Final Report

Jie TANG, Zi LONG, Peihan TIAN, Zhengxuan CAO

1. Exploratory Data Analysis

I. Datasets Processing

We mainly use the listings_detail dataset and calendar dataset. The process of data processing is exploring the data, cleaning the data, selecting useful features and constructing new features. We use R language to process the data.

1.1 Listings_detail dataset processing

Dealing with useless columns
Firstly, we dropped some useless variables such as scrape id. Also, We dropped 11 variables that are constant almost all the time and have no relationship with price. Then we dropped 1 variable that has no difference with another variable. After that, we dropped the variables that is irrelevant with the price, such as the url variables. Then we dropped all the variables which is hard to use into the price predict model. which has more than 50000 NAs. At the end, we checked the length of observations in the each scraped and discovered that there are totally 5 rows in three different scraped date which is abnormal, so we dropped these rows.

Secondly, we cleaned the dataset. We processed each kind of variables as following:
id variables: change the type as numeric variables.
logic variables: change into 0-1 variable.
date variables: change into the type as date variables.
price variables: remove the $ sign, comma sign and change the type as numeric variables.
variables 'host_response_rate': remove the % sign and change the type as numeric variables.
variables 'zipcode': check all the zip code, then combine same zipcode by removing the "MA" or reform it as 4-digit zipcode.

Dealing with NAs
Thirdly, we remove all the NAs. NAs appears in some of the variables and some variables have relatively large amount of NAs. We dealt with the NAs as following:
For the variable 'beds', 'bedrooms' and 'bathrooms', we remove the rows that contains at least 1 NAs, because there are only about 300 NAs in these three columns in total, which is not a big deal.
For continuous numeric, we filled NAs with the mean of the each column without outliers.
For 0-1 variables, we calculated the probability of having 0 and 1 in that column and filled NAs with the 0 or 1 that has higher probability.

Dealing with outliers
Fourthly, we removed the outliers in continuous numeric variables at 99% interval.

1.2 calendar dataset processing

Since the calendar dataset is rather clean, we process the dataset in less process.

Dealing with useless columns and data cleaning
Firstly, we keep variables 'listing_id', 'date' and 'adjusted_price' and dropped all other variables in that dataset. Because we only want to keep the effect of time for of the price. **We use adjusted price to be our target** instead of price because we thought the 'adjusted_price' is more accurate than variable 'price'. Since one listing may has more than one adjust price, so we calculate the average adjust price by grouping

the listing id and date.
Secondly, we change the type of variables ‘date’ into date. Then we removed the $ sign, comma sign from variable ‘adjusted_price’ and changed its type as numeric variables for further steps.

2. Feature Engineering

Choosing features with exist variables
After cleaning all the dataset, we choose our features which are relevant and important for the price.
We choose first choose variables that seems to have relationship with price including:
'host_response_rate','host_is_superhost','host_has_profile_pic','host_identity_verified','is_location_exact', 'accommodates', 'bathrooms', 'bedrooms', 'beds', 'price', 'security_deposit', 'cleaning_fee', 'guests_included', 'extra_people', 'minimum_nights', 'maximum_nights','availability_60','availability_365', 'number_of_reviews', 'review_scores_accuracy','review_scores_cleanliness','review_scores_checkin', 'review_scores_communication', 'review_scores_location', 'review_scores_value', 'requires_license', 'instant_bookable', 'require_guest_profile_picture','require_guest_phone_verification', 'reviews_per_month', 'review_scores_rating', 'host_years', 'zip_has', 'host_response_price', 'neighbourhood_cleansed_price', 'property_type_price', 'room_type_price', 'bed_type_price', 'cancellation_policy_price'.

We did one variable and two variables analysis to all those variables, trying to figure out if they are important factors of affecting housing price.
We divide our features into three parts: host-relevant, house-relevant, reviewing-relevant
After we did the explanatory data analysis by drawing histograms, scatter plots, and maps. We find out several variables that relevant with price.

Host-relevant*: 'host_response_rate', 'host_is_superhost','host_has_profile_pic', 'host_identity_verified'*

'host_response_price'.

Reviewing-relevant: *'review_scores_accuracy', 'review_scores_cleanliness','review_scores_checkin',*

'review_scores_communication', 'review_scores_location'.

House-relevant: *'neighbourhood_cleansed', 'property_type_price','room_type', 'bed_type',*

'cancellation_policy'.

Plotting to find out pattern of existing features

Super-host effect on price:

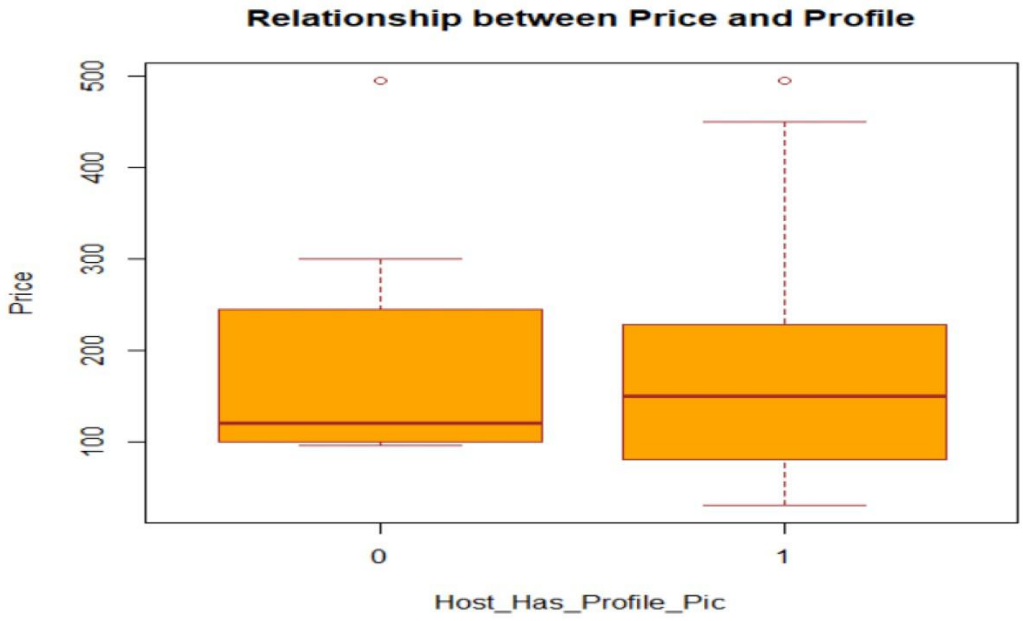
The box plot show a significant difference between none-superhost and superhost.

Host identification effect on price:

The box plot shows that the median, and distribution of price are different based on whether host has identified verification or not.

Host profiles' effect on price:

The box plot shows a significant difference of price in whether host has a profile picture based on the difference of median and distribution of price.

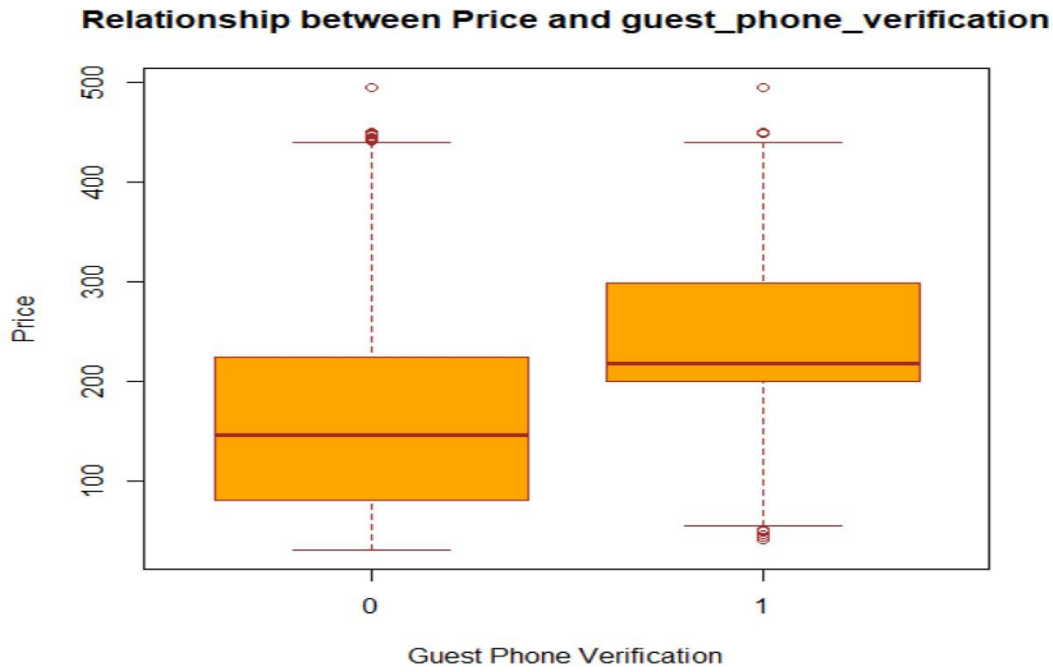


Location's effect on price:

The box plot shows a significant difference of price on whether the location is exact or not.
We can conclude that price will be slightly higher if host provides an exact location.

Guest phone verification effect on price:

As the plot shows, price is significantly different on whether guest has a phone verification or not.

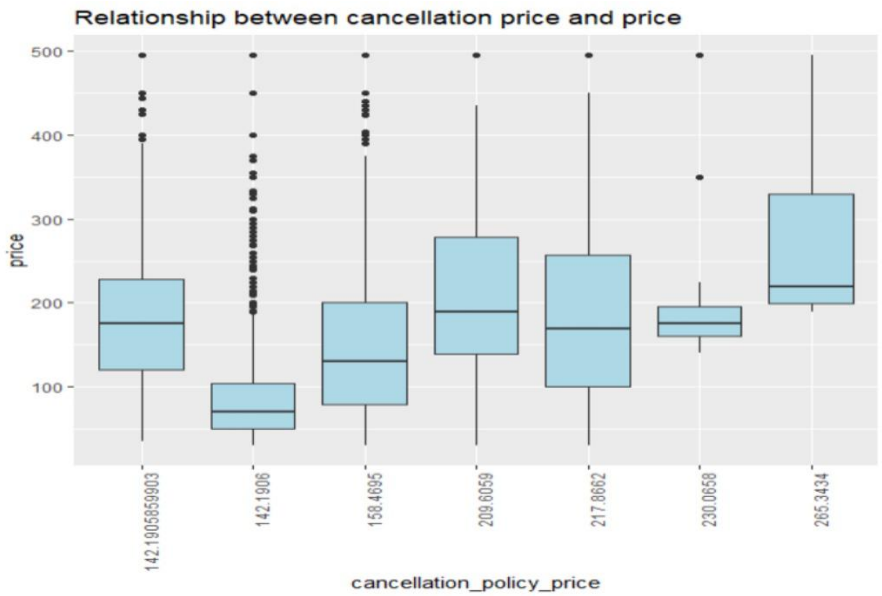


Cancellation police effect on price:

As the plot shows, price and cancellation policy prices are significantly different. The higher the cancellation price, the higher the house price.

Descriptive statistics description of cancellation policy price.

Min	Max	Mean	Median	Standard Deviation
142.2	265.3	187.1	217.9	35.46



Property type price's effect on price:

The box plot shows a trend that the higher the property type price, the higher the house price. Property type price represents the average price for different kinds of property type.

Descriptive statistics description of property type price.

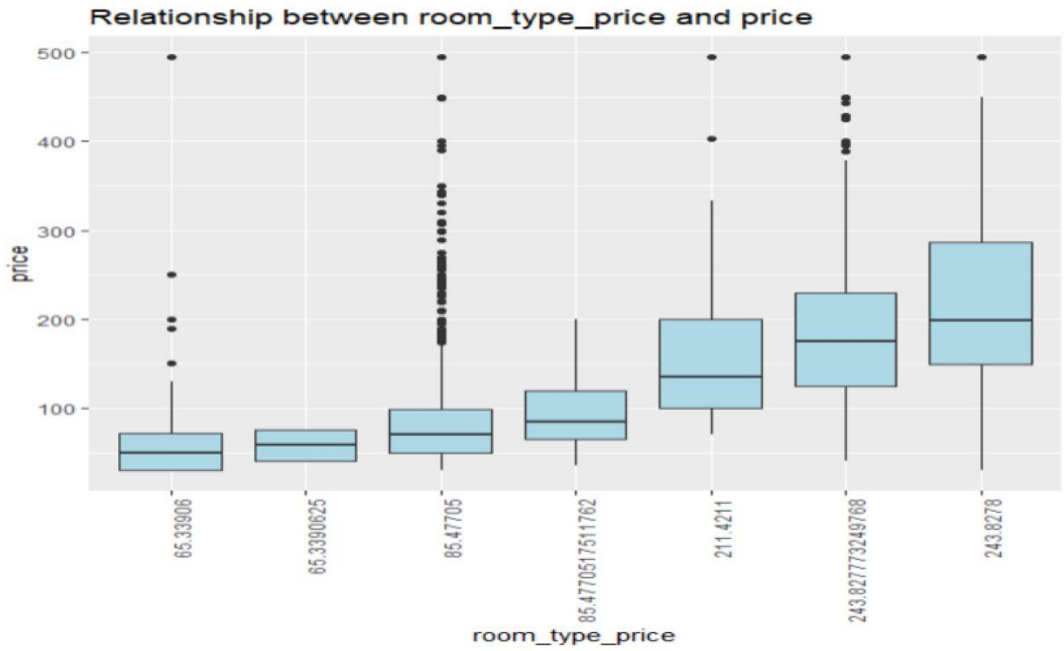
Min	Max	Mean	Median	Standard Deviation
30	500	187.1	199.2	36.74

Room type price's effect on price:

The plot show that as the room type price increases,the price also increases. Room type price stands for the average price for different kinds of room types in Boston area.

Descriptive statistics description of room type price.

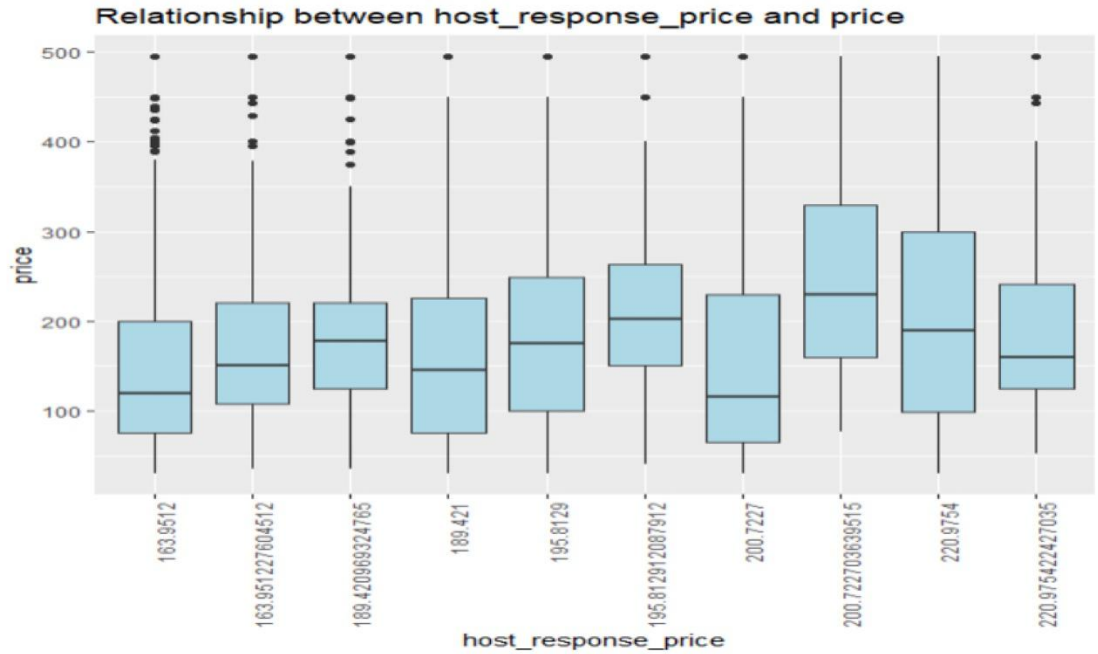
Min	Max	Mean	Median	Standard Deviation
65.34	243.83	187.06	243.83	76.18



Host response price's effect on price:

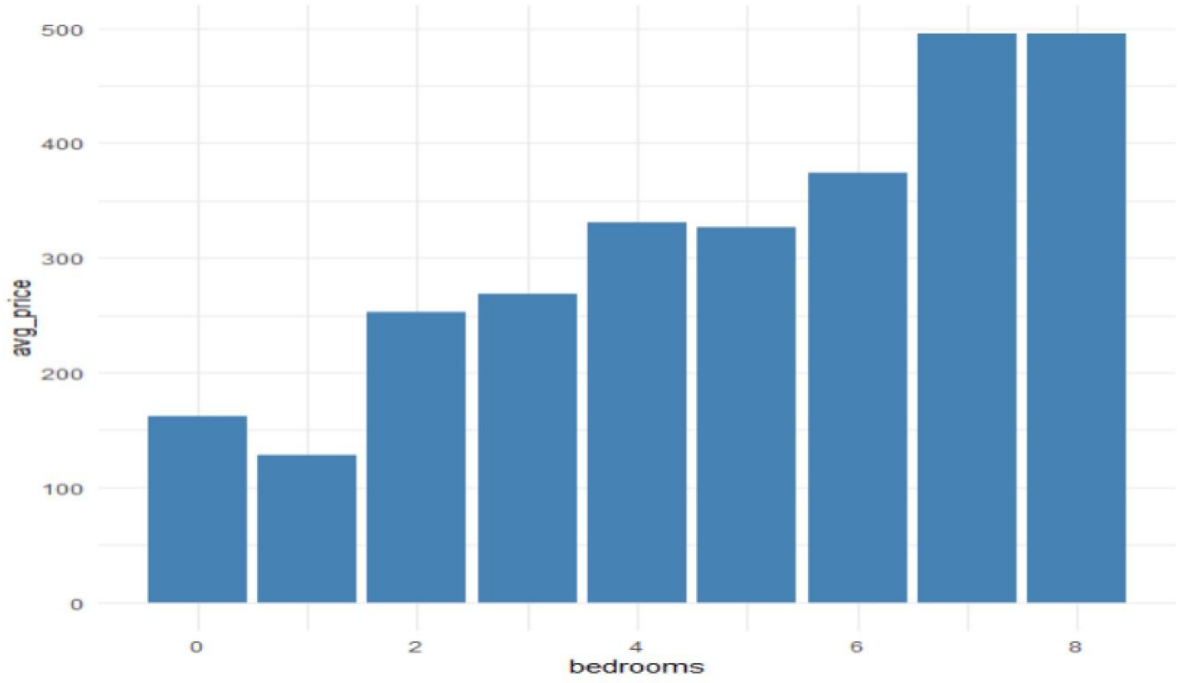
Descriptive statistics description of host response price.

Min	Max	Mean	Median	Standard Deviation
164	221	187	189.4	14.57

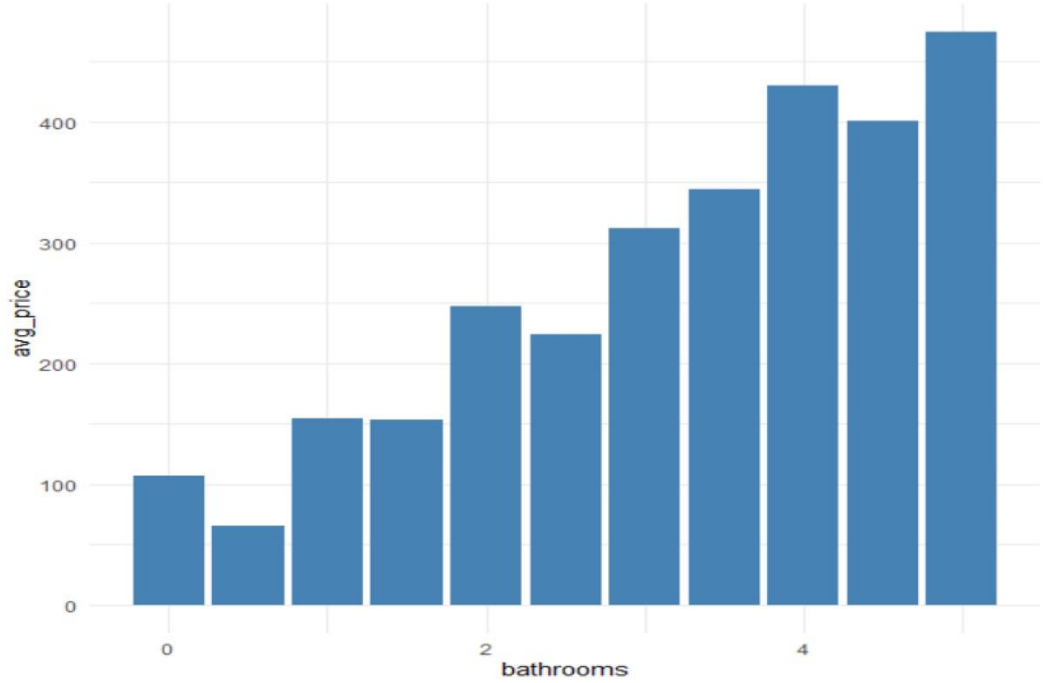


The number of bedrooms on average price:

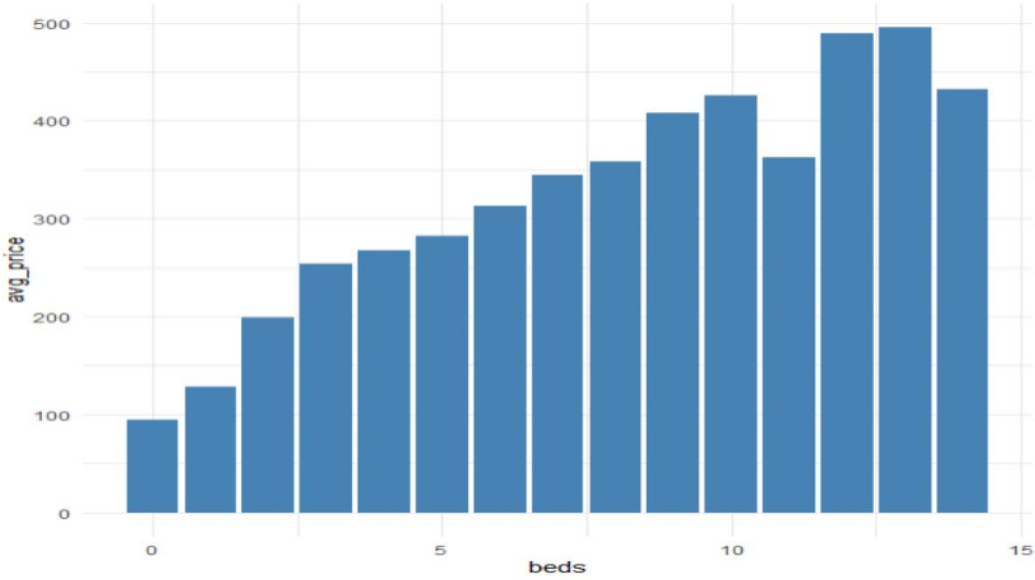
As the plot shows, with the increase of bedroom numbers, the average price of house increases.



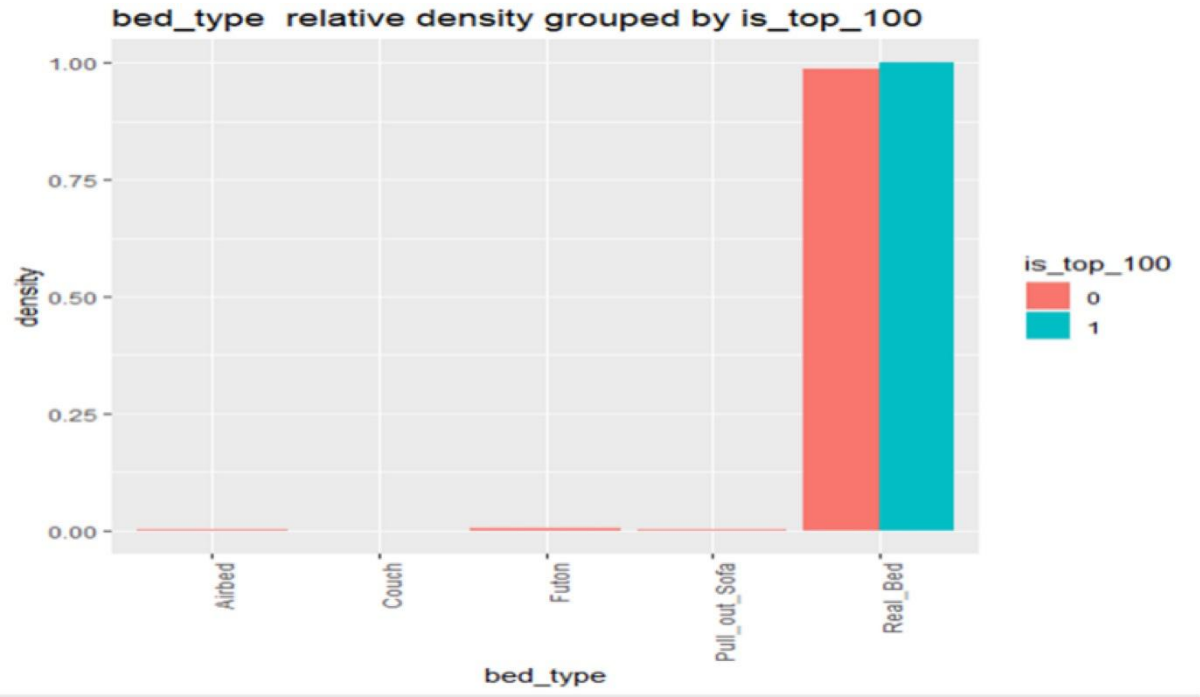
The number of bathrooms on average price:



The number of beds on average price:

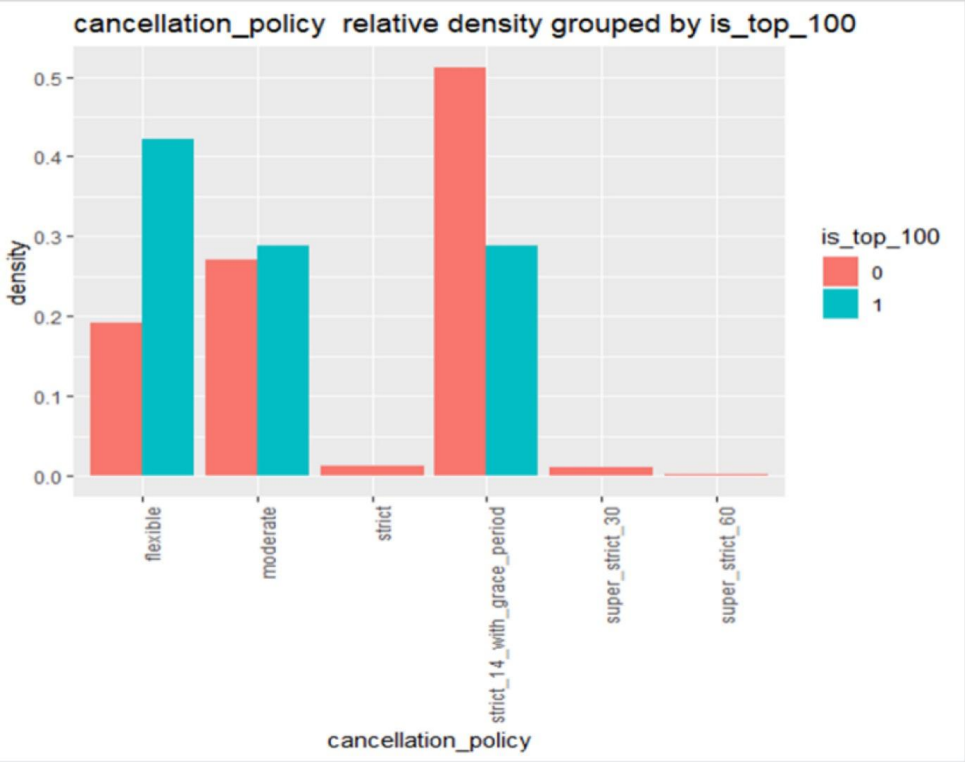


13, Different bed types' effects on top_100 houses:



As the result shows, most customers like real beds.

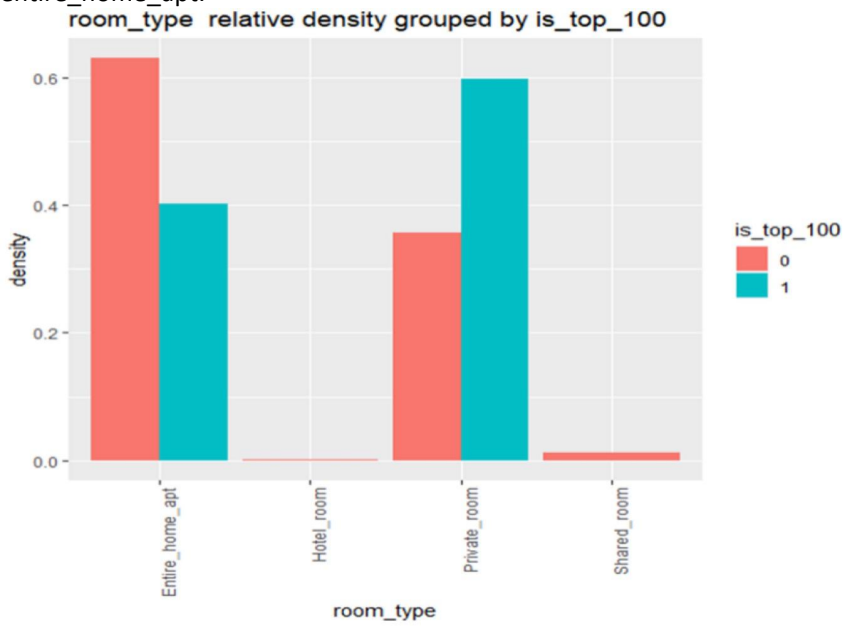
14, Different types of cancellation policy on top_100 houses:



As the plot shows, the stricter the policy, the fewer customers want to book the houses.

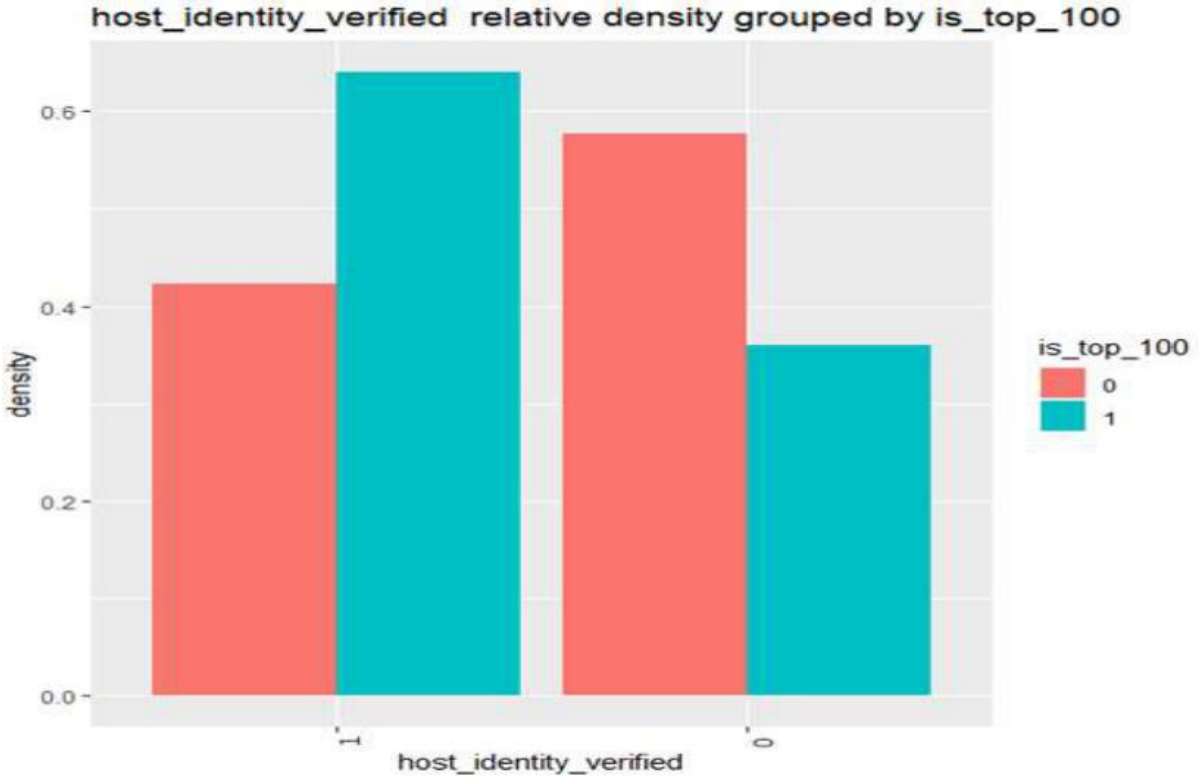
15, different room_type on the density of top_100 houses:

As the result shows, customers prefer private rooms than any other types. Besides it, they also like entire_home_apt.

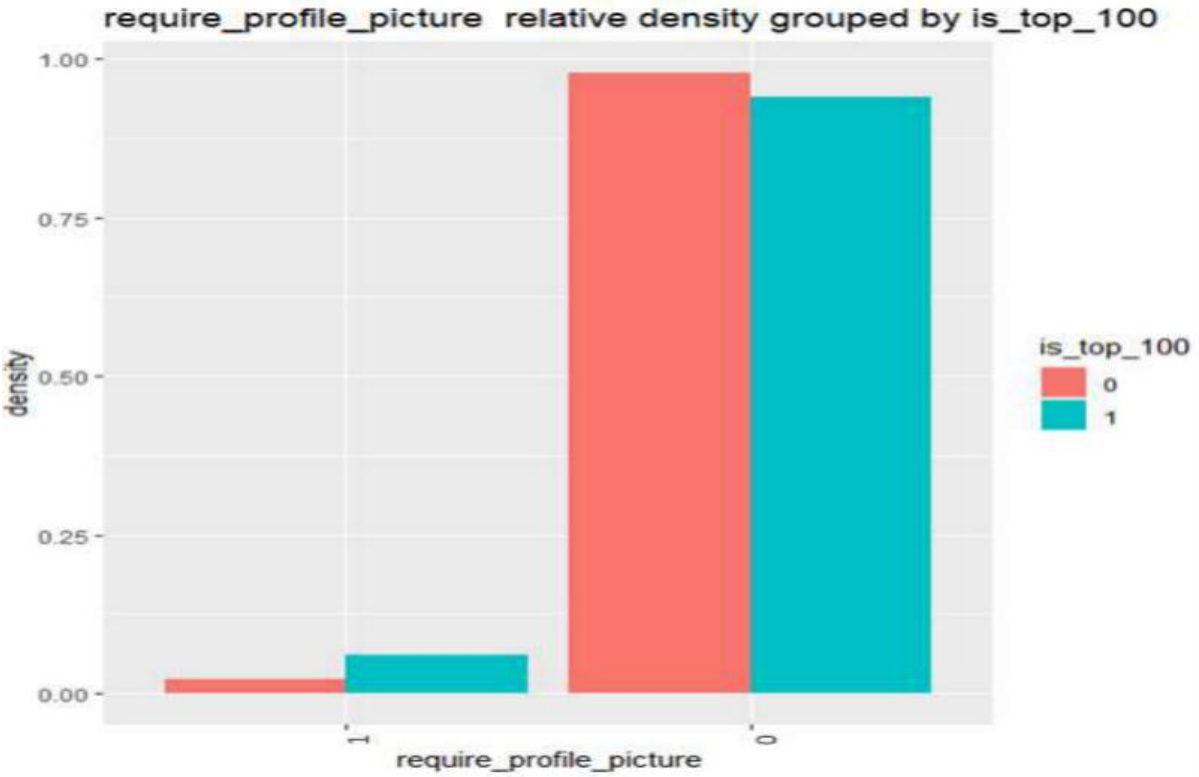


16,Host identification effects on top_100 houses:

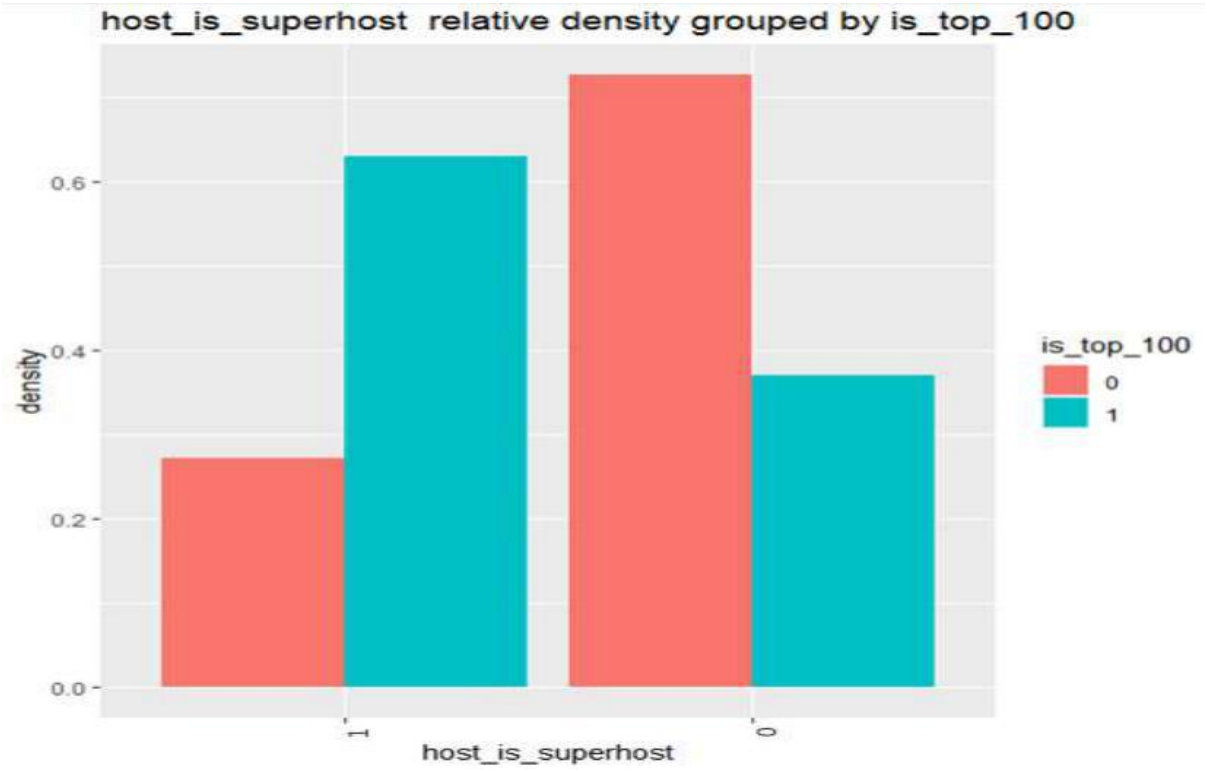
Most of the top_100 houses are houses with identification verification.



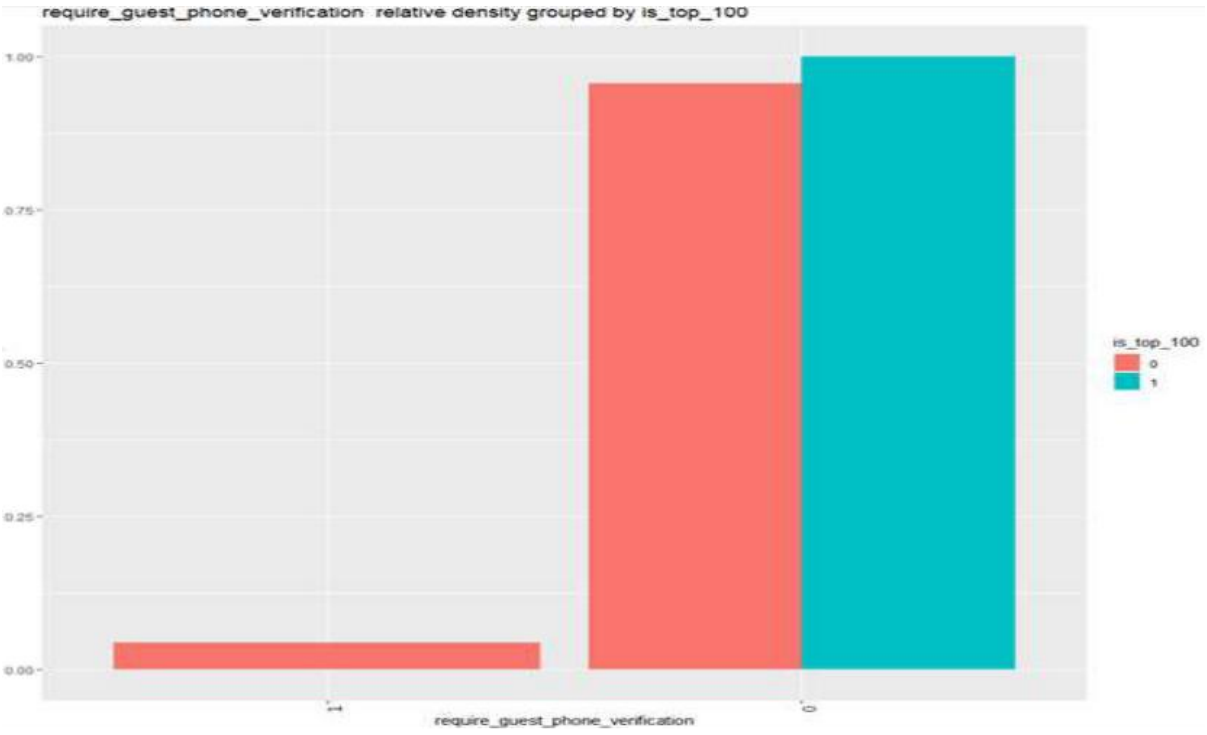
17, required profile effect for top_100 houses:



18, host_is_superhost effect on top_100 houses:



19, require_guest_verification effect on top_100_houses:



Constructing new features with exist data

Because there are some important information that contains in the dataset, but we cannot use it directly, thus, we need to create new features with our dataset.

Our **new constructed features** are as follows:

zip_has: numeric variable, the number of listings under that zip code. Because there more listings under the zip code it has, the more prosperous this area are likely to be, thus, the higher housing price it should be.

host_response_price: numeric variable, the average price in each category of 'host_response_time'. 'host_response_time' is a categorical at first, the less host response rate a house is, the more popular and higher score it will be. So it will influence the house price somehow. Instead of using dummy variable here , we use the average price in each category to represent it 0-1 variable cannot show all the information it should be. Changing it to the average price in each category can reflect the information it contains.

neighbourhood_cleansed_price: numeric variable, the average price in each category of 'neighbourhood_cleansed'. Price may vary from neighborhoods to neighborhoods. The reason why we change 'neighbourhood_cleansed' to the average price in each category is that there are so many categories, if we use dummy variables, the useful information the variable it contains cannot reflect very well in the models.

property_type_price: numeric variable, the average price in each category of 'property_type'

The price of different property type can be different. For example, an apartment can be more expensive than a house if they are in the same conditions, because an apartment is safer with more advanced equipment. The reasons we use the average price in each category is the same as above.

room_type_price: numeric variable, the average price in each category of 'room_type'. Better room type might cost more. The reasons we use the average price in each category is the same as above.

bed_type_price: numeric variable, the average price in each category of 'bed_type'. Better bed type might cost more. The reasons we use the average price in each category is the same as above.

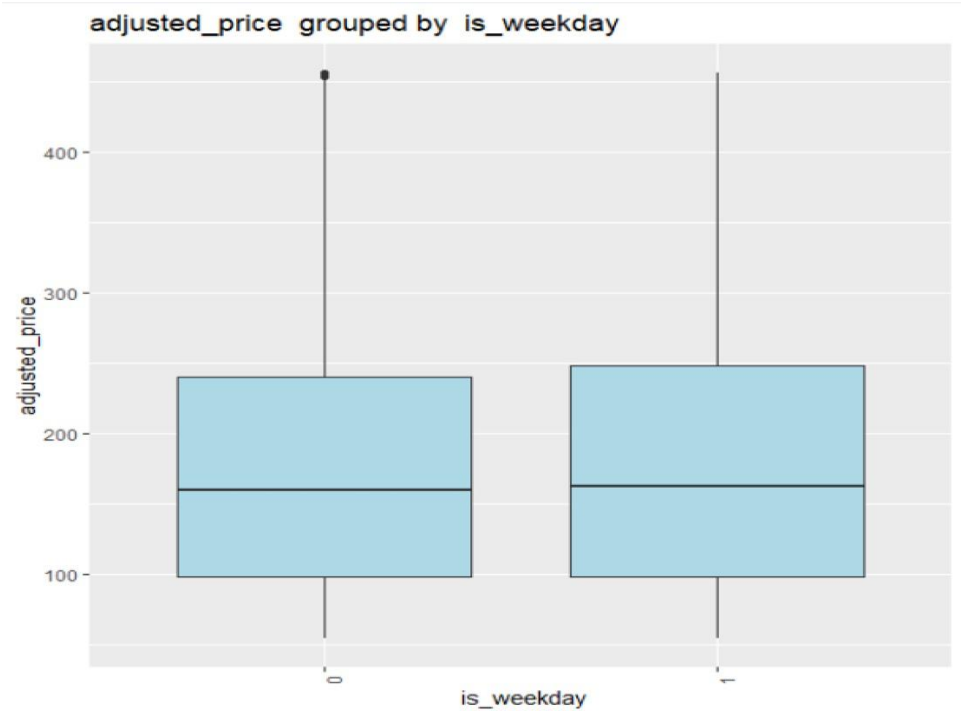
cancellation_policy_price: numeric variable, the average price in each category of 'cancellation_policy'. A more strict cancellation policy might discourage customers, thus affecting the price. The reasons we use the average price in each category is the same as above.

Also, we constructed our time features based on calendar dataset.

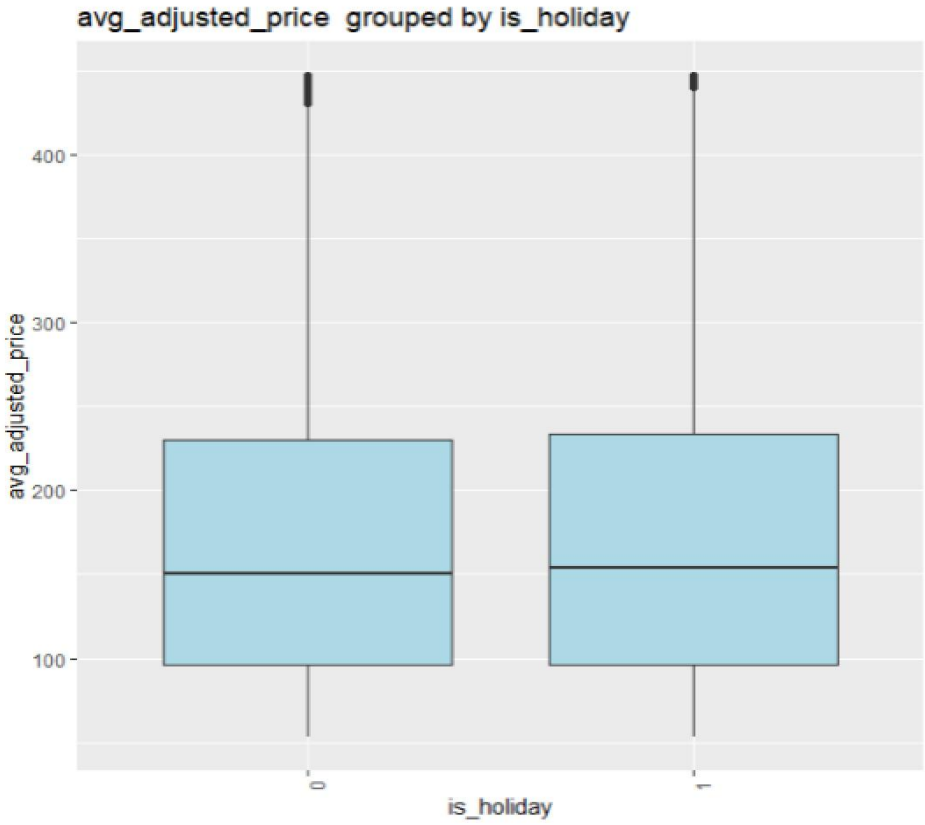
is_weekend : logic variable, 1 if the date is Friday or Saturday, 0 if the date is a day other than Friday or Saturday. We believe that, more people will go out and have fun on weekends, the demand of house will increase, thus increasing the price. The information below shows

Table1-1 Weekend effect on listing price

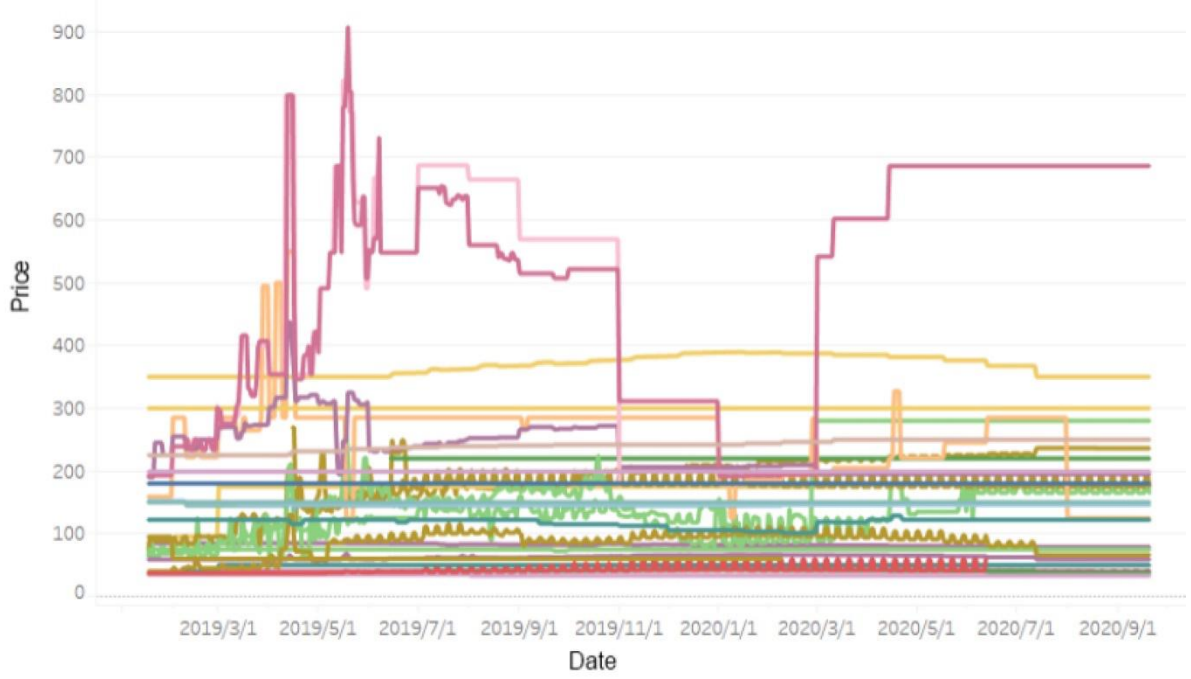
Effect	Have an increase	5% increase	10% increase	15% increase
Number of listing	70.5%	39.4%	13.6%	8.3%

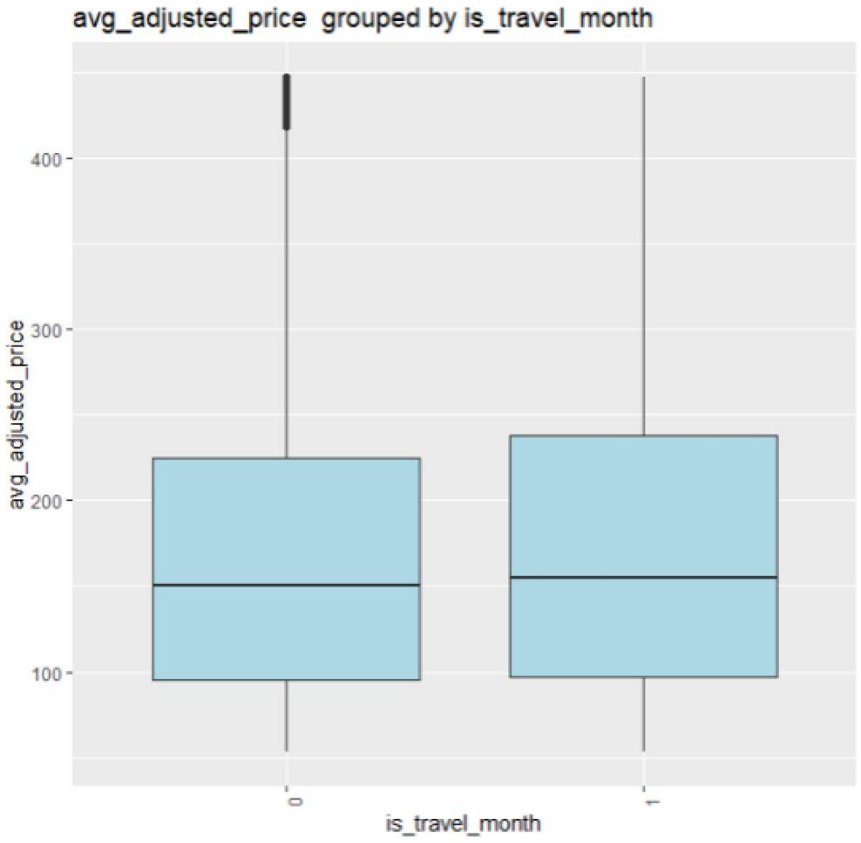


is_holiday: logic variable, 1 if the date is in holiday period, 0 if the date is not in holiday period. On holiday, people are more likely to have vacation, the demand of house will increase, thus increasing the price. We include the date of some very common holidays such as Christmas, Labor Day, Independence Day, and Thanksgiving Day. And we includes 7 days before these holidays reaching, for example, we include 21th November to 28th November to represent Thanksgiving Day, …. Because, when a holiday is reaching, people will plan to book houses ahead of time, thus the demand of houses will increase, raising the prices of houses. The graph below also shows that there's variance of holiday and non-holiday prices.



is_travel_month: logic variable, 1 if the date is in travel season(March, April, May, July and August), 0 if the date is not in the travel season. March, April, May, July and August are good months to travel, because there are very beautiful scenery outside during Spring and Summer. People are likely to go out and travel, demanding more houses, raising house price. The graph below also shows that there's variance during Spring and Summer. And the boxplot shows that there is obvious variance of the price of travel-month and non-travel months





After explanatory data Analysis, we finally chose these variables to be our features: 'is_weekday','is_travel_month','time_neighbor',
'is_holiday', 'zip_has', 'host_response_price',
'neighbourhood_cleansed_price', 'property_type_price',
'room_type_price', 'bed_type_price', 'cancellation_policy_price','host_years',
'host_response_rate', 'host_is_superhost','host_has_profile_pic', 'host_identity_verified',
'is_location_exact', 'bathrooms', 'bedrooms',
'cleaning_fee', 'guests_included',
'extra_people', 'minimum_nights',
'availability_60','availability_365',
'review_scores_accuracy', 'review_scores_cleanliness',
'review_scores_checkin', 'review_scores_communication',
'review_scores_location', 'review_scores_value','reviews_per_month',
'instant_bookable', 'require_guest_profile_picture',
'require_guest_phone_verification'

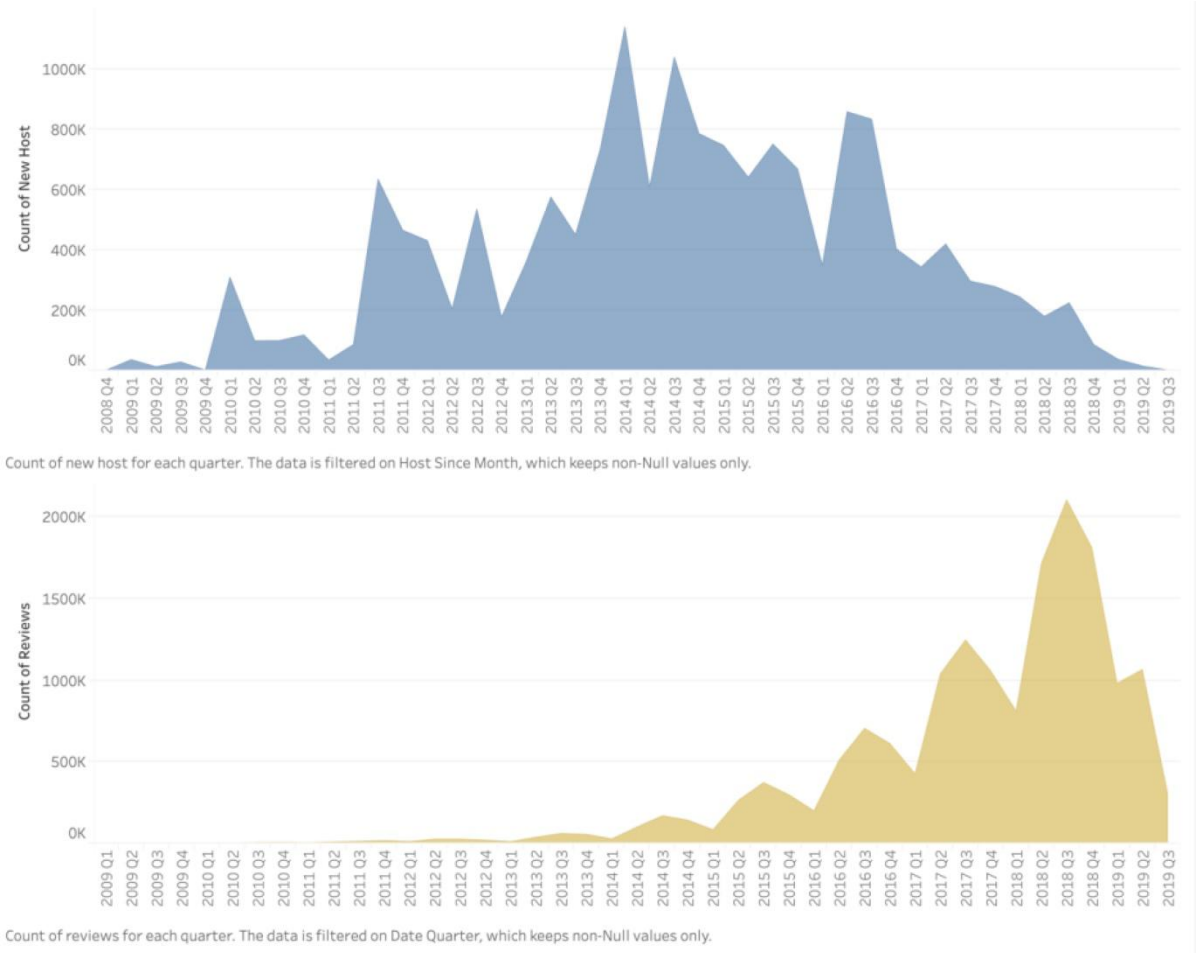
We use avg_adjusted_price to be our labels.

avg_adjusted_price: because there are some repeated dates of one house, so we use the average adjusted price of the same date of a household to represent the price of that day.

Create visualizations to answer the following questions:

How popular has Airbnb become in Boston?

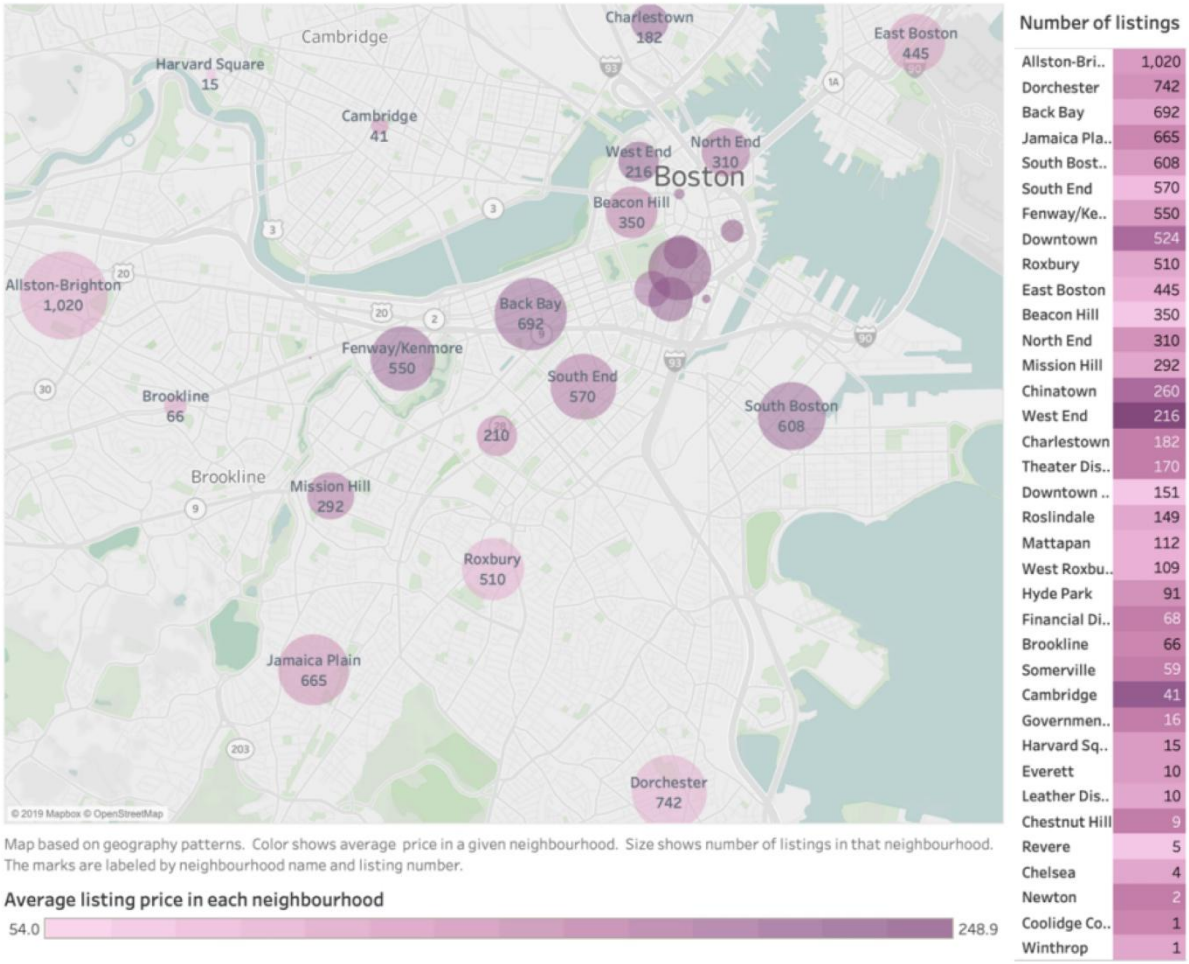
Airbnb, an online marketplace for accommodations, has experienced a staggering growth accompanied by intense debates and scattered regulations around the world. In this project, we study several aspects of Airbnb listings and prices in Boston Area. As one of the main cities in the United States, Boston has been a target of leisure tour and business trips for travelers from all over the world. Sharing housing, especially, in Boston, has experienced a booming development in recent years.



As the graph above indicates, the number of registered hosts has been growing since 2008, with a peak value of over 1000K new host in the second quarter of 2014. 2014 and 2015 are booming years for Airbnb, in terms of host side.

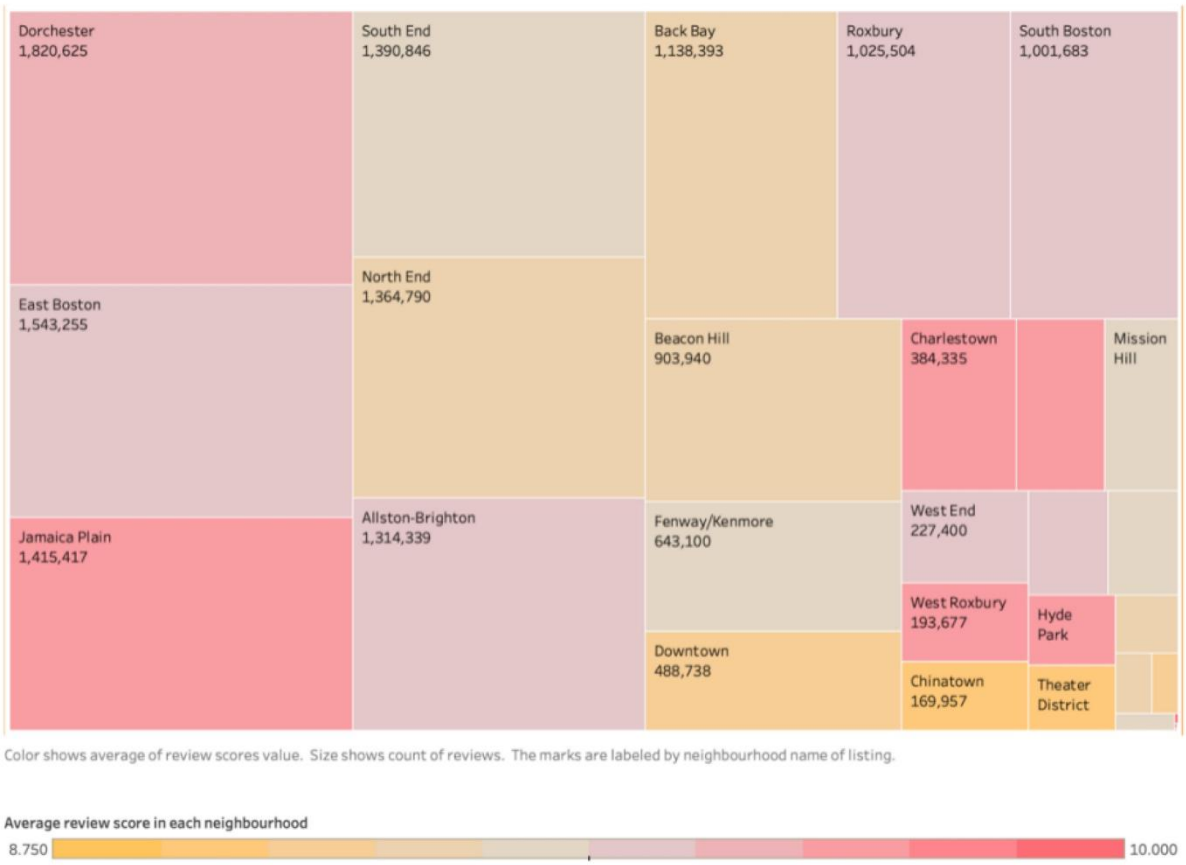
As for guests, it witnessed a dramatic growth from 2013 to 2018. The number of reviews left by guest exceeded 2000K in the third quarter of 2018, being the largest value in our dataset. Airbnb has been very popular in 2018 on the guest side.

What geography patterns appear in the Airbnb property listings?



If we group the data by neighbourhood, we will find that property listings in downtown Boston tend to have higher price. Also, neighbourhood near school and university, Fenway for instance, has higher price compared to others, too. As for the number of listings, Allston, Dorchester, Jamaica Plain and South Bost have more than 600 properties posted on Airbnb for stays. Some of them are in downtown area while others are not. Possible reason can be people tend to sell or post their properties not used on the website and live in downtown themselves for convenience.

Which neighborhood is the most popular among customers?



Looking at number of reviews only, Dorchester, East Boston, Jamaica Plain, South End and North End are the top five neighbourhoods chosen by customers. However, not all of them can be regarded as the most popular neighbourhoods for their rather low review scores. Charlestown, West Roxbury, Hyde Park and Jamaica Plain are those which get higher average review scores. We should consider both count and score of reviews when decide which neighbourhood is most popular among customers because some neighbourhood has more property listings than others and some of reviews can be very negative. In this case, we would say Jamaica Plain is popular among customer for its high average review score and large number of reviews.

3.Developing a Model

3.1 Data Choosing

Before we develop our model, we need to talk about our data choosing first. Because there are large amount of data in our cleansed dataset, it's impossible for us to train and test all the data in our model. Also, when our data reaches a limit, there will not be significant improvement of our model results when we increase our amount of data. Therefore, we decided to randomly choose a set of data from our cleansed dataset. In this section, we will explain how we choose the training and testing data.

In our cleansed dataset, there are two tables, one called calendar, which contains label(the adjusted price) and time-related new features, another one called listings, which contains host and house related features. We randomly choose 1000 unique units(listings) with all their time features from calendar data(there are about 270000 rows), and then, inner join it with the listing dataset by listing_id. After this, we got a new dataset which contains both time-related and house related featu res(about 2 millions rows). But this dataset is still too large for us, so we randomly choose 200000 rows of the new dataset we got. One thing to mention that we actually tried several times with different amount of data when we do sampling. For example, when we randomly choose unique units(listings) from calendar data, we tried 500 – 1000 unique units(listings), and when we do randomly choose rows from the new dataset, we tried 100000 – 200000 rows.

3.2 Developing a Model

In this section, we develop the tools and techniques necessary for a model to make a prediction.

We use Xgboost model to train and test our dataset.

Implementation: Shuffle and Split Data

we take the housing dataset and split the data into training and testing subsets. Typically, the data is also shuffled into a random order when creating the training and testing subsets to remove any bias in the ordering of the dataset.

- Use train_test_split from sklearn.model_selection to shuffle and split the features and prices data into training and testing sets.
  - Split the data into 70% training and 30% testing.
  - Set the random_state for train_test_split to a value of our choice. This ensures results are consistent.
- Assign the train and testing splits to train_X, test_X, train_y, and test_y.

Evaluating Model Performance

In this final section of the project, we will construct a model and make a prediction on the client's feature set using an optimized model from fit_model.

Perfomance Metric

The R2 score is the proportion of the variance in the dependent variable that is predictable from the independent variable. In our model, we use R2 as well as the MSE to evaluate our predicting results. Here are the R squared values for our model.

```
## the explained_variance_score of test dataset
predictions = my_model.predict(test_X)
print(explained_variance_score(predictions,test_y))
```

```
0.8445081914640743
```

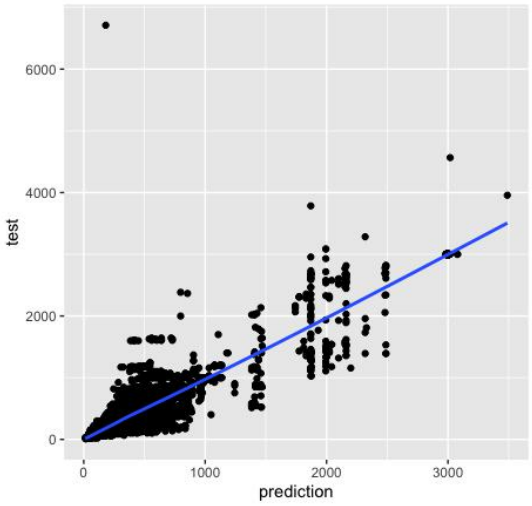
```
## the explained_variance_score of train dataset
predictions = my_model.predict(train_X)
print(explained_variance_score(predictions,train_y))
```

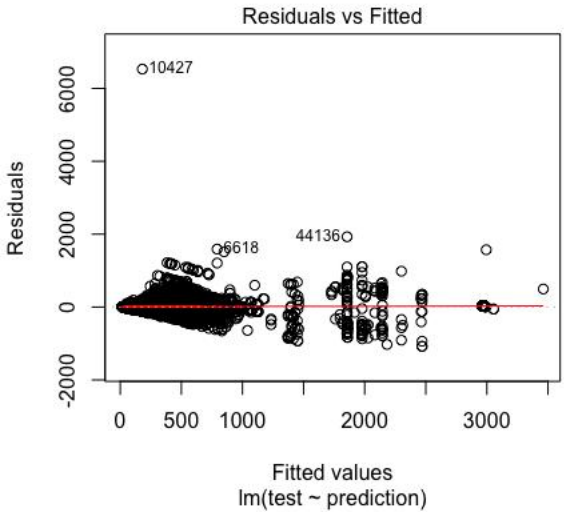
```
0.8774047833438765
```

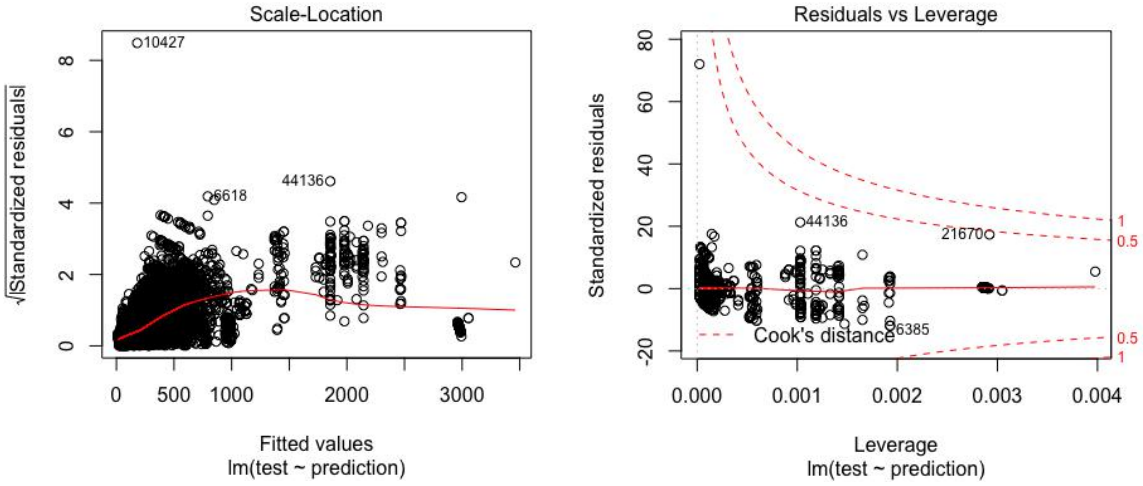
From the result, we can know that, the explained variance score of train dataset is about 87.7%,

While the explained variance score of test dataset is about 84.5%, which are quite high. Also, there is no huge difference between the two scores, which indicates that there's no overfitting problem.

Visualization





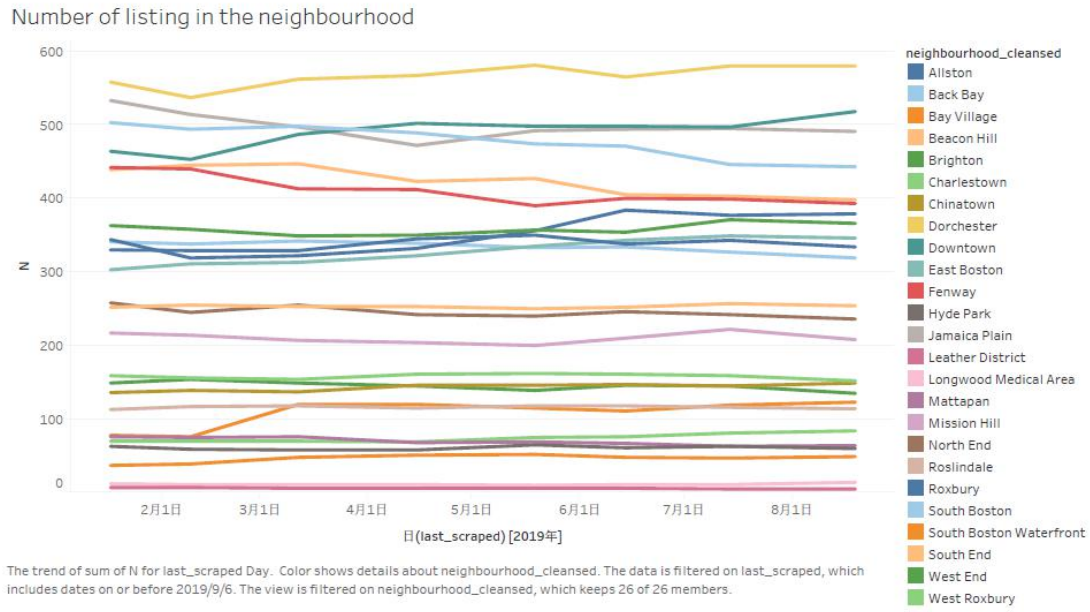


From graph 1, the fitted values and test values fit very well. Graph 2 shows that there is no correlation between fitted values and residuals, which indicates that there's no omitted variable bias in our model. Graph 3 indicates the same information like graph 1. And in our last graph, there's no points that Cook distance are larger than 1, showing that no obvious outliers are in our model. Based on the analysis, we reach the conclusion that our model is reliable and precise.

4.Insight and Recommendations

Insight1: More and more listings in the greater Boston area

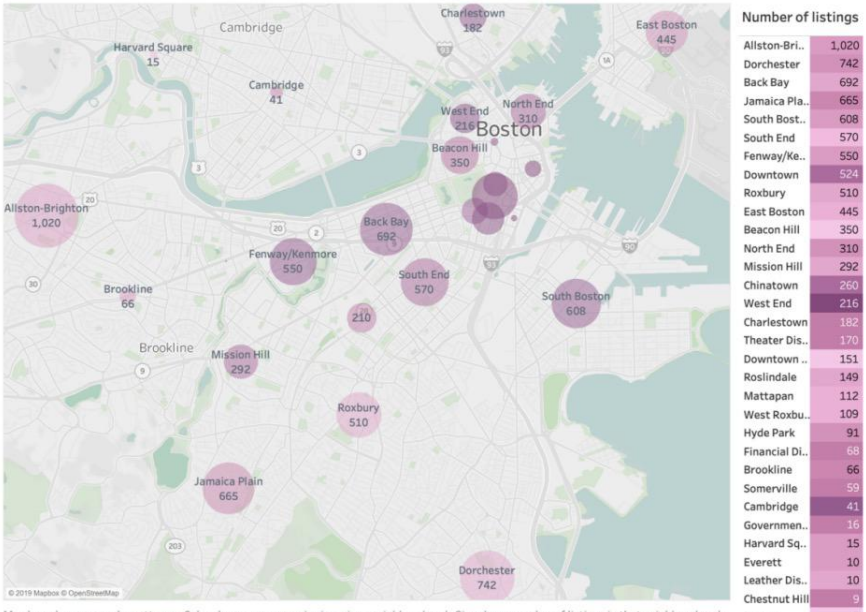
As we can see in the graph below, most of the neighborhood saw an increase of listing number. For example, Dorchester has the largest number of listing during the period and increase about 50 listing in this period. Allston has the largest number of increase of listing, an increase of about 100 listings.



The trend of sum of N for last_scraped Day. Color shows details about neighbourhood_cleansed. The data is filtered on last_scraped, which includes dates on or before 2019/9/6. The view is filtered on neighbourhood_cleansed, which keeps 26 of 26 members.

Insight2: Airbnb covered larger area during the period

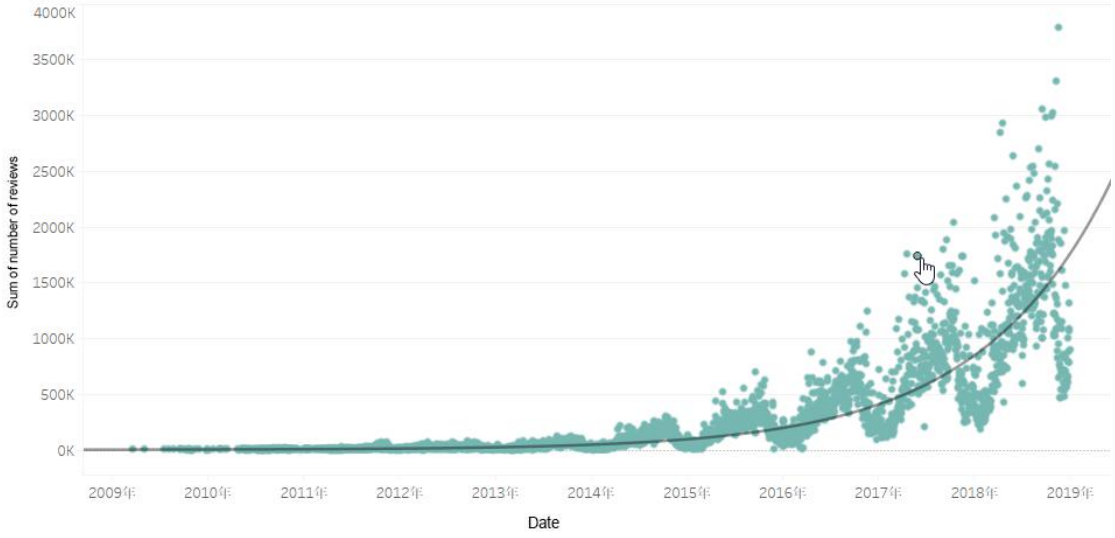
We delighted to see Airbnb covered larger area in Boston during the period with more listing. It makes the neighborhood more prosperous, so more and more people want to travel to the Boston.



Map based on geography patterns. Color shows average price in a given neighbourhood. Size shows number of listings in that neighbourhood. The marks are labeled by neighbourhood name and listing number.

Insight3: More and more people choose Airbnb

From the given dataset, we find that using Airbnb become more and more popular in greater Boston area from February to September,2019. As more and more people choose Airbnb, the number of reviews increases exponentially and the trend line indicates that the growth are likely to continue in following years.



Recommendations for Airbnb:

1. Give users the ability to set time perimeters around the pricing of listings. For example, a host's listing regularly goes for $95/night, but perhaps if I always want it filled, I'm willing to drop it to $70 3 days before the applicable day, $50 1 days before, and so on. You could allow users to add time-based pricing models so as to ensure minimal management of listings with maximum return and success in filling listings.
2. Do away with the sideways-scrolling calendar functionality for users with multiple listings. The calendar UI is better, so at least figure out a way to overlay two listings onto one calendar or show multiple calendars on one page (side-by-side or one on top of the other).
3. Usually listing near travel spotlights thend to be booked. Recent years Airbnb has launched Experience section for guests to explore places near there stay. They coyld also launched Airbnb Car: I pay an extra fee and the service gets my rental car (or Zipcar), takes care of picking me up at airport, etc. This will encourage people to spend more on this single app instead of look at several different website for their all travel needs.

Recommendations for hosts in Airbnb:

From part 1.1's statistical description and data visualization, we can find that variables such as 'Host_is_superhost', 'host_identification_verification', 'Host_profile_picture' can significantly affect house price. In order to rent out their houses in a high price, hosts should try their best to have verified identification and provide their profiles on the websites, which can let consumers trust them and they are more likely to rent their houses. Other factors, like exact location, can slightly affect house prices, and it is better for hosts to post this kind of information as well.

For different rooms structures, more rooms a house has, the higher its price is, which makes sense in the real world.

For cancellation policy price, property type price and room type price, the higher these kind of prices, the higher the house price. Thus, there is some kinds of relationship between these variables.

If hosts want their houses become top_100 houses in the market, they should try their best to become superhost, provides customers with a very flexible cancellation policy, offer them private rooms instead of hotel rooms or shared rooms. If hosts do not have a picture of house profiles, they will be highly unlikely to lose competition in the market. What is more, if hosts have identity verification and can provide instant bookable rooms for customers, they also have chances to get into the top_100.

Finally, among all the variables we list above, whether hosts require guest verification or not will significantly affect both the price and the probability of becoming top_100. If hosts want to become competitive in the market, they should not ask guests for their phone verification.

5.Appendix

(1) R code for cleaning the dataset 'listings_details'

```
# preliminaries                                                                    # ####
rm(list=ls()); gc();graphics.off()
library(data.table)
library(stringr)
PATH  <- 'D:/Brandeis/data competition/data'
setwd(PATH)
e0 <- list()

# load data                                                                        # ####
listings_details <- fread("listings_details.csv")
calendar <- fread("calendar.csv")
#head(listings_details)
str(listings_details)
colname <- colnames(listings_details)
length(colname)

e0$listings_details <- listings_details
# drop variables                                                                   # ####
# drop columns with no variation
col_variation <- sapply(listings_details, function(x)return(length(unique(x))))
col_variation[col_variation == 1]
listings_details[, experiences_offered          := NULL]
listings_details[, thumbnail_url                := NULL]
listings_details[, medium_url                   := NULL]
listings_details[, xl_picture_url               := NULL]
listings_details[, host_acceptance_rate         := NULL]
listings_details[, neighbourhood_group_cleansed := NULL]
listings_details[, country_code                 := NULL]
listings_details[, country                      := NULL]
listings_details[, has_availability             := NULL]
listings_details[, is_business_travel_ready     := NULL]
# drop url
listings_details[, listing_url         := NULL]
listings_details[, picture_url         := NULL]
listings_details[, host_url            := NULL]
listings_details[, host_thumbnail_url  := NULL]
listings_details[, host_picture_url    := NULL]

# drop text
listings_details[, name                    := NULL]
listings_details[, summary                 := NULL]
listings_details[, space                   := NULL]
listings_details[, description             := NULL]
listings_details[, neighborhood_overview   := NULL]
listings_details[, notes                   := NULL]
listings_details[, transit                 := NULL]
listings_details[, access                  := NULL]
listings_details[, interaction             := NULL]
listings_details[, house_rules             := NULL]
listings_details[, host_name               := NULL]
listings_details[, host_about              := NULL]
listings_details[, host_verifications      := NULL]
listings_details[, amenities               := NULL]
# drop meaningless value to us
listings_details[, host_location         := NULL]
listings_details[, host_neighbourhood    := NULL]
listings_details[, street                := NULL]
listings_details[, neighbourhood         := NULL]
listings_details[, city                  := NULL]
listings_details[, state                 := NULL]
listings_details[, smart_location        := NULL]
listings_details[, license               := NULL]
listings_details[, jurisdiction_names    := NULL]
listings_details[, weekly_price          := NULL]
listings_details[, monthly_price         := NULL]
listings_details[, first_review          := NULL]
listings_details[, last_review           := NULL]
listings_details[, market                := NULL]
listings_details[, calendar_updated      := NULL]
listings_details[, calendar_last_scraped := NULL]
listings_details[, last_scraped          := NULL]

e0$listings_details <- listings_details

#drop same column
setdiff(listings_details$host_total_listings_count, listings_details$host_listings_count)
listings_details[, host_total_listings_count := NULL]

# drop column that has lots of NAs
listings_details[, square_feet               := NULL]
```

```r
# drop rows
scraped <- unique(listings_details$last_scraped)
vec_scr <- vector()
for (i in 1:length(unique(scraped))){
  vec_scr[i] <- length(unique(listings_details$id[listings_details$last_scraped==scraped[i]]))
}
listings_details <- listings_details[listings_details$last_scraped != "10/11/19"]
listings_details <- listings_details[listings_details$last_scraped != "7/15/19"]
listings_details <- listings_details[listings_details$last_scraped != "5/24/19"]
listings_details$last_scraped <- as.Date(listings_details$last_scraped, "%m/%d/%y")

# clean data                                                                    # ####
# id
listings_details$id <- as.numeric(listings_details$id)

# date
listings_details$last_scraped <- as.Date(listings_details$last_scraped, "%m/%d/%y")
listings_details$host_since <- as.Date(listings_details$host_since, "%m/%d/%y")


# "t"&"f" >> 1 & 0
vec_tf <- c("host_is_superhost", "host_has_profile_pic", "host_identity_verified","is_location_exact",
            "requires_license","instant_bookable","require_guest_profile_picture",
            "require_guest_phone_verification")
for (i in 1:length(vec_tf)){
  vec_1 <- listings_details[[vec_tf[i]]]
  vec_1[vec_1 == "t"] <- 1
  vec_1[vec_1 == "f"] <- 0
  vec_1[vec_1 == ""]  <- NA
  listings_details[[vec_tf[i]]] <- as.numeric(vec_1)
}

# $price
vec_price <- c("price","security_deposit","cleaning_fee","extra_people")
for (i in 1:length(vec_price)){
  vec_3 <- listings_details[[vec_price[i]]]
  vec_3 <- str_replace_all(vec_3, fixed("$"), "")
  vec_3 <- str_replace_all(vec_3, fixed(".00"), "")
  vec_3 <- str_replace_all(vec_3, fixed(","), "")
  vec_3[vec_3 == ""]  <- NA
  listings_details[[vec_price[i]]] <- as.numeric(vec_3)
}

# host response rate
host_response_rate <- listings_details$host_response_rate
host_response_rate <- str_replace_all(host_response_rate, fixed("%"), "")
listings_details$host_response_rate <- as.numeric(host_response_rate)



# reduce NAs
vec_na <- c("host_since","host_is_superhost","host_listings_count","host_has_profile_pic",
            "host_identity_verified","first_review","security_deposit","cleaning_fee")
for (j in 1:length(vec_na)){
  na_id <- unique(listings_details$id[which(is.na(listings_details[[vec_na[j]]]))])
  for (i in 1:length(na_id)){
    need <- unique(listings_details[[vec_na[j]]][listings_details$id == na_id[i]])
    if(length(need)==2){
      listings_details[[vec_na[j]]][listings_details$id == na_id[i]][is.na(listings_details[[vec_na[j]]]
    }
  }
}
# host year
listings_details$host_since <- as.Date(listings_details$host_since, "%m/%d/%y")
host_years <- round(difftime(strptime("2019-10-31", "%Y-%m-%d"),listings_details$host_since,units='days')/365,2)
listings_details$host_years <- as.numeric(host_years)
listings_details[, host_since           := NULL]
listings_details$host_response_time[listings_details$host_response_time == ""] <- "NA"

# outlier
vec_outlier <- c("host_listings_count","price","security_deposit","cleaning_fee","guests_included",
                 "extra_people","minimum_nights","maximum_nights","minimum_minimum_nights",
                 "maximum_minimum_nights","minimum_maximum_nights","maximum_maximum_nights",
                 "minimum_nights_avg_ntm","maximum_nights_avg_ntm","number_of_reviews",
                 "number_of_reviews_ltm")
for (i in 1:length(vec_outlier)){
  q1  <- quantile(listings_details[[vec_outlier[i]]],0.01,na.rm = TRUE)
  q99 <- quantile(listings_details[[vec_outlier[i]]],0.99,na.rm = TRUE)
  listings_details[[vec_outlier[i]]][listings_details[[vec_outlier[i]]] < q1] <- q1
  listings_details[[vec_outlier[i]]][listings_details[[vec_outlier[i]]] > q99] <- q99
}

# fill NAs
vec_na_fill <- c("host_response_rate","security_deposit","cleaning_fee",
                 "review_scores_accuracy","review_scores_cleanliness","review_scores_checkin",
                 "review_scores_communication","review_scores_location","review_scores_value",
                 "reviews_per_month")
for (i in 1:length(vec_na_fill)){
  listings_details[[vec_na_fill[i]]][is.na(listings_details[[vec_na_fill[i]]])] <- mean(listings_details[[vec_na_
}
```

```
length(listings_details$host_identity_verified[listings_details$host_identity_verified==0])
length(listings_details$host_identity_verified[listings_details$host_identity_verified==1])
listings_details$host_identity_verified[is.na(listings_details$host_identity_verified)] <- 0

listings_details <- listings_details[!is.na(listings_details$beds)]
listings_details <- listings_details[!is.na(listings_details$bedrooms)]
listings_details <- listings_details[!is.na(listings_details$bathrooms)]

#catergory
#zip_has
zip <- listings_details$zipcode
zip <- str_replace_all(zip, fixed("MA "), "")
zip[zip == "02108 02111"] <- "2111"
zip[zip == "02446-4570"] <- "2446"
zip[zip == ""] <- NA
zip <- as.character((as.numeric(zip)))
unique(zip)
zip_has <- listings_details[ ,.(zip_has=.N), by=zipcode]
listings_details <- merge(listings_details,zip_has, by='zipcode')
listings_details[, zipcode := NULL]
#host_response_price
host_response_time <- listings_details$host_response_time
host_response_price <- listings_details[ ,.(host_response_price=mean(price)), by=host_response_time]
listings_details <- merge(listings_details,host_response_price, by="host_response_time")
listings_details[, host_response_time := NULL]

#neighbourhood_cleansed_price
neighbourhood_cleansed <- listings_details$neighbourhood_cleansed
neighbourhood_cleansed_price <- listings_details[ ,.(neighbourhood_cleansed_price=mean(price)), by=neighbourho
listings_details <- merge(listings_details,neighbourhood_cleansed_price, by="neighbourhood_cleansed")
listings_details[, neighbourhood_cleansed := NULL]

#property_type_price
property_type <- listings_details$property_type
property_type_price <- listings_details[ ,.(property_type_price=mean(price)), by=property_type]
listings_details <- merge(listings_details,property_type_price, by="property_type")
listings_details[, property_type := NULL]
#room_type_price
room_type <- listings_details$room_type
room_type_price <- listings_details[ ,.(room_type_price=mean(price)), by=room_type]
listings_details <- merge(listings_details,room_type_price, by="room_type")
listings_details[, room_type := NULL]
#bed_type_price
bed_type <- listings_details$bed_type
bed_type_price <- listings_details[ ,.(bed_type_price=mean(price)), by=bed_type]
listings_details <- merge(listings_details,bed_type_price, by="bed_type")
listings_details[, bed_type := NULL]
#cancellation_policy_price
cancellation_policy <- listings_details$cancellation_policy
cancellation_policy_price <- listings_details[ ,.(cancellation_policy_price=mean(price)), by=cancellation_policy]
listings_details <- merge(listings_details,cancellation_policy_price, by="cancellation_policy")
listings_details[, cancellation_policy := NULL]

str(listings_details)
summary(listings_details)

write.csv(listings_details, file = "listings_details_nice.csv")
```

(2) R code for cleaning the dataset 'calendar'

```
## getting new table calendar_new and adding new features
calendar_new = calendar[,.(listing_id , date, adjusted_price)]
calendar_new = calendar_new[,.(avg_adjusted_price=mean(adjusted_price,na.rm=TRUE)), by=.(listing_id,date)]
calendar_new$month_day = strftime(calendar_new$date, format='%m-%d')
calendar_new$is_weekday = 0
calendar_new[weekday %in% c('Friday','Saturday')]$is_weekday = 1
calendar_new$is_travel_month = 0
calendar_new$months = as.numeric(calendar_new$months)
calendar_new[months %in% c(3,4,7,8,9)]$is_travel_month = 1
holiday <- c(seq(from = as.Date("2019-12-23","%Y-%m-%d"), to = as.Date("2019-12-26","%Y-%m-%d"), by = "days"),
             seq(from = as.Date("2019-11-27","%Y-%m-%d"), to = as.Date("2019-11-29","%Y-%m-%d"), by = "days"),
             as.Date("2019-07-04","%Y-%m-%d"),as.Date("2019-09-02","%Y-%m-%d"),
             seq(from = as.Date("2020-12-23","%Y-%m-%d"), to = as.Date("2020-12-26","%Y-%m-%d"), by = "days"),
             seq(from = as.Date("2020-11-27","%Y-%m-%d"), to = as.Date("2020-11-29","%Y-%m-%d"), by = "days"),
             as.Date("2020-07-04","%Y-%m-%d"),as.Date("2020-09-02","%Y-%m-%d"),
             seq(from = as.Date("2019-04-11","%Y-%m-%d"), to = as.Date("2019-04-16","%Y-%m-%d"), by = "days"),
             seq(from = as.Date("2020-04-11","%Y-%m-%d"), to = as.Date("2020-04-16","%Y-%m-%d"), by = "days"),
             seq(from = as.Date("2019-05-17","%Y-%m-%d"), to = as.Date("2019-05-22","%Y-%m-%d"), by = "days"),
             seq(from = as.Date("2020-05-17","%Y-%m-%d"), to = as.Date("2020-05-22","%Y-%m-%d"), by = "days"))

calendar_new$is_holiday = 0
calendar_new[date %in% holiday]$is_holiday  =1
calendar_new_sample = calendar_new[calendar_new$listing_id %in% sample(unique(calendar_new$listing_id),1000
                                                                       ,replace=FALSE)]
calendar_new_sample =  calendar_new_sample[date %in% seq(from = as.Date("2019-02-09","%Y-%m-%d"),
                                                         to = as.Date("2020-03-09","%Y-%m-%d"), by = "days")]

## running and plotting the regression
predict_test = fread(paste(path,'prediction_test.csv', sep=''))
ggplot(predict_test, aes(x=prediction, y= test))+geom_point() +geom_smooth()
reg = lm(test~prediction, predict_test)
plot(reg)
```

(3) R code for plotting

```
remove(list = ls())
library(data.table)
dataset2=fread('D:/Brandeis Homework/data competition/listings_details_nice.csv')
head(dataset2$zip_has)

reg_price_host_response=lm(price~ zip_has,dataset2)
plot(reg_price_host_response)
library(ggplot2)
ggplot(dataset2, aes(x=host_response_rate,y=price))+ geom_point()+
         geom_smooth()

summary(dataset2$security_deposit)
sd(dataset2$security_deposit)
colnames(dataset2)



summary(dataset2$host_response_rate)
sd(dataset2$host_response_rate)

summary(dataset2$availability_365)
sd(dataset2$availability_365)

summary(dataset2$availability_60)
sd(dataset2$availability_60)

summary(dataset2$availability_30)
sd(dataset2$availability_30)

summary(dataset2$availability_60)
sd(dataset2$availability_60)

summary(dataset2$host_response_price)
sd(dataset2$host_response_price)

summary(dataset2$host_listings_count)
sd(dataset2$host_listings_count)

summary(dataset2$zip_has)
sd(dataset2$zip_has)

summary(dataset2$review_scores_value)
sd(dataset2$review_scores_value)

summary(dataset2$cancellation_policy_price)
sd(dataset2$cancellation_policy_price)

summary(dataset2$bed_type_price)
sd(dataset2$bed_type_price)


dataset2$property_type_price
ggplot(dataset2, aes(x=neighbourhood_cleansed_price,y=price))+ geom_point()+
         geom_smooth()



qnt <- quantile(dataset2$price, probs=c(.25, .75), na.rm = T)
caps <- quantile(dataset2$price, probs=c(.05, .95), na.rm = T)
H <- 1.5 * IQR(dataset2$price, na.rm = T)
dataset2$price[dataset2$price < (qnt[1] - H)] <- caps[1]
dataset2$price[dataset2$price > (qnt[2] + H)] <- caps[2]

boxplot(price~host_is_superhost,
        data=dataset2,
        main="Relationship between Price and Superhost",
        xlab="Host_is_Superhost",
        ylab="Price",
        col="orange",
        border="brown")
dataset2$host_identity_verified

dataset2$is_location_exact
boxplot(price~require_guest_phone_verification,
        data=dataset2,
        main="Relationship between Price and guest_phone_verification",
        xlab="Guest Phone Verification",
        ylab="Price",
        col="orange",
        border="brown")

dataset2$require_guest_phone_verification

plot(dataset2$price~dataset2$host_response_rate, pch=16, cex=0.6)

dataset2$cancella
plot <- ggplot(data=dataset2, aes(x=as.factor(host_response_price), y=dataset2[["price"]])) +
  geom_boxplot(fill="lightblue") + labs(x = "host_response_price", y = "price",
                                        title = ('Relationship between host_response_price and price')) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
print(plot)

summary(dataset2$property_type_price)
sd(dataset2$property_type_price)

unique(dataset2$neighbourhood_cleansed_price)
```

```
summary(dataset2$host_response_price)
sd(dataset2$host_response_price)

dataset2$bedrooms
library(dplyr)
temp1=dataset2%>%group_by(beds)%>%summarise(avg_price=mean(price))
head(temp1)

plot <- ggplot(data=temp1, aes(x=as.factor(beds), y=temp1[["avg_price"]])) +
  geom_boxplot(fill="lightblue") + labs(x = "beds", y = "avg_price",
                                        title = ('Relationship between bedrooms and avg_price')) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
print(plot)


p<-ggplot(data=temp1, aes(x=beds, y=avg_price)) +
  geom_bar(stat="identity", fill="steelblue")+
  theme_minimal()

temp <- subset(dataset, is_top_100 == 1)
temp=temp%>%group_by(is_top_100,neighbourhood_cleansed)%>%summarise(density = n()/nrow(.))
temp1=subset(dataset,is_top_100==0)
temp1=temp1%>%group_by(is_top_100,neighbourhood_cleansed)%>%summarise(density = n()/nrow(.))
temp2 <- rbind(temp, temp1)
head(temp)
plot <- ggplot(data=temp2, aes(x=as.factor(temp2[["neighbourhood_cleansed"]]), y=density, fill=as.factor(is_top_
  geom_bar(position = 'dodge', stat='identity') + labs(fill = "is_top_100", x = "neighbourhood_cleansed",
                                                       title = paste("neighbourhood_cleansed", " relative densit
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
print(plot)
```

(4) Model Coding

```
import pandas as pd
import matplotlib.pyplot as plt
import xgboost
import math
from scipy.stats import pearsonr
from sklearn.linear_model import LinearRegression
from sklearn import model_selection, tree, linear_model
from sklearn.model_selection import train_test_split
from sklearn.metrics import explained_variance_score
from xgboost import XGBRegressor
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import Imputer

## getting data
data_listing= pd.read_csv('/Users/tjmask/Desktop/data competition/listings_details_nice.csv', delimiter=",")
data_listing = data_listing.drop(columns=["Unnamed: 0", 'scrape_id',"latitude",'longitude',"host_id"])
data_listing.head()

data_calendar = pd.read_csv('/Users/tjmask/Desktop/data competition/calendar_sample.csv', delimiter=",")
df_merge = pd.merge(data_calendar, data_listing, how='inner', left_on=data_calendar['listing_id'],
                    right_on=data_listing['id'])
## merging data to get the final training and testing data
df_merge.eval('time_neighbor = is_weekday*neighbourhood_cleansed_price' , inplace=True)
df_merge = df_merge.drop(columns=["Unnamed: 0", "key_0", "date"])
df_random = df_merge.sample(n=180000,replace=False, random_state=12)

## getting features
features = df_random[['is_weekday','is_travel_month','time_neighbor',
       'is_holiday', 'zip_has', 'host_response_price',
       'neighbourhood_cleansed_price', 'property_type_price',
       'room_type_price', 'bed_type_price', 'cancellation_policy_price','host_years',
        'host_response_rate', 'host_is_superhost','host_has_profile_pic', 'host_identity_verified',
       'is_location_exact', 'bathrooms', 'bedrooms',
       'cleaning_fee', 'guests_included',
       'extra_people', 'minimum_nights',
       'availability_60','availability_365',
       'review_scores_accuracy', 'review_scores_cleanliness',
       'review_scores_checkin', 'review_scores_communication',
       'review_scores_location', 'review_scores_value','reviews_per_month',
       'instant_bookable', 'require_guest_profile_picture',
       'require_guest_phone_verification']]
```

```
## getting prices
prices = pd.DataFrame(df_random["avg_adjusted_price"])

## using linear regression model
regr = linear_model.LinearRegression()
X_train, X_test, y_train, y_test = model_selection.train_test_split(features, prices ,test_size=0.3)
regr.fit(X_train, y_train)
print(regr.predict(X_test))
regr.score(X_test,y_test)

## using xgboost model
train_X, test_X, train_y, test_y = train_test_split(features, prices, test_size=0.25)
my_imputer = Imputer()
train_X = my_imputer.fit_transform(train_X)
test_X = my_imputer.transform(test_X)


my_model = XGBRegressor(learning_rate=0.1, n_estimators=550, max_depth=4, min_child_weight=5, seed=0,
                            subsample=0.7, colsample_bytree=0.7, gamma=0.1, reg_alpha=1, reg_lambda=1)
# Add silent=True to avoid printing out updates with each cycle
my_model.fit(train_X, train_y, verbose=False)

## the explained_variance_score of test dataset
predictions = my_model.predict(test_X)
print(explained_variance_score(predictions,test_y))

## the explained_variance_score of train dataset
predictions = my_model.predict(train_X)
print(explained_variance_score(predictions,train_y))
```