Project Title: Team 14 Airbnb Project Final Report
Market Assigned: Chicago

Data Mining and Predictive Analysis

“We, the undersigned, certify that the report submitted is our own original work; all authors participated in this work in a substantive way; all authors have seen and approved the report as submitted; the text, images, illustrations, and other items included in the manuscript do not carry any infringement/plagiarism issue upon any existing copyrighted materials.”

	Names of the signed team members
Contact Member	Jiahong Yu
Member 2	Wenjing Jiang
Member 3	Lina Hu
Member 4	Jeffrey Chen
Member 5	Yimin Liu

Executive Summary

Investing in rental property using Airbnb's platform can be a lucrative business. We want to give recommendations to potential real estate investors on whether or not their property could potentially yield high returns in the city of Chicago. As a midway point between the east and west coast of America, Chicago is a year-round destination with several notable tourist attractions and it's geographical location next to Lake Michigan. As the financial hub of the midwest, the reasons to goto Chicago are endless, with Millennium Park and the University of Chicago close by, Chicago is the closest metropolitan shopping center that many people in the midwest would choose to go to for a weekend getaway. However, Chicago is also notorious for being a city with a high crime rate. Using historical Airbnb data from Chicago, we want to understand what factors contribute to a property's ability to have high booking rates. Then, for the properties with high booking rates, what was the optimal pricing to have for those properties, and what unique niche markets can we discover. Factors like the location of the property, crime rate in that area, prices of previous bookings, and accomodation were considered.

We have discovered that in Chicago, although there are less Airbnb listings in areas with high crime rates, the proportion of high booking rates in both the area with high crime rates and low crime rates is not affected. The significance of this is that potential properties to invest in can still be in neighborhoods where the crime rate is high. We also concluded that Airbnb listings that are closer to downtown Chicago, in the Central area have a higher median and overall listing price, which coincides with the popularity of the number of Airbnb listings for that area, with North Side and West Side ranking second and third consecutively. This has us conclude that location is still the primary determination for pricing in Airbnb listings, with the median price of Central being $159, NorthSide $119, and West $95. For a property with similar amenities and features in both Central and NorthSide, we would recommend charging $40 more if it is located in Central. Lastly, we discovered that properties who accommodate even numbers of guests tend to have higher bookings rates and that properties who can accommodate more guests have higher average booking rates. Following these guidelines of pricing their property based on area, and accomodation accordingly, investors should find high returns in their properties, regardless of crime rate in the neighborhood.

Our investor is not interested in becoming the Airbnb monopoly of Chicago and only wants to make sure his/her investment worths, he would prefer that all the houses we recommend him to purchase are a hit. Therefore, we use the precision as the key metric to evaluating our models and cutoff threshold.

Methodology

R Notebook

```
d <- getwd()
setwd(d)
#Don't forget to set your working directory before you start!

library("tidyverse")
```

```
## -- Attaching packages ---------------------------------------------------
------------------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  3.0.0     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------------------------------------------------------------
------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library("tidymodels")
```

```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo

## -- Attaching packages ---------------------------------------------------
------------------------------------------------ tidymodels 0.1.0 --

## v broom     0.5.5      v rsample   0.0.5
## v dials     0.0.4      v tune      0.1.0
## v infer     0.5.1      v workflows 0.1.1
## v parsnip   0.0.5      v yardstick 0.0.4
## v recipes   0.1.10

## -- Conflicts ------------------------------------------------------------
------------------------------------------ tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x dials::margin()   masks ggplot2::margin()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
```

```
library("plotly")
```

```
##
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout

library("skimr")
library(readr)
library('caret')

## Loading required package: lattice

##
## Attaching package: 'caret'

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

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

library(usethis)
#usethis::edit_r_environ()

dfTrain <-read_csv("airbnbTrain.csv")

## Parsed with column specification:
## cols(
##   .default = col_character(),
##   id = col_double(),
##   high_booking_rate = col_double(),
##   accommodates = col_double(),
##   availability_30 = col_double(),
##   availability_365 = col_double(),
##   availability_60 = col_double(),
##   availability_90 = col_double(),
##   bathrooms = col_double(),
##   bedrooms = col_double(),
##   beds = col_double(),
##   guests_included = col_double(),
##   host_has_profile_pic = col_logical(),
```

```
##   host_identity_verified = col_logical(),
##   host_is_superhost = col_logical(),
##   host_listings_count = col_double(),
##   host_since = col_date(format = ""),
##   instant_bookable = col_logical(),
##   is_business_travel_ready = col_logical(),
##   is_location_exact = col_logical(),
##   latitude = col_double()
##   # ... with 15 more columns
## )

## See spec(...) for full column specifications.

dfTrain<-dfTrain%>%filter(str_extract(dfTrain$`{randomControl}`, "[0-9]{3}")=
='105')
```

We removed all the dollar signs, percentage signs in the numeric columns like 'cleaning_fee' from the raw dataset.

```
dfTrain$cleaning_fee = as.numeric(gsub("[\\$,]", "", dfTrain$cleaning_fee))
dfTrain$extra_people = as.numeric(gsub("[\\$,]", "", dfTrain$extra_people))
dfTrain$security_deposit = as.numeric(gsub("[\\$,]", "", dfTrain$security_dep
osit))
dfTrain$price = as.numeric(gsub("[\\$,]", "", dfTrain$price))
dfTrain$host_response_rate = as.numeric(gsub("[\\%,]", "", dfTrain$host_respo
nse_rate))

## Warning: NAs introduced by coercion

dfTrain$weekly_price = as.numeric(gsub("[\\$,]", "", dfTrain$weekly_price))
dfTrain$monthly_price = as.numeric(gsub("[\\$,]", "", dfTrain$monthly_price))
```

We used the median of each column to replace the NAs in that column. We also had the plan to write a function to find the mode value of each column to replace its NAs. However, this plan was discarded as not all members agree with this approach.

```
colSums(is.na(dfTrain))

##                              id                high_booking_rate
##                               0                                0
##                          access                     accommodates
##                            1797                                0
##                       amenities                  availability_30
##                               0                                0
##                 availability_365                  availability_60
##                               0                                0
##                  availability_90                       bathrooms
##                               0                                0
##                        bed_type                        bedrooms
##                               0                                3
##                            beds              cancellation_policy
##                               1                                0
```

```
##                             city                      cleaning_fee
##                                2                               526
##                      description                      extra_people
##                               75                                 0
##                  guests_included                       host_about
##                                0                              1617
##             host_acceptance_rate              host_has_profile_pic
##                                0                                 0
##           host_identity_verified                 host_is_superhost
##                                0                                 0
##              host_listings_count                     host_location
##                                0                                 6
##              host_neighbourhood                host_response_rate
##                              463                               491
##              host_response_time                      host_since
##                                0                                 0
##              host_verifications                     house_rules
##                                0                              1372
##                instant_bookable                     interaction
##                                0                              1528
##         is_business_travel_ready                is_location_exact
##                                0                                 0
##                         latitude                       longitude
##                                0                                 0
##                           market                    maximum_nights
##                                3                                 0
##                  minimum_nights                    monthly_price
##                                0                              4817
##            neighborhood_overview                    neighbourhood
##                             1244                                16
##                            notes                             price
##                             2219                                 0
##                    property_type  require_guest_phone_verification
##                                0                                 0
##    require_guest_profile_picture                 requires_license
##                                0                                 0
##           review_scores_accuracy            review_scores_checkin
##                              761                               761
##        review_scores_cleanliness      review_scores_communication
##                              761                               762
##           review_scores_location            review_scores_rating
##                              761                               761
##              review_scores_value                         room_type
##                              761                                 0
##                 security_deposit                             space
##                             1282                               920
##                      square_feet                             state
##                             5086                                10
##                          transit                    weekly_price
##                             1394                              4803
```

```
##                           zipcode                   {randomControl}
##                                 9                                 0
```

```
dfTrain$security_deposit[is.na(dfTrain$security_deposit)]=
  median(dfTrain$security_deposit[!is.na(dfTrain$security_deposit)])

dfTrain$review_scores_value[is.na(dfTrain$review_scores_value)]=
  median(dfTrain$review_scores_value[!is.na(dfTrain$review_scores_value)])

dfTrain$review_scores_location[is.na(dfTrain$review_scores_location)]=
  median(dfTrain$review_scores_location[!is.na(dfTrain$review_scores_locatio
n)])


dfTrain$review_scores_checkin[is.na(dfTrain$review_scores_checkin)]=
  median(dfTrain$review_scores_checkin[!is.na(dfTrain$review_scores_checki
n)])

dfTrain$review_scores_accuracy[is.na(dfTrain$review_scores_accuracy)]=
  median(dfTrain$review_scores_accuracy[!is.na(dfTrain$review_scores_accurac
y)])

dfTrain$review_scores_communication[is.na(dfTrain$review_scores_communicatio
n)]=
  median(dfTrain$review_scores_communication[!is.na(dfTrain$review_scores_com
munication)])

dfTrain$review_scores_cleanliness[is.na(dfTrain$review_scores_cleanliness)]=
  median(dfTrain$review_scores_cleanliness[!is.na(dfTrain$review_scores_clean
liness)])

dfTrain$review_scores_rating[is.na(dfTrain$review_scores_rating)]=
  median(dfTrain$review_scores_rating[!is.na(dfTrain$review_scores_rating)])

dfTrain$host_response_rate[is.na(dfTrain$host_response_rate)]=
  median(dfTrain$host_response_rate[!is.na(dfTrain$host_response_rate)])

dfTrain$cleaning_fee[is.na(dfTrain$cleaning_fee)]=
  median(dfTrain$cleaning_fee[!is.na(dfTrain$cleaning_fee)])

dfTrain$bathrooms[is.na(dfTrain$bathrooms)]=
  median(dfTrain$bathrooms[!is.na(dfTrain$bathrooms)])

dfTrain$bedrooms[is.na(dfTrain$bedrooms)]=
  median(dfTrain$bedrooms[!is.na(dfTrain$bedrooms)])

dfTrain$beds[is.na(dfTrain$beds)]=
  median(dfTrain$beds[!is.na(dfTrain$beds)])
```

```
dfTrain$host_identity_verified[is.na(dfTrain$host_identity_verified)]=
  median(dfTrain$host_identity_verified[!is.na(dfTrain$host_identity_verifie
d)])

dfTrain$host_is_superhost[is.na(dfTrain$host_is_superhost)]=
  median(dfTrain$host_is_superhost[!is.na(dfTrain$host_is_superhost)])

dfTrain$host_listings_count[is.na(dfTrain$host_listings_count)]=
  median(dfTrain$host_listings_count[!is.na(dfTrain$host_listings_count)])

dfTrain$weekly_price[is.na(dfTrain$weekly_price)]=
  median(dfTrain$weekly_price[!is.na(dfTrain$weekly_price)])

dfTrain$monthly_price[is.na(dfTrain$monthly_price)]=
  median(dfTrain$monthly_price[!is.na(dfTrain$monthly_price)])

dfTrain$square_feet[is.na(dfTrain$square_feet)]=
  median(dfTrain$square_feet[!is.na(dfTrain$square_feet)])
#################################################################
```

After doing some research on Airbnb hotels, we deleted variables that were duplicated, variables that only had one value(all NAs or all False), variables that have nothing to do with the host, variables that are correlated to another variable(Collinearity), variables that would only be acknowledged after a booking happens, variables that are not needed in this Chicago research(like state, city, random control, etc.).

```
dfTrain[c(3,20,32,34, 43, 45)]=NULL   #Text columns which are duplicated to t
he other variables
dfTrain$space=NULL
dfTrain$transit=NULL
dfTrain$description=NULL



dfTrain$is_business_travel_ready=NULL #All False
dfTrain$host_acceptance_rate=NULL     #All N/A
dfTrain$host_has_profile_pic=NULL     #Nothing to do with our host
dfTrain$monthly_price=NULL            #Correlated with weekly_price
dfTrain$host_since=NULL               #People don't care how long has a host
been

dfTrain$latitude=NULL                 #Duplicated with neighborhood variable
for locating a record
dfTrain$longitude=NULL
dfTrain$zipcode=NULL

dfTrain$host_verifications=NULL       #Verification of the host won't be know
n by the guest
dfTrain$require_guest_phone_verification=NULL #Verification happens after the
guest intent to book a house
```

```
dfTrain$require_guest_profile_picture=NULL

dfTrain$state=NULL                      #We are only doing Chicago,IL's data
dfTrain$city=NULL
dfTrain$market=NULL

dfTrain$host_location=NULL              #Host's location have nothing to do wit
h the house people want to book
dfTrain$host_neighbourhood=NULL

dfTrain$id=NULL                         #ID
dfTrain$requires_license=NULL           #All license are assumed verified on Ai
rbnb by the guests
dfTrain$`{randomControl}`=NULL          #Used for determine the Chicago records
only
dfTrain$bed_type=NULL                   #All beds are normal real beds
```

We firstly removed the special signs in the amenities column

```
dfTrain$amenities<-gsub("\\{|\\}","",dfTrain$amenities)
```

The amenities variable is considered as a possible influential factor in high_booking_rate because of some amenities may be critical for some rentees. However, considering there are more than 150 types of amenities, we needed to reduce the number of dummy variables so that it doesn't bias our model. Therefore, we chose only those amenities dummies that have at least 2000 Trues so that only the most important amenities remained.

```
library(splitstackshape)
b<-cSplit_e(dfTrain, split.col = "amenities", sep = ",", type = "character",
mode = "binary", fixed = TRUE, fill = 0)

b<-b[,39:219]

num<-c()
for (i in 1:181) {
  if(sum(b[,i])<2000){    #we don't want too many dummy variables which can b
ias our models, therefore we need to make sure
    num<-c(num,i)         #the dummies are dominantly common features
  }
}

colnames(b[num])

##   [1] "amenities_\"Accessible-height bed\""
##   [2] "amenities_\"Accessible-height toilet\""
##   [3] "amenities_\"Air purifier\""
##   [4] "amenities_\"Alfresco bathtub\""
##   [5] "amenities_\"Amazon Echo\""
##   [6] "amenities_\"Baby bath\""
```

```
##   [7] "amenities_\"Baby monitor\""
##   [8] "amenities_\"Babysitter recommendations\""
##   [9] "amenities_\"Bath towel\""
##  [10] "amenities_\"Bathroom essentials\""
##  [11] "amenities_\"Bathtub with bath chair\""
##  [12] "amenities_\"BBQ grill\""
##  [13] "amenities_\"Beach essentials\""
##  [14] "amenities_\"Beach view\""
##  [15] "amenities_\"Bedroom comforts\""
##  [16] "amenities_\"Body soap\""
##  [17] "amenities_\"Breakfast table\""
##  [18] "amenities_\"Building staff\""
##  [19] "amenities_\"Buzzer/wireless intercom\""
##  [20] "amenities_\"Cable TV\""
##  [21] "amenities_\"Ceiling fan\""
##  [22] "amenities_\"Central air conditioning\""
##  [23] "amenities_\"Changing table\""
##  [24] "amenities_\"Children’s books and toys\""
##  [25] "amenities_\"Children’s dinnerware\""
##  [26] "amenities_\"Cleaning before checkout\""
##  [27] "amenities_\"Convection oven\""
##  [28] "amenities_\"Day bed\""
##  [29] "amenities_\"Disabled parking spot\""
##  [30] "amenities_\"Double oven\""
##  [31] "amenities_\"DVD player\""
##  [32] "amenities_\"Electric profiling bed\""
##  [33] "amenities_\"En suite bathroom\""
##  [34] "amenities_\"Espresso machine\""
##  [35] "amenities_\"Ethernet connection\""
##  [36] "amenities_\"EV charger\""
##  [37] "amenities_\"Exercise equipment\""
##  [38] "amenities_\"Extra space around bed\""
##  [39] "amenities_\"Family/kid friendly\""
##  [40] "amenities_\"Fire pit\""
##  [41] "amenities_\"Fireplace guards\""
##  [42] "amenities_\"Firm mattress\""
##  [43] "amenities_\"Fixed grab bars for shower\""
##  [44] "amenities_\"Fixed grab bars for toilet\""
##  [45] "amenities_\"Flat path to guest entrance\""
##  [46] "amenities_\"Formal dining area\""
##  [47] "amenities_\"Free parking on premises\""
##  [48] "amenities_\"Full kitchen\""
##  [49] "amenities_\"Game console\""
##  [50] "amenities_\"Garden or backyard\""
##  [51] "amenities_\"Gas oven\""
##  [52] "amenities_\"Ground floor access\""
##  [53] "amenities_\"Handheld shower head\""
##  [54] "amenities_\"HBO GO\""
##  [55] "amenities_\"Heat lamps\""
##  [56] "amenities_\"Heated floors\""
```

```
##  [57] "amenities_\"Heated towel rack\""
##  [58] "amenities_\"High-resolution computer monitor\""
##  [59] "amenities_\"High chair\""
##  [60] "amenities_\"Host greets you\""
##  [61] "amenities_\"Hot tub\""
##  [62] "amenities_\"Hot water kettle\""
##  [63] "amenities_\"Indoor fireplace\""
##  [64] "amenities_\"Jetted tub\""
##  [65] "amenities_\"Lake access\""
##  [66] "amenities_\"Luggage dropoff allowed\""
##  [67] "amenities_\"Memory foam mattress\""
##  [68] "amenities_\"Mini fridge\""
##  [69] "amenities_\"Murphy bed\""
##  [70] "amenities_\"No stairs or steps to enter\""
##  [71] "amenities_\"Other pet(s)\""
##  [72] "amenities_\"Outdoor kitchen\""
##  [73] "amenities_\"Outdoor parking\""
##  [74] "amenities_\"Outdoor seating\""
##  [75] "amenities_\"Outlet covers\""
##  [76] "amenities_\"Pack ’n Play/travel crib\""
##  [77] "amenities_\"Paid parking off premises\""
##  [78] "amenities_\"Paid parking on premises\""
##  [79] "amenities_\"Patio or balcony\""
##  [80] "amenities_\"Pets allowed\""
##  [81] "amenities_\"Pets live on this property\""
##  [82] "amenities_\"Pillow-top mattress\""
##  [83] "amenities_\"Pocket wifi\""
##  [84] "amenities_\"Pool with pool hoist\""
##  [85] "amenities_\"Private bathroom\""
##  [86] "amenities_\"Private living room\""
##  [87] "amenities_\"Projector and screen\""
##  [88] "amenities_\"Rain shower\""
##  [89] "amenities_\"Room-darkening shades\""
##  [90] "amenities_\"Safety card\""
##  [91] "amenities_\"Shared gym\""
##  [92] "amenities_\"Shared hot tub\""
##  [93] "amenities_\"Shared pool\""
##  [94] "amenities_\"Shower chair\""
##  [95] "amenities_\"Single level home\""
##  [96] "amenities_\"Smart lock\""
##  [97] "amenities_\"Smart TV\""
##  [98] "amenities_\"Smoking allowed\""
##  [99] "amenities_\"Soaking tub\""
## [100] "amenities_\"Sound system\""
## [101] "amenities_\"Stair gates\""
## [102] "amenities_\"Stand alone steam shower\""
## [103] "amenities_\"Step-free shower\""
## [104] "amenities_\"Suitable for events\""
## [105] "amenities_\"Sun loungers\""
## [106] "amenities_\"Table corner guards\""
```

```
## [107] "amenities_\"Toilet paper\""
## [108] "amenities_\"Touchless faucets\""
## [109] "amenities_\"translation missing: en.hosting_amenity_49\""
## [110] "amenities_\"translation missing: en.hosting_amenity_50\""
## [111] "amenities_\"Walk-in shower\""
## [112] "amenities_\"Warming drawer\""
## [113] "amenities_\"Washer / Dryer\""
## [114] "amenities_\"Well-lit path to entrance\""
## [115] "amenities_\"Wheelchair accessible\""
## [116] "amenities_\"Wide clearance to shower"
## [117] "amenities_\"Wide doorway to guest bathroom\""
## [118] "amenities_\"Wide entrance for guests\""
## [119] "amenities_\"Wide entrance\""
## [120] "amenities_\"Wide entryway\""
## [121] "amenities_\"Wide hallways\""
## [122] "amenities_\"Window guards\""
## [123] "amenities_\"Wine cooler\""
## [124] "amenities_Balcony"
## [125] "amenities_Bathtub"
## [126] "amenities_Beachfront"
## [127] "amenities_Bidet"
## [128] "amenities_Breakfast"
## [129] "amenities_Cat(s)"
## [130] "amenities_Crib"
## [131] "amenities_Dog(s)"
## [132] "amenities_Doorman"
## [133] "amenities_Elevator"
## [134] "amenities_Gym"
## [135] "amenities_Hammock"
## [136] "amenities_Internet"
## [137] "amenities_Keypad"
## [138] "amenities_Kitchenette"
## [139] "amenities_Lockbox"
## [140] "amenities_Mudroom"
## [141] "amenities_Netflix"
## [142] "amenities_Other"
## [143] "amenities_Pool"
## [144] "amenities_Printer"
## [145] "amenities_Sauna"
## [146] "amenities_Ski-in/Ski-out"
## [147] "amenities_Terrace"
## [148] "amenities_toilet\""
## [149] "amenities_Waterfront"

b<-b[!colnames(b) %in% colnames(b[num])]

dfTrain$amenities=NULL

dfTrain<-dfTrain%>%cbind(b)
```

Because the 'property_type' variable is quite rank deficient, we re-engineered this variable so that those rare property types are all merged as 'Other', leaving only four categories in that variable.

```
dfTrain$property_type[dfTrain$property_type!='Apartment'&dfTrain$property_type!='Condominium'&dfTrain$property_type!='House']='Other'
```

We looked up Chicago's homicide distribution in each neighborhood and made a dummy variable called 'is_high_crime_area' for those high homicide rate neighborhoods.We also made a dummy variable for those neighborhoods that are next to Lake Michigan because we thought they may have a higher booking rate than those who are not.

```
dfTrain$is_next_to_Lake<-ifelse(dfTrain$neighbourhood %in% c('Rogers Park','Edgewater','Uptown','Lakeview','Lincoln Park',
                                                             'Near North Side','Loop','River North','South Loop/Printers Row',
                                                             'Old Town','Streeterville','Gold Coast','Bronzeville','Oakland',
                                                             'Kenwood','Hyde Park','Woodlawn','South Shore','South Chicago',
                                                             'East Side','Hegewisch'),1,0)

dfTrain$is_high_crime_area<-ifelse(dfTrain$neighbourhood %in% c('Austin','Humboldt Park','Garfield Park','North Lawndale',
                                                                'South Lawndale','Little Village','Bronzeville','Englewood',
                                                                'Auburn Gresham","Chatham",'Avalon Park','South Chicago',
                                                                'Calumet Heights','East Side',
                                                                'South Deering','Hegewisch','Pullman','West Pullman',
                                                                'Riverdale','Roseland','Burnside'),1,0)
```

Due to the 'neighbourhood' variable having too many levels, we decide to modify it by merging several neighborhoods into a community area based on Chicago's zone standard. The modified variable is now called 'Community_areas'.

```
dfTrain$neighbourhood[dfTrain$neighbourhood %in% c('Loop','Near North Side','River West','River North',
                                                   'South Loop/Printers Row','Old Town','Streeterville','Gold Coast')]='Central'
dfTrain$neighbourhood[dfTrain$neighbourhood %in% c('North Center','Lakeview','Avondale','Boystown','Logan Square',
                                                   'Lincoln Park','Bucktown','Roscoe Village','Wrigleyville')]='NorthSide'
dfTrain$neighbourhood[dfTrain$neighbourhood %in% c("O'Hare",'Norwood Park','Edison Park','Jefferson Park','North Park',
```

```
                                                    'Albany Park','West Ridge
','Rogers Park','Lincoln Square',
                                                    'Edgewater','Uptown','Ande
rsonville','Sauganash')]='FarNorthSide'
dfTrain$neighbourhood[dfTrain$neighbourhood %in% c("Dunning",'Portage Park','
Irving Park','Montclare','Belmont Cragin',
                                                    'Hermosa')]='NorthWestSide
'
dfTrain$neighbourhood[dfTrain$neighbourhood %in% c("Austin",'Humboldt Park','
West Town/Noble Square','West Loop/Greektown',
                                                    'Near West Side','Pilsen',
'Little Village','North Lawndale',
                                                    'Garfield Park','South Law
ndale','Galewood','Little Italy/UIC',
                                                    'Ukrainian Village','Wicke
r Park')]='WestSide'
dfTrain$neighbourhood[dfTrain$neighbourhood %in% c("Garfield Ridge",'Clearing
','Archer Heights','Brighton Park',
                                                    'McKinley Park','Back of t
he Yards','Gage Park','West Elsdon',
                                                    'West Lawn','Marquette Par
k', 'Englewood' )]='SouthWestSide'
dfTrain$neighbourhood[dfTrain$neighbourhood %in% c("Bridgeport",'Armour Squar
e','Chinatown','Bronzeville',
                                                    'Washington Park','Oakland
','Kenwood','Hyde Park',
                                                    'Woodlawn','Grand Crossing
', 'South Shore')]='SouthSide'
dfTrain$neighbourhood[dfTrain$neighbourhood %in% c("Ashburn",'Auburn Gresham
','Washington Heights','Beverly',
                                                    'Morgan Park','Mount Green
wood','Mt. Greenwood')]='FarSouthWestSide'
dfTrain$neighbourhood[dfTrain$neighbourhood %in% c("Chatham",'Avalon Park','S
outh Chicago','Calumet Heights',
                                                    'East Side','South Deering
','Hegewisch','Pullman','West Pullman',
                                                    'Riverdale','Roseland','Bu
rnside')]='FarSouthEastSide'
dfTrain$neighbourhood[is.na(dfTrain$neighbourhood)]='N/A'


dfTrain<-dfTrain%>%rename(Community_areas='neighbourhood')
```

Based on whether a variable is in the major region of Chicago(North Side, South Side, West Side, Central), we made a dummy variable called ‘is_major_section’.

```
dfTrain$is_major_section<-ifelse(dfTrain$Community_areas %in% c('WestSide','N
orthSide','Central','SouthSide'),1,0)
```

Eventually, we finished up the data processing by converting the data type of the dependent variable 'high_booking_rate' to factor. We also made sure the column names of the dataset are easy to read.

```
dfTrain$high_booking_rate=factor(dfTrain$high_booking_rate)
colnames(dfTrain)<-make.names(colnames(dfTrain),unique = TRUE)

#Setting the seed
set.seed(333)

#Partitioning the dataset into a 70% - 30% split
Train <- dfTrain %>% sample_frac(.7)
Test <- setdiff(dfTrain, Train)
```

Random Forest Method

```
# Random Forest Method
library("randomForest")

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dials':
##
##     margin

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

## The following object is masked from 'package:ggplot2':
##
##     margin

colnames(Train[, sapply(Train, class) == 'character'])

## [1] "cancellation_policy" "host_response_time"  "Community_areas"
## [4] "property_type"       "room_type"

Train$cancellation_policy=as.factor(Train$cancellation_policy)
Train$host_response_time=as.factor(Train$host_response_time)
Train$Community_areas=as.factor(Train$Community_areas)
Train$property_type=as.factor(Train$property_type)
Train$room_type=as.factor(Train$room_type)

Test$cancellation_policy=as.factor(Test$cancellation_policy)
Test$host_response_time=as.factor(Test$host_response_time)
Test$Community_areas=as.factor(Test$Community_areas)
```

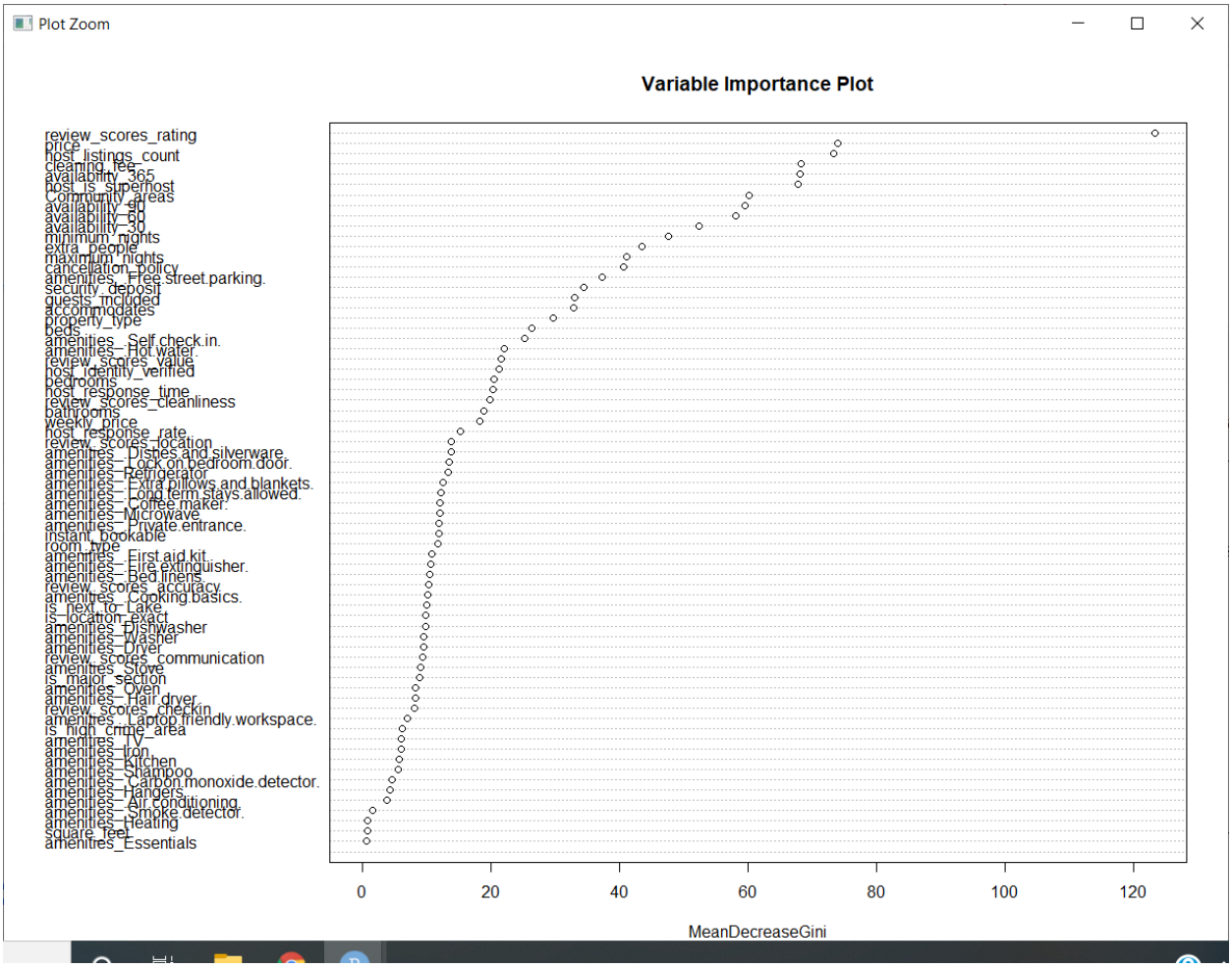
```r
Test$property_type=as.factor(Test$property_type)
Test$room_type=as.factor(Test$room_type)



fit.forest <- randomForest(high_booking_rate~., data=Train, importance=TRUE)
#grows the forest


# resultsrandomforest<-fit.forest%>%
#   predict(Test, type = 'prob') %>%
#   cbind(Test)
#
# resultsrandomforest$`0`=NULL
# resultsrandomforest$predictedraw<-ifelse(resultsrandomforest$`1`>0.5,1,0)

# library(cvAUC)
# resultsrandomforest$predictedraw<-as.numeric(resultsrandomforest$predictedraw)
# AUC(resultsrandomforest$predictedraw, resultsrandomforest$high_booking_rate)

resultsforest<- fit.forest%>%
  predict(Test, type = 'response') %>%
  bind_cols(Test, predictedClass=.)

resultsforest %>%
  xtabs(~predictedClass+ high_booking_rate,.) %>%
  confusionMatrix(positive = '1')
```

```
## Confusion Matrix and Statistics
##
##               high_booking_rate
## predictedClass   0   1
##              0 879 203
##              1 102 351
##
##                Accuracy : 0.8013
##                  95% CI : (0.7804, 0.821)
##     No Information Rate : 0.6391
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5515
##
##  Mcnemar's Test P-Value : 1.028e-08
##
##             Sensitivity : 0.6336
##             Specificity : 0.8960
##          Pos Pred Value : 0.7748
```

```
##          Neg Pred Value : 0.8124
##              Prevalence : 0.3609
##          Detection Rate : 0.2287
##    Detection Prevalence : 0.2951
##       Balanced Accuracy : 0.7648
##
##        'Positive' Class : 1
##
```

determine variable importance

```
#variable importance chart
varImpPlot(fit.forest, n.var = 71, type = 2, main = "Variable Importance Plot
")
```




```
#Run it in the console and click the zoom button on the 'Plots' tag to zoom i
n
```

After evaluating the variable importance of all 71 processed variables, we decided to use the elbow method to select only the most contributing variables(those whose mean decrease gini is above the elbow variable's, which in this case,is 'weekly_price')

Random Forest with the most contributing variables:

```
fit.forest_tuned <- randomForest(high_booking_rate~review_scores_rating+price
+host_listings_count+cleaning_fee+availability_365+
                                 host_is_superhost+Community_areas+availability_9
0+availability_60+availability_30+
                                 minimum_nights+extra_people+maximum_nights+cance
llation_policy+amenities_.Free.street.parking.+
                                 security_deposit+guests_included+accommodates+pr
operty_type+beds+
                                 amenities_.Self.check.in.+amenities_.Hot.water.+
review_scores_value+host_identity_verified+
                                   bedrooms+host_response_time+review_scores_clea
nliness+bathrooms+weekly_price, data=Train, importance=TRUE) #grows the fores
t

resultsforesttuned<- fit.forest_tuned%>%
  predict(Test, type = 'response') %>%
  bind_cols(Test, predictedClass=.)

resultsforesttuned %>%
  xtabs(~predictedClass+ high_booking_rate,.) %>%
  confusionMatrix(positive = '1')

## Confusion Matrix and Statistics
##
##                high_booking_rate
## predictedClass   0   1
##              0 881 181
##              1 100 373
##
##                Accuracy : 0.8169
##                  95% CI : (0.7967, 0.836)
##     No Information Rate : 0.6391
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5901
##
##  Mcnemar's Test P-Value : 1.82e-06
##
##             Sensitivity : 0.6733
##             Specificity : 0.8981
##          Pos Pred Value : 0.7886
##          Neg Pred Value : 0.8296
##              Prevalence : 0.3609
##          Detection Rate : 0.2430
##    Detection Prevalence : 0.3081
##       Balanced Accuracy : 0.7857
##
```

```
##        'Positive' Class : 1
##
```

Bayes GLM:

```
#Bayes GLM method
library('arm')

## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following object is masked from 'package:plotly':
##
##     select

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

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loading required package: lme4

##
## arm (Version 1.11-1, built: 2020-4-27)

## Working directory is C:/Users/victo/Downloads

##
## Attaching package: 'arm'

## The following object is masked from 'package:scales':
##
##     rescale

fitbayesglm <- bayesglm(high_booking_rate ~., data=Train, family="binomial")

#display(fit)
#summary(fit)

resultsbayesglm<-fitbayesglm%>%
  predict(Test, type = 'response') %>%
  bind_cols(Test, predictedresponse=.)
```

```
resultsbayesglm$predicted<-ifelse(resultsbayesglm$predictedresponse>0.5,1,0)

resultsbayesglm %>%
  xtabs(~predicted+ high_booking_rate,.) %>%
  confusionMatrix(positive = '1')

## Confusion Matrix and Statistics
##
##          high_booking_rate
## predicted   0   1
##         0 835 229
##         1 146 325
##
##                Accuracy : 0.7557
##                  95% CI : (0.7334, 0.777)
##     No Information Rate : 0.6391
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4526
##
##  Mcnemar's Test P-Value : 2.291e-05
##
##             Sensitivity : 0.5866
##             Specificity : 0.8512
##          Pos Pred Value : 0.6900
##          Neg Pred Value : 0.7848
##              Prevalence : 0.3609
##          Detection Rate : 0.2117
##    Detection Prevalence : 0.3068
##       Balanced Accuracy : 0.7189
##
##        'Positive' Class : 1
##
```

Bayes GLM method using tuned variables:

```
#Bayes GLM method using tuned variables
library('arm')
fitbayesglmtuned <- bayesglm(high_booking_rate~review_scores_rating+price+hos
t_listings_count+cleaning_fee+availability_365+
                               host_is_superhost+Community_areas+availability_9
0+availability_60+availability_30+
                               minimum_nights+extra_people+maximum_nights+cance
llation_policy+amenities_.Free.street.parking.+
                               security_deposit+guests_included+accommodates+pr
operty_type+beds+
                               amenities_.Self.check.in.+amenities_.Hot.water.+
review_scores_value+host_identity_verified+
                                 bedrooms+host_response_time+review_scores_clea
nliness+bathrooms+weekly_price, data=Train, family="binomial")
```

```
#display(fit)
#summary(fit)

resultsbayesglmtuned<-fitbayesglmtuned%>%
  predict(Test, type = 'response') %>%
  bind_cols(Test, predictedresponse=.)

resultsbayesglmtuned$predicted<-ifelse(resultsbayesglmtuned$predictedrespons
e>0.5,1,0)

resultsbayesglmtuned %>%
  xtabs(~predicted+ high_booking_rate,.) %>%
  confusionMatrix(positive = '1')
```

```
## Confusion Matrix and Statistics
##
##          high_booking_rate
## predicted   0   1
##         0 840 266
##         1 141 288
##
##                Accuracy : 0.7349
##                  95% CI : (0.712, 0.7568)
##     No Information Rate : 0.6391
##     P-Value [Acc > NIR] : 7.976e-16
##
##                   Kappa : 0.3955
##
##  Mcnemar's Test P-Value : 7.924e-10
##
##            Sensitivity : 0.5199
##            Specificity : 0.8563
##         Pos Pred Value : 0.6713
##         Neg Pred Value : 0.7595
##             Prevalence : 0.3609
##         Detection Rate : 0.1876
##    Detection Prevalence : 0.2795
##       Balanced Accuracy : 0.6881
##
##        'Positive' Class : 1
##
```

glmnet with all variables:

```
#glmnet
set.seed(333)
lambda <- 10^seq(-5, 2, length = 100)

fitElasticNet <- train(high_booking_rate~.,family='binomial', data = Train, m
```

```
ethod = "glmnet",
                        trControl = trainControl("cv", number = 10),
                        tuneGrid = expand.grid(alpha = 0.5, lambda = lambda))
fitElasticNet$bestTune$lambda
```

```
## [1] 0.0001873817
```

```
resultsElasticNetCaret <-
fitElasticNet %>%
predict(Test, type='raw') %>%
bind_cols(Test, predictedClass=.)

resultsElasticNetCaret %>%
xtabs(~predictedClass+high_booking_rate, .) %>%
confusionMatrix(positive = '1')
```

```
## Confusion Matrix and Statistics
##
##                high_booking_rate
## predictedClass   0   1
##              0 835 229
##              1 146 325
##
##                Accuracy : 0.7557
##                  95% CI : (0.7334, 0.777)
##     No Information Rate : 0.6391
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4526
##
##  Mcnemar's Test P-Value : 2.291e-05
##
##             Sensitivity : 0.5866
##             Specificity : 0.8512
##          Pos Pred Value : 0.6900
##          Neg Pred Value : 0.7848
##              Prevalence : 0.3609
##          Detection Rate : 0.2117
##    Detection Prevalence : 0.3068
##       Balanced Accuracy : 0.7189
##
##        'Positive' Class : 1
##
```

glmnet with key variables only:

```
#glmnet with selected variables
set.seed(333)
lambda <- 10^seq(-5, 2, length = 100)

fitElasticNetTuned <- train(high_booking_rate~review_scores_rating+price+host
```

```
_listings_count+cleaning_fee+availability_365+
                              host_is_superhost+Community_areas+availability_9
0+availability_60+availability_30+
                              minimum_nights+extra_people+maximum_nights+cance
llation_policy+amenities_.Free.street.parking.+
                              security_deposit+guests_included+accommodates+pr
operty_type+beds+
                              amenities_.Self.check.in.+amenities_.Hot.water.+
review_scores_value+host_identity_verified+
                                bedrooms+host_response_time+review_scores_clea
nliness+bathrooms+weekly_price,family='binomial', data = Train, method = "glm
net",
                             trControl = trainControl("cv", number = 10),
                             tuneGrid = expand.grid(alpha = 0.5, lambda = lamb
da))
fitElasticNetTuned$bestTune$lambda
```

```
## [1] 0.001321941
```

```
resultsElasticNetTuned <-
fitElasticNetTuned %>%
predict(Test, type='raw') %>%
bind_cols(Test, predictedClass=.)

resultsElasticNetTuned %>%
  xtabs(~predictedClass+high_booking_rate, .) %>%
  confusionMatrix(positive = '1')
```

```
## Confusion Matrix and Statistics
##
##               high_booking_rate
## predictedClass   0   1
##              0 838 268
##              1 143 286
##
##                Accuracy : 0.7322
##                  95% CI : (0.7093, 0.7543)
##     No Information Rate : 0.6391
##     P-Value [Acc > NIR] : 4.756e-15
##
##                   Kappa : 0.3896
##
##  Mcnemar's Test P-Value : 9.567e-10
##
##             Sensitivity : 0.5162
##             Specificity : 0.8542
##          Pos Pred Value : 0.6667
##          Neg Pred Value : 0.7577
##              Prevalence : 0.3609
##          Detection Rate : 0.1863
```

```
##    Detection Prevalence : 0.2795
##       Balanced Accuracy : 0.6852
##
##        'Positive' Class : 1
##
```

LDA with all variables:

```
#LDA
train.control <- trainControl(method = "cv", number = 10)
# Train the model
fitLDA <- train(high_booking_rate ~.,family='binomial', data = Train, method
= "lda",trControl = train.control)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
resultsLDACaret <-fitLDA %>%
  predict(Test, type='raw') %>%
  bind_cols(Test, predictedClass=.)

resultsLDACaret %>%
  xtabs(~predictedClass+high_booking_rate, .) %>%
  confusionMatrix(positive = '1')
```

```
## Confusion Matrix and Statistics
##
##                high_booking_rate
## predictedClass   0   1
##              0 830 235
##              1 151 319
```

```
##
##                Accuracy : 0.7485
##                  95% CI : (0.726, 0.7701)
##     No Information Rate : 0.6391
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4363
##
##  Mcnemar's Test P-Value : 2.394e-05
##
##             Sensitivity : 0.5758
##             Specificity : 0.8461
##          Pos Pred Value : 0.6787
##          Neg Pred Value : 0.7793
##              Prevalence : 0.3609
##          Detection Rate : 0.2078
##    Detection Prevalence : 0.3062
##       Balanced Accuracy : 0.7109
##
##        'Positive' Class : 1
##
```

LDA with key variables only:

```
#LDA with selected variables

train.control <- trainControl(method = "cv", number = 10)
# Train the model
fitLDATuned <- train(high_booking_rate~review_scores_rating+price+host_listin
gs_count+cleaning_fee+availability_365+
                               host_is_superhost+Community_areas+availability_9
0+availability_60+availability_30+
                               minimum_nights+extra_people+maximum_nights+cance
llation_policy+amenities_.Free.street.parking.+
                               security_deposit+guests_included+accommodates+pr
operty_type+beds+
                               amenities_.Self.check.in.+amenities_.Hot.water.+
review_scores_value+host_identity_verified+
                                 bedrooms+host_response_time+review_scores_clea
nliness+bathrooms+weekly_price,family='binomial', data = Train, method = "lda
",
                trControl =train.control)

resultsLDATuned <-fitLDATuned %>%
  predict(Test, type='raw') %>%
  bind_cols(Test, predictedClass=.)

resultsLDATuned %>%
  xtabs(~predictedClass+high_booking_rate, .) %>%
  confusionMatrix(positive = '1')
```

```
## Confusion Matrix and Statistics
##
##                high_booking_rate
## predictedClass   0   1
##              0 840 262
##              1 141 292
##
##                Accuracy : 0.7375
##                  95% CI : (0.7147, 0.7593)
##     No Information Rate : 0.6391
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4025
##
##  Mcnemar's Test P-Value : 2.264e-09
##
##             Sensitivity : 0.5271
##             Specificity : 0.8563
##          Pos Pred Value : 0.6744
##          Neg Pred Value : 0.7623
##              Prevalence : 0.3609
##          Detection Rate : 0.1902
##    Detection Prevalence : 0.2821
##       Balanced Accuracy : 0.6917
##
##        'Positive' Class : 1
##
```

KNN with all variables:

```
#KNN
fitKNN <- train(high_booking_rate ~ ., method='knn', data=Train, trControl=tr
ainControl(method='cv', number=10), preProcess = c("center", "scale"), tuneLe
ngth = 30)

fitKNN$finalModel
```

```
## 13-nearest neighbor model
## Training set outcome distribution:
##
##    0    1
## 2331 1253
```

```
#Let's run KNN on the same model, make predictions, and calculate performance
resultsKNN <-
  fitKNN %>%
  predict(Test, type='raw') %>%
  bind_cols(Test, predictedClass=.)

resultsKNN %>%
```

```
  xtabs(~predictedClass+high_booking_rate, .) %>%
  confusionMatrix(positive = '1')
```

```
## Confusion Matrix and Statistics
##
##                high_booking_rate
## predictedClass   0   1
##              0 783 208
##              1 198 346
##
##                Accuracy : 0.7355
##                  95% CI : (0.7127, 0.7574)
##     No Information Rate : 0.6391
##     P-Value [Acc > NIR] : 5.063e-16
##
##                   Kappa : 0.4244
##
##  Mcnemar's Test P-Value : 0.6551
##
##             Sensitivity : 0.6245
##             Specificity : 0.7982
##          Pos Pred Value : 0.6360
##          Neg Pred Value : 0.7901
##              Prevalence : 0.3609
##          Detection Rate : 0.2254
##    Detection Prevalence : 0.3544
##       Balanced Accuracy : 0.7114
##
##        'Positive' Class : 1
##
```

KNN with key variables only:

```
#KNN Selected Variables
fitKNNTuned <- train(high_booking_rate~review_scores_rating+price+host_listin
gs_count+cleaning_fee+availability_365+
                               host_is_superhost+Community_areas+availability_9
0+availability_60+availability_30+
                               minimum_nights+extra_people+maximum_nights+cance
llation_policy+amenities_.Free.street.parking.+
                               security_deposit+guests_included+accommodates+pr
operty_type+beds+
                               amenities_.Self.check.in.+amenities_.Hot.water.+
review_scores_value+host_identity_verified+
                                 bedrooms+host_response_time+review_scores_clea
nliness+bathrooms+weekly_price, method='knn', data=Train, trControl=trainCont
rol(method='cv', number=10), preProcess = c("center", "scale"), tuneLength =
30)
```

```r
#See the final model output:
fitKNNTuned$finalModel
```

```
## 37-nearest neighbor model
## Training set outcome distribution:
##
##    0    1
## 2331 1253
```

```r
#Let's run KNN on the same model, make predictions, and calculate performance
resultsKNNTuned <-
  fitKNNTuned %>%
  predict(Test, type='raw') %>%
  bind_cols(Test, predictedClass=.)

resultsKNNTuned %>%
  xtabs(~predictedClass+high_booking_rate, .) %>%
  confusionMatrix(positive = '1')
```

```
## Confusion Matrix and Statistics
##
##               high_booking_rate
## predictedClass   0   1
##              0 812 233
##              1 169 321
##
##                Accuracy : 0.7381
##                  95% CI : (0.7153, 0.76)
##     No Information Rate : 0.6391
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4176
##
##  Mcnemar's Test P-Value : 0.001677
##
##             Sensitivity : 0.5794
##             Specificity : 0.8277
##          Pos Pred Value : 0.6551
##          Neg Pred Value : 0.7770
##              Prevalence : 0.3609
##          Detection Rate : 0.2091
##    Detection Prevalence : 0.3192
##       Balanced Accuracy : 0.7036
##
##        'Positive' Class : 1
##
```

It seems that Random Forest with those most contributing variables only) won the prize of the best model in most of the metrics, notably the precision as our client wants as many hits in our predictions as possible.

Reduced Variables Random Forest's Confusion Matrix:

```
Confusion Matrix and Statistics

               high_booking_rate
predictedClass   0   1
             0 881 181
             1 100 373

               Accuracy : 0.8169
                 95% CI : (0.7967, 0.836)
    No Information Rate : 0.6391
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5901

 Mcnemar's Test P-Value : 1.82e-06

            Sensitivity : 0.6733
            Specificity : 0.8981
         Pos Pred Value : 0.7886
         Neg Pred Value : 0.8296
             Prevalence : 0.3609
         Detection Rate : 0.2430
   Detection Prevalence : 0.3081
      Balanced Accuracy : 0.7857

       'Positive' Class : 1
```

Moreover, we researched the meaning of kappa.According to Chen (https://towardsdatascience.com/interpretation-of-kappa-values-2acd1ca7b18f), the kappa is a statistic to measure the extent to which data collectors(raters) assign the same score to the same variable(called interrater reliability). It is used to account for the possibility that raters actually guess on at least some variables due to uncertainty.

In general, the higher the kappa is, the better the model is.In this case, the reduced model has a higher kappa than any other models, means the raters have higher probability to assign the same score to this model.

Afterall, our group decided to use the reduced Random Forest model as our final model. Now we need to determine the cutoff threshold for this model in order to maximize the precision.

```r
#Research the best cutoff

#What is the default cutoff of random forest classifier

resultsforesttuned<- fit.forest_tuned%>%
  predict(Test, type = 'prob') %>%
  cbind(Test)
```

```
resultsforesttuned$predictedClass<-ifelse(resultsforesttuned$`1`>0.5,1,0)

resultsforesttuned %>%
  xtabs(~predictedClass+ high_booking_rate,.) %>%
  confusionMatrix(positive = '1')
```

```
## Confusion Matrix and Statistics
##
##                high_booking_rate
## predictedClass   0   1
##              0 881 181
##              1 100 373
##
##                Accuracy : 0.8169
##                  95% CI : (0.7967, 0.836)
##     No Information Rate : 0.6391
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5901
##
##  Mcnemar's Test P-Value : 1.82e-06
##
##             Sensitivity : 0.6733
##             Specificity : 0.8981
##          Pos Pred Value : 0.7886
##          Neg Pred Value : 0.8296
##              Prevalence : 0.3609
##          Detection Rate : 0.2430
##    Detection Prevalence : 0.3081
##       Balanced Accuracy : 0.7857
##
##        'Positive' Class : 1
##
```

The default probability cutoff for the random forest classifier is 0.5. In general, a higher cutoff value stands for higher precision(how many positive predictions are true) and lower sensitivity(how many actual positives are predicted), because we only predict the most possible records of all the possible records to be true, thereby drops a lot of less possible records and decreases the sensitivity. Due to the same reason, a lower cutoff value stands for lower precision and higher sensitivity.

```
#We kept rising the cutoff value to see the change in Pos Pred Value(precisio
n)

#What is the minimum cutoff value if we want to maximize the precision?
resultsforesttuned$predictedClass<-ifelse(resultsforesttuned$`1`>0.88,1,0)

resultsforesttuned %>%
```

```
  xtabs(~predictedClass+ high_booking_rate,.) %>%
  confusionMatrix(positive = '1')
```

```
## Confusion Matrix and Statistics
##
##                high_booking_rate
## predictedClass   0   1
##              0 981 539
##              1   0  15
##
##                Accuracy : 0.6489
##                  95% CI : (0.6244, 0.6728)
##     No Information Rate : 0.6391
##     P-Value [Acc > NIR] : 0.2208
##
##                   Kappa : 0.0343
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.027076
##             Specificity : 1.000000
##          Pos Pred Value : 1.000000
##          Neg Pred Value : 0.645395
##              Prevalence : 0.360912
##          Detection Rate : 0.009772
##    Detection Prevalence : 0.009772
##       Balanced Accuracy : 0.513538
##
##        'Positive' Class : 1
##
```

```
#The minimum cutoff to maximize the precision is 0.88
```

Because our investor prefers us to ensure all our predictions are a hit rather than covering as many potential positives(high booking rate properties) as possible, our key concern is to find the minimum cutoff that maximizes the precision. According to our testing, when the cutoff threshold rises to 0.88 or above, the value of precision reaches 1.0, which means every single positive prediction of our model is a hit. Therefore, the minimum cutoff to maximize the precision is 0.88

Now, let's do some exploratory analysis on the most contributing characteristics to a high booking rate property.

```
#Chicago Airbnb Characteristic Analsysis through exploring the variable importance
fit.forest_tuned$importance
```

```
> fit.forest_tuned$importance
                                         0          1 MeanDecreaseAccuracy MeanDecreaseGini
review_scores_rating           0.010951996 0.089203529          0.038321492        190.33498
price                          0.005333104 0.020650366          0.010672595        110.27131
host_listings_count            0.015043972 0.037456712          0.022874252        109.14985
cleaning_fee                   0.007728096 0.021180730          0.012427980        101.91818
availability_365               0.015566357 0.011848131          0.014268902        102.37497
host_is_superhost              0.022064526 0.079865036          0.042287995         90.53698
Community_areas                0.006916229 0.009541317          0.007823136         88.99599
availability_90                0.013320164 0.014851659          0.013889621         82.50838
availability_60                0.012901257 0.017101309          0.014394619         81.83116
availability_30                0.008497023 0.015587198          0.010973884         74.32515
minimum_nights                 0.004821752 0.021528291          0.010671516         63.36991
extra_people                   0.005843099 0.009191872          0.007011951         60.03458
maximum_nights                 0.003552238 0.006145947          0.004461484         60.85379
cancellation_policy            0.006068937 0.019398137          0.010721850         54.04143
amenities_.Free.street.parking. 0.013533062 0.015279750          0.014158059         52.67878
security_deposit               0.005354993 0.004050531          0.004901556         51.85863
guests_included                0.006912296 0.012838842          0.008982470         47.61284
accommodates                   0.002890051 0.005047464          0.003632185         49.11401
property_type                  0.002917679 0.003213532          0.003029081         43.85083
beds                           0.002498505 0.004050534          0.003046293         40.67406
amenities_.Self.check.in.      0.007071861 0.011637952          0.008673401         34.30628
amenities_.Hot.water.          0.004702160 0.014299637          0.008056344         33.38058
> |
```

The reduced random forest model gives us quite a lot of exploratory insights regarding Chicago's Airbnb market. First of all, visitors to Chicago's Airbnb have many characteristics that fit our common senses. They care about the review ratings, price, cleaning fees, availability, and if the host is an Airbnb superhost. However, they also pay high intention to whether the host has multiple listings on Airbnb. It may because those hosts who have many listings may have a professional manager to manage and promote them, who in return, make these listings more competitive.

Community areas are also a significantly important variable that affects the probability of being a high booking rate property. It means there is a strong regional difference in Chicago's Airbnb market. Some areas are more popular than others. The regional effect of Chicago Airbnb is also our research focus for this project.

Some other interesting and unique factors that affect Chicago Airbnb market are some amenities. For instance, those properties that have an amenity for free road parking are more popular than others. Based on our personal life experience in Chicago, I'd say this is probably true as the City of Chicago charges so high for overnight parking that even the hotel fees could be suppressed.

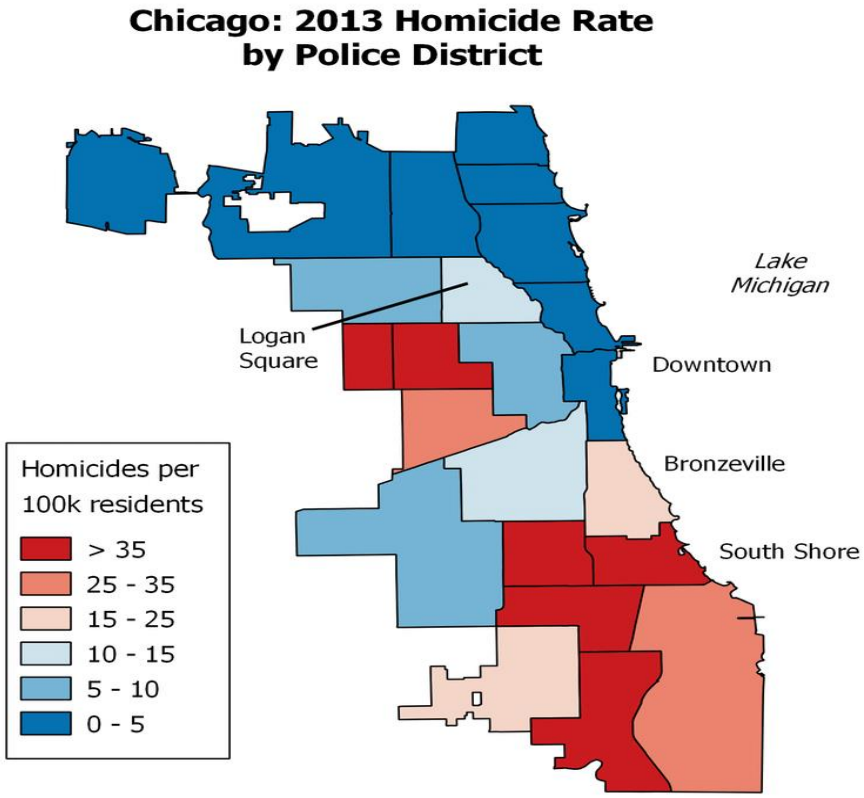
Research Questions And Findings

For our analysis, we wanted to understand if an area with historical high crime rates would affect the booking rate for Airbnb properties in the area, and if so by how much. This is so we can understand if Airbnb users would take into consideration not booking in an area where crime rates are high. We also wanted to understand if popular areas have higher price premiums for Airbnb, and whether this had a correlation with areas with high crime rates. This is so we can understand if pricing is based on safety versus other factors. Lastly, we wanted to understand how accomodation could affect our booking rate and listing price. This is so we can understand if properties that can accommodate more guests would generate more revenue, or if price per person scales off.

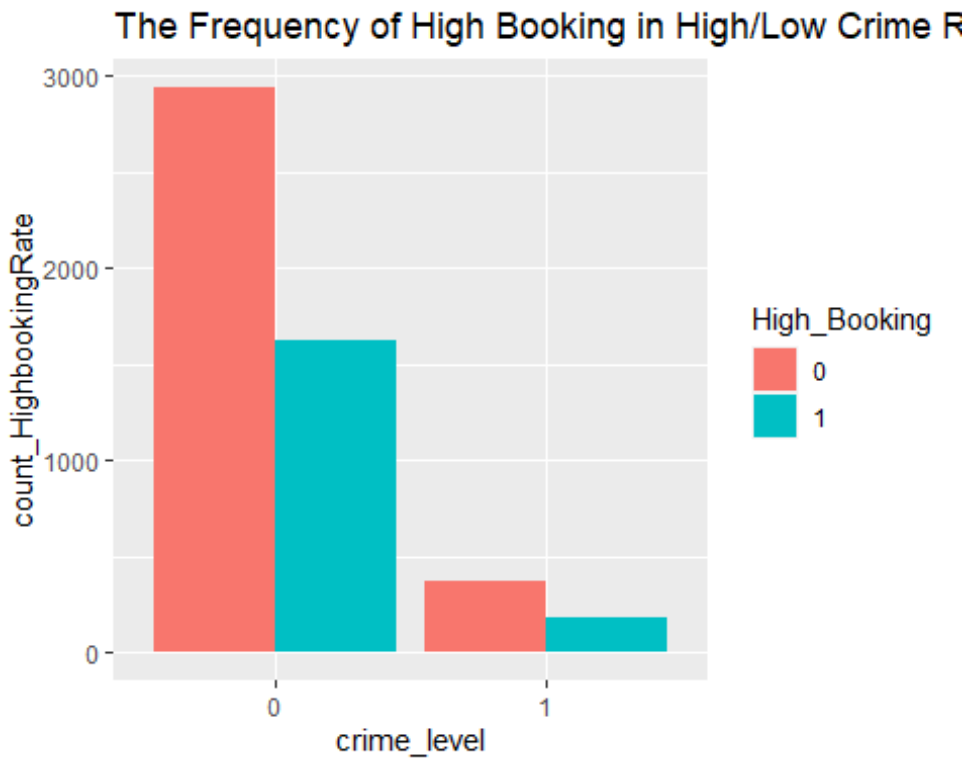
Question 1: Does Chicago's high crime rate areas affect their booking rate of Airbnb?

We separate Chicago into eight areas to check how specific characteristics of a District affect the booking rate.

Since the West Side, South Side and Far Southeast Side have a higher crime rate according to our research. We determine one area has a high crime rate if it is affiliated with these three pre-identified areas.



```
CrimeRate <-
  dfTrain %>%
  group_by(crime_level = as.factor(is_high_crime_area),High_Booking =
as.factor(high_booking_rate)) %>%
  summarize(count_HighbookingRate = n())

CrimeRatePlot <- ggplot(data.frame(CrimeRate), aes( x = crime_level,
                                                    y =
count_HighbookingRate,
                                                    fill= High_Booking)) +
  geom_bar(stat = "identity", position="dodge") +
  ggtitle("The Frequency of High Booking in High/Low Crime Rate Areas")


CrimeRatePlot
```



On X-axis, Crime _level 0 represents for low crime rate area, Crime _level 1 represents for high crime rate. From the above bar chart, Red bar refers to low booking rate,and blue bar refers to high booking rate. As we can see from this graph, obviously low crime rate area has higher number of high booking rate listings. From the 5120 data point, only roughly 30% of the listings have high booking rate. The airbnb market in Chicago is not as popular as we expected.

Using Bayesian Linear Regression to testify our idea:

```
fit <- bayesglm(high_booking_rate ~.-Community_areas, data=dfTrain,
family="binomial")

summary(fit)

##
## Call:
## bayesglm(formula = high_booking_rate ~ . - Community_areas, family =
"binomial",
##     data = dfTrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4856  -0.7851  -0.3674   0.8502   5.0903
```

```
amenities_Washer                    0.3234632  0.4114877   0.786 0.431819
is_next_to_Lake                     0.0185216  0.0780846   0.237 0.812502
is_high_crime_area                 -0.2075156  0.1226181  -1.692 0.090575 .
is_major_section                    0.3166172  0.0903942   3.503 0.000461 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)
```

Results and Findings of Q1:

We ran the bayesian model with high booking rate as our dependent variable, all the variables, except Commuinty_areas variable because we create three dummy variables(is_next_to_Lake, is_high_crime_area, is_major_section) from this categorical variable. From this model, we can find that if the house is in the high crime neighbourhoods, the coefficient is negative which implies that being a high crime rate neighbourhood has negative effect on the booking rate, holding everything else constant. If we set alpha=0.1, the coefficient of the variable is significant.

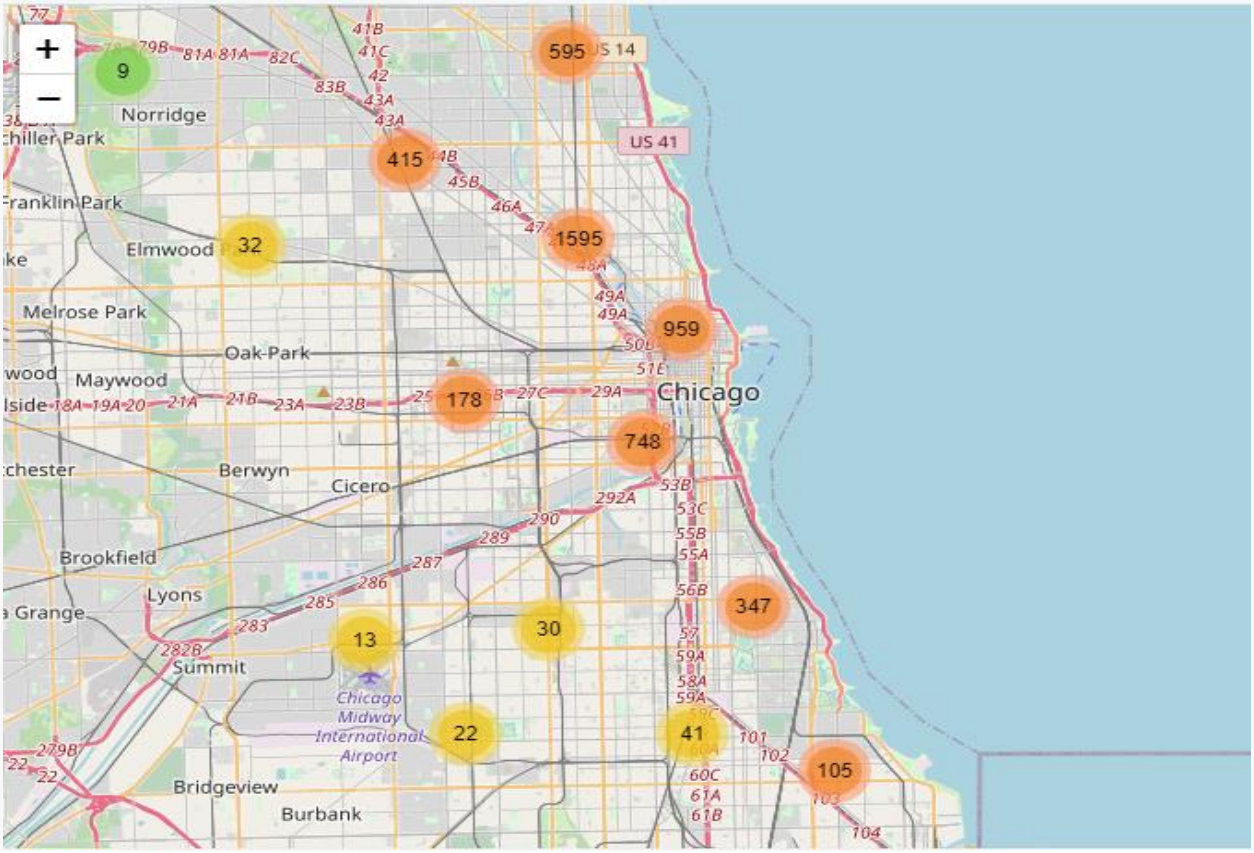
We also have other interesting findings when doing research regarding the relationship between location and high booking rate.For example, if the house is in major section, the coefficient of the variable is positive and significant. So if we consider realtionship between the charactistic of location and booking rate in Chicago, the most important factors are whether it is close to the downtown and whether it is in a high crime rate neighbourhood.

Q2: Does popular areas have price premium than those not?

The second question we would like to explore is whether popular areas have price premium than those not. So what we did is we map out all the listings using the longitude and latitude in the dataset and comparing these clusters on the map with the Chicago

district map which is divided into 8 main areas, we merge the smaller N into these 8 areas as we have mentioned.

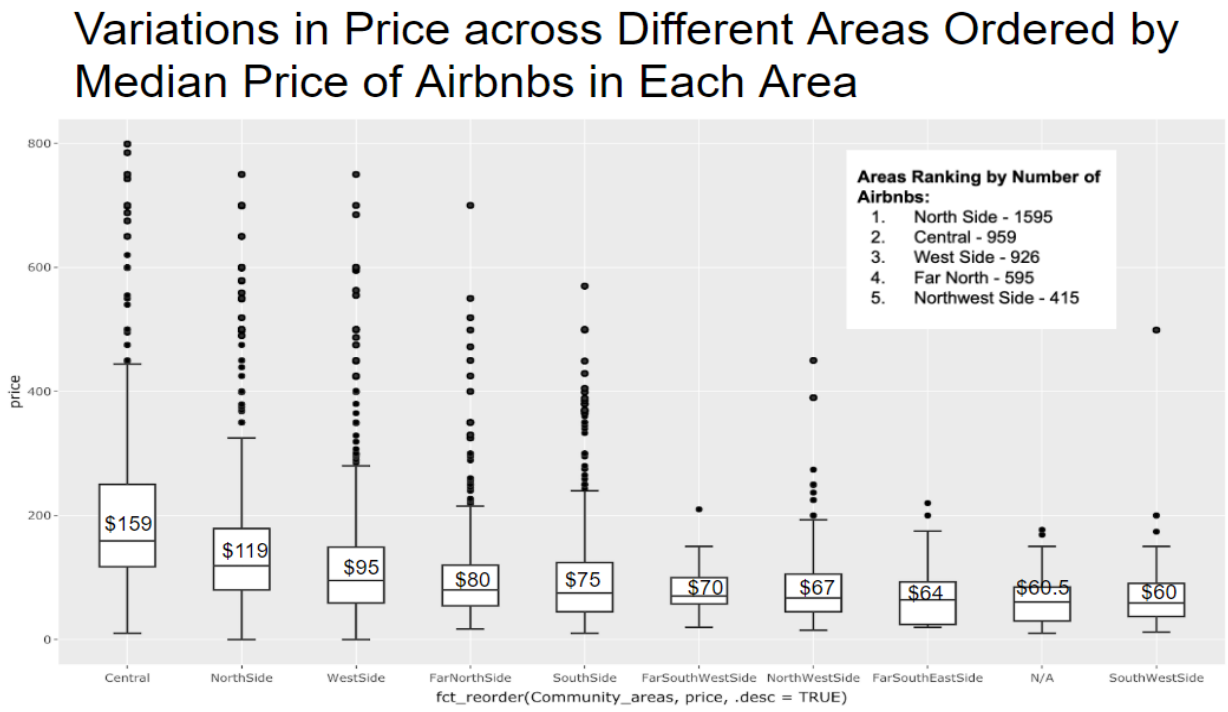
```
#Import original training dataset which includes longtitude & latitude

dfTrainOriginal<-read_csv('airbnbTrain.csv')

#map the airbnb location
library('leaflet')

dfTrainOriginal <-
dfTrainOriginal %>%filter(str_extract(dfTrainOriginal$`{randomControl}`, "[0-9]{3}")=='105')

dfTrainOriginal %>%
  leaflet(height=5000, width=1500) %>%
  addProviderTiles(providers$OpenStreetMap.Mapnik) %>%
  addMarkers(label = dfTrainOriginal$neighbourhood, clusterOptions =
markerClusterOptions())
```



We overlapped the areas map on top of the map with listings. The picture is not that clear, but we listed out the top five popular airbnb areas, they are north side, which have 1600 listings in one area, followed by Central and west side of the chicago. West side of Chicago is what we identified as a high crime rate area, but surprisingly, it is the third most popular

area for airbnb listings. The total listings for this area is combined by two smaller areas, the first part is the area with high crime rate, which has only 178 listings, and the area that is closer to central is what contributes the most to the listings; it has a total of 748 listings. This location is right next to where all the popular museums, and tourist attractions are located. Therefore, this is a very hot area due to being close to Downtown Chicago , despite the high crime rate on the west side.

```
#boxplot for accomodate and price
filterprice<- dfTrain %>% filter(price < 300)
boxPlotsForAll<-

  ggplot(data = filterprice, aes(x = fct_reorder(Community_areas,
price, .desc = TRUE), y = price)) +

  geom_boxplot() +

  labs(title="Variations in Price across Different Areas Ordered by Median
price of Airbnbs in Each Area")


ggplotly(boxPlotsForAll)
```

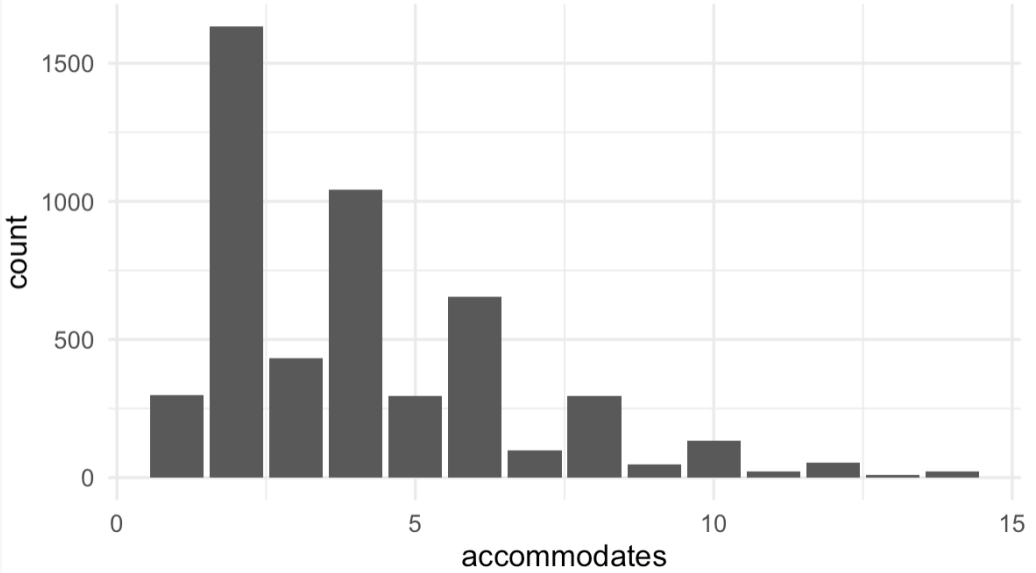


For the boxplot for the price variation across different areas. The median price for central is $159, $40 higher than the north side of the chicago. Median for Unpopular areas like the far southeast side and southwest side is around 65 bucks.

Q3:How does accommodation affect the booking rate of Airbnb/listing price?

```
dfTrain$high_booking_rate <- as.factor(dfTrain$high_booking_rate) #convert to factor
```

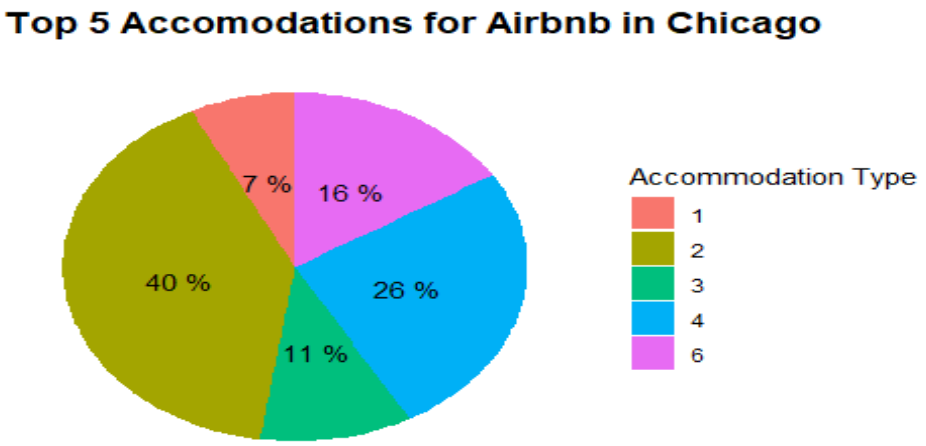
```
plotAcc<-

ggplot(dfTrain) + geom_histogram(mapping = aes(accommodates), stat = "count")
+ theme_minimal(base_size=13)+ ggtitle("The Proportion of Room Type in Each
Area")

ggplotly(plotAcc)
```



We can see that Aribnb properties that accommodate 2 people had the highest proportion of booking rates. We see a pattern of even numbers: 2, 4, 6, 8, having a significantly higher booking rate than odd numbers. This indicates the pattern of people traveling as couples together.

```
top_5_counts_df <- acc_types_counts_df %>% #select top 5 accomodations
   group_by(group) %>%
   tally(value) %>%
   top_n(5, n)

 pie <- ggplot(top_5_counts_df, aes(x = "", y = n, fill = group))+
        geom_bar(width = 1, stat = "identity") +
```

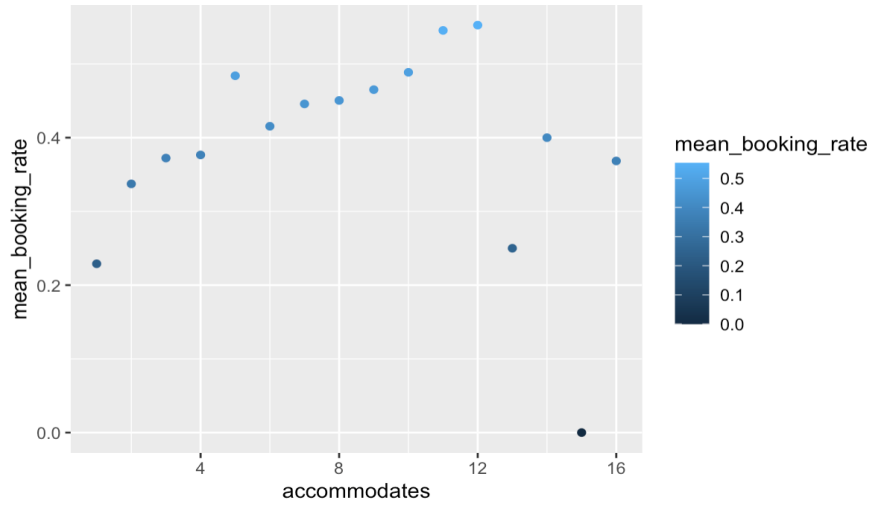
```
          coord_polar("y", start=0) +
          geom_text(aes(label = paste(round(n / sum(n) * 100), "%")),
position = position_stack(vjust = 0.5)) +
          labs(fill = "Accommodation Type",
          x = NULL,
          y = NULL,
          title = "Top 5 Accomodations for Airbnb in Chicago")

  blank_theme <- theme_minimal()+ #create a blank theme
    theme(
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    panel.border = element_blank(),
    panel.grid=element_blank(),
    axis.ticks = element_blank(),
    plot.title=element_text(size=14, face="bold")
  )

  pie <- pie + blank_theme + theme(axis.text.x=element_blank())
  pie
```



The pie chart above shows the top 5 accommodation types for Airbnb in Chicago

```
# boxplot for accomodate and price
filterprice<- dfTrain %>% filter(price < 300)

gg<- filterprice %>%

  group_by(accommodates) %>%

  summarize(mean_booking_rate = mean(as.numeric(high_booking_rate)))
```

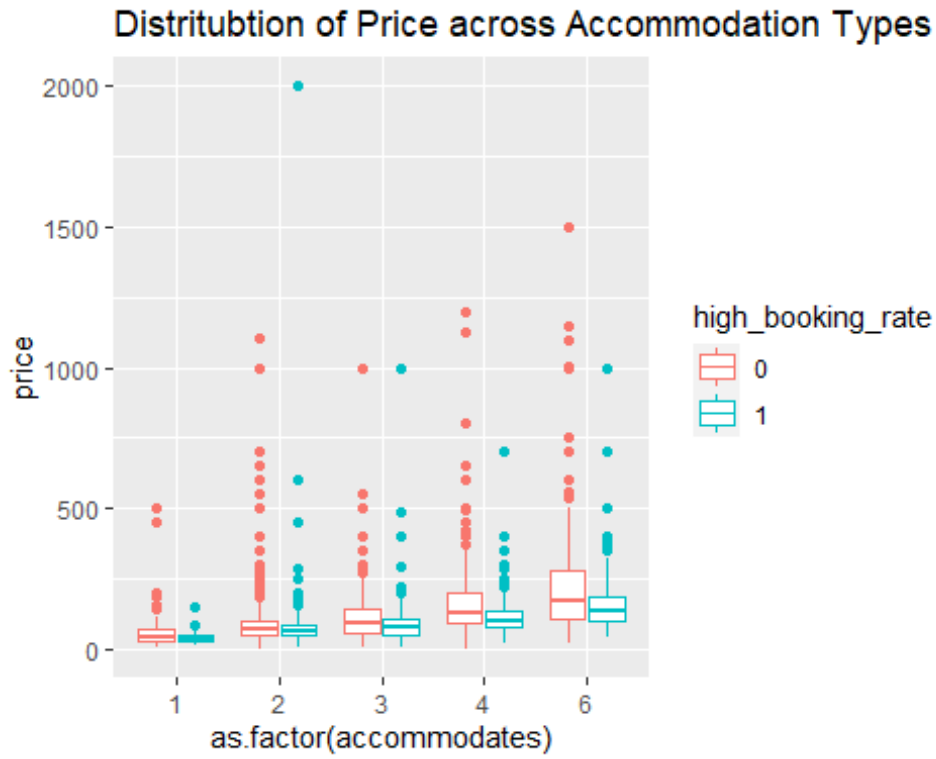
```
gg %>%

  filter(accommodates< 17) %>%

        ggplot(aes(x = accommodates , y= mean_booking_rate,color=
mean_booking_rate )) + geom_point()
```



From the chart above, we can see there is clearly a positive correlation between the probability of being a high booking rate property and the number of accommodates as long as the number of accommodates is lower or equal to 12.

We would like to research the details of the top 5 accommodates.

```
  # Boxplot to show the distribution of price across accommodation types
top_5_data <- dfTrain %>%
  filter(accommodates %in% c(1, 2, 3, 4, 6))%>%
  filter(price < 2500)


bplot <- top_5_data %>%
    ggplot(aes(x = as.factor(accommodates), y = price, color =
high_booking_rate)) +
    geom_boxplot() +
    ggtitle("Distritubtion of Price across Accommodation Types")

bplot
```



According to the boxplot above for the top 5 accommodations, as the accommodation increases, the median price of the high booking rate properties increases slower than the non-high booking rate properties. That means, for the same number of accommodations, the price the host is charging per night becomes a very influential factor in determining whether people will book a property.

Conclusion

Overall, we have concluded that:

1. Certain Chicago neighbourhoods' high crime rate do affect Airbnb properties' booking rate negatively.
2. Popular community areas like, Central, North Side and West Side have price premium over those non-popular community area
3. Accommodation number affects the booking rate of a property positively,though for the same number of accommodations, the property that has a higher price has a lower chance of becoming a high booking rate property.

We recommend our investor to purchase properties in regions other than the West Side and Far SouthEast Side where the crime rate is high. If the investor has purchased houses in popular areas like Central or North Side, the investor could potentially raise their booking price to above $95 a day, far greater than the average price of the whole Chicago.

Because the accommodation of two occupy the most significant proportion of accommodation types, followed by four-person accomodation we recommend the investor to modify their bedrooms to two-person bedrooms and make sure two-person booking is an option.

Afterall, we have built a well-performed random forest classifier with a cutoff that can maximize the prediction precision. We promise that if we predict one property to be positive, we are at least 95% confident that the predicted property is actually a high booking rate property.

Bibliography

Berlatsky, Noah. “How Bad Is Violence in Chicago? Depends on Your Race.” *The Atlantic,* Atlantic Media Company, 29 Sept. 2013, www.theatlantic.com/nationhal/archive/2013/09/how-bad-is-violence-in-chicago-depends-on-your-race/280019/.

Chen, Yingting Sherry. “Interpretation of Kappa Values.” *Medium*, Towards Data Science, 6 July 2019, towardsdatascience.com/interpretation-of-kappa-values-2acd1ca7b18f.

“Community Areas in Chicago.” *Wikipedia*, Wikimedia Foundation, 9 May 2020, en.wikipedia.org/wiki/Community_areas_in_Chicago

Kirin. “Airbnb Host’s Chicago Neighborhood: Most Murders in the City, 6th Worst Overall.” *Launch*, LAUNCH, 24 Aug. 2011, launch.co/blog/airbnb-hosts-chicago-neighborhood-most-murders-in-the-ciy