Airbnb San Diego Market Analysis Report

BUDT 758T Data Mining and Predictive Analysis Team 1

May 7th, 2020

“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 the manuscript do not carry infringement/plagiarism issue upon any existing copyright materials.”

Team Member	Names of the Signed team members
Contact Member	Qiyuan Peng

Team Member	Names of the Signed team members
Team member 2	Di Hu
Team member 3	Xin Xu
Team member 4	Qinyu Wei
Team member 5	Munish Khurana

Introduction

San Diego was the second most popular California destination for Airbnb users last year, bringing hosts $213 million in income. Most of the customers of Airbnb are coming for vacation in San Diego. Roughly 482,400 people stayed in an Airbnb somewhere in the county between Memorial Day and Labor day, according to the 2019 Airbnb's data. The rentals generated about $112 million in supplemental income countywide and $75 million in the city of San Diego.

Executive Summary

Our perspective is focusing on the initial acquisition and features set up for investors. Investors want to know how to get an Airbnb with a high booking rate to maximize the return on Investment. Thus, our study is focusing on the location and features of a new house. We conducted a series of data analysis, research, and business analytics for the Airbnb Market at San Diego. After features site selection and feature analysis, the final location will be the Ocean Beach area because it has the sea view, beaches, and more restaurants and bars nearby. The optimal house type is a cottage with one bedroom and one bathroom with two beds.

We want to find a site that has a sea view, near the beach, restaurants, downtown, and bars. Customers are willing to book an Airbnb with Internet access, wifi, laptop, AC, no smoking, kitchen (with washing machine, microwave, coffee, tableware, oven, stove and grill), some essentials such as shampoo, hair dryer, hand sanitizer and bathtub. Since most people in the US have dogs, they want to book an Airbnb which is pet friendly. The room should be safe and private. Therefore, an intelligent lock that could be used in self-Check in is necessary. People don't want to book an Airbnb with a strict cancellation policy. They want the Host to have an email, phone, google, and facebook verification to make sure the host is certified.

Research Questions & Expected Findings

Preliminary questions related to our perspective:

Where is the best place to own an Airbnb house in San Diego?

1.Where is the best location?

2.Which location has relatively high demand? Are those locations near hot tourist sites(attractions)?

3.Which neighborhood is crowded with Airbnb hosts in certain location?

4.Which neighborhoods has high proportion of high booking rate?

We found that most of the customers of Airbnb are coming for vacation. Therefore, we expected that People are more likely to live near San Diego's most popular tourist attractions such as the Spanish Colonial-style architecture found in Balboa Park; the world famous San Diego Zoo; the Ocean Beach; the SeaWorld San Diego and the Midway Aircraft Carrier Museum.

What kind of homes?

1.Which property type is more likely to have a high booking rate?

2.Which room type is more likely to have a high booking rate?

3.How many beds&bathrooms should we include in our house to make it more popular?

From marketing researched we had done at the beginning of our project, we find that people who visit San Diego always expect to have a chill vacation with sunshine and beaches. Therefore, we assumed that most of visitors are couples or families, who may prefer private space in order to hava a peaceful vacation, so we expected to find that a small entire house having 1-3 bedrooms may be more popular.

What features may help to become more popular?

1.What kind of amenities can attract more customers to book?

2.What kind of keywords should a host include in the description to attract more customers?

San Diego is famous by its Latin American culture, sea coasts, seafoods and beaches. Therefore, we tought that the house which have easily access to these key features like sea view, restaurants, and beaches will attract more visitor to rent.

Methodology

Data Preparation

Splited the dataset into training set and testing set

We need to use some aggregation functions to process the null values so we need to first split the dataset to avoid the training data influencing the test data. The processing code for training set and test set are almost the same, so we only present the coding for training set and set include=False for test set chunks.

```
df<-
  read_csv("airbnbSanDiego.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 16 more columns
## )
```

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

```
skim(df)
```

Data summary

Name	df
Number of rows	8144
Number of columns	66
_______________________	
Column type frequency:	
character	30
Date	1
logical	9
numeric	26
________________________	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
access	3062	0.62	1	1000	0	4516	0
amenities	0	1.00	2	1390	0	7641	0
bed_type	0	1.00	5	13	0	5	0
cancellation_policy	0	1.00	6	27	0	7	0
city	0	1.00	2	28	0	31	0
cleaning_fee	871	0.89	5	9	0	325	0
description	134	0.98	2	1000	0	7714	0
extra_people	0	1.00	5	7	0	58	0
host_about	2560	0.69	1	6990	0	3039	7

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
host_acceptance_rate	3	1.00	3	3	0	1	0
host_location	25	1.00	2	98	0	330	0
host_neighbourhood	1165	0.86	5	31	0	186	0
host_response_rate	3	1.00	2	4	0	48	0
host_response_time	3	1.00	3	18	0	5	0
host_verifications	0	1.00	4	156	0	258	0
house_rules	2088	0.74	1	1000	0	4847	1
interaction	2672	0.67	1	1000	0	4459	0
market	12	1.00	7	22	0	5	0
monthly_price	7610	0.07	7	10	0	212	0
neighborhood_overview	2382	0.71	2	1000	0	4770	0
neighbourhood	300	0.96	6	28	0	111	0
notes	3672	0.55	1	1000	0	3835	0
price	0	1.00	5	10	0	653	0
property_type	0	1.00	3	22	0	36	0
room_type	0	1.00	10	15	0	4	0
security_deposit	1618	0.80	5	9	0	78	0
space	1757	0.78	1	1000	0	5966	0
state	2	1.00	2	15	0	5	0
transit	2950	0.64	1	1000	0	4471	0
weekly_price	7527	0.08	7	9	0	246	0

Variable type: Date

skim_variable	n_missing	complete_rate	min	max	median	n_unique
host_since	3	1	2008-07-08	2019-11-20	2015-12-01	2357

Variable type: logical

skim_variable	n_missing	complete_rate	mean	count
host_has_profile_pic	3	1	1.00	TRU: 8120, FAL: 21
host_identity_verified	3	1	0.46	FAL: 4416, TRU: 3725
host_is_superhost	3	1	0.38	FAL: 5078, TRU: 3063
instant_bookable	0	1	0.53	TRU: 4344, FAL: 3800
is_business_travel_ready	0	1	0.00	FAL: 8144
is_location_exact	0	1	0.79	TRU: 6414, FAL: 1730
require_guest_phone_verification	0	1	0.03	FAL: 7873, TRU: 271
require_guest_profile_picture	0	1	0.03	FAL: 7921, TRU: 223
requires_license	0	1	0.00	FAL: 8144

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
id	0	1.00	1101139.95	58767.28	1000005.00	1049779.50	1101546.00	1152876.00	1202078.00	
high_booking_rate	0	1.00	0.29	0.45	0.00	0.00	0.00	1.00	1.00	
accommodates	0	1.00	4.50	3.09	1.00	2.00	4.00	6.00	24.00	
availability_30	0	1.00	13.31	11.13	0.00	0.00	14.00	23.00	30.00	
availability_365	0	1.00	154.93	131.35	0.00	24.00	136.00	289.00	365.00	
availability_60	0	1.00	27.77	21.53	0.00	0.00	30.00	47.00	60.00	
availability_90	0	1.00	44.57	33.08	0.00	3.00	50.00	74.00	90.00	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
bathrooms	4	1.00	1.51	0.94	0.00	1.00	1.00	2.00	27.50	
bedrooms	10	1.00	1.65	1.21	0.00	1.00	1.00	2.00	14.00	
beds	9	1.00	2.38	1.95	0.00	1.00	2.00	3.00	22.00	
guests_included	0	1.00	2.33	2.37	1.00	1.00	1.00	2.00	24.00	
host_listings_count	3	1.00	42.82	170.82	0.00	1.00	2.00	9.00	1820.00	
latitude	0	1.00	32.77	0.07	32.53	32.72	32.76	32.80	33.09	
longitude	0	1.00	-117.18	0.06	-117.28	-117.24	-117.17	-117.14	-116.93	
maximum_nights	0	1.00	608.73	576.96	1.00	29.00	999.00	1125.00	16960.00	
minimum_nights	0	1.00	4.89	17.06	1.00	1.00	2.00	3.00	800.00	
review_scores_accuracy	1376	0.83	9.71	0.72	2.00	10.00	10.00	10.00	10.00	
review_scores_checkin	1383	0.83	9.83	0.58	2.00	10.00	10.00	10.00	10.00	
review_scores_cleanliness	1374	0.83	9.59	0.80	2.00	9.00	10.00	10.00	10.00	
review_scores_communication	1376	0.83	9.81	0.59	2.00	10.00	10.00	10.00	10.00	
review_scores_location	1383	0.83	9.79	0.57	2.00	10.00	10.00	10.00	10.00	
review_scores_rating	1373	0.83	95.33	7.00	20.00	94.00	97.00	100.00	100.00	
review_scores_value	1385	0.83	9.51	0.80	2.00	9.00	10.00	10.00	10.00	
square_feet	8035	0.01	778.93	522.75	0.00	500.00	650.00	1000.00	3600.00	
zipcode	81	0.99	92037.20	2064.54	22010.00	92101.00	92107.00	92110.00	92386.00	
{randomControl}	0	1.00	121494.40	288.15	121000.00	121242.75	121491.50	121741.00	121999.00	

```
set.seed(123)
dfTrain <- df%>% sample_frac(.75)
dfTest <- dplyr::setdiff(df, dfTrain)
```

Remove dollar sign and change price related character to numeric

```
dfTrain$cleaning_fee<-gsub("\\$","",dfTrain$cleaning_fee)#remove dollar sign
dfTrain$monthly_price<-gsub("\\$","",dfTrain$monthly_price)
dfTrain$price<-gsub("\\$","",dfTrain$price)
dfTrain$security_deposit<-gsub("\\$","",dfTrain$security_deposit)
dfTrain$weekly_price<-gsub("\\$","",dfTrain$weekly_price)
dfTrain$extra_people<-gsub("\\$","",dfTrain$extra_people)

dfTrain$cleaning_fee <- as.numeric(dfTrain$cleaning_fee)
dfTrain$monthly_price <- as.numeric(dfTrain$monthly_price)
dfTrain$price <- as.numeric(dfTrain$price)
dfTrain$security_deposit <- as.numeric(dfTrain$security_deposit)
dfTrain$weekly_price <- as.numeric(dfTrain$weekly_price)
dfTrain$extra_people <- as.numeric(dfTrain$extra_people)
```

Transfer response rate from percentage to decimel

```
dfTrain$host_response_rate <- as.numeric(sub("%", "",dfTrain$host_response_rate,fixed=TRUE))/100
```

Data cleaning for original variables

Filling null for original numeric variables

```
df%>%
  filter(is.na(review_scores_accuracy))%>%
  group_by(high_booking_rate)%>%
  tally()
```

high_booking_rate	n
<dbl>	<int>
0	1376

1 row

```
df%>%
  filter(is.na(review_scores_cleanliness))%>%
  group_by(high_booking_rate)%>%
  tally()
```

high_booking_rate <dbl>	n <int>
0	1374

1 row

```
df%>%
  filter(is.na(review_scores_checkin))%>%
  group_by(high_booking_rate)%>%
  tally()
```

high_booking_rate <dbl>	n <int>
0	1383

1 row

```
df%>%
  filter(is.na(review_scores_communication))%>%
  group_by(high_booking_rate)%>%
  tally()
```

high_booking_rate <dbl>	n <int>
0	1376

1 row

```
df%>%
  filter(is.na(review_scores_location))%>%
  group_by(high_booking_rate)%>%
  tally()
```

high_booking_rate <dbl>	n <int>
0	1383

1 row

```
df%>%
  filter(is.na(review_scores_value))%>%
  group_by(high_booking_rate)%>%
  tally()
```

high_booking_rate <dbl>	n <int>
0	1385

1 row

```
df%>%
  filter(is.na(review_scores_rating))%>%
  group_by(high_booking_rate)%>%
  tally()
```

high_booking_rate <dbl>	n <int>
0	1373

1 row

```
dfTrain%>%
  filter(price==0)
```

id <dbl>	high_booking_rate <dbl>
1132591	0
1037133	0
1200896	0

3 rows | 1-2 of 66 columns

In general, we filled null values using the mean or median except 3 special situations below:
1. After data exploration, we found that if any reviw_scores is null, the porprotion of high booking rate is 0% in our dataset. However, the mean of those review scores columns are too high (above 9), so we will use 0 to fill in the review_scores instead of using mean or median.
2. In addition, we found some 0 in price column, which may be some mistakes, so we take those 0 price as null values and fill them with mean.
3. Weekly price and Monthly price have a lot of null values, but we want to keep those non-null values. So, we looked into the Airbnb website. We found that many hosts did not list out they weekly price and monthly price directly, it may indicate that the weekly price is just simply times the price per nigh by days. Other hosts who listed they weekly and monthly price are using them to indicate the discont for long-term renting. Therefore, we will use price to calculate null weekly and monthly prices.

```
dfTrain$host_response_rate[is.na(dfTrain$host_response_rate)] <- median(dfTrain$host_response_rate, na.rm=TRUE)

dfTrain$host_listings_count[is.na(dfTrain$host_listings_count)] <- 0

dfTrain$review_scores_value[is.na(dfTrain$review_scores_value)] <- 0

dfTrain$review_scores_rating[is.na(dfTrain$review_scores_rating)] <-0

dfTrain$review_scores_location[is.na(dfTrain$review_scores_location)] <-0

dfTrain$review_scores_communication[is.na(dfTrain$review_scores_communication)] <- 0

dfTrain$review_scores_cleanliness[is.na(dfTrain$review_scores_cleanliness)] <- 0

dfTrain$review_scores_checkin[is.na(dfTrain$review_scores_checkin)] <- 0

dfTrain$review_scores_accuracy[is.na(dfTrain$review_scores_accuracy)] <- 0

dfTrain$beds[is.na(dfTrain$beds)] <- median(dfTrain$beds, na.rm=TRUE)

dfTrain$bedrooms[is.na(dfTrain$bedrooms)] <- median(dfTrain$bedrooms, na.rm=TRUE)

dfTrain$bathrooms[is.na(dfTrain$bathrooms)] <- median(dfTrain$bathrooms, na.rm=TRUE)

dfTrain$extra_people[is.na(dfTrain$extra_people)] <- mean(dfTrain$extra_people, na.rm=TRUE)

dfTrain$security_deposit[is.na(dfTrain$security_deposit)] <- mean(dfTrain$security_deposit, na.rm=TRUE)

dfTrain$cleaning_fee[is.na(dfTrain$cleaning_fee)] <- mean(dfTrain$cleaning_fee, na.rm=TRUE)

dfTrain$price[is.na(dfTrain$price)] <- mean(dfTrain$price, na.rm=TRUE)

dfTrain$price[dfTrain$price==0] <- mean(dfTrain$price, na.rm=TRUE)

dfTrain$weekly_price<-ifelse(is.na(dfTrain$weekly_price),dfTrain$price*7,dfTrain$weekly_price)

dfTrain$monthly_price<-ifelse(is.na(dfTrain$monthly_price),dfTrain$price*30,dfTrain$monthly_price)
```

Managing missing values for original character variables

1. Keep only year for host_since, and use “Unknown” to fill the null values
2. Replace null values with “Other” in host_response _time
3. Replace null values wuth “FALSE” for host_has_profile_pic, host_identity_verified, host_is_superhost
4. Change bedtype to real bed and non-real bed into dummy varaible

```
dfTrain$host_since <- year(dfTrain$host_since)
dfTrain$host_since[is.na(dfTrain$host_since)] <-"Unknown"
```

```
dfTrain$host_response_time[is.na(dfTrain$host_response_time)] <- "Other"
dfTrain$host_response_time[dfTrain$host_response_time == 'N/A'] <- 'Other'
```

```
dfTrain %>%
  group_by(host_has_profile_pic ) %>%
  tally()
```

host_has_profile_pic <lgl>	n <int>

host_has_profile_pic <lgl>	n <int>
FALSE	16
TRUE	6089
NA	3

3 rows

```
dfTrain %>%
  group_by(host_identity_verified ) %>%
  tally()
```

host_identity_verified <lgl>	n <int>
FALSE	3296
TRUE	2809
NA	3

3 rows

```
dfTrain %>%
  group_by(host_is_superhost ) %>%
  tally()
```

host_is_superhost <lgl>	n <int>
FALSE	3778
TRUE	2327
NA	3

3 rows

```
dfTrain$host_has_profile_pic[is.na(dfTrain$host_has_profile_pic)] <- FALSE
dfTrain$host_identity_verified[is.na(dfTrain$host_identity_verified)] <- FALSE
dfTrain$host_is_superhost[is.na(dfTrain$host_is_superhost)] <- FALSE
```

Use zipcode for defining locations.

Change zipcodes where have less than 5 airbnb homes and null values to “Other”.
This process can manage the null value and avoid new level(Did not appear in training set) appearing in the test set.

```
smalllocation <- dfTrain %>%
  group_by(zipcode) %>%
  tally() %>%
  filter(n<5)
smalllocation
```

zipcode <dbl>	n <int>
22050	1
91901	1
91902	2
91932	1
91941	2
91945	1
91950	1
92025	3
92071	1
92075	1

1-10 of 12 rows Previous **1** 2 Next

```
dfTrain$zipcode[dfTrain$zipcode %in% smalllocation$zipcode]<-"Other"
dfTrain$zipcode[is.na(dfTrain$zipcode)]<-"Other"

dfTrain%>%
  group_by(zipcode) %>%
  tally()
```

zipcode <chr>	n <int>
22010	6
91910	58
91911	42
91913	33
91914	10
91915	22
91942	8
92014	43
92037	398
92101	974

1-10 of 41 rows Previous **1** 2 3 4 5 Next

Creating derived variables

Get meaningful ratios from numeric variables

1. Bathroom per room is an important factor that people will see when they buy or rent a house.
2. Bed per room can tell how many beds for one room on average.
3. We use “room” here instead of bedroom becuase bedroom can be zero. When the bedroom equals to 0 we used 1 to be the denominator. 1 here then represnts the room that the guest will stay no matter it is a living room or other room.

```
dfTrain$availability_bathvsroom <- ifelse(dfTrain$bedrooms==0,dfTrain$bathrooms/1,dfTrain$bathrooms/dfTrain$bedro
oms)
dfTrain$availability_bedvsroom <- ifelse(dfTrain$bedrooms==0,dfTrain$beds/1,dfTrain$beds/dfTrain$bedrooms)
dfTrain$pricevsroom <- ifelse(dfTrain$bedrooms==0,dfTrain$price/1,dfTrain$price/dfTrain$bedrooms)

dfTrain$cleaning_feevsprice <- dfTrain$cleaning_fee / dfTrain$price
dfTrain$security_depositvsprice <- dfTrain$security_deposit / dfTrain$price
dfTrain$monthly_pricevsprice <- dfTrain$monthly_price / dfTrain$price
dfTrain$weekly_pricevsprice <- dfTrain$weekly_price / dfTrain$price
dfTrain$extra_peoplevsprice <- dfTrain$extra_people / dfTrain$price
```

Get length for text data

Length of text may indicate that information provided for customers.
For example: Longer host_about may provide more information about the host for potential customer, a good self introduction may attract more customers

```
dfTrain$access_length <- nchar(dfTrain$access)
dfTrain$description_length <- nchar(dfTrain$description)
dfTrain$host_about_length <- nchar(dfTrain$host_about)
dfTrain$interaction_length <- nchar(dfTrain$interaction)
dfTrain$notes_length <- nchar(dfTrain$notes)
dfTrain$transit_length <- nchar(dfTrain$transit)
dfTrain$house_rules_length <- nchar(dfTrain$house_rules)
dfTrain$space_length <- nchar(dfTrain$space)

dfTrain$access_length[is.na(dfTrain$access_length)] <- 0

dfTrain$description_length[is.na(dfTrain$description_length)] <- 0

dfTrain$host_about_length[is.na(dfTrain$host_about_length)] <-0

dfTrain$interaction_length[is.na(dfTrain$interaction_length)] <- 0

dfTrain$notes_length[is.na(dfTrain$notes_length)] <- 0

dfTrain$space_length[is.na(dfTrain$space_length)] <- 0

dfTrain$transit_length[is.na(dfTrain$transit_length)] <- 0

dfTrain$house_rules_length[is.na(dfTrain$house_rules_length)] <- 0
```

Keywards Extraction and Dummy Variables Creation

Extract keywords from description column, description give customers more information other than amenities. In description, customers can know about some interest facts around the destination.

```
dfTrain$description_sea = 0
dfTrain$description_sea[which(grepl(pattern = "sea",ignore.case =TRUE, x = dfTrain$description) == TRUE)] = 1

dfTrain$description_view = 0
dfTrain$description_view[which(grepl(pattern = "water view",ignore.case =TRUE, x = dfTrain$description) == TRUE)]
= 1

dfTrain$description_lake = 0
dfTrain$description_lake[which(grepl(pattern = "lake",ignore.case =TRUE, x = dfTrain$description) == TRUE)] = 1

dfTrain$description_mountain = 0
dfTrain$description_mountain[which(grepl(pattern = "mountain",ignore.case =TRUE, x = dfTrain$description) == TRUE
)] = 1

dfTrain$description_museum = 0
dfTrain$description_museum[which(grepl(pattern = "museum",ignore.case =TRUE, x = dfTrain$description) == TRUE)] =
1

dfTrain$description_beach = 0
dfTrain$description_beach[which(grepl(pattern = "beach", ignore.case =TRUE, x = dfTrain$description) == TRUE)] =
1

dfTrain$description_restaurant = 0
dfTrain$description_restaurant[which(grepl(pattern = "restaurant", ignore.case =TRUE, x = dfTrain$description) ==
TRUE)] = 1

dfTrain$description_shopping = 0
dfTrain$description_shopping[which(grepl(pattern = "shopping",ignore.case =TRUE, x = dfTrain$description) == TRUE
)] = 1

dfTrain$description_downtown = 0
dfTrain$description_downtown[which(grepl(pattern = "downtown",ignore.case =TRUE, x = dfTrain$description) == TRUE
)] = 1

dfTrain$description_university = 0
dfTrain$description_university[which(grepl(pattern = "university",ignore.case =TRUE, x = dfTrain$description) ==
TRUE)] = 1

dfTrain$description_station = 0
dfTrain$description_station[which(grepl(pattern = "station",ignore.case =TRUE, x = dfTrain$description) == TRUE)]
= 1

dfTrain$description_balcony = 0
dfTrain$description_balcony[which(grepl(pattern = "balcony",ignore.case =TRUE, x = dfTrain$description) == TRUE)]
= 1

dfTrain$description_bars = 0
dfTrain$description_bars[which(grepl(pattern = "bars", ignore.case =TRUE, x = dfTrain$description) == TRUE)] = 1
```

Break down amenities into sub amenities dummies

```
dfTrain$amenities_wifi = 0
dfTrain$amenities_wifi[which(grepl(pattern = "Wifi", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_air = 0
dfTrain$amenities_air[which(grepl(pattern = "Air", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_pool = 0
dfTrain$amenities_pool[which(grepl(pattern = "Pool", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_kitchen = 0
dfTrain$amenities_kitchen[which(grepl(pattern = "Kitchen", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_free = 0
dfTrain$amenities_free[which(grepl(pattern = "Free", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_heating = 0
dfTrain$amenities_heating[which(grepl(pattern = "Heating", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_washer = 0
dfTrain$amenities_washer[which(grepl(pattern = "Washer", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_dryer = 0
dfTrain$amenities_dryer[which(grepl(pattern = "Dryer", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_smoke = 0
dfTrain$amenities_smoke[which(grepl(pattern = "Smoke", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_carbon = 0
dfTrain$amenities_carbon[which(grepl(pattern = "Carbon", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_aid = 0
dfTrain$amenities_aid[which(grepl(pattern = "aid", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_essentials = 0
dfTrain$amenities_essentials[which(grepl(pattern = "Essentials", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_shampoo = 0
dfTrain$amenities_shampoo[which(grepl(pattern = "Shampoo", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_lock = 0
dfTrain$amenities_lock[which(grepl(pattern = "Lock", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_hanger = 0
dfTrain$amenities_hanger[which(grepl(pattern = "Hanger", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_hair = 0
dfTrain$amenities_hair[which(grepl(pattern = "Hair", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_iron = 0
dfTrain$amenities_iron[which(grepl(pattern = "Iron", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_laptop = 0
dfTrain$amenities_laptop[which(grepl(pattern = "Laptop", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_private = 0
dfTrain$amenities_private[which(grepl(pattern = "Private", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_host = 0
dfTrain$amenities_host[which(grepl(pattern = "Host", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_tv = 0
dfTrain$amenities_tv[which(grepl(pattern = "TV", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_internet = 0
dfTrain$amenities_internet[which(grepl(pattern = "Internet", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_kid = 0
dfTrain$amenities_kid[which(grepl(pattern = "kid", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_events = 0
dfTrain$amenities_events[which(grepl(pattern = "events", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_safe = 0
dfTrain$amenities_safe[which(grepl(pattern = "Safe", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_fire = 0
dfTrain$amenities_fire[which(grepl(pattern = "Fire", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_pet = 0
dfTrain$amenities_pet[which(grepl(pattern = "Pets", x = dfTrain$amenities) == TRUE)] = 1
```

```
dfTrain$amenities_dog = 0
dfTrain$amenities_dog[which(grepl(pattern = "Dog", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_water = 0
dfTrain$amenities_water[which(grepl(pattern = "water", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_microwave = 0
dfTrain$amenities_microwave[which(grepl(pattern = "Microwave", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_coffee = 0
dfTrain$amenities_coffee[which(grepl(pattern = "Coffee", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_dishes = 0
dfTrain$amenities_dishes[which(grepl(pattern = "Dishes", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_luggage = 0
dfTrain$amenities_luggage[which(grepl(pattern = "Luggage", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_gym = 0
dfTrain$amenities_gym[which(grepl(pattern = "Gym", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_elevator = 0
dfTrain$amenities_elevator[which(grepl(pattern = "Elevator", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_essentials = 0
dfTrain$amenities_essentials[which(grepl(pattern = "Essentials", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_selfcheckin = 0
dfTrain$amenities_selfcheckin[which(grepl(pattern = "Self check-in", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_hottub = 0
dfTrain$amenities_hottub[which(grepl(pattern ="tub", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_breakfast = 0
dfTrain$amenities_breakfast[which(grepl(pattern ="Breakfast", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_24hour = 0
dfTrain$amenities_24hour[which(grepl(pattern ="24", x = dfTrain$amenities) == TRUE)] = 1


dfTrain$amenities_oven = 0
dfTrain$amenities_oven[which(grepl(pattern ="Oven", x = dfTrain$amenities) == TRUE)] = 1

dfTrain$amenities_stove = 0
dfTrain$amenities_stove[which(grepl(pattern ="Stove", x = dfTrain$amenities) == TRUE)] = 1


dfTrain$amenities_bbq = 0
dfTrain$amenities_bbq[which(grepl(pattern ="BBQ", x = dfTrain$amenities) == TRUE)] = 1
```

Break down verification methods into dummies. Verification is essential for proving the reliability for the host

```
dfTrain$host_verifications_email = 0
dfTrain$host_verifications_email[which(grepl(pattern = "email", x = dfTrain$host_verifications) == TRUE)] = 1

dfTrain$host_verifications_phone = 0
dfTrain$host_verifications_phone[which(grepl(pattern = "phone", x = dfTrain$host_verifications) == TRUE)] = 1

dfTrain$host_verifications_jumio = 0
dfTrain$host_verifications_jumio[which(grepl(pattern = "jumio", x = dfTrain$host_verifications) == TRUE)] = 1

dfTrain$host_verifications_gover = 0
dfTrain$host_verifications_gover[which(grepl(pattern = "gover", x = dfTrain$host_verifications) == TRUE)] = 1

dfTrain$host_verifications_self = 0
dfTrain$host_verifications_self[which(grepl(pattern = "self", x = dfTrain$host_verifications) == TRUE)] = 1

dfTrain$host_verifications_identity = 0
dfTrain$host_verifications_identity[which(grepl(pattern = "identity", x = dfTrain$host_verifications) == TRUE)] = 
1

dfTrain$host_verifications_facebook = 0
dfTrain$host_verifications_facebook[which(grepl(pattern = "facebook", x = dfTrain$host_verifications) == TRUE)] = 
1

dfTrain$host_verifications_kba = 0
dfTrain$host_verifications_kba[which(grepl(pattern = "kba", x = dfTrain$host_verifications) == TRUE)] = 1

dfTrain$host_verifications_review = 0
dfTrain$host_verifications_review[which(grepl(pattern = "review", x = dfTrain$host_verifications) == TRUE)] = 1

dfTrain$host_verifications_google = 0
dfTrain$host_verifications_google[which(grepl(pattern = "google", x = dfTrain$host_verifications) == TRUE)] = 1

dfTrain$host_verifications_online = 0
dfTrain$host_verifications_online[which(grepl(pattern = "online", x = dfTrain$host_verifications) == TRUE)] = 1

dfTrain$host_verifications_offline = 0
dfTrain$host_verifications_offline[which(grepl(pattern = "offline", x = dfTrain$host_verifications) == TRUE)] = 1
```

Divided cancellation_policy to three main types

```
dfTrain %>%
  group_by(cancellation_policy) %>%
  tally()
```

cancellation_policy <chr>	**n** <int>
flexible	1469
luxury_moderate	1
moderate	1630
strict_14_with_grace_period	2712
super_strict_30	54
super_strict_60	242

6 rows

```
dfTrain$cancellation_flexible = 0
dfTrain$cancellation_flexible[which(grepl(pattern = "flexible", x = dfTrain$cancellation_policy) == TRUE)] = 1

dfTrain$cancellation_moderate = 0
dfTrain$cancellation_moderate[which(grepl(pattern = "moderate", x = dfTrain$cancellation_policy) == TRUE)] = 1

dfTrain$cancellation_strict = 0
dfTrain$cancellation_strict[which(grepl(pattern = "strict", x = dfTrain$cancellation_policy) == TRUE)] = 1
```

Sentiment Analysis for neighborhood_overview.
Creat new dummy variable named neighborhood_overview_positive to record if the neighborhood overview is positive or negative.

```
dfTidy <-
  dfTrain %>%
  unnest_tokens(word, neighborhood_overview)

dfTidy <-
  dfTidy %>%
  anti_join(stop_words)
```

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

```
dfTidy <-
  dfTidy %>%
  select(id, word)
dfTidy
```

id	word
<dbl>	<chr>
1072575	sherman
1072575	heights
1072575	named
1072575	matthew
1072575	sherman
1072575	bought
1072575	160
1072575	acres
1072575	phone
1072575	hidden

1-10 of 10,000 rows Previous **1** 2 3 4 5 6 ... 1000 Next

```
sentimentBING <-
  dfTidy %>%
  inner_join(get_sentiments("bing")) %>%
  count(id, sentiment) %>%
  spread(sentiment, n, fill=0) %>%
  mutate(score = log(positive+0.5) - log(negative+0.5)) %>%
  setNames(c(names(.)[1],paste0('BING', names(.)[-1])))
```

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

```
dfTrain<-left_join(dfTrain,sentimentBING,by="id")

dfTrain$neighborhood_overview_positive=0
dfTrain$neighborhood_overview_positive[dfTrain$BINGscore>0]=1
dfTrain$neighborhood_overview_positive[is.na(dfTrain$BINGscore)]=0
dfTrain$neighborhood_overview_positive[is.nan(dfTrain$BINGscore)]=0
```

Predictive Model

Remove variables will not included in the model

```
trainx<- subset(dfTrain,select=-c(id,host_location,host_neighbourhood,neighbourhood,latitude,longitude,`{randomCo
ntrol}`,host_acceptance_rate,square_feet,neighborhood_overview,BINGnegative,BINGpositive,BINGscore,state,market,c
ity,access,description,host_about,interaction,notes,transit,house_rules,space,host_verifications,amenities,cancel
lation_policy))
```

Variables included in the model

Original Variables:

“accommodates”, “availability_30”, “availability_365”,“availability_60”, “availability_90”, “bathrooms”, “bed_type”, “bedrooms”, “beds” , “cleaning_fee”, “extra_people”, “guests_included”, “host_has_profile_pic”, “host_identity_verified”, “host_is_superhost”, “host_listings_count”, “host_response_rate”, “host_response_time”, “host_since”, “instant_bookable”, “is_business_travel_ready”, “is_location_exact”, “maximum_nights”, “minimum_nights”, “monthly_price”, “price”, “property_type”, “require_guest_phone_verification”, “require_guest_profile_picture”, “requires_license”, “review_scores_accuracy”, “review_scores_checkin”, “review_scores_cleanliness”, “review_scores_communication”, “review_scores_location”, “review_scores_rating”, “review_scores_value”, “room_type”, “security_deposit”, “weekly_price”, “zipcode”

Ratios:

“availability_bathvsroom”, “availability_bedvsroom”, “pricevsroom”, “cleaning_feevsprice”, “security_depositvsprice”, “monthly_pricevsprice”, “weekly_pricevsprice”, “extra_peoplevsprice”

TextLength:

“access_length”, “host_about_length”, “interaction_length”, “notes_length”, “transit_length”, “house_rules_length”, “space_length”

description:

“description_length”, “description_sea”, “description_view”, “description_lake”, “description_mountain”, “description_museum”, “description_beach”, “description_restaurant”, “description_shopping”, “description_downtown”, “description_university”, “description_station”, “description_balcony”, “description_bars”

amentities:

“amenities_wifi”,“amenities_air,”amenities_pool“,”amenities_kitchen“,”amenities_free“,”amenities_heating“,”amenities_washer“,”amenities_dryer“,”amenities_smo

host_verifications:

“host_verifications_email”, “host_verifications_phone”, “host_verifications_jumio”, “host_verifications_gover”, “host_verifications_self”, “host_verifications_identity”, “host_verifications_facebook”, “host_verifications_kba”, “host_verifications_review”, “host_verifications_google”, “host_verifications_online”, “host_verifications_offline”

cancellation_policy:

“cancellation_flexible”, “cancellation_moderate”, “cancellation_strict”

neighborhood_overview:

“neighborhood_overview_positive”

Change high_booking_rate colums into factor and replace 1/0 to X1_class/X0_class

```
levels(trainx$high_booking_rate) <- c("0_class", "1_class")
trainx<-trainx  %>%
  mutate(high_booking_rate = factor(high_booking_rate,
          labels = make.names(levels(high_booking_rate))))
trainx
```

high_booking_rate <fctr>	accommodates <dbl>	availability_30 <dbl>	availability_365 <dbl>	availability_60 <dbl>	availability_90 <dbl>
X1_class	2	20	75	45	75
X1_class	8	5	235	18	23
X0_class	7	21	269	41	68
X0_class	2	27	362	57	87
X0_class	4	8	17	10	17
X0_class	6	0	108	14	24
X1_class	4	26	97	56	86
X0_class	4	1	266	2	2
X1_class	6	14	293	41	71
X0_class	6	1	266	1	2

1-10 of 6,108 rows | 1-6 of 129 columns Previous **1** 2 3 4 5 6 ... 611 Next

XGBoost model based on the Kaggle part

```
trctrl <- trainControl(method = "cv", number = 10,
  classProbs = TRUE,
  verboseIter = TRUE,
  summaryFunction = prSummary,
  savePredictions = TRUE,
  allowParallel = TRUE)

tune_grid <- expand.grid(nrounds = 300,
                        max_depth = 10,
                        eta = 0.05,
                        gamma = 0.01,
                        colsample_bytree = 0.75,
                        min_child_weight = 0,
                        subsample = 0.6)

xgb_fit <- train(as.factor(high_booking_rate) ~., data = trainx, method = "xgbTree",
                trControl=trctrl,
                metric = "AUC",
                tuneGrid = tune_grid,
                tuneLength = 10)
```

```
## + Fold01: nrounds=300, max_depth=10, eta=0.05, gamma=0.01, colsample_bytree=0.75, min_child_weight=0, subsampl
e=0.6
## - Fold01: nrounds=300, max_depth=10, eta=0.05, gamma=0.01, colsample_bytree=0.75, min_child_weight=0, subsampl
e=0.6
## + Fold02: nrounds=300, max_depth=10, eta=0.05, gamma=0.01, colsample_bytree=0.75, min_child_weight=0, subsampl
e=0.6
## - Fold02: nrounds=300, max_depth=10, eta=0.05, gamma=0.01, colsample_bytree=0.75, min_child_weight=0, subsampl
e=0.6
## + Fold03: nrounds=300, max_depth=10, eta=0.05, gamma=0.01, colsample_bytree=0.75, min_child_weight=0, subsampl
e=0.6
## - Fold03: nrounds=300, max_depth=10, eta=0.05, gamma=0.01, colsample_bytree=0.75, min_child_weight=0, subsampl
e=0.6
## + Fold04: nrounds=300, max_depth=10, eta=0.05, gamma=0.01, colsample_bytree=0.75, min_child_weight=0, subsampl
e=0.6
## - Fold04: nrounds=300, max_depth=10, eta=0.05, gamma=0.01, colsample_bytree=0.75, min_child_weight=0, subsampl
e=0.6
## + Fold05: nrounds=300, max_depth=10, eta=0.05, gamma=0.01, colsample_bytree=0.75, min_child_weight=0, subsampl
e=0.6
## - Fold05: nrounds=300, max_depth=10, eta=0.05, gamma=0.01, colsample_bytree=0.75, min_child_weight=0, subsampl
e=0.6
## + Fold06: nrounds=300, max_depth=10, eta=0.05, gamma=0.01, colsample_bytree=0.75, min_child_weight=0, subsampl
e=0.6
## - Fold06: nrounds=300, max_depth=10, eta=0.05, gamma=0.01, colsample_bytree=0.75, min_child_weight=0, subsampl
e=0.6
## + Fold07: nrounds=300, max_depth=10, eta=0.05, gamma=0.01, colsample_bytree=0.75, min_child_weight=0, subsampl
e=0.6
## - Fold07: nrounds=300, max_depth=10, eta=0.05, gamma=0.01, colsample_bytree=0.75, min_child_weight=0, subsampl
e=0.6
## + Fold08: nrounds=300, max_depth=10, eta=0.05, gamma=0.01, colsample_bytree=0.75, min_child_weight=0, subsampl
e=0.6
## - Fold08: nrounds=300, max_depth=10, eta=0.05, gamma=0.01, colsample_bytree=0.75, min_child_weight=0, subsampl
e=0.6
## + Fold09: nrounds=300, max_depth=10, eta=0.05, gamma=0.01, colsample_bytree=0.75, min_child_weight=0, subsampl
e=0.6
## - Fold09: nrounds=300, max_depth=10, eta=0.05, gamma=0.01, colsample_bytree=0.75, min_child_weight=0, subsampl
e=0.6
## + Fold10: nrounds=300, max_depth=10, eta=0.05, gamma=0.01, colsample_bytree=0.75, min_child_weight=0, subsampl
e=0.6
## - Fold10: nrounds=300, max_depth=10, eta=0.05, gamma=0.01, colsample_bytree=0.75, min_child_weight=0, subsampl
e=0.6
## Aggregating results
## Fitting final model on full training set
```

```
# have a look at the model
xgb_fit
```

```
## eXtreme Gradient Boosting
##
## 6108 samples
##  128 predictor
##    2 classes: 'X0_class', 'X1_class'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 5497, 5497, 5497, 5497, 5497, 5497, ...
## Resampling results:
##
##   AUC        Precision  Recall     F
##   0.9692554  0.8878859  0.9190866  0.9031625
##
## Tuning parameter 'nrounds' was held constant at a value of 300
## Tuning
##
## Tuning parameter 'min_child_weight' was held constant at a value of 0
##
## Tuning parameter 'subsample' was held constant at a value of 0.6
```

Prediction Results

```
results <-
  xgb_fit%>%
  predict(dfTest, type = 'prob') %>%
  bind_cols(dfTest, predictedProb=.)
results
```

id <dbl>	high_booking_rate <dbl>

id <dbl>	high_booking_rate <dbl>
1188092	1
1169740	0
1193914	0
1090654	0
1047597	0
1131610	0
1020328	0
1128214	0
1026233	0
1016227	0

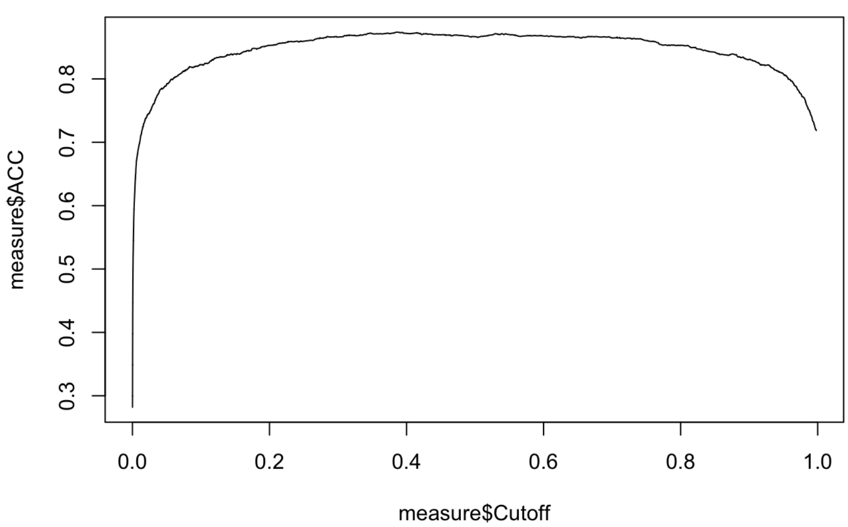
1-10 of 2,036 rows | 1-2 of 158 columns Previous **1** 2 3 4 5 6 ... 204 Next

Draw Accuracy, Sensitivity, Specificity VS Cutoff

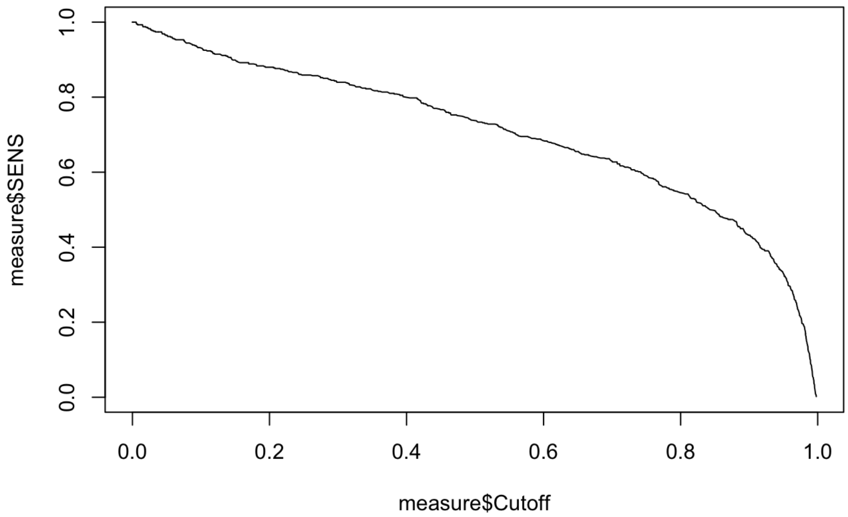
```
class <- results$high_booking_rate
score <- results$X1_class
measure <- measureit(score = score, class = class,
                     measure = c("ACC", "SENS", "SPEC"))
names(measure)
```

```
## [1] "Cutoff" "Depth"  "TP"     "FP"     "TN"     "FN"     "ACC"    "SENS"
## [9] "SPEC"
```

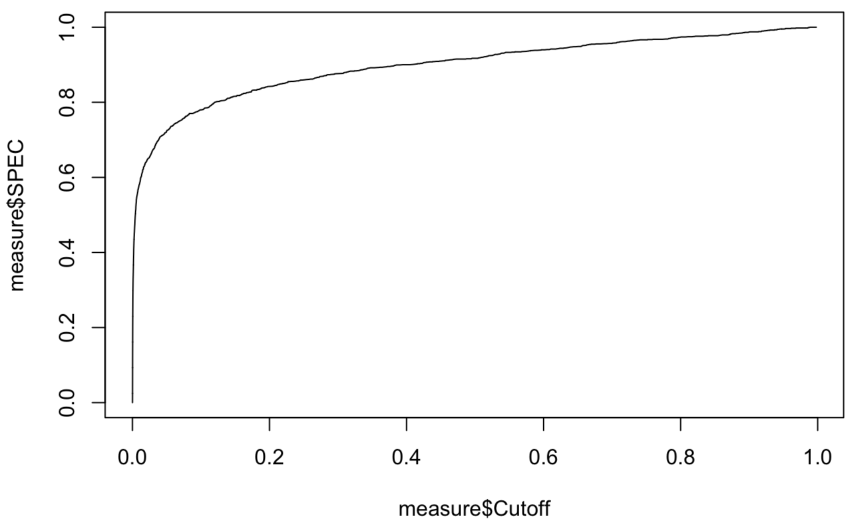
```
#> [1] "Cutoff" "Depth"  "TP"     "FP"     "TN"     "FN"     "ACC"    "SENS"
#> [9] "FSCR"
plot1<-plot(measure$ACC~measure$Cutoff, type = "l")
```




```
plot2<-plot(measure$SENS~measure$Cutoff, type = "l")
```




```
plot3<-plot(measure$SPEC~measure$Cutoff, type = "l")
```



High sensitivity

```
results1<-results%>%
  mutate(predictedClass = ifelse(X1_class > 0.15, 1, 0))

results1<-
  results1 %>%
  mutate(high_booking_rate = as.factor(high_booking_rate), predictedClass = as.factor(predictedClass))

results1 %>%
  xtabs(~predictedClass+high_booking_rate, .) %>%
  confusionMatrix(positive = '1')
```

```
## Confusion Matrix and Statistics
##
##                high_booking_rate
## predictedClass    0    1
##              0 1193   59
##              1  269  515
##
##                Accuracy : 0.8389
##                  95% CI : (0.8222, 0.8546)
##     No Information Rate : 0.7181
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6419
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8972
##             Specificity : 0.8160
##          Pos Pred Value : 0.6569
##          Neg Pred Value : 0.9529
##              Prevalence : 0.2819
##          Detection Rate : 0.2529
##    Detection Prevalence : 0.3851
##       Balanced Accuracy : 0.8566
##
##        'Positive' Class : 1
##
```

High specificity

```
results2<-results%>%
  mutate(predictedClass = ifelse(X1_class > 0.65, 1, 0))

results2<-
  results2 %>%
  mutate(high_booking_rate = as.factor(high_booking_rate), predictedClass = as.factor(predictedClass))

results2 %>%
  xtabs(~predictedClass+high_booking_rate, .) %>%
  confusionMatrix(positive = '1')
```

```
## Confusion Matrix and Statistics
##
##                high_booking_rate
## predictedClass    0    1
##              0 1387  198
##              1   75  376
##
##                Accuracy : 0.8659
##                  95% CI : (0.8503, 0.8804)
##     No Information Rate : 0.7181
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6458
##
##  Mcnemar's Test P-Value : 1.539e-13
##
##             Sensitivity : 0.6551
##             Specificity : 0.9487
##          Pos Pred Value : 0.8337
##          Neg Pred Value : 0.8751
##              Prevalence : 0.2819
##          Detection Rate : 0.1847
##    Detection Prevalence : 0.2215
##       Balanced Accuracy : 0.8019
##
##        'Positive' Class : 1
##
```

Exploratory data analysis

Location Selection: By using zipcode as our index, we grouped all Airbnb houses within certain locations and calculated the proportion of houses that have high booking rate in each location to help us determine whether a location is popular among Airbnb users, and then we sorted Top 10 locations.

```
df$neighbourhood[is.na(df$neighbourhood)] <- "Other"
df %>%
  group_by(neighbourhood ) %>%
  tally()
```

neighbourhood <chr>	n <int>
Adams North	32
Allied Gardens	27
Alta Vista	2
Azalea/Hollywood Park	15
Balboa Park	22
Barrio Logan	19
Bay Ho	56
Bay Park	140
Bay Terraces	26
Birdland	19

1-10 of 112 rows Previous **1** 2 3 4 5 6 ... 12 Next

Calculate the Number of airbnb homes that getting high booking rate in each location.

```
df1<-
  df %>%
  group_by(zipcode) %>%
  mutate(sum_high = sum(high_booking_rate))
head(df1)
```

id <dbl>	high_booking_rate <dbl>
1147275	1
1063167	0
1188092	1
1169740	0
1128880	0
1042092	1

6 rows | 1-2 of 149 columns

```
dfNumber<-
  df1 %>%
  group_by(zipcode) %>%
  tally() %>%
  filter(n>10)
dfNumber
```

zipcode <dbl>	n <int>
91910	78
91911	51
91913	43
91914	14
91915	28
92014	53
92037	542
92101	1293
92102	376
92103	506

1-10 of 38 rows Previous **1** 2 3 4 Next

Merge the two dataframes

```
total <-
  merge(dfNumber,df1,by="zipcode")
total
```

zipcode	n	id	high_booking_rate
<dbl>	<int>	<dbl>	<dbl>
91910	78	1161039	0
91910	78	1118111	0
91910	78	1074383	0
91910	78	1187007	1
91910	78	1021014	0
91910	78	1009772	0
91910	78	1046738	0
91910	78	1145061	1
91910	78	1198603	1
91910	78	1009845	0

1-10 of 8,102 rows | 1-4 of 150 columns Previous **1** 2 3 4 5 6 ... 811 Next

Calculate the Proportion of the Airbnb homes that has high booking rate in each location.

```
total<-
  total %>%
  group_by(zipcode) %>%
  mutate(rate = sum_high/n)
total
```

zipcode	n	id	high_booking_rate
<dbl>	<int>	<dbl>	<dbl>
91910	78	1161039	0
91910	78	1118111	0
91910	78	1074383	0
91910	78	1187007	1
91910	78	1021014	0
91910	78	1009772	0
91910	78	1046738	0
91910	78	1145061	1
91910	78	1198603	1
91910	78	1009845	0

1-10 of 8,102 rows | 1-4 of 151 columns Previous **1** 2 3 4 5 6 ... 811 Next

```
totalNoDuplicate <- total[!duplicated(total$zipcode),]
```

Select zipcode, sum of high_booking rate, n(number of airbnb homes), proportion rate column and filter the top 10 locations according to the proportion of having high booking rate.

```
total_rank<-
  totalNoDuplicate %>%
  select(zipcode, sum_high, n, rate) %>%
  arrange(desc(rate))
total_rank
```

zipcode	sum_high	n	rate
<dbl>	<dbl>	<int>	<dbl>
92104	203	466	0.43562232
92102	160	376	0.42553191
92116	98	232	0.42241379
92107	189	459	0.41176471

zipcode <dbl>	sum_high <dbl>	n <int>	rate <dbl>
92121	6	15	0.40000000
92113	39	100	0.39000000
92105	37	105	0.35238095
92114	30	86	0.34883721
92110	83	247	0.33603239
92106	47	140	0.33571429

1-10 of 38 rows Previous **1** 2 3 4 Next

```
head(total_rank,10)
```

zipcode <dbl>	sum_high <dbl>	n <int>	rate <dbl>
92104	203	466	0.4356223
92102	160	376	0.4255319
92116	98	232	0.4224138
92107	189	459	0.4117647
92121	6	15	0.4000000
92113	39	100	0.3900000
92105	37	105	0.3523810
92114	30	86	0.3488372
92110	83	247	0.3360324
92106	47	140	0.3357143

1-10 of 10 rows

The top 10 location with high proportion of having high booking rate are:
92104,92102,92116,92107,92121,92113,92105,92114,92110,92106(zipcode)

Neighborhood Selection: We divided the top 10 location into 5 popular areas and then we found out the top 10 neighborhood that has high proportion of having high booking rates in each area.

Found the top 10 neighborhoods that has high proportion of having high booking rate in the area that contained three locations: 92104, 92102, 92116. (Three locations are near each other)

```
dfNeighbor1<-
  df %>%
  filter(zipcode %in% c(92104, 92102, 92116)) %>%
  group_by(neighbourhood) %>%
  mutate(sum_high_neighbor = sum(high_booking_rate)) %>%
  select(neighbourhood, sum_high_neighbor)

dfNeighbor_NumOfHomes1<-
  dfNeighbor1 %>%
  group_by(neighbourhood) %>%
  tally()

total_nei1 <- merge(dfNeighbor_NumOfHomes1, dfNeighbor1, by = "neighbourhood")

total_nei1<-
  total_nei1 %>%
  mutate(neigh_rate = sum_high_neighbor/n)

totalNeiNoDuplicate1 <- total_nei1[!duplicated(total_nei1$neighbourhood),]

total_nei_rank1<-
  totalNeiNoDuplicate1 %>%
  select(neighbourhood, sum_high_neighbor, n, neigh_rate) %>%
  arrange(desc(neigh_rate))
total_nei_rank1
```

neighbourhood <chr>	sum_high_neighbor <dbl>	n <int>	neigh_rate <dbl>

neighbourhood <chr>	sum_high_neighbor <dbl>	n <int>	neigh_rate <dbl>
Balboa Park	2	2	1.0000000
Stockton	7	11	0.6363636
Adams North	17	32	0.5312500
South Park	45	85	0.5294118
Sherman Heights	35	75	0.4666667
Kensington	15	33	0.4545455
North Park	167	384	0.4348958
Cherokee Point	10	23	0.4347826
Grant Hill	18	42	0.4285714
University Heights	42	102	0.4117647

1-10 of 21 rows Previous **1** 2 3 Next

```
total_nei_top10<-head(total_nei_rank1,10)
total_nei_top10
```

	neighbourhood <chr>	sum_high_neighbor <dbl>	n <int>	neigh_rate <dbl>
1	Balboa Park	2	2	1.0000000
2	Stockton	7	11	0.6363636
3	Adams North	17	32	0.5312500
4	South Park	45	85	0.5294118
5	Sherman Heights	35	75	0.4666667
6	Kensington	15	33	0.4545455
7	North Park	167	384	0.4348958
8	Cherokee Point	10	23	0.4347826
9	Grant Hill	18	42	0.4285714
10	University Heights	42	102	0.4117647

1-10 of 10 rows

The neighbourhood that has highest proportion of having high booking rate in this area is Balboa Park

Found the top 10 neighborhoods according to the proportion of having high booking rates in each neighborhood in the area that contained two locations: 92107, 92106. (Two locations are next to each other)

```
dfNeighbor2<-
  df %>%
  filter(zipcode %in% c(92107, 92106)) %>%
  group_by(neighbourhood) %>%
  mutate(sum_high_neighbor = sum(high_booking_rate)) %>%
  select(neighbourhood, sum_high_neighbor)

dfNeighbor_NumOfHomes2<-
  dfNeighbor2 %>%
  group_by(neighbourhood) %>%
  tally()

total_nei2 <- merge(dfNeighbor_NumOfHomes2, dfNeighbor2, by = "neighbourhood")

total_nei2<-
  total_nei2 %>%
  mutate(neigh_rate = sum_high_neighbor/n)

totalNeiNoDuplicate2 <- total_nei2[!duplicated(total_nei2$neighbourhood),]

total_nei_rank2<-
  totalNeiNoDuplicate2 %>%
  select(neighbourhood, sum_high_neighbor, n, neigh_rate) %>%
  arrange(desc(neigh_rate))
total_nei_rank2
```

neighbourhood <chr>	sum_high_neighbor <dbl>	n <int>	neigh_rate <dbl>
Golden Hill	1	1	1.0000000
Loma Portal	11	19	0.5789474
Ocean Beach	145	324	0.4475309
Wooded Area	2	5	0.4000000
Roseville/Fleet Ridge	20	55	0.3636364
Point Loma Heights	45	133	0.3383459
Sunset Cliffs	7	26	0.2692308
La Playa	5	34	0.1470588
Midway District	0	1	0.0000000
Other	0	1	0.0000000

1-10 of 10 rows

```
total_nei_top10_2<-head(total_nei_rank2,10)
total_nei_top10_2
```

	neighbourhood <chr>	sum_high_neighbor <dbl>	n <int>	neigh_rate <dbl>
1	Golden Hill	1	1	1.0000000
2	Loma Portal	11	19	0.5789474
3	Ocean Beach	145	324	0.4475309
4	Wooded Area	2	5	0.4000000
5	Roseville/Fleet Ridge	20	55	0.3636364
6	Point Loma Heights	45	133	0.3383459
7	Sunset Cliffs	7	26	0.2692308
8	La Playa	5	34	0.1470588
9	Midway District	0	1	0.0000000
10	Other	0	1	0.0000000

1-10 of 10 rows

The neighbourhood that has highest proportion of having high booking rate in this area is Wooded Area.

Found the top 10 neighborhoods according to the proportion of having high booking rates in each neighborhood in the area that contained two locations: 92113, 92114. (Two locations are next to each other)

```
dfNeighbor3<-
  df %>%
  filter(zipcode %in% c(92113, 92114)) %>%
  group_by(neighbourhood) %>%
  mutate(sum_high_neighbor = sum(high_booking_rate)) %>%
  select(neighbourhood, sum_high_neighbor)

dfNeighbor_NumOfHomes3<-
  dfNeighbor3 %>%
  group_by(neighbourhood) %>%
  tally()

total_nei3 <- merge(dfNeighbor_NumOfHomes3, dfNeighbor3, by = "neighbourhood")

total_nei3<-
  total_nei3 %>%
  mutate(neigh_rate = sum_high_neighbor/n)
totalNeiNoDuplicate3 <- total_nei3[!duplicated(total_nei3$neighbourhood),]
total_nei_rank3<-
  totalNeiNoDuplicate3 %>%
  select(neighbourhood, sum_high_neighbor, n, neigh_rate) %>%
  arrange(desc(neigh_rate))
total_nei_top10_3<-head(total_nei_rank3,10)
total_nei_top10_3
```

	neighbourhood <chr>	sum_high_neighbor <dbl>	n <int>	neigh_rate <dbl>

	neighbourhood	sum_high_neighbor	n	neigh_rate
	<chr>	<dbl>	<int>	<dbl>
1	Southcrest	1	1	1.0000000
2	Skyline	4	6	0.6666667
3	Valencia Park	9	18	0.5000000
4	Jamacha Lomita	7	16	0.4375000
5	Logan Heights	20	46	0.4347826
6	Shelltown	3	7	0.4285714
7	Barrio Logan	7	18	0.3888889
8	Encanto	9	27	0.3333333
9	Mountain View	8	25	0.3200000
10	Bay Terraces	1	8	0.1250000

1-10 of 10 rows

The neighbourhood that has highest proportion of having high booking rate in this area is Southcrest.

Found the top 10 neighborhoods according to the proportion of having high booking rates in each neighborhood in the area: 92105.

```
dfNeighbor4<-
  df %>%
  filter(zipcode ==92105) %>%
  group_by(neighbourhood) %>%
  mutate(sum_high_neighbor = sum(high_booking_rate)) %>%
  select(neighbourhood, sum_high_neighbor)

dfNeighbor_NumOfHomes4<-
  dfNeighbor4 %>%
  group_by(neighbourhood) %>%
  tally()

total_nei4 <- merge(dfNeighbor_NumOfHomes4, dfNeighbor4, by = "neighbourhood")

total_nei4<-
  total_nei4 %>%
  mutate(neigh_rate = sum_high_neighbor/n)
total_nei4
```

neighbourhood	n	sum_high_neighbor	neigh_rate
<chr>	<int>	<dbl>	<dbl>
Azalea/Hollywood Park	15	3	0.20000000
Azalea/Hollywood Park	15	3	0.20000000
Azalea/Hollywood Park	15	3	0.20000000
Azalea/Hollywood Park	15	3	0.20000000
Azalea/Hollywood Park	15	3	0.20000000
Azalea/Hollywood Park	15	3	0.20000000
Azalea/Hollywood Park	15	3	0.20000000
Azalea/Hollywood Park	15	3	0.20000000
Azalea/Hollywood Park	15	3	0.20000000
Azalea/Hollywood Park	15	3	0.20000000

1-10 of 105 rows Previous **1** 2 3 4 5 6 ... 11 Next

```
totalNeiNoDuplicate4 <- total_nei4[!duplicated(total_nei4$neighbourhood),]

total_nei_rank4<-
  totalNeiNoDuplicate4 %>%
  select(neighbourhood, sum_high_neighbor, n, neigh_rate) %>%
  arrange(desc(neigh_rate))
total_nei_top10_4<-head(total_nei_rank4,10)

total_nei_top10_4
```

	neighbourhood <chr>	sum_high_neighbor <dbl>	n <int>	neigh_rate <dbl>
1	Chollas Creek	1	1	1.0000000
2	Fairmont Park	6	11	0.5454545
3	Cherokee Point	3	6	0.5000000
4	Normal Heights	1	2	0.5000000
5	Swan Canyon	6	12	0.5000000
6	Teralta East	1	2	0.5000000
7	Oak Park	5	11	0.4545455
8	Fairmont Village	4	9	0.4444444
9	Castle	3	7	0.4285714
10	Ridgeview/Webster	2	7	0.2857143

1-10 of 10 rows

The neighbourhood that has highest proportion of having high booking rate in this area is Fairmont Park.

Found the top 10 neighborhoods according to the proportion of having high booking rates in each neighborhood in the area: 92110.

```
dfNeighbor5<-
  df %>%
  filter(zipcode == 92110) %>%
  group_by(neighbourhood) %>%
  mutate(sum_high_neighbor = sum(high_booking_rate))

dfNeighbor_NumOfHomes5<-
  dfNeighbor5 %>%
  group_by(neighbourhood) %>%
  tally()

total_nei5 <- merge(dfNeighbor_NumOfHomes5, dfNeighbor5, by = "neighbourhood")

total_nei5<-
  total_nei5 %>%
  mutate(neigh_rate = sum_high_neighbor/n)

totalNeiNoDuplicate5 <- total_nei5[!duplicated(total_nei5$neighbourhood),]

total_nei_rank5<-
  totalNeiNoDuplicate5 %>%
  arrange(desc(neigh_rate))
total_nei_top10_5<-head(total_nei_rank5,10)
total_nei_top10_5
```

	neighbourhood <chr>	n <int>	id <dbl>	high_booking_rate <dbl>
1	Midtown	1	1057167	1
2	Midway District	13	1140622	1
3	Morena	65	1154481	1
4	Mission Hill	33	1126374	1
5	Bay Park	73	1062704	0
6	Old Town	29	1138576	1
7	Point Loma Heights	25	1073236	0
8	Loma Portal	1	1200364	0
9	Mission Valley West	7	1073779	0

9 rows | 1-5 of 152 columns

The neighbourhood that has highest proportion of having high booking rate in this area is Midtown.

Features Finding: Calculate the proportion of having high booking rate for Airbnb homes with specific feature appeared.

We checked almost all features but here we only keep features whcih may not be notied esily and maybe helpful for improving the probability to get high booking rate. #### Original Proportion of high booking rate and non-high booking rate

```
df %>%
  group_by(high_booking_rate) %>%
  tally() %>%
  mutate(rate = 100*n/sum(n))
```

high_booking_rate	n	rate
<dbl>	<int>	<dbl>
0	5800	71.21807
1	2344	28.78193

2 rows

Calculate the rate of getting high booking rate when the property type is cottage.

```
df %>%
  filter(property_type=="Cottage") %>%
  group_by(high_booking_rate) %>%
  tally() %>%
  mutate(rate = 100*n/sum(n))
```

high_booking_rate	n	rate
<dbl>	<int>	<dbl>
0	85	45.94595
1	100	54.05405

2 rows

Calculate the rate of getting high booking rate when the room type is Entire home/apt.

```
df %>%
  filter(room_type=="Entire home/apt") %>%
  group_by(high_booking_rate) %>%
  tally() %>%
  mutate(rate = 100*n/sum(n))
```

high_booking_rate	n	rate
<dbl>	<int>	<dbl>
0	4150	69.95954
1	1782	30.04046

2 rows

Calculate the rate of getting high booking rate when the number of bedrooms is one.

```
df %>%
  filter(bedrooms==1) %>%
  group_by(high_booking_rate) %>%
  tally() %>%
  mutate(rate = 100*n/sum(n))
```

high_booking_rate	n	rate
<dbl>	<int>	<dbl>
0	2793	69.46033
1	1228	30.53967

2 rows

Calculate the rate of getting high booking rate when bathroom/bedroom == 1.

```
df %>%
  filter(availability_bathvsroom==1) %>%
  group_by(high_booking_rate) %>%
  tally() %>%
  mutate(rate = 100*n/sum(n))
```

high_booking_rate	n	rate
<dbl>	<int>	<dbl>
0	3755	70.25257
1	1590	29.74743

2 rows

Calculate the rate of getting high booking rate when beds/bedroom == 2.

```
df %>%
  filter(availability_bedvsroom==2) %>%
  group_by(high_booking_rate) %>%
  tally() %>%
  mutate(rate = 100*n/sum(n))
```

high_booking_rate <dbl>	n <int>	rate <dbl>
0	951	66.97183
1	469	33.02817

2 rows

Amenities to be checked: Dog friendly, Luggage dropped off, Self check-in, Barbecue, Microwave, Coffee.

```
df %>%
  filter(amenities_dog==1) %>%
  group_by(high_booking_rate) %>%
  tally() %>%
  mutate(rate = 100*n/sum(n))
```

high_booking_rate <dbl>	n <int>	rate <dbl>
0	165	47.68786
1	181	52.31214

2 rows

```
df %>%
  filter(amenities_luggage==1) %>%
  group_by(high_booking_rate) %>%
  tally() %>%
  mutate(rate = 100*n/sum(n))
```

high_booking_rate <dbl>	n <int>	rate <dbl>
0	1006	53.71062
1	867	46.28938

2 rows

```
df %>%
  filter(amenities_selfcheckin==1) %>%
  group_by(high_booking_rate) %>%
  tally() %>%
  mutate(rate = 100*n/sum(n))
```

high_booking_rate <dbl>	n <int>	rate <dbl>
0	1915	54.66743
1	1588	45.33257

2 rows

```
df %>%
  filter(amenities_bbq==1) %>%
  group_by(high_booking_rate) %>%
  tally() %>%
  mutate(rate = 100*n/sum(n))
```

high_booking_rate <dbl>	n <int>	rate <dbl>
0	1283	62.76908
1	761	37.23092

2 rows

```
df %>%
  filter(amenities_microwave==1) %>%
  group_by(high_booking_rate) %>%
  tally() %>%
  mutate(rate = 100*n/sum(n))
```

high_booking_rate <dbl>	n <int>	rate <dbl>
0	2643	59.28668
1	1815	40.71332

2 rows

```
df %>%
  filter(amenities_coffee==1) %>%
  group_by(high_booking_rate) %>%
  tally() %>%
  mutate(rate = 100*n/sum(n))
```

high_booking_rate <dbl>	n <int>	rate <dbl>
0	2557	57.83759
1	1864	42.16241

2 rows

Calculate the rate of getting high booking rate when the word (like sea) is contained in the desciption.

```
df %>%
  filter(description_sea==1) %>%
  group_by(high_booking_rate) %>%
  tally() %>%
  mutate(rate = 100*n/sum(n))
```

high_booking_rate <dbl>	n <int>	rate <dbl>
0	1512	66.28672
1	769	33.71328

2 rows

Keywords to be checked: restaurant, bars.

```
df %>%
  filter(description_restaurant==1) %>%
  group_by(high_booking_rate) %>%
  tally() %>%
  mutate(rate = 100*n/sum(n))
```

high_booking_rate <dbl>	n <int>	rate <dbl>
0	2223	67.79506
1	1056	32.20494

2 rows

```
df %>%
  filter(description_bars==1) %>%
  group_by(high_booking_rate) %>%
  tally() %>%
  mutate(rate = 100*n/sum(n))
```

high_booking_rate <dbl>	n <int>	rate <dbl>
0	944	66.61962
1	473	33.38038

2 rows

Cancellation policy

```
df %>%
  filter(cancellation_moderate==1) %>%
  group_by(high_booking_rate) %>%
  tally() %>%
  mutate(rate = 100*n/sum(n))
```

high_booking_rate	n	rate
<dbl>	<int>	<dbl>
0	1239	57.01795
1	934	42.98205

2 rows

Host response time

```
df %>%
  filter(host_response_time=="within an hour") %>%
  group_by(high_booking_rate) %>%
  tally() %>%
  mutate(rate = 100*n/sum(n))
```

high_booking_rate	n	rate
<dbl>	<int>	<dbl>
0	3224	62.01193
1	1975	37.98807

2 rows

Host is super host

```
df %>%
  filter(host_is_superhost==TRUE) %>%
  group_by(high_booking_rate) %>%
  tally() %>%
  mutate(rate = 100*n/sum(n))
```

high_booking_rate	n	rate
<dbl>	<int>	<dbl>
0	1553	50.70193
1	1510	49.29807

2 rows

Host identify verified by Airbnb

```
df %>%
  filter(host_identity_verified==TRUE) %>%
  group_by(high_booking_rate) %>%
  tally() %>%
  mutate(rate = 100*n/sum(n))
```

high_booking_rate	n	rate
<dbl>	<int>	<dbl>
0	2457	65.95973
1	1268	34.04027

2 rows

Host verified by review

```
df %>%
  filter(host_verifications_review==1) %>%
  group_by(high_booking_rate) %>%
  tally() %>%
  mutate(rate = 100*n/sum(n))
```

high_booking_rate	n	rate
<dbl>	<int>	<dbl>
0	4073	66.44372
1	2057	33.55628

2 rows

Result & Findings

Location Selection Results

After data visualization and analysis, we found that those zip codes with relatively higher booking rates can be grouped into two main areas: “Balboa Park Area” and “Ocean Beach Area”. “Balboa Park Area” is an area centered on Balboa Park. ”Ocean Beach Area” is an area located on the upper bay of San Diego Bay.

Balboa Park Area (zip code: 92104, 92102, 92116, 92113, 92114, 92105)

The wonderful Balboa Park is definitely the best attraction in San Diego and it includes a lot of different places to have fun such as San Diego Art & Space Museum, San Diego Zoo, and Japanese friendship garden. The house at or near Balboa park will never worry about booking. It is just like the Mammoth Hot Springs Hotel & Cabins at YellowStone Park. People even need to book it one year in advance because it is rare that a hotel is located at the YellowStone.
The Balboa Park neighborhood has the highest proportion (100%) of getting high booking rates in this area, but the number of Airbnb homes in that neighborhood is too small, so considering both the proportion of getting high booking rates and the number of homes in each neighborhood, we chose the North Park as the best neighborhood for investors in Balboa Park area.

Ocean Beach & UCSD Area (zip code: 92107, 92106, 92110, 92121)

Ocean Beach lies on the Pacific Ocean at the estuary of the San Diego River, at the western terminus of Interstate 8. Located about 7 miles (11 km) northwest of Downtown San Diego.
It is a beautiful beach town and historic business district with attractions, restaurants, sunsets, shops, events,and fun.
After analyzing the descriptions in the data set, we found that Airbnb, which has sea views, beaches, restaurants, bars, and downtown, has a higher booking rate for Airbnb, which is relatively higher. Compared to the Balboa park area and Ocean Beach & UCSD area, the second one will be more likely to have a sea view, bars, and beach. Additionally, UCSD is a potential area because the academic student population is increasing by 3% annually. Therefore, our final site selection will be the Ocean Beach & UCSD area.
Since most of the customers of Airbnb are coming for vacation. We will take care about the tourist attractions, public securities, cost of living, and transportations when we analyze the location. To extract more information from the data, we need to analyze the neighborhood of each Airbnb in a specific area. We choose those neighborhoods whose neighborhood rate is above 0.5 from these two locations. The average neighborhood rate of San Diego is 0.29. Therefore, our choices are 70% above the average.
Although some specific neighborhoods, midtown and university city, have a 100% high booking rate, we are going to choose some place with relatively higher supply. Finally, we choose the ocean beach neighborhood which is a place located in the northwest of Ocean Beach & UCSD area. It is in a quiet residential area with sea views, bars, restaurants, and beach. The residential areas are close to a lot of attractions and landmarks. It has convenient transportation, restaurants, and supermarkets. The real estate market is mature and has large supply. The professional housing management company is nearby. If investors want to purchase multiple houses, this is the best place to invest.

Features Finding Results

This part we looked into property type, room type, amenities, description, host verification method, cancellation policy, super host, etc.
We compare the original proportion of high booking rate with the original proportion of high booking rate with specific features
The original proportion of high booking rate is only 28.8%
But for Property type-Cottage, the proportion of cottage having a high booking rate increase up to 54%.
For Room type-Entire apartment/house, the proportion increased to 31%.
One bedroom airbnb homes has the proportion of high booking rate up to to 32%.
1B1B homes have the proportion of high booking rate up to 30%.
When the ratio between beds and bedrooms is 2, the proportion of high booking rate increase to 33%.
Dog Friendly homes has the proportion of high booking rate up to 52%.
Homes providing Luggage Drop-off Allowance have the proportion of high booking rate up to 46%.
Homes providing Self Check-in Service have the proportion of high booking rate up to 45%.
Homes providing coffee have the proportion of high booking rate up to 42%.
Homes providing microwave have the proportion of high booking rate up to 41%.
Homes providing barbecue place or tool have the proportion of high booking rate up to 37%.
Homes including sea view in the description have the proportion of high booking rate up to 34%.
Homes mentioning bar in the description have the proportion of high booking rate up to 33%.
Homes mentioning restaurants in the description have the proportion of high booking rate up to 32%.
Homes having moderate cancellation policy have the proportion of high booking rate up to 43%.
Homes with host response time within 1 hour have the proportion of high booking rate up to 38%.
Homes with a super host have the proportion of high booking rate up to 49%.
Homes were verified by Airbnb or having review as verification way have the proportion of high booking rate up to 34%.

Conclusion and Discussion

Project Summary

San Diego, one of the most attractive cities in the United States, attracts lots of visitors around the world every year. The beaches, the sunshine, the sea wildlife and Latin American culture make every visitor who arrives in this city feel like living in a paradise. Therefore, Airbnb business in this city could be a very profitable and risky business. On the profitable side, millions of visitors came to this city to make San Diego become a very big rental market. On the risky side, thousands of business owners are running their rental business, so this is a very competitive market. In order to make more profits, we conducted this research to find a good start point for running Airbnb business in this city.

For location, we decided to choose North Park and Ocean Beach as our candidate locations to open our Airbnb home, because we found that these two locations are the most popular destinations for visitors who travel to San Diego based on our research. Therefore, we will seek our target house within these two areas. However, after we did features finding, the cottage stands out as the proporty type, so we will choose Ocean Beach, which is a better location for a cottage, to be our final decision.

For our recommendation, we want to buy or build a cottage as our Airbnb home. In San Diego, cottage as a small house typically near lake or beach is very popular and are more likely to have a high booking rate based on our analysis results. In addition, our Airbnb house will have one bathroom and one bedroom with two beds, and the cottage will be rented in its entirety. The policy of cancellation of the booking better be moderate and hosts should try to respond with one hour.

For the amenities, besides the necessary features like AC, Heating and so on, we found that the features like Dog Friendliness, Luggage Drop-off Allowance, Self-check-in, Coffee, Microwave, Barbecue are very important. Sea View, Bars Access and Restaurants Access should be mentioned in the description of Airbnb home to attract more attentions from customers.

Moreover, after we simulated our recommendation and test it with our model, we found that improvement is still needed. Based on our analysis results, there are some other espects we should pay attention to for increasing the probability of becoming a high_booking_rate home, which includes, host_is_super_host,host_identity_verified and features related to reviews. They may influence the high booking rate a lot. Even we did not include them when having our objective as initial aqusition of a Airbnb home, we will recommend hosts to work on them once their business starts in order to accomplish a high booking rate.

Limitation

From the dataset we had, we found several columns which were related to time. These columns were availability 30, availability 60, availability 90 and availability 365. Therefore, we noticed that many Airbnb business owners didn't open their homes for the whole year. However, we did not have the specific time for the availability, whcih may indicate peak seasons and lack seasons within one year. The availability as well as the information about peak seasons and lack seasons would be important for high booking rate and very crucial for Airbnb business strategy, because they could be a guideline for hosts to decide their optimal schedule for setting prices and opening their home for rental. In theory, Opening home during peak season may be more likely to have a high booking rate. These strategies can increase revenues and reduce costs. Moreover, we didn't have detailed cost data to help us to conduct our research, but cost is still a very important aspect for running a business. If we can't obtain detailed cost data, our research would have a big drawback which could bring failure for our business.

Future research

Future research will be implemented by two aspects. The first aspect will be analysing the time series of prices of Airbnb rooms within our target locations. We want to know how the prices change during certain time periods to help us fully understand the complete picture of San Diego's Airbnb Market. The time series of prices will also help us to find peak seasons and lack seasons, so we can easily decide our opening season of our Airbnb home. Another aspect will be obtaining cost data and analyzing cost data. Exact cost data can help us to have a more detailed pricing strategy and more well-organized house and feature development plan.

Reference

Airbnb: Summer Rentals in San Diego County Generated $122 Million: link (https://timesofsandiego.com/business/2019/09/16/airbnb-summer-rentals-in-san-diego-county-generated-122-million/)

Airbnb® | San Diego Bay - Vacation Rentals & Places to Stay:link (https://zh.airbnb.com/s/San-Diego-Bay--CA)

San Diego's Airbnb Hotspots: link (https://www.voiceofsandiego.org/business/san-diegos-airbnb-hotspots/)

What is Airbnb's Superhost Status Really Worth?: link (https://www.airdna.co/blog/airbnb_superhost_status)

Appendeix

Dataset for San Deigo Market: link (https://drive.google.com/file/d/1uZzJNwzCobZjtfpgBxp8ZlTLYXEuTNxT/view?usp=sharing)

Nice vedio about XGBoost for Classification: link (https://www.youtube.com/watch?v=8b1JEDvenQU)