Investor Report for Airbnb's in San Diego

Group 12

Project Title : Investor Report for Airbnbs in San Diego

Market Assigned to the Team : San Diego

"We, the undersigned, certify that the report submitted is our own original work; all authors participated in the 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."

Name of the signed team members

|Team member 1 | Chinmay Gupta | |Team member 2 | Shruti Sharma | |Team member 3 | Siddhita Bagwe | |Team member 4 | Mansi Kosmakar | |Team member 5 | Harsh Sharma | |Team member 6 | Anuj Doshi |

I Executive Summary

With our analysis, we have developed a business case for an investor who is interested in acquiring homes in San Diego to put them up as AirBnB rentals.For the investor to determine where to invest, we have provided a comprehensive study based on predictive analysis.With the help of statistical modeling, we have considered certain variables which have a strong effect on the investor's decision. The selection of variables was based on certain factors which proved to serve the primary purpose of our project; to maximize the investor's ROI.

Some of the factors that were considered include the discerning of neighbourhoods which would yield high returns.One of our principal findings is that achieving a superhost status would help increase the booking rate. Identification of certain amenities which drive the customers' selection of an Airbnb property in San Diego is a novel realisation in our analysis.

```r
library("tidyverse")
```

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

## v ggplot2 3.3.0     v purrr   0.3.4
## v tibble  3.0.1     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()
```

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

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

```
## v broom     0.5.6      v rsample   0.0.6
## v dials     0.0.6      v tune      0.1.0
## v infer     0.5.1      v workflows 0.1.1
## v parsnip   0.1.0      v yardstick 0.0.6
## v recipes   0.1.12

## -- 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()
```

```r
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
```

```r
library("skimr")
library("caret")
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

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

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

```r
library("lubridate")
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:dplyr':
##
##     intersect, setdiff, union

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library("plyr")
```

```
## ------------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------------

##
## Attaching package: 'plyr'

## The following objects are masked from 'package:plotly':
##
##     arrange, mutate, rename, summarise

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following object is masked from 'package:purrr':
##
##     compact
```

```r
library("Hmisc")
```

```
## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: Formula

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:plyr':
##
##     is.discrete, summarize

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

## The following object is masked from 'package:parsnip':
##
##     translate

## The following objects are masked from 'package:dplyr':
##
##     src, summarize

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```r
library("pheatmap")
library('corrplot')
```

```
## corrplot 0.84 loaded
```

```r
library("glmnet")
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

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

## Loaded glmnet 3.0-2
```

```r
library("tidytext")
library("topicmodels")
library(tokenizers)
```

```r
dfTrain <- read_csv("airbnb_SanDiego_Train.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(),
##   instant_bookable = col_logical(),
##   is_business_travel_ready = col_logical(),
##   is_location_exact = col_logical(),
##   latitude = col_double(),
##   longitude = col_double()
##   # ... with 15 more columns
## )

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

## Warning: 2 parsing failures.
##  row     col               expected actual                       file
## 4636 zipcode no trailing characters  -4131 'airbnb_SanDiego_Train.csv'
## 5698 zipcode no trailing characters  -4131 'airbnb_SanDiego_Train.csv'
```

```r
dfTest <- read_csv("airbnb_SanDiego_Test.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   id = 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(),
##   instant_bookable = col_logical(),
##   is_business_travel_ready = col_logical(),
##   is_location_exact = col_logical(),
##   latitude = col_double(),
##   longitude = col_double(),
##   maximum_nights = col_double()
##   # ... with 14 more columns
## )
## See spec(...) for full column specifications.
```

#feature-selection

```
colsToDrop <- c("weekly_price","zipcode","{randomControl}","host_acceptance_rate","host_has_profile_pic


dfTrain <- dfTrain %>% select(-all_of(colsToDrop))
dfTest <- dfTest %>% select(-all_of(colsToDrop))
```

#Data cleaning

```
dfTrain$neighbourhood <- as.character(dfTrain$neighbourhood)
dfTest$neighbourhood <- as.character(dfTest$neighbourhood)

dfTrain$amenities <- as.character(dfTrain$amenities)
dfTest$amenities <- as.character(dfTest$amenities)

dfTrain$security_deposit <- as.numeric(gsub('\\$|,', '', dfTrain$security_deposit))
dfTest$security_deposit <- as.numeric(gsub('\\$|,', '', dfTest$security_deposit))

dfTrain$cleaning_fee <- as.numeric(gsub('\\$|,', '', dfTrain$cleaning_fee))
dfTest$cleaning_fee <- as.numeric(gsub('\\$|,', '', dfTest$cleaning_fee))

dfTrain$extra_people <- as.numeric(gsub('\\$|,', '', dfTrain$extra_people))
dfTest$extra_people <- as.numeric(gsub('\\$|,', '', dfTest$extra_people))

dfTrain$price <- as.numeric(gsub('\\$|,', '', dfTrain$price))
dfTest$price <- as.numeric(gsub('\\$|,', '', dfTest$price))
```

```
dfTrain$cleaning_fee <- ifelse(is.na(dfTrain$cleaning_fee),0,dfTrain$cleaning_fee)
dfTest$cleaning_fee <- ifelse(is.na(dfTest$cleaning_fee),0,dfTest$cleaning_fee)

dfTrain$extra_people <- ifelse(is.na(dfTrain$extra_people),0,dfTrain$extra_people)
dfTest$extra_people <- ifelse(is.na(dfTest$extra_people),0,dfTest$extra_people)

dfTrain$price <- ifelse(is.na(dfTrain$price),0,dfTrain$price)
dfTest$price <- ifelse(is.na(dfTest$price),0,dfTest$price)

dfTrain$security_deposit <- ifelse(is.na(dfTrain$security_deposit),0,dfTrain$security_deposit)
dfTest$security_deposit <- ifelse(is.na(dfTest$security_deposit),0,dfTest$security_deposit)
```

```
dfTrain$bedrooms = ifelse(is.na(dfTrain$bedrooms), ave(dfTrain$bedrooms, FUN = function(x) median(x, na
dfTrain$beds = ifelse(is.na(dfTrain$beds), ave(dfTrain$beds, FUN = function(x) median(x, na.rm = TRUE))
dfTrain$bathrooms = ifelse(is.na(dfTrain$bathrooms), ave(dfTrain$bathrooms, FUN = function(x) median(x,
dfTrain$host_identity_verified = ifelse(is.na(dfTrain$host_identity_verified),FALSE, dfTrain$host_identi
dfTrain$host_is_superhost = ifelse(is.na(dfTrain$host_is_superhost),FALSE, dfTrain$host_is_superhost)
dfTrain$neighbourhood = ifelse(is.na(dfTrain$neighbourhood),"San Diego", dfTrain$neighbourhood)


dfTest$bedrooms = ifelse(is.na(dfTest$bedrooms), ave(dfTest$bedrooms, FUN = function(x) median(x, na.rm
dfTest$beds = ifelse(is.na(dfTest$beds), ave(dfTest$beds, FUN = function(x) median(x, na.rm = TRUE)), d
dfTest$bathrooms = ifelse(is.na(dfTest$bathrooms), ave(dfTest$bathrooms, FUN = function(x) median(x, na
dfTest$host_identity_verified = ifelse(is.na(dfTest$host_identity_verified),FALSE, dfTest$host_identity_
dfTest$host_is_superhost = ifelse(is.na(dfTest$host_is_superhost),FALSE, dfTest$host_is_superhost)
dfTest$neighbourhood = ifelse(is.na(dfTest$neighbourhood),"San Diego", dfTest$neighbourhood)
```

```
colsToFactor <- c("host_is_superhost","host_identity_verified","instant_bookable","bed_type","cancellati

dfTrain <-  dfTrain %>%  mutate_at(colsToFactor,~factor(.))
dfTrain$high_booking_rate <- as.factor(dfTrain$high_booking_rate)


dfTest <- dfTest %>% mutate_at(colsToFactor,~factor(.))
```

#Loading Data for part two of exploratory analysis

```
dftr <- read.csv("SD_Train_Clean.csv")
#dftr$neighbourhood
#unique(dftr$neighbourhood)
```

```
set.seed(123)
dffTrain<- dftr %>% sample_frac(0.7)
dffTest<- dplyr::setdiff(dftr,dffTrain)
```

```
dft <- read.csv("SD_Test_Clean.csv")
```

#Creating dummy variables

```
dffTrain$room_type_home <- ifelse(dffTrain$room_type == 'Entire home/apt' , 1 , 0)

dffTrain$room_type_shared <- ifelse(dffTrain$room_type == 'Shared room' , 1 , 0)

dffTrain$room_type_pvt <- ifelse(dffTrain$room_type == 'Private room' , 1 , 0)
dffTrain$room_type_hotel <- ifelse(dffTrain$room_type == 'Hotel room' , 1 , 0)

dffTest$room_type_pvt <- ifelse(dffTest$room_type == 'Private room' , 1 , 0)

dffTest$room_type_home <- ifelse(dffTest$room_type == 'Entire home/apt' , 1 , 0)

dffTest$room_type_shared <- ifelse(dffTest$room_type == 'Shared room' , 1 , 0)

dffTest$host_is_superhostTRUE <- ifelse(dffTest$host_is_superhost == TRUE , 1 , 0)
dffTrain$host_is_superhostTRUE <- ifelse(dffTrain$host_is_superhost == TRUE , 1 , 0)
```

```r
dffTrain$cancellation_policystrict_14_with_grace_period <- ifelse(dffTrain$cancellation_policy == 'stri

dffTrain$cancellation_policysuper_strict_60 <- ifelse(dffTrain$cancellation_policy == 'super_strict_60'

dffTrain$host_identity_verifiedTRUE <- ifelse(dffTrain$host_identity_verified == TRUE , 1 , 0)


dffTrain$cancellation_policymoderate   <- ifelse(dffTrain$cancellation_policy == 'moderate' , 1 , 0)

dffTest$cancellation_policystrict_14_with_grace_period <- ifelse(dffTest$cancellation_policy == 'strict_
```

```r
dffTest$cancellation_policysuper_strict_60 <- ifelse(dffTest$cancellation_policy == 'super_strict_60' ,

dffTest$host_identity_verifiedTRUE <- ifelse(dffTest$host_identity_verified == TRUE , 1 , 0)

dffTest$cancellation_policymoderate   <- ifelse(dffTest$cancellation_policy == 'moderate' , 1 , 0)


dffTrain$room_type_home <- ifelse(dffTrain$room_type == 'Entire home/apt' , 1 , 0)

dffTrain$room_type_shared <- ifelse(dffTrain$room_type == 'Shared room' , 1 , 0)

dffTrain$room_type_pvt <- ifelse(dffTrain$room_type == 'Private room' , 1 , 0)
dffTrain$room_type_hotel <- ifelse(dffTrain$room_type == 'Hotel room' , 1 , 0)

dffTest$room_type_pvt <- ifelse(dffTest$room_type == 'Private room' , 1 , 0)

dffTest$room_type_home <- ifelse(dffTest$room_type == 'Entire home/apt' , 1 , 0)

dffTest$room_type_shared <- ifelse(dffTest$room_type == 'Shared room' , 1 , 0)
dffTest$room_type_hotel <- ifelse(dffTest$room_type == 'Hotel room' , 1 , 0)
```

#Understanding the data

```r
dfTrain$amenities <- str_remove_all(dfTrain$amenities,"[{}]")

###Dissociate Words
word_space <- dfTrain %>% select(c(id,amenities)) %>%
  unnest_tokens(word, amenities,token = "regex",pattern=",")

###Remove Stop Words
data(stop_words)
word_space <- word_space %>%
  anti_join(stop_words)
```

```
## Joining, by = "word"
```

```r
#word_space
```

```r
countr <- word_space %>%  group_by(word) %>% tally() %>% arrange(-n)

wt1 <- gsub("[^[:alnum:]]", " ", countr$word) %>% as_tibble()
```

```r
count_freq <- cbind(wt1,countr$n)
colnames(count_freq) <- c("word","freq")
count_freq
```

```
##                                word freq
## 1                              wifi 7956
## 2                        essentials 7794
## 3                   smoke detector  7610
## 4                           kitchen 7378
## 5                                tv 7253
## 6                           hangers 7039
## 7                           heating 7016
## 8         carbon monoxide detector  6742
## 9                           shampoo 6679
## 10                      hair dryer  6469
## 11                             iron 6362
## 12                           washer 6311
## 13                            dryer 6294
## 14       laptop friendly workspace  5936
## 15        free parking on premises  5263
## 16                       hot water  5258
## 17               fire extinguisher  4900
## 18                     refrigerator 4651
## 19                air conditioning  4642
## 20                        microwave 4458
## 21                    coffee maker  4421
## 22           dishes and silverware  4414
## 23             free street parking  3987
## 24                            stove 3893
## 25                  cooking basics  3829
## 26                private entrance  3828
## 27                             oven 3778
## 28                   first aid kit  3739
## 29                      bed linens  3679
## 30                   self check in  3503
## 31             family kid friendly  3138
## 32                        cable tv  3102
## 33            lock on bedroom door  3097
## 34                patio or balcony  2996
## 35     extra pillows and blankets  2929
## 36                       dishwasher 2693
## 37          long term stays allowed  2516
## 38     no stairs or steps to enter  2486
## 39                         internet 2257
## 40                        bbq grill  2044
## 41         luggage dropoff allowed  1873
## 42              garden or backyard  1815
## 43                indoor fireplace  1748
## 44                         hot tub  1671
## 45                             pool 1632
## 46                    pets allowed  1602
## 47                           keypad 1497
## 48              private living room  1421
```

```
## 49                                      elevator 1413
## 50                                       bathtub 1400
## 51                                       lockbox 1322
## 52                            single level home  1311
## 53                                           gym 1247
## 54                             beach essentials  1113
## 55                      pack  n play travel crib  1076
## 56                             24 hour check in  1072
## 57                                  safety card   990
## 58                    well lit path to entrance   927
## 59                               host greets you   784
## 60   translation missing  en hosting amenity 50   780
## 61                                     breakfast  749
## 62                        room darkening shades   654
## 63                    children s books and toys   647
## 64                                   high chair   611
## 65                          ethernet connection   609
## 66   translation missing  en hosting amenity 49   595
## 67                     wide entrance for guests   587
## 68                    pets live on this property   572
## 69                                   smart lock   531
## 70                         extra space around bed   487
## 71                                 wide hallways   481
## 72                   flat path to guest entrance   438
## 73                                 wide entryway   427
## 74                                 wide entrance   404
## 75                           suitable for events   370
## 76                     paid parking off premises   365
## 77                                        dog s   346
## 78                        accessible height bed   344
## 79                              smoking allowed   341
## 80                                    beachfront  326
## 81                        children s dinnerware   325
## 82                     accessible height toilet   308
## 83                        wheelchair accessible   306
## 84                     cleaning before checkout   305
## 85                          handheld shower head   292
## 86                   babysitter recommendations   233
## 87                                          crib  229
## 88              wide doorway to guest bathroom   211
## 89                     paid parking on premises   195
## 90                                  pocket wifi   192
## 91                                outlet covers   186
## 92                               building staff   185
## 93                             fireplace guards   181
## 94                         disabled parking spot   174
## 95                                       toilet   167
## 96                     wide clearance to shower  167
## 97                                  game console   161
## 98                                    waterfront  157
## 99                                  full kitchen   152
## 100                    buzzer wireless intercom   151
## 101                                       cat s   146
## 102                                 stair gates   143
```

```
## 103                          baby bath   121
## 104                bathroom essentials   120
## 105                   bedroom comforts   120
## 106                         bath towel   114
## 107                          body soap   114
## 108                       toilet paper   114
## 109                         ev charger    95
## 110       fixed grab bars for shower    92
## 111                     changing table    78
## 112                      window guards    69
## 113                   hot water kettle    56
## 114                    outdoor seating    55
## 115                        ceiling fan    54
## 116                            doorman   51
## 117                       baby monitor    44
## 118         central air conditioning    40
## 119                           smart tv    39
## 120                       other pet s     35
## 121                       shower chair    35
## 122                           gas oven    33
## 123                  en suite bathroom    32
## 124                table corner guards    32
## 125                     walk in shower    32
## 126       fixed grab bars for toilet    30
## 127                   step free shower    29
## 128                            netflix   29
## 129                            terrace   28
## 130                    breakfast table    26
## 131                  wireless intercom    26
## 132                        kitchenette   26
## 133                        lake access    23
## 134                            balcony   23
## 135           bathtub with bath chair    20
## 136               memory foam mattress    20
## 137                   espresso machine    16
## 138                 formal dining area    15
## 139                       sound system    12
## 140                        rain shower    11
## 141                        soaking tub    11
## 142                       sun loungers    11
## 143                         beach view    10
## 144            electric profiling bed    10
## 145                pillow top mattress    10
## 146                           fire pit     9
## 147                             hbo go     8
## 148                        mini fridge     8
## 149                    convection oven     7
## 150                         dvd player     7
## 151                    outdoor parking     7
## 152                    private hot tub     7
## 153                        wine cooler     7
## 154                        amazon echo     6
## 155                            day bed     6
## 156                      firm mattress     4
```

```
## 157                       pool with pool hoist     4
## 158                             chef s kitchen     3
## 159                                dining area     3
## 160                             heated floors     3
## 161                            outdoor kitchen     3
## 162                          private bathroom     3
## 163                               private pool     3
## 164                             ski in ski out     3
## 165                               air purifier     2
## 166                          alfresco bathtub     2
## 167                              breakfast bar     2
## 168                             garage parking     2
## 169                        ground floor access     2
## 170                                 heat lamps     2
## 171                                ice machine     2
## 172                              ironing board     2
## 173                                 jetted tub     2
## 174                              mountain view     2
## 175                  stand alone steam shower     2
## 176                            warming drawer     2
## 177                                      bidet     2
## 178                                      patio     2
## 179                                    printer     2
## 180                           alfresco shower     1
## 181                               ceiling fans     1
## 182                                double oven     1
## 183                        exercise equipment     1
## 184                                  gas grill     1
## 185                         heated towel rack     1
## 186                                 murphy bed     1
## 187                                  pool toys     1
## 188                          security cameras     1
## 189                                 shared gym     1
## 190                             shared hot tub     1
## 191                                shared pool     1
## 192                            standing valet     1
## 193                              tennis court     1
## 194                                      beach     1
## 195                                    hammock     1
## 196                                      sauna     1
```

```
count_am <- word_space %>%  group_by(id) %>% tally() %>% arrange(-n)
colnames(count_am) <- c("id","count_amenities")
#count_am
```

#varImp

```
dft <- left_join(dfTrain,count_am,by="id") %>% select(-amenities)
dft$count_amenities <- ifelse(is.na(dft$count_amenities),0,dft$count_amenities)

dft$property_type <- as.character(dft$property_type)
```

```r
lambdaValues <- 10^seq(-5, 2, length = 100)
set.seed(123)

fitRidge <-
  train(high_booking_rate ~ .-(id+neighbourhood+property_type), family='binomial', data=dft, method='gl

varImp(fitRidge)$importance %>%    # Add scale=FALSE inside VarImp if you don't want to scale
  rownames_to_column(var = "Variable") %>%
  mutate(Importance = scales::percent(Overall/100)) %>%
  arrange(desc(Overall)) %>%
  as_tibble()
```

```
## # A tibble: 29 x 3
##    Variable                                        Overall Importance
##    <chr>                                             <dbl> <chr>
##  1 cancellation_policyluxury_moderate                100   100.0000%
##  2 cancellation_policystrict                          97.3 97.2612%
##  3 host_is_superhostTRUE                              83.5 83.5083%
##  4 room_typeShared room                               61.5 61.4655%
##  5 cancellation_policysuper_strict_60                 60.6 60.5941%
##  6 bed_typeCouch                                      59.2 59.1622%
##  7 cancellation_policymoderate                        57.6 57.6482%
##  8 cancellation_policysuper_strict_30                 53.3 53.2616%
##  9 cancellation_policystrict_14_with_grace_period     37.5 37.5478%
## 10 host_identity_verifiedTRUE                         33.4 33.3861%
## # ... with 19 more rows
```

```r
lambdaValues <- 10^seq(-5, 2, length = 100)
set.seed(123)

fitLasso2 <-
  train(high_booking_rate ~ .-(id+neighbourhood+property_type), family='binomial', data=dft, method='gl

varImp(fitLasso2)$importance %>%    # Add scale=FALSE inside VarImp if you don't want to scale
  rownames_to_column(var = "Variable") %>%
  mutate(Importance = scales::percent(Overall/100)) %>%
  arrange(desc(Overall)) %>%
  as_tibble()
```

```
## # A tibble: 29 x 3
##    Variable                                        Overall Importance
##    <chr>                                             <dbl> <chr>
##  1 host_is_superhostTRUE                             100   100%
##  2 cancellation_policysuper_strict_30                 86.6 87%
##  3 room_typeShared room                               82.9 83%
##  4 cancellation_policymoderate                        75.8 76%
##  5 bed_typeCouch                                      65.9 66%
##  6 cancellation_policystrict_14_with_grace_period     55.8 56%
##  7 cancellation_policysuper_strict_60                 54.5 55%
##  8 host_identity_verifiedTRUE                         43.3 43%
##  9 room_typePrivate room                              30.1 30%
## 10 bed_typePull-out Sofa                              29.4 29%
## # ... with 19 more rows
```

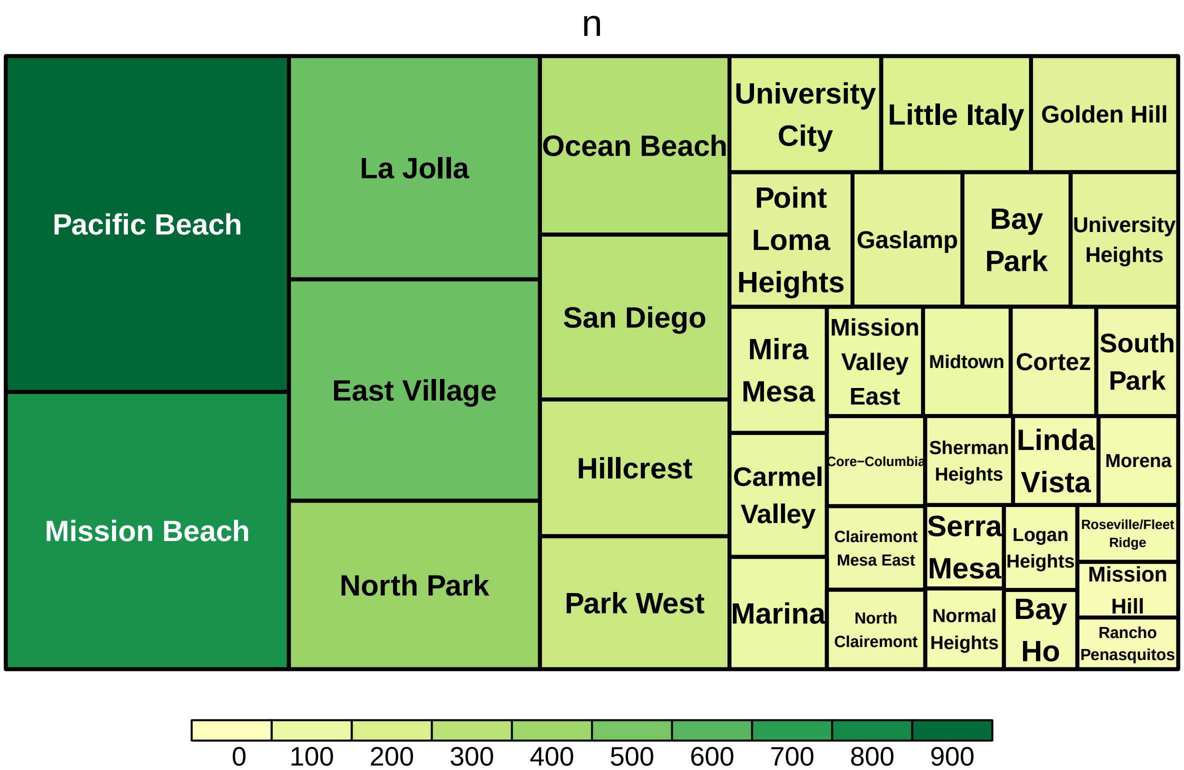
```r
lambdaValues <- 10^seq(-5, 2, length = 100)
set.seed(123)

fitNet <-
  train(high_booking_rate ~ .-(id+neighbourhood+property_type), family='binomial', data=dft, method='gl

varImp(fitNet)$importance %>%    # Add scale=FALSE inside VarImp if you don't want to scale
  rownames_to_column(var = "Variable") %>%
  mutate(Importance = scales::percent(Overall/100)) %>%
  arrange(desc(Overall)) %>%
  as_tibble()
```

```
## # A tibble: 29 x 3
##    Variable                                         Overall Importance
##    <chr>                                              <dbl> <chr>
##  1 cancellation_policystrict                          100   100.0000%
##  2 host_is_superhostTRUE                               97.9 97.9236%
##  3 cancellation_policysuper_strict_30                  89.6 89.5887%
##  4 room_typeShared room                                86.1 86.1317%
##  5 bed_typeCouch                                       76.7 76.7354%
##  6 cancellation_policymoderate                         75.1 75.1424%
##  7 cancellation_policystrict_14_with_grace_period      55.6 55.5992%
##  8 cancellation_policysuper_strict_60                  54.6 54.5723%
##  9 host_identity_verifiedTRUE                          43.1 43.1072%
## 10 bed_typePull-out Sofa                               35.1 35.1061%
## # ... with 19 more rows
```

#treemap

```r
dfPlotTM <-
  dfTrain %>% select(c("neighbourhood"))%>%group_by(neighbourhood) %>% tally() %>%filter(n>=50) %>%  ar
```

```r
library(treemap)

# treemap
treemap(dfPlotTM,
            index = "neighbourhood",
            vSize = "n",
            type = "value",
            vColor = "n" ,
            title.legend="Treemap for Variable: Neighbourhood"
            )
```



Treemap for Variable: Neighbourhood

```
dfPlotPT <-
  dfTrain %>% select(c("property_type"))%>%group_by(property_type) %>% tally() %>% arrange(desc(n))
```

```
dfFull <- read_csv("airbnb_SanDiego_Train.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(),
##   instant_bookable = col_logical(),
##   is_business_travel_ready = col_logical(),
```

```
##   is_location_exact = col_logical(),
##   latitude = col_double(),
##   longitude = col_double()
##   # ... with 15 more columns
## )

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

## Warning: 2 parsing failures.
##  row     col               expected actual                        file
## 4636 zipcode no trailing characters  -4131 'airbnb_SanDiego_Train.csv'
## 5698 zipcode no trailing characters  -4131 'airbnb_SanDiego_Train.csv'
```

#map-1

```r
#devtools::install_github("dkahle/ggmap",force = TRUE)
library("ggmap")
```

```
## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.

## Please cite ggmap if you use it! See citation("ggmap") for details.

##
## Attaching package: 'ggmap'

## The following object is masked from 'package:plotly':
##
##     wind
```

```r
# this sets your google map for this session
register_google(key = "AIzaSyCc7DdojQhaonak2e6Fo8zrSIP2xSkyNDY")

# this sets your google map permanently
register_google(key = "AIzaSyCc7DdojQhaonak2e6Fo8zrSIP2xSkyNDY", write = TRUE)
```

```
## Replacing old key (AIzaSyCc7DdojQhaonak2e6Fo8zrSIP2xSkyNDY) with new key in C:\Users\cvg10\Documents/
```

```r
has_google_key()
```

```
## [1] TRUE
```

```r
google_key()
```

```
## [1] "AIzaSyCc7DdojQhaonak2e6Fo8zrSIP2xSkyNDY"
```

```r
has_google_client()
```

```
## [1] FALSE
```

```r
has_google_signature()
```

```
## [1] FALSE
```

```r
library(leaflet)
```

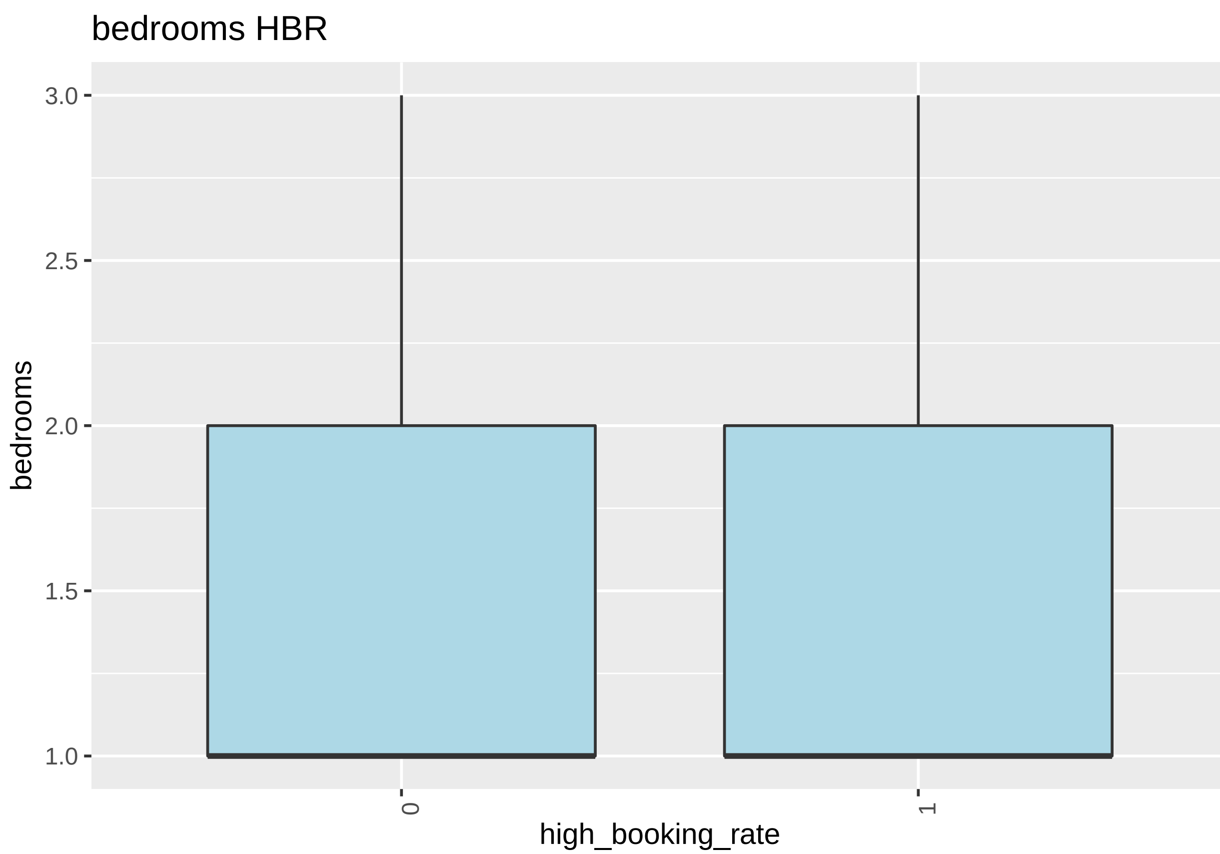
```r
dfLoc <- dfFull %>% select(c("latitude","longitude","high_booking_rate"))
```

```r
#dfLoc$high_booking_rate <- factor(dfLoc$high_booking_rate, TRUE)


hbr1 <- dfLoc %>% filter(high_booking_rate==1)
hbr0 <- dfLoc %>% filter(high_booking_rate==0)
factpal <- colorFactor(c("Blue","Red"), dfLoc$high_booking_rate)


leaflet(dfLoc) %>% setView(lng = -117.161087, lat = 32.715736, zoom = 12) %>%
  addTiles() %>%
  addPolygons(lat =32.715736 ,lng = -117.161087 , color = "#444444", weight = 1) %>%
  addCircleMarkers(  lng = hbr0$longitude,
                     lat = hbr0$latitude,
                     radius = 2,
                     stroke = FALSE,
                     color = "blue",
                     fillOpacity = 0.5
                     )  %>%
  addCircleMarkers(  lng = hbr1$longitude,
                     lat = hbr1$latitude,
                     radius = 2,
                     stroke = FALSE,
                     color = "red",
                     fillOpacity = 0.9
                     )  %>%

  addLegend("bottomleft",pal=factpal ,values = ~high_booking_rate)
```

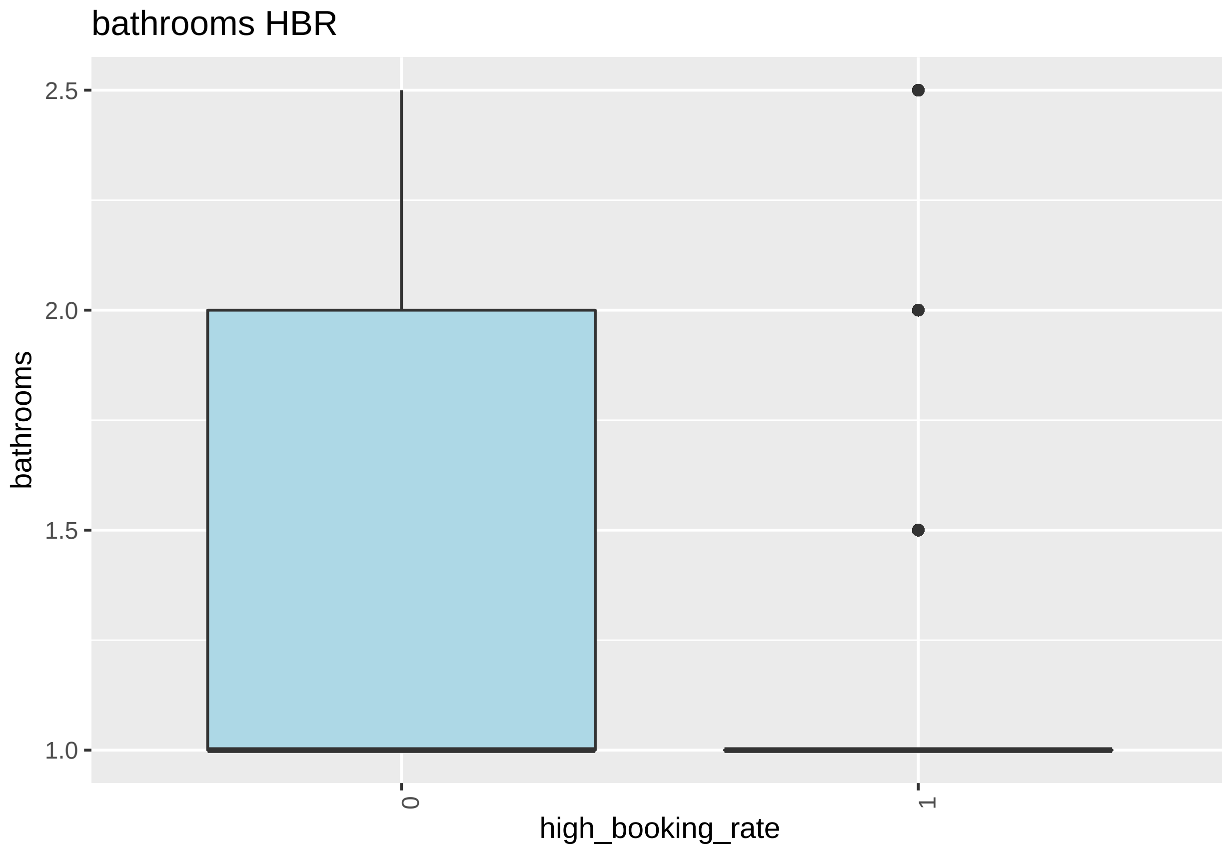
#box-plots

```r
cont <- c("bedrooms","bathrooms","beds","extra_people","guests_included","security_deposit","count_amen
for (colname in cont){
  plot <- ggplot(data=dft, aes(x=high_booking_rate, y=dft[[colname]])) +
      geom_boxplot(fill="lightblue") + labs(x = "high_booking_rate", y = colname,
            title = paste(colname, "HBR")) +
            theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
            scale_y_continuous(limits = quantile(dft[[colname]], c(0.1, 0.9)))
  print(plot)
}
```

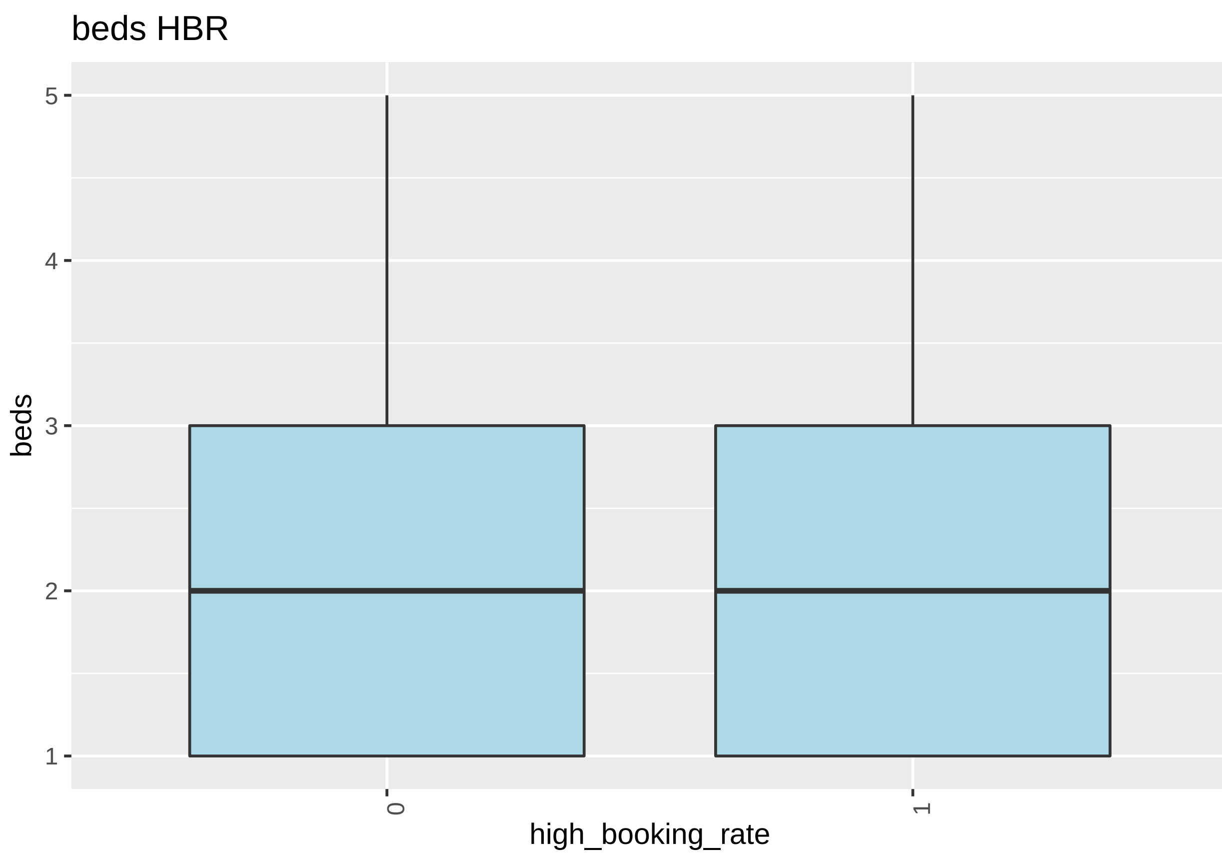
```
## Warning: Removed 1340 rows containing non-finite values (stat_boxplot).
```



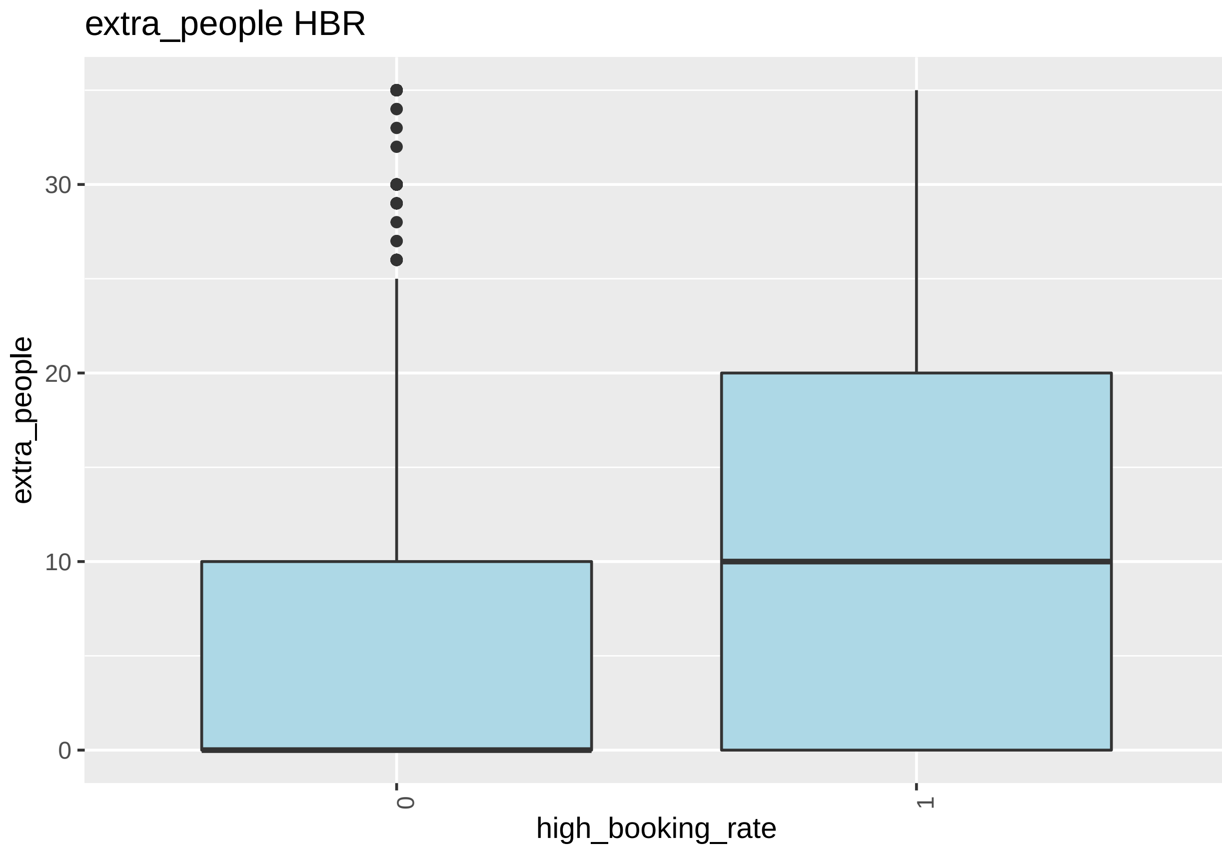

```
## Warning: Removed 738 rows containing non-finite values (stat_boxplot).
```



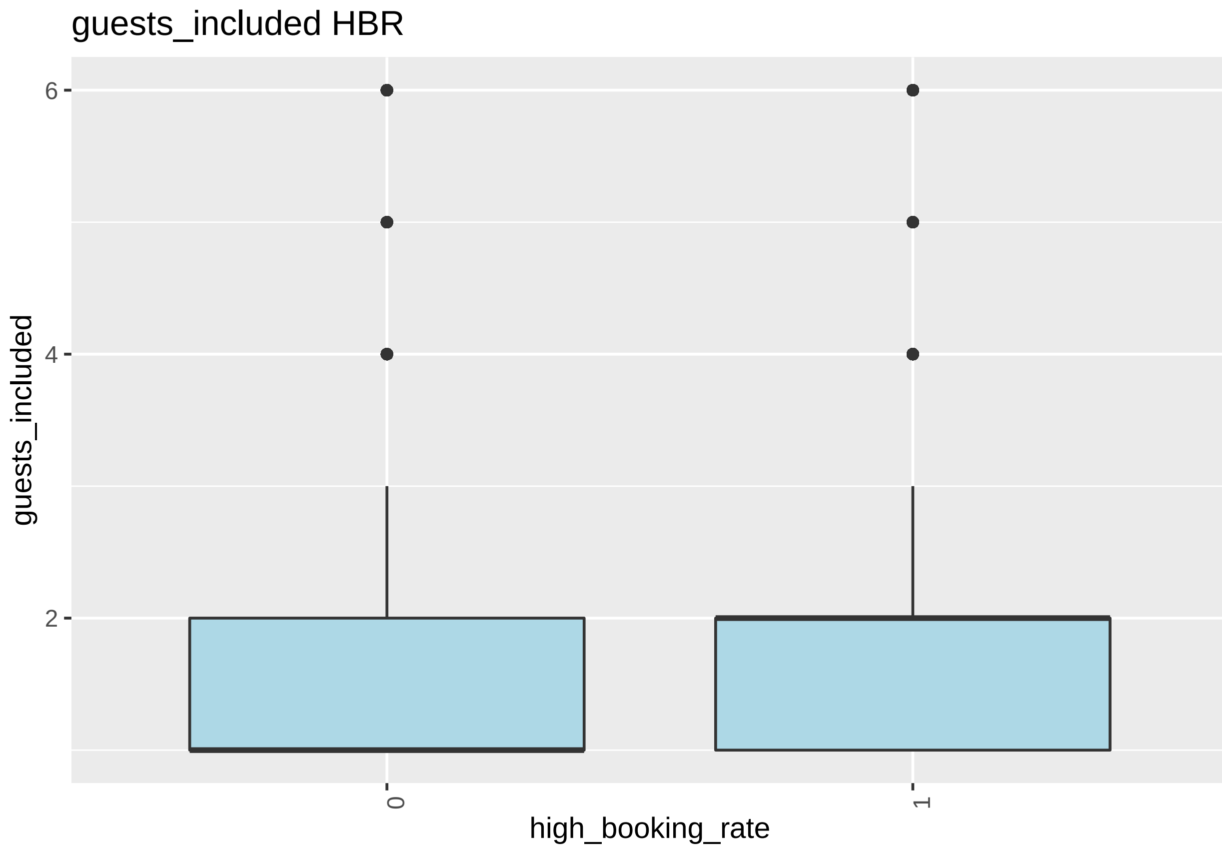

```
## Warning: Removed 649 rows containing non-finite values (stat_boxplot).
```



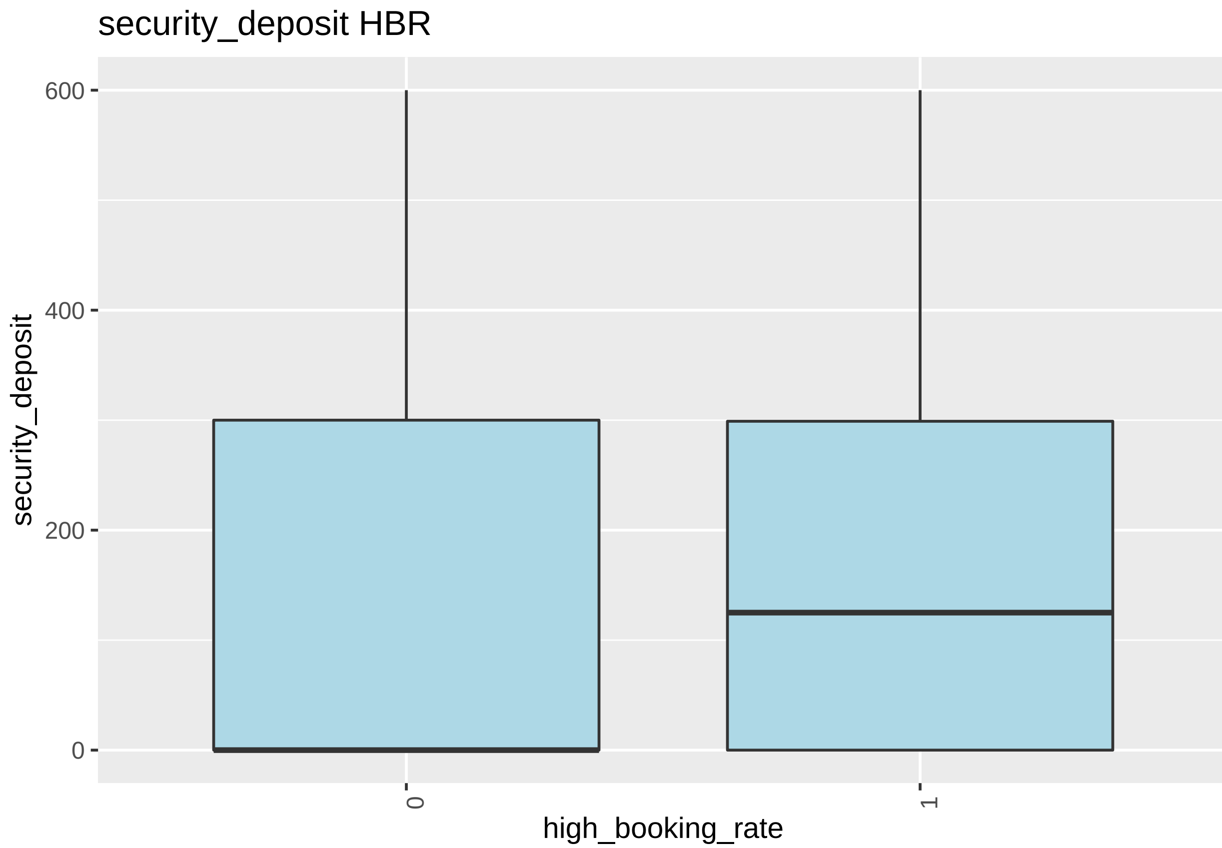

```
## Warning: Removed 766 rows containing non-finite values (stat_boxplot).
```



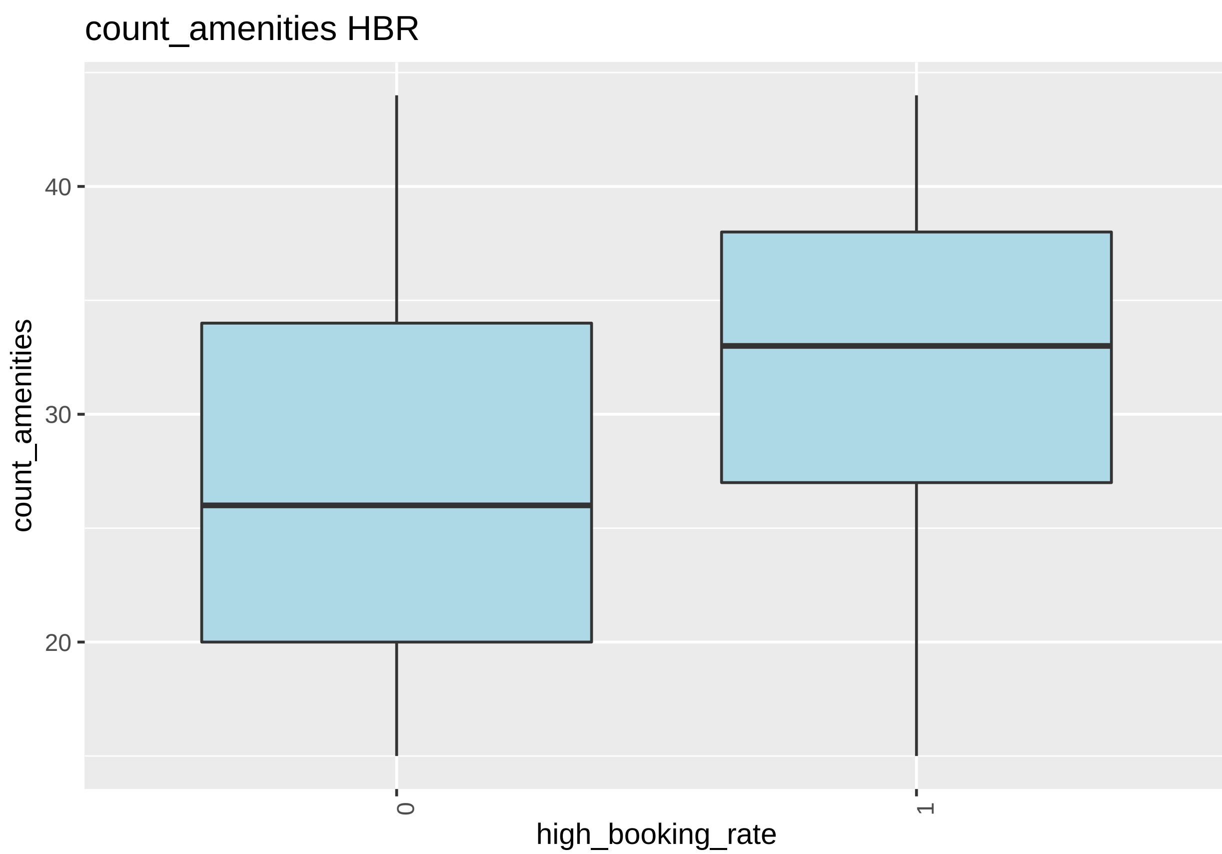

```
## Warning: Removed 480 rows containing non-finite values (stat_boxplot).
```



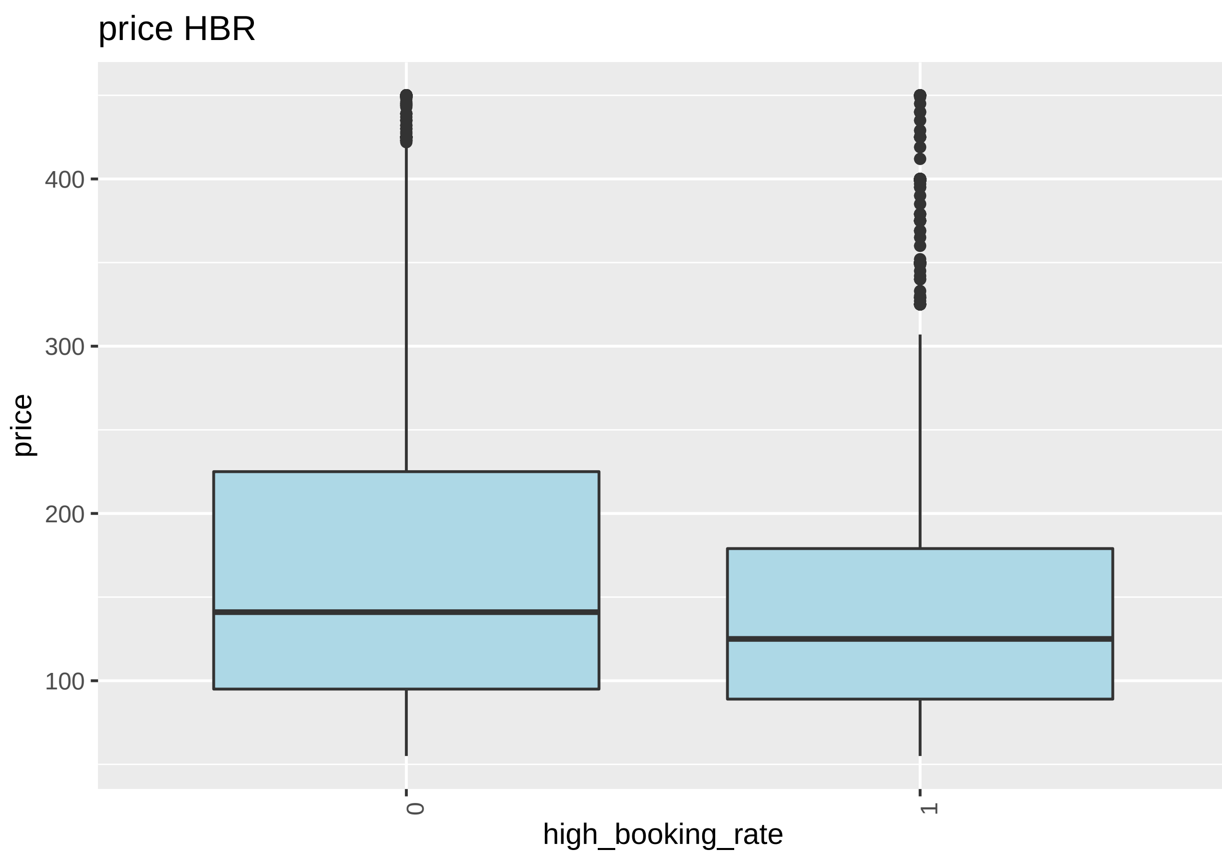

```
## Warning: Removed 778 rows containing non-finite values (stat_boxplot).
```




```
## Warning: Removed 1551 rows containing non-finite values (stat_boxplot).
```




```
## Warning: Removed 1597 rows containing non-finite values (stat_boxplot).
```



#word_cloud

```r
library("tm")
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```r
library("wordcloud")
```

```
## Loading required package: RColorBrewer
```

```r
library("factoextra")
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```r
library("NbClust")
```

Part 2

II Research Questions 1. Can the price alone determine the High Booking Rates?

For this question, first we determined the correlation between all the important features by plotting a heatmap. After that we specifically chose price and high_booking_rate to numerically measure the correlation between them. The correlation was low and negative. To further discover the relation between these two features, we divided the data set in two categories. One with high_booking_rate value equal to 1 and one with high_booking_rate value equals 0. We plotted the scatter plot and box plot of the divided data against price. After that we concluded that prices don't determine booking rates. Also, the properties which have higher booking rates have a median price of $120.

```r
dfTrain2 <- read.csv("SD_Train_Clean.csv")
```

Find Correlation between the variables

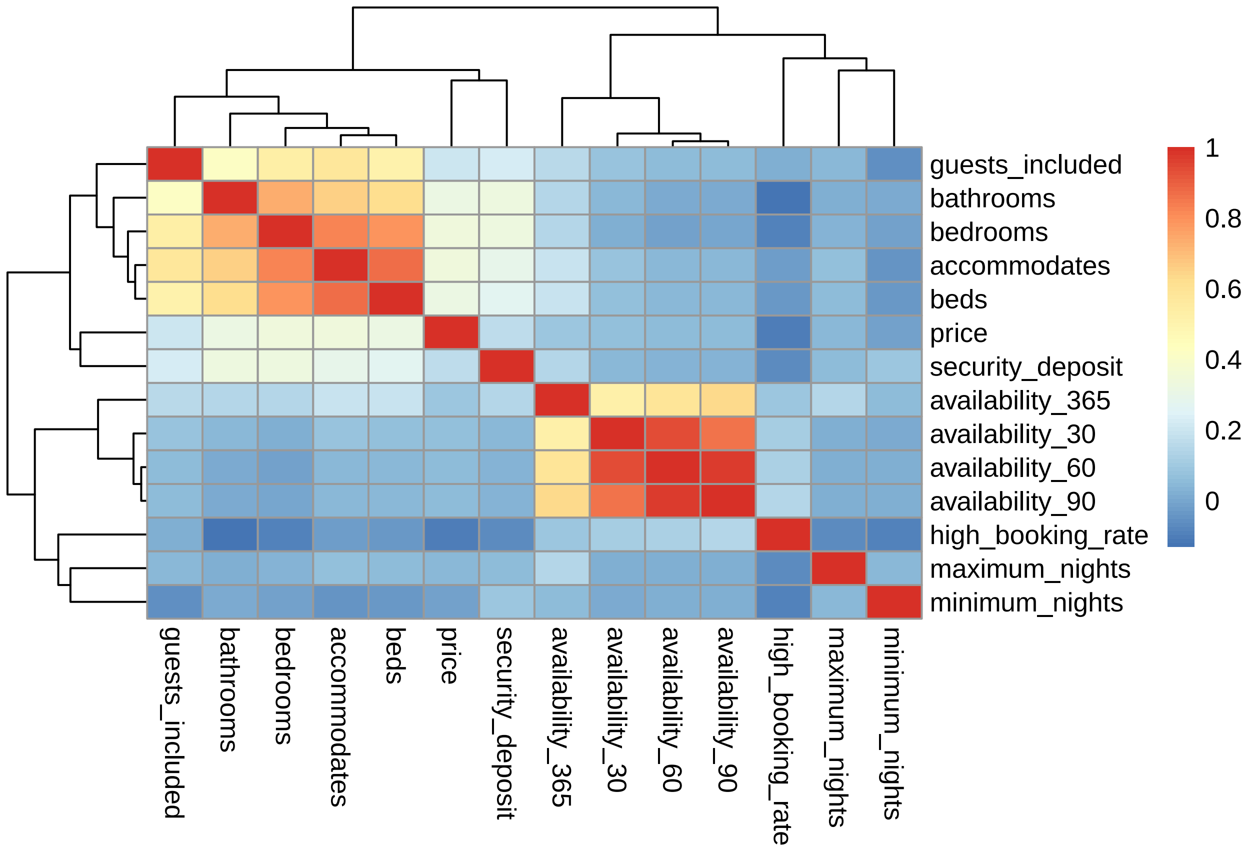
```r
dfcorr <- dfTrain2 %>% select(high_booking_rate,price,accommodates,availability_30,availability_365,ava
cor(dfcorr)
```

```
##                   high_booking_rate       price accommodates availability_30
## high_booking_rate        1.00000000 -0.10110641  -0.02940139      0.10783093
## price                   -0.10110641  1.00000000   0.34207732      0.06927464
## accommodates            -0.02940139  0.34207732   1.00000000      0.07360081
## availability_30          0.10783093  0.06927464   0.07360081      1.00000000
## availability_365         0.09344066  0.08900469   0.19412822      0.51655635
## availability_60          0.11942998  0.05892950   0.04284433      0.93321004
## availability_90          0.14954705  0.05011868   0.04883359      0.86139487
## bathrooms               -0.13180958  0.31919169   0.65681793      0.04274660
## bedrooms                -0.09531985  0.33477288   0.82560532      0.02408691
## maximum_nights          -0.07051451  0.03986246   0.06321031      0.02133981
## minimum_nights          -0.09174419 -0.01832858  -0.04826501      0.01218375
## security_deposit        -0.06499495  0.16807762   0.29465447      0.04218440
## guests_included          0.01737128  0.19695487   0.57610424      0.07272869
## beds                    -0.04049593  0.31759544   0.86559948      0.06396822
##                   availability_365 availability_60 availability_90    bathrooms
## high_booking_rate       0.09344066      0.11942998     0.149547051 -0.131809579
## price                   0.08900469      0.05892950     0.050118679  0.319191695
## accommodates            0.19412822      0.04284433     0.048833595  0.656817933
## availability_30         0.51655635      0.93321004     0.861394873  0.042746595
## availability_365        1.00000000      0.58244864     0.632688800  0.145007774
## availability_60         0.58244864      1.00000000     0.966221439  0.010358513
## availability_90         0.63268880      0.96622144     1.000000000  0.008048044
## bathrooms               0.14500777      0.01035851     0.008048044  1.000000000
## bedrooms                0.14179023     -0.01092082    -0.006337649  0.734937083
## maximum_nights          0.14263875      0.01899571     0.018196587  0.019475213
## minimum_nights          0.05996173      0.02546331     0.026185579  0.012779220
## security_deposit        0.14262350      0.02918267     0.028095322  0.326595421
## guests_included         0.15642638      0.05050846     0.057989610  0.415661814
## beds                    0.18742757      0.03819823     0.045928953  0.624543620
##                       bedrooms maximum_nights minimum_nights security_deposit
```

```
## high_booking_rate -0.095319852    -0.07051451    -0.09174419      -0.06499495
## price              0.334772875     0.03986246    -0.01832858       0.16807762
## accommodates       0.825605319     0.06321031    -0.04826501       0.29465447
## availability_30    0.024086906     0.02133981     0.01218375       0.04218440
## availability_365   0.141790231     0.14263875     0.05996173       0.14262350
## availability_60   -0.010920821     0.01899571     0.02546331       0.02918267
## availability_90   -0.006337649     0.01819659     0.02618558       0.02809532
## bathrooms          0.734937083     0.01947521     0.01277922       0.32659542
## bedrooms           1.000000000     0.02751624    -0.01550134       0.32954610
## maximum_nights     0.027516241     1.00000000     0.04737957       0.05590002
## minimum_nights    -0.015501335     0.04737957     1.00000000       0.08558506
## security_deposit   0.329546099     0.05590002     0.08558506       1.00000000
## guests_included    0.529368565     0.04361228    -0.05824978       0.22580914
## beds               0.792901621     0.05211988    -0.03313733       0.27209844
##                   guests_included        beds
## high_booking_rate      0.01737128 -0.04049593
## price                  0.19695487  0.31759544
## accommodates           0.57610424  0.86559948
## availability_30        0.07272869  0.06396822
## availability_365       0.15642638  0.18742757
## availability_60        0.05050846  0.03819823
## availability_90        0.05798961  0.04592895
## bathrooms              0.41566181  0.62454362
## bedrooms               0.52936856  0.79290162
## maximum_nights         0.04361228  0.05211988
## minimum_nights        -0.05824978 -0.03313733
## security_deposit       0.22580914  0.27209844
## guests_included        1.00000000  0.50359859
## beds                   0.50359859  1.00000000
```

#Heatmap

```r
pheatmap(cor(dfcorr))
```



Data with price and High Booking Rate

```
dfbp <- dfTrain2 %>% select(high_booking_rate,price)
```

Correlation between Price and High Booking Rate

```
cor(dfbp)
```

```
##                   high_booking_rate      price
## high_booking_rate         1.0000000 -0.1011064
## price                    -0.1011064  1.0000000
```

```
dfTrain$high_booking_rate<- as.factor(dfTrain2$high_booking_rate)
```

Dataset with only High Booking rate and Price

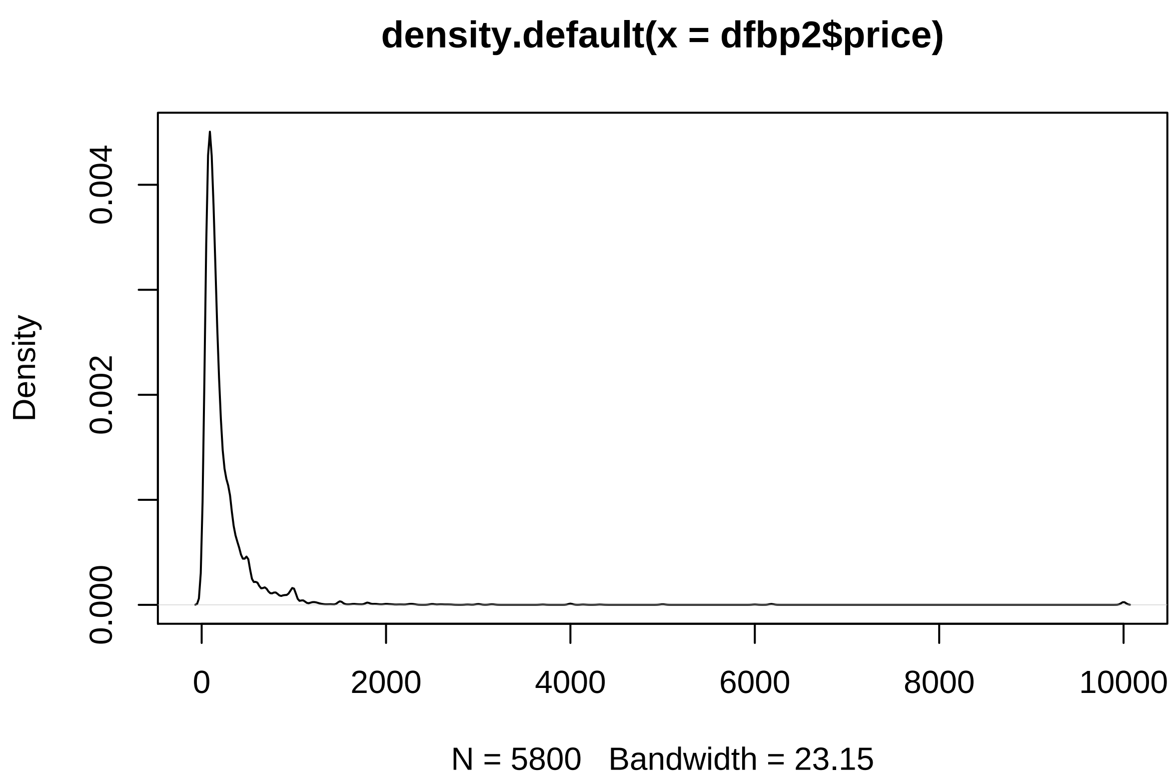
```
dfbp1 <- dfbp[ which(dfbp$high_booking_rate==1), ]
rownames(dfbp1) <- seq(length=nrow(dfbp1))
```

Dataset with only Low booking rate and price

```
dfbp2 <- dfbp[ which(dfbp$high_booking_rate==0), ]
rownames(dfbp2) <- seq(length=nrow(dfbp2))
```

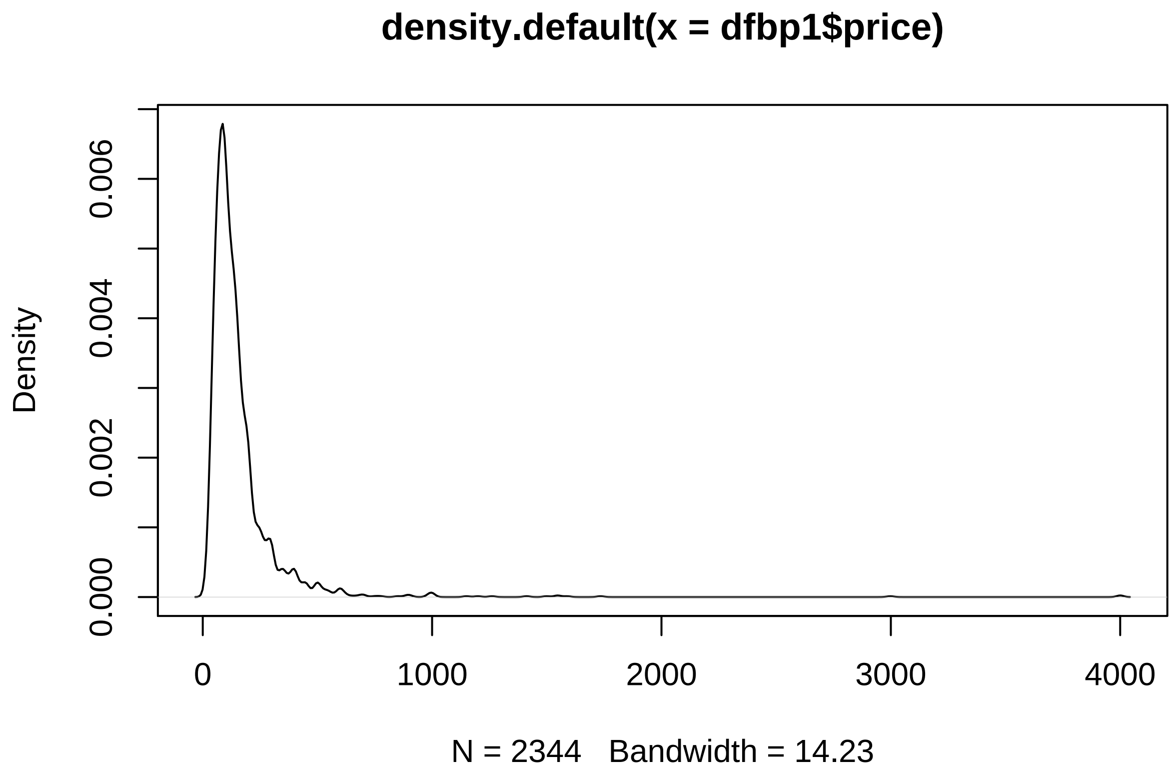
Plot Data with Low Booking Rate

```
d <- density(dfbp2$price)
plot (d)
```



Plot data with high Booking rate

```r
d <- density(dfbp1$price)
plot (d)
```



2. *Is the Booking Rate of a property dependent on how "Accommodative" an AirBnb property is?*

For this question first we selected only those variables that according to our domain knowledge are most closely related to the accommodation at a particular property. These variables are High_booking_rate (DV), accommodates, availability_30, availability_365, availability_60, availability_90, bathrooms, bedrooms, maximum_nights, minimum_nights, guests_included, beds.

After selecting variables, we ran a logistic regression model with high_booking_rate as DV. In the summary of the regression model, we found out that accommodates,availability_30, availability_365,availability_60, availability_90, bathrooms, bedrooms, maximum_nights, minimum_nights, guests_included - these variables are necessary for determining high_booking_rate as per their p-values. We determined the accuracy of the model using a confusion matrix. From this model, we concluded that for an Airbnb property to have higher booking rates, it must be flexible with respect to its booking duration and be able to provide accommodation to the guests.

Logistic Regression Model on variables related to Property

```r
dfm1 <- dfTrain2 %>% select(high_booking_rate,accommodates,availability_30,availability_365,availabilit

dfcTrain <- dfm1 %>% sample_frac(0.65)
```

```r
dfcTest <- dplyr::setdiff(dfm1, dfcTrain)

dfcTrain$high_booking_rate <- as.factor(dfcTrain$high_booking_rate)
dfcTest$high_booking_rate <- as.factor(dfcTest$high_booking_rate)
```

```r
fit_glm <- glm(formula = high_booking_rate~., family='binomial',dfcTrain)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
summary(fit_glm)
```

```
##
## Call:
## glm(formula = high_booking_rate ~ ., family = "binomial", data = dfcTrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4796  -0.8620  -0.6367   1.2003   3.0765
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -3.886e-01  9.441e-02  -4.116 3.85e-05 ***
## accommodates      7.019e-02  2.481e-02   2.829  0.00466 **
## availability_30   3.602e-02  8.931e-03   4.033 5.50e-05 ***
## availability_365  7.364e-04  3.253e-04   2.264  0.02356 *
## availability_60  -6.355e-02  9.254e-03  -6.867 6.57e-12 ***
## availability_90   3.761e-02  4.502e-03   8.354  < 2e-16 ***
## bathrooms        -6.793e-01  7.602e-02  -8.936  < 2e-16 ***
## bedrooms         -1.581e-01  5.793e-02  -2.730  0.00633 **
## maximum_nights   -3.360e-04  6.139e-05  -5.472 4.45e-08 ***
## minimum_nights   -5.706e-02  7.392e-03  -7.719 1.17e-14 ***
## guests_included   6.984e-02  1.773e-02   3.938 8.21e-05 ***
## beds              2.409e-02  3.611e-02   0.667  0.50473
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6316.9  on 5293  degrees of freedom
## Residual deviance: 5795.6  on 5282  degrees of freedom
## AIC: 5819.6
##
## Number of Fisher Scoring iterations: 6
```

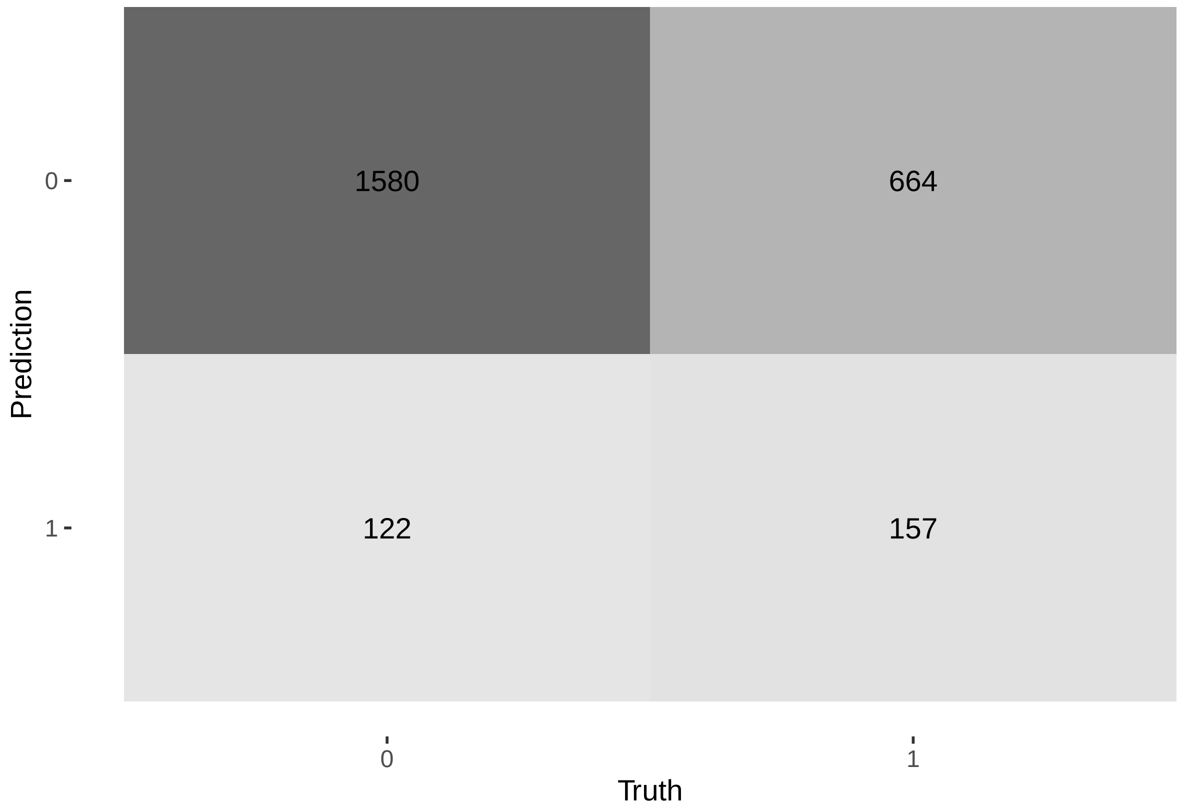
Model Accuracy and COnfusion Matrix

```r
resultsLPM <-
    glm(high_booking_rate~.,family='binomial',data=dfcTest) %>%
    predict(dfcTest,type='response' ) %>%
```

```r
    bind_cols(dfcTest, predictedProb=.) %>%
    mutate(predictedClass = as.factor(ifelse(predictedProb>0.5,1,0)))

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

```
## Confusion Matrix and Statistics
##
##               high_booking_rate
## predictedClass    0    1
##              0 1580  664
##              1  122  157
##
##                Accuracy : 0.6885
##                  95% CI : (0.67, 0.7065)
##     No Information Rate : 0.6746
##     P-Value [Acc > NIR] : 0.07093
##
##                   Kappa : 0.1442
##
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 0.19123
##             Specificity : 0.92832
##          Pos Pred Value : 0.56272
##          Neg Pred Value : 0.70410
##              Prevalence : 0.32541
##          Detection Rate : 0.06223
##    Detection Prevalence : 0.11058
##       Balanced Accuracy : 0.55977
##
##        'Positive' Class : 1
##
```

```r
resultsLPM %>% conf_mat(truth=high_booking_rate,estimate=predictedClass) %>%
autoplot(type='heatmap')
```



3. Is High Booking Rate affected by various charges included with price?

In this question, we tried to determine whether the factors which add to the price can affect the high_booking_rate. For this we selected the variables like cleaning fee, extra people fee, price, security deposit. After selecting features, we ran a regression model with high_booking_rate as a dependent variable. From the model summary, we can see that the cleaning fee and the security deposit barely matter for high_booking_rate. Factors like price and extra people fee are the statistically important variables. Also when we determine the accuracy of the model using a confusion matrix, the accuracy was 68%. So we concluded that (i) If a property provides for extra_people despite charging for the same - its booking rate improves, (ii) Cleaning fee and security deposits do not have any impact on the higher booking rates (iii) If the price of a property is high, its booking rate decreases.

Logistic Regression Model on variables related to Price

```
dfm2 <- dft %>% select(high_booking_rate,cleaning_fee,extra_people,price,security_deposit)

dfcTrain <- dfm2 %>% sample_frac(0.70)
dfcTest <- dplyr::setdiff(dfm2, dfcTrain)

dfcTrain$high_booking_rate <- as.factor(dfcTrain$high_booking_rate)
dfcTest$high_booking_rate <- as.factor(dfcTest$high_booking_rate)
```

```r
fit_glm <- train(high_booking_rate~., family='binomial',dfcTrain,method = "glm")
summary(fit_glm)
```

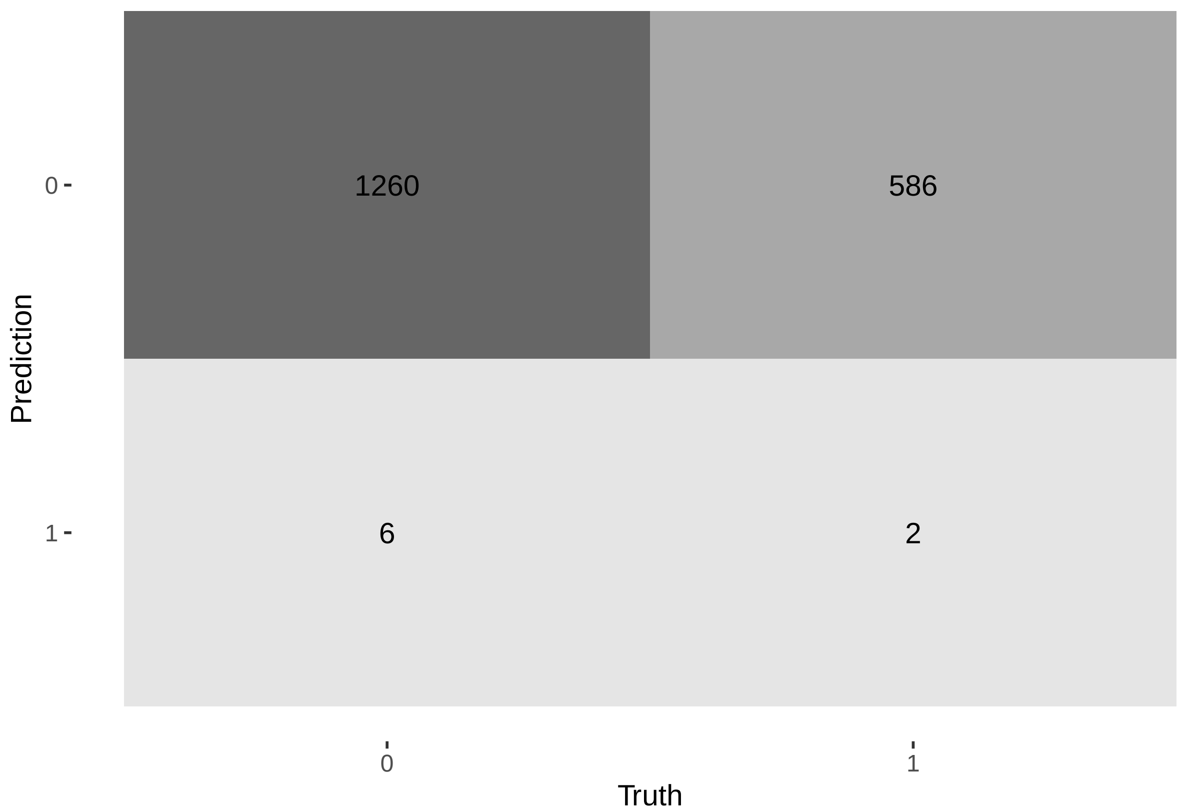
```
##
## Call:
## NULL
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.638  -0.886  -0.771   1.430   3.774
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -5.464e-01  4.707e-02 -11.607  < 2e-16 ***
## cleaning_fee     -1.110e-03  4.681e-04  -2.372   0.0177 *
## extra_people      6.106e-03  1.125e-03   5.427 5.73e-08 ***
## price            -1.630e-03  2.402e-04  -6.787 1.14e-11 ***
## security_deposit -7.507e-05  8.195e-05  -0.916   0.3596
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6870.6  on 5700  degrees of freedom
## Residual deviance: 6672.4  on 5696  degrees of freedom
## AIC: 6682.4
##
## Number of Fisher Scoring iterations: 6
```

```r
resultsLPM <- fit_glm %>%
    predict(dfcTest,type='raw') %>%
    bind_cols(dfcTest, predictedClass=.)

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

```
## Confusion Matrix and Statistics
##
##               high_booking_rate
## predictedClass    0    1
##              0 1260  586
##              1    6    2
##
##                Accuracy : 0.6807
##                  95% CI : (0.6589, 0.7019)
##     No Information Rate : 0.6828
##     P-Value [Acc > NIR] : 0.59
##
##                   Kappa : -0.0018
##
##  Mcnemar's Test P-Value : <2e-16
```

```
##
##             Sensitivity : 0.003401
##             Specificity : 0.995261
##          Pos Pred Value : 0.250000
##          Neg Pred Value : 0.682557
##              Prevalence : 0.317152
##          Detection Rate : 0.001079
##    Detection Prevalence : 0.004315
##       Balanced Accuracy : 0.499331
##
##        'Positive' Class : 1
##
```

```r
resultsLPM %>% conf_mat(truth=high_booking_rate,estimate=predictedClass) %>%
autoplot(type='heatmap')
```



#XGBoost

4. Does being superhost affect booking rate?

From our XG boost model, we saw that the host_is_superhost is a really important variable. So, we analyzed this variable further. Selected the superhost and high booking rate columns, filtered the high booking rate = 1 then, grouped by host_is_superhost and got the count of booking rate =1 for both groups (superhost = true and supehost = false). The results of this query shows that Properties with a superhost has a higher chance of having a high booking rate.

```r
df_sd_train <- dft %>% sample_frac(.65)
df_sd_test <- dplyr::setdiff(dft, df_sd_train)
```

```r
set.seed(2020)

fitXGBoost <- train(high_booking_rate ~ ., data=df_sd_train, method='xgbTree')

#See the CV output (accuracy per pruning parameter etc.)
fitXGBoost$results %>%
  arrange(-Accuracy)

#See the variables plotted by importance (according to the bagged tree):
plot(varImp(fitXGBoost), top=20)

#See the variables listed by importance (according to the bagged tree)
varImp(fitXGBoost)$importance %>%    # Add scale=FALSE inside VarImp if you don't want to scale
  rownames_to_column(var = "Variable") %>%
  mutate(Importance = scales::percent(Overall/100)) %>%
  arrange(desc(Overall)) %>%
  as_tibble()

#Make predictions:
resultsXGBoost <-
  fitXGBoost %>%
  predict(df_sd_test, type='raw') %>%
  bind_cols(df_sd_test, predictedClass=.)

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

5. *Which neighbourhood results in a higher booking rate?*

Further to determine which neighbourhoods are most likely to get high_booking_rate , we plot a map. On the map we plotted the circles whose area is basically dependent on the number of high_booking_rate properties in that area. So when we plotted we found out that the regions closer to the seafront or beaches are more likely to have a high booking rate than the regions in the middle. The reason is tourism is a major part of San Diego's economy and also the events like Comic Con, Surfing Tournaments, La Jolla Festival, happen at or near beaches. These events attract many people every year, so that's why properties near sea are most likely to have high booking rates.

#Plotting map for neighbourhoods

```r
df_sd_map <- read.csv("sd_map.csv")

df_sd_map$neighbourhood = ifelse(is.na(df_sd_map$neighbourhood),"Others", dfTrain$neighbourhood)

sd_map <- dfFull %>%
  select(high_booking_rate,neighbourhood, latitude, longitude) %>%
  filter(high_booking_rate == 1) %>% filter(neighbourhood != "Others") %>%
  group_by(neighbourhood) %>%
  summarise(total_high_bookings = sum(high_booking_rate),lat = mean(latitude),lng = mean(longitude) )%>%

  arrange(desc(total_high_bookings))
```

```r
m <- leaflet() %>%
  addTiles() %>%  # Add default OpenStreetMap map tiles
  addCircles(lat = df_sd_map$lat, lng = df_sd_map$lng, weight=1,radius=df_sd_map$total_high_bookings*5)
m  # Print the map
```

6. *How do the provided amenities affect the booking rate of an AirBnb property?*

To understand the importance of the amenities provided, a column with the count of amenities was introduced. After applying a Random Forest model to the variables including the amenities count, it was identified that the count of amenities was the most significant variable. To further add onto this point a density graph was plotted which depicted the amenities provided to rentals with both a high and a low booking rate. This plot was compared against a plot of amenities provided against the high booking rate. This proved that the high booking rate does not depend on the type of amenity provided in the Airbnb rental, but highly depends on the number of amenities provided. A word cloud plotted for both the amenities provided in a low booking rate rental and a high booking rate rental showed almost the same. Out of which free parking space, WiFi, smoke detectors and laptop friendly workspace and the most frequently available amenities.

#VarImp RF

```r
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
```

```r
dft1 <- dft
dft1$property_type <- as.factor(dft1$property_type)
```

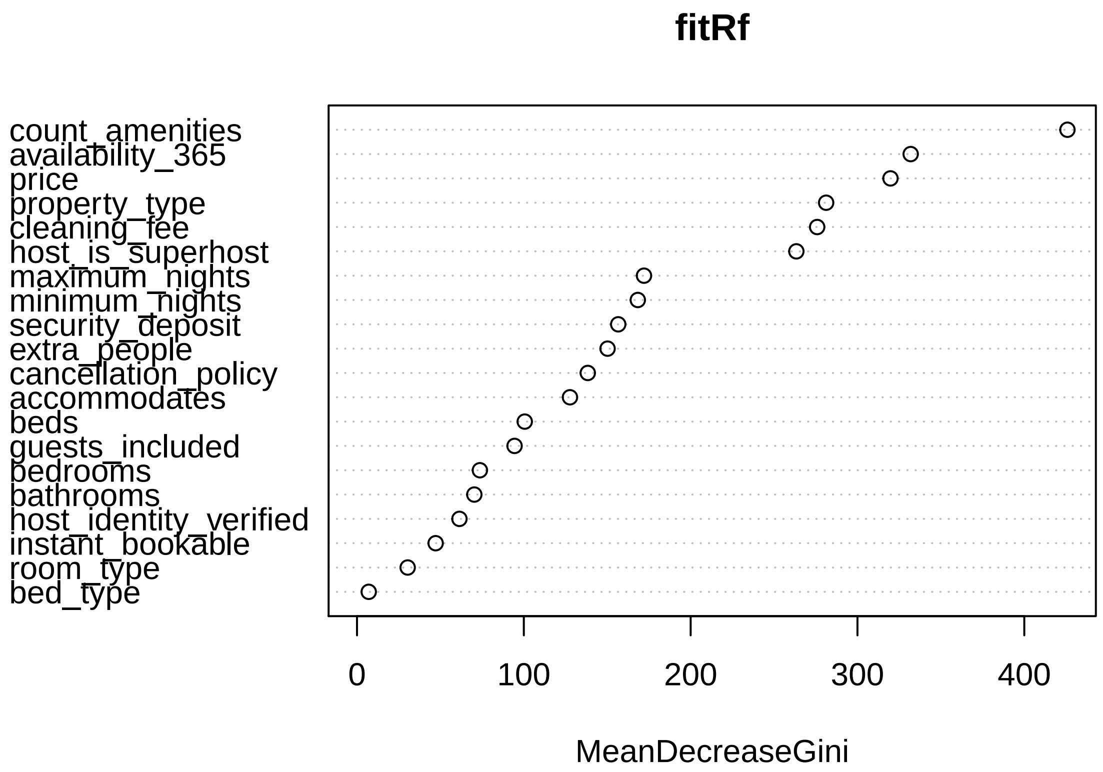
```r
set.seed(123)

fitRf <-
  randomForest(high_booking_rate ~ .-(id+ neighbourhood), family="binomial", data = dft1 )

print(fitRf)
```

```
##
## Call:
##  randomForest(formula = high_booking_rate ~ . - (id + neighbourhood),      data = dft1, family = "bi
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 18.09%
## Confusion matrix:
##      0    1 class.error
## 0 5281  519  0.08948276
## 1  954 1390  0.40699659
```

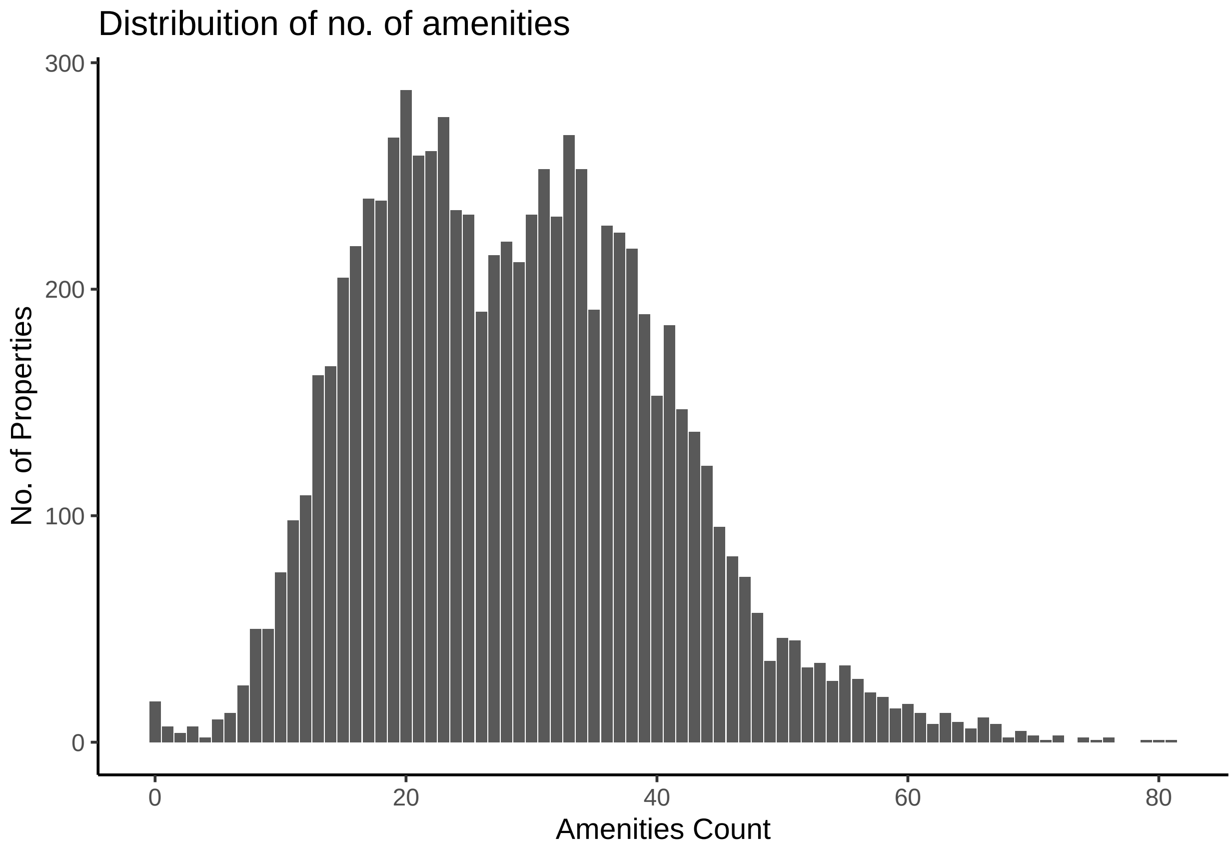
```
varImp(fitRf) %>%    # Add scale=FALSE inside VarImp if you don't want to scale
  rownames_to_column(var = "Variable") %>%
  #mutate(Importance = scales::percent(Overall/100)) %>%
  arrange(desc(Overall)) %>%
  as_tibble()
```

```
## # A tibble: 20 x 2
##    Variable               Overall
##    <chr>                    <dbl>
##  1 count_amenities         426.
##  2 availability_365        332.
##  3 price                   320.
##  4 property_type           281.
##  5 cleaning_fee            276.
##  6 host_is_superhost       263.
##  7 maximum_nights          172.
##  8 minimum_nights          168.
##  9 security_deposit        157.
## 10 extra_people            150.
## 11 cancellation_policy     138.
## 12 accommodates            128.
## 13 beds                    101.
## 14 guests_included          94.4
## 15 bedrooms                 73.7
## 16 bathrooms                70.3
## 17 host_identity_verified   61.4
## 18 instant_bookable         47.1
## 19 room_type                30.4
## 20 bed_type                  7.03
```

```
varImpPlot(fitRf, sort=TRUE)
```

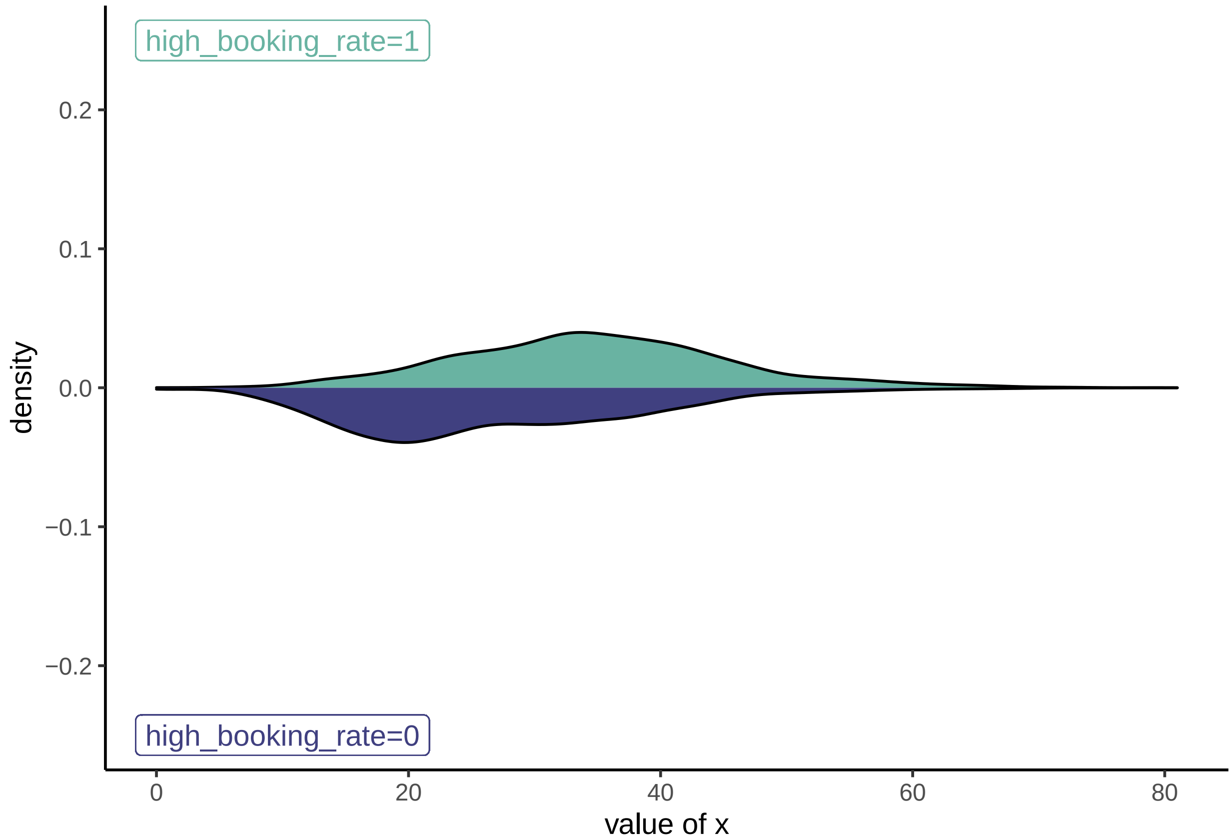



```
#library(ggplot2)
theme_set(theme_classic())

# Plot
g <- ggplot(dft, aes(x=count_amenities))+geom_bar() + labs(title="Distribuition of no. of amenities", x=
g
```



#amenities-density-comparision

```
p <- ggplot(dft, aes(x=x) ) +
  # Top
  geom_density( aes(x = count_amenities, y = ..density..), fill="#69b3a2",data = dft %>% filter(high_boo
  geom_label( aes(x=10, y=0.25, label="high_booking_rate=1"), color="#69b3a2") +
  # Bottom
  geom_density( aes(x = count_amenities, y = -..density..), fill= "#404080",data = dft %>% filter(high_
  geom_label( aes(x=10, y=-0.25, label="high_booking_rate=0"), color="#404080")  +
  xlab("value of x")
p
```




```
dft3 <- dfTrain %>% filter(high_booking_rate==1)

docs <- Corpus(VectorSource(dft3$amenities))# %>%  unnest_tokens(word, amenities,token = "regex",patter

toSpace <- content_transformer(function (x , pattern ) gsub(pattern, " ", x))  #unnest_tokens(word, ame
docs <- tm_map(docs, toSpace, "[^[:alnum:]]")
```

```
## Warning in tm_map.SimpleCorpus(docs, toSpace, "[^[:alnum:]]"): transformation
## drops documents
```

```
# Convert the text to lower case
docs <- tm_map(docs, content_transformer(tolower))
```

```
## Warning in tm_map.SimpleCorpus(docs, content_transformer(tolower)):
## transformation drops documents
```

```
# Remove english common stopwords
docs <- tm_map(docs, removeWords, stopwords("english"))
```

```
## Warning in tm_map.SimpleCorpus(docs, removeWords, stopwords("english")):
## transformation drops documents
```

```
docs <- tm_map(docs, removeWords, c("essentials","free","carbon","monoxide","smoke","street","premises"
```

```
## Warning in tm_map.SimpleCorpus(docs, removeWords, c("essentials", "free", :
## transformation drops documents
```

```
dtm <- TermDocumentMatrix(docs)
m <- as.matrix(dtm)
v <- sort(rowSums(m),decreasing=TRUE)
d <- data.frame(word = names(v),freq=v)


set.seed(123)
wordcloud(words = d$word, freq = d$freq,scale=c(4,.1), min.freq = 1000,
          random.order=FALSE,
          colors=brewer.pal(8, "Greens"))
```


```
dft4 <- dfTrain %>% filter(high_booking_rate==0)

docs <- Corpus(VectorSource(dft4$amenities))

toSpace <- content_transformer(function (x , pattern ) gsub(pattern, " ", x))
docs <- tm_map(docs, toSpace, "[^[:alnum:]]")
```

```
## Warning in tm_map.SimpleCorpus(docs, toSpace, "[^[:alnum:]]"): transformation
## drops documents
```

```r
# Convert the text to lower case
docs <- tm_map(docs, content_transformer(tolower))
```

```
## Warning in tm_map.SimpleCorpus(docs, content_transformer(tolower)):
## transformation drops documents
```

```r
# Remove english common stopwords
docs <- tm_map(docs, removeWords, stopwords("english"))
```

```
## Warning in tm_map.SimpleCorpus(docs, removeWords, stopwords("english")):
## transformation drops documents
```

```r
docs <- tm_map(docs, removeWords, c("essentials","free","carbon","monoxide","smoke","street","premises"
```

```
## Warning in tm_map.SimpleCorpus(docs, removeWords, c("essentials", "free", :
## transformation drops documents
```

```r
dtm <- TermDocumentMatrix(docs)
m <- as.matrix(dtm)
v <- sort(rowSums(m),decreasing=TRUE)
d <- data.frame(word = names(v),freq=v)

set.seed(123)
wordcloud(words = d$word, freq = d$freq,scale=c(4,.1), min.freq = 1000,
          random.order=FALSE,
          colors=brewer.pal(8, "Reds"))
```

III Methodology

1. Determined the correlation between all the important features. Chose price and high_booking_rate to numerically measure the correlation between them. To further discover the relation between these two features, we plotted the scatter plot and box plot of the divided data against price. Concluded that prices don't determine booking rates. Also, the properties which have higher booking rates have a median price of $120.

2. Selected only those variables that are most closely related to the accommodation at a particular property. Ran logistic regression model with high_booking_rate as DV. We determined the accuracy of the model using a confusion matrix. Concluded that for an Airbnb property to have higher booking rates, it must be flexible with respect to its booking duration and be able to provide accommodation to the guests.

3. Determined whether the factors which add to the price. Ran a regression model with high_booking_rate as a dependent variable. Determined the accuracy of the model using a confusion matrix Concluded that (i) If a property provides for extra_people despite charging for the same - its booking rate improves, (ii) Cleaning fee and security deposits do not have any impact on the higher booking rates (iii) If the price of a property is high, its booking rate decreases.

4. From our XG boost model, we concluded that the host_is_superhost is a really important variable. So, we analyzed this variable further. Selected the superhost and high booking rate columns, filtered the high booking rate = 1 then, grouped by host_is_superhost and got the count of booking rate =1 for both groups (superhost = true and supehost = false). The results of this query shows that Properties with a superhost has a higher chance of having a high booking rate.

5. To determine which neighbourhoods are most likely to get high_booking_rate, we plot a map. On the map we plotted the circles whose area is basically dependent on the number of high_booking_rate

properties in that area. We concluded that the regions closer to the seafront or beaches are more likely to have a high booking rate than the regions in the middle.

6. To understand the importance of the amenities provided, a column with the count of amenities was introduced. After applying a Random Forest model to the variables including the amenities count, we identified that the count of amenities was the most significant variable. To further add onto this point a density graph was plotted which depicted the amenities provided to rentals with both a high and a low booking rate. This plot was compared against a plot of amenities provided against the high booking rate. This proved that the high booking rate does not depend on the type of amenity provided in the Airbnb rental, but highly depends on the number of amenities provided. A word cloud plotted for both the amenities provided in a low booking rate rental and a high booking rate rental showed almost the same. Out of which free parking space, WiFi, smoke detectors and laptop friendly workspace and the most frequently available amenities.

IV Results and findings

- Neighbourhoods which would yield high returns will be Pacific beach, Mission Beach, Ocean Beach, La Jolla and North Park.
- An appropriate price range will be $100 - $120.
- The property must be flexible with respect to its booking duration and be able to provide accommodation to the guests.
- Achieving a superhost status will help in increasing the booking rates.
- A house or an apartment will have more bookings as compared to other property types.
- Amenities which matter the most are Free Parking, Smoke detectors, Wifi, and Laptop friendly workspace.
- Providing more amenities will affect the booking rate.

V Conclusion

The data analysis for this project was done keeping in mind the various factors that would largely affect the customer booking rates based on domain knowledge and the various models built for variable selection. The logistic models applied to variables of choice to explain different business cases helped identify the importance of each variable for that particular case. For instance the relation of price and high booking rate in customers showed that prices do not primarily or directly affect the decision making while booking a rental spot around San Diego. Where as the accommodative comfort provided to the customers in terms of the in house services provided largely sways their decision to book. The superhost program launched by Airbnb has proved to be very helpful to both the property owners and the customers alike. The superhost standing is achieved by an owner when he has more reservations for his properties, maintains a response rate and low cancellation rate and attains an overall high rating from his customers. Thus being a superhost propels more customers to book rentals from the owner owing to his status and the ability to provide extraordinary hospitality to his customers.

San Diego being the city on the Pacific coast of California is majorly known for its beaches, parks and warm climate. Thus, when an investor looks at prospective spaces for renting out in the area, locations with beach view, minimum commute distance to the beach, broad walks and local markets are a primary choice. From the research that we conducted as a team, it was reflected that this location provides abundant potential for prospective plot buyers in terms of the wide range of property types it has. Out of all the available property types, the house and apartment is found to be the most rented. This is due to the fact that San Diego has a small but well established IT hub, which has individuals from different backgrounds visiting the city on various occasions.

As for Airbnb rental property owners who are looking to improve their existing customer base, there were multiple factors that were established to be important. As this data analysis was done after reviewing the data from the customer- working on this will be of help to the property owners as the majority of customers reviewed in favor of particular factors. These factors included the amenities provided, the price range and allowing the customers to book their rentals well in advance. As these results were derived after working on the data provided by a large mass of customers of the website, this analysis definitely benefits the customers equally as it tends to their requests based renting habits of the majority of the customer population.

VI References

1. San Diego Official Tourism website. https://www.sandiego.org/explore/events.aspx
2. Leaflet for R. https://rstudio.github.io/leaflet/
3. Plots in R. https://www.statmethods.net/graphs/boxplot.html
4. Airbnb Superhost. https://www.airbnb.com/how-do-i-become-a-superhost
5. A Gentle Introduction to XGBoost for Applied Machine Learning. https://machinelearningmastery.com

VII Appendix

The analysis conducted over the San Diego data for the Airbnb rental markets was done by applying various data cleaning techniques to the data. Initially, missing values were filled in using appropriate techniques needed of each of the business cases tackled in that particular section. For better comprehension of the data, various visualizations were rendered. A map that plotted the locations with high booking rates on the San Diego map, a tree map showing various popular neighbourhoods and a map showing popular property types in the area all helped in understanding the data better.

The logistic regression models are appropriate to be used when the dependent variable is of dichotomous nature. Since, the majority of the business cases in this report consists of identifying the factors affecting the dependent variables, logistic regression method was employed. Another model used was XG Boost, which helped to identify the important variables that affect the business case. Here, the library is laser focused on computational speed and model performance. XGBoost method dominates structured or tabular datasets on classification and regression predictive modeling problems.

Furthermore, a Random Forest model was run, to again identify the important variable, but this time including a new column containing the count of amenities provided in the Airbnb rentals. This new variable, according to the Random Forest method proves to be most significant when a customer books a property. A density graph and the distribution of the amenities used help understand that the number of amenities provided by the owner trumps the types of amenities provided when renting an Airbnb. To add to this point a word cloud of the amenities provided in the homes with a high booking rate and the ones with a low booking rate was plotted. It showed similar amenities highlighted in both the word clouds. Thus, the number of amenities provided in an Airbnb rental is of utmost importance to the customers while booking.