Group Project BUS AN 514: Analytics for Marketing Decisions

Aakanksha Baid, Isha Malte Irene Sung

2/23/2020

# Introduction

Founded in 2008, Airbnb is a homestay marketplace that allows home-owners or hosts to list their properties, or listings online, so that guests can pay to stay in them. Though Airbnb gives general giudelines to hosts for prices based on the listing type, rooms, accomodation etc. hosts usually set their own prices for their listings. A detailed pricing study is important and relevant for New York which is Airbnb’s biggest market in USA and faces lot of competition from hotels.

This analysis aims at providing Airbnb and hosts some data driven factors that can help them set the right prices for their properties by using machine learning models to predict the price for properties in New York.

# Business Objectives

* Help hosts in better understanding of price for their property based on multiple parameters
* Compete with nearby hotels by offering attractive price

# Project Objective

* Provide price prediction as a foundation model for Airbnb competitive pricing and supporting host in price setting
* Learn and apply data processing and model learnt in the course

# Dataset

The dataset used for this project comes from [here](https://public.opendatasoft.com/explore/dataset/airbnb-ratings/export/?disjunctive.city&disjunctive.neighbourhood_cleansed&sort=number_of_reviews&dataChart=eyJxdWVyaWVzIjpbeyJjb25maWciOnsiZGF0YXNldCI6ImFpcmJuYi1yYXRpbmdzIiwib3B0aW9ucyI6eyJkaXNqdW5jdGl2ZS5jaXR5Ijp0cnVlLCJkaXNqdW5jdGl2ZS5uZWlnaGJvdXJob29kX2NsZWFuc2VkIjp0cnVlfX0sImNoYXJ0cyI6W3siYWxpZ25Nb250aCI6dHJ1ZSwidHlwZSI6ImxpbmUiLCJmdW5jIjoiQVZHIiwieUF4aXMiOiJob3N0X2xpc3RpbmdzX2NvdW50Iiwic2NpZW50aWZpY0Rpc3BsYXkiOnRydWUsImNvbG9yIjoiI0ZGNTE1QSJ9XSwieEF4aXMiOiJsYXN0X3JldmlldyIsIm1heHBvaW50cyI6IiIsInRpbWVzY2FsZSI6InllYXIiLCJzb3J0IjoiIn1dLCJkaXNwbGF5TGVnZW5kIjp0cnVlLCJhbGlnbk1vbnRoIjp0cnVlfQ%3D%3D&refine.last_scraped=2017&refine.state=NY).

* Airbnb NY Listing data across 2017
* Original dataset: 44,308 observations, 94 variables
* Cleaned dataset:43,760 observations 180 variables (including dummy)
* Some of the more important features this project will investigate are the following:
  + *Price*: nightly price for the rental
  + *cleaning\_fee*: one-time fee charged by hosts to cover the cost of cleaning their property when guests depart
  + *number\_of\_reviews*: number of reviews that previous guests have left
  + *accommodates*: the number of guests the rental can accommodate
  + *bedrooms*: number of bedrooms included in the rental
  + *bathrooms*: number of bathrooms included in the rental
  + *beds*: number of beds included in the rental
  + *minimum\_nights*: minimum number of nights a guest can stay for the rental
  + *maximum\_nights*: maximum number of nights a guest can stay for the rental

# Exploratory Analysis:

## Read in the data

airbnb <- read.csv("airbnb-ratings.csv", sep=";")

## Explore the data

library(psych)  
describe(airbnb$Price)

## vars n mean sd median trimmed mad min max range skew kurtosis  
## X1 1 44065 137.96 109.02 103 118.86 66.72 0 999 999 2.78 11.55  
## se  
## X1 0.52

nrow(airbnb)

## [1] 44308

ncol(airbnb)

## [1] 94

There are 44308 observations in the data. The mean daily price is $137.96 while median daily price is lower at $103.

## Filter only relevant variables

#Create Id for unique listings  
airbnb$Id <- seq(44308)  
airbnb\_filter <- airbnb[c(2,3,20,21,23,24,26,29:30,32:36,39:43,46:67,69:72,74:83,87:93,95)]

## Check for missing values

# colSums(is.na(airbnb\_filter))

The variables “Host.is.superhost”,“Host.listings.count”, “Host.has.profile.pic”, “Host.identity.verified”,“Bathrooms”, “Bedrooms”,“Beds”,“Square.feet,”Price“,”Weekly.price“,”Monthly.price“,”Security.deposit“,”Cleaning.fee“,”Review.scores.rating“,”Review.scores.accuracy“,”Review.scores.cleanliness“,”Review.scores.checkin“,”Review.scores.communication“,”Review.scores.location“,”Review.scores.value“,”Is.business.travel.ready“,”Reviews.per.month" have mising values in the New York data.

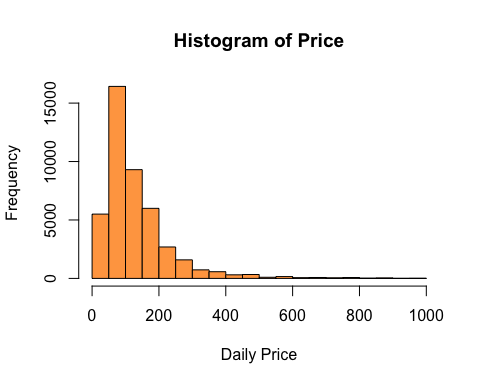
Top missing value variables are -“Square.feet” and “Monthly.price” have almost 99% missing values, “Weekly.price” has 86% missing values. So, we have dropped these.

## Keep variables with missing values less than threshold

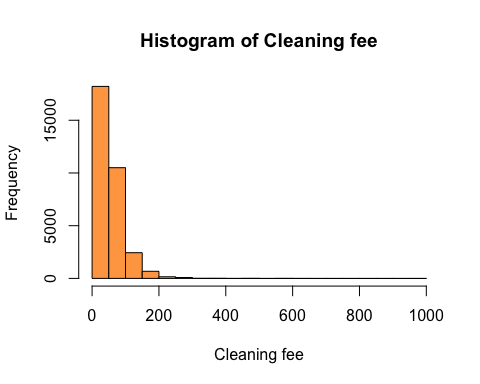
threshold = 0.4  
airbnb\_nmiss <- airbnb\_filter[colSums(is.na(airbnb\_filter))/nrow(airbnb\_filter) < threshold]  
  
# 4 variables deleted. The new data has 59 variables with less than 40% missing values.

## Histograms

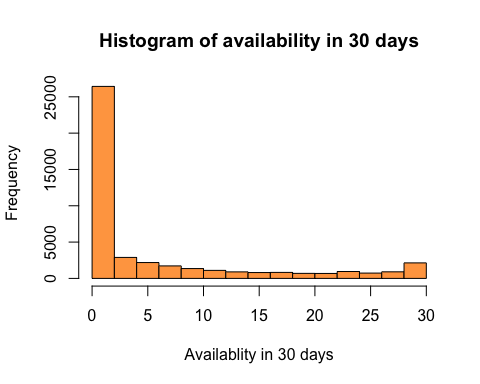
hist(airbnb\_nmiss$Price,main ="Histogram of Price",xlab = "Daily Price",col = "tan1")



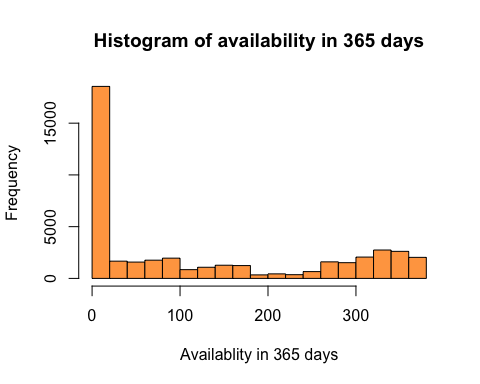
hist(airbnb\_nmiss$Cleaning.fee,main ="Histogram of Cleaning fee",xlab = "Cleaning fee",col = "tan1")



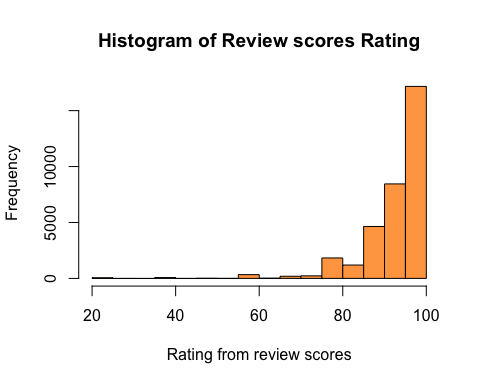
hist(airbnb\_nmiss$Availability.30,main ="Histogram of availability in 30 days",xlab = "Availablity in 30 days",col = "tan1")



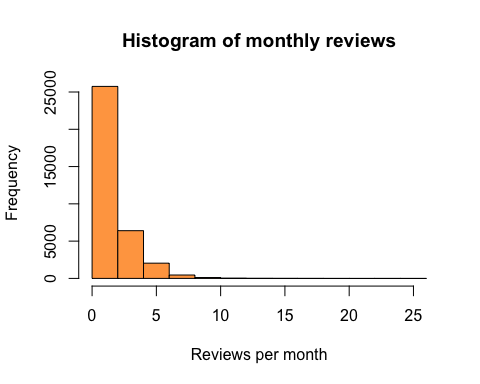
hist(airbnb\_nmiss$Availability.365,main ="Histogram of availability in 365 days",xlab = "Availablity in 365 days",col = "tan1")



hist(airbnb\_nmiss$Review.scores.rating,main ="Histogram of Review scores Rating",xlab = "Rating from review scores",col = "tan1")



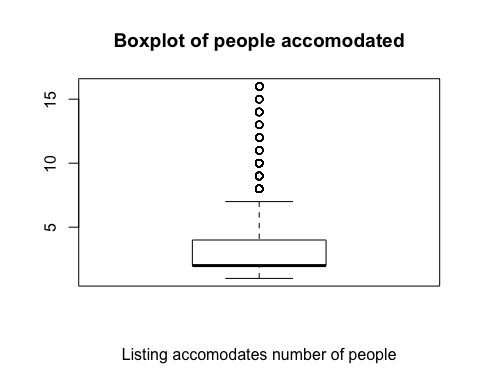
hist(airbnb\_nmiss$Reviews.per.month,main ="Histogram of monthly reviews",xlab = "Reviews per month",col = "tan1")



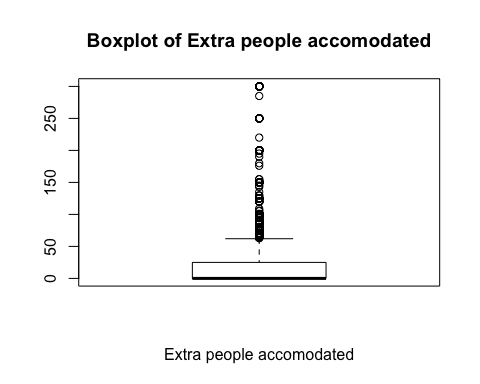
Price, Cleaning fee, Availability in 30 and 365 days and Monthly reviews are positively skewed. Review rating score is negatively skewed.

## Boxplots

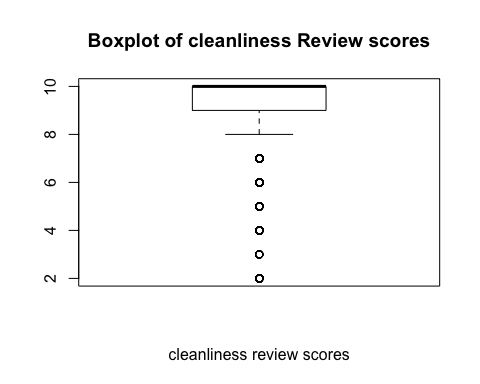
boxplot(airbnb\_nmiss$Accommodates,main ="Boxplot of people accomodated",xlab = "Listing accomodates number of people")



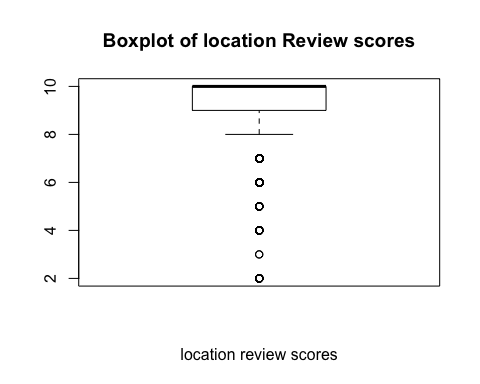
boxplot(airbnb\_nmiss$Extra.people,main ="Boxplot of Extra people accomodated",xlab = "Extra people accomodated")



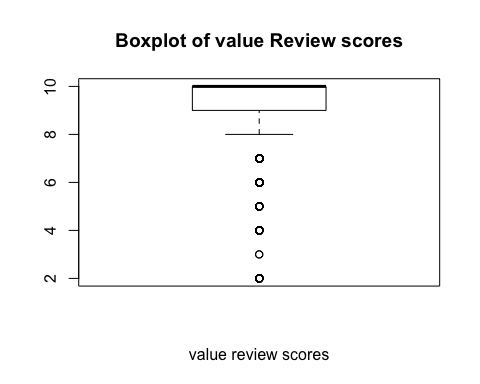
boxplot(airbnb\_nmiss$Review.scores.cleanliness,main ="Boxplot of cleanliness Review scores",xlab = "cleanliness review scores")



boxplot(airbnb\_nmiss$Review.scores.location,main ="Boxplot of location Review scores",xlab = "location review scores")



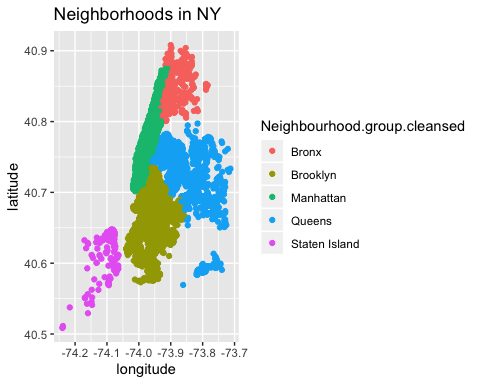
boxplot(airbnb\_nmiss$Review.scores.value,main ="Boxplot of value Review scores",xlab = "value review scores")



There are outliers in all variables.

## Neighboorhood plot

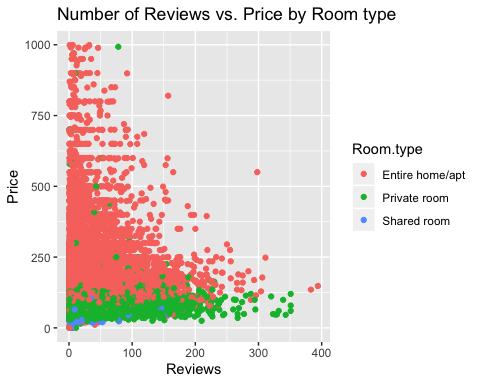
ggplot(na.omit(airbnb\_nmiss), mapping = aes(x = longitude, y = latitude, color = Neighbourhood.group.cleansed)) + geom\_point() +  
 labs(title = "Neighborhoods in NY")



Maximum listings are located in Manhattan and Brooklyn.

## Bivariate distribution

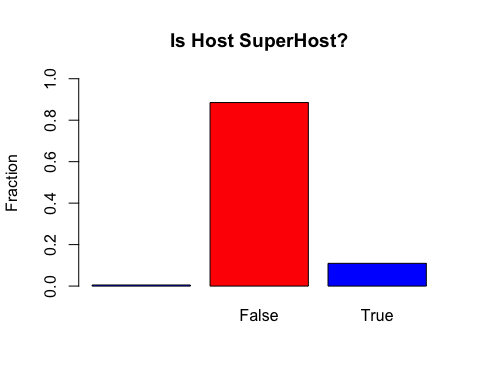
ggplot(na.omit(airbnb\_nmiss)) +  
 geom\_point(mapping = aes(x=Number.of.reviews,y=Price,color = Room.type)) +  
 labs(title = "Number of Reviews vs. Price by Room type",  
 x = "Reviews", y = "Price")



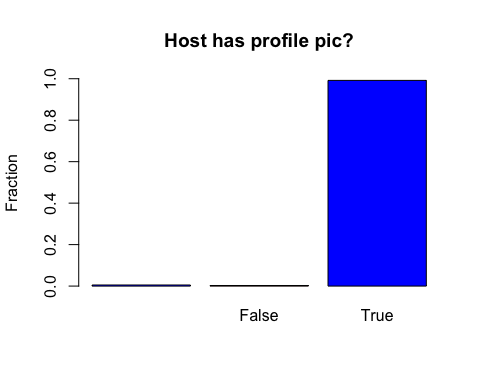
As price increases, number of reviews decreases for each room type.

## Plots

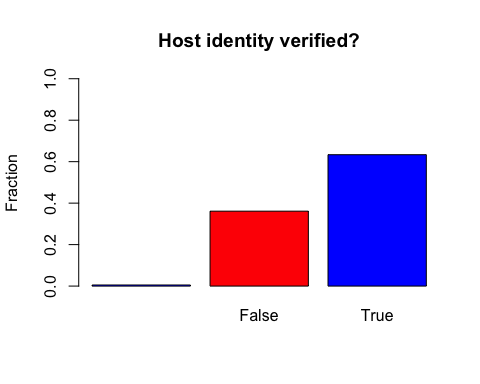
barplot(table(airbnb\_nmiss$Host.is.superhost)/length(airbnb\_nmiss$Host.is.superhost),col = c("blue","red"), main = "Is Host SuperHost?",ylim=c(0,1),ylab = "Fraction")



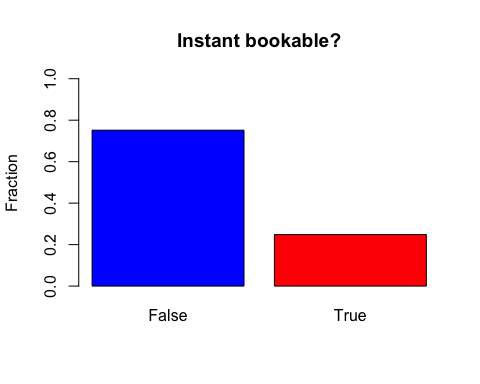
barplot(table(airbnb\_nmiss$Host.has.profile.pic)/length(airbnb\_nmiss$Host.has.profile.pic),col = c("blue","red"), main = "Host has profile pic?",ylim=c(0,1),ylab = "Fraction")



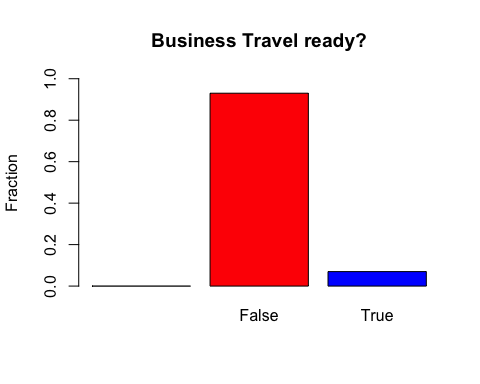
barplot(table(airbnb\_nmiss$Host.identity.verified)/length(airbnb\_nmiss$Host.identity.verified),col = c("blue","red"), main = "Host identity verified?",ylim=c(0,1),ylab = "Fraction")



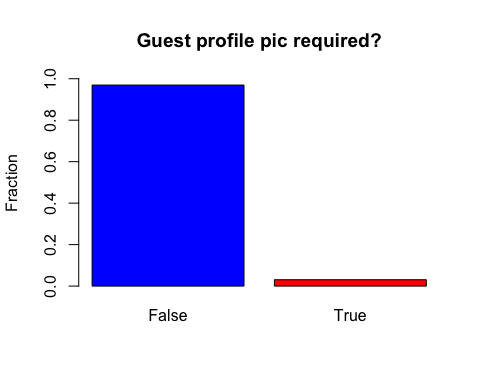
barplot(table(airbnb\_nmiss$Instant.bookable)/length(airbnb\_nmiss$Instant.bookable),col = c("blue","red"), main = "Instant bookable?",ylim=c(0,1),ylab = "Fraction")



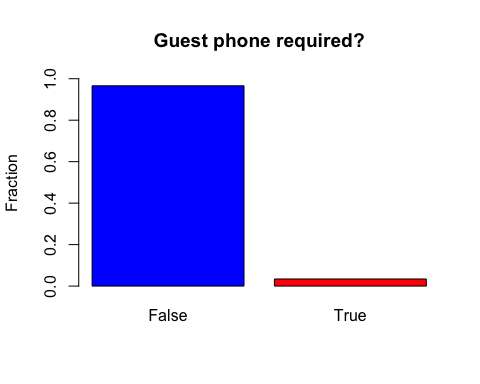
barplot(table(airbnb\_nmiss$Is.business.travel.ready)/length(airbnb\_nmiss$Is.business.travel.ready),col = c("blue","red"), main = "Business Travel ready?",ylim=c(0,1),ylab = "Fraction")



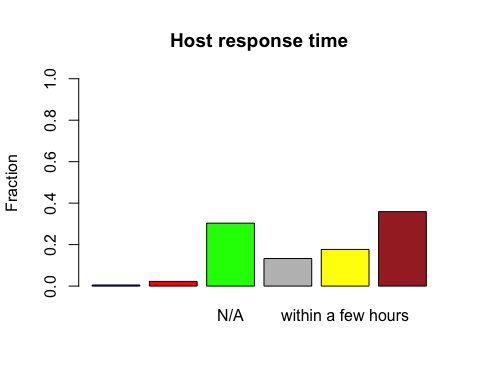
barplot(table(airbnb\_nmiss$Require.guest.profile.picture)/length(airbnb\_nmiss$Require.guest.profile.picture),col = c("blue","red"), main = "Guest profile pic required?",ylim=c(0,1),ylab = "Fraction")



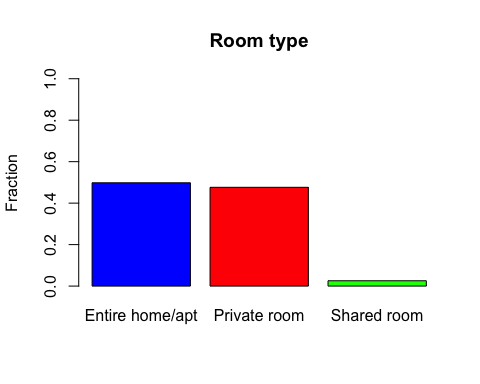
barplot(table(airbnb\_nmiss$Require.guest.phone.verification)/length(airbnb\_nmiss$Require.guest.phone.verification),col = c("blue","red"), main = "Guest phone required?",ylim=c(0,1),ylab = "Fraction")



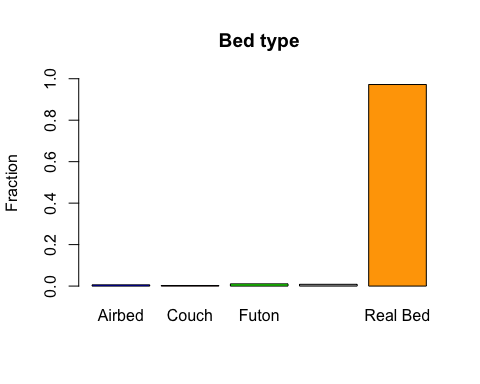
barplot(table(airbnb\_nmiss$Host.response.time)/length(airbnb\_nmiss$Host.response.time),col = c("blue","red","green","grey","yellow","brown"), main = "Host response time",ylim=c(0,1),ylab = "Fraction")



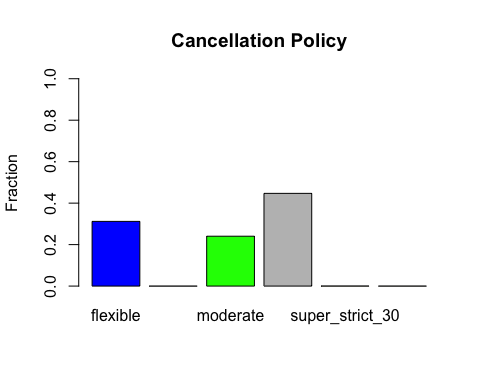
barplot(table(airbnb\_nmiss$Room.type)/length(airbnb\_nmiss$Room.type),col = c("blue","red","green"), main = "Room type",ylim=c(0,1),ylab = "Fraction")



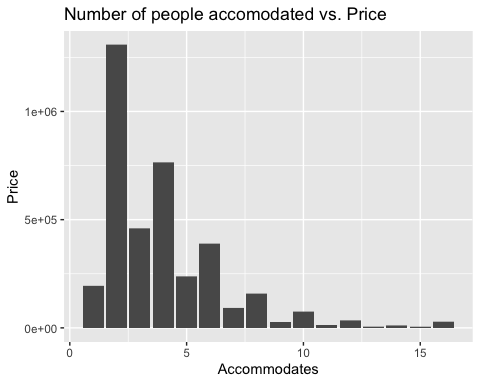
barplot(table(airbnb\_nmiss$Bed.type)/length(airbnb\_nmiss$Bed.type),col = c("blue","red","green","grey","orange"), main = "Bed type",ylim=c(0,1),ylab = "Fraction")



barplot(table(airbnb\_nmiss$Cancellation.policy)/length(airbnb\_nmiss$Cancellation.policy),col = c("blue","red","green","grey","yellow","orange"), main = "Cancellation Policy",ylim=c(0,1),ylab = "Fraction")



ggplot(na.omit(airbnb\_nmiss),aes(Accommodates,Price)) +  
 labs(title = "Number of people accomodated vs. Price")+ geom\_col()



Around less than 90% hosts in New York are not super hosts. So, we hypothesize that this will not be an important predictor for price.

Since over 95% hosts have profile pic, this variable doesn’t add lot of value and can be dropped. Similarly, guest profile pic variable can also be dropped. Other variables like phone verification from guests, business travel ready and bed type having majorly real bed can be dropped -

Host has profile pic, Require guest profile picture, Require guest phone verification, Is.business.travel.ready and Bed type are dropped from data. Also, using calculated host listings count instead of host listings count as we found from online data research, that it is a more accurate representation of host’s listings.

# D. In-depth analysis Part 1 - Data pre-processing

library(dplyr)  
airbnb1 <- airbnb\_nmiss[c(-4,-8,-9, -10,-11,-16,-17,-19:-22,-29,-37,-55,-56)]  
#The data now has 44 variables

## Drop rows which have low missing data across columns

library(tidyr)  
airbnb1 <- airbnb1 %>% drop\_na(c('Price','Host.is.superhost', 'Host.identity.verified', 'Bathrooms', 'Bedrooms', 'Beds','Is.business.travel.ready'))  
# The data reduces to 44 variables

## Missing value treatment

#Replace NA of Review parameters by median of that column  
  
airbnb1$Review.scores.rating<- replace(airbnb1$Review.scores.rating,is.na(airbnb1$Review.scores.rating),median(airbnb1$Review.scores.rating, na.rm = TRUE))  
#describe(airbnb1$Review.scores.rating)  
  
airbnb1$Review.scores.accuracy<- replace(airbnb1$Review.scores.accuracy,is.na(airbnb1$Review.scores.accuracy),median(airbnb1$Review.scores.accuracy, na.rm = TRUE))  
#describe(airbnb1$Review.scores.accuracy)  
  
airbnb1$Review.scores.cleanliness<- replace(airbnb1$Review.scores.cleanliness,is.na(airbnb1$Review.scores.cleanliness),median(airbnb1$Review.scores.cleanliness, na.rm = TRUE))  
#describe(airbnb1$Review.scores.cleanliness)  
  
airbnb1$Review.scores.checkin<- replace(airbnb1$Review.scores.checkin,is.na(airbnb1$Review.scores.checkin),median(airbnb1$Review.scores.checkin, na.rm = TRUE))  
#describe(airbnb1$Review.scores.checkin)  
  
airbnb1$Review.scores.communication<- replace(airbnb1$Review.scores.communication,is.na(airbnb1$Review.scores.communication),median(airbnb1$Review.scores.communication, na.rm = TRUE))  
#describe(airbnb1$Review.scores.communication)  
  
airbnb1$Review.scores.location<- replace(airbnb1$Review.scores.location,is.na(airbnb1$Review.scores.location),median(airbnb1$Review.scores.location, na.rm = TRUE))  
#describe(airbnb1$Review.scores.location)  
  
airbnb1$Review.scores.value<- replace(airbnb1$Review.scores.value,is.na(airbnb1$Review.scores.value),median(airbnb1$Review.scores.value, na.rm = TRUE))  
#describe(airbnb1$Review.scores.value)  
  
airbnb1$Reviews.per.month<- replace(airbnb1$Reviews.per.month,is.na(airbnb1$Reviews.per.month),median(airbnb1$Reviews.per.month, na.rm = TRUE))

## Cleaning fee missing value treatment by neighborhood group

median\_cf <- airbnb1 %>% group\_by(Neighbourhood.group.cleansed) %>% summarise(median\_fee=median(Cleaning.fee, na.rm = TRUE))  
str(median\_cf)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 6 obs. of 2 variables:  
## $ Neighbourhood.group.cleansed: Factor w/ 6 levels "","Bronx","Brooklyn",..: 1 2 3 4 5 6  
## $ median\_fee : num 20 25 50 60 30 30

airbnb1$Cleaning.fee<- ifelse(is.na(airbnb1$Cleaning.fee),as.numeric(median\_cf$median\_fee),airbnb1$Cleaning.fee)

## Handling string variable - “Amenities”

# Check what's included in the column of 'amenitiies'  
airbnb1$Amenities %>% head(1)

## [1] {"Cable TV",Internet,"Wireless Internet",Kitchen,"Free parking on premises","Buzzer/wireless intercom",Heating,"Smoke detector","Carbon monoxide detector","First aid kit","Safety card","Fire extinguisher",Essentials,Shampoo,"Lock on bedroom door",Hangers,"Hair dryer",Iron,"Laptop friendly workspace","translation missing: en.hosting\_amenity\_49","translation missing: en.hosting\_amenity\_50","Private living room","Private entrance","Hot water","Bed linens","Extra pillows and blankets",Microwave,Refrigerator,"Dishes and silverware","Cooking basics",Stove,"Garden or backyard","Luggage dropoff allowed","Long term stays allowed","Cleaning before checkout"}  
## 39714 Levels: {"Air conditioning","Buzzer/wireless intercom",Heating,"Smoke detector",Essentials,Shampoo,Hangers,"Hair dryer",Iron} ...

#It's a long list of strings that require us to convert each ammenity to one column (dummy)

# Get all unique amenities across all listings  
level\_index <- 0  
all\_amenities <- vector()  
for (level in levels(airbnb1$Amenities))   
 {  
 level\_index <- level\_index + 1  
 #if (level\_index %% 1000 == 0)   
 # {  
 #print(paste("In level", level\_index))  
 #}  
 cleanstr <- gsub("[][{}\"]", "", level)  
 items <- strsplit(cleanstr, ",")[[1]]  
 for (item in items)   
 {  
 all\_amenities <- append(all\_amenities, item)  
 }  
 all\_amenities <- unique(all\_amenities)  
 }  
all\_amenities <- sort(all\_amenities)  
print(paste("Number of total amenities", length(all\_amenities)))

## [1] "Number of total amenities 117"

# For each listing, compute an amenity one-hot vector.  
# Return a list of all listings' amenity one-hot vector.  
amentity\_vec\_of\_all\_listings <- list()  
i <- 0  
for (amenities\_of\_listing in airbnb1$Amenities)   
 {  
 i <- i + 1  
 #if (i %% 100 == 0)   
 #{  
 # print(i)  
 # }  
 amenstr <- gsub("[][{}\"]", "", amenities\_of\_listing)  
 amenity\_items <- strsplit(amenstr, ",")[[1]]  
 amentity\_vec\_of\_listing <- vector()  
 for (amen in all\_amenities)   
 {  
 if (amen %in% amenity\_items)   
 {  
 amentity\_vec\_of\_listing <- append(amentity\_vec\_of\_listing, 1)  
 } else   
 {  
 amentity\_vec\_of\_listing <- append(amentity\_vec\_of\_listing, 0)  
 }  
 }  
 amentity\_vec\_of\_all\_listings[[i]] <- amentity\_vec\_of\_listing  
}  
length(amentity\_vec\_of\_all\_listings)

## [1] 43809

## Convert each unique ammenity to a dummy

# dummy for all amenities  
# Create a new table with amenities columns and values  
amen\_dummy\_table <- do.call(rbind, amentity\_vec\_of\_all\_listings)  
colnames(amen\_dummy\_table) <- all\_amenities

# Check amenities counts  
amen\_count <- colSums(amen\_dummy\_table[])  
#amen\_count  
#names(amen\_count[order(amen\_count,decreasing=TRUE)])  
  
# combine new dummy amenities with original table and make a new one called 'ny\_copy'  
ny\_copy <- cbind(airbnb1, amen\_dummy\_table)

Top 5 amenities in NY listings are Wireless Internet, Kitchen, Heating, Essentials and Air conditioning

## Create dummy for categorical variables

ny\_copy1 <- ny\_copy # make a copy in case any issue would ruin the original one  
  
dummy\_Neighbourhood.group.cleansed <- dummy\_cols(ny\_copy1$Neighbourhood.group.cleansed, ignore\_na = TRUE)  
colnames(dummy\_Neighbourhood.group.cleansed)[1] <- c('New\_Neighbourhood.group')  
colnames(dummy\_Neighbourhood.group.cleansed)[2] <- c('New\_Neighbourhood.group\_missing')  
colnames(dummy\_Neighbourhood.group.cleansed)[3] <- c('Neighbourhood\_Bronx')  
colnames(dummy\_Neighbourhood.group.cleansed)[4] <- c('Neighbourhood\_Brooklyn')  
colnames(dummy\_Neighbourhood.group.cleansed)[5] <- c('Neighbourhood\_Manhattan')  
colnames(dummy\_Neighbourhood.group.cleansed)[6] <- c('Neighbourhood\_Queens')  
colnames(dummy\_Neighbourhood.group.cleansed)[7] <- c('Neighbourhood\_StatenIsland')  
colnames(dummy\_Neighbourhood.group.cleansed)

## [1] "New\_Neighbourhood.group" "New\_Neighbourhood.group\_missing"  
## [3] "Neighbourhood\_Bronx" "Neighbourhood\_Brooklyn"   
## [5] "Neighbourhood\_Manhattan" "Neighbourhood\_Queens"   
## [7] "Neighbourhood\_StatenIsland"

ny\_copy1<- cbind(ny\_copy1, dummy\_Neighbourhood.group.cleansed)

# create dummy for "Host.is.superhost"  
  
dummy\_superhost <- ny\_copy1$Host.is.superhost %>% dummy\_cols(ignore\_na = TRUE)  
colnames(dummy\_superhost)[1] <- c('New\_superhost')  
colnames(dummy\_superhost)[2] <- c('New\_superhost\_missing')  
colnames(dummy\_superhost)[3] <- c('New\_superhost\_F')  
colnames(dummy\_superhost)[4] <- c('New\_superhost\_T')  
colnames(dummy\_superhost)

## [1] "New\_superhost" "New\_superhost\_missing" "New\_superhost\_F"   
## [4] "New\_superhost\_T"

ny\_copy1 <- cbind(ny\_copy1, dummy\_superhost)

# create dummy for property type  
  
dummy\_propety.type <- ny\_copy1$Property.type %>% dummy\_cols(ignore\_na = TRUE)  
colnames(dummy\_propety.type)[1] <-c('New\_Property.type')  
colnames(dummy\_propety.type)[2] <-c('Property.type\_Apartment')  
colnames(dummy\_propety.type)[3] <-c('Property.type\_Bed&Breakfast')  
colnames(dummy\_propety.type)[4] <-c('Property.type\_Boat')  
colnames(dummy\_propety.type)[5] <-c('Property.type\_BoutiqueHotel')  
colnames(dummy\_propety.type)[6] <-c('Property.type\_Bungalow')  
colnames(dummy\_propety.type)[7] <-c('Property.type\_Cabin')  
colnames(dummy\_propety.type)[8] <-c('Property.type\_Castle')  
colnames(dummy\_propety.type)[9] <-c('Property.type\_Cave')  
colnames(dummy\_propety.type)[10] <-c('Property.type\_Chalet')  
colnames(dummy\_propety.type)[11] <-c('Property.type\_Condominium')  
colnames(dummy\_propety.type)[12] <-c('Property.type\_Dorm')  
colnames(dummy\_propety.type)[13] <-c('Property.type\_EarthHouse')  
colnames(dummy\_propety.type)[14] <-c('Property.type\_GuestSuite')  
colnames(dummy\_propety.type)[15] <-c('Property.type\_GuestHouse')  
colnames(dummy\_propety.type)[16] <-c('Property.type\_Hostel')  
colnames(dummy\_propety.type)[17] <-c('Property.type\_House')  
colnames(dummy\_propety.type)[18] <-c('Property.type\_In-law')  
colnames(dummy\_propety.type)[19] <-c('Property.type\_Loft')  
colnames(dummy\_propety.type)[20] <-c('Property.type\_Other')  
colnames(dummy\_propety.type)[21] <-c('Property.type\_ServicedApartment')  
colnames(dummy\_propety.type)[22] <-c('Property.type\_Tent')  
colnames(dummy\_propety.type)[23] <-c('Property.type\_Timeshare')  
colnames(dummy\_propety.type)[24] <-c('Property.type\_Townhouse')  
colnames(dummy\_propety.type)[25] <-c('Property.type\_Train')  
colnames(dummy\_propety.type)[26] <-c('Property.type\_Treehouse')  
colnames(dummy\_propety.type)[27] <-c('Property.type\_VacationHome')  
colnames(dummy\_propety.type)[28] <-c('Property.type\_Villa')  
colnames(dummy\_propety.type)[29] <-c('Property.type\_Yurt')  
colnames(dummy\_propety.type)

## [1] "New\_Property.type" "Property.type\_Apartment"   
## [3] "Property.type\_Bed&Breakfast" "Property.type\_Boat"   
## [5] "Property.type\_BoutiqueHotel" "Property.type\_Bungalow"   
## [7] "Property.type\_Cabin" "Property.type\_Castle"   
## [9] "Property.type\_Cave" "Property.type\_Chalet"   
## [11] "Property.type\_Condominium" "Property.type\_Dorm"   
## [13] "Property.type\_EarthHouse" "Property.type\_GuestSuite"   
## [15] "Property.type\_GuestHouse" "Property.type\_Hostel"   
## [17] "Property.type\_House" "Property.type\_In-law"   
## [19] "Property.type\_Loft" "Property.type\_Other"   
## [21] "Property.type\_ServicedApartment" "Property.type\_Tent"   
## [23] "Property.type\_Timeshare" "Property.type\_Townhouse"   
## [25] "Property.type\_Train" "Property.type\_Treehouse"   
## [27] "Property.type\_VacationHome" "Property.type\_Villa"   
## [29] "Property.type\_Yurt"

ny\_copy1 <- cbind(ny\_copy1, dummy\_propety.type)

# create dummy for room type  
  
dummy\_room.type <- ny\_copy1$Room.type %>% dummy\_cols(ignore\_na = TRUE)  
colnames(dummy\_room.type)[1] <-c('New\_Room.type')  
colnames(dummy\_room.type)[2] <-c('Room.type\_EntireHome/apt')  
colnames(dummy\_room.type)[3] <-c('Room.type\_PrivateRroom')  
colnames(dummy\_room.type)[4] <-c('Room.type\_SharedRroom')  
colnames(dummy\_room.type)

## [1] "New\_Room.type" "Room.type\_EntireHome/apt"  
## [3] "Room.type\_PrivateRroom" "Room.type\_SharedRroom"

ny\_copy1 <- cbind(ny\_copy1, dummy\_room.type)

ny\_copy1$Cancellation.policy %>% summary()

## flexible long\_term moderate strict super\_strict\_30   
## 13639 1 10582 19573 13   
## super\_strict\_60   
## 1

# create a new cancellation policy (flexible to strict- scale 1 to 4)  
ny\_copy1 <- ny\_copy1 %>% mutate(New\_cancellation.policy =   
 case\_when(Cancellation.policy == "flexible" ~ 1,  
 Cancellation.policy == "moderate" ~ 2,  
 Cancellation.policy =="long\_term"~ 4, # group this one with super strict  
 Cancellation.policy =="strict"~ 3,  
 Cancellation.policy == "super\_strict\_30"~ 4,  
 Cancellation.policy == "super\_strict\_60" ~ 4))  
  
ny\_copy1 %>% group\_by(New\_cancellation.policy) %>% count()

## # A tibble: 4 x 2  
## # Groups: New\_cancellation.policy [4]  
## New\_cancellation.policy n  
## <dbl> <int>  
## 1 1 13639  
## 2 2 10582  
## 3 3 19573  
## 4 4 15

#Instant bookable  
ny\_copy1$Instant.bookable%>% summary()

## False True   
## 32903 10906

# create dummy for instant bookable  
dummy\_Instant.bookable <- ny\_copy1$Instant.bookable %>% dummy\_cols(ignore\_na = TRUE)  
colnames(dummy\_Instant.bookable)[1] <-c('New\_Instant.bookable')  
colnames(dummy\_Instant.bookable)[2] <-c('Instant.bookable\_F')  
colnames(dummy\_Instant.bookable)[3] <-c('Instant.bookable\_T')  
colnames(dummy\_Instant.bookable)

## [1] "New\_Instant.bookable" "Instant.bookable\_F" "Instant.bookable\_T"

ny\_copy1 <- cbind(ny\_copy1, dummy\_Instant.bookable)

# Calculate host's duration from "host.since" & "last.scraped" in years  
ny\_copy1 <- ny\_copy1 %>% mutate(host\_duration = as.numeric(format(as.Date(ny\_copy1$Last.scraped),'%Y'))-as.numeric(format(as.Date(ny\_copy1$Host.since),'%Y')))  
  
# Fill NA  
ny\_copy1$host\_duration<- replace(ny\_copy1$host\_duration,is.na(ny\_copy1$host\_duration),median(ny\_copy1$host\_duration, na.rm = TRUE))  
unique(ny\_copy1$host\_duration)

## [1] 3 6 4 2 7 1 0 5 8 9

#Check the result  
ny\_copy1 %>% group\_by(host\_duration) %>% count()

## # A tibble: 10 x 2  
## # Groups: host\_duration [10]  
## host\_duration n  
## <dbl> <int>  
## 1 0 3970  
## 2 1 6789  
## 3 2 8652  
## 4 3 8268  
## 5 4 6360  
## 6 5 5168  
## 7 6 3041  
## 8 7 1158  
## 9 8 366  
## 10 9 37

# Convert host.response.rate into numeric variable and replace missing values with median response rate  
ny\_copy1$Host.response.rate <- gsub("[][!#$%()\*,.:;<=>@^\_`|~.{}]", "", ny\_copy1$Host.response.rate) %>% as.numeric()

## Warning in function\_list[[k]](value): NAs introduced by coercion

ny\_copy1$Host.response.rate<- replace(ny\_copy1$Host.response.rate,is.na(ny\_copy1$Host.response.rate),median(ny\_copy1$Host.response.rate, na.rm = TRUE))  
describe(ny\_copy1$Host.response.rate)

## vars n mean sd median trimmed mad min max range skew kurtosis se  
## X1 1 43809 95.56 14.65 100 99.41 0 0 100 100 -4.54 22.83 0.07

# Cleaned data after pre-prosessing

# Select the columns we need   
ny\_cleaned <- ny\_copy1 %>% select(-1:-4,-6:-13,-18, -30:-31, -39:-40,-41,-44,-147,-148,-162, -163,-169, -170,-171,-173, -202,-207,-208)  
  
#Checking data values, found price with 0 values in 48 listings, so deleting those listings  
ny\_cleaned1 <- ny\_cleaned[ny\_cleaned$Price != 0, ]  
  
# check reduced to 180 columns   
ncol(ny\_cleaned1)

## [1] 180

# Data reduced to 43760 rows  
nrow(ny\_cleaned1)

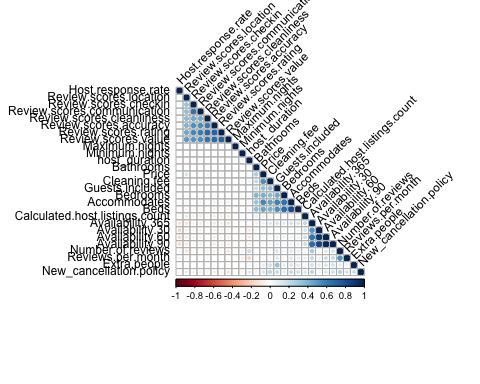
## [1] 43760

# Check the Correlations

library(corrplot)

## corrplot 0.84 loaded

par(cex = 0.75)  
  
## text labels rotated 45 degrees  
corrplot(cor(ny\_cleaned1[,c(1:25,178,180)]), type = "lower", order = "hclust", tl.col = "black", tl.srt = 45)



Review scores, Availability and property variables are highly positively correlated

# In-depth analysis Part 2 - Modeling and performance comparison

## Model 1: Linear regression model

# log transformation of skewed variables  
ny\_cleaned1$Price <- log(ny\_cleaned1$Price)  
ny\_cleaned1$Calculated.host.listings.count <- log(ny\_cleaned1$Calculated.host.listings.count)  
ny\_cleaned1$Accommodates <- log(ny\_cleaned1$Accommodates)  
ny\_cleaned1$Guests.included <- log(ny\_cleaned1$Guests.included)  
ny\_cleaned1$Minimum.nights <- log(ny\_cleaned1$Minimum.nights)  
ny\_cleaned1$Maximum.nights <- log(ny\_cleaned1$Maximum.nights)  
ny\_cleaned1$Review.scores.rating <- log(ny\_cleaned1$Review.scores.rating)  
ny\_cleaned1$Review.scores.accuracy <- log(ny\_cleaned1$Review.scores.accuracy)  
ny\_cleaned1$Review.scores.cleanliness <- log(ny\_cleaned1$Review.scores.cleanliness)  
ny\_cleaned1$Review.scores.checkin <- log(ny\_cleaned1$Review.scores.checkin)  
ny\_cleaned1$Review.scores.communication <- log(ny\_cleaned1$Review.scores.communication)  
ny\_cleaned1$Review.scores.location <- log(ny\_cleaned1$Review.scores.location)  
ny\_cleaned1$Review.scores.value <- log(ny\_cleaned1$Review.scores.value)  
ny\_cleaned1$Reviews.per.month<- log(ny\_cleaned1$Reviews.per.month)  
  
  
# Scaling and store the result as a new copy "ny\_cleaned2"   
ny\_cleaned2 <- scale(ny\_cleaned1)  
ny\_cleaned2 <- as.data.frame(ny\_cleaned2)

## First build a regression model with all the variables

# remove three columns that have NA errors (Flat -Property.type\_Yurt -Room.type\_SharedRroom)  
all\_var <- lm(Price ~ .-Flat -Property.type\_Yurt -Room.type\_SharedRroom , data = ny\_cleaned2)

## Identify multilinearity

#Any variable with a high VIF value (above 5 or 10) should be removed from the model. This leads to a simpler model without compromising the model accuracy, which is good.  
# Identify multilinearity > 10 = there are 31 variables being identified  
which(vif(all\_var) > 10) %>% print()

## Availability.60 Availability.90   
## 12 13   
## `Cooking basics` `Dishes and silverware`   
## 50 53   
## Oven Refrigerator   
## 100 110   
## Stove Neighbourhood\_Bronx   
## 123 139   
## Neighbourhood\_Brooklyn Neighbourhood\_Manhattan   
## 140 141   
## Neighbourhood\_Queens Neighbourhood\_StatenIsland   
## 142 143   
## Property.type\_Apartment `Property.type\_Bed&Breakfast`   
## 145 146   
## Property.type\_BoutiqueHotel Property.type\_Bungalow   
## 148 149   
## Property.type\_Condominium Property.type\_Dorm   
## 154 155   
## Property.type\_GuestSuite Property.type\_GuestHouse   
## 157 158   
## Property.type\_Hostel Property.type\_House   
## 159 160   
## Property.type\_Loft Property.type\_Other   
## 162 163   
## Property.type\_ServicedApartment Property.type\_Timeshare   
## 164 166   
## Property.type\_Townhouse Property.type\_VacationHome   
## 167 170   
## Property.type\_Villa `Room.type\_EntireHome/apt`   
## 171 172   
## Room.type\_PrivateRroom   
## 173

## Build a new regression

# Removed variables with VIF >10  
var\_after\_selection <- lm(Price ~ .- Availability.60 - Availability.90 - `Cooking basics`-`Dishes and silverware` -Oven - Refrigerator - Stove - Neighbourhood\_Bronx - Neighbourhood\_Brooklyn- Neighbourhood\_Manhattan- Neighbourhood\_Queens -Neighbourhood\_StatenIsland -Property.type\_Apartment -`Property.type\_Bed&Breakfast` - Property.type\_BoutiqueHotel -Property.type\_Bungalow - Property.type\_Condominium-Property.type\_Dorm - Property.type\_GuestSuite -Property.type\_GuestHouse -Property.type\_Hostel -Property.type\_House-Property.type\_Loft -Property.type\_Other -Property.type\_ServicedApartment-Property.type\_Timeshare - Property.type\_Townhouse -Property.type\_VacationHome- Property.type\_Villa -`Room.type\_EntireHome/apt` - Room.type\_PrivateRroom, data = ny\_cleaned2)

# Further remove variables that are less irrelant to this analysis or statistically insignificant  
var\_after\_selection1 <- lm(Price ~ .- Availability.60 - Availability.90 - `Cooking basics`-`Dishes and silverware` -Oven - Refrigerator - Stove - Neighbourhood\_Bronx - Neighbourhood\_Brooklyn- Neighbourhood\_Manhattan- Neighbourhood\_Queens -Neighbourhood\_StatenIsland -Property.type\_Apartment -`Property.type\_Bed&Breakfast` - Property.type\_BoutiqueHotel -Property.type\_Bungalow - Property.type\_Condominium-Property.type\_Dorm - Property.type\_GuestSuite -Property.type\_GuestHouse -Property.type\_Hostel -Property.type\_House-Property.type\_Loft -Property.type\_Other-Property.type\_ServicedApartment-Property.type\_Timeshare - Property.type\_Townhouse -Property.type\_VacationHome- Property.type\_Villa -`Room.type\_EntireHome/apt` - Room.type\_PrivateRroom - Beds -Maximum.nights -Availability.365 - Review.scores.accuracy - Review.scores.checkin- Review.scores.communication - Reviews.per.month - ` smooth pathway to front door` -`24-hour check-in`-`Accessible-height bed` -`Accessible-height toilet`-`Baby bath` -`Babysitter recommendations`- Bathtub -`BBQ grill` -`Beach essentials`-`Bed linens` -Breakfast -`Carbon monoxide detector`- `Changing table` - `Cat(s)` -`Children’s dinnerware`-`Cleaning before checkout`-`Coffee maker` - `Disabled parking spot` - `Dog(s)`- `Elevator in building` - `Ethernet connection` -`EV charger`-`Extra pillows and blankets` -`Fire extinguisher` -`Fireplace guards` -`Firm mattress` -`Fixed grab bars for shower & toilet` -`Game console`-`Ground floor access`-`Handheld shower head`-Heating -`High chair`-`Hot water`-`Hot water kettle` -`Long term stays allowed`-Other - `Other pet(s)` -`Patio or balcony`-`Pocket wifi` -`Roll-in shower with chair`- `Room-darkening shades` -`Single level home`- `Ski in/Ski out`- `Smoke detector` - `Stair gates`-`Step-free access` -`Table corner guards`-Property.type\_Yurt-Property.type\_Treehouse - Property.type\_Train -Property.type\_Train-`Property.type\_In-law`- Property.type\_EarthHouse -Property.type\_Chalet-Property.type\_Cave -Property.type\_Cabin- Property.type\_Boat-`Wide hallway clearance`-`Wide entryway` -`Wide doorway` -`Wide clearance to shower & toilet`-`Wide clearance to bed` -`Well-lit path to entrance`-Waterfront-`Washer / Dryer`-Washer -Property.type\_Castle -Property.type\_Tent -`Smart lock`-`Self Check-In`-Lockbox -Flat -`Air purifier` -`Bathtub with shower chair`-`Children’s books and toys`-`Baby monitor`-`Laptop friendly workspace`-Guests.included -Extra.people,data = ny\_cleaned2)

# Check the VIF for each variable   
# Now all the variables have VIF <10   
which(vif(var\_after\_selection1) > 10) %>% print()

## named integer(0)

# Create training and testing sets

## Final model: Linear Regression

# Finally, we built the linear regresion model by using train data  
var\_train <- lm(Price ~ .- Availability.60 - Availability.90 - `Cooking basics`-`Dishes and silverware` -Oven - Refrigerator - Stove - Neighbourhood\_Bronx - Neighbourhood\_Brooklyn- Neighbourhood\_Manhattan- Neighbourhood\_Queens -Neighbourhood\_StatenIsland -Property.type\_Apartment -`Property.type\_Bed&Breakfast` - Property.type\_BoutiqueHotel -Property.type\_Bungalow - Property.type\_Condominium-Property.type\_Dorm - Property.type\_GuestSuite -Property.type\_GuestHouse -Property.type\_Hostel -Property.type\_House-Property.type\_Loft -Property.type\_Other-Property.type\_ServicedApartment-Property.type\_Timeshare - Property.type\_Townhouse -Property.type\_VacationHome- Property.type\_Villa -`Room.type\_EntireHome/apt` - Room.type\_PrivateRroom - Beds -Maximum.nights -Availability.365 - Review.scores.accuracy - Review.scores.checkin- Review.scores.communication - Reviews.per.month - ` smooth pathway to front door` -`24-hour check-in`-`Accessible-height bed` -`Accessible-height toilet`-`Baby bath` -`Babysitter recommendations`- Bathtub -`BBQ grill` -`Beach essentials`-`Bed linens` -Breakfast -`Carbon monoxide detector`- `Changing table` - `Cat(s)` -`Children’s dinnerware`-`Cleaning before checkout`-`Coffee maker` - `Disabled parking spot` - `Dog(s)`- `Elevator in building` - `Ethernet connection` -`EV charger`-`Extra pillows and blankets` -`Fire extinguisher` -`Fireplace guards` -`Firm mattress` -`Fixed grab bars for shower & toilet` -`Game console`-`Ground floor access`-`Handheld shower head`-Heating -`High chair`-`Hot water`-`Hot water kettle` -`Long term stays allowed`-Other - `Other pet(s)` -`Patio or balcony`-`Pocket wifi` -`Roll-in shower with chair`- `Room-darkening shades` -`Single level home`- `Ski in/Ski out`- `Smoke detector` - `Stair gates`-`Step-free access` -`Table corner guards`-Property.type\_Yurt-Property.type\_Treehouse - Property.type\_Train -Property.type\_Train-`Property.type\_In-law`- Property.type\_EarthHouse -Property.type\_Chalet-Property.type\_Cave -Property.type\_Cabin- Property.type\_Boat-`Wide hallway clearance`-`Wide entryway` -`Wide doorway` -`Wide clearance to shower & toilet`-`Wide clearance to bed` -`Well-lit path to entrance`-Waterfront-`Washer / Dryer`-Washer -Property.type\_Castle -Property.type\_Tent -`Smart lock`-`Self Check-In`-Lockbox -Flat -`Air purifier` -`Bathtub with shower chair`-`Children’s books and toys`-`Baby monitor`-`Laptop friendly workspace`-Guests.included -Extra.people ,data = nyTrain1)  
  
# summary(var\_train)

install.packages("knitr")  
install.packages("broom")

# load the library for kable  
library(broom)  
library(knitr)

## Convert linear regression model result to kable

var\_train\_tidy <- tidy(var\_train)  
kable(var\_train\_tidy, digit = 3)

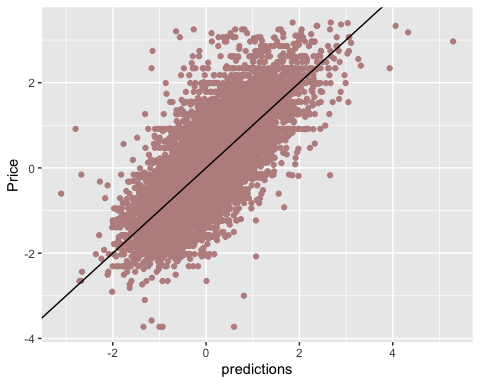
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | 0.002 | 0.004 | 0.640 | 0.522 |
| Host.response.rate | -0.006 | 0.004 | -1.651 | 0.099 |
| Accommodates | 0.370 | 0.005 | 69.509 | 0.000 |
| Bathrooms | 0.017 | 0.004 | 3.899 | 0.000 |
| Bedrooms | 0.043 | 0.005 | 8.847 | 0.000 |
| Cleaning.fee | 0.241 | 0.005 | 52.094 | 0.000 |
| Minimum.nights | -0.043 | 0.004 | -10.646 | 0.000 |
| Availability.30 | 0.059 | 0.004 | 14.318 | 0.000 |
| Number.of.reviews | -0.041 | 0.005 | -9.039 | 0.000 |
| Review.scores.rating | 0.019 | 0.006 | 2.896 | 0.004 |
| Review.scores.cleanliness | 0.046 | 0.006 | 8.019 | 0.000 |
| Review.scores.location | 0.157 | 0.005 | 34.433 | 0.000 |
| Review.scores.value | -0.100 | 0.006 | -15.790 | 0.000 |
| Calculated.host.listings.count | -0.106 | 0.004 | -25.071 | 0.000 |
| Air conditioning | 0.063 | 0.004 | 15.257 | 0.000 |
| Beachfront | -0.011 | 0.004 | -2.745 | 0.006 |
| Buzzer/wireless intercom | 0.023 | 0.004 | 5.225 | 0.000 |
| Cable TV | 0.050 | 0.004 | 11.376 | 0.000 |
| Crib | 0.008 | 0.004 | 1.989 | 0.047 |
| Dishwasher | 0.030 | 0.005 | 6.141 | 0.000 |
| Doorman | 0.070 | 0.005 | 15.612 | 0.000 |
| Dryer | 0.015 | 0.004 | 3.436 | 0.001 |
| Elevator | 0.056 | 0.005 | 11.703 | 0.000 |
| Essentials | -0.009 | 0.004 | -2.226 | 0.026 |
| Family/kid friendly | 0.034 | 0.004 | 7.645 | 0.000 |
| First aid kit | -0.035 | 0.004 | -7.988 | 0.000 |
| Free parking on premises | -0.060 | 0.004 | -15.137 | 0.000 |
| Free parking on street | -0.012 | 0.004 | -3.210 | 0.001 |
| Garden or backyard | -0.012 | 0.004 | -2.913 | 0.004 |
| Gym | 0.047 | 0.004 | 10.755 | 0.000 |
| Hair dryer | 0.042 | 0.005 | 8.379 | 0.000 |
| Hangers | -0.031 | 0.005 | -6.371 | 0.000 |
| Host greets you | -0.011 | 0.004 | -2.783 | 0.005 |
| Hot tub | -0.012 | 0.004 | -3.043 | 0.002 |
| Indoor fireplace | 0.041 | 0.004 | 10.636 | 0.000 |
| Internet | -0.024 | 0.005 | -5.251 | 0.000 |
| Iron | -0.015 | 0.005 | -2.962 | 0.003 |
| Keypad | -0.023 | 0.004 | -5.882 | 0.000 |
| Kitchen | -0.009 | 0.004 | -2.164 | 0.030 |
| Lake access | -0.007 | 0.004 | -1.996 | 0.046 |
| Lock on bedroom door | -0.072 | 0.004 | -17.263 | 0.000 |
| Luggage dropoff allowed | 0.009 | 0.004 | 1.988 | 0.047 |
| Microwave | -0.023 | 0.005 | -4.290 | 0.000 |
| Outlet covers | -0.012 | 0.004 | -2.877 | 0.004 |
| Pack ’n Play/travel crib | 0.010 | 0.004 | 2.441 | 0.015 |
| Pets allowed | -0.004 | 0.004 | -0.921 | 0.357 |
| Pets live on this property | -0.053 | 0.004 | -13.490 | 0.000 |
| Pool | 0.012 | 0.004 | 2.991 | 0.003 |
| Private bathroom | 0.000 | 0.004 | -0.097 | 0.923 |
| Private entrance | 0.007 | 0.004 | 1.655 | 0.098 |
| Private living room | -0.022 | 0.004 | -5.554 | 0.000 |
| Safety card | 0.018 | 0.004 | 4.190 | 0.000 |
| Shampoo | 0.039 | 0.004 | 8.985 | 0.000 |
| Smoking allowed | -0.015 | 0.004 | -3.843 | 0.000 |
| Suitable for events | 0.015 | 0.004 | 3.755 | 0.000 |
| TV | 0.076 | 0.004 | 17.670 | 0.000 |
| Wheelchair accessible | -0.012 | 0.004 | -2.758 | 0.006 |
| Window guards | -0.022 | 0.004 | -5.337 | 0.000 |
| Wireless Internet | -0.030 | 0.004 | -7.353 | 0.000 |
| New\_superhost\_T | 0.000 | 0.004 | -0.104 | 0.917 |
| Room.type\_SharedRroom | -0.099 | 0.004 | -25.477 | 0.000 |
| New\_cancellation.policy | 0.002 | 0.004 | 0.483 | 0.629 |
| Instant.bookable\_T | -0.018 | 0.004 | -4.314 | 0.000 |
| host\_duration | 0.052 | 0.004 | 12.336 | 0.000 |

# Linear Regression - prediction

# predict on test data  
nyTest1$predictions = predict(var\_train, newdata = nyTest1)  
  
# check the result  
head(nyTest1$predictions)

## [1] -0.07743908 -0.62293026 0.29643673 -0.71076446 -1.27994162 -0.14258481

# Graphically compare the predictions to actual price. Put predictions on the x axis and see how close are the results to the line of perfect prediction  
   
ggplot(nyTest1, aes(x = predictions, y = Price)) +   
 geom\_point(color = "rosybrown") +  
 geom\_abline(color = "black")



# RMSE for linear regression

# Install Metrics package for RMSE evaluation  
# install.packages("Metrics")  
library(Metrics)

# RMSE for our linear regression model  
rmse(nyTest1$predictions, nyTest1$Price)

## [1] 0.6590071

# Because we took log to our dependent variable, if we want to make a comparsion with differnt model, we revert it using exponetial function, another RMSE is as below:  
rmse(exp(nyTest1$predictions), exp(nyTest1$Price))

## [1] 2.55171

### Inference Linear regression:

* The RMSE: Log price: 0.66 (actual price: 2.55)
* They are 41 variables that are statistically significant in affecting the listing price. The top variables that drive up the listing price are: Accommodates, Cleaning.fee, Review.scores.location, and the top variables that drive down the listing price are: Review.scores.value, Calculated.host.listings, and Room.type\_SharedRroom.

# Model 2: CART

## Create Training, validation and Testing Dataset from the filled dataset without log or scaled values (ny\_cartxgb.df)

library(caTools)  
  
set.seed(88)  
split = sample.split(ny\_cartxgb.df$Price, SplitRatio = 0.70)  
nyTrain = subset(ny\_cartxgb.df, split==TRUE)  
nytemp = subset(ny\_cartxgb.df, split==FALSE)  
split1 = sample.split(nytemp$Price, SplitRatio = 0.50)  
nyTest = subset(nytemp, split1==FALSE)  
nyVal = subset(nytemp, split1==TRUE)

install.packages("rpart")  
install.packages("rpart.plot")

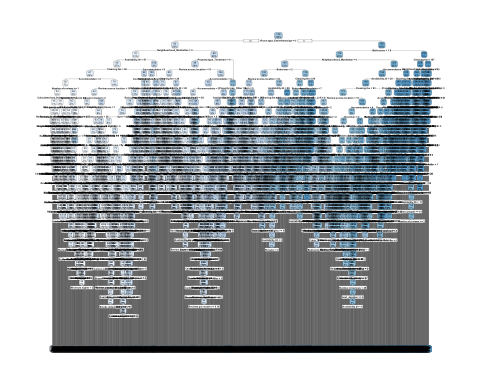
# RMSE for CART

#we first define the RMSE function   
rmse <- function(pred,actual){  
 rmse=sqrt(mean((actual)^2 - (pred)^2))  
 return(rmse)  
}  
  
# now we make a grid of all complexity parameters that we want to select from  
# this vector could be smaller or larger depending on how detailed we want to be in our prediction task  
# here I just use 7 values  
cp\_vec = c(0.0000003,0.002,0.0002,0.5,0.00002,0.000002,0.000000002)  
  
tree.eq <- Price ~ .  
  
# now I build a matrix to store the mse of models with different complexity parameters in the validation set  
valid\_fit <- matrix(0,7,2)  
for(i in 1:7){  
 ny.valcart.cp <- rpart(formula = tree.eq,   
 data = nyTrain, control = rpart.control(cp = cp\_vec[i]))  
 ny\_pred\_valid = predict(ny.valcart.cp, nyVal)  
 valid\_fit[i,1] = cp\_vec[i] # first column: complexity parameter  
 valid\_fit[i,2] = rmse(ny\_pred\_valid,nyVal$Price) # second column: rmse in the validation set  
}

## [,1] [,2]  
## [1,] 3e-07 48.93035  
## [2,] 2e-03 74.66704  
## [3,] 2e-04 60.59450  
## [4,] 5e-01 114.27472  
## [5,] 2e-05 50.43291  
## [6,] 2e-06 48.93901  
## [7,] 2e-09 48.93035

ny.cart.train <- rpart(formula = tree.eq, data = nyTrain, control = rpart.control(cp = 0.000002))  
rpart.plot(ny.cart.train)

## Warning: labs do not fit even at cex 0.15, there may be some overplotting



install.packages("party")  
install.packages("partykit")

## CART - predict on test data

#Predicting on test data  
ny.cart.predict <- predict(ny.cart.train, nyTest)  
rmse\_cart = rmse(ny.cart.predict,nyTest$Price)  
rmse\_cart

## [1] 31.89787

### Inference CART

1. RMSE is 31.89 which is quite high for a price variable
2. The msot important variables as per CART are cleaning fee, availability 365, extra people, host durations, maximum nights, minimum nights, number of reviews.

# Model3: XGBoost

install.packages('xgboost')

#Train model  
library(xgboost)

##   
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':  
##   
## slice

xgb.train.ny = xgboost(data=data.matrix(nyTrain[,]),   
 label = nyTrain$Price,  
 eta = 0.1,  
 max\_depth = 6,  
 subsample = 0.5,  
 nround=100,  
 colsample\_bytree = 1,  
 num\_class = 1,  
 min\_child\_weight = 5,  
 gamma = 15,  
 nthread = 30,  
 eval\_metric = "rmse",  
 objective = "reg:linear",  
 verbose = 0)

## RMSE for XgBoost

#Test model  
xgb.predict.ny = predict(xgb.train.ny,data.matrix(nyTest[,]))  
rmse\_xgb=rmse(xgb.predict.ny,nyTest$Price)  
rmse\_xgb

## [1] 2.398489

## Feature prioritization in XgBoost

library(Ckmeans.1d.dp)  
library(stringr)  
library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following objects are masked from 'package:Metrics':  
##   
## precision, recall

## The following object is masked from 'package:purrr':  
##   
## lift

library(car)  
  
   
model <- xgb.dump(xgb.train.ny, with.stats = T)

## Warning: 'with.stats' is deprecated.  
## Use 'with\_stats' instead.  
## See help("Deprecated") and help("xgboost-deprecated").

model[1:10] #This statement prints top 10 nodes of the model

## [1] "booster[0]"   
## [2] "0:[f5<236.5] yes=1,no=2,missing=1,gain=112987072,cover=15252"   
## [3] "1:[f5<117.5] yes=3,no=4,missing=3,gain=25090464,cover=13330"   
## [4] "3:[f5<72.5] yes=7,no=8,missing=7,gain=3073200,cover=8298"   
## [5] "7:[f5<52.5] yes=15,no=16,missing=15,gain=417742,cover=4175"   
## [6] "15:[f5<41.5] yes=31,no=32,missing=31,gain=73224.5,cover=1915"   
## [7] "31:[f5<32.5] yes=45,no=46,missing=45,gain=17423.5625,cover=763"  
## [8] "45:leaf=2.64789724,cover=213"   
## [9] "46:leaf=3.74664235,cover=550"   
## [10] "32:[f5<47.5] yes=47,no=48,missing=47,gain=5041.25,cover=1152"

# Get the feature real names  
names <- dimnames(data.matrix(nyTrain[,-6]))[[2]]

# Compute feature importance matrix  
importance\_matrix <- xgb.importance(names, model = xgb.train.ny)  
   
importance\_matrix[order(-importance\_matrix$Gain),]  
  
xgb.plot.importance(importance\_matrix= importance\_matrix[1:5], xlab="Gain")  
xgb.plot.importance(importance\_matrix= importance\_matrix[2:20], xlab="Gain") #Plotting features again because in the earlier plot, other features' bars are not visible due to higher value of cleaning fee on the scale.

### XgBoost Inference

1. The RMSE is low at 2.4.
2. The most important variables are cleaning fee, number of reviews, minimum nights, host response rate and beds. The importance for these features in “gain” metric are quite low.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

# Result

|  |  |  |  |
| --- | --- | --- | --- |
|  | Linear Regression | CART | XGBoost |
| RMSE | 2.55 | 31.82 | 2.4 |

The RMSE for XgBoost is the least and best which means that this model is giving better price prediction than CART and Linear Regression. The most important features (and common across all models) for predicting price for NY Airbnb properties is cleaning fee, no. of reviews, minimum nights, free parking on street and availability.

# Conclusion

1. It is seen that cleaning fee is a dominant feature in deciding price. Many hosts cover up for their cost by pricing a higher cleaning fee. At the same time, cleaning fee is positively correlated with price showing that if price increases, cleaning fee also increases.
2. Number of reviews affects the price such that as no. of reviews increases, price decreases. This could be because if a property is economically priced, more people stay and thus no. of reviews eventually increases.
3. Availability for past 30 or 90 days influences price because it is an indicator of popularity/occupancy rate of the property.

# Recommendation

1. Airbnb hosts can use the factors provided by XgBoost model to set prices instead of using arbitary prices for their properties.
2. Certain amenities increase price while some decrease it. Therefore, new hosts can use the most important amenities (like Wireless, TV, refrigerator etc.) to understand how to price their properties.
3. The pricing model will help Airbnb have more competitive prices compared to hotels in the area. While an in depth competitor analysis will be more helpful, this model is a good preliminary check.

# Future work

1. This model is the bulding block to answer the question what commissions Airbnb should charge their hosts. Further data is needed on demand for each property’s price point so that the commission can be fixed based on this.
2. Demand and Price variation as per season is important. During peak tourism season, the prices might be higher. This effect of price change on bookings can also affect commissions of Airbnb.
3. Incorporate image quality into the data to get better predictive models.
4. Augment the model with natural language processing (NLP) of listing descriptions and/or reviews, e.g. for sentiment analysis or looking for keywords.
5. Tailor the model specifically to new listings in order to help hosts set prices for new properties, by removing features that would not be known at the time — e.g. availability and reviews.