REPORT - NYC AIRBNB PRICE PREDICTION CAPSTONE PROJECT

PRATIK

22/05/2020

1. INTRODUCTION:

The project is based on New York City Airbnb dataset. The Aim of the project is to build a model to predict the Rental prices.

The dataset is provided with the information about hosts, geographical availability and all other necessary metrics available.

The goal of the project is to obtain the best fitting model for prediciting prices by taking the parameter RMSE into consideration.

The key steps to be performed ahead are the **Analysis** of the data and further building a **Model** on the basis of the anaklysis.

The **Strucuture** of the dataset is as follows :

Structure :

```
str(data)
```

```
## 'data.frame':    48895 obs. of  16 variables:
##  $ id                            : int  2539 2595 3647 3831 5022 5099 5121 5178 5203 5238 ...
##  $ name                          : Factor w/ 47906 levels "","" 1 Bed Apt in Utopic Williamsburg ",..
##  $ host_id                       : int  2787 2845 4632 4869 7192 7322 7356 8967 7490 7549 ...
##  $ host_name                     : Factor w/ 11453 levels "","" Valéria",..: 4997 4791 2913 6210 5929
##  $ neighbourhood_group           : Factor w/ 5 levels "Bronx","Brooklyn",..: 2 3 3 2 3 3 2 3 3 3 ...
##  $ neighbourhood                 : Factor w/ 221 levels "Allerton","Arden Heights",..: 109 128 95 42
##  $ latitude                      : num  40.6 40.8 40.8 40.7 40.8 ...
##  $ longitude                     : num  -74 -74 -73.9 -74 -73.9 ...
##  $ room_type                     : Factor w/ 3 levels "Entire home/apt",..: 2 1 2 1 1 1 2 2 2 1 ...
##  $ price                         : int  149 225 150 89 80 200 60 79 79 150 ...
##  $ minimum_nights                : int  1 1 3 1 10 3 45 2 2 1 ...
##  $ number_of_reviews             : int  9 45 0 270 9 74 49 430 118 160 ...
##  $ last_review                   : Factor w/ 1765 levels "","2011-03-28",..: 1503 1717 1 1762 1534 1
##  $ reviews_per_month             : num  0.21 0.38 NA 4.64 0.1 0.59 0.4 3.47 0.99 1.33 ...
##  $ calculated_host_listings_count: int  6 2 1 1 1 1 1 1 1 4 ...
##  $ availability_365              : int  365 355 365 194 0 129 0 220 0 188 ...
```

2. DATA EXPLORATORY & ANALYSIS :

2.1 SUMMARY OF THE DATASET:

```
summary(data)
```

```
##        id                                          name
##  Min.   :    2539   Hillside Hotel                  :   18
##  1st Qu.: 9471945   Home away from home             :   17
##  Median :19677284                                   :   16
##  Mean   :19017143   New york Multi-unit building    :   16
##  3rd Qu.:29152178   Brooklyn Apartment              :   12
##  Max.   :36487245   Loft Suite @ The Box House Hotel:   11
##                     (Other)                         :48805
##     host_id                host_name        neighbourhood_group
##  Min.   :     2438   Michael     :  417   Bronx        : 1091
##  1st Qu.:  7822033   David       :  403   Brooklyn     :20104
##  Median : 30793816   Sonder (NYC):  327   Manhattan    :21661
##  Mean   : 67620011   John        :  294   Queens       : 5666
##  3rd Qu.:107434423   Alex        :  279   Staten Island:  373
##  Max.   :274321313   Blueground  :  232
##                      (Other)     :46943
##            neighbourhood      latitude       longitude
##  Williamsburg      : 3920   Min.   :40.50   Min.   :-74.24
##  Bedford-Stuyvesant: 3714   1st Qu.:40.69   1st Qu.:-73.98
##  Harlem            : 2658   Median :40.72   Median :-73.96
##  Bushwick          : 2465   Mean   :40.73   Mean   :-73.95
##  Upper West Side   : 1971   3rd Qu.:40.76   3rd Qu.:-73.94
##  Hell's Kitchen    : 1958   Max.   :40.91   Max.   :-73.71
##  (Other)           :32209
##           room_type         price         minimum_nights    number_of_reviews
##  Entire home/apt:25409   Min.   :    0.0   Min.   :   1.00   Min.   :  0.00
##  Private room   :22326   1st Qu.:   69.0   1st Qu.:   1.00   1st Qu.:  1.00
##  Shared room    : 1160   Median :  106.0   Median :   3.00   Median :  5.00
##                          Mean   :  152.7   Mean   :   7.03   Mean   : 23.27
##                          3rd Qu.:  175.0   3rd Qu.:   5.00   3rd Qu.: 24.00
##                          Max.   :10000.0   Max.   :1250.00   Max.   :629.00
##
##      last_review    reviews_per_month calculated_host_listings_count
##            :10052   Min.   : 0.010    Min.   :  1.000
##  2019-06-23: 1413   1st Qu.: 0.190    1st Qu.:  1.000
##  2019-07-01: 1359   Median : 0.720    Median :  1.000
##  2019-06-30: 1341   Mean   : 1.373    Mean   :  7.144
##  2019-06-24:  875   3rd Qu.: 2.020    3rd Qu.:  2.000
##  2019-07-07:  718   Max.   :58.500    Max.   :327.000
##  (Other)   :33137   NA's   :10052
##  availability_365
##  Min.   :  0.0
##  1st Qu.:  0.0
##  Median : 45.0
##  Mean   :112.8
##  3rd Qu.:227.0
##  Max.   :365.0
```

##

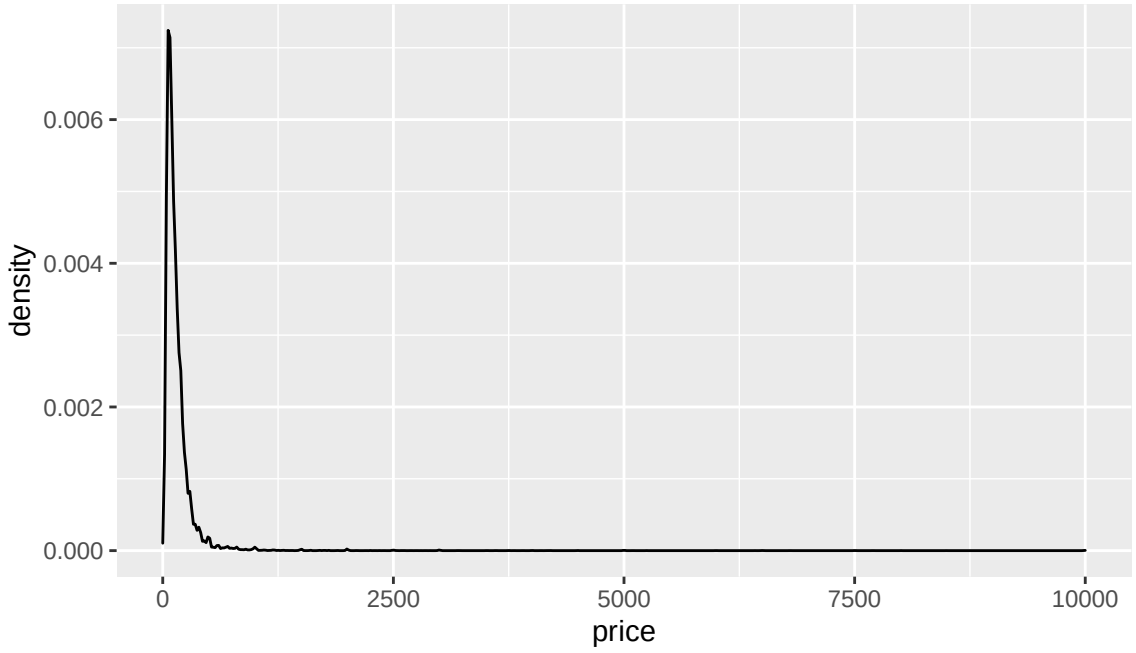
2.2 FIRST 6 ELEMENTS OF THE DATASET :

```
head(data)
```

```
##     id                                              name host_id   host_name
## 1 2539               Clean & quiet apt home by the park    2787        John
## 2 2595                           Skylit Midtown Castle    2845    Jennifer
## 3 3647             THE VILLAGE OF HARLEM....NEW YORK !    4632   Elisabeth
## 4 3831                  Cozy Entire Floor of Brownstone    4869 LisaRoxanne
## 5 5022 Entire Apt: Spacious Studio/Loft by central park    7192       Laura
## 6 5099        Large Cozy 1 BR Apartment In Midtown East    7322       Chris
##   neighbourhood_group neighbourhood latitude longitude       room_type price
## 1            Brooklyn    Kensington 40.64749 -73.97237    Private room   149
## 2           Manhattan       Midtown 40.75362 -73.98377 Entire home/apt   225
## 3           Manhattan        Harlem 40.80902 -73.94190    Private room   150
## 4            Brooklyn  Clinton Hill 40.68514 -73.95976 Entire home/apt    89
## 5           Manhattan   East Harlem 40.79851 -73.94399 Entire home/apt    80
## 6           Manhattan   Murray Hill 40.74767 -73.97500 Entire home/apt   200
##   minimum_nights number_of_reviews last_review reviews_per_month
## 1              1                 9  2018-10-19              0.21
## 2              1                45  2019-05-21              0.38
## 3              3                 0                          NA
## 4              1               270  2019-07-05              4.64
## 5             10                 9  2018-11-19              0.10
## 6              3                74  2019-06-22              0.59
##   calculated_host_listings_count availability_365
## 1                              6              365
## 2                              2              355
## 3                              1              365
## 4                              1              194
## 5                              1                0
## 6                              1              129
```

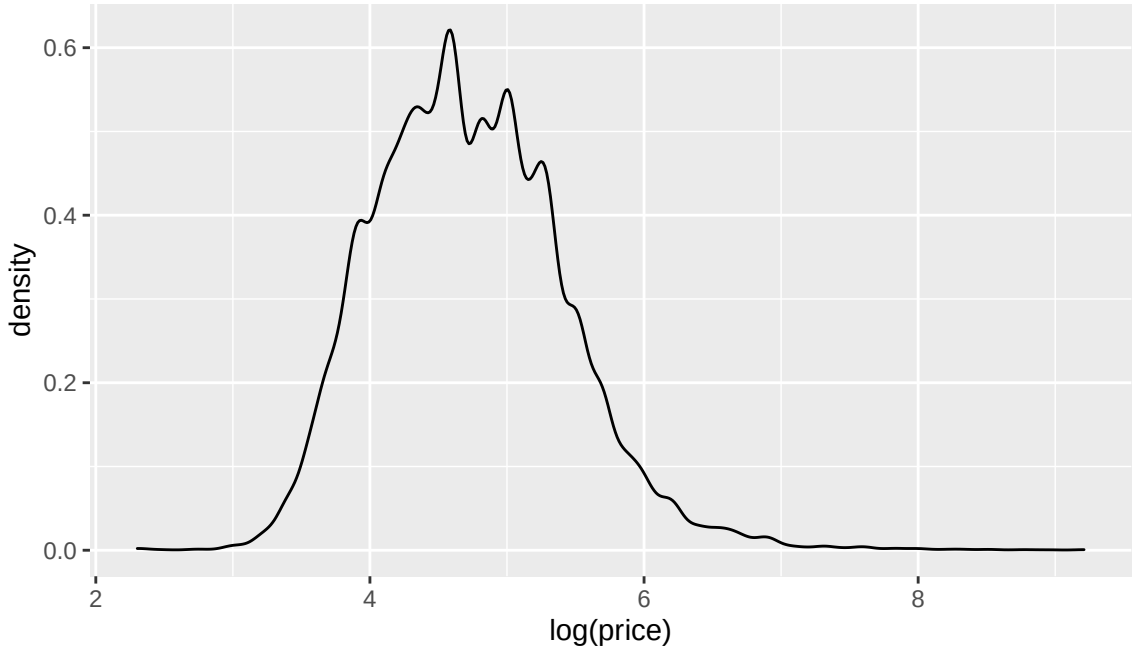
2.3 DISTRIBUTION OF THE PRICES :

The distribution of the data has a very high density for prices in the lower range. Therefore logarithmic scaling of the price data is to be further performed to observed the distribution.



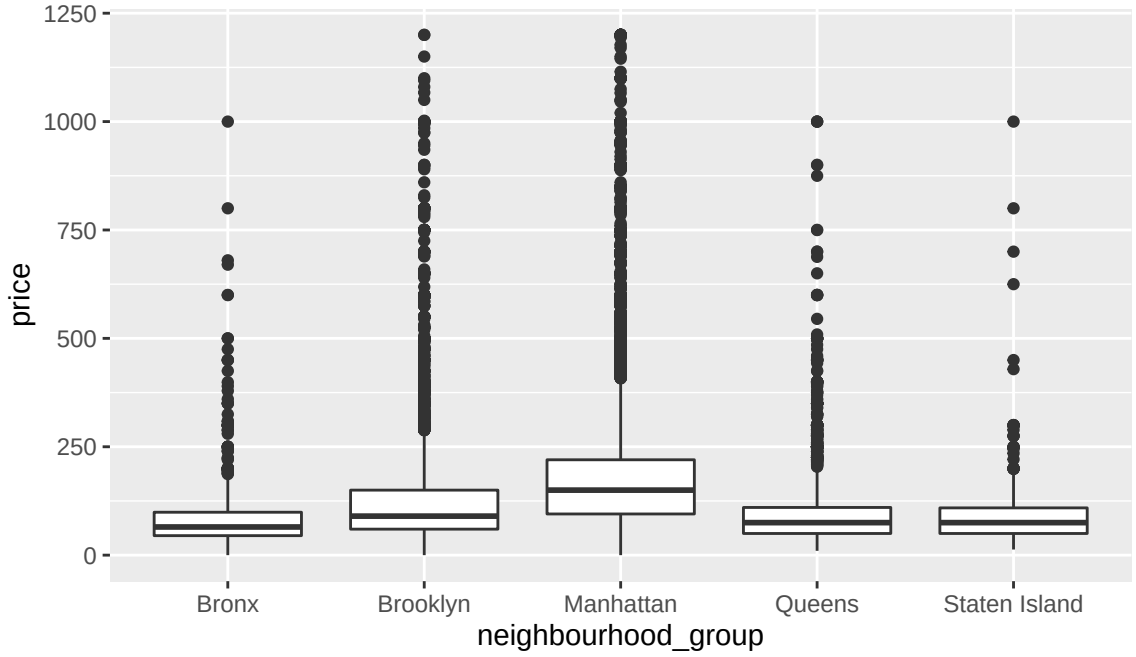
2.4 DISTRIBUTION OF LOG(PRICES) :

```
## Warning: Removed 11 rows containing non-finite values (stat_density).
```



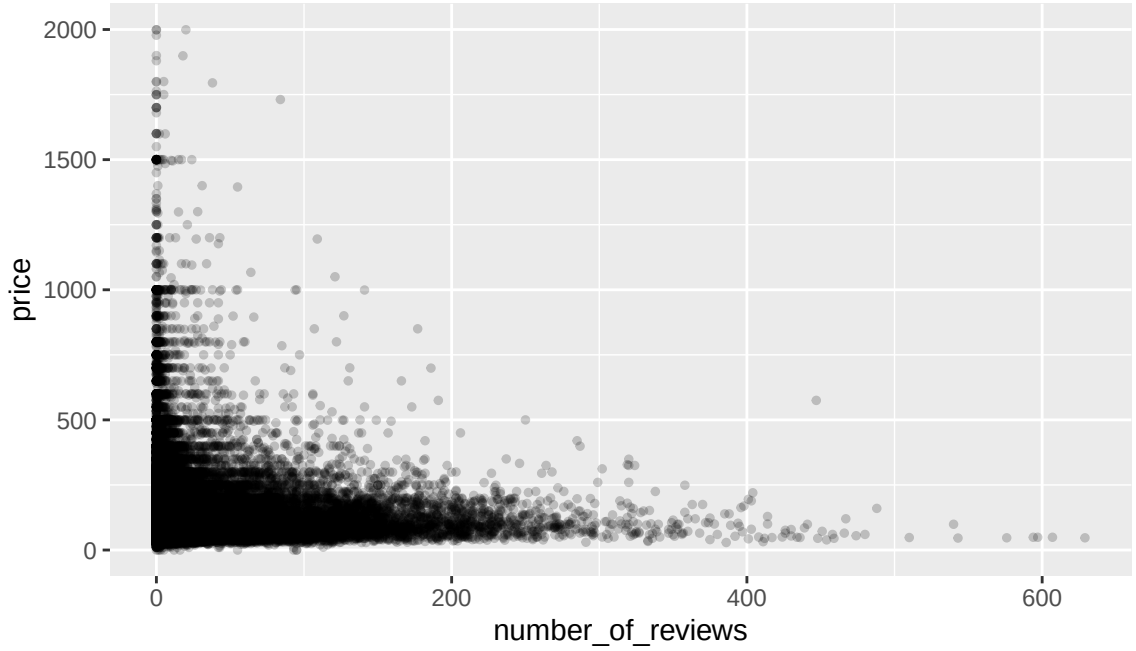
2.5 DISTRIBUTION OF PRICES WITH NEIGHBOURHOOD GROUP :

Removing the price outliers by filtering to contain the prices below 1250, we observe that the price distribution (median,25% & 75% percentile) vary with the neighbourhood groups.



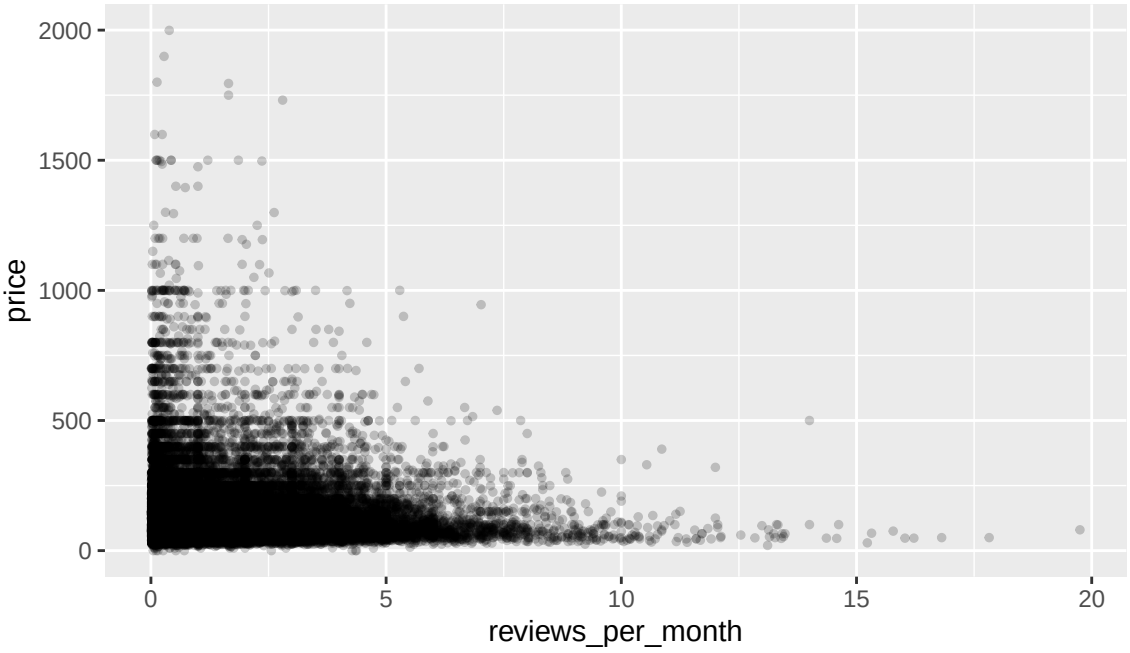
2.6 DISTRIBUTION OF PRICES WITH NUMBER OF REVIEWS :

As evident from the graph, for data set having large number of reviews the price is in the lower range. Lesser the number of reviews, the wider is the price range for the data set

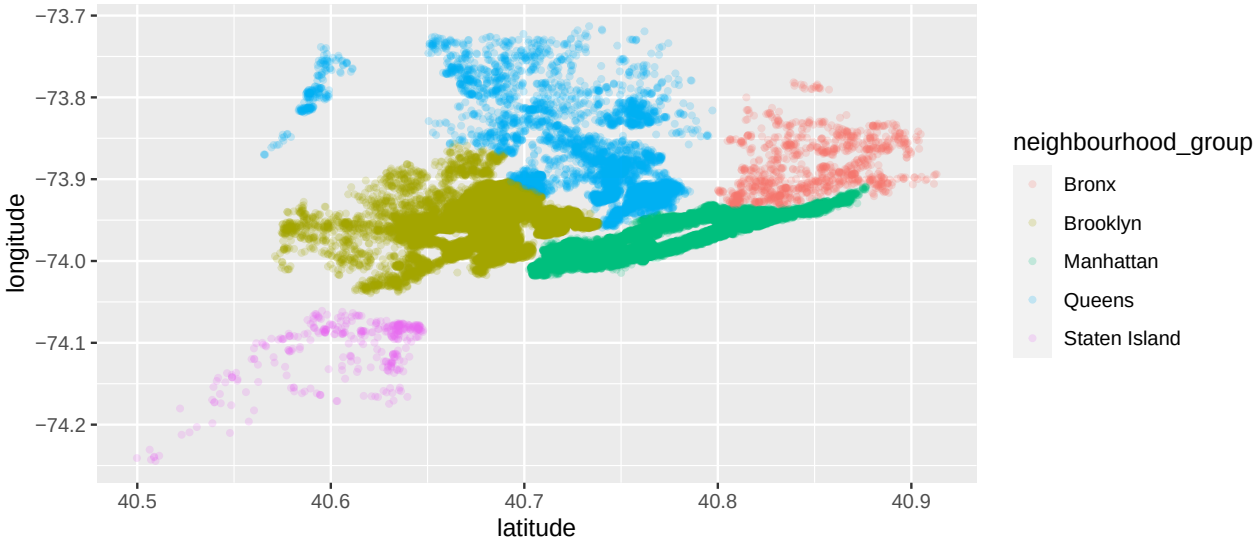


2.6 DISTRIBUTION OF PRICES WITH REVIEWS PER MONTH :

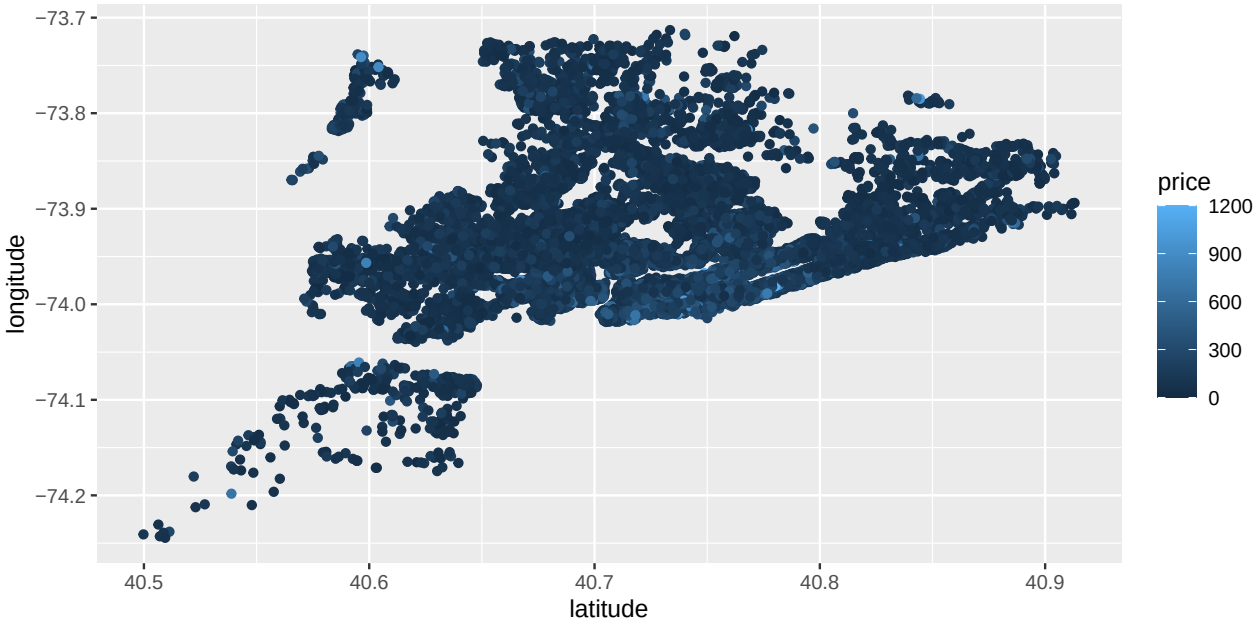
As evident from the graph, for data set having large number of reviews per month, the price is in the lower range. Lesser the number of reviews per month, the wider is the price range for the data set



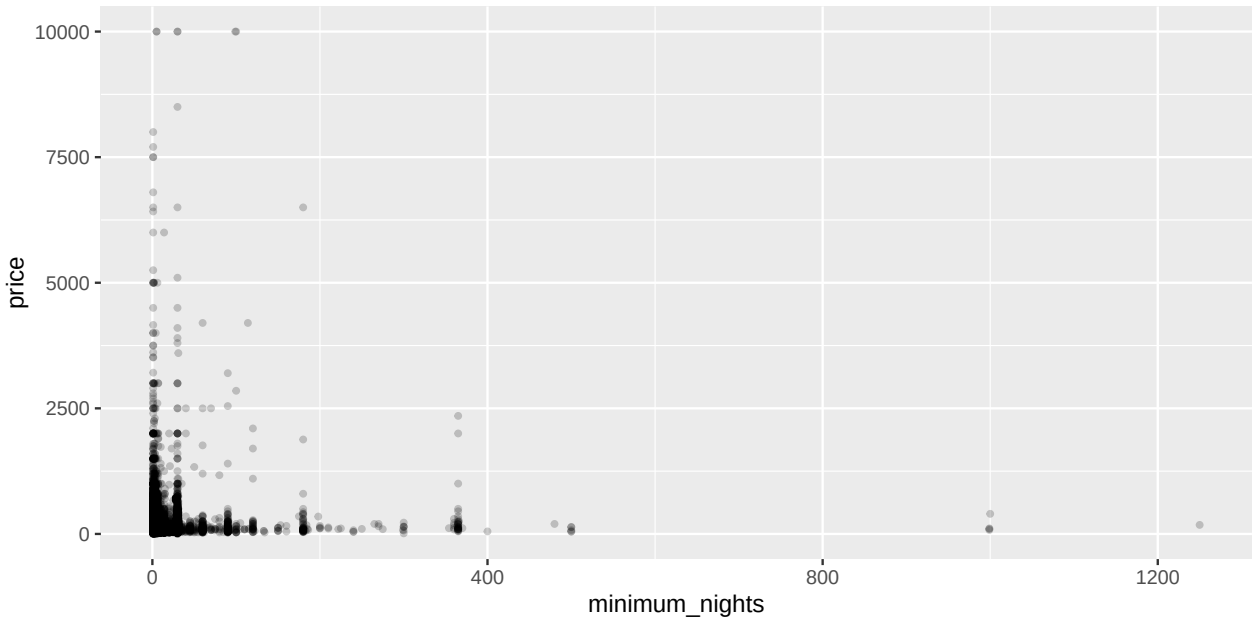
2.7 DISTRIBUTION OF NEIGHBOURHOOD GROUPS WITH LATITUDE AND LONGITUDE :



2.8 DISTRIBUTION OF PRICE WITH LATITUDE AND LONGITUDE :

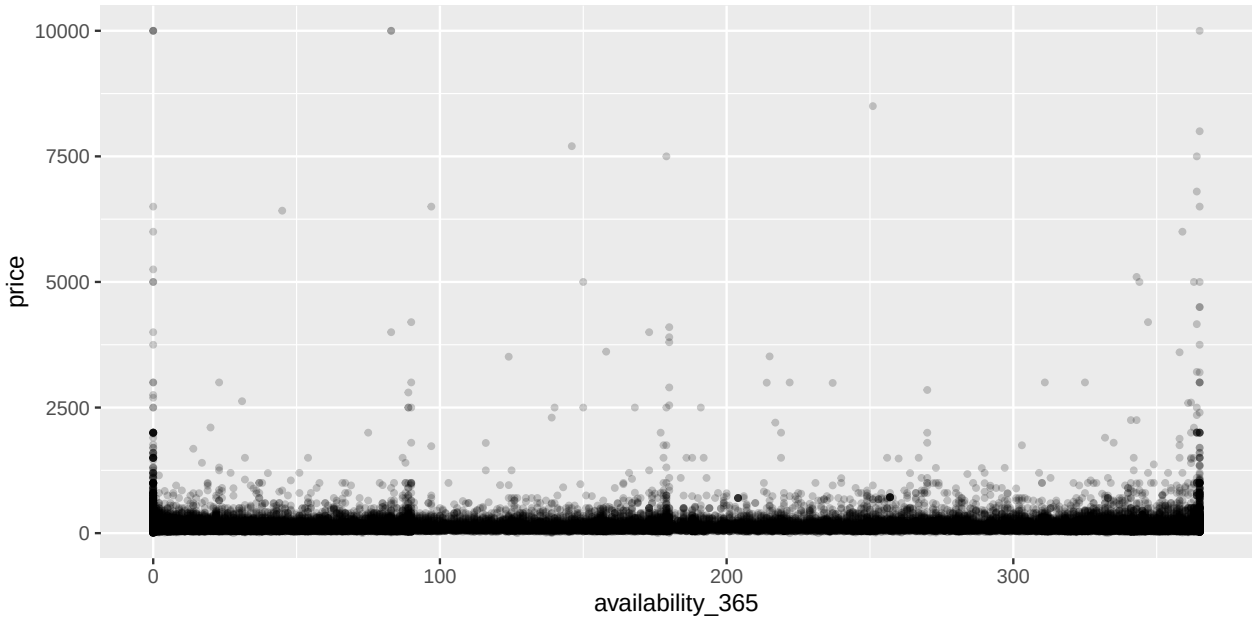


2.9 DISTRIBUTION OF PRICE WITH MINIMUM NIGHTS :

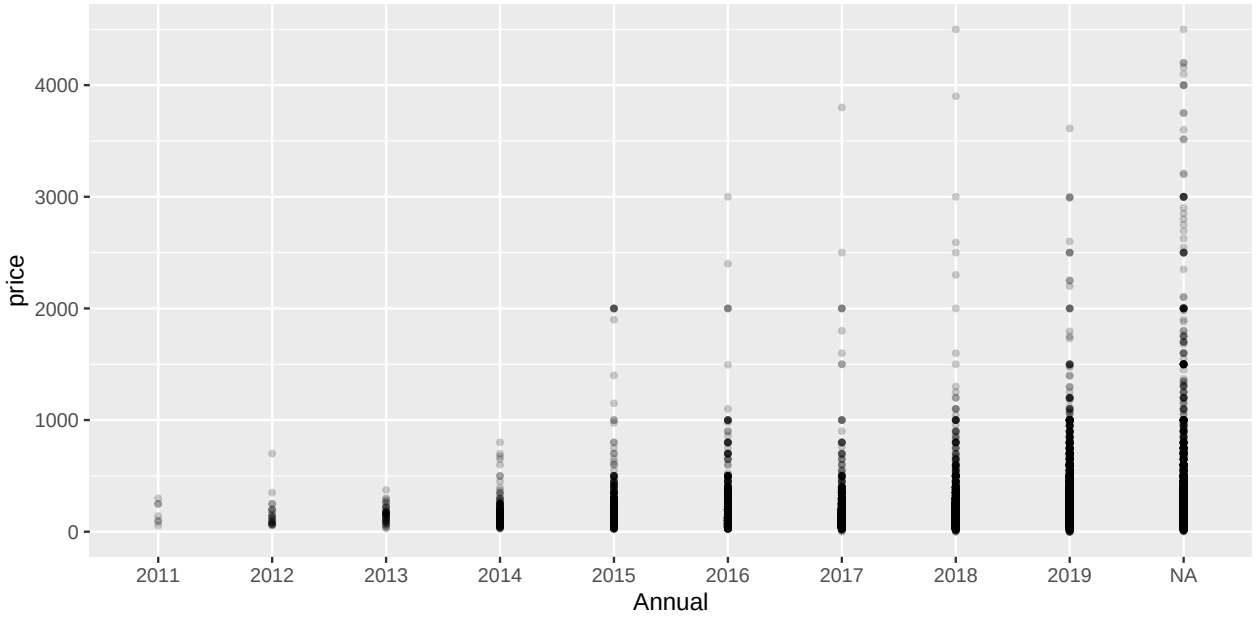


2.10 DISTRIBUTION OF PRICE VS AVAILABLE 365 :

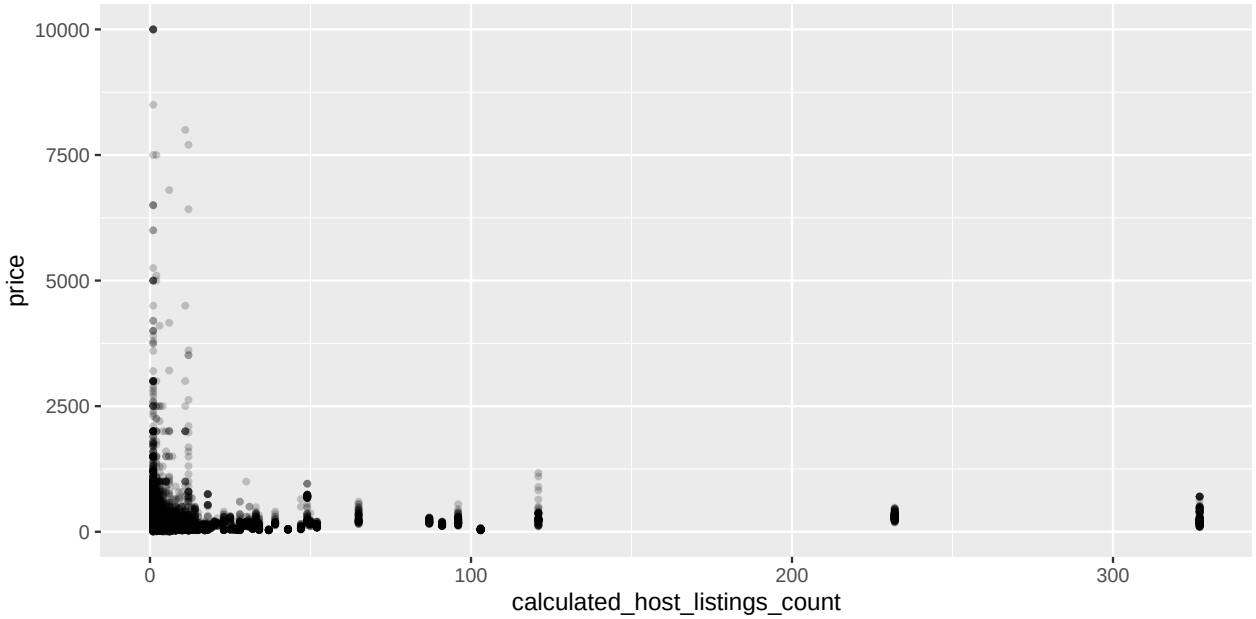
As through the graph below, no meaningful distribution is observed for price vs availability_365



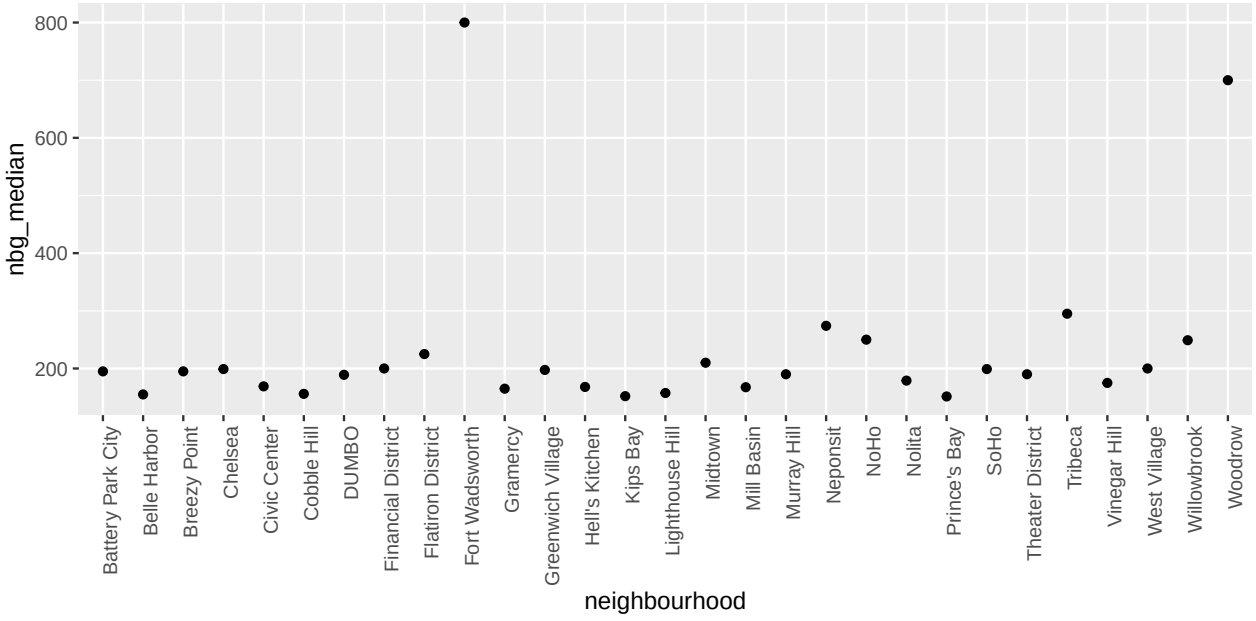
2.11 DISTRIBUTION OF PRICE VS LAST REVIEW YEAR (FEATURE EXTRACTED FROM LAST_REVIEW_DATE) :



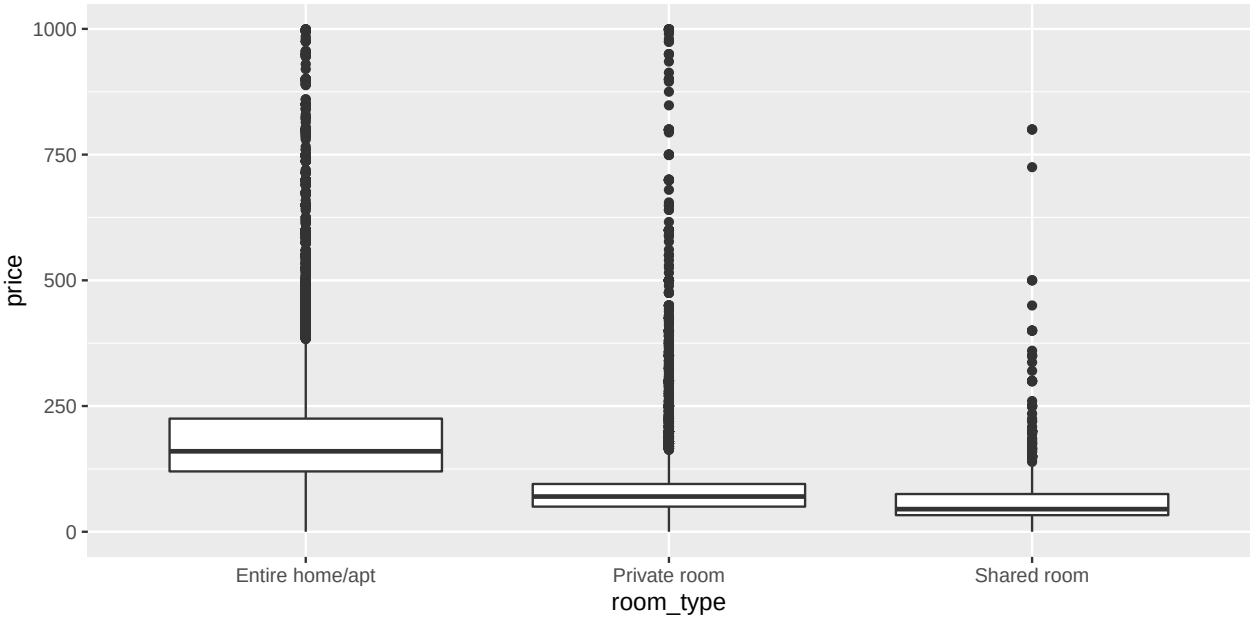
2.12 DISTRIBUTION OF PRICE VS HOST LISTINGS :



2.13 DISTRIBUTION OF PRICE VS NEIGHBOURHOOD MEDIAN (GRAPGH WITH nbg_median>150) :



2.14 DISTRIBUTION OF PRICE VS ROOMTYPE :



3 METHODOLOGY :

After the above analysis of data, Model is built by taking the columns : **latitude**, **longitude**, **neighbourhood_group**, **neighbourhood**, **room_type**, **number_of_reviews**, **calculated_host_listings_count**, **minimum_nights**, **reviews_per_month**, **Annual**

Annual - Contains the year which had the last review, a feature exctracted from **last_review**. Missing Values of the year in the data are filled by the taking the mean of the years and rounding to the nearest year which turns out to be 2018.

The dataset is split into three sets : **Train set** , **Test Set** , **Validation Set** in the **0.6 : 0.2 : 0.2**

3.1 SIMPLE LINEAR REGRESSION MODEL :

A simple linear regression is performed on **Y** = **price** and **X** containing the columns **latitude**, **longitude**, **neighbourhood_group**, **neighbourhood**, **room_type**, **number_of_reviews**, **calculated_host_listings_count**, **minimum_nights**, **reviews_per_month**, **Annual**.

Training is performed on the train_set :

```
Model_1 <- train(data.price ~ ., data = train_set,
                      method = "lm")
```

Summary of the trained model :

```
Model_1$results
```

```
##   intercept     RMSE   Rsquared      MAE   RMSESD RsquaredSD     MAESD
## 1      TRUE 232.9706 0.09713847 73.43469 20.06138 0.01157788 0.9783841
```

Prediction is performed on the test_set :

```
Model_1_predict<-predict(Model_1,test_set)
```

RMSE Error on test_set :

```
RMSE(Model_1_predict,test_set$data.price)
```

```
## [1] 215.6614
```

```
## Warning: `data_frame()` is deprecated as of tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

3.2 LOG LINEAR REGRESSION MODEL OF PRICE :

A log linear regression is performed on **Y = log(1+price)** and **X** containing the columns **latitude**, **longitude**, **neighbourhood_group**, **neighbourhood**, **room_type**, **number_of_reviews**, **calculated_host_listings_count**, **minimum_nights**, **reviews_per_month**, **Annual**.

Training is performed on the train_set :

```
Model_2 <- train(log(1+data.price) ~ ., data = train_set,
                         method = "lm")
```

Summary of the trained model :

```
Model_2$results
```

```
##   intercept      RMSE  Rsquared       MAE      RMSESD  RsquaredSD       MAESD
## 1      TRUE 0.4858173 0.5166043 0.3480623 0.005629482 0.007280804 0.002519849
```

Prediction is performed on the test_set :

```
Model_2_predict<-predict(Model_2,test_set)
```

RMSE Error on test_set :

```
RMSE(exp(Model_2_predict)-1,test_set$data.price)
```

```
## [1] 216.4989
```

3.3 XGBOOST LINEAR MODEL :

XGBoost Linear Algorithm which is designed to be highly efficient and flexible is applied on **train_set** and further tested on **test_set**.

Training is performed on the train_set :

```
Model_3 <- train(data.price ~ ., data = train_set,
                         method = "xgbLinear",
                 tuneGrid = expand.grid(nrounds = 50,lambda = seq(0.1, 0.5, 0.2),alpha =seq(0.1,0.5, 0.
```

Prediction is performed on the test_set :

```
Model_3_predict<-predict(Model_3,test_set)
```

RMSE Error on test_set :

```
RMSE(Model_3_predict,test_set$data.price)
```

```
## [1] 207.2234
```

3.4 DECISION TREE MODEL :

A decison tree algorithm is performed on **Y** = **price** and **X** containing the columns **latitude**, **longitude**, **neighbourhood_group**, **neighbourhood**, **room_type**, **number_of_reviews**, **calculated_host_listings_count**, **minimum_nights**, **reviews_per_month**, **Annual**.

Training is performed on the train_set :

```
MODEL_4 <- train(data.price ~ ., data = train_set,
                 method = "rpart")
```

Summary of the trained model :

```
MODEL_4$results
```

```
##           cp     RMSE   Rsquared      MAE   RMSESD  RsquaredSD    MAESD
## 1 0.01710268 233.4680 0.06829783 76.41937 21.67637 0.020731714 1.520075
## 2 0.01797207 233.2756 0.06762027 76.61295 21.66075 0.018494725 1.533780
## 3 0.05696926 236.9710 0.05226020 85.82577 21.26300 0.005611467 7.168284
```

Prediction is performed on the test_set :

```
MODEL_4_predict<-predict(MODEL_4,test_set)
```

RMSE Error on test_set :

```
RMSE(MODEL_4_predict,test_set$data.price)
```

```
## [1] 220.7402
```

4 RESULTS :

The results obtained by running the above four models by performing training on **train_set** dataset and further testing on the **test_set** dataset are as follows :

method	RMSE
SIMPLE LINEAR REGRESSION MODEL	215.6614
LOG LINEAR REGRESSION MODEL	216.4989
XGBOOST LINEAR MODEL	207.2234
DECISION TREE MODEL	220.7402

Therefore, the RMSE error is least for the **XGBoost Linear Model**.

Applying the **XGBoost Linear Model** on the **validation_set**

Prediction is performed on the validation_set :

```
VALIDATION_predict<-predict(Model_3,validation_set)
```

RMSE Error on validation_set :

```
RMSE(VALIDATION_predict,validation_set$data.price)
```

```
## [1] 129.8694
```

The high RMSE Value are due to the Outliers present in the form of High Rental Price Houses in the dataset. Thus this makes it necessary to calculate an RMSE on dataset by filtering the outliers.

Calculating the RMSE for the validation data_Set excluding the top 5 percent values.

```
index<-validation_set$data.price < quantile(validation_set$data.price,0.95)
VALIDATION_RMSE_<-RMSE(VALIDATION_predict[index],validation_set$data.price[index])
```

RMSE Error for the validation set excluding the outliers

```
tibble(method = "XGBLINEAR BOOST ON VALIDATION_SET AFTER FILTERING OUT OUTLIERS"
       , RMSE = VALIDATION_RMSE_)%>%knitr::kable()
```

method	RMSE
XGBLINEAR BOOST ON VALIDATION_SET AFTER FILTERING OUT OUTLIERS	68.22254

5 CONCLUSIONS :

XgbLinear is the algorithm leading to the least RMSE value for the **test_set**. Further utilization of this algorithm for evalution of the **validation_set** leads to an RMSE error value of :

```
tibble(method = "XGBLINEAR BOOST ON VALIDATION_SET "
       , RMSE =RMSE(VALIDATION_predict,validation_set$data.price))%>%knitr::kable()
```

method	RMSE
XGBLINEAR BOOST ON VALIDATION_SET	129.8694

Also the further analysis of the final model after removing outliers for the **validation_set**, provides a conclusion that the model fits in the better way for the housings after filtering the top 5 percent outliers. This is evident by the RMSE value :

```
tibble(method = "VALIDATION_SET RMSE AFTER FILTERING OUT OUTLIERS"
       , RMSE = VALIDATION_RMSE_)%>%knitr::kable()
```

method	RMSE
VALIDATION_SET RMSE AFTER FILTERING OUT OUTLIERS	68.22254