(mydata, room_type, accommodates, bathrooms, bedrooms, beds, price, cleaning_fee)
# drop NA
md.pattern(mydata1)
mydata1 <- na.omit(mydata1)
anyNA(mydata1)
# bin cleaning_fee
summary(mydata1$cleaning_fee)
mydata1$cleaning_fee <- cut(mydata1$cleaning_fee,breaks=c(-1,0,35,73,450),labels=c("No Fee","Low","Medium","High"))
```

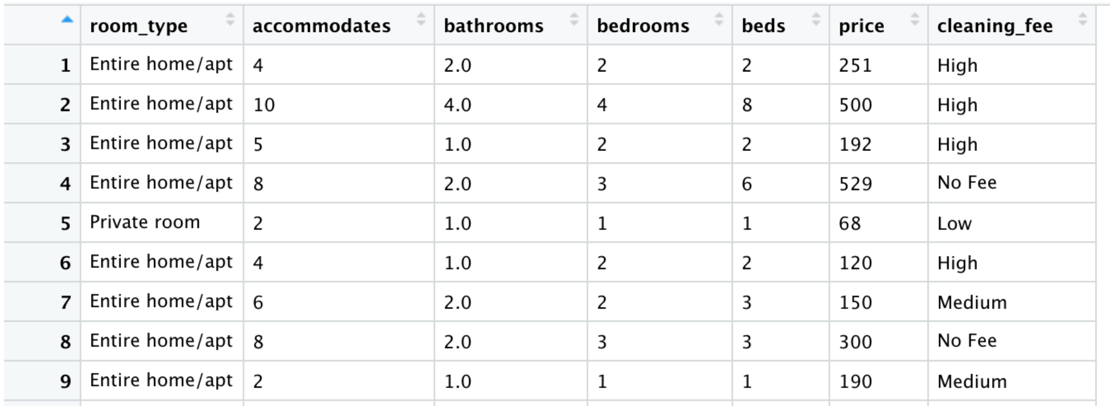


```
> # drop NA
> md.pattern(mydata1)
     room_type accommodates bathrooms bedrooms price beds cleaning_fee
1037         1            1         1        1     1    1            1   0
209          1            1         1        1     1    1            0   1
1            1            1         1        1     1    0            1   1
1            1            1         1        1     1    0            0   2
             0            0         0        0     0    2          210 212
> mydata1 <- na.omit(mydata1)
> anyNA(mydata1)
[1] FALSE
> # bin cleaning_fee
> summary(mydata1$cleaning_fee)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   0.00   35.00   70.00   72.34   99.00  450.00 
> mydata1$cleaning_fee <- cut(mydata1$cleaning_fee,breaks=c(-1,0,35,73,450),labels=c("No Fee","Low","Medium","High"))
```



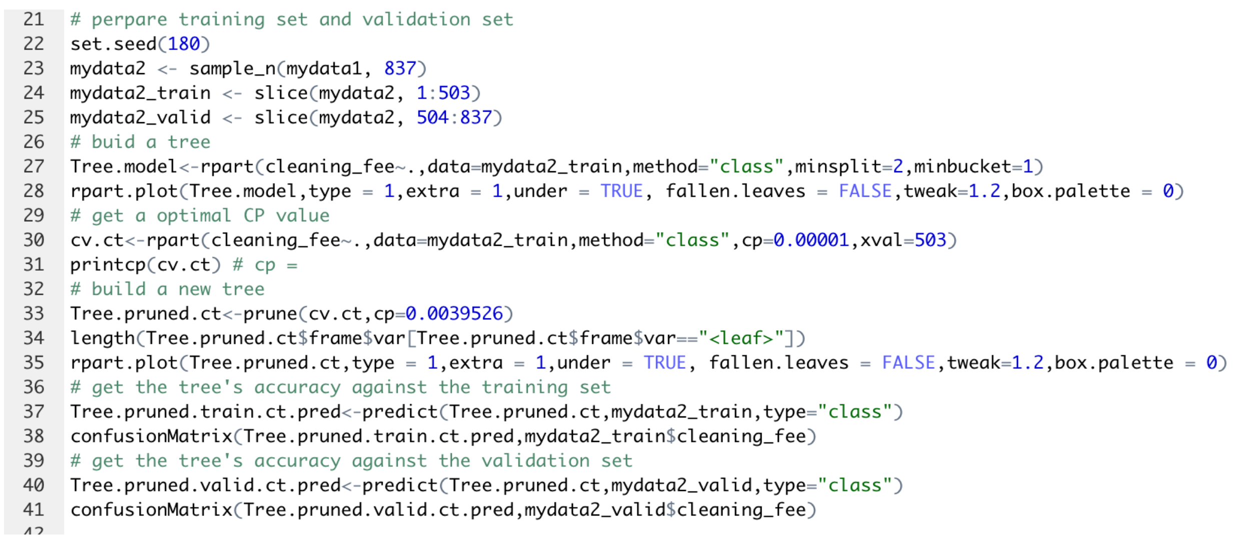
```
> summary(mydata1$cleaning_fee)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   0.00   35.00   70.00   72.34   99.00  450.00 
```



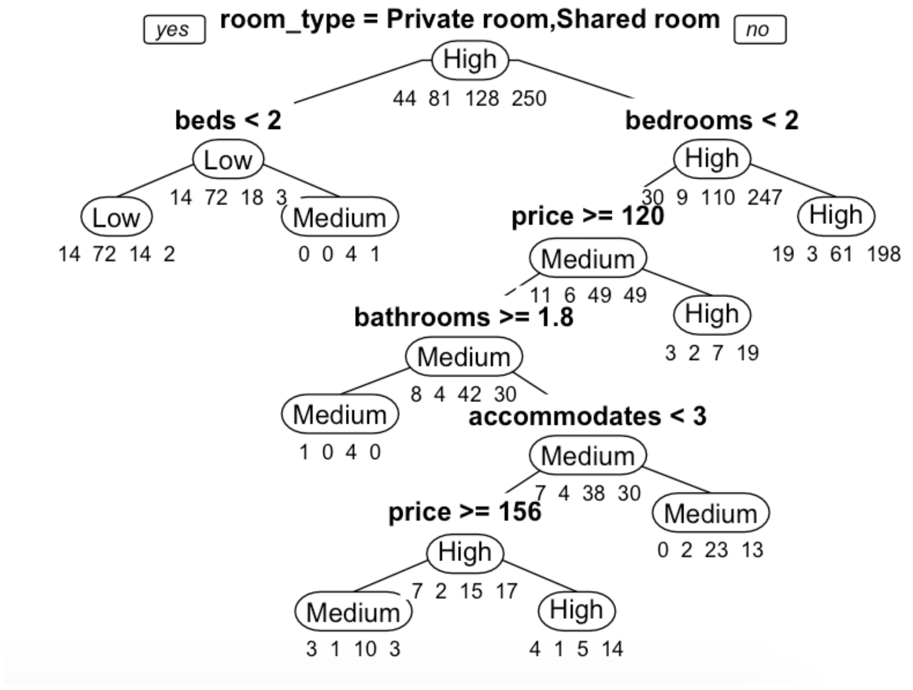


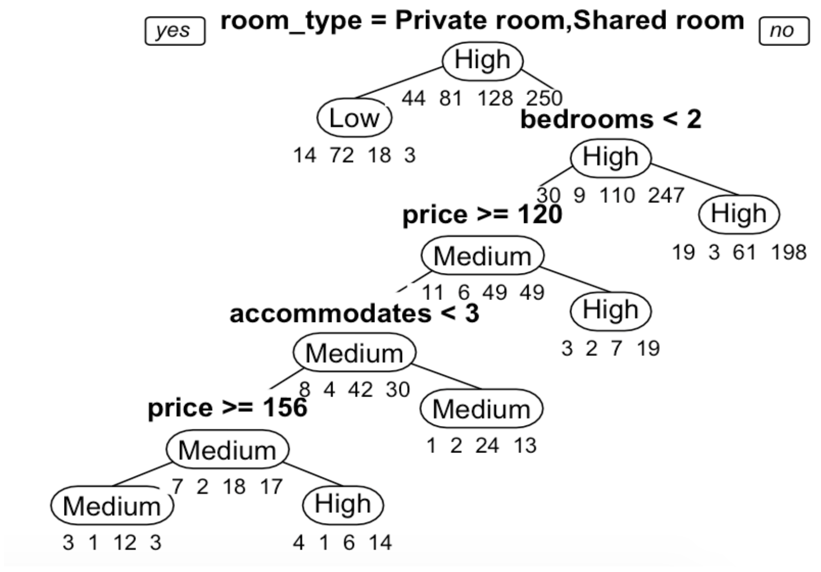
	room_type	accommodates	bathrooms	bedrooms	beds	price	cleaning_fee
1	Entire home/apt	4	2.0	2	2	251	High
2	Entire home/apt	10	4.0	4	8	500	High
3	Entire home/apt	5	1.0	2	2	192	High
4	Entire home/apt	8	2.0	3	6	529	No Fee
5	Private room	2	1.0	1	1	68	Low
6	Entire home/apt	4	1.0	2	2	120	High
7	Entire home/apt	6	2.0	2	3	150	Medium
8	Entire home/apt	8	2.0	3	3	300	No Fee
9	Entire home/apt	2	1.0	1	1	190	Medium

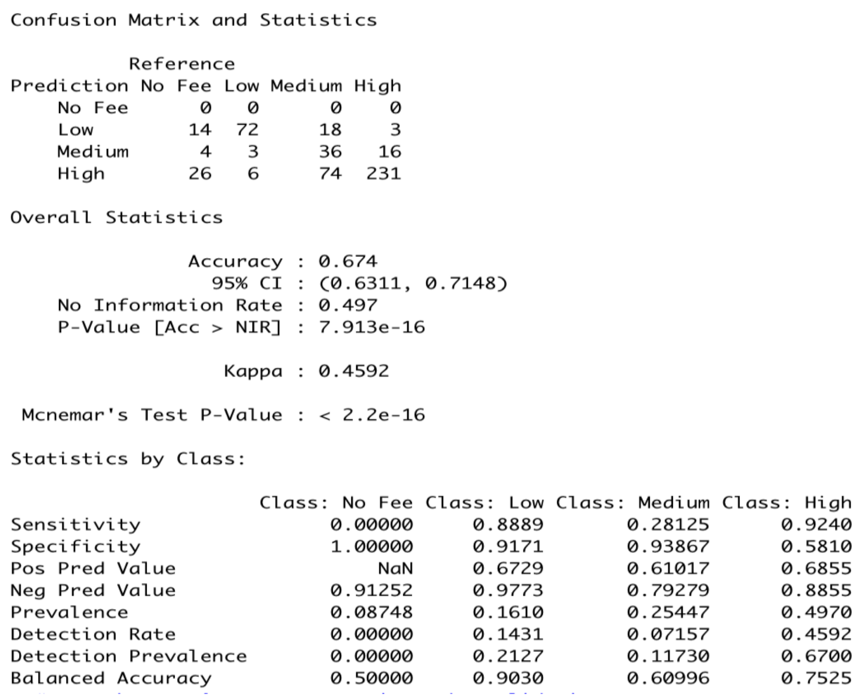
Thirdly, after the data set (837 records) is randomly divided into the training set (503 records) and the validation set (334 records), a classification tree based on the training set is built and its dependent variable is the cleaning fee. The reason why these parameters in building the tree are adopted is that those are making the tree look better. Also, the group of cross-validation is 503 to get the highest accuracy, which matches the number of records in the training set. Even though the first tree built is good because it looks clear, we still desire to get a clearer tree, therefore the optimal tree having only six leaves is obtained after being pruned by using the optimal CP value. As a result, the classification rule is understandable. For instance, the bottom-left-most terminal node indicates that the size of cleaning fee is "Medium" <u>if it is the entire room or apartment and the bedrooms are smaller than 2 and the price is not smaller than 120 and accommodates is smaller than 3 and the price is not smaller than 156.</u> Thus, we can conclude that in what level the cleaning fee is if the room type, the number of bedrooms, the number of accommodates, the price. It is beneficial for customers to select the rental and offer them a reference when considering the cleaning fee. According to the confusion matrix, the accuracy between the training set and the result from the best classification tree based on the training set is about 68%, and the accuracy between the validation set and the result from the best classification tree based on the validation set is about 62%. Thus, these two kinds of accuracy are so high that my tree model is proved to be good. Moreover, the performance against these two sets of data is so similar because their accuracy is very close, which means that the final results are also similar when comparing the results of each rule in the training set and validation set.



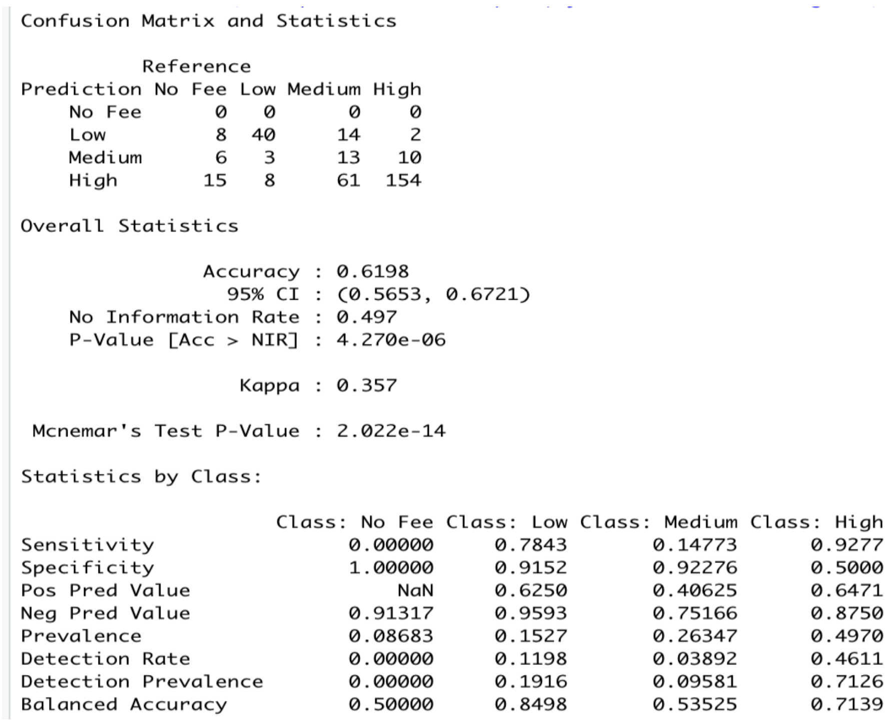
```
# perpare training set and validation set
set.seed(180)
mydata2 <- sample_n(mydata1, 837)
mydata2_train <- slice(mydata2, 1:503)
mydata2_valid <- slice(mydata2, 504:837)
# buid a tree
Tree.model<-rpart(cleaning_fee~.,data=mydata2_train,method="class",minsplit=2,minbucket=1)
rpart.plot(Tree.model,type = 1,extra = 1,under = TRUE, fallen.leaves = FALSE,tweak=1.2,box.palette = 0)
# get a optimal CP value
cv.ct<-rpart(cleaning_fee~.,data=mydata2_train,method="class",cp=0.00001,xval=503)
printcp(cv.ct) # cp =
# build a new tree
Tree.pruned.ct<-prune(cv.ct,cp=0.0039526)
length(Tree.pruned.ct$frame$var[Tree.pruned.ct$frame$var=="<leaf>"])
rpart.plot(Tree.pruned.ct,type = 1,extra = 1,under = TRUE, fallen.leaves = FALSE,tweak=1.2,box.palette = 0)
# get the tree's accuracy against the training set
Tree.pruned.train.ct.pred<-predict(Tree.pruned.ct,mydata2_train,type="class")
confusionMatrix(Tree.pruned.train.ct.pred,mydata2_train$cleaning_fee)
# get the tree's accuracy against the validation set
Tree.pruned.valid.ct.pred<-predict(Tree.pruned.ct,mydata2_valid,type="class")
confusionMatrix(Tree.pruned.valid.ct.pred,mydata2_valid$cleaning_fee)
```




yes
room_type = Private room,Shared room
no
High
44 81 128 250
Low
14 72 18 3
bedrooms < 2
High
30 9 110 247
High
19 3 61 198
price >= 120
Medium
11 6 49 49
High
3 2 7 19
accommodates < 3
Medium
8 4 42 30
Medium
1 2 24 13
price >= 156
Medium
7 2 18 17
Medium
3 1 12 3
High
4 1 6 14



```
Confusion Matrix and Statistics

          Reference
Prediction No Fee Low Medium High
    No Fee      0   0      0    0
    Low        14  72     18    3
    Medium      4   3     36   16
    High       26   6     74  231

Overall Statistics

               Accuracy : 0.674
                 95% CI : (0.6311, 0.7148)
    No Information Rate : 0.497
    P-Value [Acc > NIR] : 7.913e-16

                  Kappa : 0.4592

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: No Fee Class: Low Class: Medium Class: High
Sensitivity                0.00000     0.8889       0.28125      0.9240
Specificity                1.00000     0.9171       0.93867      0.5810
Pos Pred Value                 NaN     0.6729       0.61017      0.6855
Neg Pred Value             0.91252     0.9773       0.79279      0.8855
Prevalence                 0.08748     0.1610       0.25447      0.4970
Detection Rate             0.00000     0.1431       0.07157      0.4592
Detection Prevalence       0.00000     0.2127       0.11730      0.6700
Balanced Accuracy          0.50000     0.9030       0.60996      0.7525
```

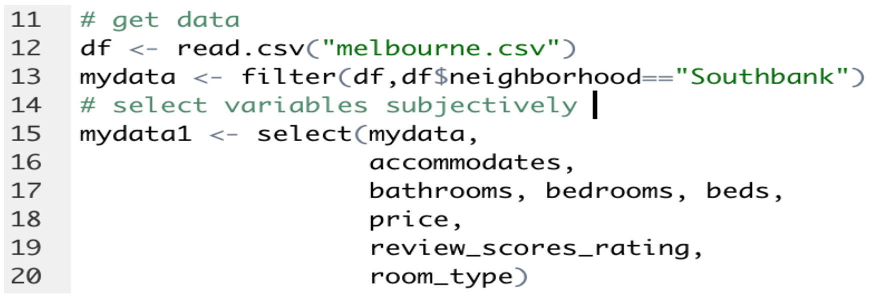


```
Confusion Matrix and Statistics

          Reference
Prediction No Fee Low Medium High
    No Fee      0   0      0    0
    Low         8  40     14    2
    Medium      6   3     13   10
    High       15   8     61  154

Overall Statistics

               Accuracy : 0.6198
                 95% CI : (0.5653, 0.6721)
    No Information Rate : 0.497
    P-Value [Acc > NIR] : 4.270e-06

                  Kappa : 0.357

 Mcnemar's Test P-Value : 2.022e-14

Statistics by Class:

                     Class: No Fee Class: Low Class: Medium Class: High
Sensitivity                0.00000     0.7843       0.14773      0.9277
Specificity                1.00000     0.9152       0.92276      0.5000
Pos Pred Value                 NaN     0.6250       0.40625      0.6471
Neg Pred Value             0.91317     0.9593       0.75166      0.8750
Prevalence                 0.08683     0.1527       0.26347      0.4970
Detection Rate             0.00000     0.1198       0.03892      0.4611
Detection Prevalence       0.00000     0.1916       0.09581      0.7126
Balanced Accuracy          0.50000     0.8498       0.53525      0.7139
```

Step IV: Clustering

The purpose of this part is to make our whole data set into several groups in terms of some variables, and then clearly realize what kind of customer is in every group. Eventually, the two functions can be achieved. Firstly, customers can easily find out what kind of houses they want if they know their demand. Secondly, according to customers' attributes, there are some recommendations being sent to customers if they are not sure what they want.

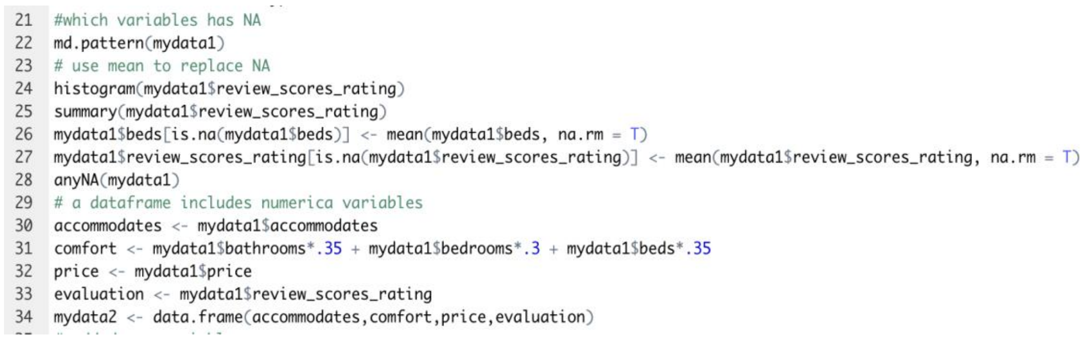
The first step is to select some variables subjectively, which have an essential impact on our results. The reason why these variables are selected is that our team want to cluster the data from five dimensions. Thus, the fives dimensions respectively are "accommodates" which means how many people can live in this house, "comfort" which includes how many bathrooms, bedrooms, beds in the house by giving each one a weight in order to describe comprehensively, "price" which is how much money is needed for living one night, "review_scores_rating" that is the evaluation from the former people, "room_type" which is what type of this house is.



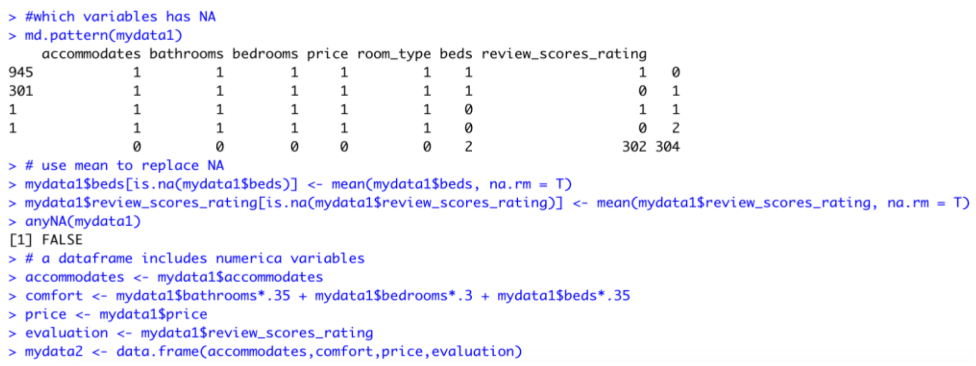
```r
# get data
df <- read.csv("melbourne.csv")
mydata <- filter(df,df$neighborhood=="Southbank")
# select variables subjectively
mydata1 <- select(mydata,
                  accommodates,
                  bathrooms, bedrooms, beds,
                  price,
                  review_scores_rating,
                  room_type)
```

The second step is to process the data.

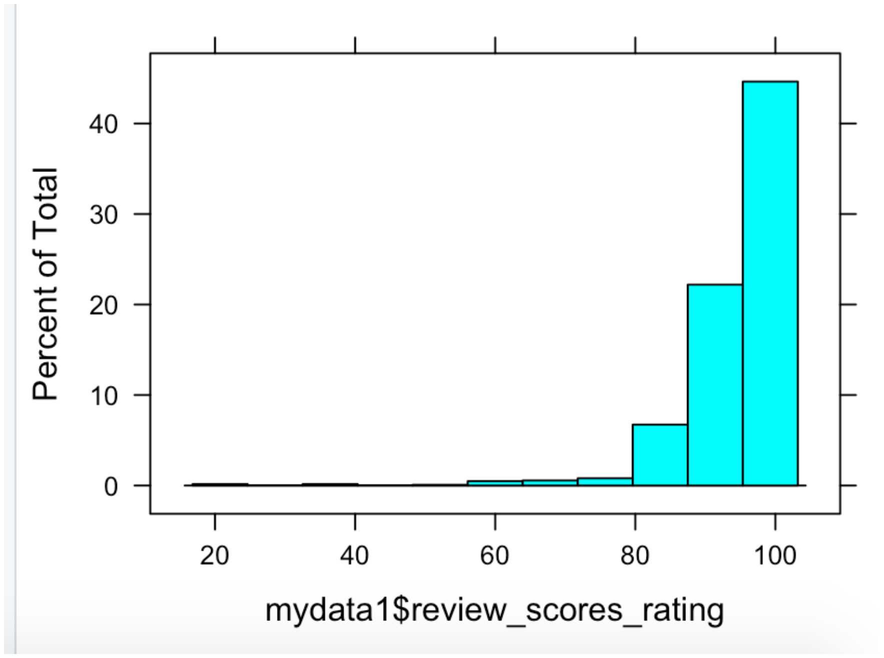
The first task is to process numeric data. Our team has to figure out which variables have "NA" and utilize the mean value to replace "NA" to improve accuracy and avoid losing much data. As a result, there are two "NA" in "beds" and 302 "NA" in "review_scores_rating". After replacing them by their mean value, there is no "NA" in the data set which is represented by "FALSE". The reason why "NA" can be replaced by the mean value is that according to the histogram and summary of the “review_socres_rating”, the most data concentrate on the mean value. Finally, there is a data frame including “accommodates”, “comfort”, “comfort”, “evaluation”.

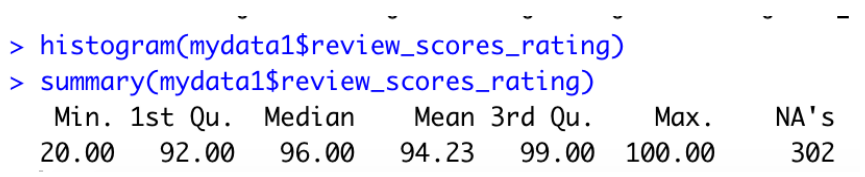


```
#which variables has NA
md.pattern(mydata1)
# use mean to replace NA
histogram(mydata1$review_scores_rating)
summary(mydata1$review_scores_rating)
mydata1$beds[is.na(mydata1$beds)] <- mean(mydata1$beds, na.rm = T)
mydata1$review_scores_rating[is.na(mydata1$review_scores_rating)] <- mean(mydata1$review_scores_rating, na.rm = T)
anyNA(mydata1)
# a dataframe includes numerica variables
accommodates <- mydata1$accommodates
comfort <- mydata1$bathrooms*.35 + mydata1$bedrooms*.3 + mydata1$beds*.35
price <- mydata1$price
evaluation <- mydata1$review_scores_rating
mydata2 <- data.frame(accommodates,comfort,price,evaluation)
```



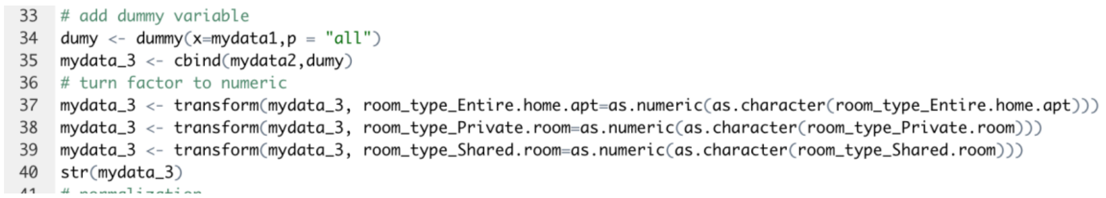
```
> #which variables has NA
> md.pattern(mydata1)
    accommodates bathrooms bedrooms price room_type beds review_scores_rating
945            1         1        1     1         1    1                    1   0
301            1         1        1     1         1    1                    0   1
1              1         1        1     1         1    0                    1   1
1              1         1        1     1         1    0                    0   2
               0         0        0     0         0    2                  302 304
> # use mean to replace NA
> mydata1$beds[is.na(mydata1$beds)] <- mean(mydata1$beds, na.rm = T)
> mydata1$review_scores_rating[is.na(mydata1$review_scores_rating)] <- mean(mydata1$review_scores_rating, na.rm = T)
> anyNA(mydata1)
[1] FALSE
> # a dataframe includes numerica variables
> accommodates <- mydata1$accommodates
> comfort <- mydata1$bathrooms*.35 + mydata1$bedrooms*.3 + mydata1$beds*.35
> price <- mydata1$price
> evaluation <- mydata1$review_scores_rating
> mydata2 <- data.frame(accommodates,comfort,price,evaluation)
```



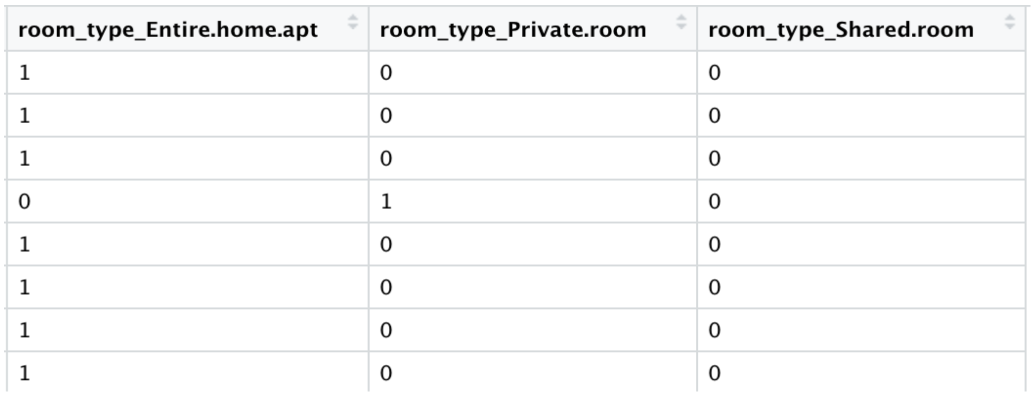


```
> histogram(mydata1$review_scores_rating)
> summary(mydata1$review_scores_rating)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
  20.00   92.00   96.00   94.23   99.00  100.00     302 
```

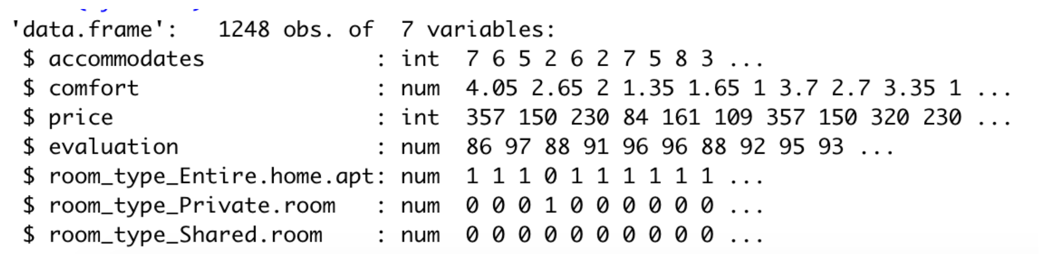
The second task is to process the categorical data. Our team decides to make it in the dummy variable way because we desire to know what kinds of customers like which types of rooms. As a result, the types of rooms are “Entire,home,apt”, “Private.room”, “Shared.room”, and then these three variables are transformed in the class of “numeric” from the class of “factor”. Finally, the final data set is completed.



```
# add dummy variable
dumy <- dummy(x=mydata1,p = "all")
mydata_3 <- cbind(mydata2,dumy)
# turn factor to numeric
mydata_3 <- transform(mydata_3, room_type_Entire.home.apt=as.numeric(as.character(room_type_Entire.home.apt)))
mydata_3 <- transform(mydata_3, room_type_Private.room=as.numeric(as.character(room_type_Private.room)))
mydata_3 <- transform(mydata_3, room_type_Shared.room=as.numeric(as.character(room_type_Shared.room)))
str(mydata_3)
```

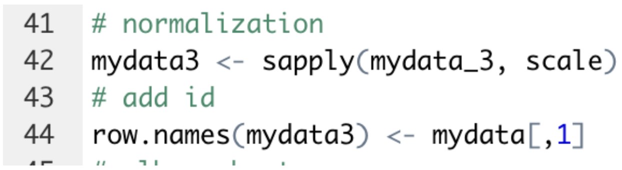


room_type_Entire.home.apt	room_type_Private.room	room_type_Shared.room
1	0	0
1	0	0
1	0	0
0	1	0
1	0	0
1	0	0
1	0	0
1	0	0

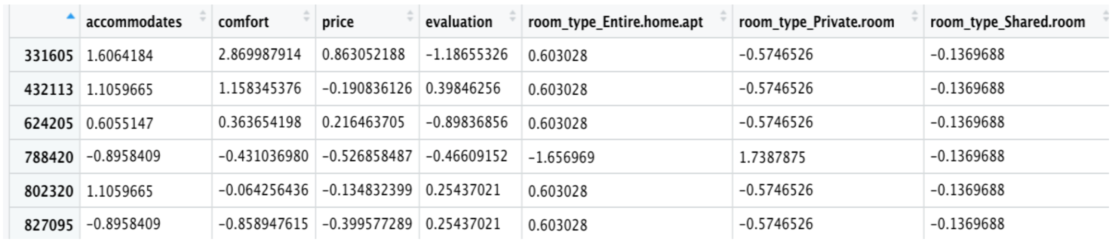


```
'data.frame':	1248 obs. of  7 variables:
 $ accommodates             : int  7 6 5 2 6 2 7 5 8 3 ...
 $ comfort                  : num  4.05 2.65 2 1.35 1.65 1 3.7 2.7 3.35 1 ...
 $ price                    : int  357 150 230 84 161 109 357 150 320 230 ...
 $ evaluation               : num  86 97 88 91 96 96 88 92 95 93 ...
 $ room_type_Entire.home.apt: num  1 1 1 0 1 1 1 1 1 1 ...
 $ room_type_Private.room   : num  0 0 0 1 0 0 0 0 0 0 ...
 $ room_type_Shared.room    : num  0 0 0 0 0 0 0 0 0 0 ...
```

The third task is to normalize all of data in the final data set for making them into a same magnitude and add the original “id” back.

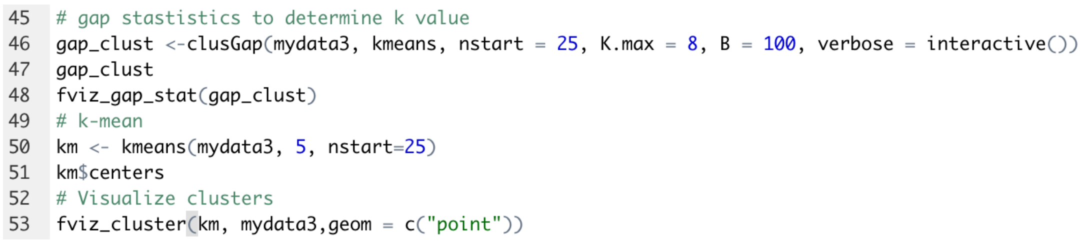


```
# normalization
mydata3 <- sapply(mydata_3, scale)
# add id
row.names(mydata3) <- mydata[,1]
```

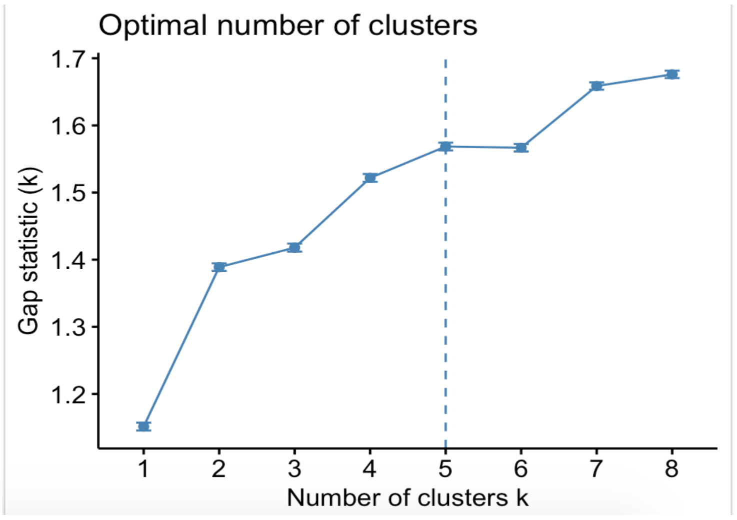


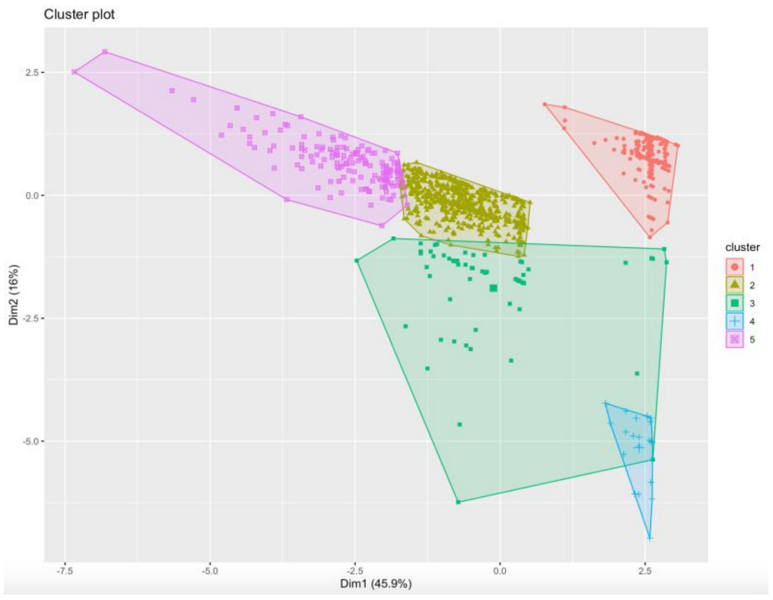
	accommodates	comfort	price	evaluation	room_type_Entire.home.apt	room_type_Private.room	room_type_Shared.room
331605	1.6064184	2.869987914	0.863052188	-1.18655326	0.603028	-0.5746526	-0.1369688
432113	1.1059665	1.158345376	-0.190836126	0.39846256	0.603028	-0.5746526	-0.1369688
624205	0.6055147	0.363654198	0.216463705	-0.89836856	0.603028	-0.5746526	-0.1369688
788420	-0.8958409	-0.431036980	-0.526858487	-0.46609152	-1.656969	1.7387875	-0.1369688
802320	1.1059665	-0.064256436	-0.134832399	0.25437021	0.603028	-0.5746526	-0.1369688
827095	-0.8958409	-0.858947615	-0.399577289	0.25437021	0.603028	-0.5746526	-0.1369688

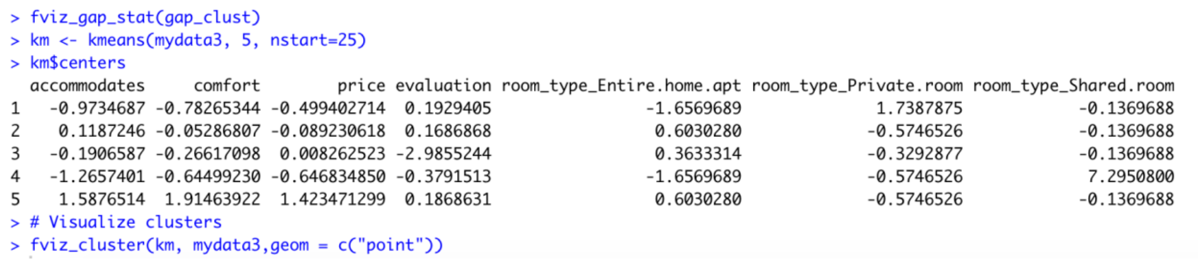
The last step is to determine the k-value through a method of gap statistics and cluster our data by the approach of K-mean. As a result, the optimal k-value is five and these five clustering are visualized.



```
# gap stastistics to determine k value
gap_clust <-clusGap(mydata3, kmeans, nstart = 25, K.max = 8, B = 100, verbose = interactive())
gap_clust
fviz_gap_stat(gap_clust)
# k-mean
km <- kmeans(mydata3, 5, nstart=25)
km$centers
# Visualize clusters
fviz_cluster(km, mydata3,geom = c("point"))
```



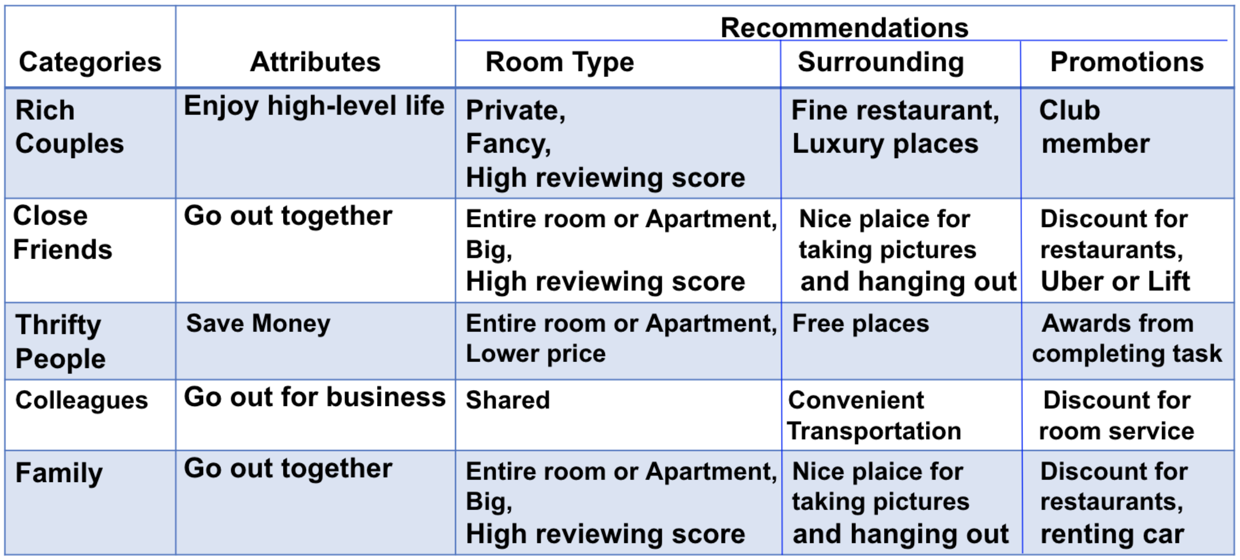




```
> fviz_gap_stat(gap_clust)
> km <- kmeans(mydata3, 5, nstart=25)
> km$centers
  accommodates     comfort        price evaluation room_type_Entire.home.apt room_type_Private.room room_type_Shared.room
1   -0.9734687 -0.78265344 -0.499402714  0.1929405                -1.6569689              1.7387875            -0.1369688
2    0.1187246 -0.05286807 -0.089230618  0.1686868                 0.6030280             -0.5746526            -0.1369688
3   -0.1906587 -0.26617098  0.008262523 -2.9855244                 0.3633314             -0.3292877            -0.1369688
4   -1.2657401 -0.64499230 -0.646834850 -0.3791513                -1.6569689             -0.5746526             7.2950800
5    1.5876514  1.91463922  1.423471299  0.1868631                 0.6030280             -0.5746526            -0.1369688
> # Visualize clusters
> fviz_cluster(km, mydata3,geom = c("point"))
```

Based on the result from the model, the total rental can simply divide into five groups that are suitable for five categories of customers. The analysis and description for the groups of customers is as follow:

	Accommodates	Comfort	Price	Evaluation	Entire room/ Apartment	Private room	Share room
1	Extremely don't care	Extremely don't care	don't care	care	don't care	Extremely care	A little bit doesn't care
2	A little bit care	A little bit doesn't care	A little bit doesn't care	care	Extremely care	don't care	A little bit doesn't care
3	A little bit doesn't care	A little bit doesn't care	A little bit care	Extremely don't care	care	A little bit doesn't care	A little bit doesn't care
4	Extremely don't care	Extremely don't care	Extremely don't care	A little bit doesn't care	Extremely don't care	Extremely don't care	Extremely care
5	Extremely care	Extremely care	Extremely care	care	Extremely care	Extremely don't care	A little bit doesn't care



Categories	Attributes	Recommendations		
		Room Type	Surrounding	Promotions
Rich Couples	Enjoy high-level life	Private, Fancy, High reviewing score	Fine restaurant, Luxury places	Club member
Close Friends	Go out together	Entire room or Apartment, Big, High reviewing score	Nice plaice for taking pictures and hanging out	Discount for restaurants, Uber or Lift
Thrifty People	Save Money	Entire room or Apartment, Lower price	Free places	Awards from completing task
Colleagues	Go out for business	Shared	Convenient Transportation	Discount for room service
Family	Go out together	Entire room or Apartment, Big, High reviewing score	Nice plaice for taking pictures and hanging out	Discount for restaurants, renting car

Step V: Conclusions

Overall data process can be divided into four parts: data preparation and exploration, prediction, classification and clustering. Firstly, we need to deal with raw data, error data, missing data, and redundant data needs to be deleted, but it also depends on the model itself for processing data preparation. There are 84 variables, we need to pick up what we need and then delete useless variables, this can make the data easy to see and accelerate the speed of data processing. After simply processing the data, we use summary statistics and visualization to have a general and directly overview of data. We can learn from data like what kind of property type or bed type is the most popular and then use chart to show the result clearly. This can help Airbnb learn their customer preferences roughly. Secondly, find several variables that will affect price and then build a regression model to predict price. After building the model, we need to ensure that independent variables what we pick up do have relationship with price and our model is accurate so that the result of prediction base on this model can be reliable. prediction can help Airbnb determine price of a particular apartment or other types based on the several known properties.

Then, by analyzing data classification, we can earn more about our data and find more useful information what Airbnb cares about. There are three kinds of classification and all of them can help Airbnb classify or predict a new record by choosing several predictors what we consider is relevant. Knn model simply divides cancellation policy into 12 kinds and predicts what the new record belongs to which kind of cancellation policy without any parametric assumptions. Naïve Bayes model studies on whether the apartment is instantly bookable or not and then assigns the class to the new record. Classification tree model figures out the size of the cleaning fee when giving a particular rental. All of three kinds of classification need to do test to ensure the accuracy of model before predicting. Finally, we do the clustering to segment rentals in our neighborhood into clusters of homogeneous records. There are total 5 clusters, Airbnb can focus on each cluster rather than the entire heterogeneous dataset. We try to figure out the particular properties of each cluster and then infer the demands of these five kinds of customers. Focus on their different demands, provide some recommendations on what kinds of apartment are more suitable.

The whole data process extracts and derives valuable and meaningful data for Airbnb from a large amount of messy and incomprehensible data. It classifies the data into groups, which is much easier to understand than reviewing more than hundreds of data, it also gives a prediction on a specific property of apartment by inputting a set of frictional value of relevant predictors. Every time before Airbnb makes a decision, it can use the model to see whether the apartment they choose can achieve their purpose or what price of this apartment is feasible. Data process can help Airbnb learn more about their guests and give Airbnb a direction on what improvement it can do on the existing apartment, and how the future apartment should be more suitable and attract more guests.

Therefore, based on the above summary, Airbnb, stakeholder, and hosts are all beneficiaries during the data process. Airbnb can be based on the data collection through different categories from classification to offer advice to the host, in order to attract most of the customers, as well as providing satisfying services to the customers. For example, based on the regression model, Airbnb would make its strategy according to the most related predicators from the results, and adjust its policy or housing resources accordingly. Also, for stakeholders, they may want to invest in the Airbnb housing in the Southbank area, while they have no idea what kind of rooms/properties that visitors are mostly interested in. From the analysis, the results are clear: visitors would prefer privacy as most of the hosts are given, so entire home/apt or private room would be really important for the investment. Also, there is a gap between property types, we think investors can also consider on developing new apartments or houses or condo more in the region since this region involves many activities, and it is really suitable for relaxing during short-term and long-term vacations. Overall, there is still a lot of potentials for Airbnb to grow in Southbank, Melbourne.