KAGGLE REPORT

-Shashank Chaudhary

RMSE public: 55.98079
RMSE private: 57.43220

Data exploration and visualizations

- I imported the AnalysisData file into R and ran an str() function on the file to see the various types of variables which were present in the file.
- The is.na() function helped me find the number of NA in each column and help me determine which columns should be dropped from the model. Square_feet, weekly_price and monthly_price had almost no values and had 95% of their values missing. Hence the best option was to drop it.
- Initially I decided to run my models based on only numerical data since I was not sure of how to deal with the variables which contained text and long descriptions. I dropped all the variables which included text.
- After seeing the data, I realized there was not one type of NA but there was NA, N/A and " ". I had to read the data again and needed to make sure that all types of NA were captured so that the model or predictions won't give error.
- I started plotting the numerical data to check its range and to give me an idea of the variable's average values. Doing this gave me some interesting insights.
- The variable "price" had some values with zero, which according to me would prove to worsen the dataset thus I removed these values.
- The variables maximum_night, mimimum_night, minimum_maximum_nights and minimum_nights_avg had a lot of outliers due to which predictions would be skewed. Thus, I capped them off at 6000 after plotting them and checking for outliers.

Data Cleaning and Imputing

- After checking for the NA values, I knew which variables had how many NA values. It was essential for me to decide whether to fill the NA with zeros or with median/mode values.

- Initially I filled the values with the average of the rest of the values. Then I realized that blank spaces might have meant zero cleaning fee or security deposit, thus I replaced them with zeroes. I replaced the host_listing_count and reviews_per_month with the mean values.
- I started running some basic linear regression models by now by adding all the variables which I had cleaned. By adding 2-3 variables to the model by RMSE started coming down. Due to this I got the wrong perception that more variables reflected lower RMSE.
- I started adding variables without checking if they were relevant or not after cleaning and eliminating the NA values.
- Neighbourhood_cleansed and property_type had some levels missing which were not in the Analysis_data thus I added them from the scoring_data.

<u>Data Feature Engineering</u>

- At this point I realized that dropping the variables which had text was a mistake. I removed the function which was removing the text columns.
- By counting the number of commas in the Amenities, I could count the number of amenities which were existing in the house. Thus, I extracted the number into a new variable amenities_nchar using the stringr library. This function had a function str_count which counted the number of characters (commas in this case).
- I then converted the description, name, summary, space, notes, access and interaction into numerical data by counting the number of words in the string using the nchar () function. The logic behind this was that more the number the characters in a description, it would be a good review or the other way around too, i.e. long descriptions equal bad places which meant less price. Thus, number of characters would affect the pricing. Hence a lot of new variables with number of characters were formed to be added in the model.
- I was researching about how to use zip codes effectively in a model. I tried converting them to numeric type but the problem was the number was changing which affected the model. For example, the zip code “00071” converted to numeric was only 71 which was not a correct zip code for a place. After researching on the internet, there was something called geocoding. The package “zipcode” had all the place, state, longitude and latitude of the corresponding code. Running a simple SQL code which, I found online helped me add latitude and longitude to my dataset. The code was: merge(x = data, y = zipcode, by.x = "zipcode", by.y = "zip")
- I realized the word luxury, luxurious and Luxurious in the columns name, space, description, summary and neighbourhood_overview indicated that the place was very good and highly priced. Thus, I extracted the word from the string and stored its count in a new variable. I added the count of all the luxury, Luxurious and luxurious into a new variable “luxury”. This was another new variable in my model.

- I then found another new variable in a similar way. The number of transport ways available in a locality would help me determine if the area was in a good connected neighborhood or if it was poorly connected. A good connected area would have higher prices compared to a poorly connected area. Similar to luxury, I counted the words such as bus, buses, subways and trains and stored them in new variables. I totaled all of the counts and stored them in transit_modes.

Modelling - Finding the right model

- As stated earlier, I inserted all the variables in my linear regression model thinking more variables meant a better model and lower RMSE.
- After seeing the p-values of each variable, I realized my mistake that a lot variables were not relevant and influencing the model. They were just noise and actually making the model worse.
- I wanted to do feature selection using Lasso model but couldn't troubleshoot an error thus I reverted back to using lm() model to select my relevant variables.
- I added one variable at a time and observed the R square value increased or decreases and accordingly added or removed the variables. This was I was able to filter out a lot of variables which were just plain noise.
- Using the linear regression model helped me reach a score of RMSE -65. I was stuck at this point by using linear regression.
- I tried squaring the values for co-relations but the model had overfit my numerous variables. Thus, at this point I decided to move on to random forest.
- Random forest was useful in getting my score down to around RMSE-60. But there was a problem that I couldn't select variables which had more than 53 factors, hence I think I had to drop an important predictor neighbourhood_cleansed.
- I initially put the number of trees to 100 and then increased to 150 and 200 but there was not much significant difference on the scores. It has to be noted that I had not engineered the zipcode and transit variables yet.
- Random forest took very long(about 1.5 hours) to run the model due to which I was not too excited to continue with it.
- I moved on to Gradient boosting model, which was notorious for overfitting but due to its shorter running speed I preferred this over the randomForest.
- Fine tuning this model proved quite a challenge. I started with 1000 trees and 1 interactions and shrinkage value of 0.01.
- I actually got an RMSE of 90 after this which was depressing!
- But after reading about gbm(), I increased the interactions to 8 and the shrinkage to 0.001 eventually. Because of the low shrinkage value, I had to drastically increase my trees to 30000! I did not reach to 30000 trees in one go but I reached there eventually since I could see improvements in my RMSE.

- I heard about Ranger model from a friend and tried but I couldn’t use or had time to use it properly.

Thus, my final score of RMSE = **55.98079** on public leaderboard came from the **gbm()** model, which was on the final day of Kaggle.

I think I left it for too late but I have some ideas which I could have implemented and improved my score:

- Count the number of “NO” in the House Rules columns to see how many restrictions were present in the house which could affect the prize.
- Find ways of using the bag model and fine tuning it.
- Find out more about using categorical variables effectively and increase interactions.

APPENDIX

```
getwd()
rm(list = ls())
setwd('/Users/shashank/Desktop/Frameworks/Kaggle competition')

data = read.csv('analysisData.csv', na.strings = c("NA","N/A"," "))
#analysing the data
colnames(data)
str(data)
sapply(data, class)

#counting na values in each column
colSums(is.na(data))

#converting character to numneric
data$host_listings_count = as.numeric(data$host_listings_count)
data$host_total_listings_count = as.numeric(data$host_total_listings_count)
data$beds = as.numeric(data$beds)

#converting date to days from today
data$first_review_day = as.Date(Sys.Date())-as.Date(data$first_review)
data$last_review_day = as.Date(Sys.Date())-as.Date(data$last_review)
data$host_since_day = as.Date(Sys.Date())-as.Date(data$host_since)


#counting characters
data$name_nchar<-nchar(as.character(data$name), type = 'chars')
data$description_nchar<-nchar(as.character(data$description), type = 'chars')
data$neighborhood_overview_nchar<-nchar(as.character(data$neighborhood_overview), type
= 'chars')
data$summary_nchar<-nchar(as.character(data$summary), type = 'chars')
data$space_nchar<-nchar(as.character(data$space), type = 'chars')
data$notes_nchar<-nchar(as.character(data$notes), type = 'chars')
data$access_nchar<-nchar(as.character(data$access), type = 'chars')
data$interaction_nchar<-nchar(as.character(data$interaction), type = 'chars')

#number of amenities
library(stringr)
data$amenities_nchar = str_count(data$amenities, ',')


#replacing NA with mean or mode
```

```
data$cleaning_fee[which(is.na(data$cleaning_fee))] = 0
data$security_deposit[which(is.na(data$security_deposit))] = 0
data$beds[which(is.na(data$beds))] = 0
data$host_total_listings_count[which(is.na(data$host_total_listings_count))] =
mean(data$host_total_listings_count, na.rm = TRUE)
data$host_listings_count[which(is.na(data$host_listings_count))] =
mean(data$host_listings_count, na.rm = TRUE)
data$reviews_per_month[which(is.na(data$reviews_per_month))] =
mean(data$reviews_per_month, na.rm = TRUE)
data$host_since_day[is.na(data$host_since_day)] =
mean(data$host_since_day[!is.na(data$host_since_day)])
data$description_nchar[is.na(data$description_nchar)] = 0
data$summary_nchar[is.na(data$summary_nchar)] = 0
data$notes_nchar[is.na(data$notes_nchar)] = 0
data$access_nchar[is.na(data$access_nchar)] = 0
data$interaction_nchar[is.na(data$interaction_nchar)] = 0


colSums(is.na(data))
str(data)

# OUTLIERS

# MAXIMUM_NIGHTS
mask_maximum_nights_outlier = (data$maximum_nights>6000)
data[mask_maximum_nights_outlier, 'maximum_nights'] =
median(data$maximum_nights,na.rm = T)
summary(data$maximum_nights)
#plot(data$maximum_nights)

# MINIMUM_MAXIMUM_NIGHTS
mask_minimum_maximum_nights_outlier = (data$minimum_maximum_nights>6000)
data[mask_minimum_maximum_nights_outlier, 'minimum_maximum_nights'] =
median(data$minimum_maximum_nights,na.rm = T)
summary(data$minimum_maximum_nights)
#plot(data$minimum_maximum_nights)

# MAXIMUM_MAXIMUM_NIGHTS
mask_maximum_maximum_nights_outlier = (data$maximum_maximum_nights>6000)
data[mask_maximum_maximum_nights_outlier, 'maximum_maximum_nights'] =
median(data$maximum_maximum_nights,na.rm = T)
summary(data$maximum_maximum_nights)
#plot(data$maximum_maximum_nights)
```

```
# MAXIMUM_MAXIMUM_NIGHTS
mask_maximum_nights_avg_ntm_outlier = (data$maximum_nights_avg_ntm>6000)
data[mask_maximum_nights_avg_ntm_outlier, 'maximum_nights_avg_ntm'] =
median(data$maximum_nights_avg_ntm,na.rm = T)
summary(data$maximum_nights_avg_ntm)
#plot(data$maximum_nights_avg_ntm)

#library(ggplot2)
ggplot(data=data, aes(x=price)) +
  geom_histogram(fill="blue", binwidth = 10)

table(data$price)      # 15 rows have Price = 0, which is not good data

# Remove Rows which have Price = 0, these are not good values

data = data[!data$price==0,]
table(data$price)

#add levels to factors
levels(data$neighbourhood_cleansed) <- c(levels(data$neighbourhood_cleansed), "Howland
Hook", "New Dorp", "New Dorp Beach")
levels(data$property_type) <- c(levels(data$property_type), "Casa particular (Cuba)", "Castle",
"Farm stay")


#####################################3


# Read scoring data and apply model to generate predictions
scoringData = read.csv('scoringData.csv', na.strings = c("NA","N/A"," "))



#setting levels to analysis data
scoringData$property_type = factor(scoringData$property_type, levels =
levels(data$property_type))
scoringData$neighbourhood_cleansed = factor(scoringData$neighbourhood_cleansed, levels =
levels(data$neighbourhood_cleansed))

#counting na values in each column
colSums(is.na(scoringData))

#converting cahracter to numneric
scoringData$host_listings_count = as.numeric(scoringData$host_listings_count)
scoringData$host_total_listings_count = as.numeric(scoringData$host_total_listings_count)
```

```
scoringData$beds = as.numeric(scoringData$beds)


#converting date to days from today
scoringData$first_review_day = as.Date(Sys.Date())-as.Date(scoringData$first_review)
scoringData$last_review_day = as.Date(Sys.Date())-as.Date(scoringData$last_review)
scoringData$host_since_day = as.Date(Sys.Date())-as.Date(scoringData$host_since)

#counting characters
scoringData$name_nchar = nchar(as.character(scoringData$name), type = 'chars')
scoringData$description_nchar = nchar(as.character(scoringData$description), type = 'chars')
scoringData$neighborhood_overview_nchar =
nchar(as.character(scoringData$neighborhood_overview), type = 'chars')
scoringData$summary_nchar = nchar(as.character(scoringData$summary), type = 'chars')
scoringData$space_nchar = nchar(as.character(scoringData$space), type = 'chars')
scoringData$notes_nchar = nchar(as.character(scoringData$notes), type = 'chars')
scoringData$access_nchar = nchar(as.character(scoringData$access), type = 'chars')
scoringData$interaction_nchar = nchar(as.character(scoringData$interaction), type = 'chars')

#number of amenities
library(stringr)
scoringData$amenities_nchar = str_count(scoringData$amenities, ',')


#replacing NA with mean or mode
scoringData$cleaning_fee[which(is.na(scoringData$cleaning_fee))] = 0
scoringData$security_deposit[which(is.na(scoringData$security_deposit))] = 0
scoringData$beds[which(is.na(scoringData$beds))] = 0
scoringData$host_total_listings_count[which(is.na(scoringData$host_total_listings_count))] =
mean(scoringData$host_total_listings_count, na.rm = TRUE)
scoringData$host_listings_count[which(is.na(scoringData$host_listings_count))] =
mean(scoringData$host_listings_count, na.rm = TRUE)
scoringData$reviews_per_month[which(is.na(scoringData$reviews_per_month))] =
mean(scoringData$reviews_per_month, na.rm = TRUE)
scoringData$host_since_day[is.na(scoringData$host_since_day)] =
mean(scoringData$host_since_day[!is.na(scoringData$host_since_day)])
scoringData$description_nchar[is.na(scoringData$description_nchar)] = 0
scoringData$summary_nchar[is.na(scoringData$summary_nchar)] = 0
scoringData$notes_nchar[is.na(scoringData$notes_nchar)] = 0
scoringData$access_nchar[is.na(scoringData$access_nchar)] = 0
scoringData$interaction_nchar[is.na(scoringData$interaction_nchar)] = 0
#scoringData$property_type[is.na(scoringData$property_type)] = "Apartment"
#scoringData$neighbourhood_cleansed[is.na(scoringData$neighbourhood_cleansed)] =
"Williamsburg"
```

```r
# OUTLIERS

# MAXIMUM_NIGHTS
mask_maximum_nights_outlier1 = (scoringData$maximum_nights>6000)
scoringData[mask_maximum_nights_outlier1, 'maximum_nights'] =
median(scoringData$maximum_nights,na.rm = T)
summary(data$maximum_nights)

# MINIMUM_MAXIMUM_NIGHTS
mask_minimum_maximum_nights_outlier1 = (scoringData$maximum_nights>6000)
scoringData[mask_minimum_maximum_nights_outlier1, 'minimum_maximum_nights'] =
median(scoringData$minimum_maximum_nights,na.rm = T)

# MAXIMUM_MAXIMUM_NIGHTS
mask_maximum_maximum_nights_outlier1 = (scoringData$maximum_maximum_nights>6000)
scoringData[mask_maximum_maximum_nights_outlier1, 'maximum_maximum_nights'] =
median(scoringData$maximum_maximum_nights,na.rm = T)


# MAXIMUM_MAXIMUM_NIGHTS
mask_maximum_nights_avg_ntm_outlier1 = (scoringData$maximum_nights_avg_ntm>6000)
scoringData[mask_maximum_nights_avg_ntm_outlier1, 'maximum_nights_avg_ntm'] =
median(scoringData$maximum_nights_avg_ntm,na.rm = T)


#Zipcoding
data$zipcode[data$zipcode == ""] <- "10002"
scoringData$zipcode[scoringData$zipcode == ""] <- "10002"

data$zipcode[data$zipcode == "11249\n11249"] <- "11249"
scoringData$zipcode[scoringData$zipcode == "11249\n11249"] <- "11249"

data$zipcode[data$zipcode == "11385-2308"] <- "11385"
scoringData$zipcode[scoringData$zipcode == "11385-2308"] <- "11385"

data$zipcode[data$zipcode == "11413-3220"] <- "11413"
scoringData$zipcode[scoringData$zipcode == "11413-3220"] <- "11413"

data$zipcode[data$zipcode == "11103-3233"] <- "11103"
scoringData$zipcode[scoringData$zipcode == "11103-3233"] <- "11103"

data$zipcode[data$zipcode == "111211"] <- "11221"
scoringData$zipcode[scoringData$zipcode == "111211"] <- "11221"
```

```
data$zipcode[data$zipcode == "1m"] <- "10002"
scoringData$zipcode[scoringData$zipcode == "1m"] <- "10002"
#install.packages("zipcode")
library(zipcode)
data("zipcode")
data <- merge(x = data, y = zipcode, by.x = "zipcode", by.y = "zip")
scoringData <- merge(x = scoringData, y = zipcode, by.x = "zipcode", by.y = "zip")


colSums(is.na(scoringData))
str(scoringData)

##############################

#Find Keyword
library(dplyr)
#install.packages("sqldf")
library(sqldf)
library(data.table)

# luxury
data$luxury1 = data$name %like% "luxury"
scoringData$luxury1 = scoringData$name %like% "luxury"
data$luxury1 = as.numeric(as.logical(data$luxury1))
scoringData$luxury1 = as.numeric(as.logical(scoringData$luxury1))

data$luxury2 = data$summary %like% "luxury"
scoringData$luxury2 = scoringData$summary %like% "luxury"
data$luxury2 = as.numeric(as.logical(data$luxury2))
scoringData$luxury2 = as.numeric(as.logical(scoringData$luxury2))

data$luxury3 = data$space %like% "luxury"
scoringData$luxury3 = scoringData$space %like% "luxury"
data$luxury3 = as.numeric(as.logical(data$luxury3))
scoringData$luxury3 = as.numeric(as.logical(scoringData$luxury3))

data$luxury4 = data$description %like% "luxury"
scoringData$luxury4 = scoringData$description %like% "luxury"
data$luxury4 = as.numeric(as.logical(data$luxury4))
scoringData$luxury4 = as.numeric(as.logical(scoringData$luxury4))

data$luxury5 = data$neighborhood_overview %like% "luxury"
scoringData$luxury5 = scoringData$neighborhood_overview %like% "luxury"
```

```
data$luxury5 = as.numeric(as.logical(data$luxury5))
scoringData$luxury5 = as.numeric(as.logical(scoringData$luxury5))

# Luxury
data$luxury6 = data$name %like% "Luxury"
scoringData$luxury6 = scoringData$name %like% "Luxury"
data$luxury6 = as.numeric(as.logical(data$luxury6))
scoringData$luxury6 = as.numeric(as.logical(scoringData$luxury6))

data$luxury7 = data$summary %like% "Luxury"
scoringData$luxury7 = scoringData$summary %like% "Luxury"
data$luxury7 = as.numeric(as.logical(data$luxury7))
scoringData$luxury7 = as.numeric(as.logical(scoringData$luxury7))

data$luxury8 = data$space %like% "Luxury"
scoringData$luxury8 = scoringData$space %like% "Luxury"
data$luxury8 = as.numeric(as.logical(data$luxury8))
scoringData$luxury8 = as.numeric(as.logical(scoringData$luxury8))

data$luxury9 = data$description %like% "Luxury"
scoringData$luxury9 = scoringData$description %like% "Luxury"
data$luxury9 = as.numeric(as.logical(data$luxury9))
scoringData$luxury9 = as.numeric(as.logical(scoringData$luxury9))

data$luxury10 = data$neighborhood_overview %like% "Luxury"
scoringData$luxury10 = scoringData$neighborhood_overview %like% "Luxury"
data$luxury10 = as.numeric(as.logical(data$luxury10))
scoringData$luxury10 = as.numeric(as.logical(scoringData$luxury10))

# luxurious
data$luxury11 = data$name %like% "luxurious"
scoringData$luxury11 = scoringData$name %like% "luxurious"
data$luxury11 = as.numeric(as.logical(data$luxury11))
scoringData$luxury11 = as.numeric(as.logical(scoringData$luxury11))

data$luxury12 = data$summary %like% "luxurious"
scoringData$luxury12 = scoringData$summary %like% "luxurious"
data$luxury12 = as.numeric(as.logical(data$luxury12))
scoringData$luxury12 = as.numeric(as.logical(scoringData$luxury12))

data$luxury13 = data$space %like% "luxurious"
scoringData$luxury13 = scoringData$space %like% "luxurious"
data$luxury13 = as.numeric(as.logical(data$luxury13))
scoringData$luxury13 = as.numeric(as.logical(scoringData$luxury13))
```

```
data$luxury14 = data$description %like% "luxurious"
scoringData$luxury14 = scoringData$description %like% "luxurious"
data$luxury14 = as.numeric(as.logical(data$luxury14))
scoringData$luxury14 = as.numeric(as.logical(scoringData$luxury14))

data$luxury15 = data$neighborhood_overview %like% "luxurious"
scoringData$luxury15 = scoringData$neighborhood_overview %like% "luxurious"
data$luxury15 = as.numeric(as.logical(data$luxury15))
scoringData$luxury15 = as.numeric(as.logical(scoringData$luxury15))

# Luxurious
data$luxury16 = data$name %like% "Luxurious"
scoringData$luxury16 = scoringData$name %like% "Luxurious"
data$luxury16 = as.numeric(as.logical(data$luxury16))
scoringData$luxury16 = as.numeric(as.logical(scoringData$luxury16))

data$luxury17 = data$summary %like% "Luxurious"
scoringData$luxury17 = scoringData$summary %like% "Luxurious"
data$luxury17 = as.numeric(as.logical(data$luxury17))
scoringData$luxury17 = as.numeric(as.logical(scoringData$luxury17))

data$luxury18 = data$space %like% "Luxurious"
scoringData$luxury18 = scoringData$space %like% "Luxurious"
data$luxury18 = as.numeric(as.logical(data$luxury18))
scoringData$luxury18 = as.numeric(as.logical(scoringData$luxury18))

data$luxury19 = data$description %like% "Luxurious"
scoringData$luxury19 = scoringData$description %like% "Luxurious"
data$luxury19 = as.numeric(as.logical(data$luxury19))
scoringData$luxury19 = as.numeric(as.logical(scoringData$luxury19))

data$luxury20 = data$neighborhood_overview %like% "Luxurious"
scoringData$luxury20 = scoringData$neighborhood_overview %like% "Luxurious"
data$luxury20 = as.numeric(as.logical(data$luxury20))
scoringData$luxury20 = as.numeric(as.logical(scoringData$luxury20))


data$luxury = (data$luxury1 + data$luxury2 + data$luxury3 + data$luxury4 + data$luxury5 +
data$luxury6 + data$luxury7 + data$luxury8 + data$luxury9 + data$luxury10
        + data$luxury11 + data$luxury12 + data$luxury13 + data$luxury14 + data$luxury15 +
data$luxury16 + data$luxury17 + data$luxury18 + data$luxury19
        + data$luxury20)
```

```
scoringData$luxury = (scoringData$luxury1 + scoringData$luxury2 + scoringData$luxury3 +
scoringData$luxury4 + scoringData$luxury5 + scoringData$luxury6
                + scoringData$luxury7 + scoringData$luxury8 + scoringData$luxury9 +
scoringData$luxury10 + scoringData$luxury11 + scoringData$luxury12
                + scoringData$luxury13 + scoringData$luxury14 + scoringData$luxury15 +
scoringData$luxury16 + scoringData$luxury17 + scoringData$luxury18
                + scoringData$luxury19 + scoringData$luxury20)


#transit

data$transit_bus = data$transit %like% "bus"
scoringData$transit_bus = scoringData$name %like% "bus"
data$transit_bus = as.numeric(as.logical(data$transit_bus))
scoringData$transit_bus = as.numeric(as.logical(scoringData$transit_bus))

data$transit_bus1 = data$transit %like% "buses"
scoringData$transit_bus1 = scoringData$name %like% "bus"
data$transit_bus1 = as.numeric(as.logical(data$transit_bus1))
scoringData$transit_bus1 = as.numeric(as.logical(scoringData$transit_bus1))

data$transit_train = data$transit %like% "train"
scoringData$transit_train = scoringData$name %like% "train"
data$transit_train = as.numeric(as.logical(data$transit_train))
scoringData$transit_train = as.numeric(as.logical(scoringData$transit_train))

data$transit_subway = data$transit %like% "subway"
scoringData$transit_subway = scoringData$name %like% "subway"
data$transit_subway = as.numeric(as.logical(data$transit_subway))
scoringData$transit_subway = as.numeric(as.logical(scoringData$transit_subway))

data$transit_modes = data$transit_bus + data$transit_subway + data$transit_train
scoringData$transit_modes = scoringData$transit_bus + scoringData$transit_subway +
scoringData$transit_train


#Linear model

model1 = lm(price~
          host_total_listings_count
        + host_is_superhost
        + host_has_profile_pic
#         + host_identity_verified
        + is_location_exact
```

```
        + property_type
        + neighbourhood_group_cleansed
        + room_type
        + accommodates
        + security_deposit
        + cleaning_fee
        + guests_included
        + extra_people
        + minimum_nights
        + maximum_maximum_nights
        + minimum_maximum_nights
        + maximum_nights
        + review_scores_rating
        + bathrooms
        + bedrooms
        + beds
        + bed_type
        + availability_30
        + availability_60
        + availability_90
        + availability_365
        + neighbourhood_cleansed
        + number_of_reviews
        + number_of_reviews_ltm
        + review_scores_accuracy
        + review_scores_cleanliness
        + review_scores_checkin
#        + review_scores_communication
#         + review_scores_location
        + review_scores_value
#         + reviews_per_month
#         + instant_bookable
        + cancellation_policy
#         + require_guest_profile_picture
#         + require_guest_phone_verification
        + calculated_host_listings_count
        + calculated_host_listings_count_entire_homes
        + calculated_host_listings_count_private_rooms
        +  name_nchar
#         + description_nchar
#         +neighborhood_overview_nchar
        + summary_nchar
        + space_nchar
#        + notes_nchar
```

```
#           + access_nchar
#          + interaction_nchar
          + amenities_nchar
          + first_review_day
          + last_review_day
          + latitude
          + longitude
          + luxury
          + transit_modes
#          + host_since_day

,data)
summary(model1)

levels(data$property_type)
levels(data$neighbourhood_cleansed)
levels(scoringData$neighbourhood_cleansed)

#Random forest model

library(randomForest)
set.seed(617)
model2 = randomForest(price~ host_total_listings_count
                  + host_is_superhost
                  + host_has_profile_pic
                  #           + host_identity_verified
                  + is_location_exact
                  + property_type
                  + neighbourhood_group_cleansed
                  + room_type
                  + accommodates
                  + security_deposit
                  + cleaning_fee
                  + guests_included
                  + extra_people
                  + minimum_nights
                  + maximum_nights
                  + review_scores_rating
                  + bathrooms
                  + bedrooms
                  + beds
                  + bed_type
                  + availability_30
                  + availability_60
```

```
                 + availability_90
                 + availability_365
                 # + neighbourhood_cleansed
                 + number_of_reviews
                 + number_of_reviews_ltm
                 + review_scores_accuracy
                 + review_scores_cleanliness
                 + review_scores_checkin
                 #          + review_scores_communication
                 #          + review_scores_location
                + review_scores_value
                 #          + reviews_per_month
                 #          + instant_bookable
                #  + cancellation_policy
                 #          + require_guest_profile_picture
                 #          + require_guest_phone_verification
                + calculated_host_listings_count
                + calculated_host_listings_count_entire_homes
                + calculated_host_listings_count_private_rooms
                +  name_nchar
                #          + description_nchar
                #          +neighborhood_overview_nchar
                + summary_nchar
                + space_nchar
                #         + notes_nchar
                #          + access_nchar
                #         + interaction_nchar
                + amenities_nchar
                + first_review_day
                + last_review_day
                #          + host_since_day
                 +latitude
                +longitude
                + maximum_maximum_nights
                + minimum_maximum_nights
                + luxury
                + transit_modes
                   ,data=data,ntree = 150, na.action=na.exclude)

pred = predict(model2,newdata=scoringData)
rmseForest = sqrt(mean((pred-data$price)^2)); rmseForest
table(is.na(pred))
```

```
#boosting
library(gbm)
set.seed(617)
model3 = gbm(price~ host_total_listings_count
            + host_is_superhost
            + host_has_profile_pic
            + host_identity_verified
            + is_location_exact
            + property_type
            + neighbourhood_group_cleansed
            + room_type
            + accommodates
            + security_deposit
            + cleaning_fee
            + guests_included
            + extra_people
            + minimum_nights
            + maximum_nights
            + review_scores_rating
            + bathrooms
            + bedrooms
            + beds
            + bed_type
            + availability_30
            + availability_60
            + availability_90
            + availability_365
             + neighbourhood_cleansed
            + number_of_reviews
            + number_of_reviews_ltm
            + review_scores_accuracy
            + review_scores_cleanliness
            + review_scores_checkin
            #           + review_scores_communication
            #           + review_scores_location
            + review_scores_value
                        + reviews_per_month
            #           + instant_bookable
              + cancellation_policy
                      + require_guest_profile_picture
            #           + require_guest_phone_verification
            + calculated_host_listings_count
            + calculated_host_listings_count_entire_homes
            + calculated_host_listings_count_private_rooms
```

```
          +  name_nchar
          #          + description_nchar
          #          +neighborhood_overview_nchar
          + summary_nchar
          + space_nchar
          #         + notes_nchar
                  + access_nchar
          #         + interaction_nchar
          + amenities_nchar
         # + first_review_day
         # + last_review_day
          #          + host_since_day
         +latitude
         +longitude
         + luxury
         + transit_modes
          , data=data, distribution="gaussian", n.trees = 30000,interaction.depth = 8,shrinkage =
0.001)
predBoostTrain = predict(model3,n.trees = 30000)
rmseBoostTrain = sqrt(mean((predBoostTrain-data$price)^2)); rmseBoostTrain

pred = predict(model3,newdata=scoringData, n.trees = 30000)
rmseBoostTest = sqrt(mean((pred-data$price)^2)); rmseBoostTest


table(is.na(pred))


#ranger
install.packages("ranger")
library(ranger)
ranger_rf = ranger(formula   = price~host_total_listings_count
               + host_is_superhost
               + host_has_profile_pic
               + host_identity_verified
               + is_location_exact
               + property_type
               + neighbourhood_group_cleansed
               + room_type
               + accommodates
               + security_deposit
               + cleaning_fee
               + guests_included
               + extra_people
```

```
+ minimum_nights
+ maximum_nights
+ review_scores_rating
+ bathrooms
+ bedrooms
+ beds
+ bed_type
+ availability_30
+ availability_60
+ availability_90
+ availability_365
+ neighbourhood_cleansed
+ number_of_reviews
+ number_of_reviews_ltm
+ review_scores_accuracy
+ review_scores_cleanliness
+ review_scores_checkin
#         + review_scores_communication
#          + review_scores_location
+ review_scores_value
+ reviews_per_month
#          + instant_bookable
+ cancellation_policy
+ require_guest_profile_picture
#          + require_guest_phone_verification
+ calculated_host_listings_count
+ calculated_host_listings_count_entire_homes
+ calculated_host_listings_count_private_rooms
+  name_nchar
#          + description_nchar
#          +neighborhood_overview_nchar
+ summary_nchar
+ space_nchar
#         + notes_nchar
+ access_nchar
#         + interaction_nchar
+ amenities_nchar
# + first_review_day
# + last_review_day
          + host_since_daye,
data      = data,num.trees = 500,write.forest = TRUE)
```

```
# Construct submission from predictions
submissionFile = data.frame(id = scoringData$id, price = pred)
write.csv(submissionFile, 'boost_submission.csv',row.names = F)


str(data)
```