# Case Study for Airbnb in Washington D.C:

# What should investors do to achieve high booking rate?

## 1. Cover Page

**Data Mining and Predictive Analysis**

**Title:**

Case Study for Airbnb in Washington D.C:
What should investors do to achieve high booking rate?
Explanatory Model and Prediction Model for Achieveing High Airbnb Booking Rate

**Market:** Washington D.C.

"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, illusions, and other items included in the manuscript do not carry any infringement/plagiarism issue upon any existing copyright materials."

Team member 1 Rui Ma

Team member 2 Wanyun Yang

Team member 3 Jingyu Liao

Team member 4 Zilinmei Ye

Team member 5 Gongshun Wang

Team member 6 Mengyuan Jin

## 2. Executive Summary

Airbnb has been essential part of leisure and tourism industry. An increasing number of travelers takes rental home as their first choice for destination accommodation. Therefore, making a right purchase decision for an Airbnb property will finally turn out to be a good investment. Our goal of this study is to help investors make such decisions, specifically in the D.C. market.

The dataset for this D.C market study has over 5450 Airbnb records. Variables are divided into 4 categories: community features, property features, management features and review features. For community features, we integrated external dataset to include features like safety(crime), economy(income per capita) and transportation(distance to the nearest

metro entrance). We used visualization and regression to figure out what are the community features of the properties that have a high booking rate; for property features, we followed a similar path as community part. We also include all four categories of variables to predict a potentially high booking rate property using ensemble methods and analyze the features that could improve the possibility to be a good buy.

As a result, we found that for all the ward areas in Washington DC., ward 1, 2, 6 would be the most optimistic choice. Apartment and townhouses are the most popular type of property on airbnb listing. The booking rate would even be higher if there are multiple bedrooms and bathrooms. If the host owns several airbnb properties, this would further boost the booking rate. What is novel and interesting about our study is that the crime rate in each ward actually does not contribute to any significant effects on the booking rate, which it can implies that in the city of washington DC, there is not a clear segregation of safe neighborhoods with relatively chaotic neighborhoods. Hopefully these findings would serve as an useful guide for Airbnb business investors in D.C. and help to achieve a high booking rate.

## 3. Research Questions

**The main research question** for the study is to analyze what are the features of a property that could potentially have a high booking rate in the Washington DC area. Under this main question, there are three subsequent questions.

**The first sub question** is what are the community features of the properties that have a high booking rate. For securing a housing property. With the background domain knowledge in the real estate market, the location of the property is one of the most critical factors to be considered. The property with much smaller space but located in a metropolitan area of the city could be listed on airbnb for a much higher price. The safety level of the allocated neighbourhood within Washington DC would also be another potential factor that affects booking rate. Therefore, we chose to consider this community location features to be the first vital component to take into account for the analysis.

**The second sub question** we have raised is what are the significant housing features of airbnb rentals that could potentially give a high booking rate. Other than location, the capacity of houses and the amenities accompanied with the houses is the next vital component to be considered for customers on deciding which airbnb listing to book. Customers visiting airbnb have various purposes. The most common reasons for DC traveling could be attending a meeting. As Washington DC is the capital of America, there are frequent holdings of nation-wide meetings and gatherings compared to other cities.The other reason could be for tourism purposes, where capital hill and washington monument are both signatures tourism sites on the east coast. These different purposes of customers brought up different demands and requirements when booking and airbnb houses. The size of customer groups and the different preference on amenities all contributes to this factor. Especially for the DC market, this portion of analysis would help the investor understand what are the most popular settings for property features on airbnb.

**The third sub question** we have raised is that overall, will a certain property have a high booking rate consider all the related characteristics. After the detailed analysis on the community influence and property's physical feature effects, it is important that we included all the relevant characters of certain airbnb properties to perform a combined analysis. In this case, any interactive effects of different type of features would also be considered and create more accurate analysis results.

# 4. Methodology and Research

## 4.1 Question 1: Community characteristics (Where the house should be?)

### 4.1.1 Data preparation (Integrating external data set)

Firstly, we added an external dataset to calculate the following variables that would aid in the project analysis. These newly added variables are :1. the distance of each home to its nearest metro station; 2. The ward number of the neighborhood that each home belongs to; 3. The total crime number of each ward for the last two years; 4. The average income per capita for the resident within each ward area. The following are the coding portion for adding external dataset and construct the above new variables.

**import libraries**

```
library("tidyverse")
library("tidymodels")
library("plotly")
library("skimr")
library("caret")
library('cowplot')
library("ggmap")
library("Imap")
library("caret")
library(rpart.plot)
library("randomForest")
```

**Filter out DC data with a random control starting with '107'**

```
df_dc <-
  read_csv("airbnbTrain.csv")

colnames(df_dc)[66] <- "randomControl"
df_dc <-
  df_dc %>%
  filter(randomControl >= 107000) %>%
  filter(randomControl < 108000)
```

**adding of variable - min_MetroEntranceDist: Distance to the Nearest Metro Entrance.**

```
# Get DC metro entrances geographic coordinates.
metro_address <-
```

```
  read_csv("./original_data/Metro_Station_Entrances_in_DC.csv") %>%
  select(X,Y)

# Calculate the geodesic distance to the nearest metro entrance.
# Geographical distance is the distance measured along the surface of the ear
th.
df_dc$min_MetroEntranceDist <- NA
for (i in 1:nrow(df_dc)) {
     df_dc$min_MetroEntranceDist[i] = gdist(lon.1 = metro_address$X[1],
                           lat.1 = metro_address$Y[1],
                           lon.2 = df_dc$longitude[i],
                           lat.2 = df_dc$latitude[i],
                           units="miles")
     for (j in 2:nrow(metro_address)) {
     temp = gdist(lon.1 = metro_address$X[j],
                           lat.1 = metro_address$Y[j],
                           lon.2 = df_dc$longitude[i],
                           lat.2 = df_dc$latitude[i],
                           units="miles")
     if (df_dc$min_MetroEntranceDist[i] > temp){
       df_dc$min_MetroEntranceDist[i] = temp
     }
    }
}
```

**Adding of variable - Ward Number: which ward does the neighborhood of a property belong to.**

```
ward1 <- c("Adams Morgan", "Columbia Heights", "Howard University","Lanier He
ights", "Mount Pleasant", "Park View", "Pleasant Plains")
ward2 <- c("Burleith",            "Georgetown", "Logan Circle", "Penn Quarter
", "Sheridan Kalorama","West End")
ward3 <- c("American University Park",       "Forest Hills", "Foxhall", "Frie
ndship Heights", "Glover Park", "Kent", "Massachusetts Heights")
ward4 <- c("Barnaby Woods", "Brightwood", "Brightwood Park", "Chevy Chase", "
Colonial Village", "Crestwood",           "Takoma")
ward5 <- c("Arboretum", "Bloomingdale           "Pleasant Hill", "Queens Chap
el", "Stronghold", "Trinidad", "Truxton Circle", "Woodridge")
ward6 <- c("Barney Circle",            "Near Northeast", "NoMa", "Shaw", "Sou
thwest Waterfront", "Sursum Corda", "Swampoodle")
ward7 <- c("Benning Heights",            "Twining")
ward8 <- c("Anacostia", "Congress Heights",            "Garfield Heights", "K
nox Hill", "Shipley Terrace", "Washington Highlands", "Woodland")

# neighborhoods that cross several wards
ward13 <- c("Woodley Park")
ward34 <- c("Chevy Chase")
ward45 <- c("Riggs Park")
ward126 <- c("Shaw")
ward26 <- c("Mount Vernon Square")
```

```
ward12 <- c("U Street Corridor")
wardNeighboorhood <- list(ward1, ward2, ward3, ward4, ward5, ward6, ward7, wa
rd8, ward12, ward13, ward26, ward126, ward34, ward45)
names(wardNeighboorhood) <- c("1", "2", "3", "4", "5", "6", "7", "8","12", "1
3", "26", "126", "34", "45")
df_dc$wardName <- NA
for (i in 1:nrow(df_dc)) {
  for (j in 1:length(wardNeighboorhood)) {
    if (df_dc$neighbourhood[i] %in% wardNeighboorhood[[j]]){
      df_dc$wardName[i] = names(wardNeighboorhood[j])
    }
  }
}
```

**Adding of variable - Crime: total crime numbers its ward over past two years.**

```
df_dcCrime <- read_csv("./original_data/dc-crimes-search-results.csv")

df_dcCrime_ward <-
  df_dcCrime %>%
  group_by(WARD) %>%
  tally()
df_dcCrime_ward <-
  df_dcCrime_ward %>%
  rbind(c(12, (9148+13409)/2)) %>%
  rbind(c(13, (9148+4036)/2)) %>%
  rbind(c(26, (13409+11241)/2)) %>%
  rbind(c(126, (9148+13409+11241)/3)) %>%
  rbind(c(34, (4036+6004)/2)) %>%
  rbind(c(45, (6004+10161)/2))

df_dc$wardCrime <- NA
for (i in 1:nrow(df_dc)) {
  for (j in 1:nrow(df_dcCrime_ward)) {
    if (!is.na(df_dc$wardName[i]) && toString(df_dc$wardName[i]) == df_dcCrim
e_ward$WARD[j]){
      df_dc$wardCrime[i] = df_dcCrime_ward$n[j]
    }
  }
}
```

**Adding of variable - Per Capita Income: Per capita income of its ward. Per capita income is total income divided by total population.**

```
df_dcPerCapIncome <- read_csv("./original_data/perCapIncomeWard.csv")

df_dc$wardPerCapIncome <- NA
for (i in 1:nrow(df_dc)) {
  for (j in 1:nrow(df_dcPerCapIncome)) {
    if (!is.na(df_dc$wardName[i]) && toString(df_dc$wardName[i]) == df_dcPerC
apIncome$Ward[j]){
```

```
      df_dc$wardPerCapIncome[i] = df_dcPerCapIncome$PerCapitalIncome[j]
    }
  }
}
```

### 4.1.2 Varible analysis

For the model analysis on the community feature of the homes, our first step is to perform variable analysis to filter out the appropriate variables to include in this model. From all the relevant variables in the data set, we have picked 10 variables to begin the variable analysis and filtered out 5 variables to include in the model. The following are the reasons variables are being filtered out through variable analysis. **City**: All are basically (except one) Washington DC, but written in different ways like "Washington, D.C.","Washington DC", and "Washington D.C.". Therefore, we will not include this factor in the model. **Neighborhood_overview**: The variable is strings written by the renters and contains many missing values (1653 missing records), so it is difficult to use. **Neighbourhood**: There are 107 unique neighbourhoods in the training set. We have to do some classification since the number is very big, we cannot just create a dummy variable based on that. **transit**: Similar to some other variables, the variable is strings written by the renters, so it is hard to use. **zipcode**: There are 31 unique zip codes in the training set and 26 unique values in the validation set. We have to do some classification since the number is very big, we cannot just create a dummy variable based on that.

```
dcTrain <-read_csv("airbnbDC.csv")

skim(dcTrain)
```

The dataset has 5492 rows and 70 columns that contain 31 character type variables, 1 date variable, 9 logical variables, and 29 numeric ones. As we are only considering community features here, according to the data dictionary, I picked out the following variable for further exploration.

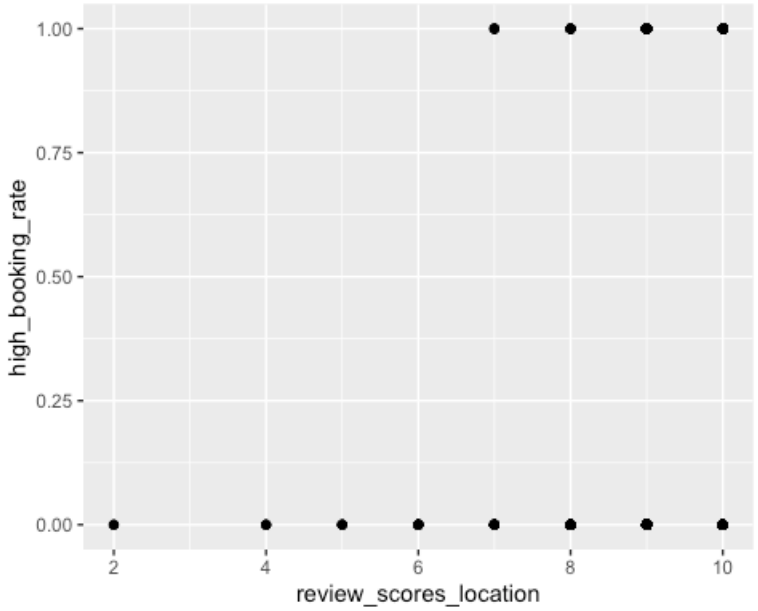
```
factors<-dcTrain %>%
  select(high_booking_rate,city , neighborhood_overview , neighbourhood, revi
ew_scores_location , transit , zipcode, min_MetroEntranceDist, wardCrime, war
dPerCapIncome)
factors

dcTrain$zipcode<-gsub( " ", "",dcTrain$zipcode)
dcTrain$zipcode<-gsub( "DC", "",dcTrain$zipcode)
dcTrain$zipcode<-substr(dcTrain$zipcode, 0, 5)
unique(dcTrain$zipcode)

map(factors, ~sum(is.na(.)))

## $high_booking_rate
## [1] 0
##
## $city
## [1] 3
```

```
##
## $neighborhood_overview
## [1] 1667
##
## $neighbourhood
## [1] 20
##
## $review_scores_location
## [1] 1127
##
## $transit
## [1] 1564
##
## $zipcode
## [1] 113
##
## $min_MetroEntranceDist
## [1] 0
##
## $wardCrime
## [1] 534
##
## $wardPerCapIncome
## [1] 534
```

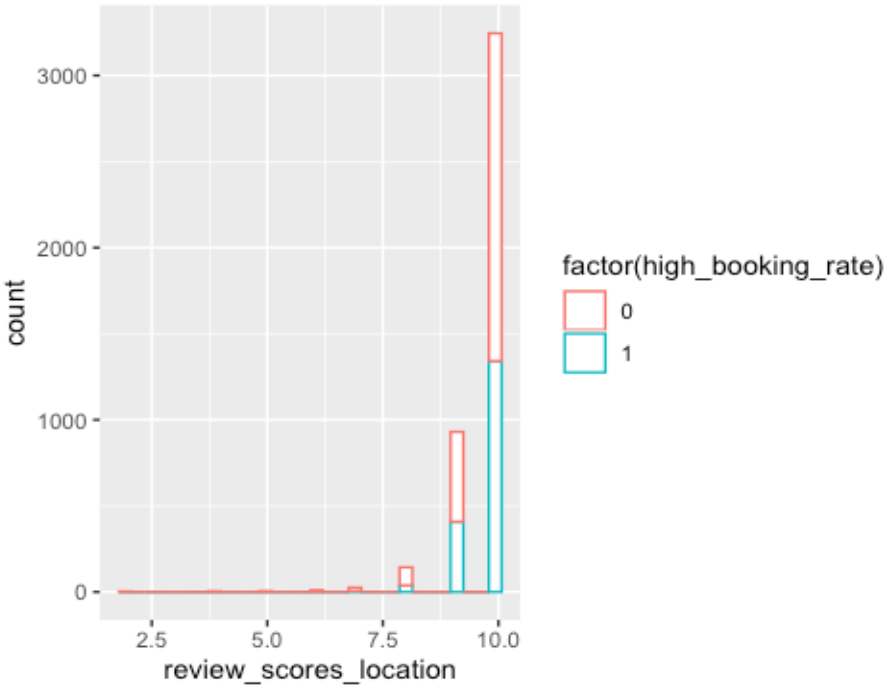
These five are the final ones I decided to use for modeling. They are wardName, review_scores_location, min_MetroEntranceDist, wardCrime, wardPerCapIncome, and wardName.

### review_scores_location variable analysis

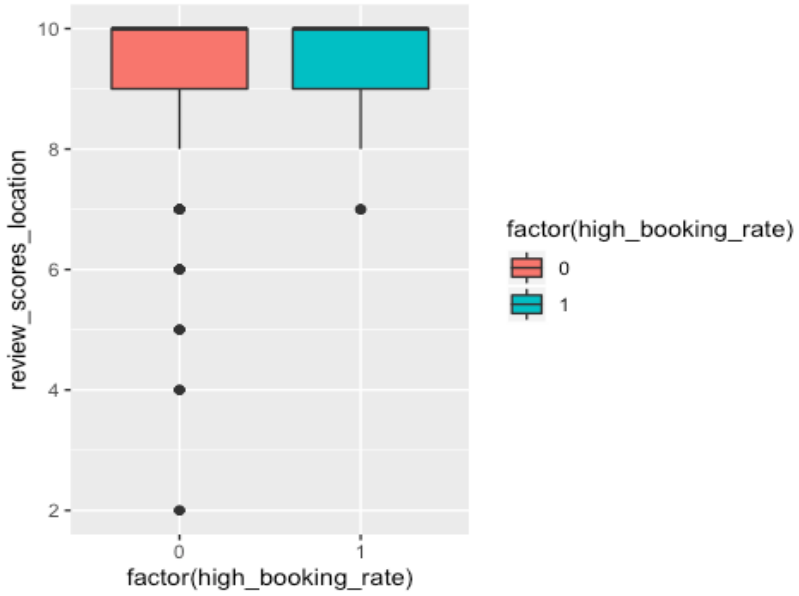
```
plot1 <- dcTrain %>%
  ggplot(aes(x =review_scores_location, y = high_booking_rate)) + geom_point(
)
plot1
```




```
plot2<-dcTrain%>%ggplot(aes(x=review_scores_location, color=factor(high_booki
ng_rate))) +geom_histogram(fill="white")
plot2
```



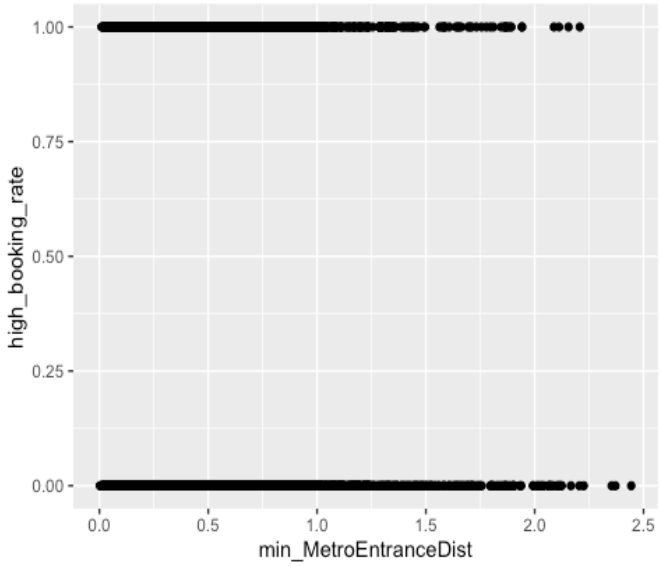

```
plot3<-dcTrain%>%ggplot(aes(x=factor(high_booking_rate),y=review_scores_locat
ion,fill=factor(high_booking_rate)))+geom_boxplot()
plot3
```



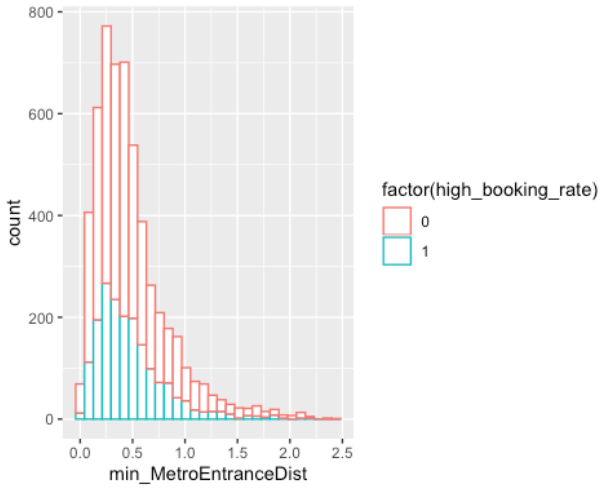

Review_scores_location is actually what we need, a numerical measure of location review. However, there are 1127 missing scores in the dataset. From the scatter plot of the available scores and the high booking rate, we can see if the score is high, the houses may have a high booking rate or not. While when the score is less or equal to 6, no houses in this category have high booking rate. The histogram colored by high booking rate is clearer that most renders give a high review score of the location. From the boxplot, we can see the mean, and quartile of both groups are high. The low scores are determined as outliers. When we use it in the model, we choose to leave it be, just deleting the rows with NA value in this case.

**min_MetroEntranceDist variable analysis**

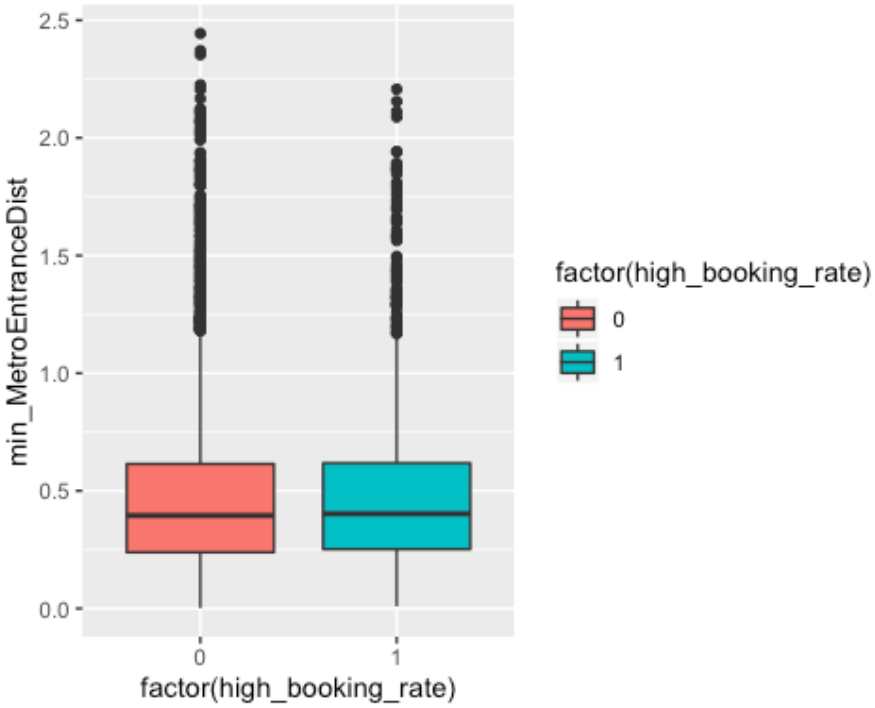
```
plot4 <- dcTrain %>%
  ggplot(aes(x =min_MetroEntranceDist, y = high_booking_rate)) + geom_point()
plot4
```




```
plot5<-dcTrain%>%ggplot(aes(x=min_MetroEntranceDist, color=factor(high_bookin
g_rate))) +geom_histogram(fill="white")
plot5
```



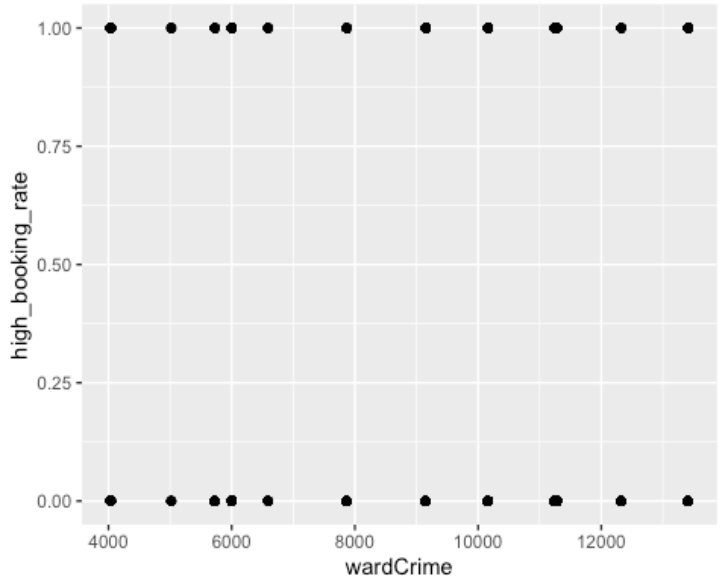

```
plot6<-dcTrain%>%ggplot(aes(x=factor(high_booking_rate),y=min_MetroEntranceDi
st,fill=factor(high_booking_rate)))+geom_boxplot()
plot6
```



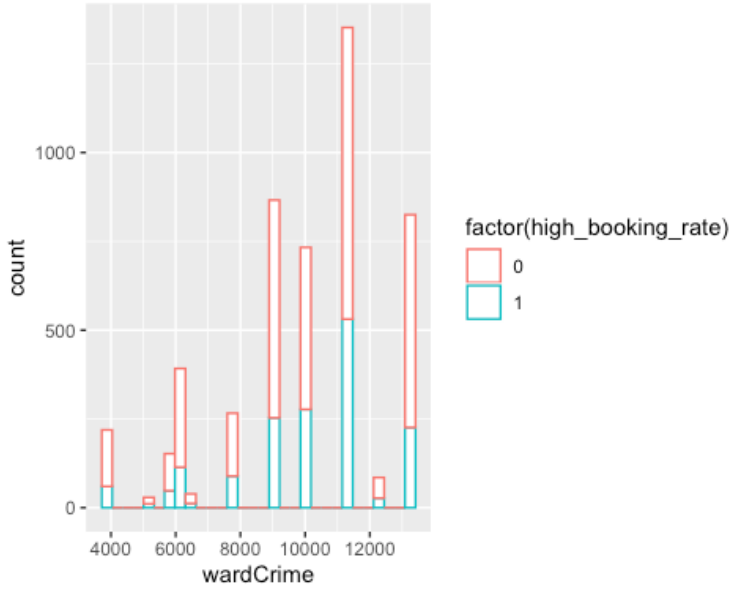

From the histogram, we can tell people tend to choose the properties that are close to the metro station. The bar reaches the peak around 0.5 miles, which is reasonable since people do not have vehicle to use when they visit other places. At that moment, the transportation resource nearby is crucial. From the boxplot, we still cannot see much difference between the two groups. Means and quartile are roughly the same. The range of the distance for non-popular houses are bigger because of the outliers at the upper corner.

**wardCrime variable analysis**

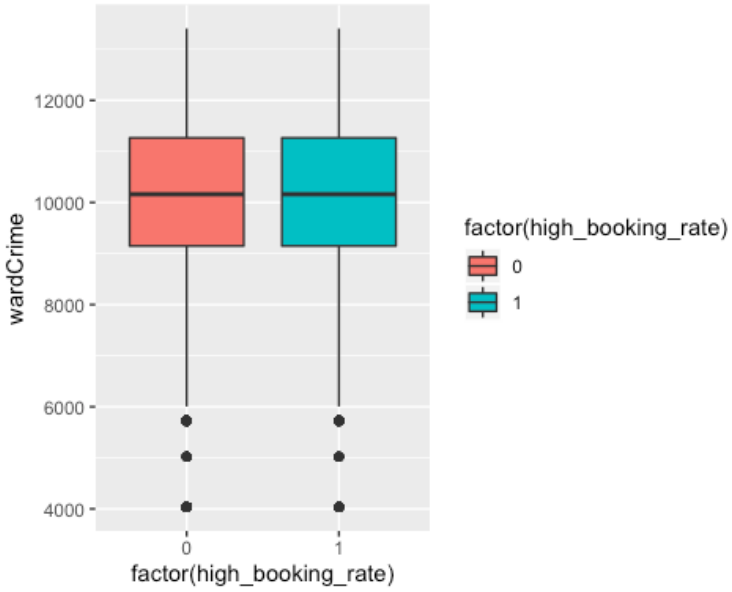
```
plot7 <- dcTrain %>%
  ggplot(aes(x =wardCrime, y = high_booking_rate)) + geom_point()
plot7
```




```
plot8<-dcTrain%>%ggplot(aes(x=wardCrime, color=factor(high_booking_rate))) +g
eom_histogram(fill="white")
plot8
```




```
plot9<-dcTrain%>%ggplot(aes(x=factor(high_booking_rate),y=wardCrime,fill=fact
or(high_booking_rate)))+geom_boxplot()
plot9
```

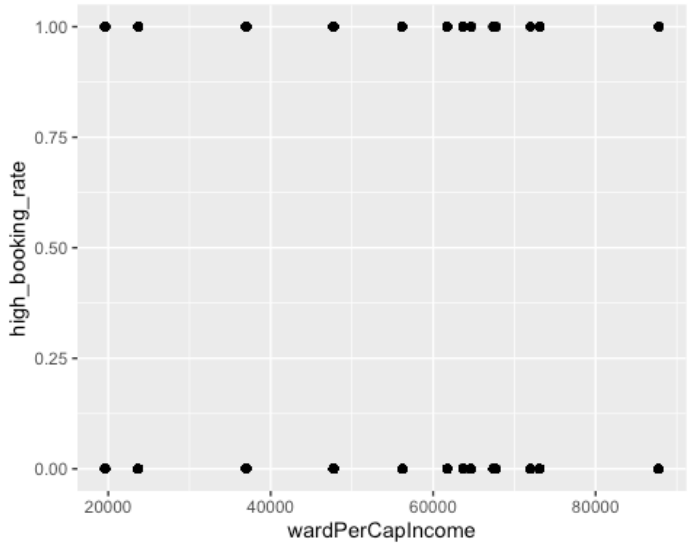



From the histogram plot, we can tell the total number of houses in the more dangerous areas is larger. The possible reason could be urban area tend to have bigger population and higher crime rate than rural places or the suburban. The means are very close to each other, but the quartile of houses with high booking rate is larger.

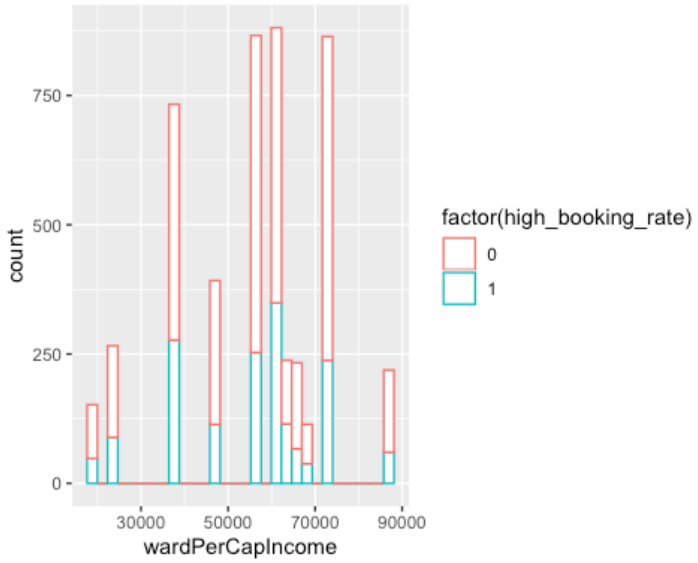
**wardPerCapIncome variable analysis**

For the variable wardPerCapIncome, the means are very close to each other, but the quartile of houses with high booking rate is larger. Its quartile boundaries are also lower than the ones of non-popular houses.

```
plot10 <- dcTrain %>%
  ggplot(aes(x =wardPerCapIncome, y = high_booking_rate)) + geom_point()
plot10
```



```
plot11<-dcTrain%>%ggplot(aes(x=wardPerCapIncome, color=factor(high_booking_rate))) +geom_histogram(fill="white")
plot11
```




```
plot12<-dcTrain%>%ggplot(aes(x=factor(high_booking_rate),y=wardPerCapIncome,fill=factor(high_booking_rate)))+geom_boxplot()
plot12
```


```
dcTrain$wardCrime[is.na(dcTrain$wardCrime)] <- median(dcTrain$wardCrime, na.rm=TRUE)
dcTrain$wardPerCapIncome[is.na(dcTrain$wardPerCapIncome)] <- median(dcTrain$wardPerCapIncome[], na.rm=TRUE)
dcTrain<-dcTrain[!is.na(dcTrain$review_scores_location), ]
```

### 4.1.3 Explanatory Techniques used for Question 1 - Community features

For question 1, we performed two models for explanatory purposes.

### Linear Model

```
fitlm <- lm(formula = high_booking_rate ~ review_scores_location+min_MetroEnt
ranceDist+wardCrime+ wardPerCapIncome+factor(wardName), data = dcTrain)

summary(fitlm)

##
## Call:
## lm(formula = high_booking_rate ~ review_scores_location + min_MetroEntranc
eDist +
##     wardCrime + wardPerCapIncome + factor(wardName), data = dcTrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.5913 -0.3906 -0.3458  0.5733  0.7367
##
## Coefficients: (2 not defined because of singularities)
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -1.470e+00  3.029e-01  -4.853 1.27e-06 ***
## review_scores_location  4.460e-02  1.296e-02   3.441 0.000586 ***
## min_MetroEntranceDist  -2.226e-02  2.741e-02  -0.812 0.416857
## wardCrime               2.592e-05  1.602e-05   1.618 0.105702
## wardPerCapIncome        2.078e-05  5.142e-06   4.042 5.40e-05 ***
## factor(wardName)2      -4.575e-01  8.115e-02  -5.637 1.85e-08 ***
## factor(wardName)3      -5.065e-01  2.074e-01  -2.443 0.014623 *
## factor(wardName)4       2.762e-01  6.713e-02   4.115 3.95e-05 ***
## factor(wardName)5       4.706e-01  1.175e-01   4.007 6.27e-05 ***
## factor(wardName)6      -3.911e-02  3.597e-02  -1.087 0.277023
## factor(wardName)7       7.907e-01  1.731e-01   4.568 5.08e-06 ***
## factor(wardName)8       9.025e-01  1.882e-01   4.796 1.68e-06 ***
## factor(wardName)12     -2.643e-01  5.122e-02  -5.160 2.60e-07 ***
## factor(wardName)13     -2.398e-01  1.359e-01  -1.765 0.077622 .
## factor(wardName)26     -2.111e-01  8.317e-02  -2.539 0.011165 *
## factor(wardName)34             NA         NA      NA       NA
## factor(wardName)126            NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4888 on 3955 degrees of freedom
##   (395 observations deleted due to missingness)
## Multiple R-squared:  0.01959,    Adjusted R-squared:  0.01612
## F-statistic: 5.646 on 14 and 3955 DF,  p-value: 5.654e-11

#plot(fitlm)
```

We first run a linear regression model with the five independent variable select. This will show us if there is any linear correlation and impact contributed from each variable selected to the high booking rate. The code for performing linear regression is displayed.

### Logistic Model

```
fitLog<-glm(data=dcTrain,family = 'binomial', high_booking_rate ~ review_scor
es_location+min_MetroEntranceDist+wardCrime+ wardPerCapIncome+factor(wardName
))
summary(fitLog)

##
## Call:
## glm(formula = high_booking_rate ~ review_scores_location + min_MetroEntran
ceDist +
##     wardCrime + wardPerCapIncome + factor(wardName), family = "binomial",
##     data = dcTrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3393  -0.9950  -0.9201   1.3077   1.6244
##
## Coefficients: (2 not defined because of singularities)
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)            -8.215e+00  1.286e+00  -6.388 1.68e-10 ***
## review_scores_location  1.987e-01  5.856e-02   3.393 0.000692 ***
## min_MetroEntranceDist  -9.317e-02  1.152e-01  -0.809 0.418458
## wardCrime               1.050e-04  6.602e-05   1.590 0.111802
## wardPerCapIncome        8.512e-05  2.131e-05   3.995 6.48e-05 ***
## factor(wardName)2      -1.868e+00  3.376e-01  -5.534 3.14e-08 ***
## factor(wardName)3      -2.081e+00  8.549e-01  -2.434 0.014914 *
## factor(wardName)4       1.131e+00  2.825e-01   4.003 6.26e-05 ***
## factor(wardName)5       1.941e+00  4.880e-01   3.978 6.94e-05 ***
## factor(wardName)6      -1.562e-01  1.487e-01  -1.050 0.293680
## factor(wardName)7       3.252e+00  7.208e-01   4.512 6.43e-06 ***
## factor(wardName)8       3.703e+00  7.861e-01   4.711 2.47e-06 ***
## factor(wardName)12     -1.089e+00  2.166e-01  -5.028 4.96e-07 ***
## factor(wardName)13     -9.824e-01  5.628e-01  -1.745 0.080900 .
## factor(wardName)26     -8.596e-01  3.436e-01  -2.502 0.012358 *
## factor(wardName)34             NA         NA      NA       NA
## factor(wardName)126            NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5388.6  on 3969  degrees of freedom
## Residual deviance: 5310.6  on 3955  degrees of freedom
##   (395 observations deleted due to missingness)
## AIC: 5340.6
##
## Number of Fisher Scoring iterations: 4
```

Then, we performed a logistic regression model on the same independent variables. This is also an explanatory model that could give us parameters on explaining the degree that each

variable is affecting the dependent variable, which is the high booking rate. The code for performing logistic regression is displayed below.

## 4.2 Question 2: Property characteristics (what kind of house should be to invest)

### 4.2.1 Data preparation

**input dataset and filter data to merely contain data that relate to D.C. market**

```
library(readr)
airbnbTrain1 <- read_csv("airbnbTrain.csv")

dsAirbnb<-airbnbTrain1 %>%
         filter(market=="D.C.")
```

**a. Using tally( ) function to check the categorical variable(property_type, room_type, bed_type) to see if each of them has sufficient data to be divided into the train dataset and test dataset.**

```
dsAirbnb %>%
   group_by( property_type) %>%
  tally() %>%
  arrange(n)

dsAirbnb %>%
   group_by( room_type) %>%
  tally() %>%
  arrange(n)

dsAirbnb %>%
   group_by( bed_type) %>%
  tally() %>%
  arrange(n)
```

**b. Recode property-type factor variable by combining its sub-variables which have insufficient data together; convert categorical variables into factor class.**

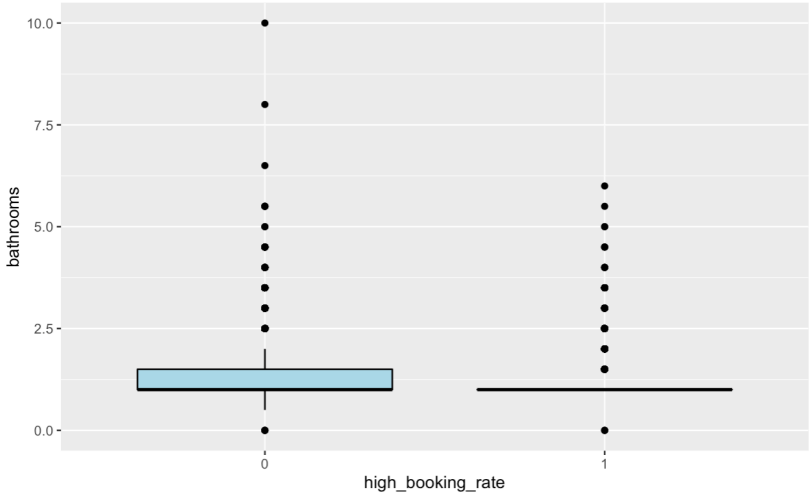
```
dsAirbnb <-
  dsAirbnb %>%
  mutate(property_type= ifelse(property_type %in% c('Barn', 'Boat', 'Boutique
 hotel', 'Tiny house','Resort','Cottage','Aparthotel'), 'Other', property_typ
e))

   dsAirbnb %>%
   group_by( property_type) %>%
  tally()

#convert char to factor
colsToFactor <- c('high_booking_rate', 'bed_type', 'room_type', "property_typ
e")
dsAirbnb <- dsAirbnb %>%
```

```
  mutate_at(colsToFactor, ~factor(.))

#price
#numeric price
dsAirbnb<-dsAirbnb %>%
mutate(price=as.numeric(gsub("[$,]","",dsAirbnb$price)))
```

### 4.2.2 Variable Analysis:

**1. the number of bathrooms**

```
#distribution of bathroom in D.C. market
bathroomsPlot <- ggplot( data=dsAirbnb ) +
  geom_histogram(aes(x = bathrooms, fill=high_booking_rate,color='black'))
ggplotly(bathroomsPlot )

#The relationship between high-booking-rate Airbnb and the number of bathroom
s.
plot1 <- ggplot(data = dsAirbnb, aes(x=high_booking_rate, y=bathrooms)) +
            geom_boxplot(fill="lightblue", color="black")
ggplotly(plot1)
```

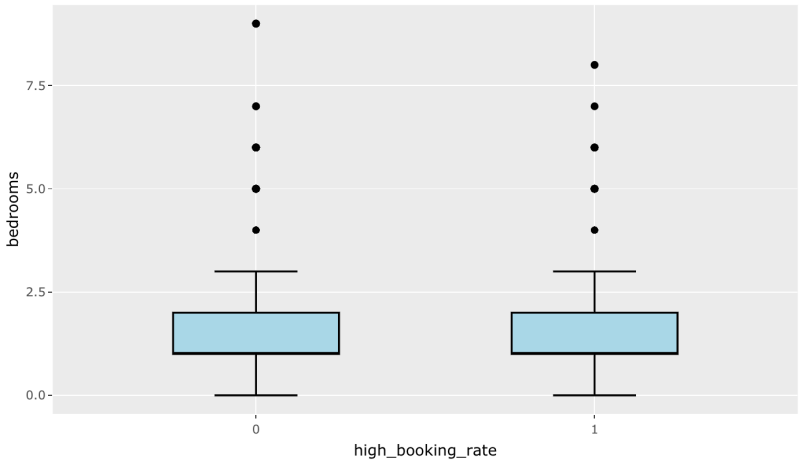



As the histogram showing, among 5452 Airbnbs in the D.C. market, most of them that around 2518 Airbnb provide only one bathroom, and 54% of these one-bathroom Airbnbs have high-booking rate. According to the boxplot, the variance of the number of bathrooms in non-high-booking-rate Airbnbs group is larger than it in high-booking-rate. It indicates that providing more bathrooms is not a direct factor to increase the booking rate for D.C. market's Airbnbs.

**2. the number of bedrooms**

```
#distribution of bathroom of high-booking-rate airbnb
bedroomsPlot <- ggplot(data=dsAirbnb) +
  geom_histogram(aes(x = bedrooms, fill=high_booking_rate),color='black')
ggplotly(bedroomsPlot)
```

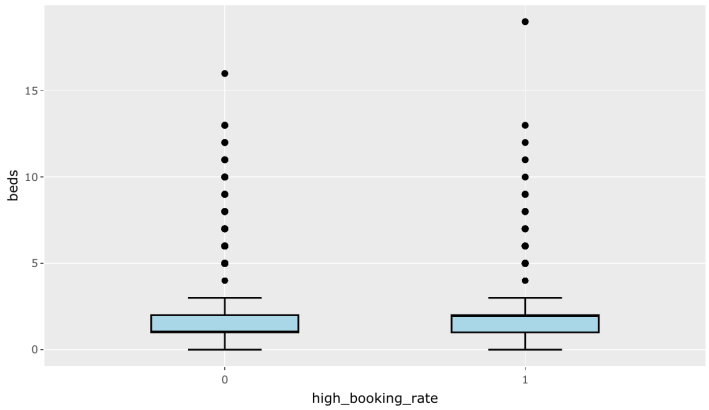
```
#The relationship between high-booking-rate Airbnb and the number of bathroom
s.
plot2 <- ggplot(data = dsAirbnb, aes(x=high_booking_rate, y=bedrooms)) +
            geom_boxplot(fill="lightblue", color="black")
ggplotly(plot2)
```




As the histogram showing, among 5452 Airbnbs in the D.C. market, most of them provide only one bedroom, and 52% of these one-bedroom Airbnbs have high-booking rate. According to the boxplot, the distribution of the number of bedrooms in high-booking-rate Airbnb group and non-high-booking-rate Airbnb group are almost identical excepting the max value. The high-booking-rate Airbnb group has the maximum value-9 bedrooms.

### 3. the number of beds

```
#distribution of beds of high-booking-rate airbnb

bedsPlot <- ggplot( data=dsAirbnb ) +
  geom_histogram(aes(x = beds, fill=high_booking_rate),color='black')
ggplotly(bedsPlot )

# The relationship between the growth rate of high-booking-rate and the numbe
r of beds
plot3 <- ggplot(data = dsAirbnb, aes(x=high_booking_rate, y=beds)) +
            geom_boxplot(fill="lightblue", color="black")
ggplotly(plot3)
```




Most of Airbnb around 1879 of them in the D.C. market provide one bed, following that there are 986 Airbnb offering two beds. Among one-bed providers, 45% of them have high-booking-rate. For the second large group- two-beds-providers, 55% of them are able to reach high booking rate. As the boxplot showing, if excluding the outliers, these two groups have almost same distributions.

### 4. bed type

```
dsbedtype<-dsAirbnb %>%
group_by(bed_type)%>%
tally()%>%
mutate(pct = 100*n/sum(n))
arrange(dsbedtype,desc(pct))

## # A tibble: 5 x 3
##   bed_type          n    pct
##   <fct>         <int>  <dbl>
## 1 Real Bed       5386 98.8
## 2 Pull-out Sofa    29  0.532
## 3 Futon            17  0.312
## 4 Airbed           11  0.202
## 5 Couch             9  0.165

dsbedtype<-dsAirbnb %>%
  filter(high_booking_rate==1) %>%
group_by(bed_type)%>%
tally()%>%
mutate(pct = 100*n/sum(n))
arrange(dsbedtype,desc(pct))

## # A tibble: 4 x 3
##   bed_type          n     pct
##   <fct>         <int>   <dbl>
## 1 Real Bed       1766 98.9
## 2 Pull-out Sofa    15  0.840
## 3 Futon             4  0.224
## 4 Couch             1  0.0560
```

Out of the four bed types of Real bed, pull out sofa, futon, air bed and couch. 98.78% of the beds in Airbnb listing are real beds. Among the high booking rate homes, 98.88% of them are having real beds. There are five types of bed provided by the D.C.s Airbnbs, but only four of them are offered by high-booking-rate Airbnbs. It shows Airbed isn`t a welcomed bed type.

### 5. room-type

```
dsroomtype<-dsAirbnb %>%
group_by(room_type)%>%
tally()%>%
mutate(pct = 100*n/sum(n))
arrange(dsroomtype,desc(pct))
```

```
## # A tibble: 4 x 3
##   room_type           n    pct
##   <fct>           <int>  <dbl>
## 1 Entire home/apt  3893 71.4
## 2 Private room     1407 25.8
## 3 Shared room       114  2.09
## 4 Hotel room         38  0.697
```

```
dsroomtype<-dsAirbnb %>%
  filter(high_booking_rate==1) %>%
group_by(room_type)%>%
tally()%>%
mutate(pct = 100*n/sum(n))
arrange(dsroomtype,desc(pct))
```

```
## # A tibble: 4 x 3
##   room_type           n    pct
##   <fct>           <int>  <dbl>
## 1 Entire home/apt  1324 74.1
## 2 Private room      419 23.5
## 3 Shared room        29  1.62
## 4 Hotel room         14  0.784
```

The most popular room type is the entire home or apartment, with a percentage up to 74.13%. Following next is the private room type which accounts for 23.46% of the airbnb homes. Shared rooms and hotel rooms are least popular ones with less than 2%.

### 6. property type

```
dsPro<-dsAirbnb %>%
group_by(property_type)%>%
tally()%>%
mutate(pct = 100*n/sum(n))
arrange(dsPro,desc(pct))
```

```
## # A tibble: 13 x 3
##    property_type          n    pct
##    <fct>              <int>  <dbl>
##  1 Apartment           2567 47.1
##  2 House               1125 20.6
##  3 Townhouse            831 15.2
##  4 Condominium          449  8.24
##  5 Guest suite          303  5.56
##  6 Loft                  34  0.624
##  7 Serviced apartment    31  0.569
##  8 Bed and breakfast     30  0.550
##  9 Guesthouse            29  0.532
## 10 Other                 28  0.514
## 11 Bungalow              10  0.183
## 12 Hostel                 8  0.147
## 13 Villa                  7  0.128
```

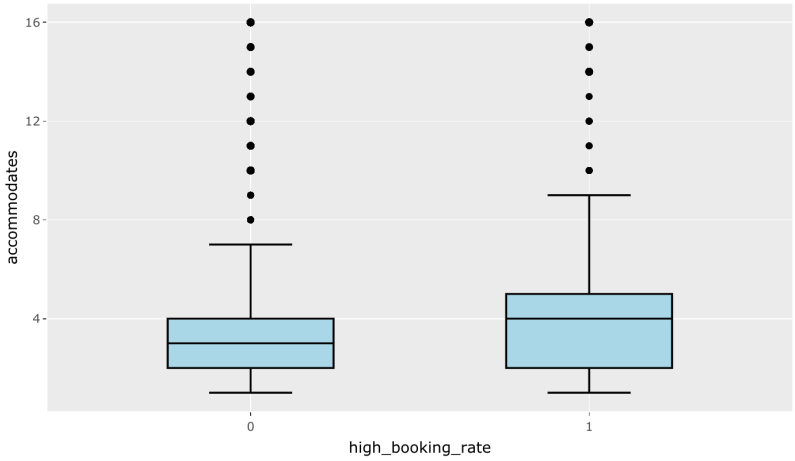
```
dsPro<-dsAirbnb %>%
  filter(high_booking_rate==1) %>%
group_by(property_type)%>%
tally()%>%
mutate(pct = 100*n/sum(n))

arrange(dsPro,desc(pct))
```

```
## # A tibble: 13 x 3
##    property_type          n    pct
##    <fct>              <int>  <dbl>
##  1 Apartment            800 44.8
##  2 Townhouse            372 20.8
##  3 House                301 16.9
##  4 Guest suite          179 10.0
##  5 Condominium           84  4.70
##  6 Loft                  11  0.616
##  7 Serviced apartment    11  0.616
##  8 Guesthouse            10  0.560
##  9 Bed and breakfast      6  0.336
## 10 Other                  5  0.280
## 11 Villa                  3  0.168
## 12 Bungalow               2  0.112
## 13 Hostel                 2  0.112
```

The top five popular property types of the Airbnb offerings are apartment, house, townhouse, condos and guest suite. What worth to notice is that among the homes with high booking rates, even though the number of houses is more than the number of townhouses, more townhouses property type have high booking rate. It means it is more popular. Same thing happens to guest suites and condos. More condos are served as Airbnb than guest suites, but guest suites have a larger percentage with high booking rate than condos. Which implies that with the apartment type is the most popular property type, and the followings are townhouses, houses, guest suites and condos. .

### 7. accommodates

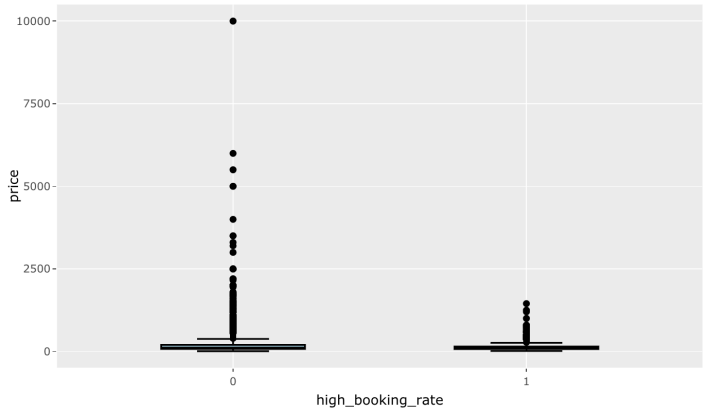
```
acHis <- ggplot(data=dsAirbnb) +
  geom_histogram(aes(x = accommodates, fill=high_booking_rate),color='black')
ggplotly(acHis)

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

plot4 <- ggplot(data = dsAirbnb, aes(x=high_booking_rate, y=accommodates)) +
            geom_boxplot(fill="lightblue", color="black")
ggplotly(plot4)
```




According to the histogram, most Airbnb in the D.C. area can accommodate two guests. Following by that, the second large group can accommodate four guests. The boxplot shows that 75th percentile of high-booking-rate Airbnb provide five guests accommodation. Comparing the non-high-booking-rate Airbnb group with the high-booking-rate one, the high-booking-rate Airbnb group offer a larger accommodation range.

### 8.Price group

```
plot5 <- ggplot(data = dsAirbnb, aes(x=high_booking_rate, y=price)) +
            geom_boxplot(fill="lightblue", color="black")
ggplotly(plot5)
```




According to the boxplot, the price of those high-booking-rate Airbnb centralize in 265 to 1450 price interval. If price is higher than $1450, it has high probability to be avoided by Airbnb guests.

### 4.2.3 Explanatory Techniques used for Question 2 and Underlying Reasoning-Property features

In question 2 on analyzing property features, we have tried logistic regression, LDA regression for the same purpose as question 1 to find out the level of influence of each variable to a high booking rate. However, the result accuracy after exam model performance is not quite satisfied. Then, we tried lasso regression, ridge regression and elastic net to figure out the variable importance, which is by ranking the significance of each independent variable in regard to high booking rate. The variable with highest importance will affect the result of high booking rate the most, and vice versa. Furthermore, when analyzing model performance, we would like to achieve the lowest false positive rate. The business ideology here is that false positives rate appears when the prospective home would not achieve a high booking rate but the model mistakenly predicts it to be a high booking rate home. This is the worst-case scenario for business investors, and could potentially result in tremendous losses for their investment. Therefore, when evaluating model performance. Our goal is to pick the model with the lowest false positive rate.

```
#create a small dataset; clean missing value
dsfit<-  subset(dsAirbnb,select=c("high_booking_rate","bathrooms", 'bedrooms'
,'beds','bed_type','room_type','property_type','accommodates','price'))
dsfit<-na.omit(dsfit)

# Set the seed to 52156, randomly split the dataset into a training dataset a
nd a test dataset. Use 65% of the data for training and hold out 35%.
set.seed(52156)
dsfitTr<- dsfit%>% sample_frac(0.65)
dsfitTe <- setdiff(dsfit, dsfitTr)
```

**Logistic regression**

```
#Run logistic regression model to predict a high-booking-rate Airbnb in D.C.
market:
```

```r
Logfit  <- glm(high_booking_rate ~ ., family='binomial', data=dsfitTr)
summary(Logfit )
```

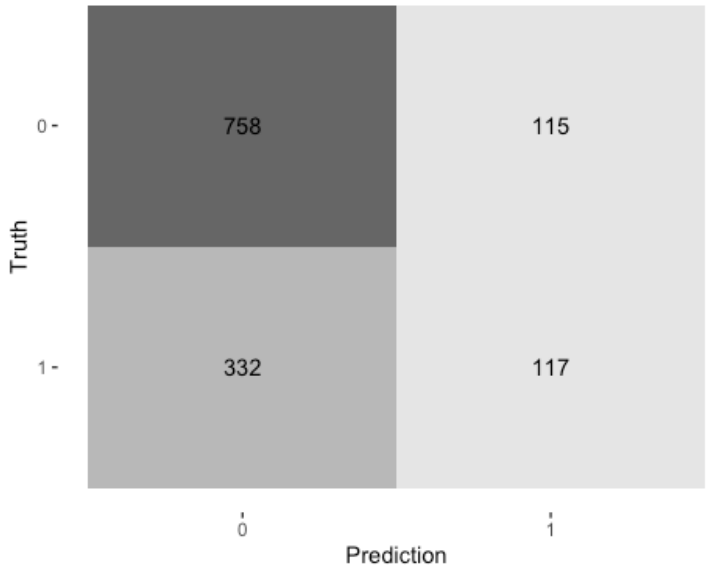
```
##
## Call:
## glm(formula = high_booking_rate ~ ., family = "binomial", data = dsfitTr)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2519  -0.8912  -0.6904   1.2093   3.1524
##
## Coefficients:
##                                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   -1.298e+01  1.778e+02  -0.073  0.94180
## bathrooms                     -4.350e-01  9.649e-02  -4.508 6.55e-06 ***
## bedrooms                      -2.320e-01  7.675e-02  -3.023  0.00250 **
## beds                          -2.315e-02  5.360e-02  -0.432  0.66576
## bed_typeCouch                  1.191e+01  1.778e+02   0.067  0.94657
## bed_typeFuton                  1.100e+01  1.778e+02   0.062  0.95064
## bed_typePull-out Sofa          1.395e+01  1.778e+02   0.078  0.93746
## bed_typeReal Bed               1.260e+01  1.778e+02   0.071  0.94349
## room_typeHotel room            4.194e-02  7.234e-01   0.058  0.95377
## room_typePrivate room         -1.904e-01  1.162e-01  -1.639  0.10124
## room_typeShared room          -2.849e-01  2.869e-01  -0.993  0.32078
## property_typeBed and breakfast -6.435e-01  7.312e-01  -0.880  0.37878
## property_typeBungalow         -7.959e-01  1.137e+00  -0.700  0.48390
## property_typeCondominium      -4.644e-01  1.560e-01  -2.976  0.00292 **
## property_typeGuest suite       9.061e-01  1.655e-01   5.475 4.37e-08 ***
## property_typeGuesthouse       -3.208e-01  5.784e-01  -0.555  0.57919
## property_typeHostel           -2.443e+00  1.302e+00  -1.877  0.06055 .
## property_typeHouse             1.665e-01  1.226e-01   1.358  0.17444
## property_typeLoft              2.991e-01  4.409e-01   0.678  0.49763
## property_typeOther            -8.824e-01  7.813e-01  -1.129  0.25876
## property_typeServiced apartment  1.084e-01  6.180e-01   0.175  0.86077
## property_typeTownhouse         7.455e-01  1.211e-01   6.157 7.40e-10 ***
## property_typeVilla             2.918e-01  9.048e-01   0.323  0.74707
## accommodates                   2.959e-01  3.627e-02   8.158 3.41e-16 ***
## price                         -3.932e-03  4.563e-04  -8.618  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4469.1  on 3531  degrees of freedom
## Residual deviance: 4066.3  on 3507  degrees of freedom
## AIC: 4116.3
##
## Number of Fisher Scoring iterations: 12
```

```r
car::vif(Logfit )
```

```
## Registered S3 methods overwritten by 'car':
##   method                          from
##   influence.merMod                lme4
##   cooks.distance.influence.merMod lme4
##   dfbeta.influence.merMod         lme4
##   dfbetas.influence.merMod        lme4

##                   GVIF Df GVIF^(1/(2*Df))
## bathrooms     2.292091  1        1.513965
## bedrooms      3.559056  1        1.886546
## beds          3.710819  1        1.926349
## bed_type      1.018041  4        1.002238
## room_type     4.512877  3        1.285510
## property_type 4.075828 12        1.060292
## accommodates  4.899644  1        2.213514
## price         1.549860  1        1.244934

# vif shows room_type, property_type and accommodates have multicolliearity
# find way to dismiss it

# Using cut-off 0.5 and do classification in the test data. Compute and repor
t the confusion matrix for the test data prediction

resultsLog <-glm(high_booking_rate ~ ., family='binomial', data=dsfitTr) %>%
  predict(dsfitTe, type='response') %>%
  bind_cols(dsfitTe, predictedProb=.) %>%
  mutate(predictedClass = as.factor(ifelse(predictedProb > 0.5, 1, 0)))
resultsLog

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

## Confusion Matrix and Statistics
##
##               high_booking_rate
## predictedClass   0   1
##              0 758 332
##              1 115 117
##
##                Accuracy : 0.6619
##                  95% CI : (0.6357, 0.6874)
##     No Information Rate : 0.6604
##     P-Value [Acc > NIR] : 0.4665
##
##                   Kappa : 0.146
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.2606
##             Specificity : 0.8683
```

```
##          Pos Pred Value : 0.5043
##          Neg Pred Value : 0.6954
##              Prevalence : 0.3396
##          Detection Rate : 0.0885
##    Detection Prevalence : 0.1755
##       Balanced Accuracy : 0.5644
##
##        'Positive' Class : 1
##
```

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




### 10-fold cross validation LDA regression

```
# Try other models
#1. Using 10-fold cross validation resampling method to run a LDA regression
set.seed(123)

fitLDA <- train(high_booking_rate ~ ., data=dsfitTr, method='lda', trControl=
trainControl(method='cv', number=10))

resultsLDA <-
  fitLDA %>%
  predict(dsfitTe, type='raw') %>%
  bind_cols(dsfitTe, predictedClass=.)

resultsLDA %>%
  mutate(isCorrect = ifelse(predictedClass == high_booking_rate, 1, 0)) %>%
  xtabs(~predictedClass+high_booking_rate, .) %>%
  confusionMatrix(positive = '1')
```

```
## Confusion Matrix and Statistics
##
##                high_booking_rate
## predictedClass   0   1
##              0 772 343
##              1 101 106
##
##                Accuracy : 0.6641
##                  95% CI : (0.638, 0.6896)
##     No Information Rate : 0.6604
##     P-Value [Acc > NIR] : 0.398
##
##                   Kappa : 0.1385
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.23608
##             Specificity : 0.88431
##          Pos Pred Value : 0.51208
##          Neg Pred Value : 0.69238
##              Prevalence : 0.33964
##          Detection Rate : 0.08018
##    Detection Prevalence : 0.15658
##       Balanced Accuracy : 0.56019
##
##        'Positive' Class : 1
##
```

```
#1-specificity
#Logistics:
1-0.89158
```

```
## [1] 0.10842
```

```
#1-specificity
#LDA:
1- 0.90081
```

```
## [1] 0.09919
```

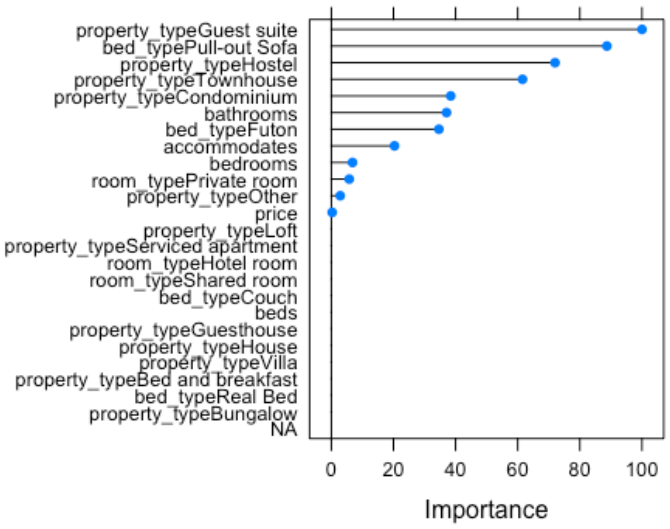
Comparing the performance of the Logistics regression by being evaluated through validation set resampling apporach with the performance of LDA model by being evaluated through 10-fold cross validation resampling approach (Accuracy : 0.6765 VS. Accuracy : 0.6811 ), it shows the LDA regression has a higher accurancy.In this business case, false positive rate(1-specificity) is a more important test metric, it shoule be as low as possible so that the probability of cost incurred by mistaken prediction can be reduced as much as possible. Between these two regression models, LDA has a lower false positive rate, so LDA is a better fitting model rather than the logistics regression.

**Lasso Model**

```r
#Lasso

lambdaValues <- 10^seq(-5, 2, length = 100)

set.seed(123)

fitLasso  <- train(high_booking_rate ~ ., family='binomial', data=dsfitTr, me
thod='glmnet', trControl=trainControl(method='cv', number=10), tuneGrid = exp
and.grid(alpha=1, lambda=0.01))
fitLasso

varImp(fitLasso)$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: 24 x 3
##    Variable                  Overall Importance
##    <chr>                       <dbl> <chr>
##  1 property_typeGuest suite  100    100%
##  2 bed_typePull-out Sofa      88.7  89%
##  3 property_typeHostel        72.0  72%
##  4 property_typeTownhouse     61.5  62%
##  5 property_typeCondominium   38.4  38%
##  6 bathrooms                  37.1  37%
##  7 bed_typeFuton              34.6  35%
##  8 accommodates               20.3  20%
##  9 bedrooms                    6.83 7%
## 10 room_typePrivate room       5.77 6%
## # … with 14 more rows
```

```r
#Variable importance plot with the 25 most important variables
plot(varImp(fitLasso), top = 25)
```




```
#Optimum lambda selected by the algorithm
fitLasso$bestTune$lambda   # You can also run fitLasso$finalModel$lambdaOpt
```

```
## [1] 0.01
```

```
#Not so useful but helps with understanding -See how different lambda values
drop variables
#plot(fitLasso$finalModel, xvar="lambda", label = TRUE)

#Not so useful but helps with understanding -See the coefficients from the fi
nal lasso model
coef(fitLasso$finalModel, fitLasso$bestTune$lambda)   # You can also use fitL
asso$finalModel$lambdaOpt
```

```
## 25 x 1 sparse Matrix of class "dgCMatrix"
##                                          1
## (Intercept)                   -0.566719548
## bathrooms                     -0.307320026
## bedrooms                      -0.056568822
## beds                           .
## bed_typeCouch                  .
…
## price                         -0.002332013
```

```
resultsLasso <-
  fitLasso %>%
  predict(dsfitTe, type='raw') %>%
  bind_cols(dsfitTe, predictedClass=.)

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

```
## Confusion Matrix and Statistics
##
##                high_booking_rate
## predictedClass   0   1
##              0 807 373
##              1  66  76
##
##                Accuracy : 0.6679
##                  95% CI : (0.6418, 0.6933)
##     No Information Rate : 0.6604
##     P-Value [Acc > NIR] : 0.2913
##
##                   Kappa : 0.1123
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.16927
##             Specificity : 0.92440
##          Pos Pred Value : 0.53521
##          Neg Pred Value : 0.68390
##              Prevalence : 0.33964
##          Detection Rate : 0.05749
##    Detection Prevalence : 0.10741
##       Balanced Accuracy : 0.54683
##
##        'Positive' Class : 1
##
```

```
#Accuracy : 0.6765 VS.
#1-specificity
#Logistics:
1-0.89158
```

```
## [1] 0.10842
```

```
#1-specificity
#LDA:
1- 0.90081
```

```
## [1] 0.09919
```

```
#Accuracy : 0.6811

#1-specificity
#Lasso
1-0.93541
```

```
## [1] 0.06459
```

```
# Accuracy : 0.6749
```

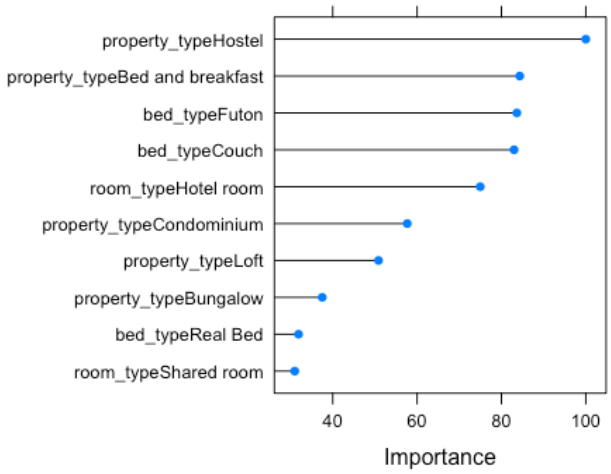
In contrast with the Logistcs model and the LDA model, Lasso model has the lowest accuracy (0.6749) and lowest false positive rate(1-specificity), so lasso is better than LDA.

**Ridge Model**

```
#Ridge

lambdaValues <- 10^seq(-5, 2, length = 100)

set.seed(123)

fitRidge <- train(high_booking_rate ~ ., family='binomial', data=dsfitTe, met
hod='glmnet', trControl=trainControl(method='cv', number=10), tuneGrid = expa
nd.grid(alpha=0, lambda=lambdaValues))

resultsRidge <-
  fitRidge %>%
  predict(dsfitTe, type='raw') %>%
  bind_cols(dsfitTe, predictedClass=.)
resultsRidge

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

## Confusion Matrix and Statistics
##
##               high_booking_rate
## predictedClass   0   1
##              0 813 372
##              1  60  77
##
##                Accuracy : 0.6732
##                  95% CI : (0.6472, 0.6985)
##     No Information Rate : 0.6604
##     P-Value [Acc > NIR] : 0.169
##
##                   Kappa : 0.1236
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.17149
##             Specificity : 0.93127
##          Pos Pred Value : 0.56204
##          Neg Pred Value : 0.68608
##              Prevalence : 0.33964
##          Detection Rate : 0.05825
##    Detection Prevalence : 0.10363
##       Balanced Accuracy : 0.55138
##
```

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

```
#Variable importance plot with the 25 most important variables
plot(varImp(fitRidge), top = 10)
```




```
#Optimum lambda selected by the algorithm
fitRidge$bestTune$lambda
```

```
## [1] 0.02477076
```

```
#Not so useful but helps with understanding -See the coefficients from the fi
nal lasso model
coef(fitRidge$finalModel, fitRidge$bestTune$lambda)
```

```
## 25 x 1 sparse Matrix of class "dgCMatrix"
##                                       1
## (Intercept)                -0.916234107
## bathrooms                  -0.266029975
## bedrooms                   -0.214171202
…
## property_typeVilla                    .
## accommodates                0.136921453
## price                      -0.001380432
```

```
#Logistics:
#Accuracy : 0.6765
#1-specificity
  0.10842
```

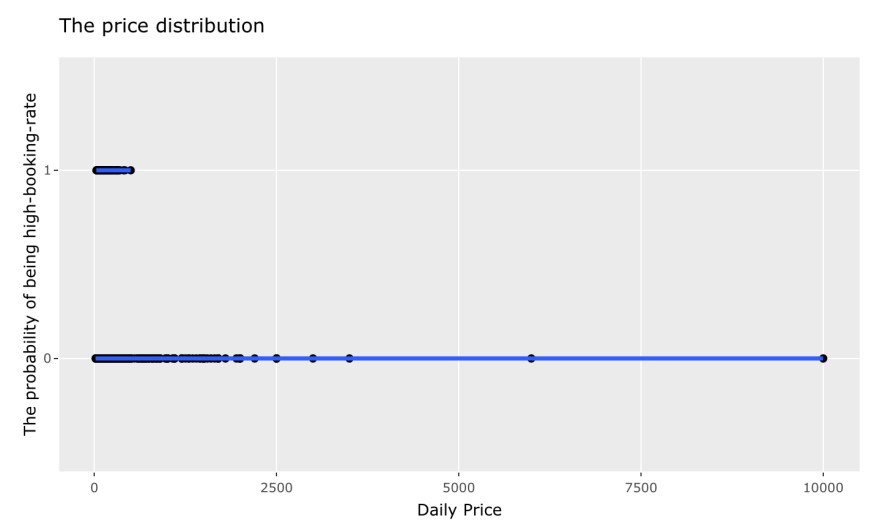
```
## [1] 0.10842
```

```
#LDA:
#Accuracy : 0.6811
```

```
#1-specificity
1- 0.90081

## [1] 0.09919

#Lasso
# Accuracy : 0.6749
#1-specificity
1-0.93541

## [1] 0.06459

#Ridge
# Accuracy :0.6865
#1-specificity
1-0.96770

## [1] 0.0323
```

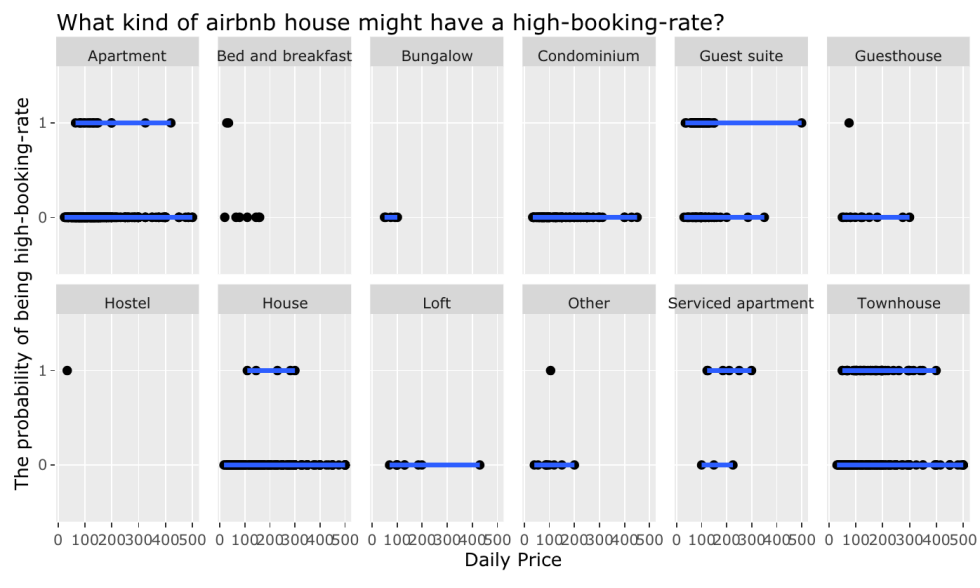
Comparing with the Lasso regression, the Rigde shows higher accuracy which means the IVs are largely relevant to the DV. What`s more, the Ridge has the lowest false positive rate (1-specificity) test metric, so the Ridge is the best model among all of models so far.

```
#visulization

plotbedtype1<- ggplot(aes(y=predictedClass, x=price), data=resultsRidge) + ge
om_point() + geom_smooth() + labs(title="The price distribution", y = "The prob
ability of being high-booking-rate", x ="Daily Price")
ggplotly(plotbedtype1)
```

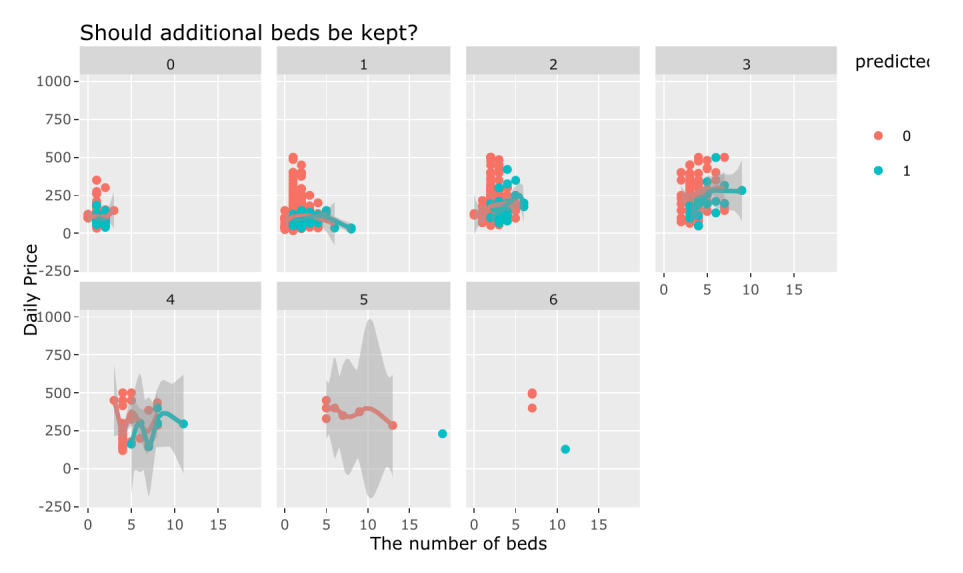



```
VfitRidge<-
  resultsRidge %>%
  filter(price<=500)

plotbedrooms2<- ggplot(aes(y=predictedClass , x=price ), data=VfitRidge) + ge
```

```
om_point() + geom_smooth() + facet_wrap(~ property_type, nrow = 2)+ labs(title=
"What kind of airbnb house might have a high-booking-rate?", y = "The probabi
lity of being high-booking-rate", x ="Daily Price")
ggplotly(plotbedrooms2)
```

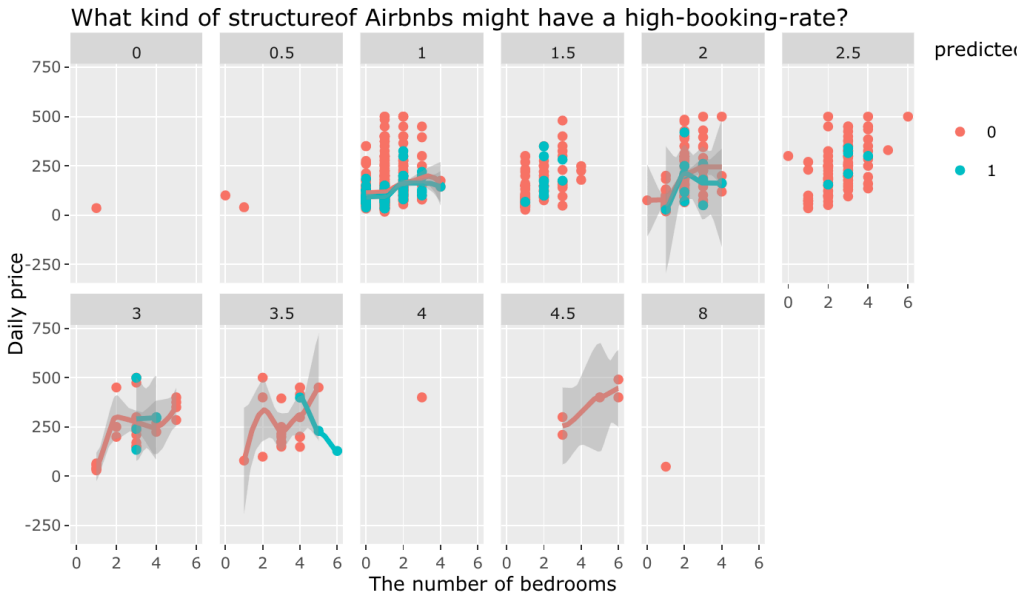



```
plotbedtype3<- ggplot(aes(y=price, x=beds,color=predictedClass),data=VfitRidg
e) + geom_point() + geom_smooth() + facet_wrap(~ bedrooms, nrow = 2)+ labs(titl
e="Should additional beds be kept?", y = "Daily Price", x ="The number of bed
s")
ggplotly(plotbedtype3)
```




```
plotbedrooms4<- ggplot(aes(y=price , x=bedrooms ,color=predictedClass), data=
VfitRidge) + geom_point() + geom_smooth() + facet_wrap(~ bathrooms , nrow = 2)+
 labs(title="What kind of structureof Airbnbs might have a high-booking-rate?
```

```
", y = "Daily price", x ="The number of bedrooms")
ggplotly(plotbedrooms4)
```




We did a visualization to illuminate the relationship among different variables.
As the feature1: The price distribution shows, the daily price of Airbnb should be considered under $500. According to the Plotbedroom2, for apartment and house type Airbnb, the higher price, the lower probability to be a high-booking-rate Airbnb; for Town house Airbnb, with price increasing, the probability of being high-booking-rate and non-high-booking rate is basically same.

According to the plotbedtype3, it is a good consideration to keep extra beds for two- and three-bedrooms Airbnb with relatively low-price interval. It might be helpful for increasing the booking rate.

The plotbedroom4 indicates that in the D.C. Airbnb market, most Airbnbs are 1 bathroom and 2 bedrooms structures. For one-bathroom Airbnbs, under the same level price, the more bedrooms, the higher probability to be a high-booking-rate Airbnb. For the Airbnbs containing one and half or two bathrooms, even with a higher price, the more bedrooms, the probability to be a high-booking-rate Airbnb is high.

### Elastic net Model

```
#ElasticNet
set.seed(123)


fitElasticNet <- train(high_booking_rate ~ ., family='binomial', data=dsfitTe
, method='glmnet', trControl=trainControl(method='cv', number=10), tuneLength
=10)
```

```r
resultsElasticNet <-
  fitElasticNet %>%
  predict(dsfitTe, type='raw') %>%
  bind_cols(dsfitTe, predictedClass=.)

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

```
## Confusion Matrix and Statistics
##
##                high_booking_rate
## predictedClass   0   1
##              0 785 335
##              1  88 114
##
##                Accuracy : 0.68
##                  95% CI : (0.6541, 0.7051)
##     No Information Rate : 0.6604
##     P-Value [Acc > NIR] : 0.0688
##
##                   Kappa : 0.1767
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.25390
##             Specificity : 0.89920
##          Pos Pred Value : 0.56436
##          Neg Pred Value : 0.70089
##              Prevalence : 0.33964
##          Detection Rate : 0.08623
##    Detection Prevalence : 0.15280
##       Balanced Accuracy : 0.57655
##
##        'Positive' Class : 1
##
```

```r
#Logistics:
#Accuracy : 0.6765
#1-specificity
  0.10842
```

```
## [1] 0.10842
```

```r
#LDA:
#Accuracy : 0.6811
#1-specificity
0.09919
```

```
## [1] 0.09919
```

```
#Lasso
# Accuracy : 0.6749
#1-specificity
0.06459
```

```
## [1] 0.06459
```

```
#Ridge
# Accuracy :0.6865
#1-specificity
0.0323
```

```
## [1] 0.0323
```

```
#ElasticNet
# Accuracy :0.6858
#1-specificity
0.05075
```

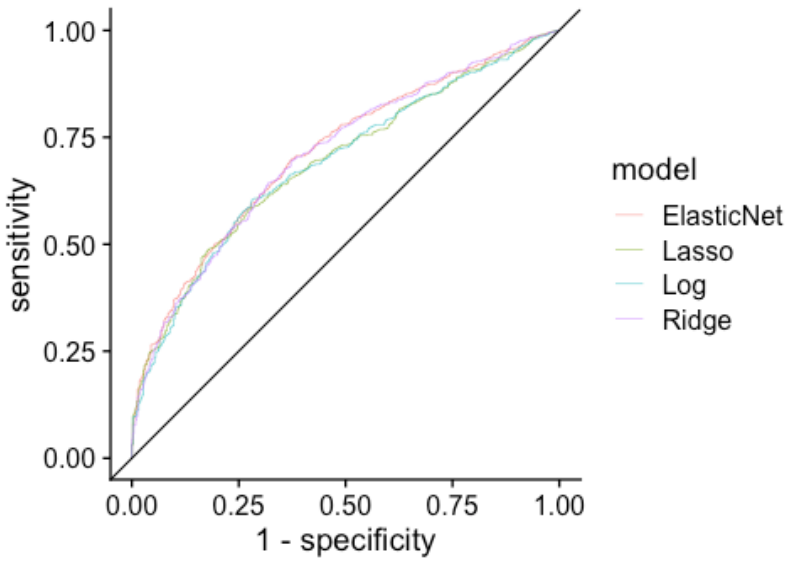
```
## [1] 0.05075
```

ElasticNet has the higher the false positive rate (1-specificity) test metric than lasso. Thus, Lasso is the best model for this business case.

**ROC evalutation**

```
#ROC

Log_to_roc <-
  train(high_booking_rate ~ ., family='binomial', data=dsfitTr, method = 'glm
') %>%
  predict(dsfitTe, type="prob") %>%
  cbind(dsfitTe,predictedProb=.) %>%
  mutate(model='Log')

LDA_to_roc <-
  train(high_booking_rate ~ ., data=dsfitTr, method='lda', trControl=trainCon
trol(method='cv', number=10)) %>%
  predict(dsfitTe, type="prob") %>%
  cbind(dsfitTe, predictedProb=.) %>%
  mutate(model='LDA')

Lasso_to_roc <-
  fitLasso %>%
  predict(dsfitTe, type="prob") %>%
  cbind(dsfitTe, predictedProb=.) %>%
  mutate(model='Lasso')

Ridge_to_roc <-
  fitRidge %>%
  predict(dsfitTe, type="prob") %>%
```

```
  cbind(dsfitTe, predictedProb=.) %>%
  mutate(model='Ridge')

ElasticNet_to_roc <-
  fitElasticNet %>%
  predict(dsfitTe, type="prob") %>%
  cbind(dsfitTe, predictedProb=.) %>%
  mutate(model='ElasticNet')

glmOutAll <- bind_rows(Lasso_to_roc, Ridge_to_roc, ElasticNet_to_roc, Log_to_
roc,ElasticNet_to_roc)

glmOutAll %>%
  group_by(model) %>% # group to get individual ROC curve for each model
  roc_curve(truth = high_booking_rate,predictedProb.1) %>% # get values to pl
ot an ROC curve
  ggplot(aes(x = 1 - specificity, y = sensitivity, color = model)) + # plota
ROC curve for each model
  geom_line(size = 0.15) +
  geom_abline(slope = 1, intercept = 0, size = 0.4) +
  coord_fixed() +
  theme_cowplot()
```




According to the ROC Curve, in the (1-specificity) 0 to 0.25 interval, Ridge has higher sensitivity, so Ridge is the best model for this business case.

## 4.3 Question 3: Predict whether a property in DC would have a high booking rate

### 4.3.1 Data Preparation and variable selection

As the "summary" model, let's first take a look at the variable selection. For variables we use, we did mainly 3 kinds of processing: Recategorizing, transforming and filling. We marked variables like "access, host_about" as 1 if there are word descriptions in these columns otherwise 0, because we think providing more infos in these areas would help customer learn more about the property. We also combined some categories for some columns such as "cancelation policy" "property type", because some types only have less than 10 observations that might create bias for judgement, and might cause rank efficiency issues since there are too many categories.We transformed the text column "amenities" to several dummy columns to detect if there are some essential and commonly used amenities. We also create "bedperCapita" and "host_years" based on calculation.For missing values, we mostly fill them with medians or the most common category.For Variables we deleted, they most fall in the following categories: Irrelevant By Nature, Inaccessible, Not Distinguishable, Location that already transformed, Many NAs. At last, we filter out pricing outliers and have 4,869 rows of data and 44 ready-for-use variables.

1. These variables are not relevant to our model by nature: "market", "randomControl", "host_location", "host_neighbourhood", "is_location_exact","city"

2. The following columns are not available until someone lives in: "host_acceptance_rate","review_scores_checkin","review_scores_cleanliness","review_scores_communication","review_scores_location","review_scores_rating", "review_scores_value","review_scores_accuracy";

3. By scanning the value of the following columns, they either have same value or does not help much in distinguishing one property from another: "description","is_business_travel_ready","requires_license","maximum_nights","neighborhood_overview","host_verifications" The reason we delete "desription" here is that we did not do text mining for this column and almost all the properties have description here.

4. Location infos are already transformed to variables such as "min_MetroEntranceDist", thus these colunms will be used no more: "zipcode","latitude","longitude","state","neighbourhood"

```
# Create a list of the above irrelevant columns and delete them from the data frame
irr  <- c("market", "randomControl", "host_location", "host_neighbourhood", "host_verifications","is_location_exact","host_acceptance_rate","review_scores_accuracy","review_scores_checkin","review_scores_cleanliness","review_scores_communication","review_scores_location","review_scores_rating","review_scores_value","description","is_business_travel_ready","requires_license","maximum_nights", "neighbourhood","city","neighborhood_overview","zipcode","latitude","longitude","state")
```

```r
df_dc <- df_dc[ , !(names(df_dc) %in% irr)]

# Change word description columns to categorical var, more information may ma
ke the property a popular spot then a high rated property.
df_dc$access <- ifelse(is.na(df_dc$access), 0, 1 )
df_dc$host_about <- ifelse(is.na(df_dc$host_about), 0, 1 )
df_dc$house_rules <- ifelse(is.na(df_dc$house_rules), 0, 1 )
df_dc$notes <- ifelse(is.na(df_dc$notes), 0, 1 )
df_dc$interaction <- ifelse(is.na(df_dc$interaction), 0, 1 )
df_dc$space <- ifelse(is.na(df_dc$space), 0, 1 )
df_dc$transit <- ifelse(is.na(df_dc$transit), 0, 1 )

#Transform host_since to a variable that indicates how long in years that the
 host has experience for
df_dc$host_years <- 2020-as.numeric(substr(df_dc$host_since, start = 1, stop
= 4))

# Convert currency(previous type: $00.00 in character) to number.
df_dc$cleaning_fee <- as.numeric(gsub("\\$", "", df_dc$cleaning_fee))
df_dc$price <- as.numeric(gsub("\\$", "", df_dc$price))

## Warning: NAs introduced by coercion

df_dc$extra_people <- as.numeric(gsub("\\$", "", df_dc$extra_people))
df_dc$security_deposit <- as.numeric(gsub("\\$", "", df_dc$security_deposit))

## Warning: NAs introduced by coercion

# Recategorize some varibles to fewer meaningful categories

#cancellation_policy
df_dc$cancellation_policy[df_dc$cancellation_policy %in% c("super_strict_60"
,
  "strict_14_with_grace_period","super_strict_30")] <- "strict"

#accommodates
df_dc <- df_dc %>%
  mutate(accommodateCat = ifelse(accommodates <= 2, "1-2",ifelse(accommodates
 >3 & accommodates <= 4, "3-4", ifelse(accommodates >4 & accommodates <= 10,
"5-10",">10"))))
#host_listings_count
df_dc <- df_dc %>%
  replace_na(list(host_listings_count = mode(df_dc$host_listings_count))) %>%
  mutate(host_listings_count = ifelse(host_listings_count <= 2, "0-2", ifelse
(host_listings_count >3 & host_listings_count <= 15, "3-15",ifelse(host_listi
ngs_count >15 & host_listings_count <= 100, "15-100",">100"))))
# property_type
df_dc$property_type[df_dc$property_type %in% c("Barn" , "Boat", "Tiny house",
```

```
 "Cottage","Aparthotel","Boutique hotel","Hostel","Villa", "Bungalow")] <- "O
ther"

#Process columns to get new meaningful columns

#bedperCapita
df_dc <- df_dc %>%
  mutate(bedperCapita = beds / accommodates)

# wifi, TV, kitchen and washer
df_dc <- df_dc %>%
  mutate(amenity_wifi = str_detect(amenities,"Wifi")) %>%
  mutate(amenity_TV = str_detect(amenities,"TV")) %>%
  mutate(amenity_Washer = str_detect(amenities,"Washer")) %>%
  mutate(amenity_Kitchen = str_detect(amenities,"Kitchen"))

# Change amenities to the number of amenity it has
df_dc$amenities <- str_count(df_dc$amenities, ',')+1

df_dc$host_response_rate <- as.numeric(gsub("%", "", df_dc$host_response_rate
))

## Warning: NAs introduced by coercion

#Delete transformed columns
trans <- c("host_since","beds","accommodates")
df_dc <- df_dc[ , !(names(df_dc) %in% trans)]

#Remove pricing outliers.
nrow(df_dc)

## [1] 5492

Q <- quantile(df_dc$price, probs=c(.25, .75), na.rm = TRUE)
iqr <- IQR(df_dc$price, na.rm = TRUE)
up <-  Q[2]+1.5*iqr # Upper Range
low<- Q[1]-1.5*iqr # Lower Range
df_dc<- subset(df_dc, df_dc$price > low & df_dc$price < up)
nrow(df_dc)

## [1] 4869

#Scan for NA
skim(df_dc)

#see a special column, host_response_time
#Since this column has 1369+4 missing values, we think it cannot be simply re
placed by the most common category, thus just delete that
df_dc %>%
  group_by(host_response_time) %>%
```

```
  tally() %>%
  arrange(n)

#delete columns that have too many NAs
manyNA <- c("monthly_price","weekly_price","square_feet","host_response_time"
)
df_dc <- df_dc[ , !(names(df_dc) %in% manyNA)]

# Replace NA with the most frequent category or medians. For ward that are mi
ssing, fill them with 0.
df_dc <- df_dc %>%
  replace_na(list(host_has_profile_pic = TRUE, na.rm = TRUE)) %>%
  replace_na(list(host_identity_verified = TRUE, na.rm = TRUE)) %>%
  replace_na(list(host_is_superhost = FALSE, na.rm = TRUE)) %>%
  replace_na(list(wardName = 0, na.rm = TRUE)) %>%
  replace_na(list(bathrooms = median(df_dc$bathrooms, na.rm = TRUE)))%>%
  replace_na(list(bedrooms = median(df_dc$bedrooms, na.rm = TRUE)))%>%
  replace_na(list(cleaning_fee = 0, na.rm = TRUE)) %>%
  replace_na(list(price = median(df_dc$price, na.rm = TRUE))) %>%
  replace_na(list(security_deposit = median(df_dc$security_deposit, na.rm = T
RUE))) %>%
  replace_na(list(host_years = median(df_dc$host_years, na.rm = TRUE)))%>%
  replace_na(list(wardCrime = median(df_dc$wardCrime, na.rm = TRUE)))%>%
  replace_na(list(wardPerCapIncome = median(df_dc$wardPerCapIncome, na.rm = T
RUE)))%>%
  replace_na(list(bedperCapita = median(df_dc$bedperCapita, na.rm = TRUE))) %
>%
  replace_na(list(host_response_rate = median(df_dc$host_response_rate, na.rm
 = TRUE)))

# Make sure categorical variables are factors
colsToFactor <- c("high_booking_rate","bed_type","cancellation_policy","host_
listings_count","property_type","room_type","accommodateCat","host_has_profil
e_pic","host_identity_verified","host_is_superhost","instant_bookable","requi
re_guest_phone_verification","require_guest_profile_picture","amenity_wifi","
amenity_TV","amenity_Washer","amenity_Kitchen","wardName")
df_dc <- df_dc %>%
  mutate_at(colsToFactor, ~factor(.))
```

### 4.3.2 Explanatory and Predictive Techniques used for Question 3 and Underlying Reasoning

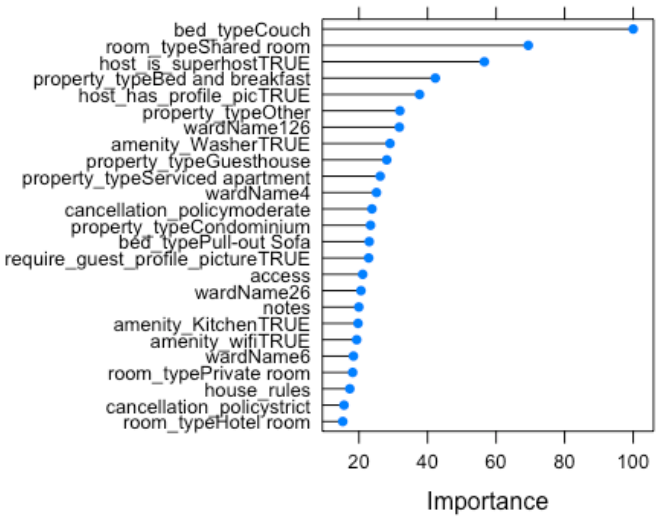
For question 3, the explanatory and predictive techniques used are similar with what we have done in question 1 and 2. The best explanatory model in question 3 used is ensemble method and logistic regression, because these methods can mark the significant and important variables and compare. For prediction models we choose random forest, which gives the lowest false positive rate to minimize the risk of loss for investors to the greatest extent. The coding portion of the models are displayed as below.

**Logistic Model**

```
fitLog<-glm(high_booking_rate ~ .-id, family ='binomial', data = df_dc)

summary(fitLog)

##
## Call:
## glm(formula = high_booking_rate ~ . - id, family = "binomial",
##     data = df_dc)
##
…
##
##     Null deviance: 6314.5  on 4868  degrees of freedom
## Residual deviance: 4132.7  on 4799  degrees of freedom
## AIC: 4272.7
##
## Number of Fisher Scoring iterations: 13
```

### Lasso regression

```
lambdaValues <- 10^seq(-3, 3, length = 100)
set.seed(2020)
fitLasso <- train(high_booking_rate ~ .-id, family ='binomial', data = df_dc,
 method='glmnet', trControl=trainControl(method='cv', number=10), tuneGrid =
expand.grid(alpha=1, lambda=lambdaValues))

#Variable importance complete table
varImp(fitLasso)$importance %>%
  rownames_to_column(var = "Variable") %>%
  mutate(Importance = scales::percent(Overall/100)) %>%
  arrange(desc(Overall)) %>%
  as_tibble()

## # A tibble: 71 x 3
##    Variable                          Overall Importance
##    <chr>                               <dbl> <chr>
##  1 bed_typeCouch                       100   100%
##  2 room_typeShared room                 69.4 69%
##  3 host_is_superhostTRUE                56.6 57%
##  4 property_typeBed and breakfast       42.3 42%
##  5 host_has_profile_picTRUE             37.7 38%
##  6 property_typeOther                   31.9 32%
##  7 wardName126                          31.8 32%
##  8 amenity_WasherTRUE                   29.0 29%
##  9 property_typeGuesthouse              28.1 28%
## 10 property_typeServiced apartment      26.2 26%
## # … with 61 more rows

#Variable importance plot with the most important variables
plot(varImp(fitLasso),top = 25)
```

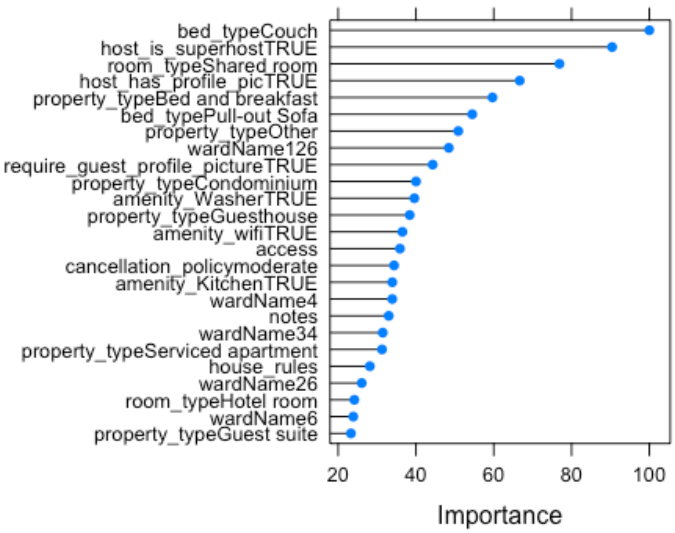



### Ridge regression

```
set.seed(2020)
fitRidge <- train(high_booking_rate ~ .-id, family ='binomial', data = df_dc,
method='glmnet', trControl=trainControl(method='cv', number=10), tuneGrid = e
xpand.grid(alpha=0, lambda=lambdaValues))

#Variable importance complete table
varImp(fitRidge)$importance %>%
  rownames_to_column(var = "Variable") %>%
  mutate(Importance = scales::percent(Overall/100)) %>%
  arrange(desc(Overall)) %>%
  as_tibble()
```

```
## # A tibble: 71 x 3
##    Variable                           Overall Importance
##    <chr>                                <dbl> <chr>
##  1 bed_typeCouch                          100   100.0000%
##  2 host_is_superhostTRUE                   90.4 90.4242%
##  3 room_typeShared room                    76.9 76.8950%
##  4 host_has_profile_picTRUE                66.7 66.6675%
##  5 property_typeBed and breakfast          59.7 59.6675%
##  6 bed_typePull-out Sofa                   54.5 54.4700%
##  7 property_typeOther                      50.9 50.8823%
##  8 wardName126                             48.4 48.4445%
##  9 require_guest_profile_pictureTRUE       44.3 44.3030%
## 10 property_typeCondominium                40.0 40.0144%
## # … with 61 more rows
```

```
#Variable importance plot with the most important variables
plot(varImp(fitRidge),top = 25)
```

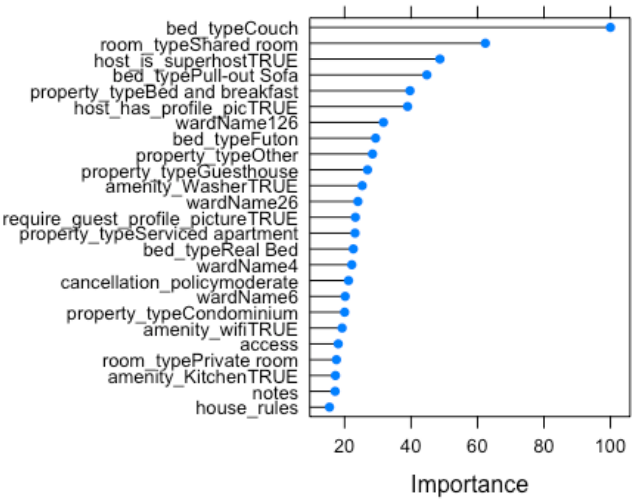



### Elastic Net

```
set.seed(2020)
fitElastic <- train(high_booking_rate ~ .-id, family ='binomial', data = df_d
c, method='glmnet', trControl=trainControl(method='cv', number=10),tuneGrid=e
xpand.grid(alpha=0.5, lambda=lambdaValues))

#Variable importance complete table
varImp(fitElastic)$importance %>%
  rownames_to_column(var = "Variable") %>%
  mutate(Importance = scales::percent(Overall/100)) %>%
  arrange(desc(Overall)) %>%
  as_tibble()
```

```
## # A tibble: 71 x 3
##    Variable                        Overall Importance
##    <chr>                             <dbl> <chr>
##  1 bed_typeCouch                     100   100%
##  2 room_typeShared room               62.4 62%
##  3 host_is_superhostTRUE              48.7 49%
##  4 bed_typePull-out Sofa              44.8 45%
##  5 property_typeBed and breakfast     39.7 40%
##  6 host_has_profile_picTRUE           39.0 39%
##  7 wardName126                        31.7 32%
##  8 bed_typeFuton                      29.3 29%
##  9 property_typeOther                 28.4 28%
## 10 property_typeGuesthouse            26.9 27%
## # … with 61 more rows
```

```
#Variable importance plot with the most important variables
plot(varImp(fitElastic), top = 25)
```




### Prediction Model

We would like a model with the smallest false rate in predicting false positives. The reason is that, property investment involves large amount of money. We would not want to make false predictions if a property would not achieve high booking rate. Thus, we would want higher specificity.

```
# Split data to train the model
set.seed(333)
df_dcTrain <- df_dc %>%
  sample_frac(0.8)
df_dcTest <- setdiff(df_dc, df_dcTrain)
```

### Lasso Prediction

```
set.seed(2020)
fitLasso0 <- train(high_booking_rate ~ .-id, family ='binomial', data = df_dc
,
                   method='glmnet', trControl=trainControl(method='cv', number
=10),
                   tuneGrid = expand.grid(alpha=1, lambda=lambdaValues))
resultsLasso <-
  fitLasso0%>%
  predict(df_dcTest, type='raw') %>%
  bind_cols(df_dcTest, predictedClass=.)

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

## Confusion Matrix and Statistics
##
##               high_booking_rate
```

```
## predictedClass   0   1
##              0 567 106
##              1  86 215
##
##                Accuracy : 0.8029
##                  95% CI : (0.7765, 0.8274)
##     No Information Rate : 0.6704
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5467
##
##  Mcnemar's Test P-Value : 0.1703
##
##             Sensitivity : 0.6698
##             Specificity : 0.8683
##          Pos Pred Value : 0.7143
##          Neg Pred Value : 0.8425
##              Prevalence : 0.3296
##          Detection Rate : 0.2207
##    Detection Prevalence : 0.3090
##       Balanced Accuracy : 0.7690
##
##        'Positive' Class : 1
##
```

### Ridge Prediction

```
set.seed(2020)
fitRidge0 <- train(high_booking_rate ~ .-id, family ='binomial', data = df_dc
Train,
                   method='glmnet', trControl=trainControl(method='cv', number
=10),
                   tuneGrid = expand.grid(alpha=0, lambda=lambdaValues))

resultsRidge <-
  fitRidge0%>%
  predict(df_dcTest, type='raw') %>%
  bind_cols(df_dcTest, predictedClass=.)

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

## Confusion Matrix and Statistics
##
##                high_booking_rate
## predictedClass   0   1
##              0 566 119
##              1  87 202
##
```

```
##                Accuracy : 0.7885
##                  95% CI : (0.7615, 0.8138)
##     No Information Rate : 0.6704
##     P-Value [Acc > NIR] : 2.588e-16
##
##                   Kappa : 0.509
##
##  Mcnemar's Test P-Value : 0.03078
##
##             Sensitivity : 0.6293
##             Specificity : 0.8668
##          Pos Pred Value : 0.6990
##          Neg Pred Value : 0.8263
##              Prevalence : 0.3296
##          Detection Rate : 0.2074
##    Detection Prevalence : 0.2967
##       Balanced Accuracy : 0.7480
##
##        'Positive' Class : 1
##
```

### Elastic Net Prediction

```
set.seed(2020)
fitElastic0 <- train(high_booking_rate ~ .-id, family ='binomial', data = df_
dcTrain, method='glmnet', trControl=trainControl(method='cv', number=10), tun
eGrid = expand.grid(alpha=0.5, lambda=lambdaValues))
resultsElastic <- fitElastic0 %>%
  predict(df_dcTest, type='raw') %>%
  bind_cols(df_dcTest, predictedClass=.)

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

```
## Confusion Matrix and Statistics
##
##               high_booking_rate
## predictedClass   0   1
##              0 568 105
##              1  85 216
##
##                Accuracy : 0.8049
##                  95% CI : (0.7786, 0.8294)
##     No Information Rate : 0.6704
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5515
##
##  Mcnemar's Test P-Value : 0.1681
```

```
##
##             Sensitivity : 0.6729
##             Specificity : 0.8698
##          Pos Pred Value : 0.7176
##          Neg Pred Value : 0.8440
##              Prevalence : 0.3296
##          Detection Rate : 0.2218
##    Detection Prevalence : 0.3090
##       Balanced Accuracy : 0.7714
##
##        'Positive' Class : 1
##
```

### Random Forest Using 10-fold CV

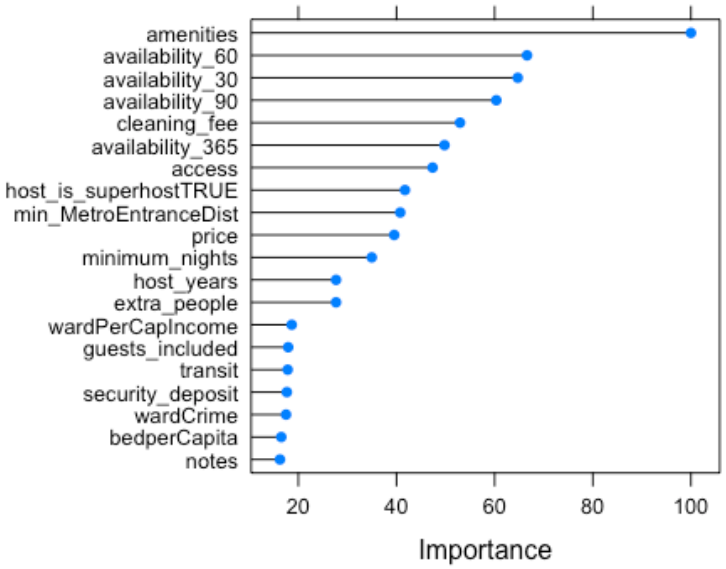
```
tuneGrid <- expand.grid(.mtry = c(1 : 10))

fitTree <-
  train(high_booking_rate ~ .-id, family ='binomial', data = df_dcTrain, meth
od = 'rf', na.action=na.pass, trControl=trainControl(method="cv",number = 10,
 allowParallel=TRUE), tuneGrid = tuneGrid, ntree = 200)

resultsRandomF <-
  fitTree %>%
  predict(df_dcTest, type = 'raw', na.action=na.pass) %>%
  bind_cols(df_dcTest, predictClass=.)

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

```
## Confusion Matrix and Statistics
##
##             high_booking_rate
## predictClass   0   1
##            0 587 100
##            1  66 221
##
##                Accuracy : 0.8296
##                  95% CI : (0.8045, 0.8527)
##     No Information Rate : 0.6704
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.6037
##
##  Mcnemar's Test P-Value : 0.01043
##
##             Sensitivity : 0.6885
##             Specificity : 0.8989
##          Pos Pred Value : 0.7700
```

```
##          Neg Pred Value : 0.8544
##              Prevalence : 0.3296
##          Detection Rate : 0.2269
##    Detection Prevalence : 0.2947
##       Balanced Accuracy : 0.7937
##
##        'Positive' Class : 1
##
```

### Bagged decision tree

```
set.seed(2020)
fitBaggedTree <- train(high_booking_rate ~ .-id, data=df_dcTrain, method='tre
ebag',
                       trControl=trainControl(method='cv', number=10))

#See the CV output (accuracy per pruning parameter etc.)
fitBaggedTree$finalModel
##Bagging classification trees with 25 bootstrap replications

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




```
#See the variables listed by importance (according to the bagged tree)
varImp(fitBaggedTree)$importance %>%
  rownames_to_column(var = "Variable") %>%
  mutate(Importance = scales::percent(Overall/100)) %>%
  arrange(desc(Overall)) %>%
  as_tibble()
```

```
## # A tibble: 83 x 3
##    Variable              Overall Importance
##    <chr>                   <dbl> <chr>
##  1 amenities               100   100%
##  2 availability_60          66.6 67%
##  3 availability_30          64.7 65%
##  4 availability_90          60.3 60%
##  5 cleaning_fee             52.9 53%
##  6 availability_365         49.8 50%
##  7 access                   47.4 47%
##  8 host_is_superhostTRUE    41.7 42%
##  9 min_MetroEntranceDist    40.8 41%
## 10 price                    39.5 40%
## # … with 73 more rows
```

```
#Make predictions:
resultsBaggedTree <-
  fitBaggedTree %>%
  predict(df_dcTest, type='raw') %>%
  bind_cols(df_dcTest, predictedClass=.)

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

```
## Confusion Matrix and Statistics
##
##               high_booking_rate
## predictedClass   0   1
##              0 572 108
##              1  81 213
##
##                Accuracy : 0.806
##                  95% CI : (0.7797, 0.8303)
##     No Information Rate : 0.6704
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.5513
##
##  Mcnemar's Test P-Value : 0.05859
##
##             Sensitivity : 0.6636
##             Specificity : 0.8760
##          Pos Pred Value : 0.7245
##          Neg Pred Value : 0.8412
##              Prevalence : 0.3296
##          Detection Rate : 0.2187
##    Detection Prevalence : 0.3018
##       Balanced Accuracy : 0.7698
##
```

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

**Cutoff selection**

In the above models, we can see that random forest model has the highest specificity(88.67%) with the default cutoff. What if we alter the cutoff? Therefore we tried to set cut-off at 0.5, 0.6, 0.7, 0.69, 0.65.

Based on our goal to minimize false positives, we choose 0.69 as our final cut-off, which achieves a specificity for random forest model of 97.55%. With this model and cutoff, there would be only 2.45% chance that the investor invests a property that would not have high booking rate.

# 5. Results and Findings

## 5.1 Results on Question 1: Community characteristics (Where the house should be to invested)

In question 1 where the model only performs analysis with community feature independent variables, the model performance is not acceptable.The result from the linear regression model implies that only 1.96% of the variation of the dependent variable is explained, which is the probability of the property becoming a house with a high booking rate.For the logistic model, The AIC and BIC values are also extremely high, which represent a bad performance for the model. The variance is high as well. Among the four variables, several wardName,wardCrime and wardCapIncome have a significant impact. Neither of these models are qualified to do further prediction. However, we can see there is a difference among the wards.

## 5.2 Results on Question 2: Property characteristics (what kind of house should be to invest)

To verify these considerations, we built some different explanatory models. Because most variables are categorical variables, we use different data classification methods to fit the model, and then do the model selection based on their performances. We firstly did logistic regression by setting the cutoff as 0.5. The VIF shows some categorical variables have a certain level correlation, but if dropping one of them through subset selection, the AIC will increase. Then we use the regularization method to build Lasso, ridge regressions and Elastic net. We want to find a model which has the lowest false positive rate (1-specificity), because the cost incurred by mispredictive ture (False Positive) is way larger than the benefit of corrective true Positive. If we predict that an Airbnb with certain features can be popular in the D.C. market, and recommend our investors to invest in, but in reality, the booking rate of his Airbnb is pretty low, which is a disaster. So by fitting with four different models with different classification methods, the ROC curve shows the ridge model has the lowest false positive rate and highest accuracy. Thus, the ridge model has best performance.

According to the variable importance table, we draw the conclusion that: property-type and bed type have significant impact on the popularity of the Airbnbs in the D.C. market. So we would like to suggest to investors that Property type of the Bed and breakfast, the Villa, the Hostel, the Bungalow and the Condominium should be avoided. When considering adding extra beds to increase the probability of being selected by potential Airbnb guests, Pull-out Sofa could be a good choice.

We also did a visualization to illuminate the relationship among different variables. For apartment and house type Airbnb, the higher price, the lower probability to be a high-booking-rate Airbnb. For Town house Airbnb, with price increasing, the probability of being high-booking-rate and non-high-booking rate is basically the same. The face wrap is classified by different numbers of bathrooms. This graph shows: For one bathroom Airbnbs, under the same level price, the more bedrooms, the higher probability to be a high-booking-rate Airbnb. For the Airbnbs containing one and half or two bathrooms, even with a higher price, the more bedrooms, the probability to be a high-booking-rate Airbnb is high. So, we would like to suggest that townhouse could be an option especially if it comes with a bathrooms range from one to two and a bedrooms rang from one to four, when others factors are at same levels.

## 5.3 Results on Question 3: Predict whether a property in DC would have a high booking rate

**Results and findings for Explanatory Model**

The logistic regression and ensemble methods list some important/significant variables to compare. Here are some important variables in a glance. 1. access, notes, house_rules: having these descriptions would boost impression of the asset. 2. wardName: properties in some wards such as 126 would have higher booking rates than others. 3. amenity_wifi: important in amenities to attract customers. 4. room_type, property_type: certain types are achieving higher booking rate, as explained in house feature studies. 5. host_is_superhost: becoming a superhost would help boost booking rate. But we assume that should give credit to the marketing strategy of Airbnb.

**Results and findings for Prediction Model**

Based on our goal to minimize false positives, we choose random forest as our final modeling method and 0.69 as our final cuttoff, which achieves a specificity of 97.55%. With this model and cutoff, there would be only 2.45% chance that the investor invests a property that would not have high booking rate.

## 5.4 Overall Recommendation to investor

1.Ward 1, Ward 2, Ward 6 are good choices to consider on purchasing airbnb homes. It is not necessary for investors to consider much about other community features in D.C.

2.The relatively popular Airbnb structure are the ones with three-bedrooms and one or two bathrooms. Most townhouse qualify such features. So, townhouse could be an good option to be considered as investment when others factors are at the same level.

3.It is a good consideration to contain extra beds for two and three bedrooms Airbnb without raising price too much. It is very helpful to increase the probability of booking rate.

4.Having more descriptions of the property and the host himself/herself would boost the impression.

5.Creating more comfortable environments to customers by providing key amenities: wifi, washer and dryer.

6.Become a superhost.

## 6. Conclusion and Discussion

We have concluded that in order for to invest wisely in airbnb homes, business investors should purchase a home preferably in ward 1, 2 or 6 of the Washington DC city. Other than the ward assignment, the rest community features could be negligible including the crime index of each ward. This could potentially imply that either customers on airbnb for DC market choose not to value much on the safety of the surrounding neighborhood or there is not a significant difference on the crime rate for different wards of DC that is enough to make an impact.

We also figured out that a three bedroom with one or two bathroom types is the most popular arrangement while there are most one bedrooms offered in the DC market. This can imply that customers booking airbnb in DC are more likely to arrive in groups than single people since three bedrooms are more popular than one bedroom type.

More importantly, after purchasing the properties. The host could work on the website listings on a thorough description of the home and the host itself. This could give customers sufficient details and a closer understanding of the property before actually arriving at the site. Also, planning to have the essential amenities like Wifi, microwave, washer etc, could contribute to reaching a high booking rate as well. Moreover, becoming a superhost with multiple airbnb home listings would help increase the booking rate. Customers would believe that superhost have sufficient experience in managing airbnb homes and have provided great rooms so that they have the chance to grow and become a superhost.

Overall, our model performance and results are reasonable with validated statistics. The only limitation to our model is that we would not be able to utilize all the variables possible in the dataset due to various reasons. The most vital one is the abundant missing values. We have replaced some missing values with mean, median or zero. These missing values would create deviance to the model result to some extent. Therefore, our future research direction would be trying to gather more data on airbnb homes with complete and detailed information so that we can perform the model analysis in an even more precise manner.

## 7. Reference

a. Metro Station Entrances in DC: 6/25/2019
https://opendata.dc.gov/datasets/metro-station-entrances-in-dc/data?orderBy=FEATURECOD&orderByAsc=false

b. How to calculate geographic distance: http://www.nagraj.net/notes/calculating-geographic-distance-with-r/
c. DC Crime Map
https://dcatlas.dcgis.dc.gov/crimecards/all:crimes/all:weapons/2:years/citywide:heat
d. DC Neighborhoods in Wards
https://en.wikipedia.org/wiki/Neighborhoods_in_Washington,_D.C.
e. Per Capita Income of Wards in D.C.
https://dcdataviz.dc.gov/page/ward-income-indicators#3