Final project

Team Rocket

2022-05-10

```
library(tidyverse)
library(dbplyr)
library(readr)
library(textdata)
library(naniar)
library(zoo)
library(simputation)
library(forecast)
library(DBI)
library(odbc)
library(tidyverse)
library(stringr)
library(lubridate)
library(simputation)
library(car)
library(factoextra)
library(leaflet)
library(leaflet.providers)
library(stopwords)
library(gutenbergr)
library(tidytext)
library(wordcloud)
library(e1071)
library(caret)
library(FNN)
library(leaflet.extras)
library(rpart)
library(readr)
library(caTools)
library(party)
library(partykit)
library(rpart.plot)
```

Importing data set

```
paris_listings <- read_csv("paris_listings.csv",  col_types = cols(scrape_id = col_do
uble()))
```

Filtering the data set with our neighbourhood

```
Batignolles <- filter(paris_listings,paris_listings$host_neighbourhood =="Batignolle
s")

### keeping only the useful rows
Batignolles  <- Batignolles[,c(5,6,7,11,13:18,22:28,30:34,36:48,50,
                               51:54,56,57,58,59:74)]
```

Checking NAs

```
a <- miss_var_summary(Batignolles)
```

Dealing with NAs

```
predictive<-Batignolles[,c(47:52)]

#impute with random number 3-4.5
predictive$review_scores_accuracy[is.na(predictive$review_scores_accuracy)]<-runif(su
m(is.na(predictive$review_scores_accuracy)), min = 3, max = 4.5)
predictive$review_scores_cleanliness[is.na(predictive$review_scores_cleanliness)]<-ru
nif(sum(is.na(predictive$review_scores_cleanliness)), min = 3, max = 4.5)
predictive$review_scores_checkin[is.na(predictive$review_scores_checkin)]<-runif(sum
(is.na(predictive$review_scores_checkin)), min = 3, max = 4.5)
predictive$review_scores_communication[is.na(predictive$review_scores_communication)]
<-runif(sum(is.na(predictive$review_scores_communication)), min = 3, max = 4.5)
predictive$review_scores_location[is.na(predictive$review_scores_location)]<-runif(su
m(is.na(predictive$review_scores_location)), min = 3, max = 4.5)
predictive$review_scores_value[is.na(predictive$review_scores_value)]<-runif(sum(is.n
a(predictive$review_scores_value)), min = 3, max = 4.5)
```

Imputing Nas in these columns with random number between 3-4.5 because in our opinion we think that if a person is really statisfied her will fill the form and also the people who are really unsatisfied Will fill the form. We use random numbers between three and 3.5 because we are assuming that people who didn't fill out the survey most probably forgot to fill it out and they Were somewhat satisfied and were not on the extreme edges of the form.

```
# importing the values back to the table
Batignolles[,c(47:52)]<-predictive[,c(1:6)]
```

To deal with review_rating_score column we Checked the Air BNB score rating on their website and created a weight for each review to get the rating score.

```
#weighting calculation
f <- data.frame(accuracy = 0.18/0.75,Cleanliness=0.18/0.75,Check_in=0.08/0.75,
                Communication=0.1/0.75,Location=0.03/0.75,Value=0.18/0.75)
```

We imported the values into the NAs in the review_rating_score column after making that feature engineering column (actual score) in our data set with name of actual score.

```
#weighting and impute for overall rating
Batignolles<-Batignolles %>% mutate(actual_score = f$accuracy*Batignolles$review_scor
es_accuracy
                                    +f$Cleanliness*Batignolles$review_scores_cleanlin
ess
                                    +f$Check_in*Batignolles$review_scores_checkin
                                    +f$Communication*Batignolles$review_scores_commun
ication
                                    +f$Location*Batignolles$review_scores_location
                                    +f$Value*Batignolles$review_scores_value)
Batignolles$review_scores_rating[is.na(Batignolles$review_scores_rating)]<-Batignolle
s$actual_score[is.na(Batignolles$review_scores_rating)]

options(digits=3)
```

we used Impute_lm function for high correlated variables as well to have better prediction

```
### imputing with lm
Batignolles <- Batignolles %>% impute_lm(bedrooms ~ accommodates)
Batignolles$bedrooms <- Batignolles$bedrooms %>% round()
Batignolles <- Batignolles %>% impute_lm(beds ~ accommodates+bedrooms)
Batignolles$beds <- Batignolles$beds %>% round()
```

For non numeric variable with NA in it we just changed the NAs to these names as shown below.

```
### replacing with 0
Batignolles$bathrooms_text[is.na(Batignolles$bathrooms_text)] <- "0 Baths"
Batignolles$description[is.na(Batignolles$description)] <- "No Description"
Batignolles$name[is.na(Batignolles$name)] <- "No name"
Batignolles$host_location[is.na(Batignolles$host_location)] <- "No location"
```

Used mean for other numeric variables which didn't have high correlation. We also though mean would be the best option to use here as it will help keep not skew the data.

```
### replacing with mean
Batignolles$first_review[is.na(Batignolles$first_review)] <- mean(Batignolles$first_r
eview,na.rm=TRUE)
Batignolles$last_review[is.na(Batignolles$last_review)] <- mean(Batignolles$last_revi
ew,na.rm=TRUE)

Batignolles$reviews_per_month <- Batignolles$reviews_per_month %>% as.numeric()
Batignolles$reviews_per_month[is.na(Batignolles$reviews_per_month)] <- mean(Batignoll
es$reviews_per_month,na.rm=TRUE)
```

Summary stats

```
# Summary 1 What does the price range in Batignolles?
summary(Batignolles$price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      15      55      70      88     100     690
```

The price for Airbnb in Batignolles is ranging from 15 to 690 with a median value at 70 and a mean value at 88. which indicates that the choice of accommodation in this area is very rich and varied.

```
# Summary 2 How many rooms are there of each type?
count(Batignolles,room_type)
```

```
## # A tibble: 4 × 2
##   room_type            n
##   <chr>            <int>
## 1 Entire home/apt   1053
## 2 Hotel room           8
## 3 Private room       176
## 4 Shared room          6
```

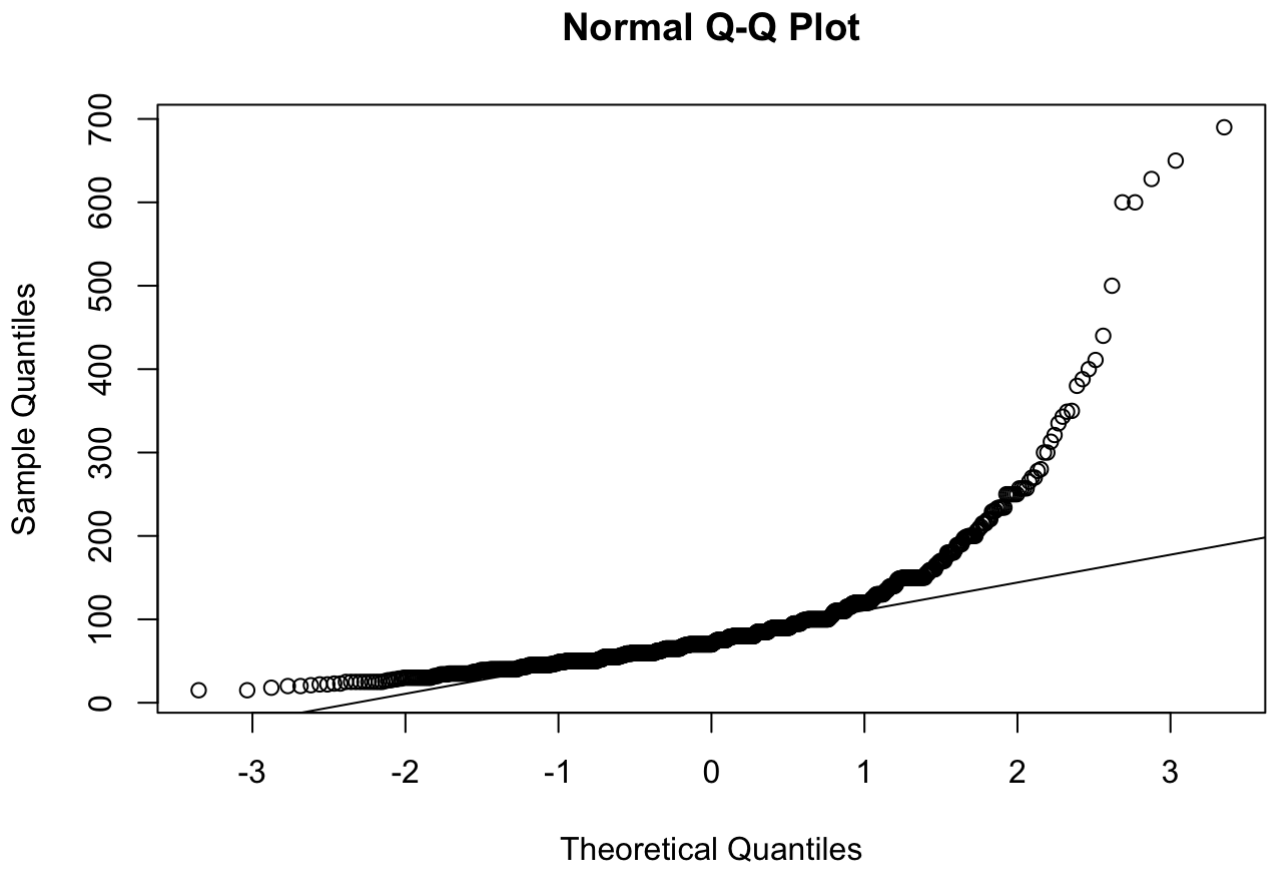
There are 1053 entire home/apt, 8 hotel rooms, 176 private rooms and 6 shared rooms.

```
# Summary 3 Does the price correlated to the number of bedroom?
cor(Batignolles$bedrooms,Batignolles$price)
```

```
## [1] 0.604
```

The correlation coefficient between room price and number of bedrooms is 0.604, indicating that these two variables are moderately correlated.

```
# Summary 4 Does the room price follow a normal distribution?
qqnorm(Batignolles$price)
qqline(Batignolles$price)
```



As can be seen from the above graph, the QQ scatter plot is not evenly distributed on both sides of the line, so the room price does not obey a normal distribution. As can be seen from the above graph, the QQ scatter plot is not evenly distributed on both sides of the line, so the room price does not obey a normal distribution.

```
# Summary 5 How many rooms are there of each type?
kurtosis(Batignolles$price)
```

```
## [1] 26.1
```

```
skewness(Batignolles$price)
```

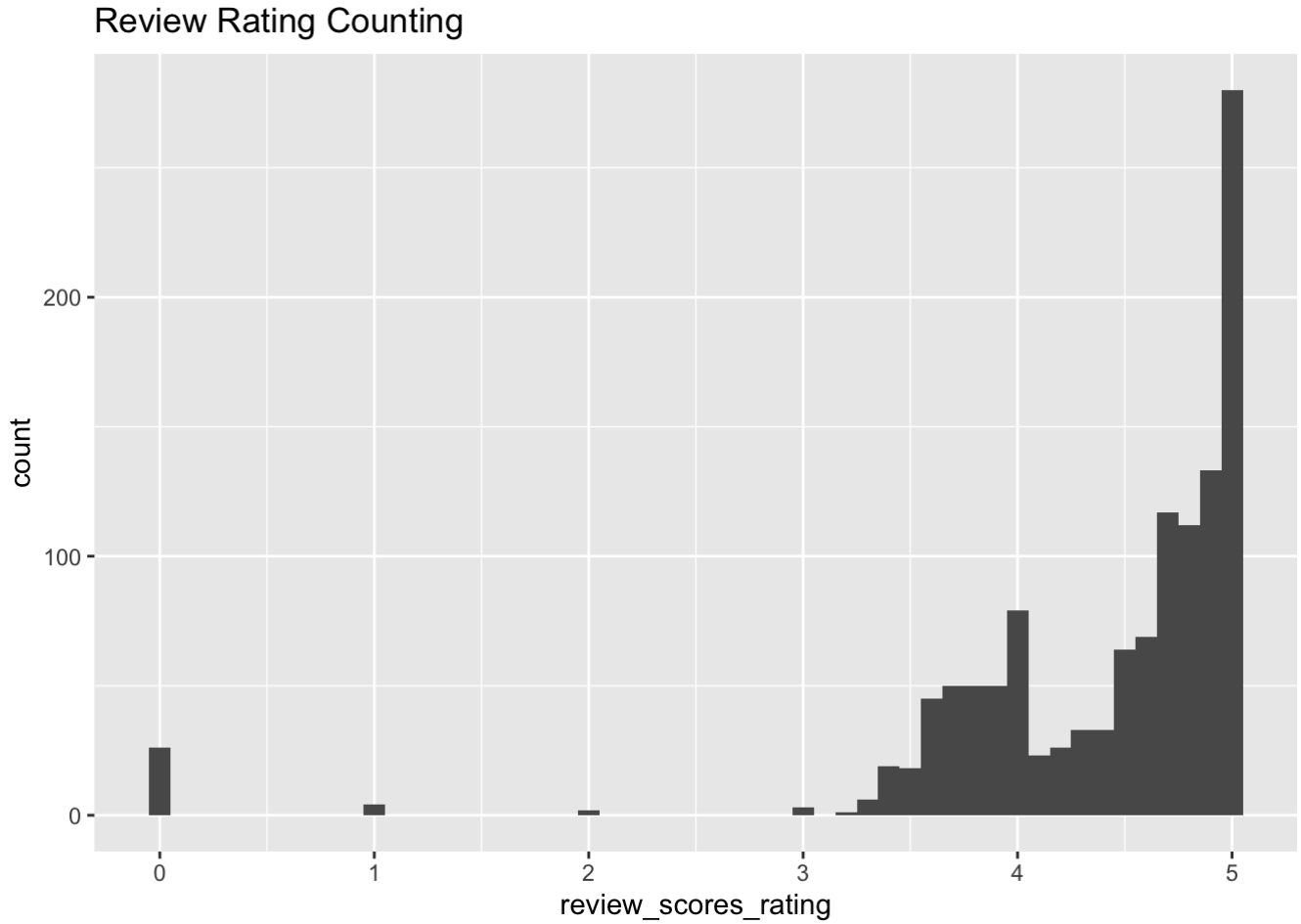
```
## [1] 4.08
```

The kurtosis of the price is 26.1, which is greater than 0, meaning that the distribution of the price is steeper compared to the normal distribution and is cusp. The skewness of the price is 4.08, which is greater than 0, meaning that the data distribution pattern is right skewed compared to the normal distribution, meaning that more of the data will be on the left side of the mean.

Data visualization

Visualization 1: The review rating counting.

```
V1 <- Batignolles[,c(46)]
ggplot(V1,aes(x=review_scores_rating)) +
  geom_histogram(binwidth = 0.1) +
  ggtitle('Review Rating Counting')
```

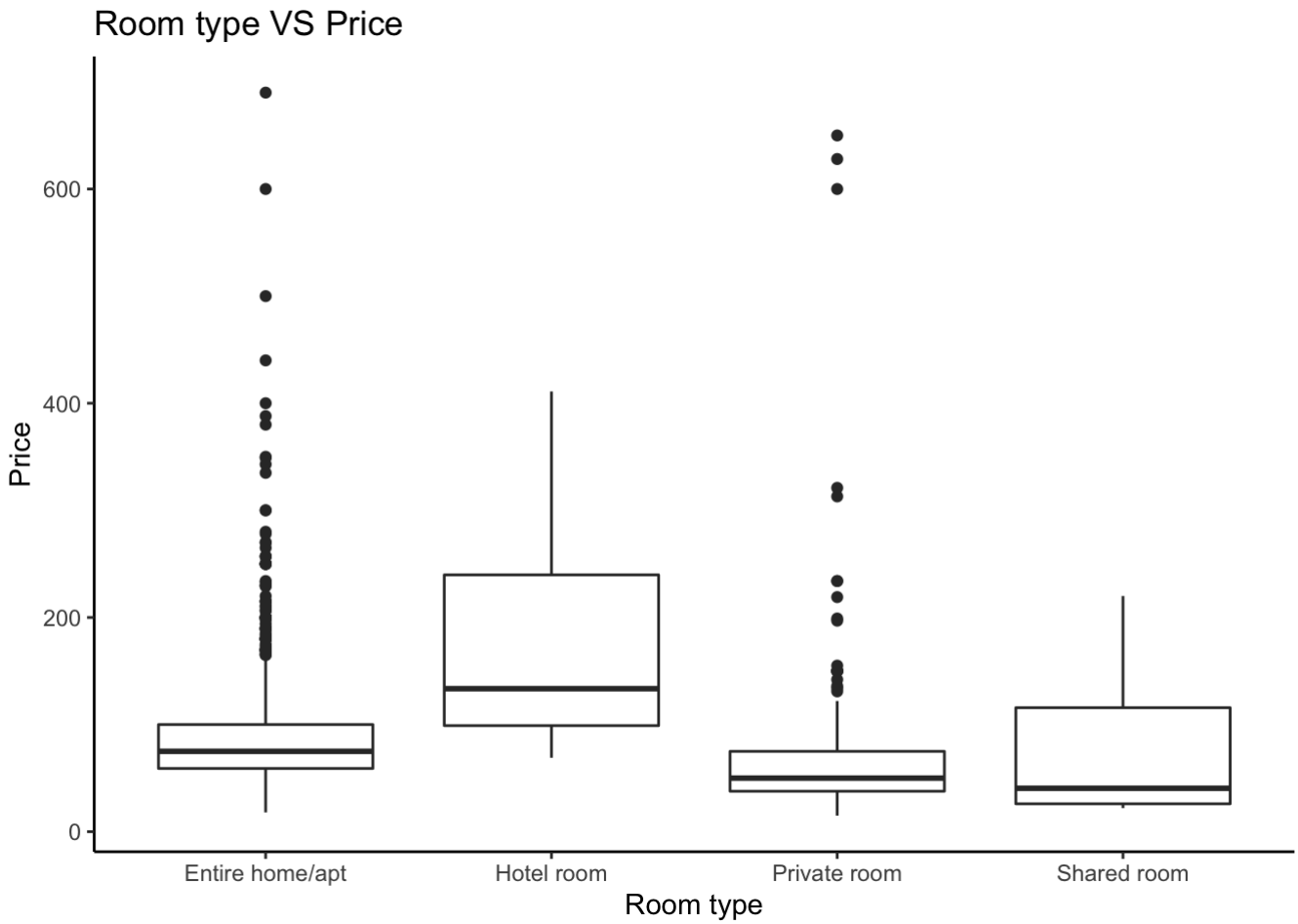


Visualization 1 shows the counting of each review rating. In fact, there are a lot of NAs in the column of 'Review Rating Counting'(actually 280 in total), but since we have already fill in the NA in the data cleaning part, it will be fine to draw the histogram directly.

In the graph we can see that most of the review rating are 5 and the majority are ranging from 3.5 to 5, showing that most of the Airbnb in Batignolles received positive feedback from customers, which indicates that the local living environment is quite good.

Visualization 2: Price for each room type.

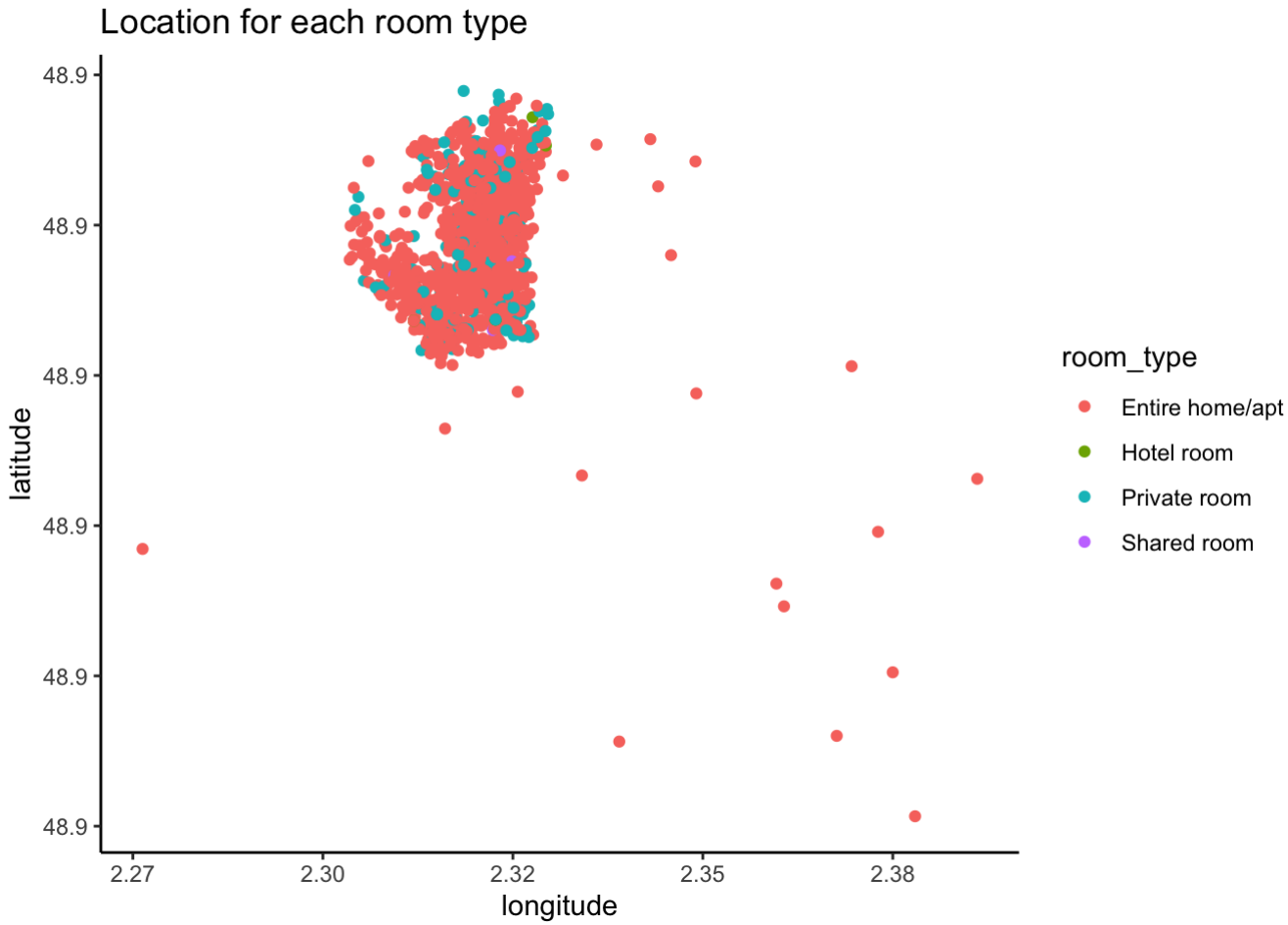
```
V2 <- Batignolles[,c(21,27)]
ggplot(V2,aes(x =room_type, y=price))+
  geom_boxplot() +
  ggtitle('Room type VS Price')+xlab("Room type")+ylab("Price")+
  theme_classic()
```



Visualization 2 shows the boxplot of the price for different room type. We can conclude that hotel room has the highest median price, and the shared room has the lowest median price. Entire room/apt has the largest price fluctuations. There are a lot of outliers in Entire home/apt, which indicates that there may be a lot of choice for this type of room.

Visualization 3: Location for each room type.

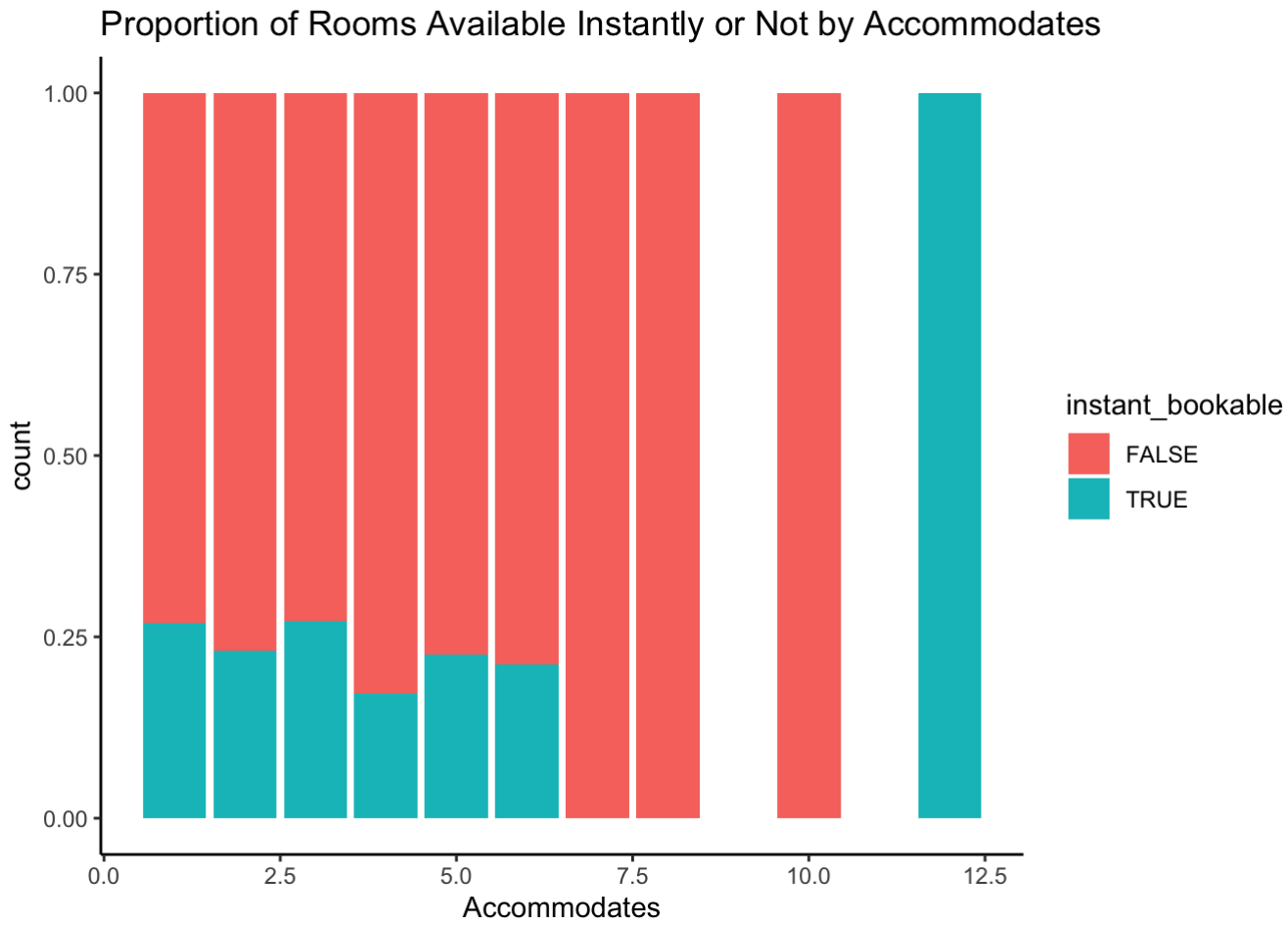
```
V3 <- Batignolles[,c(18,19,21)]
ggplot(V3,aes(x = longitude, y = latitude, color = room_type))+
  geom_point() +
  ggtitle('Location for each room type')+xlab("longitude")+ylab("latitude")+
  theme_classic()
```



Visualization 3 uses the longitude and latitude to show the location for each room type. In this graph we can conclude that most of the Airbnb in Batignolles are entire home/apt.

Visualization 4: Proportion of Rooms Available Instantly or Not by Accommodates.

```
V4 <- Batignolles[,c(22,54)]
ggplot(V4, aes(fill=instant_bookable, x=accommodates)) +
  geom_bar(position="fill") +
  ggtitle('Proportion of Rooms Available Instantly or Not by Accommodates') + xlab("A
ccommodates") +
  theme_classic()
```

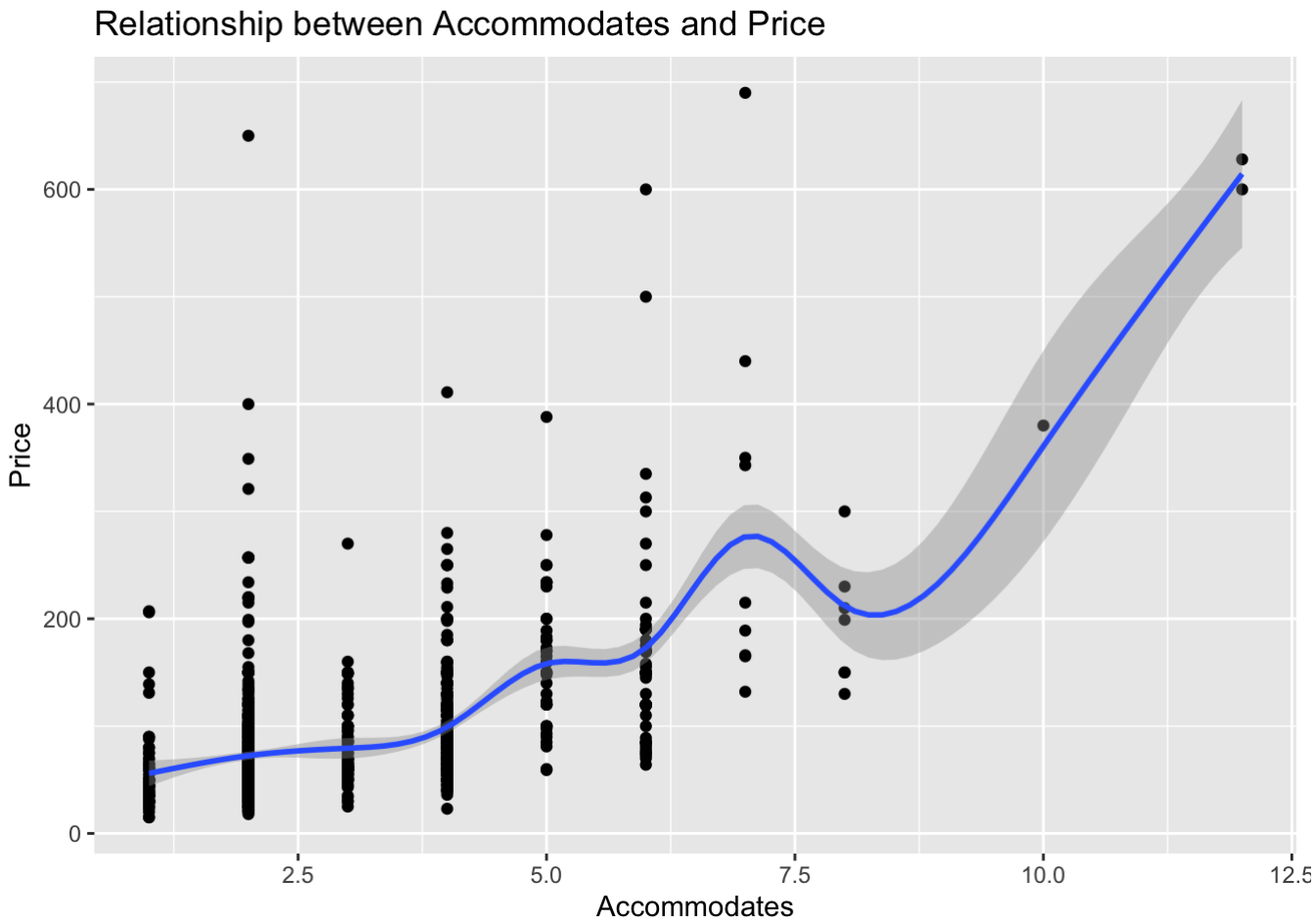


Visualization 4 uses bar plot to show the proportion of whether rooms are available instantly or not by accommodates. In the graph we can conclude that most of the rooms are not instant bookable. For accommodates of 6, 7 and 10, all rooms are not instant bookable. while for accommodates of 11, all rooms are instant bookable.

Visualization 5: Price for Location

```
V5 <- Batignolles[,c(22,27)]
ggplot(V5,aes(x = accommodates, y = price))+
  geom_point() + geom_smooth() +
  ggtitle('Relationship between Accommodates and Price')+xlab("Accommodates")+ylab("P
rice")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

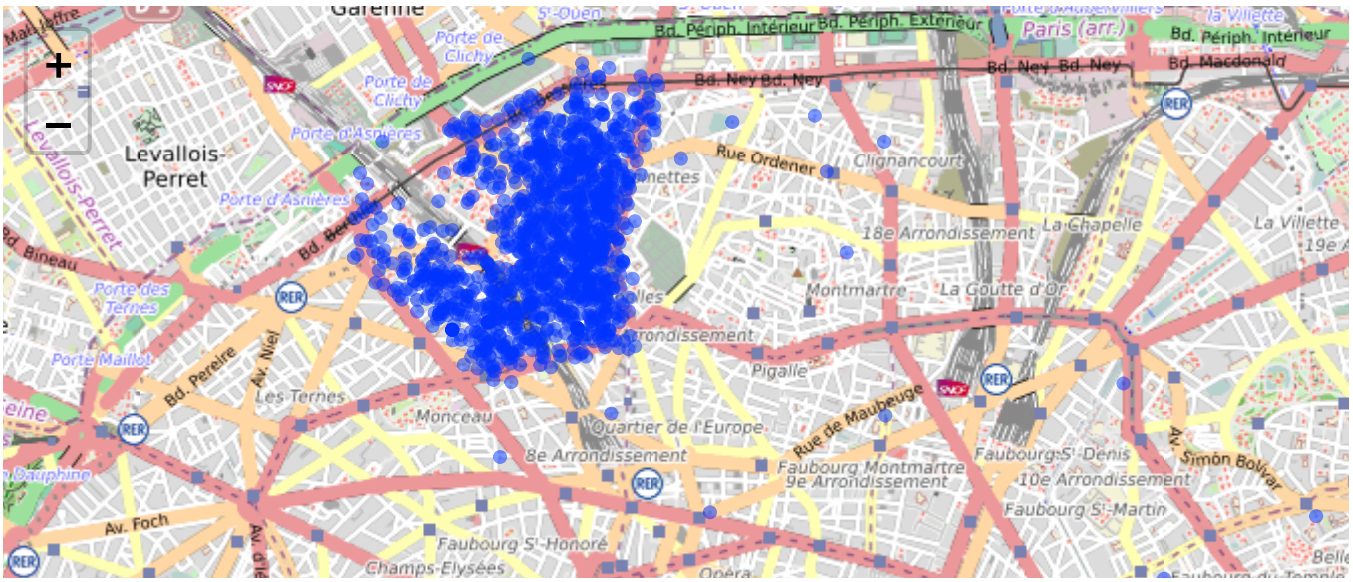


Visualization 5 shows the relationship between accommodates and the price of the room using a smoothing curve. Since the number of samples is greater than 1000, the generalized additive model (gam) is selected by default to plot the smoothing curve.

From the graph, we can very clearly find that the price of the room keeps going up as the number of accommodate rises, especially when the accommodate rises from 6 to 7, the price rises the fastest. However, the price drops when the accommodate is raised from 7 to 8.

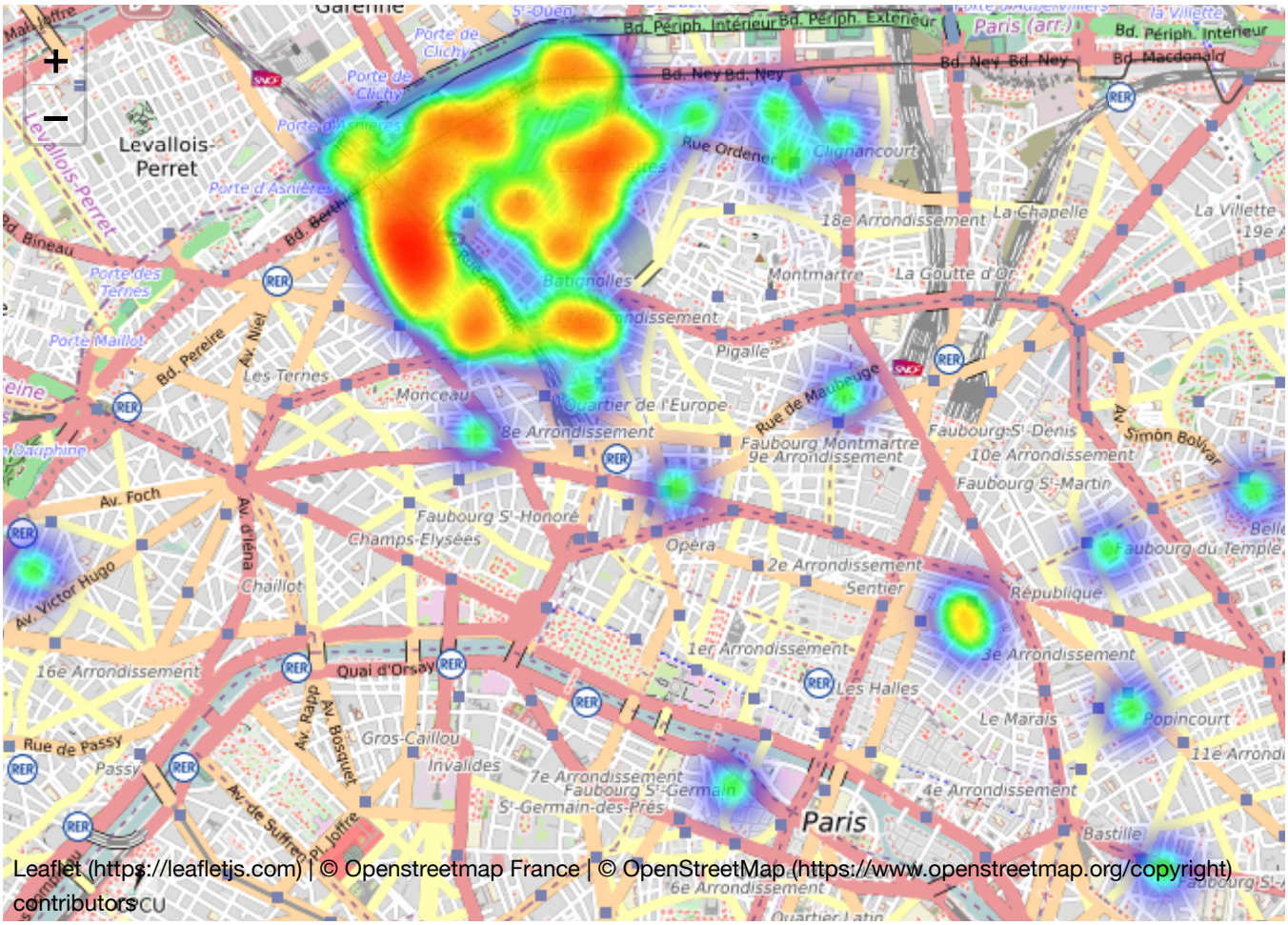
Mapping

```
B <- leaflet() %>% addTiles() %>%  addCircles(lng=Batignolles$longitude,lat=Batignoll
es$latitude) %>% addProviderTiles(providers$OpenStreetMap.France)
B
```



Most of our neighborhoods are all closed together as a group, but some of them are far away from group.

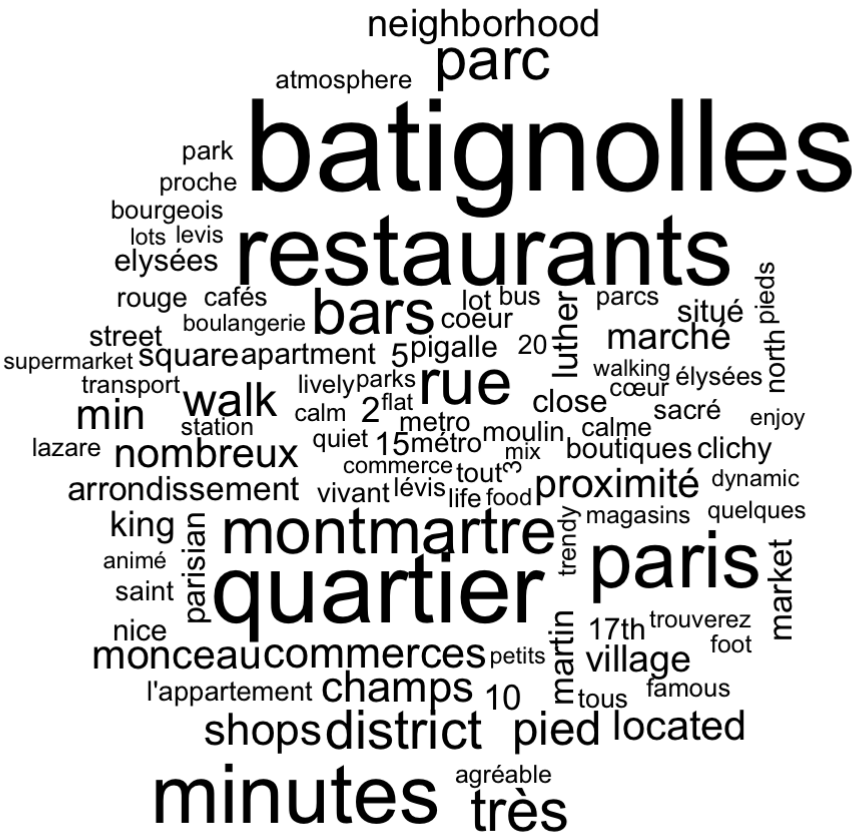
```
#Heatmap
hmap<-leaflet() %>%
  addProviderTiles(providers$OpenStreetMap.France) %>%
  #setView( 178, -20, 5 ) %>%
  addHeatmap(
    lng = Batignolles$longitude, lat = Batignolles$latitude, intensity =Batignolles$h
ost_listings_count,
    blur = 20, max = 0.05, radius = 15
  )
hmap
```



The heatmap is drawn based on the count of host listing, which means the deeper the color is, the more host listings are, and it may indicate that the area has the deeper color in heatmap is more popular than the area has the lighter color.

Wordcloud

```
overview <- Batignolles[,3]

overview <- overview %>%
  unnest_tokens(word, neighborhood_overview)

overview<-subset(overview, word!='br')

stop_french <- data.frame(word = stopwords::stopwords("fr"), stringsAsFactors = FALSE
)

stop_french <- get_stopwords("fr","snowball")

stop_french <- as.data.frame(stop_french)

overview <- overview %>%
  anti_join(stop_words, by = c('word')) %>%
  anti_join(stop_french, by = c("word")) %>% na.omit

countsFR <- overview %>%
  count(word, sort=TRUE)

top.fr <- countsFR[1:20,]

overview %>% count(word) %>% with(wordcloud(word, n, max.words=100))
```



From this wordcloud, we can make out many thing about our neighborhood. In our neighbour hood the most used words is our neighborhood name that is batignolles along side restraunts, paris, quartier, minutes, walks, district, très, montmartre and many more words. This can tell a lot about our neighborhood. After looking at the word clound the first thing i would like to say is that in our neighborhood restraunts are just minutes away and people can walk till there. Bars are also a nice in our neighborhood. There maybe many streets in our neighborhood and also Montmartre is near by our neighborhood as well. We can make many description about our neighborhood from this wordcloud.

Prediction

```
### creating a table for MLR
mlr_tab = subset(Batignolles, select = -c( license, host_about, host_verifications,
                                           amenities,neighborhood_overview, neighbour
hood, name,description, host_name,minimum_nights_avg_ntm,maximum_minimum_nights,minim
um_maximum_nights, maximum_maximum_nights, actual_score,
                                           calculated_host_listings_count,calculated_
host_listings_count_entire_homes,
                                           calculated_host_listings_count_private_roo
ms,calculated_host_listings_count_shared_rooms,
                                           host_response_time, host_response_rate, ho
st_acceptance_rate) )
```

Removing bedroom as it highly correlated with Accommodation and beds, It might create problem in the MLR model

```
mlr_tab$bathrooms_text <- fct_collapse(mlr_tab$bathrooms_text,
                                       '0 Baths' = c("0 baths", "0 Bath") ,
                                       "1 Baths" = c( '1 bath', '1 private bath',  '1
shared bath'),
                                       '1.5 Baths' = c( '1.5 baths', '1.5 shared bath
s'),
                                       '2 Baths' = c( '2 baths' ,  '2 shared baths'),
                                       '6 and 8 Bath' = c( '6 baths' ,  '8 baths')
                                       )
```

```
## Warning: Unknown levels in `f`: 0 Bath
```

```
mlr_tab$bedrooms <- NULL
```

Dividing into training set and validation set

```
mlr_tab1 <- mlr_tab
mlr_tab1$price <- log(mlr_tab1$price)

#seed value
set.seed(699)

train <- sample_frac(mlr_tab, 0.6)
valid <- setdiff(mlr_tab,train)
```

Multi-linear Regression

```
# creating it with every variable and then doing the backward elimanation
mlr <- lm( price ~ . , data = train)

# backward elimination
mlr.step <- step(mlr, direction = "backward")
```

checking the model

```
summary(mlr.step)
```

```
##
## Call:
## lm(formula = price ~ latitude + property_type + accommodates +
##     bathrooms_text + beds + minimum_nights + has_availability +
##     availability_60 + availability_365 + review_scores_rating +
##     review_scores_checkin + reviews_per_month, data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -176.5  -20.1   -4.8   14.7  573.2
##
## Coefficients:
##                                              Estimate Std. Error t value
## (Intercept)                                  5.83e+04   1.60e+04    3.64
## latitude                                    -1.19e+03   3.28e+02   -3.63
## property_typeEntire loft                     3.67e+01   2.17e+01    1.69
## property_typeEntire rental unit             -3.59e+01   1.01e+01   -3.56
## property_typeEntire residential home         1.45e+01   4.41e+01    0.33
## property_typeEntire townhouse               -9.67e+01   3.22e+01   -3.01
## property_typePrivate room in bed and breakfast -2.96e+01   2.40e+01   -1.24
## property_typePrivate room in loft           -6.11e+01   4.41e+01   -1.39
## property_typePrivate room in rental unit    -6.24e+01   1.10e+01   -5.67
## property_typePrivate room in residential home  1.77e+01   4.41e+01    0.40
## property_typeRoom in bed and breakfast      -4.86e+00   5.34e+01   -0.09
## property_typeRoom in boutique hotel         -4.72e+00   1.85e+01   -0.25
## property_typeRoom in hotel                  -2.14e+01   2.38e+01   -0.90
## property_typeShared room in rental unit     -4.27e+01   2.36e+01   -1.81
## accommodates                                 8.01e+00   2.05e+00    3.90
## bathrooms_text1 Baths                       -3.55e+01   3.01e+01   -1.18
## bathrooms_text1.5 Baths                     -3.57e+01   3.08e+01   -1.16
## bathrooms_text2 Baths                        2.14e+01   3.17e+01    0.68
## bathrooms_text2.5 baths                      2.09e+02   4.35e+01    4.80
## bathrooms_text3 baths                        7.35e+01   5.69e+01    1.29
## bathrooms_text4.5 baths                     -6.04e+01   5.21e+01   -1.16
## bathrooms_text6 and 8 Bath                   2.98e+02   5.86e+01    5.08
## bathrooms_textHalf-bath                     -5.71e+01   3.56e+01   -1.61
## beds                                         1.45e+01   2.97e+00    4.89
## minimum_nights                              -6.89e-02   1.03e-02   -6.67
## has_availabilityTRUE                        -2.54e+01   1.01e+01   -2.52
## availability_60                              2.94e-01   1.61e-01    1.82
## availability_365                             7.37e-02   2.60e-02    2.84
## review_scores_rating                         4.15e+00   2.31e+00    1.80
## review_scores_checkin                       -1.44e+01   3.69e+00   -3.91
## reviews_per_month                           -7.25e+00   2.28e+00   -3.18
##                                             Pr(>|t|)
## (Intercept)                                  0.00029 ***
## latitude                                     0.00031 ***
## property_typeEntire loft                     0.09084 .
## property_typeEntire rental unit              0.00039 ***
## property_typeEntire residential home         0.74309
## property_typeEntire townhouse                0.00273 **
## property_typePrivate room in bed and breakfast  0.21701
## property_typePrivate room in loft            0.16587
## property_typePrivate room in rental unit     2.1e-08 ***
## property_typePrivate room in residential home   0.68792
```

```
## property_typeRoom in bed and breakfast          0.92758
## property_typeRoom in boutique hotel             0.79939
## property_typeRoom in hotel                      0.36900
## property_typeShared room in rental unit         0.07132 .
## accommodates                                    0.00011 ***
## bathrooms_text1 Baths                           0.23869
## bathrooms_text1.5 Baths                         0.24698
## bathrooms_text2 Baths                           0.49979
## bathrooms_text2.5 baths                         1.9e-06 ***
## bathrooms_text3 baths                           0.19691
## bathrooms_text4.5 baths                         0.24630
## bathrooms_text6 and 8 Bath                      4.8e-07 ***
## bathrooms_textHalf-bath                         0.10867
## beds                                            1.3e-06 ***
## minimum_nights                                  5.2e-11 ***
## has_availabilityTRUE                            0.01182 *
## availability_60                                 0.06849 .
## availability_365                                0.00467 **
## review_scores_rating                            0.07234 .
## review_scores_checkin                           0.00010 ***
## reviews_per_month                               0.00153 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 42.4 on 715 degrees of freedom
## Multiple R-squared:  0.564,  Adjusted R-squared:  0.546
## F-statistic: 30.9 on 30 and 715 DF,  p-value: <2e-16
```

Before making the the final regression model, I would like to describe what we did in above code. First we created an another data frame just the exact copy of our original data set, we did this because it will help us prevent form doing anything in our original data set as that data set in being used in different models as well. To creating this table we only used variable useful for our model and removed those which would create a problem in our model. To create a Multi-linear regression model we used the data set called mlr_tab, in which we had all the useful variable for our analysis. As we have dealt with NA's already we don't have to do it again in out data set. In our data frame we also collapsed the factors of bathroom text variable to make it more use full. Then we checked the correlation between the numerical variables. IN this we found out that some had high correlation but we only removed the Bedroom variable as it was highly correlated with beds and accommodations.

After building a proper data frame we then used that data frame for our model. In the first step of the model we split the data into 60-40 ratio for 60 being train set and 0 being the validation set. This will help us check our model's performance. We used every variable in our data set as an input and price as an output variables. Then we used backward emanation on that regression model to get the most relevant independent variables. After doing that we still we getting some not that relevant/significant variables in our model hence we then judged to make the model our-self with the help of the mode we create with backward elimination. In this final model we will be using only the most significant numerical variables, as the categorical variables has so many categories in them that it was hard to split the data equally between training set and validation set. It will create sampling bias. Hence, we will only use the high significant numerical variable for our final MLR model.

making a mlr after our doing the backward elimination

```
final.mlr <- lm( price ~ beds + minimum_nights + accommodates +latitude  , data = tra
in) # or availability_365

summary(final.mlr)
```

```
##
## Call:
## lm(formula = price ~ beds + minimum_nights + accommodates + latitude,
##     data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -107.4  -24.4   -6.1   16.5  562.9
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     8.12e+04   1.84e+04    4.42  1.1e-05 ***
## beds            1.94e+01   3.34e+00    5.82  8.9e-09 ***
## minimum_nights -6.92e-02   1.02e-02   -6.77  2.6e-11 ***
## accommodates    1.48e+01   2.27e+00    6.51  1.4e-10 ***
## latitude       -1.66e+03   3.76e+02   -4.42  1.1e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 50.1 on 741 degrees of freedom
## Multiple R-squared:  0.369,  Adjusted R-squared:  0.366
## F-statistic:  108 on 4 and 741 DF,  p-value: <2e-16
```

This regression model tells us that these independentant variables have a high significance on on the out come variable that is price. In this regression model we got an equation that would look something like :

y = b0 + b1x1 + b2x2 + b3x3 + B4X4 + c

We can use this equation If we want to predict this value ourself. we can use this equation in which Y is the dependent variable that we are trying to find and b0 is the intercept. b1 is the value we provide with the x1 being the coefficient of that independent variable in model. This can go until n number of variables but as we only have 4 independent variable we will use this till 4 with c being the error which we assume is taking into consideration, hence we don't use it while calculating the dependent variable value MANUALLY. In the Equation we can see that with a single unit increase / decrease in any of the independent variable have an effect on the outcome variable that would be equal to their co-efficient.

Evaluating the model

First we look at the F value. The F value in regression is the result of a test where the null hypothesis is that all of the regression coefficients are equal to zero. In other words, the model has no predictive capability. Basically, the f-test compares the model with zero predictor variables (the intercept only model), and decides whether our added coefficients improved the model. we got a significant result of less than alpha value, then whatever coefficients we included in your model improved the model's fit. Hence,After we made this final MLR model we saw that the f statistic went up but the p-value of the f stats we still below the alpha value, so we can reject the null hypothesis. This means that our added coefficients improved the model.

We can also see that all the other independent variable are very significant for the prediction in our model.

R-squared (R2) is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model. R-squared explains to what extent the variance of one variable explains the variance of the second variable. While R-squared can return a figure that indicates a level of correlation with an index, it has certain limitations when it comes to measuring the impact of independent variables on the correlation. This is where adjusted R-squared is useful in measuring correlation. The adjusted R-squared compensates for the addition of variables and only increases if the new predictor enhances the model above what would be obtained by probability. Conversely, it will decrease when a predictor improves the model less than what is predicted by chance. In our case the

Adjusted R-square has decrease. We should expect to see that because we are not using the categorical variable in our regression model and it will decrease the adjusted R-square as we are removing some significant variable from our model.

```
vif(final.mlr)
```

```
##           beds minimum_nights   accommodates       latitude
##           2.77           1.03           2.79           1.00
```

Variance inflation factor measures how much the behavior (variance) of an independent variable is influenced, or inflated, by its interaction/correlation with the other independent variables. Variance inflation factors allow a quick measure of how much a variable is contributing to the standard error in the regression. In our situtation the VIF is lower than 4 hence there is no high correlation between the independent variables.

checking the accuracy

```
# training set
predc<- predict(final.mlr,train)
accuracy(predc, train$price)
```

```
##                  ME RMSE  MAE   MPE MAPE
## Test set -6.66e-11 49.9 30.1 -18.4 39.2
```

```
# validation set
predc1<- predict(final.mlr,valid)
accuracy(predc1, valid$price)
```

```
##               ME RMSE MAE   MPE MAPE
## Test set -0.919 50.1  31 -18.5 38.2
```

The RMSE is the square root of the variance of the residuals. It indicates the absolute fit of the model to the data–how close the observed data points are to the model's predicted values. Whereas R-squared is a relative measure of fit, RMSE is an absolute measure of fit. As the square root of a variance, RMSE can be interpreted as the standard deviation of the unexplained variance, and has the useful property of being in the same units as the response variable. Lower values of RMSE indicate better fit. RMSE is a good measure of how accurately the model predicts the response, and it is the most important criterion for fit if the main purpose of the model is prediction.

The mean percentage error (MPE) is the computed average of percentage errors by which forecasts of a model differ from actual values of the quantity being forecast. In our case both the RMSE and MPE is not that different from both training and validstion set, that means the model is a good fit for prediction of the variable price. Also we can see that the error seems to be high as it in the range of 50 that could be explained by the range of out outcome variable. The price has a range of 15 - 690. That is pretty huge range. so, we can say that our normalized RMSE would be less and hence our model is able to predict but not too properly.

KNN

partioning the data

```
set.seed(699)

knnn <- as.data.frame(Batignolles[,c(11,12,22,24,25,26,27,28,29,60)])

trainknn<-sample_frac(knnn,0.6)
validknn<-setdiff(knnn,trainknn)
```

```
trainknn_new <- trainknn %>% mutate(has_kitchen= ifelse(grepl("Kitchen|Hot water",ame
nities), '1', '0'))
trainknn_new$has_kitchen <- as.factor(trainknn_new$has_kitchen)
validknn_new <- trainknn %>% mutate(has_kitchen= ifelse(grepl("Kitchen|Hot water",ame
nities), '1', '0'))
validknn_new$has_kitchen <- as.factor(validknn_new$has_kitchen)

train1 <- trainknn_new
```

checking the mean difference of the variable

```
like <- filter(train1, has_kitchen == "1")

not_liked <- filter(train1, has_kitchen == "0")
```

```
have <- filter(train1, has_kitchen == "1")

not_have <- filter(train1, has_kitchen == "0")

have_it<- have[c(1:5,7:10)]
not_have_it<- not_have[c(1:5,7:10)]

difference<- ((colMeans(have_it)-colMeans(not_have_it))/colMeans(have_it))*100

difference
```

```
##       host_listings_count host_total_listings_count              accommodates
##                   -40.377                   -40.377                    -0.616
##                  bedrooms                      beds                     price
##                    -8.045                   -12.466                   -29.352
##            minimum_nights            maximum_nights              actual_score
##                     4.677                    -3.335                     2.513
```

removing the less than 5% difference variance because it will not help model predict the classification of the new data set

```
trainknn_new <- subset(trainknn_new, select = -c(accommodates , maximum_nights,  actu
al_score,  minimum_nights ))
validknn_new <- subset(validknn_new, select = -c(accommodates , maximum_nights,  actu
al_score,  minimum_nights ))
knnn <- subset(knnn, select = -c(accommodates , maximum_nights,  actual_score,  minim
um_nights ))
```

```
trainknn_new.df <- trainknn_new
validknn_new.df <- validknn_new
knnn.df <- knnn

norm.values <- preProcess(trainknn_new[,c(1:4,6)], Method = c("center", "scale"))
trainknn_new.df[,c(1:4,6)] <- predict(norm.values, trainknn_new[,c(1:4,6)])
validknn_new.df[,c(1:4,6)] <- predict(norm.values, validknn_new[,c(1:4,6)])
knnn.df[,c(1:4,6)] <- predict(norm.values, knnn[,c(1:4,6)])
```

```
favrt <- as.data.frame(c(host_listings_count = 23 ,host_total_listings_count = 21 , b
edrooms = 1, beds = 2 , price = 350))%>% t()

fav.df <- predict(norm.values, favrt)

#aa <- trainknn_new.df[,7, drop = TRUE]

nn <- knn(train = trainknn_new.df[ ,c(1:4,6)],  test = fav.df,
          cl = trainknn_new.df[,7] , k = 5)

nn
```

```
## [1] 1
## attr(,"nn.index")
##      [,1] [,2] [,3] [,4] [,5]
## [1,]  132  276  246  325   74
## attr(,"nn.dist")
##      [,1] [,2] [,3] [,4] [,5]
## [1,] 9.95 13.3 13.4 13.4 15.4
## Levels: 1
```

```
row.names(trainknn_new)[attr(nn, "nn.index")]
```

```
## [1] "132" "276" "246" "325" "74"
```

```
### showing the neighbour and they all have kitchen and hot water in the amenities

new <- trainknn_new[c(132, 276 ,246, 325 ,74), c(5,7)]
new
```

```
##
amenities
## 132 ["Cleaning before checkout", "Cooking basics", "Carbon monoxide alarm", "Coffe
e maker", "Host greets you", "Paid parking on premises", "Microwave", "Dedicated work
space", "Refrigerator", "Free street parking", "Hot water", "Essentials", "Private en
trance", "Shower gel", "Room-darkening shades", "Breakfast", "Hangers", "Iron", "Paid
parking off premises", "Ethernet connection", "Shampoo", "Kitchen", "Wifi", "Stove",
"TV", "First aid kit", "Dishes and silverware", "Fire extinguisher", "Smoke alarm",
"Extra pillows and blankets", "Heating", "Bed linens", "Long term stays allowed", "Ha
ir dryer", "Pocket wifi"]
## 276
["Building staff", "Crib", "Dedicated workspace", "Shampoo", "Indoor fireplace", "Kit
chen", "Heating", "Bed linens", "Long term stays allowed", "Hot water", "Essentials",
"Wifi", "Hair dryer", "TV", "Hangers", "Iron", "Smoke alarm"]
## 246
["Building staff", "Cooking basics", "Coffee maker", "Microwave", "Dedicated workspac
e", "Refrigerator", "Indoor fireplace", "Hot water", "Essentials", "Hangers", "Shampo
o", "Kitchen", "Wifi", "TV", "Dishes and silverware", "Smoke alarm", "Crib", "Heatin
g", "Bed linens", "Long term stays allowed", "Hair dryer"]
## 325
["Building staff", "Crib", "Microwave", "Dedicated workspace", "Refrigerator", "Shamp
oo", "Kitchen", "Heating", "Long term stays allowed", "Hot water", "Essentials", "Wif
i", "Hair dryer", "TV", "Cooking basics", "Dishes and silverware", "Hangers", "Coffee
maker", "Smoke alarm"]
## 74
["Building staff", "Carbon monoxide alarm", "Pack \\u2019n play/Travel crib", "Dedica
ted workspace", "Cable TV", "Hot water", "Essentials", "Children\\u2019s books and to
ys", "Hangers", "TV with standard cable", "Iron", "Ethernet connection", "Luggage dro
poff allowed", "Shampoo", "Wifi", "First aid kit", "Fire extinguisher", "Smoke alar
m", "Crib", "Extra pillows and blankets", "Heating", "Bed linens", "Long term stays a
llowed", "Hair dryer"]
##     has_kitchen
## 132           1
## 276           1
## 246           1
## 325           1
## 74            1
```

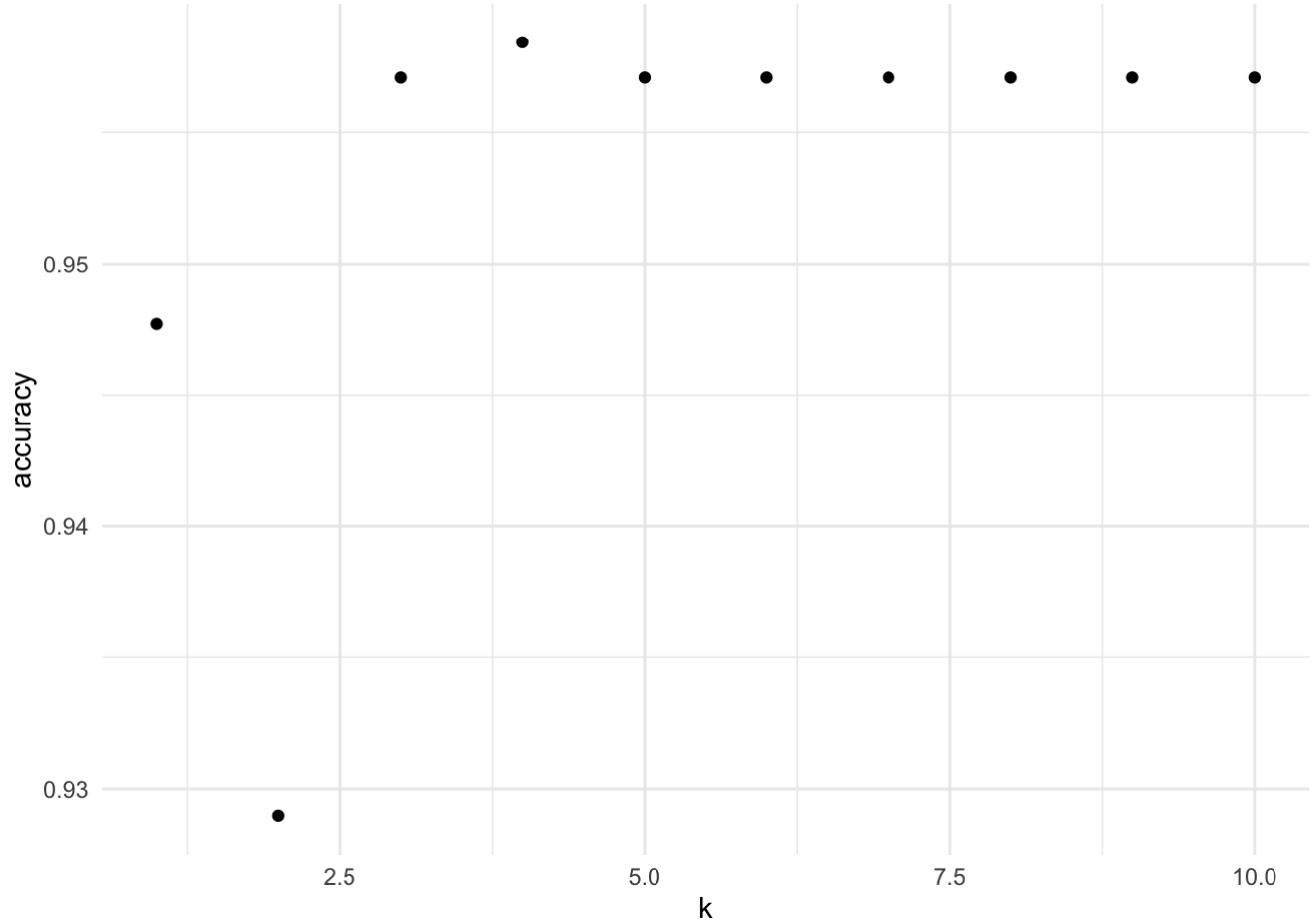
```
# initialize a data frame with two columns: k, and accuracy.
accuracy.df <- data.frame(k = seq(1, 10, 1), accuracy = rep(0, 10))

# compute knn for different k on validation.
for(i in 1:10) {
  knn.pred <- knn(train = trainknn_new.df[ ,c(1:4,6)], validknn_new.df[ ,c(1:4,6)] ,
          cl = trainknn_new.df[,7], k = i)
  accuracy.df[i, 2] <- confusionMatrix(knn.pred, validknn_new.df[,7])$overall[1]
}
```

```
## Warning in confusionMatrix.default(knn.pred, validknn_new.df[, 7]): Levels are
## not in the same order for reference and data. Refactoring data to match.

## Warning in confusionMatrix.default(knn.pred, validknn_new.df[, 7]): Levels are
## not in the same order for reference and data. Refactoring data to match.

## Warning in confusionMatrix.default(knn.pred, validknn_new.df[, 7]): Levels are
## not in the same order for reference and data. Refactoring data to match.

## Warning in confusionMatrix.default(knn.pred, validknn_new.df[, 7]): Levels are
## not in the same order for reference and data. Refactoring data to match.

## Warning in confusionMatrix.default(knn.pred, validknn_new.df[, 7]): Levels are
## not in the same order for reference and data. Refactoring data to match.

## Warning in confusionMatrix.default(knn.pred, validknn_new.df[, 7]): Levels are
## not in the same order for reference and data. Refactoring data to match.
```

```
accuracy.df
```

```
##     k accuracy
## 1   1    0.948
## 2   2    0.929
## 3   3    0.957
## 4   4    0.958
## 5   5    0.957
## 6   6    0.957
## 7   7    0.957
## 8   8    0.957
## 9   9    0.957
## 10 10    0.957
```

```
ggplot(accuracy.df, aes(x = k , y = accuracy))+geom_point()+theme_minimal()
```



K = 4 is the best to predict the class

```
favrt <- as.data.frame(c(host_listings_count = 23 ,host_total_listings_count = 21 , b
edrooms = 1, beds = 2 , price = 350))%>% t()

fav.df <- predict(norm.values, favrt)

#aa <- trainknn_new.df[,7, drop = TRUE]

nn <- knn(train = trainknn_new.df[ ,c(1:4,6)],  test = fav.df,
          cl = trainknn_new.df[,7] , k = 4)

nn
```

```
## [1] 1
## attr(,"nn.index")
##      [,1] [,2] [,3] [,4]
## [1,]  132  276  246  325
## attr(,"nn.dist")
##      [,1] [,2] [,3] [,4]
## [1,] 9.95 13.3 13.4 13.4
## Levels: 1
```

```
row.names(trainknn_new)[attr(nn, "nn.index")]
```

```
## [1] "132" "276" "246" "325"
```

```
### showing the neighbor and they all have kitchen and hot water in the amenities

new <- trainknn_new[c(132, 276 ,246, 325), c(5,7)]
new
```

```
##
amenities
## 132 ["Cleaning before checkout", "Cooking basics", "Carbon monoxide alarm", "Coffe
e maker", "Host greets you", "Paid parking on premises", "Microwave", "Dedicated work
space", "Refrigerator", "Free street parking", "Hot water", "Essentials", "Private en
trance", "Shower gel", "Room-darkening shades", "Breakfast", "Hangers", "Iron", "Paid
parking off premises", "Ethernet connection", "Shampoo", "Kitchen", "Wifi", "Stove",
"TV", "First aid kit", "Dishes and silverware", "Fire extinguisher", "Smoke alarm",
"Extra pillows and blankets", "Heating", "Bed linens", "Long term stays allowed", "Ha
ir dryer", "Pocket wifi"]
## 276
["Building staff", "Crib", "Dedicated workspace", "Shampoo", "Indoor fireplace", "Kit
chen", "Heating", "Bed linens", "Long term stays allowed", "Hot water", "Essentials",
"Wifi", "Hair dryer", "TV", "Hangers", "Iron", "Smoke alarm"]
## 246
["Building staff", "Cooking basics", "Coffee maker", "Microwave", "Dedicated workspac
e", "Refrigerator", "Indoor fireplace", "Hot water", "Essentials", "Hangers", "Shampo
o", "Kitchen", "Wifi", "TV", "Dishes and silverware", "Smoke alarm", "Crib", "Heatin
g", "Bed linens", "Long term stays allowed", "Hair dryer"]
## 325
["Building staff", "Crib", "Microwave", "Dedicated workspace", "Refrigerator", "Shamp
oo", "Kitchen", "Heating", "Long term stays allowed", "Hot water", "Essentials", "Wif
i", "Hair dryer", "TV", "Cooking basics", "Dishes and silverware", "Hangers", "Coffee
maker", "Smoke alarm"]
##     has_kitchen
## 132           1
## 276           1
## 246           1
## 325           1
```

The purpose of the KNN model in for the air bnb is to find out whether a rental in your neighborhood will have some particular amenity, or combination of amenities.We builded up the KNN model with Kitchen and Hot water since these are the two important things after bedroom and bathroom requirements, as noticed, the level are set as if the room has kitchen and hot water, Y means yes N means no, as result, only 5 of 463 will not have kitchen and hot water, which 98.9% of time the rent room will include kitchen and hot water.

Naive Bayes

```
str(Batignolles$property_type)
```

```
##  chr [1:1243] "Entire rental unit" "Entire rental unit" ...
```

```
Batignolles1 <- Batignolles[,c(11,20:22,54)]

Batignolles1$instant_bookable<-as.factor(Batignolles1$instant_bookable)
Batignolles1$property_type<-as.factor(Batignolles1$property_type)
Batignolles1$accommodates<-as.factor(Batignolles1$accommodates)
Batignolles1$room_type<-as.factor(Batignolles1$room_type)
Batignolles1$host_listings_count<-as.factor(Batignolles1$host_listings_count)

table(Batignolles1$host_listings_count)
```

```
##
##   0   1   2   3   4   5   6   8  31
## 110 935 124  26   8  24   6   8   2
```

```
str(Batignolles1)
```

```
## tibble [1,243 × 5] (S3: tbl_df/tbl/data.frame)
##  $ host_listings_count: Factor w/ 9 levels "0","1","2","3",..: 1 3 1 3 2 2 2 2 1 2
...
##  $ property_type      : Factor w/ 16 levels "Entire condominium (condo)",..: 4 4 4
4 4 1 4 3 4 4 ...
##  $ room_type          : Factor w/ 4 levels "Entire home/apt",..: 1 1 1 1 1 1 1 1 1
1 ...
##  $ accommodates       : Factor w/ 10 levels "1","2","3","4",..: 4 4 4 2 2 5 2 7 2
2 ...
##  $ instant_bookable   : Factor w/ 2 levels "FALSE","TRUE": 1 1 2 1 1 1 2 1 2 1 ...
```

```
trainnb<-sample_frac(Batignolles1,0.6)
validnb<-setdiff(Batignolles1,trainnb)

colnames(Batignolles1)
```

```
## [1] "host_listings_count" "property_type"       "room_type"
## [4] "accommodates"        "instant_bookable"
```

```
model.nb <- naiveBayes(instant_bookable ~ . , data = trainnb)
model.nb
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
## FALSE  TRUE
## 0.779 0.221
##
## Conditional probabilities:
##        host_listings_count
## Y             0       1       2       3       4       5       6       8      31
##   FALSE 0.08262 0.78830 0.09639 0.01893 0.01033 0.00172 0.00000 0.00000 0.00172
##   TRUE  0.10303 0.64242 0.10909 0.01212 0.00000 0.07879 0.02424 0.03030 0.00000
##
##        property_type
## Y       Entire condominium (condo) Entire guesthouse Entire loft
##   FALSE                    0.02238           0.00000     0.00344
##   TRUE                     0.03636           0.00606     0.00606
##        property_type
## Y       Entire rental unit Entire residential home Entire townhouse
##   FALSE            0.83649                 0.00172          0.00344
##   TRUE             0.70303                 0.00000          0.00000
##        property_type
## Y       Private room in bed and breakfast Private room in condominium (condo)
##   FALSE                           0.00344                              0.00344
##   TRUE                            0.01818                              0.00000
##        property_type
## Y       Private room in loft Private room in rental unit
##   FALSE              0.00344                    0.11360
##   TRUE               0.00000                    0.13939
##        property_type
## Y       Private room in residential home Room in bed and breakfast
##   FALSE                          0.00172                  0.00000
##   TRUE                           0.00000                  0.01212
##        property_type
## Y       Room in boutique hotel Room in hotel Room in serviced apartment
##   FALSE                0.00172       0.00000                    0.00000
##   TRUE                 0.03636       0.03030                    0.00000
##        property_type
## Y       Shared room in rental unit
##   FALSE                    0.00516
##   TRUE                     0.01212
##
##        room_type
## Y       Entire home/apt Hotel room Private room Shared room
##   FALSE         0.86747    0.00000      0.12737     0.00516
##   TRUE          0.75152    0.03636      0.20000     0.01212
##
##        accommodates
## Y             1       2       3       4       5       6       7       8      10
##   FALSE 0.05852 0.53356 0.06713 0.25129 0.03270 0.03959 0.00688 0.00861 0.00172
##   TRUE  0.06061 0.60000 0.12121 0.13333 0.03030 0.04242 0.00000 0.00000 0.00000
```

```
##        accommodates
## Y            12
##   FALSE 0.00000
##   TRUE  0.01212
```

```
# Creating a confusion matrix

train_pred<-predict(model.nb,newdata=trainnb)
confusionMatrix(train_pred,trainnb$instant_bookable)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE   574  139
##      TRUE      7   26
##
##                Accuracy : 0.804
##                  95% CI : (0.774, 0.832)
##     No Information Rate : 0.779
##     P-Value [Acc > NIR] : 0.0498
##
##                   Kappa : 0.204
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.988
##             Specificity : 0.158
##          Pos Pred Value : 0.805
##          Neg Pred Value : 0.788
##              Prevalence : 0.779
##          Detection Rate : 0.769
##    Detection Prevalence : 0.956
##       Balanced Accuracy : 0.573
##
##        'Positive' Class : FALSE
##
```

```
#validation
valid_pred<-predict(model.nb,newdata=validnb)
confusionMatrix(valid_pred,validnb$instant_bookable)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE    10   10
##      TRUE      5    9
##
##                Accuracy : 0.559
##                  95% CI : (0.379, 0.728)
##     No Information Rate : 0.559
##     P-Value [Acc > NIR] : 0.571
##
##                   Kappa : 0.136
##
##  Mcnemar's Test P-Value : 0.302
##
##             Sensitivity : 0.667
##             Specificity : 0.474
##          Pos Pred Value : 0.500
##          Neg Pred Value : 0.643
##              Prevalence : 0.441
##          Detection Rate : 0.294
##    Detection Prevalence : 0.588
##       Balanced Accuracy : 0.570
##
##        'Positive' Class : FALSE
##
```

```
# Fictional apartment
fapt<-data.frame(property_type="Entire rental unit", accommodates="4",room_type=" Ent
ire home/apt",host_listings_count= "5",host_is_superhost="FALSE")


str(fapt)
```

```
## 'data.frame':    1 obs. of  5 variables:
##  $ property_type      : chr "Entire rental unit"
##  $ accommodates       : chr "4"
##  $ room_type          : chr " Entire home/apt"
##  $ host_listings_count: chr "5"
##  $ host_is_superhost  : chr "FALSE"
```

```
nb_predict <- predict(model.nb,newdata=fapt)
nb_predict
```

```
## [1] TRUE
## Levels: FALSE TRUE
```

```
nb_predict1 <- predict(model.nb,newdata=fapt, type = "raw")
nb_predict1
```

```
##      FALSE  TRUE
## [1,] 0.147 0.853
```

```
table(Batignolles$room_type)
```

```
##
## Entire home/apt      Hotel room    Private room     Shared room
##            1053               8             176               6
```

First, we take a look at our data set, and we selected property_type;accommodates;room_type;host_listings_count;host_is_superhost as our predictors to predict instant_bookable.Then, we assigned all the variable that we selected to a new dataframe to make sure we won't mess up the original data set. Through observation we clearly see they all are not factors. If they are not factors we can't use them to make a Naive Bayes Model. Therefore, we used as.factor to change everything to factor type. Then, we set up the the Naive Bayes model as model.nb. Then, we make confusion matrix based on the predict training set and training set. The accuracy would be 0.81. And another confusion matrix for predict validation and validation set. The accuracy is lower than the accuracy on training set.

Fictional apartment We have made a fictional person for this part. His preferred property_type is Entire rental unit, the house should be able to contain 4 people, room_type is Entire home, The host should have 5 houses, and is not a super host. Then our model predicted that this fictional person is 90.3% True, which means prediction model contains everything. 9.7% False which means prediction model will not contain everything.

Classification Tree

```
Batignolles2<- Batignolles[,c(7:12,14,15,17:25,27:43,46)]
Batignolles2<-Batignolles2 %>% mutate(new_bin =
                                        cut(Batignolles2$review_scores_rating,
                                                   breaks=c(0,3.5,4.5,5),
                                                   labels= c('Low','Medium','high'
)))

Batignolles2<- Batignolles2[,-c(1:3,5,6,9,19:26,21:26,33:35)]
Batignolles2<- Batignolles2[,-c(4,5,18)]
```

```
Batignolles2 <- Batignolles2[complete.cases(Batignolles2), ]
aa <- miss_var_summary(Batignolles2)

set.seed(699)
sample_data = sample.split (Batignolles2, SplitRatio = 0.60)
train <- subset(Batignolles2)
test <- subset(Batignolles2)
options(scipen=999)
Tmodel1<-rpart(new_bin~.,method="class",data=train,cp= 0.0141 , xval=10)

b<-printcp(Tmodel1)
```

```
##
## Classification tree:
## rpart(formula = new_bin ~ ., data = train, method = "class",
##     cp = 0.0141, xval = 10)
##
## Variables actually used in tree construction:
## [1] availability_30        availability_365       host_identity_verified
## [4] host_is_superhost      price
##
## Root node error: 496/1217 = 0.4
##
## n= 1217
##
##     CP nsplit rel error xerror xstd
## 1 0.03      0       1.0    1.0 0.03
## 2 0.01      4       0.9    0.9 0.03
## 3 0.01      6       0.9    0.9 0.03
```

```
class(b)
```

```
## [1] "matrix" "array"
```

```
b<-data.frame(b)
which.min(b$xerror)
```

```
## [1] 2
```

```
which.min(b$xstd)
```

```
## [1] 2
```

```
predicting_train <- predict(Tmodel1, train, type='class')
confusion_train <- table(train$new_bin, predicting_train)
confusion_train
```

```
##         predicting_train
##          Low Medium high
##   Low      0     21   23
##   Medium   0    202  250
##   high     0    132  589
```

```
accuracy_train <- sum(diag(confusion_train)/sum(confusion_train))
accuracy_train
```

```
## [1] 0.65
```

```
predicting_test <- predict(Tmodel1, test, type='class')
```

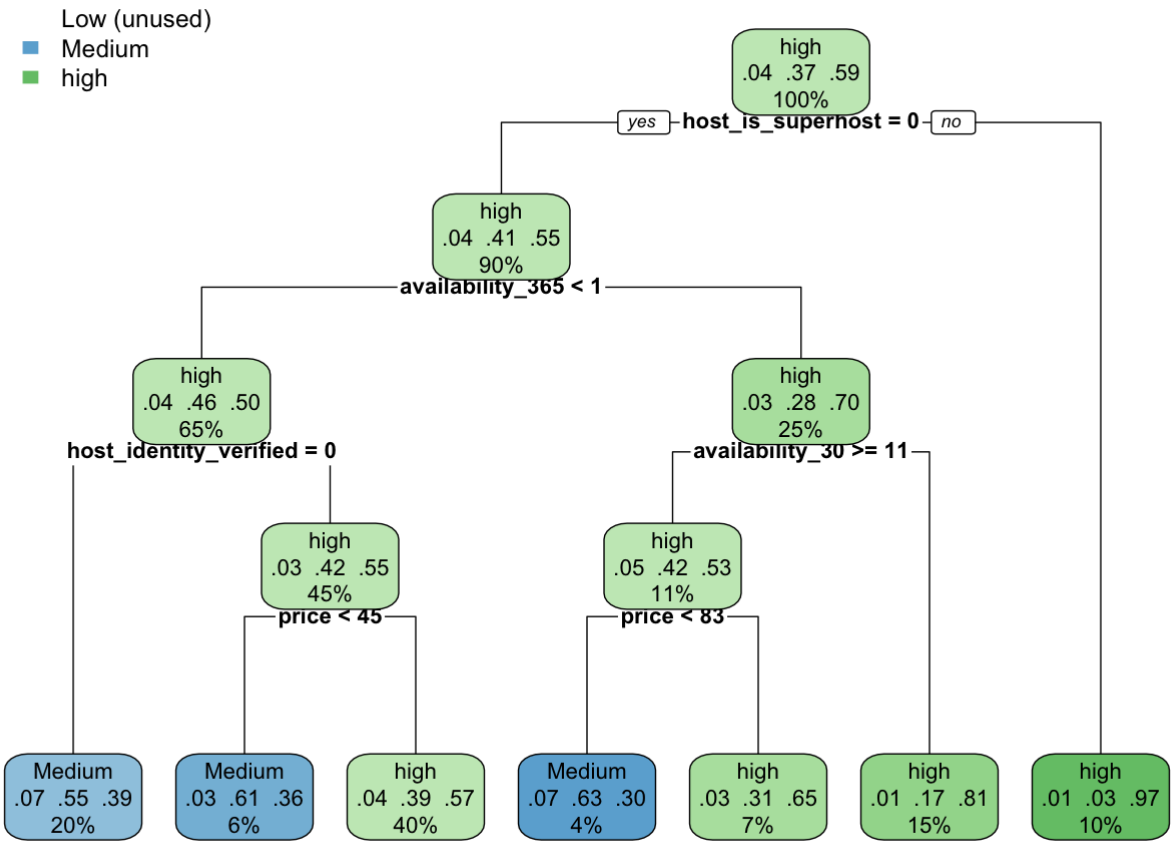
```
confusion_test <- table(test$new_bin, predicting_test)
confusion_test
```

```
##          predicting_test
##           Low Medium high
##    Low      0     21   23
##    Medium   0    202  250
##    high     0    132  589
```

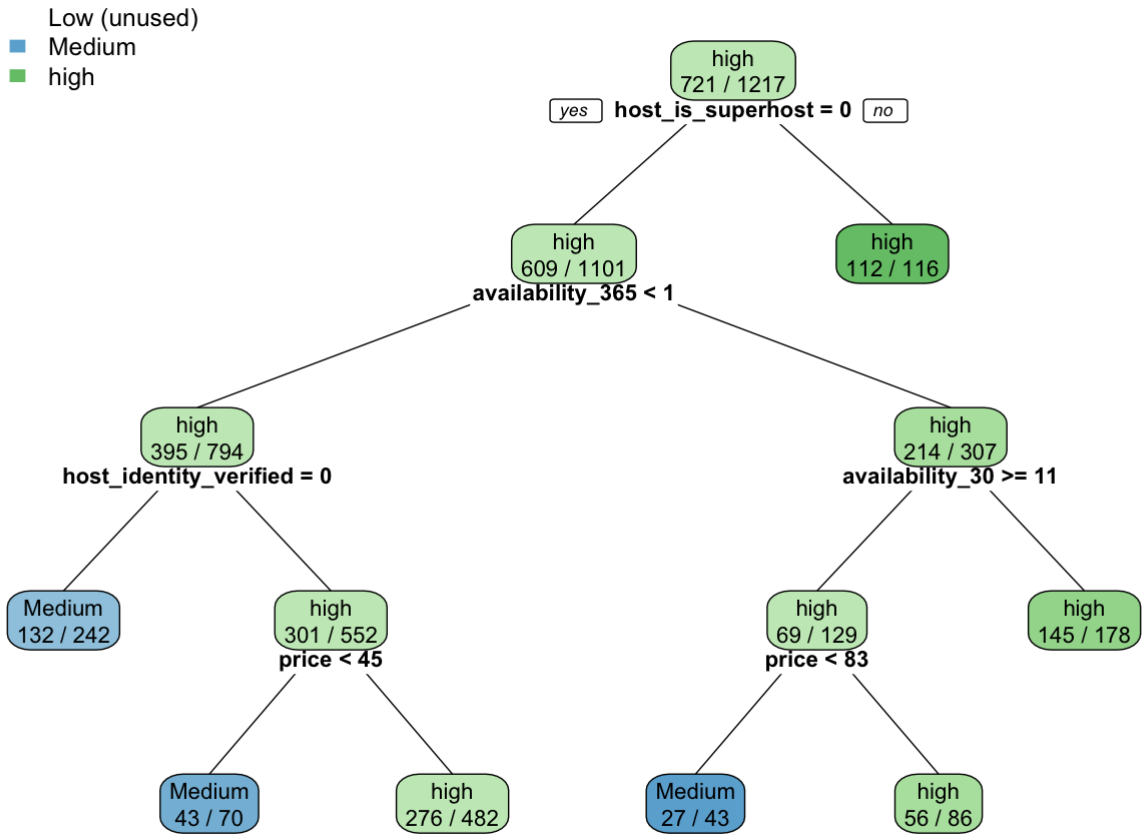
```
accuracy_test <- sum(diag(confusion_test)/sum(confusion_test))
accuracy_test
```

```
## [1] 0.65
```

```
rpart.plot(Tmodel1,cex=0.7)
```




```
Tmodel2<-rpart(new_bin~.,method="class",data=train,cp=0.0141, xval=10,minsplit=0, min
bucket=2, maxdepth=10)
rpart.plot(Tmodel2,cex=0.7,fallen.leaves  = FALSE,type=2,extra=2)
```



First and foremost, we must select the variables for the tree model. We selected 20 variables after evaluating each column and selecting the ones we believe are relevant to review scores. For the binning, we created three bins:0-3.5 low, 3.5-4.5 medium, 4.5-5 high. Then, to determine the optimal CP, we used cross validation method. We have CP as 0.0141, nsplit as 4, rel error as 0.9, xerror as 0.9,xstd as 0.03. After using 0.0141 as our CP, we have a 65.4% of accuracy rate. For example, If the rental's host is a super host and the availability is less than 1, and the host identity is validated, the rating falls into the medium category, according to the tree model. Moreover, we can find that the low category is not shown on the tree map, it is because the ratings of the most of the data is between 4.7 to 5, we rarely have bad reviews. After we imputed the random number between 3.5 to 4.5, we have more data in this range, but we still have few data below the rating 3.5.

K-means Clustering

```
cor(Batignolles[,c(28:35)])
```

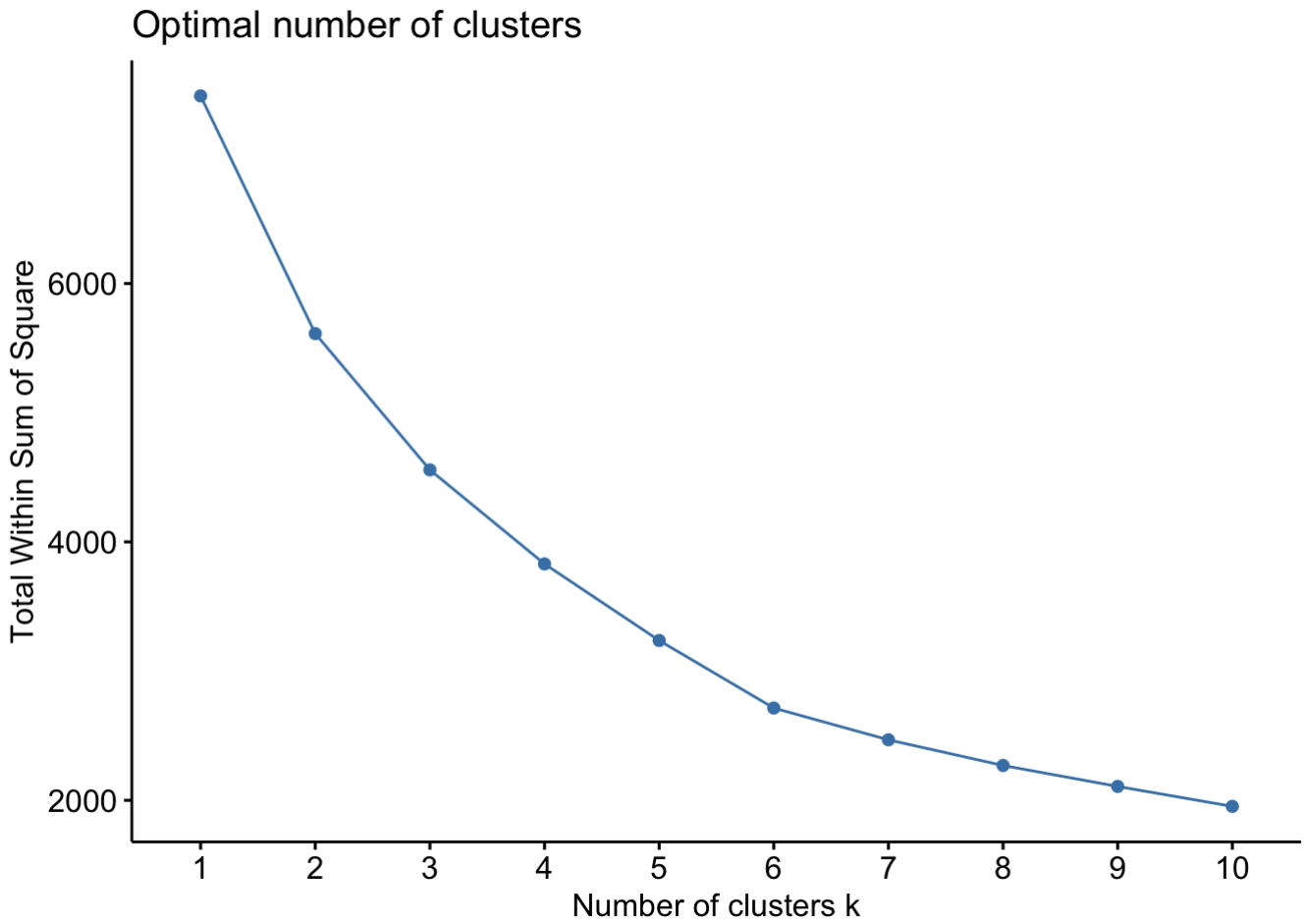
```
##                         minimum_nights maximum_nights minimum_minimum_nights
## minimum_nights                   1.000          0.516                   1.000
## maximum_nights                   0.516          1.000                   0.516
## minimum_minimum_nights           1.000          0.516                   1.000
## maximum_minimum_nights           1.000          0.517                   1.000
## minimum_maximum_nights           0.452          0.849                   0.452
## maximum_maximum_nights           0.450          0.850                   0.450
## minimum_nights_avg_ntm           1.000          0.516                   1.000
## maximum_nights_avg_ntm           0.451          0.851                   0.451
##                         maximum_minimum_nights minimum_maximum_nights
## minimum_nights                           1.000                   0.452
## maximum_nights                           0.517                   0.849
## minimum_minimum_nights                   1.000                   0.452
## maximum_minimum_nights                   1.000                   0.453
## minimum_maximum_nights                   0.453                   1.000
## maximum_maximum_nights                   0.451                   0.996
## minimum_nights_avg_ntm                   1.000                   0.453
## maximum_nights_avg_ntm                   0.452                   0.998
##                         maximum_maximum_nights minimum_nights_avg_ntm
## minimum_nights                           0.450                   1.000
## maximum_nights                           0.850                   0.516
## minimum_minimum_nights                   0.450                   1.000
## maximum_minimum_nights                   0.451                   1.000
## minimum_maximum_nights                   0.996                   0.453
## maximum_maximum_nights                   1.000                   0.450
## minimum_nights_avg_ntm                   0.450                   1.000
## maximum_nights_avg_ntm                   1.000                   0.451
##                         maximum_nights_avg_ntm
## minimum_nights                           0.451
## maximum_nights                           0.851
## minimum_minimum_nights                   0.451
## maximum_minimum_nights                   0.452
## minimum_maximum_nights                   0.998
## maximum_maximum_nights                   1.000
## minimum_nights_avg_ntm                   0.451
## maximum_nights_avg_ntm                   1.000
```

```
k_meansss <- Batignolles[,c(18,19,21,22,25,27,28,29,40,60)]
```

```
# scaling
km.norm <- sapply(k_meansss[,c(4,6:10)], scale)
```

elbow method with sum of square

```
f1 <- fviz_nbclust(km.norm,kmeans,method="wss",nstart=107)
f1
```



Making clusters

```
k1 <- kmeans(km.norm,centers = 5, nstart = 107)

k1$centers
```

```
##   accommodates  price minimum_nights maximum_nights availability_365
## 1        0.115 -0.190          0.801         0.5386           -0.415
## 2       -0.325  0.345         -0.428        -0.0628            2.444
## 3       -0.316 -0.196         -0.943        -1.8003           -0.181
## 4        2.282  2.933         -0.665        -0.0245            0.912
## 5       -0.111 -0.235          0.153         0.5485           -0.406
##   actual_score
## 1       -1.383
## 2        0.272
## 3        0.421
## 4        0.216
## 5        0.545
```

To create clusters of air BNB in our neighborhood, We used K-means analysis as it would be an effective way to place rental units within your neighborhood into clusters. K-means clustering is actually a type of unsupervised learning, which is used when you have unlabeled data (i.e., data without defined categories or groups). The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K. In our case, we are going to use different variable from our data to place the rental units within your neighborhood into clusters. We are also going to try to find the most optimal K value

We used another data frame for our K-means model, The reason is because it will help us prevent form doing anything in our original data set as that data set in being used in different models as well. We creating this data frame with only the useful variables for our clustering. These variable would be Accommodates, Prices, Minimum nights, maximum_nights, availability_365 and actual_score (feature engineered column describe while imputing NA's). We though these variable would be best to place rental units within your neighborhood into clusters because this will what helps a person decide which air BNB fits them.

After creating this data frame, we then normalized the value. The data need to be scaled because all the variable that we are going to use are in different dimensions and if we you use the data without scaling it then model will become weird and will give higher weightage to the variable which is higher in number count and will give less weightage to the variable that has less numbers.

After creating this data frame we then create the elbow chart, which gave us nothing about the optimal k value to be used. Hence, we just used different k values and find the optimal k value to be 5. Now let's describe our clusters

Cluster 1 = " Flex inn " = In this cluster people does not have the flexibility to book the air BNB as the rental units in this cluster have very low minimum_nights and maximum_nights. But the these rental units are availability about decent number of times through out the year. The have above average rating that means people do like this place and the prices are also below average and does not accommodated that many people.

Cluster 2 = "House Party" = In this cluster we can see that it accommodates a lot of people and also their prices are high above average price charged. They also have flexibility in the maximum nights you can book it for along side minimum nights too. It has more than average availability through out the year and people like it but not not that there scores are a little above average.

Cluster 3 = " Vacancy inn" = In this cluster the first thing we can see that is that they are high available through out the year maybe because we can see that the prices for this rental unit is higher than average and it doesn't even accommodated that many people in this rental unit. They do have the flexibility in booking the air-BNB but but not that much in the minimum nights as compaired to the cluster number 4.

Cluster 4 = " Holiday Inn " = We gave this cluster this name because it would be good for holidays as the prices are lower than any other clusters but also does not accommodated that many people in the rental units. They do not have the flexibility to book the rental unit for minimum nights and but there is flexibility in maximum nights for this rental. They not have availability through out the year most probability lower prices and people tend to like these rental.

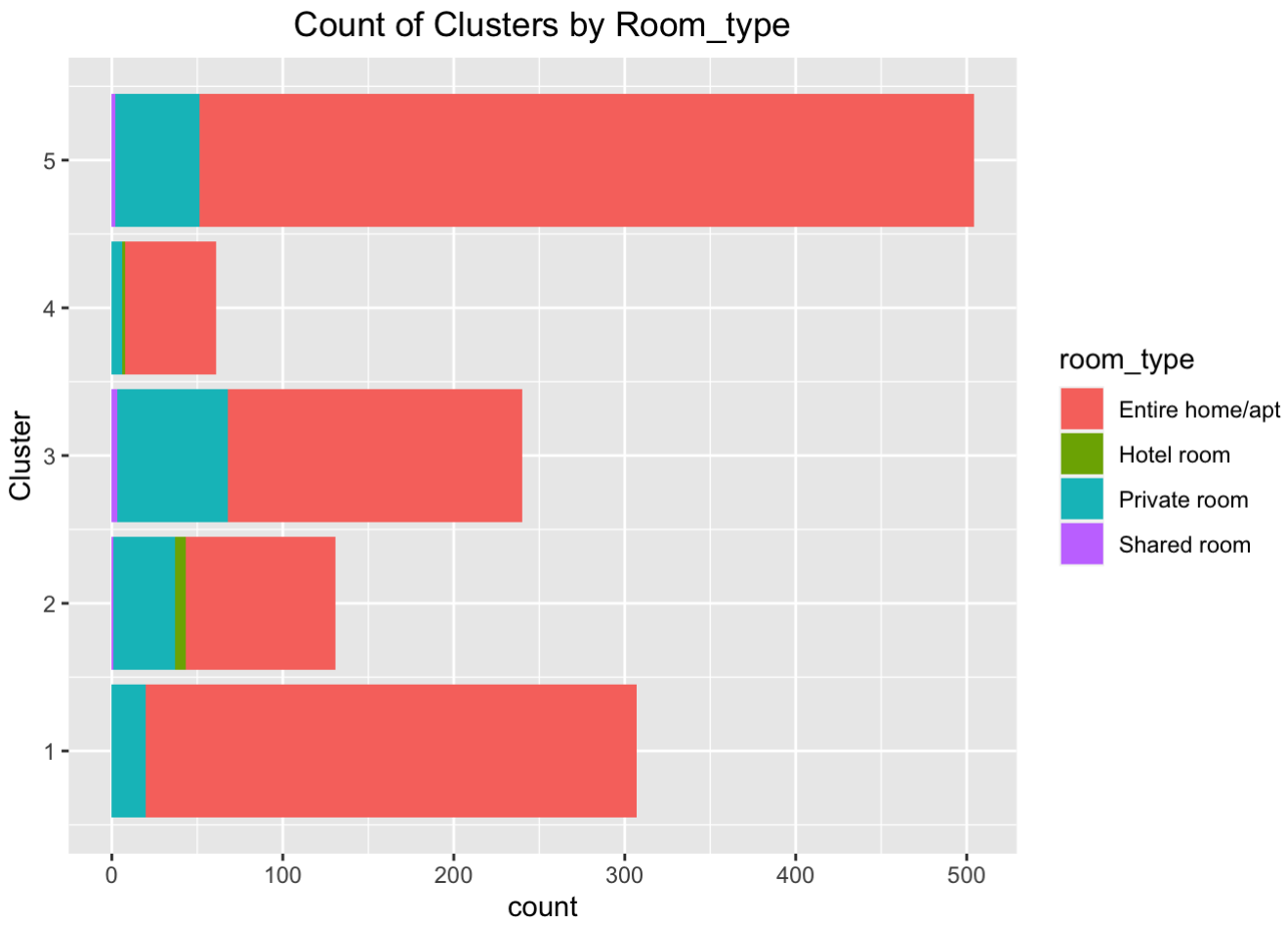
Cluster 5 = " Trash binn " = In this cluster the first observation we made was that people do not like it at all. Their actual score is way too less than average score. This could be because the did not have that much availability through out the year and also the are not very flexibility in the minimum nights but they are flexibility in maximum nights. There prices are less than average and do accommodates more than average people. Hence, this lead us to believe that these rental units does not have nice place / clean place or does not have nice service.

Assigning the cluster back to data set

```
k_means <- k_meansss %>% mutate(Cluster = k1$cluster)
```

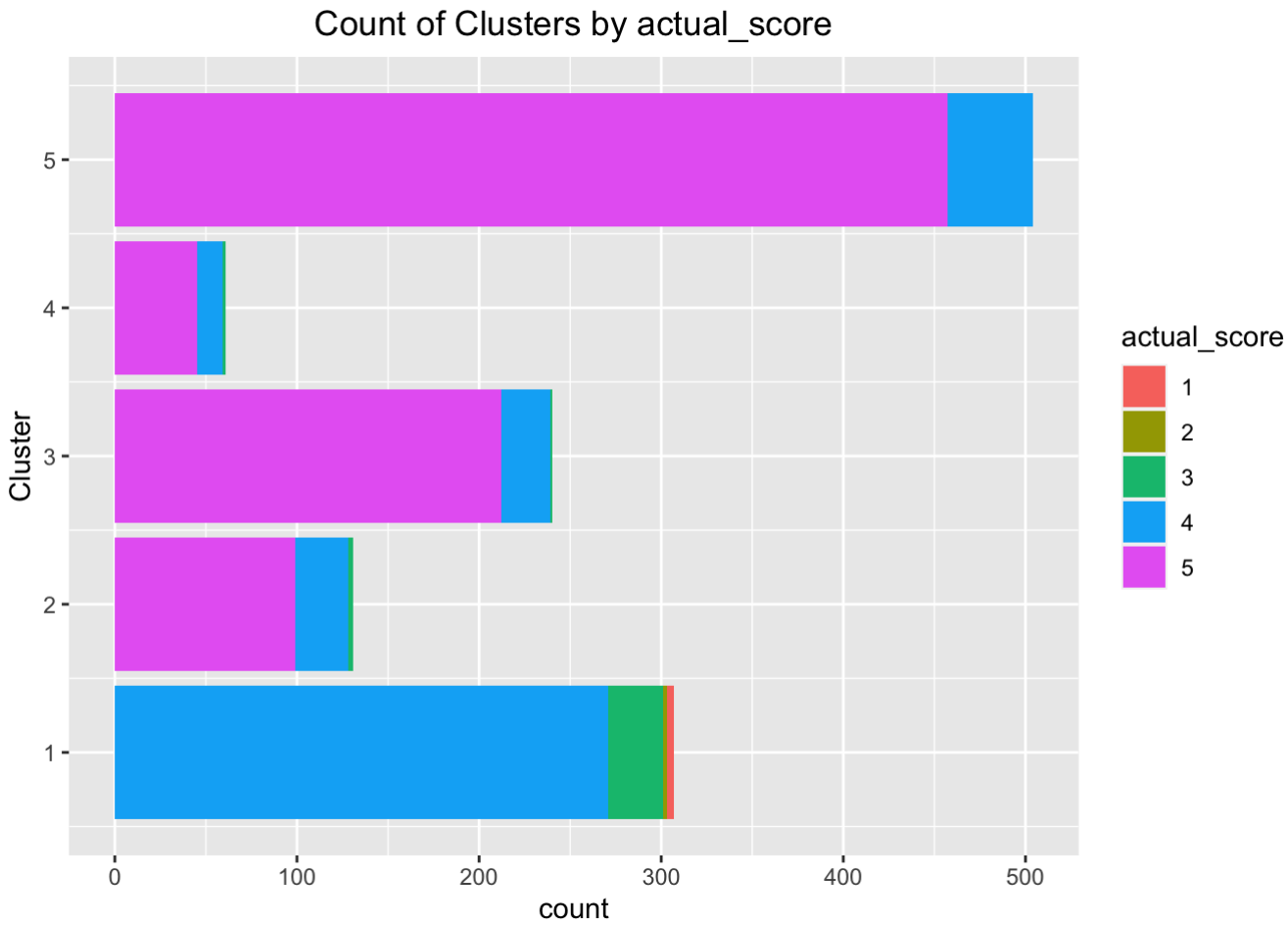
visualization for clustering

```
# Count of Clusters by Room_type
ggplot(data = k_means, aes(y = Cluster)) +
geom_bar (aes(fill = room_type)) +
ggtitle("Count of Clusters by Room_type") +
theme(plot.title = element_text(hjust = 0.5))
```



This graph shows the proportion of room type in each cluster. In the figure we can see that the percentage of entire room/apt is always the largest in either cluster. In cluster 3, the proportion of private rooms is about one-fifth, which is the largest proportion of that type of room among the five clusters. In addition, there is a complete absence of shared room in cluster 1 and cluster 4.

```
# Count of Clusters by actual_score.
k_means1 <- k_means[,c(10,11)]
k_means1$actual_score <- k_means1$actual_score %>% round() %>% as.character()
ggplot(data = k_means1, aes(y = Cluster)) +
  geom_bar(aes(fill = actual_score)) +
  ggtitle("Count of Clusters by actual_score") +
  theme(plot.title = element_text(hjust = 0.5))
```

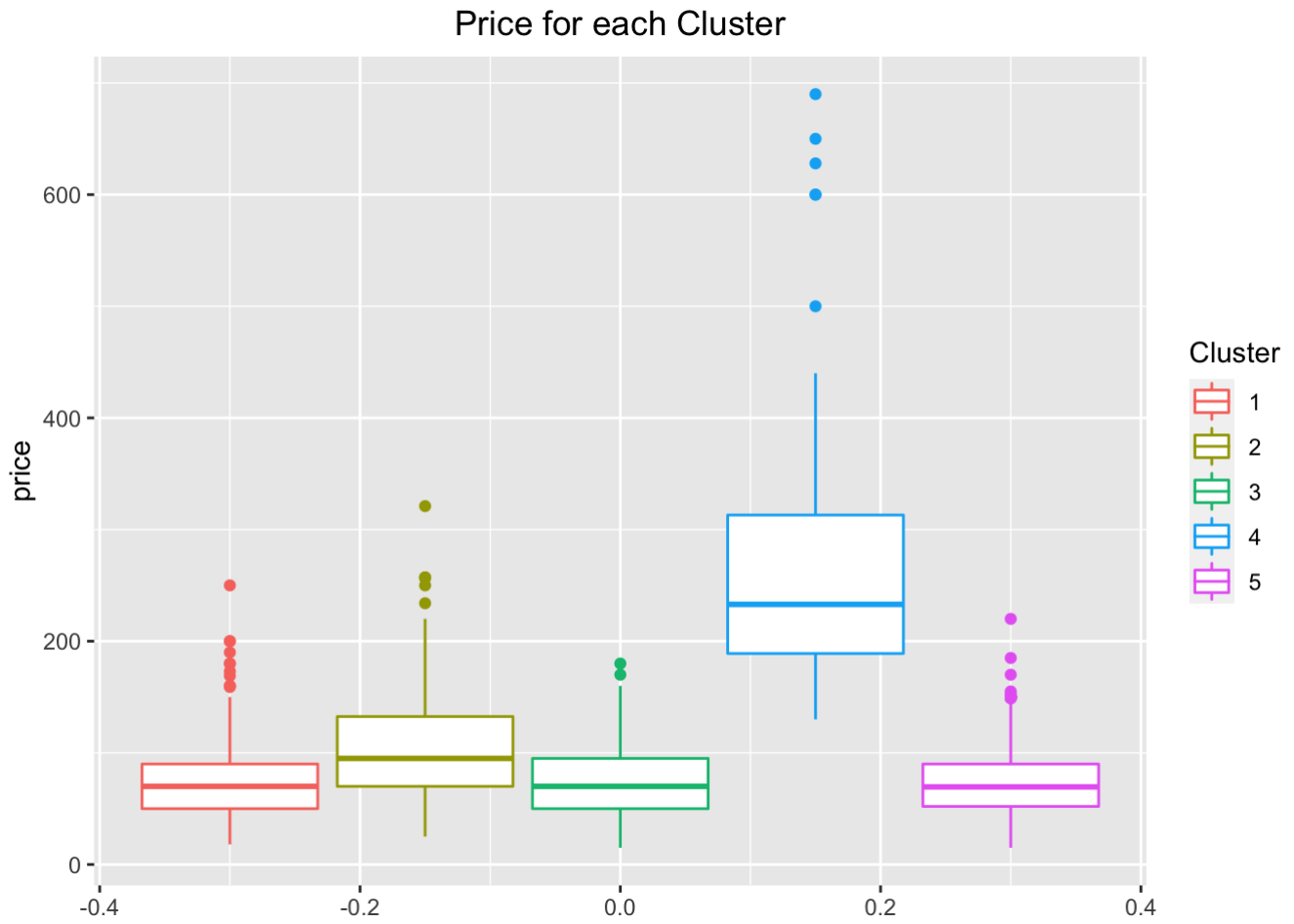


This graph shows the proportion of actual rating in each cluster. From the figure we can conclude that, except for cluster 1, the scores of all the other clusters are mainly composed of five points, accompanied by a few four points, and hardly ever below three points. For cluster 1, most of the scores are four and a few are three, and there is even the lowest score of one among all the scores for cluster 1. From this we can infer that Airbnb rooms belonging to cluster 1 are not well accepted by customers.

```
# Price for each Cluster.
k_means2 <- k_means[,c(5,6,11)]
k_means2$Cluster <- k_means2$Cluster %>% as.factor()
colnames(k_means2)
```

```
## [1] "beds"    "price"   "Cluster"
```

```
ggplot(data = k_means2, aes( y = price, color = Cluster)) +
  geom_boxplot( ) +
  ggtitle("Price for each Cluster") +
  theme(plot.title = element_text(hjust = 0.5))
```



This graph shows the price distribution of the different clusters in the form of a box plot. From the graph we can conclude that the overall price of cluster 4 is higher relative to all other clusters, the price of cluster 2 is in the second place among the 5 clusters, and the remaining prices of cluster 1, cluster 3 and cluster 5 are relatively similar.

Conclusions

Our findings are useful for Airbnb's research on its hosts and customers. We found several factors that correlate with listing ratings, such as room type, room price, and number of people a room can accommodate. Through our research, Airbnb can infer the most popular room combinations in the local area, attract hosts with that type of combination to join Airbnb, and use that type of combination to attract customers to increase market share and boost sales.

It can also be used by Airbnb to offer more shared rooms in their listings since people with a low budget can book shared rooms. Currently there are very few shared rooms available to the customers.

In the section on clustering, we found that cluster 1 has the lowest rating of all clusters. Airbnb can study the characteristics of this cluster and analyze why customers rate such rooms low. Airbnb can either find ways to improve such rooms or cut the number of rooms in this category and focus Airbnb can either find ways to improve these rooms or cut the number of rooms in this category and focus its operating costs on other room clusters that are more popular with customers to increase revenue.

For the Naive Bayes, it tells us if a person preferred property type is Entire rental unit, the house should be able to contain 4 people, room type is Entire home, The host should have 5 houses, and is not a super host 90.3% he will get a house that with all that features. However, it means 90.3% of data in our model will match this combination, which is shown us almost all of variable will follow this combination. Airbnb should have more combination with all those variables, and it is too monotonous.