Airbnb Rent Price Prediction

Kaggle Project Report

Zhenyu Cheng

1. Summary

In recent years, people became more and more interested in renting apartments or house when traveling instead of living in a hotel, since they could get closed to the local culture. This project is aimed for predicting the rental price based on the information provided on Airbnb including the property, renter, reviews, and transportation, etc. The prediction is measured by RMSE, the lower RMSE, the better prediction I have.

2. Initial exploration

After reading the training data, I took a glimpse of the data and analyzed the structure of the data, which contains multiple types of data, including numeric, categorical, and simple text data; Additionally, after exploring the scoring dataset, it is a must to process the scoring data in the same way as we did to the training data, for the linear regression model, we also need to check if the levels of the factor variable is the same between training data and scoring data.

a. Numeric Variable Processing

There are lots of numeric data we need to consider, such as cleaning fee, securities fees, this type of data is easy to be manipulated but there are some NA values we need to consider, the way I deal with those NA values is to add them with the mean of that specific feature.

b. Categorical Variable Processing

For variables like property type, room type or cancellation policy, those variables are originally stored as character, we need to convert them into factor variables for modelling.

c. Categorical Variable with more than 53 levels

A limit of random forest and gradient boost machine model is that they could not handle categorical variables with more than 53 levels, there are some variables like neighborhood cleansed which should be useful for prediction based on business sensitivity, so I choose the top 52 levels and group the other levels into a new level named “others”, for each level, to let them follow the same prediction track, which means contribute equally to the price prediction, I calculated the mean price for each group as a supplement variable for prediction. So, for each of those long level categorical variables, I have created two more variables as a substitute to better predict the price. For city variable, which contains more than 200 levels as well, I found out the text error and found out the top 11 cities mentioned, and the rest cities were represented as "others".

d. Text Variables Processing

For features like description, usually the longer description may give the customers more understanding of the apartment so it may lead to higher rent price, based on this business sensitivity, I calculated the number of words of such "description" variables for prediction. There is another kind of text variable like amenities, the more amenities may lead to higher rent price, so I calculated them as the number of amenities.

e. Time Variable Processing

There are some time variables in the dataset, like the host since, last review. To better utilize those variables, I used today to minus the date variable to get the number of days until now for further use.

3. Models and feature selection

a. Linear Regression
The primary regression model I tried is the linear regression, which is the most common regression model, I tried to run lasso regression to select features, but it takes long time to do so. Concerning that linear regression will not return me the best prediction in the end, I did not run feature selection for the project and have only the selection based on my business sensitivity (theoretical feature selection). For linear regression, predict function could not handle factors in the test set, which is not included in the train set, so I manually check which variable contains factors that is not included in the analytics dataset. I have submitted one model on predicting the scoring data using 69 features, and it has a RMSE 69 on private leaderboard. Information regarding to this model is on the appendix.

b. Random Forest
The second model I chose is the random forest, which primarily I think may be the best model for predicting this project, since I have properly deal with the multiple factor variables, I can simply run the model same as linear regression, but I tried to delete some variable that may be led to singularity, so I choose 50 out of 69 variables for predicting. For this model, I did not tune the model and used the default parameter for prediction, which is a reason of lower accuracy. The submission of this prediction will have a RMSE of 61 on private leaderboard.

c. Gradient Boost Machine

The third model I chose is the boosting model, which has more tolerance on feature selection and could better handle the overfitting issues, so I used all features I mentioned above for prediction. Regarding the parameters, I have tried shrinkage ratio (learning rate) 0.001, 0.003, 0.005 and 0.01, the best prediction learning rate is done by 0.005, and I tired 3, 5 ,7, 9 for the interaction depth as well. Tuning the model is a time-consuming process so I may not find out the best parameter for prediction based on my computer. But boosting return me the best prediction with an RMSE 57.6 on private leaderboard.

4. Model comparison

As discussed above, boosting will have the best prediction on scoring dataset. And that is the final submission I have on this project, the code for this submission is on the appendix.

5. Discussion

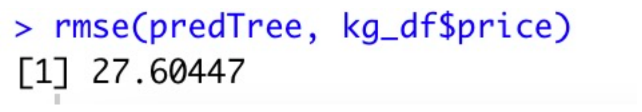
Data analytics is interesting project, from cleaning and tidying the data, and running the feature selection, then trying different models. The most interest part of the project is to tune the model and find out the difference made from different parameters.

6. Future direction

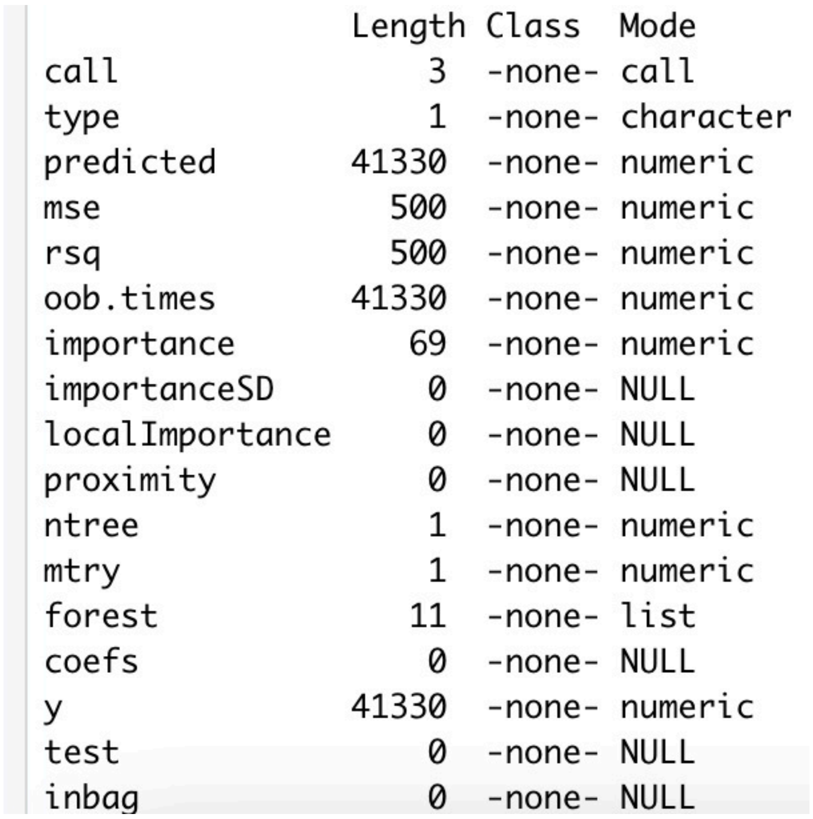
I did not consider the distribution of the data, there might have some outliers in the analytics dataset which may make my prediction became less accurate. Other than some minor misunderstandings, my model could also be useful for both renter and the users of Airbnb, they could use the model to price their house and the users could use to see if the house is overpriced. For Airbnb company itself, it also useful for them to offer suggested rent price for new renters, so in future, more datasets could be collected, and more features could be analyzed to better predict the rent price, the more data we used to train the model, the better prediction it will return to us, that is the magic of machine learning.

7. Appendix

1. Summary of Random Forest, and the RMSE on analysis data



```
> rmse(predTree, kg_df$price)
[1] 27.60447
```



```
                Length Class  Mode
call                3  -none- call
type                1  -none- character
predicted       41330  -none- numeric
mse               500  -none- numeric
rsq               500  -none- numeric
oob.times       41330  -none- numeric
importance         69  -none- numeric
importanceSD        0  -none- NULL
localImportance     0  -none- NULL
proximity           0  -none- NULL
ntree               1  -none- numeric
mtry                1  -none- numeric
forest             11  -none- list
coefs               0  -none- NULL
y               41330  -none- numeric
test                0  -none- NULL
inbag               0  -none- NULL
```

2. Summary of Linear Regression Model

```
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 74.31 on 41195 degrees of freedom
Multiple R-squared:  0.5539,    Adjusted R-squared:  0.5524
F-statistic: 381.7 on 134 and 41195 DF,  p-value: < 2.2e-16
```

3. The code I have for the best prediction

```
###Loading needed libraries
library(tidyverse)
library(scales)
library(Metrics)
library(plm)
library(caret)
library(caTools)
library(dplyr)
library(randomForest)
library(ranger)
library(Rborist)
library(ngram)
```

```
#########Reading Analysis Data#################
kg_df <- read.csv("analysisData.csv")

#######Cleaning Data if needed###############
kg_df$host_acceptance_rate <- gsub("N/A", 52.58017,
kg_df$host_acceptance_rate)  ##Convert NA into mean
kg_df$host_response_rate <- gsub("N/A", 66.51636,
kg_df$host_response_rate)  ##Convert NA into mean
kg_df <- kg_df %>%
  mutate(host_response_rate = parse_number(host_response_rate)) %>%
  mutate(host_response_rate = host_response_rate/100) %>%
  mutate(host_acceptance_rate = parse_number(host_acceptance_rate)) %>%
  mutate(host_acceptance_rate = host_acceptance_rate/100) %>%
  mutate(zipcode = as.numeric(zipcode))  ###Convert to manageable data
type
kg_df$zipcode[which(is.na(kg_df$zipcode))] = mean(kg_df$zipcode, na.rm =
TRUE)
kg_df$host_acceptance_rate[which(is.na(kg_df$host_acceptance_rate))] =
mean(kg_df$host_acceptance_rate, na.rm = TRUE)
kg_df$host_response_rate[which(is.na(kg_df$host_response_rate))] =
mean(kg_df$host_response_rate, na.rm = TRUE)
#####Counting Words for features like summary, description, etc########
kg_df$name<-ifelse(kg_df$name=='',0,sapply(strsplit(kg_df$name,' '),length))
kg_df$summary<-
ifelse(kg_df$summary=='',0,sapply(strsplit(kg_df$summary,' '),length)+1)
kg_df$space<-ifelse(kg_df$space=='',0,sapply(strsplit(kg_df$space,'
'),length)+1)
kg_df$neighborhood_overview<-
ifelse(kg_df$neighborhood_overview=='',0,sapply(strsplit(kg_df$neighborhood_o
verview,' '),length)+1)
kg_df$notes<-ifelse(kg_df$notes=='',0,sapply(strsplit(kg_df$notes,'
'),length)+1)
kg_df$transit<-ifelse(kg_df$transit=='',0,sapply(strsplit(kg_df$transit,'
'),length)+1)
kg_df$access<-ifelse(kg_df$access=='',0,sapply(strsplit(kg_df$access,'
'),length)+1)
kg_df$interaction<-
ifelse(kg_df$interaction=='',0,sapply(strsplit(kg_df$interaction,' '),length)+1)
kg_df$house_rules<-
ifelse(kg_df$house_rules=='',0,sapply(strsplit(kg_df$house_rules,' '),length)+1)
kg_df$description<-
ifelse(kg_df$description=='',0,sapply(strsplit(kg_df$description,' '),length)+1)
kg_df$host_about<-
```

```
ifelse(kg_df$host_about=='',0,sapply(strsplit(kg_df$host_about,' '),length)+1)

    ###########################Converting                                    city
variable#########################################################

    kg_df$city[which(kg_df$city=="纽约"|kg_df$city=="纽约市")] = "nyc"

    kg_df$city[which(kg_df$city=="纽约法拉盛"|kg_df$city=="布鲁克林")] =

"ny"
    kg_df$city<-tolower(kg_df$city)
    kg_df$city<-ifelse(str_detect(kg_df$city,"long"),"lic",kg_df$city)
    kg_df$city<-ifelse(str_detect(kg_df$city,"asto"),"asto",kg_df$city)
    kg_df$city<-ifelse(str_detect(kg_df$city,"brook"),"bk",kg_df$city)
    kg_df$city<-ifelse(str_detect(kg_df$city,"bron"),"bx",kg_df$city)
    kg_df$city<-ifelse(str_detect(kg_df$city,"ny"),"ny",kg_df$city)
    kg_df$city<-ifelse(str_detect(kg_df$city,"city"),"nyc",kg_df$city)
    kg_df$city<-ifelse(str_detect(kg_df$city,"manha"),"mah",kg_df$city)
    kg_df$city<-ifelse(str_detect(kg_df$city,"wood"),"wood",kg_df$city)
    kg_df$city<-ifelse(str_detect(kg_df$city,"flus"),"fls",kg_df$city)
    kg_df$city<-ifelse(str_detect(kg_df$city,"que"),"queens",kg_df$city)
    kg_df$city<-ifelse(str_detect(kg_df$city,"new"),"ny",kg_df$city)
    kg_df$city<-ifelse(str_detect(kg_df$city,"sou"),"sou",kg_df$city)
    kg_df$city<-ifelse(str_detect(kg_df$city," "),"others",kg_df$city)
    kg_df$city<-
ifelse((kg_df$city=="nyc"|kg_df$city=="ny"|kg_df$city=="lic"|kg_df$city=="ast
o"|

kg_df$city=="bk"|kg_df$city=="bx"|kg_df$city=="mah"|kg_df$city=="wood"|

kg_df$city=="fls"|kg_df$city=="queens"|kg_df$city=="sou"),kg_df$city,"others")
    kg_df$city<-as.factor(kg_df$city)
    ########Finding        room        age        and        other        time
variables########################

    today = Sys.time()
    kg_df$host_since <- as.Date(today) - as.Date(kg_df$host_since)
    kg_df$host_since <- as.numeric(kg_df$host_since)

    kg_df$first_review <- as.Date(today) - as.Date(kg_df$first_review)
    kg_df$first_review <- as.numeric(kg_df$first_review)

    kg_df$last_review <- as.Date(today) - as.Date(kg_df$last_review)
    kg_df$last_review <- as.numeric(kg_df$last_review)
```

```
      #######Checking if host in US##################
      a<-str_extract(kg_df$host_location, "United States")
      kg_df$host_location<-ifelse(a == "United States","t","f")
      kg_df$host_location[is.na(kg_df$host_location)] = "f"
      kg_df$host_location=as.factor(kg_df$host_location)

      ######Convert Features like Amenities and verification into numbers of
items################
      kg_df$amenities = gsub("\\.", "", kg_df$amenities)
      kg_df$amenities = kg_df$amenities %>% stringr::str_replace_all("\\s", "")
      kg_df$amenities = noquote(kg_df$amenities)
      kg_df$amenities<-nchar(gsub('[^,]+', '', gsub(',(?=,)|(^,|,$)', '',

gsub('(Null){1,}', '', kg_df$amenities), perl=TRUE)))+1L

      kg_df$host_verifications = gsub("\\.", "", kg_df$host_verifications)
      kg_df$host_verifications = kg_df$host_verifications %>%
stringr::str_replace_all("\\s", "")
      kg_df$host_verifications = noquote(kg_df$host_verifications)
      kg_df$host_verifications<-nchar(gsub('[^,]+', '', gsub(',(?=,)|(^,|,$)', '',
                                                              gsub('(Null){1,}', '',
kg_df$host_verifications), perl=TRUE)))+1L

      ######Converting Response time into levels, 1 as fastest and 4 as
lowest#########
      kg_df$host_response_time<-ifelse(kg_df$host_response_time%in%"within
an hour",1,kg_df$host_response_time)
      kg_df$host_response_time<-ifelse(kg_df$host_response_time%in%"within a
few hours",2,kg_df$host_response_time)
      kg_df$host_response_time<-ifelse(kg_df$host_response_time%in%"within a
day",3,kg_df$host_response_time)
      kg_df$host_response_time<-ifelse(kg_df$host_response_time%in%"a few
days or more",4,kg_df$host_response_time)
      kg_df$host_response_time<-
ifelse(kg_df$host_response_time%in%"N/A",4,kg_df$host_response_time)
      kg_df$host_response_time<-as.factor(kg_df$host_response_time)

      kg_df$calendar_updated<-
ifelse(str_detect(kg_df$calendar_updated,"months"),3,kg_df$calendar_updated)
      kg_df$calendar_updated<-
ifelse(str_detect(kg_df$calendar_updated,"day"),1,kg_df$calendar_updated)
      kg_df$calendar_updated<-
ifelse(str_detect(kg_df$calendar_updated,"week"),2,kg_df$calendar_updated)
      kg_df$calendar_updated<-
```

```
ifelse(str_detect(kg_df$calendar_updated,"nev"),4,kg_df$calendar_updated)
      kg_df$calendar_updated<-as.factor(kg_df$calendar_updated)

      #####################Working With neighborhood data by converting to levels###########################
      ###pricing each level of neighborhood
      neighborhood = kg_df %>%
         group_by(neighbourhood_cleansed = neighbourhood_cleansed) %>%
         summarize(levels = n(),
                        price_level = mean(price)) %>%
         arrange(desc(levels))
      ###Adding the new column for applying pricing segementation
      kg_df = merge(kg_df, neighborhood, by = c("neighbourhood_cleansed", "neighbourhood_cleansed"))
      kg_df$neighbourhood_cleansed = as.character(kg_df$neighbourhood_cleansed)
      kg_df = kg_df %>%
         mutate(new = ifelse(levels > 146, neighbourhood_cleansed, "other"))
      kg_df$neighbourhood_cleansed = as.factor(data$neighbourhood_cleansed)
      kg_df$new = as.factor(kg_df$new)

      ########Checking Data Types#####################
      kg_df$property_type = as.factor(kg_df$property_type)
      kg_df$room_type = as.factor(kg_df$room_type)
      kg_df$require_guest_phone_verification = as.factor(kg_df$require_guest_phone_verification)
      kg_df$require_guest_profile_picture = as.factor(kg_df$require_guest_profile_picture)
      kg_df$is_business_travel_ready = as.factor(kg_df$is_business_travel_ready)
      kg_df$instant_bookable = as.factor(kg_df$instant_bookable)
      kg_df$host_is_superhost = as.factor(kg_df$host_is_superhost)
      kg_df$host_neighbourhood = as.factor(kg_df$host_neighbourhood)
      kg_df$neighbourhood_group_cleansed = as.factor(kg_df$neighbourhood_group_cleansed)
      kg_df$is_location_exact = as.factor(kg_df$is_location_exact)
      kg_df$bed_type = as.factor(kg_df$bed_type)
      kg_df$host_has_profile_pic = as.factor(kg_df$host_has_profile_pic)
      kg_df$host_identity_verified = as.factor(kg_df$host_identity_verified)
      kg_df$has_availability = as.factor(kg_df$has_availability)
      kg_df$requires_license = as.factor(kg_df$requires_license)
      kg_df$market = as.factor(kg_df$market)
      kg_df$cancellation_policy = as.factor(kg_df$cancellation_policy)

      ########Fit numeric NA value with mean##########
```

```
    for (i in 1:ncol(kg_df)) {
      if (is.numeric(kg_df[,i])) {
        kg_df[is.na(kg_df[,i]), i] = mean(kg_df[,i], na.rm = TRUE)
      }}

    ################################Working          with          scoring
data#################################
    scoring<-read.csv("scoringData.csv")

    scoring$host_acceptance_rate        <-        gsub("N/A",        52.58017,
scoring$host_acceptance_rate)   ##Convert NA into mean
    scoring$host_response_rate        <-        gsub("N/A",        66.51636,
scoring$host_response_rate)   ##Convert NA into mean
    scoring <- scoring %>%
      mutate(host_response_rate = parse_number(host_response_rate)) %>%
      mutate(host_response_rate = host_response_rate/100) %>%
      mutate(host_acceptance_rate = parse_number(host_acceptance_rate)) %>%
      mutate(host_acceptance_rate = host_acceptance_rate/100) %>%
      mutate(zipcode = as.numeric(zipcode))   ###Convert to manageable data
type
    scoring$zipcode[which(is.na(scoring$zipcode))]   =   mean(scoring$zipcode,
na.rm = TRUE)
    scoring$host_acceptance_rate[which(is.na(scoring$host_acceptance_rate))] =
mean(scoring$host_acceptance_rate, na.rm = TRUE)
    scoring$host_response_rate[which(is.na(scoring$host_response_rate))]      =
mean(scoring$host_response_rate, na.rm = TRUE)
    #####Counting Words for features like summary, description, etc########
    scoring$name<-ifelse(scoring$name==",0,sapply(strsplit(scoring$name,'
'),length))
    scoring$summary<-
ifelse(scoring$summary==",0,sapply(strsplit(scoring$summary,' '),length)+1)
    scoring$space<-ifelse(scoring$space==",0,sapply(strsplit(scoring$space,'
'),length)+1)
    scoring$neighborhood_overview<-
ifelse(scoring$neighborhood_overview==",0,sapply(strsplit(scoring$neighborhoo
d_overview,' '),length)+1)
    scoring$notes<-ifelse(scoring$notes==",0,sapply(strsplit(scoring$notes,'
'),length)+1)
    scoring$transit<-ifelse(scoring$transit==",0,sapply(strsplit(scoring$transit,'
'),length)+1)
    scoring$access<-ifelse(scoring$access==",0,sapply(strsplit(scoring$access,'
'),length)+1)
    scoring$interaction<-
ifelse(scoring$interaction==",0,sapply(strsplit(scoring$interaction,' '),length)+1)
```

```
        scoring$house_rules<-
ifelse(scoring$house_rules==",0,sapply(strsplit(scoring$house_rules,' '),length)+1)
        scoring$description<-
ifelse(scoring$description==",0,sapply(strsplit(scoring$description,' '),length)+1)
        scoring$host_about<-
ifelse(scoring$host_about==",0,sapply(strsplit(scoring$host_about,' '),length)+1)
        ########Finding          room          age          and          other          time
variables########################

        today = Sys.time()
        scoring$host_since <- as.Date(today) - as.Date(scoring$host_since)
        scoring$host_since <- as.numeric(scoring$host_since)

        scoring$first_review <- as.Date(today) - as.Date(scoring$first_review)
        scoring$first_review <- as.numeric(scoring$first_review)

        scoring$last_review <- as.Date(today) - as.Date(scoring$last_review)
        scoring$last_review <- as.numeric(scoring$last_review)

        #######Checking if host in US###################
        a<-str_extract(scoring$host_location, "United States")
        scoring$host_location<-ifelse(a == "United States","t","f")
        scoring$host_location[is.na(scoring$host_location)] = "f"
        scoring$host_location=as.factor(scoring$host_location)

        ######Convert Features like Amenities and verification into numbers of
items################
        scoring$amenities = gsub("\\.", "", scoring$amenities)
        scoring$amenities = scoring$amenities %>% stringr::str_replace_all("\\s", "")
        scoring$amenities = noquote(scoring$amenities)
        scoring$amenities<-nchar(gsub('[^,]+', ", gsub(',(?=,)|(^,|,$)', ",
                                                                      gsub('(Null){1,}',
", scoring$amenities), perl=TRUE)))+1L

        scoring$host_verifications = gsub("\\.", "", scoring$host_verifications)
        scoring$host_verifications         =         scoring$host_verifications         %>%
stringr::str_replace_all("\\s", "")
        scoring$host_verifications = noquote(scoring$host_verifications)
        scoring$host_verifications<-nchar(gsub('[^,]+', ", gsub(',(?=,)|(^,|,$)', ",

gsub('(Null){1,}', ", scoring$host_verifications), perl=TRUE)))+1L

        ######Converting Response time into levels, 1 as fastest and 4 as
lowest#########
```

```
scoring$host_response_time<-ifelse(scoring$host_response_time%in%"within an hour",1,scoring$host_response_time)
scoring$host_response_time<-ifelse(scoring$host_response_time%in%"within a few hours",2,scoring$host_response_time)
scoring$host_response_time<-ifelse(scoring$host_response_time%in%"within a day",3,scoring$host_response_time)
scoring$host_response_time<-ifelse(scoring$host_response_time%in%"a few days or more",4,scoring$host_response_time)
scoring$host_response_time<-ifelse(scoring$host_response_time%in%"N/A",4,scoring$host_response_time)
scoring$host_response_time<-as.factor(scoring$host_response_time)

scoring$calendar_updated<-ifelse(str_detect(scoring$calendar_updated,"months"),3,scoring$calendar_updated)
scoring$calendar_updated<-ifelse(str_detect(scoring$calendar_updated,"day"),1,scoring$calendar_updated)
scoring$calendar_updated<-ifelse(str_detect(scoring$calendar_updated,"week"),2,scoring$calendar_updated)
scoring$calendar_updated<-ifelse(str_detect(scoring$calendar_updated,"nev"),4,scoring$calendar_updated)
scoring$calendar_updated<-as.factor(scoring$calendar_updated)

####################Working With neighborhood data by converting to levels###########################
## create a data frame to use the neighbourhood_cleansed
scoring1 = scoring %>%
  group_by(neighbourhood_cleansed = neighbourhood_cleansed) %>%
  summarize(levels = n()) %>%
  arrange(desc(levels))
## merge the data frame and get the mean price and record counts
scoring = merge(scoring, scoring1, by = c("neighbourhood_cleansed", "neighbourhood_cleansed"))
scoring$neighbourhood_cleansed = as.character(scoring$neighbourhood_cleansed)
scoring = scoring %>%
  mutate(new = ifelse(levels > 36, neighbourhood_cleansed, "other"))
scoring$new = as.factor(scoring$new)
## create price_mean_c for score_data
data2 = data.frame(neighbourhood_cleansed = neighborhood$neighbourhood_cleansed,
                   price_level = neighborhood$price_level)
```

```
scoring = merge(scoring, data2, by = c("neighbourhood_cleansed", "neighbourhood_cleansed"), all.x = TRUE)
### remember to use all.x = TRUE here to keep scoring as a whole
## check the missing value

###############################Working with city#########################################
##########################Converting city variable######################################################

scoring$city[which(scoring$city=="纽约"|scoring$city=="纽约市")] = "nyc"

scoring$city[which(scoring$city=="纽约法拉盛"|scoring$city=="布鲁克林")] = "ny"
scoring$city<-tolower(scoring$city)
scoring$city<-ifelse(str_detect(scoring$city,"long"),"lic",scoring$city)
scoring$city<-ifelse(str_detect(scoring$city,"asto"),"asto",scoring$city)
scoring$city<-ifelse(str_detect(scoring$city,"brook"),"bk",scoring$city)
scoring$city<-ifelse(str_detect(scoring$city,"bron"),"bx",scoring$city)
scoring$city<-ifelse(str_detect(scoring$city,"ny"),"ny",scoring$city)
scoring$city<-ifelse(str_detect(scoring$city,"city"),"nyc",scoring$city)
scoring$city<-ifelse(str_detect(scoring$city,"manha"),"mah",scoring$city)
scoring$city<-ifelse(str_detect(scoring$city,"wood"),"wood",scoring$city)
scoring$city<-ifelse(str_detect(scoring$city,"flus"),"fls",scoring$city)
scoring$city<-ifelse(str_detect(scoring$city,"que"),"queens",scoring$city)
scoring$city<-ifelse(str_detect(scoring$city,"new"),"ny",scoring$city)
scoring$city<-ifelse(str_detect(scoring$city,"sou"),"sou",scoring$city)
scoring$city<-ifelse(str_detect(scoring$city," "),"others",scoring$city)
scoring$city<-
ifelse((scoring$city=="nyc"|scoring$city=="ny"|scoring$city=="lic"|scoring$city=="asto"|

scoring$city=="bk"|scoring$city=="bx"|scoring$city=="mah"|scoring$city=="wood"|

scoring$city=="fls"|scoring$city=="queens"|scoring$city=="sou"),scoring$city,"others")
scoring$city<-as.factor(scoring$city)


########Checking Data Types#####################
scoring$property_type = as.factor(scoring$property_type)
scoring$room_type = as.factor(scoring$room_type)
scoring$require_guest_phone_verification =
as.factor(scoring$require_guest_phone_verification)
	scoring$require_guest_profile_picture                                  =
as.factor(scoring$require_guest_profile_picture)
	scoring$is_business_travel_ready                                       =
as.factor(scoring$is_business_travel_ready)
	scoring$instant_bookable = as.factor(scoring$instant_bookable)
	scoring$host_is_superhost = as.factor(scoring$host_is_superhost)
	scoring$host_neighbourhood = as.factor(scoring$host_neighbourhood)
	scoring$neighbourhood_group_cleansed                                   =
as.factor(scoring$neighbourhood_group_cleansed)
	scoring$is_location_exact = as.factor(scoring$is_location_exact)
	scoring$bed_type = as.factor(scoring$bed_type)
	scoring$host_has_profile_pic = as.factor(scoring$host_has_profile_pic)
	scoring$host_identity_verified = as.factor(scoring$host_identity_verified)
	scoring$has_availability = as.factor(scoring$has_availability)
	scoring$requires_license = as.factor(scoring$requires_license)
	scoring$market = as.factor(scoring$market)
	scoring$cancellation_policy = as.factor(scoring$cancellation_policy)
	########Fit numeric NA value with mean##########
	for (i in 1:ncol(scoring)) {
	  if (is.numeric(scoring[,i])) {
	    scoring[is.na(scoring[,i]), i] = mean(scoring[,i], na.rm = TRUE)
	  }}


	###############################Splitting                              Data
Set####################################
	set.seed(520)
	split = createDataPartition(y = kg_df$price,
	                            p = 0.7,
	                            list = F,
	                            groups = 100)
	train_kg = kg_df[split,]
	test_kg = kg_df[-split,]
	nrow(train_kg) + nrow(test_kg) == nrow(kg_df)

	str(train_kg)

	###########################modeling#####################
	set.seed(520)
	  boost   =   gbm(price   ~   summary   +   space   +   description   +
neighborhood_overview + notes + transit + access
	              + interaction + house_rules + host_since + host_location +
host_about + host_response_time + host_response_rate
```

```
                  + host_acceptance_rate + host_is_superhost +
host_listings_count + host_total_listings_count + host_verifications
                  + host_has_profile_pic + host_identity_verified +
neighbourhood_group_cleansed + is_location_exact + property_type
                  + room_type + accommodates + bathrooms + bedrooms +
beds + bed_type + amenities + security_deposit + cleaning_fee
                  + guests_included + extra_people + minimum_nights +
maximum_nights + minimum_minimum_nights + maximum_minimum_nights
                  + minimum_maximum_nights +
maximum_maximum_nights + minimum_nights_avg_ntm +
maximum_nights_avg_ntm
                  + availability_30 + availability_60 + availability_90 +
availability_365 + number_of_reviews_ltm + first_review + last_review
                  + review_scores_rating + review_scores_accuracy +
review_scores_checkin + review_scores_communication + review_scores_location
                  + review_scores_value + review_scores_cleanliness +
instant_bookable + require_guest_profile_picture +
require_guest_phone_verification
                  + calculated_host_listings_count +
calculated_host_listings_count_entire_homes +
calculated_host_listings_count_private_rooms +
calculated_host_listings_count_private_rooms
                  + calculated_host_listings_count_shared_rooms +
reviews_per_month + market + new + calendar_updated + cancellation_policy +
city
                   ,data = kg_df, distribution = "gaussian",
                   n.trees = 30000,
                   interaction.depth = 5,
                   shrinkage = 0.003,
                   n.minobsinnode = 5)
    summary(boost)
    predBoost = predict(boost, scoring, n.trees = 30000)

    ############################Submission!!!!#######################
######
    submissionFile = data.frame(id = scoring$id, price = predBoost)
write.csv(submissionFile, 'SubmitFinal.csv',row.names = F)
```