



UNIVERSITÀ  
DEGLI STUDI  
DI MILANO

---

LA STATALE

DEPARTMENT OF ECONOMICS, MANAGEMENT AND QUANTITATIVE METHODS

UNIVERSITÀ DEGLI STUDI DI MILANO

DATA SCIENCE AND ECONOMICS

## STATISTICAL LEARNING PROJECT

# ANALYSIS AND PREDICTION MODELS FOR NEW YORK CITY AIRBNB

REPORT

*Author:*

Andrea IERARDI

ACADEMIC YEAR 2019-2020

---

# CONTENTS

<b>List of Figures</b>	<b>5</b>
<b>1 Abstract</b>	<b>6</b>
<b>2 Problem Definition and Algorithm</b>	<b>7</b>
2.1 Two main Goals . . . . .	7
2.1.1 Develop predictive models for price . . . . .	7
2.1.2 Define clusters and groups . . . . .	7
2.2 Algorithms . . . . .	8
2.2.1 Linear Regression . . . . .	8
2.2.2 Decison Trees . . . . .	8
2.2.3 Random Forest . . . . .	8
2.2.4 Ranger Random Forest . . . . .	8
2.2.5 Neural Networks . . . . .	8
2.2.6 K-means . . . . .	8
2.2.7 Principal Component Analysis . . . . .	8
<b>3 Experimental Evaluation</b>	<b>9</b>
3.1 Methodology . . . . .	9
3.1.1 Data Inspection . . . . .	9
3.1.2 Data Cleaning and Pre-processing . . . . .	11
3.2 Results . . . . .	13
3.2.1 Linear Regression Results . . . . .	14
3.2.2 Decison Trees Results . . . . .	17
3.2.3 Random Forest Results . . . . .	20
3.2.4 Ranger Random Forest Results . . . . .	22
3.2.5 Neural Network Results . . . . .	24
3.2.6 K-means Results . . . . .	24
3.2.7 Principal Component Analysis Results . . . . .	24
3.3 Discussion . . . . .	25
3.3.1 Linear Regression Discussion . . . . .	25
3.3.2 Decision Tree Discussion . . . . .	25
3.3.3 Random Forest Discussion . . . . .	25

3.3.4	Ranger Random Forest Discussion . . . . .	25
3.3.5	Neural Network Discussion . . . . .	25
3.3.6	K-Means Discussion . . . . .	25
3.3.7	Principal Component Analysis Discussion . . . . .	26
<b>4</b>	<b>Conclusion</b>	<b>26</b>
4.1	Linear Regression . . . . .	26
4.2	Decision Tree . . . . .	26
4.3	Random Forest . . . . .	26
4.3.1	Ranger Random Forest . . . . .	26
<b>5</b>	<b>Appendix</b>	<b>27</b>

---

## LIST OF FIGURES

1	Price summary . . . . .	10
2	Distribution of all houses in NY colored by prices . . . . .	11
3	Distribution of all houses in NY of price between 15\$ and 500\$ per day . .	11
4	Approximated distribution map of all houses in Manhattan . . . . .	12
5	Summary of the MSE for all the models for each subset of the dataset . . .	13
6	Linear Regression output for the entire dataset . . . . .	14
7	Linear Regression output filtering by Manhattan . . . . .	15
8	Linear Regression output filtering by Manhattan and Entire home/Apartment . . . . .	16
9	Decision Tree results on the entire dataset . . . . .	17
10	Tree generated for the entire dataset . . . . .	17
11	Decision tree results for specific neighbourhood group . . . . .	17
12	Decision tree results for specific neighbourhood group and room type . .	19
13	Random Forest results on the entire dataset . . . . .	20
14	Random Forest results for specific neighbourhood group . . . . .	20
15	Random Forest results for specific neighbourhood group and room type . .	21
16	Ranger Random Forest results on the entire dataset . . . . .	22
17	Ranger Random Forest results for specific neighbourhood group . . . . .	22
18	Ranger Random Forest results for specific neighbourhood group and room type . . . . .	23

# 1

---

## ABSTRACT

The aim of the project is to analyse, develop prediction models and define data clusters from the "New York City Airbnb Open Data" from a Kaggle competition.

In particular, one part is focused on the development of predictive models to forecast house prices using these Supervised Learning technics:

- Linear Regression
- Decision Tree
- Random Forest
- Ranger Random Forest
- Neural Newtworks

For each of these, a comparison between the Mean Square Error and between all  $R^2$  measure of all methods has been made to highlight which have the best performance. Also, for training all the models, a partition of the dataset in three parts has been applied: entire dataset, filtered by neigborhood group and filtered for neighborhood group and room type. In this way, it is possible to check the perfomance giving less or more features in input.

The second part is focused on the cluster and data reduction technics using these Unsupervised Learning technics:

- K-means Algorithm
- Clustering for mixed-type data
- Principal Component Analysis

# 2

---

## PROBLEM DEFINITION AND ALGORITHM

### 2.1 TWO MAIN GOALS

#### 2.1.1 DEVELOP PREDICTIVE MODELS FOR PRICE

The first objective is the forecast of the prices given some input information. This could be usefull for a lot of scenarios. For example from a AirBnB customer point of view, he/she would like to get the list of houses more in alignment with his/her preference choice; or for a host point of view, where given the position and other information he/she could get information about the possible per day price of his property in New York City.

#### 2.1.2 DEFINE CLUSTERS AND GROUPS

The second objective is the definition of group or partition between the houses with different characteristics. For a user point of view could be usefull to have information about available houses similar to those booked in the past. This could be also usefull after the booking for a suggestion analysis having the information about last booked houses in NewYork or houses in similar cities around the world.

## 2.2 ALGORITHMS

### 2.2.1 LINEAR REGRESSION

Linear regression is a linear approach to modeling the relationship between a dependent variable and one or more independent variables.

### 2.2.2 DECISON TREES

In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making.

### 2.2.3 RANDOM FOREST

Random forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees.

### 2.2.4 RANGER RANDOM FOREST

Ranger is a fast implementation of random forests or recursive partitioning, particularly suited for high dimensional data.

### 2.2.5 NEURAL NETWORKS

Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns.

### 2.2.6 K-MEANS

K-means is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean.

### 2.2.7 PRINCIPAL COMPONENT ANALYSIS

Principal Component Analysis (PCA) is mostly used as a tool in exploratory data analysis and for making predictive models.

# 3

---

## EXPERIMENTAL EVALUATION

### 3.1 METHODOLOGY

#### 3.1.1 DATA INSPECTION

The dataset is part of a Kaggle competition, called the New York City Airbnb Open Data. It contains 48.000 rows per 16 columns. The dataset is structured with these columns:

- **id**
- **name**: name of the listing
- **host\_id**
- **host\_name**
- **neighbourhood\_group**: location
- **neighbourhood**: area
- **latitude**: coordinates
- **longitude**: coordinates
- **room\_type**: space type
- **price**: in dollars
- **minimum\_nights**: amount of nights minimum
- **number\_of\_reviews**: number of reviews
- **last\_review**: latest review
- **reviews\_per\_month**: number of reviews per month
- **calculated\_host\_listings\_count**: amount of listing per host
- **availability\_365**: number of days when listing is available for booking

It has been select only 5 of these variables: price, latitude, longitude, neighbourhood\_group and room\_type. The reason is that it is reasonable to select them to predict the prices and to obtain different clusters. The position and the neighbourhood is important since if a property is positioned near the city center will have a higher price with respect to those situated in the outskirts; also the type of room, since an entire apartment will cost more than a single room.

From the Figure 1 is possible to see the distribution of the price. The minimum price is 0 and the maximum is 10000\$. It is not possible to rent a house for free, so it is possible to filter the price with a price higher than 15\$. It is possible that a luxury house cost a lot per day, but these value can not be consider in the model, instead they are outliers and for this reason it is convenient to filter the price again and take those that have a value lower than 500\$. The reason is that the third quantile has a price of 175\$ which is a far from 10000\$ (Figure 3). It has also been checked the null and missing value in the dataset and has not been found apart from the reviews\_per\_month column. It is not a problem, since this feature has been not taken in account to train the models.

```
price
Min. : 0.0
1st Qu.: 69.0
Median : 106.0
Mean : 152.7
3rd Qu.: 175.0
Max. : 10000.0
```

Figure 1: Price summary

From Figure 2, it is possible to see the distribution of all houses in New York City and the price. This picture is not really informative since it can be noticed that the prices for the most part are in the range 0-500\$ and only a low number of instances have a price greater than 500\$. Deleting the outliers, the Figure 3 is more informative than the one before.

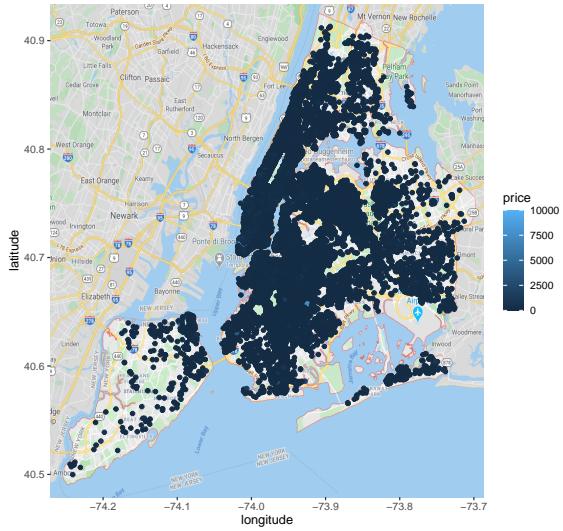


Figure 2: Distribution of all houses in NY colored by prices

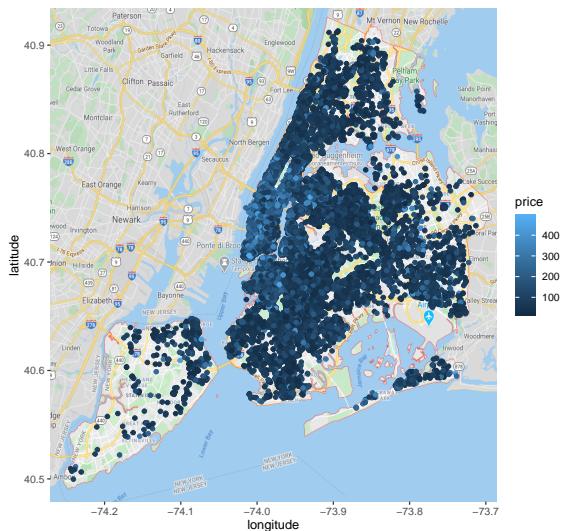


Figure 3: Distribution of all houses in NY of price between 15\$ and 500\$ per day

### 3.1.2 DATA CLEANING AND PRE-PROCESSING

The dataset has approximately 48.000 rows for each column, so an important part of the work is related to the pre-processing due to the large amount of data. For this reason, some variable has been rescaled to let the models to learn faster and better and to perform a better prediction. Latitude and longitude have not been rescaled for forecast price model, while for clustering methods to have a consistent distance calculation. Categorical variables have also been rescaled assigning a numerical value to each category, resulting as unordered factors.

The selected categorical variables are: neighbourhood\_group and room type.

The selected numerical variables are: latitude, longitude and price.

Considering that every neighbourhood group has his characteristic, also taking in account the room type that could impact largely the price. It is possible to define of different subset of the original set and try different methods of forecast. This is due to the fact that a customer should choose which of the different neighbourhood and room type is interested in. Models have been run to different scenarios (Figure 4): users interested in all New York City houses and all type of room, users interested only in a single neighbourhood and users interested in a single neighbourhood and a single room type.

Including different scenario could be potentially computationally expensive. In reality, the training proceeded without any problem.<sup>1</sup> Computational problem may emerge for the case of hyperparametrisation tuning, even using multiprocessing and multithreading technics. For semplicity, in this project no model tuning have been made.

Also for clustering have been filter every scenario, but not for the case of Hierarchical Clustering because it need to generate a readable dendrogram using the mean of all neighbourhood or room type price, Principal Component Analysis since avaible data are really small for some specific neighbourhood like Staten Island and specific room type.

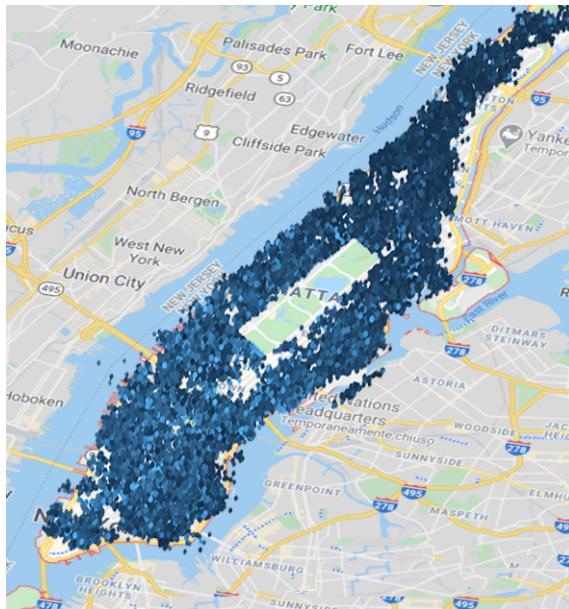


Figure 4: Approximated distribution map of all houses in Manhattan

---

<sup>1</sup>The models have been trained on a machine with quad-core 3.5 GHz processor, 8 Gb RAM and 4 Gb dedicated GPU.

## 3.2 RESULTS

Since the number of models trained are very high, the study result will be presented for different macro groups that depend on the different subset of the dataset: entire dataset, filtering by neighborhood and filtering by neighborhood and room type. This is due to the fact that neighborhood in general have similar result (like Manhattan has similar shape of Brooklyn and Staten Island has a similar shape of Bronx).

===== DA RIVEDERE ===

Subset	Linear.Regression	Decision.Tree	Random.Forest	Ranger.Random.Forest	Neural.Networks
<b>Brooklyn</b>	0.4351730	0.4395665	0.4182327	0.4141652	0.4669738
<b>Manhattan</b>	0.7936143	0.7964149	0.7468374	0.7472513	0.8750774
<b>Queens</b>	0.3721530	0.3699898	0.3513738	0.3515902	0.5491319
<b>Staten Island</b>	0.4776541	0.5173908	0.4873455	0.4956109	1.3730607
<b>Bronx</b>	0.4040408	0.4073877	0.4001876	0.4032073	0.4710128
<b>Brooklyn/Private room</b>	0.1743968	0.1691098	0.1791112	0.1785046	0.2344717
<b>Brooklyn/Entire home/apt</b>	0.7262980	0.7179003	0.7699032	0.7713528	0.7935944
<b>Brooklyn/Shared room</b>	0.2772952	0.3043294	0.2515028	0.2533654	0.2879471
<b>Manhattan/Private room</b>	0.4372064	0.3882155	0.3733743	0.3744351	0.5208271
<b>Manhattan/Entire home/apt</b>	1.0114706	1.0174490	1.0533165	1.0552250	1.1390734
<b>Manhattan/Shared room</b>	0.4904792	0.4992158	0.5619616	0.5622654	0.5155457
<b>Queens/Private room</b>	0.1680487	0.1600965	0.1625676	0.1625325	0.4625950
<b>Queens/Entire home/apt</b>	0.6817353	0.6590635	0.6514473	0.6489757	0.7061921
<b>Queens/Shared room</b>	0.0709897	0.2989787	0.2047612	0.2114044	2.0645615
<b>Staten Island/Private room</b>	0.2640100	0.2339327	0.2291305	0.2252806	0.2557696
<b>Staten Island/Entire home/apt</b>	0.5703345	0.7737648	0.7048053	0.7076976	0.6449374
<b>Staten Island/Shared room</b>	0.3209208	0.5818964	0.3060744	0.5923555	0.5363675
<b>Bronx/Private room</b>	0.1475827	0.1611146	0.1330419	0.1341573	0.4504089
<b>Bronx/Entire home/apt</b>	0.5627967	0.7622148	0.6439600	0.6650687	1.1799774
<b>Bronx/Shared room</b>	0.0690389	0.0489295	0.0608120	0.0536876	0.2355241
<b>All</b>	0.6129722	0.6053303	0.5819694	0.5463735	0.6369116

Figure 5: Summary of the MSE for all the models for each subset of the dataset

### 3.2.1 LINEAR REGRESSION RESULTS

#### Linear Regression on the entire dataset

The results on the entire dataset are acceptable. All variables are significant for the model and the  $R^2$  value is 40% (Figure 7). The Mean Square Error (Figure 5) has a value of 0.6 which is acceptable. All the neighbourhood groups have a positive effect on the price apart for the 4<sup>th</sup> one. This could be due to the fact that Staten Island does not have expensive houses compared to the other group.

Also the latitude and longitude have a slight negative effect on the price. Entire apartment have a slight positive effect while the shared house negative. This could be due to the fact that an entire house will cost more than a shared room, so the price will increase for this type of room.

```
[1] "===== all ====="
Call:
lm(formula = price ~ ., data = trains[[sub]])
Residuals:
    Min      1Q  Median      3Q     Max 
-2.1739 -0.4434 -0.1419  0.2269  4.9515 
Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -1.916e+02  1.420e+01 -13.497 < 2e-16 ***
neighbourhood_group2 5.023e-01  1.618e-02  31.041 < 2e-16 ***
neighbourhood_group3 2.216e-01  1.978e-02  11.202 < 2e-16 ***
neighbourhood_group4 -9.535e-01  5.882e-02 -16.211 < 2e-16 ***
neighbourhood_group5 2.687e-01  3.914e-02   6.865 6.81e-12 ***
latitude        -1.726e+00  1.393e-01 -12.385 < 2e-16 *** 
longitude       -3.532e+00  1.595e-01 -22.145 < 2e-16 *** 
room_type2      9.996e-01  9.622e-03 103.887 < 2e-16 *** 
room_type3     -2.379e-01  3.068e-02  -7.754 9.18e-15 *** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 

Residual standard error: 0.784 on 28680 degrees of freedom
Multiple R-squared:  0.3914,   Adjusted R-squared:  0.3912 
F-statistic: 2306 on 8 and 28680 DF,  p-value: < 2.2e-16
```

Figure 6: Linear Regression output for the entire dataset

#### Linear Regression for specific Neighbourhood group

Models give different results for the specific neighbourhood group. For the first three (Brooklyn, Manhattan and Queens)  $R^2$  are over approximately 30%. For the MSE, the value have different ranges based on the distribution of prices in the singular neighbourhood. To be noticed, is the fact that MSE of Brooklyn is significantly lower with respect to Manhattan, given they have a similar number of houses and have almost the same price distribution. The only difference between the two is the fact that Manhattan has more entire apartment type of room. All variables have a significant effect on the price apart for the latitude in Manhattan and longitude for Queens with a 0.1.

For the last two groups (Staten Island and Bronx) all variables does not have significance in the response variable apart for the Entire Apartment dummy which have a positive effect. This is due to the fact that entire properties will cost more than shared rooms.

```

[1] "===== 1 ====="
Call:
lm(formula = price ~ ., data = trains[[sub]])

Residuals:
    Min      1Q  Median      3Q     Max 
-1.8239 -0.3502 -0.1259  0.1721  5.3480 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -555.16875 20.65023 -26.884 < 2e-16 ***
latitude     5.17353  0.20932  24.577 < 2e-16 ***
longitude   -0.66795  0.21203 -30.323 < 2e-16 ***
room_type2  0.96345  0.01137  84.701 < 2e-16 ***
room_type3 -0.17115  0.03990 -4.289  1.8e-03 *** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6774 on 14889 degrees of freedom
Multiple R-squared:  0.3801, Adjusted R-squared:  0.38 
F-statistic: 2283 on 4 and 14889 DF, p-value: < 2.2e-16

[1] "===== 2 ====="
Call:
lm(formula = price ~ ., data = trains[[sub]])

Residuals:
    Min      1Q  Median      3Q     Max 
-2.3475 -0.5308 -0.1869  0.2803  4.7768 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -895.75437 58.11443 -15.414 < 2e-16 ***
latitude    -0.10719  0.35247 -0.304   0.761  
longitude   -12.16468 0.61365 -19.823 < 2e-16 ***
room_type2  1.02538  0.01494  68.621 < 2e-16 ***
room_type3 -0.25643  0.04737 -5.413 6.29e-08 *** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.877 on 15653 degrees of freedom
Multiple R-squared:  0.3391, Adjusted R-squared:  0.339 
F-statistic: 2008 on 4 and 15653 DF, p-value: < 2.e-16

[1] "===== 3 ====="
Call:
lm(formula = price ~ ., data = trains[[sub]])

Residuals:
    Min      1Q  Median      3Q     Max 
-1.3833 -0.3036 -0.1202  0.1360  4.9322 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 3.74775 12.41542  0.302  0.7628  
latitude    -0.77393 0.27981 -2.766  0.0057 ** 
longitude  -0.36613 0.19934 -1.837  0.0663  
room_type2  0.81079 0.01957 41.426 < 2e-16 ***
room_type3 -0.23506 0.05230 -4.494 7.17e-06 *** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6063 on 4219 degrees of freedom
Multiple R-squared:  0.3065, Adjusted R-squared:  0.3058 
F-statistic: 466.1 on 4 and 4219 DF, p-value: < 2.2e-16

[1] "===== 5 ====="
Call:
lm(formula = price ~ ., data = trains[[sub]])

Residuals:
    Min      1Q  Median      3Q     Max 
-0.9669 -0.2906 -0.1359  0.1124  4.9772 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 62.09595 75.12511  0.827  0.409  
latitude   -0.46076 0.89539 -0.515  0.607  
longitude  0.59638 0.72879  0.818  0.413  
room_type2 0.67380 0.04732 14.238 <2e-16 *** 
room_type3 -0.15156 0.09529 -1.591  0.112  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6266 on 806 degrees of freedom
Multiple R-squared:  0.2199, Adjusted R-squared:  0.216 
F-statistic: 56.79 on 4 and 806 DF, p-value: < 2.2e-16

```

Figure 7: Linear Regression output filtering by Manhattan

### Linear Regression for specific Neighbourhood group and room type

Adding a filter also for the room type, models give acceptable results in some cases and not in others. (Figure 8) For the larger subset the model confirm the significant of the variables, apart for latitude in Manhattan such as happened in the model using this neighbourhood data. In Manhattan with Entire apartment model, instead, all variables have a significant effect. For the remaining neighbourhood groups the variables do not result as significant in the model. This could be due to the fact that the subset obtained from this filtering are not large ( they have less than 300 rows). Also in all model the  $R^2$  result to be lower than 10% but this depends on the fact to train they have less information.

Also in this model the MSE of Brooklyn is lower in all the type of room type than the Manhattan MSE.

```

[1] "***** n1-r1 *****"
Call:
lm(formula = price ~ ., data = trains[[sub]])

Residuals:
    Min      1Q  Median      3Q     Max 
-0.6346 -0.2358 -0.0916  0.1118  5.1399 

Coefficients:
Estimate Std. Error t value Pr(>|t|) 
(Intercept) -360.5784  19.4062 -18.58 <2e-16 ***
latitude     -2.7783  19.1990 13.96 <2e-16 *** 
longitude   -3.3383  0.2114 -15.79 <2e-16 *** 
... 
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.422 on 6721 degrees of freedom
Multiple R-squared:  0.04984, Adjusted R-squared:  0.04956 
F-statistic: 176.3 on 2 and 6721 DF, p-value: < 2.2e-16

[1] "***** n2-r1 *****"
Call:
lm(formula = price ~ ., data = trains[[sub]])

Residuals:
    Min      1Q  Median      3Q     Max 
-1.2007 -0.3766 -0.1682  0.1281  4.7297 

Coefficients:
Estimate Std. Error t value Pr(>|t|) 
(Intercept) -67.3538  81.8873 -0.83 0.4104 *** 
latitude     -0.3333  0.4675  0.71 0.4774 
longitude   -0.9389  0.8670 10.820 <2e-16 *** 
... 
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6845 on 5235 degrees of freedom
Multiple R-squared:  0.1039, Adjusted R-squared:  0.1036 
F-statistic: 304.5 on 2 and 5232 DF, p-value: < 2.2e-16

[1] "***** n3-r1 *****"
Call:
lm(formula = price ~ ., data = trains[[sub]])

Residuals:
    Min      1Q  Median      3Q     Max 
-0.5259 -0.2303 -0.0979  0.1135  4.6236 

Coefficients:
Estimate Std. Error t value Pr(>|t|) 
(Intercept) -18.6830  12.6014 -1.483 0.1383 
latitude     -0.2314  0.2962 -0.781 0.4616 
longitude   -0.3707  1.1883  0.316 0.5951 
... 
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4295 on 2239 degrees of freedom
Multiple R-squared:  0.001641, Adjusted R-squared:  0.0007496 
F-statistic: 1.841 on 2 and 2239 DF, p-value: 0.159

[1] "***** n4-r1 *****"
all:
nformula = price ~ ., data = trains[[sub]]

Residuals:
    Min      1Q  Median      3Q     Max 
0.4759 -0.2744 -0.1064  0.1763  2.6581 

Coefficients:
Estimate Std. Error t value Pr(>|t|) 
Intercept) 88.5017 140.5864  0.630  0.530 
atitude     -88.8712  1.5119 -1.238  0.218 
ongitude   0.1791  1.3315  0.135  0.893 

Residual standard error: 0.4417 on 122 degrees of freedom
Multiple R-squared:  0.01517, Adjusted R-squared:  -0.000978 
F-statistic: 0.9394 on 2 and 122 DF, p-value: 0.3937

[1] "***** n5-r1 *****"
Call:
lm(formula = price ~ ., data = trains[[sub]])

Residuals:
    Min      1Q  Median      3Q     Max 
-0.46134 -0.21757 -0.08786  0.10675  2.68073 

Coefficients:
Estimate Std. Error t value Pr(>|t|) 
(Intercept) 94.6199  61.1603  1.547  0.123 
atitude     -0.5680  0.7298 -0.778  0.437 
longitude   0.9775  0.6102  1.602  0.110 

Residual standard error: 0.3838 on 428 degrees of freedom
Multiple R-squared:  0.006174, Adjusted R-squared:  0.00153 
F-statistic: 1.329 on 2 and 428 DF, p-value: 0.2657

[1] "***** n1-r2 *****"
Call:
lm(formula = price ~ ., data = trains[[sub]])

Residuals:
    Min      1Q  Median      3Q     Max 
-1.6047 -0.5780 -0.2051  0.2878  4.0869 

Coefficients:
Estimate Std. Error t value Pr(>|t|) 
(Intercept) -779.6115  40.0559 -19.46 <2e-16 *** 
latitude     -7.9775  0.4233 19.35 <2e-16 *** 
longitude   -6.1574  0.4511 -13.65 <2e-16 *** 
... 
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6896 on 6238 degrees of freedom
Multiple R-squared:  0.01233, Adjusted R-squared:  0.07093 
F-statistic: 239.2 on 2 and 6238 DF, p-value: < 2.2e-16

[1] "***** n2-r2 *****"
Call:
lm(formula = price ~ ., data = trains[[sub]])

Residuals:
    Min      1Q  Median      3Q     Max 
-2.4464 -0.6834 -0.2428  0.4327  3.8755 

Coefficients:
Estimate Std. Error t value Pr(>|t|) 
(Intercept) -89.8697  9.8895 -9.089 <2e-16 *** 
latitude     -0.9338  0.5683  1.643  0.1 
longitude   -12.5366  0.9398 -13.334 <2e-16 *** 
... 
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9896 on 8343 degrees of freedom
Multiple R-squared:  0.07902, Adjusted R-squared:  0.0788 
F-statistic: 357.9 on 2 and 8343 DF, p-value: < 2.2e-16

[1] "***** n3-r2 *****"
Call:
lm(formula = price ~ ., data = trains[[sub]])

Residuals:
    Min      1Q  Median      3Q     Max 
-1.3955 -0.5444 -0.2012  0.3158  4.1267 

Coefficients:
Estimate Std. Error t value Pr(>|t|) 
(Intercept) -86.9386  28.50482 -3.018 0.024 
atitude     -0.65874  0.61449 -1.072 0.284 
longitude   0.06699  0.62009  0.139  0.889 

Residual standard error: 0.8305 on 1381 degrees of freedom
Multiple R-squared:  0.01545, Adjusted R-squared:  0.0097e-05 
F-statistic: 1.068 on 2 and 1381 DF, p-value: 0.3438

[1] "***** n4-r2 *****"
all:
mformula = price ~ ., data = trains[[sub]]

Residuals:
    Min      1Q  Median      3Q     Max 
0.9076 -0.5490 -0.2720  0.2208  3.6389 

Coefficients:
Estimate Std. Error t value Pr(>|t|) 
Intercept) 40.350 296.277  0.136  0.892 
atitude     1.207  2.991  0.403  0.687 
ongitude   1.207  2.710  0.445  0.657 

Residual standard error: 0.8080 on 110 degrees of freedom
Multiple R-squared:  0.009854, Adjusted R-squared:  -0.008148 
F-statistic: 0.5474 on 2 and 110 DF, p-value: 0.58

[1] "***** n5-r2 *****"
Call:
lm(formula = price ~ ., data = trains[[sub]])

Residuals:
    Min      1Q  Median      3Q     Max 
-1.8884 -0.5191 -0.2658  0.1793  4.2156 

Coefficients:
Estimate Std. Error t value Pr(>|t|) 
(Intercept) -53.3410  175.2402 -0.304  0.761 
atitude     1.6975  2.1824  0.778  0.437 
longitude   0.2177  1.6531  0.138  0.895 

Residual standard error: 0.9039 on 248 degrees of freedom
Multiple R-squared:  0.003215, Adjusted R-squared:  -0.004823 
F-statistic: 0.4 on 2 and 248 DF, p-value: 0.6708

[1] "***** n1-r3 *****"
Call:
lm(formula = price ~ ., data = trains[[sub]])

Residuals:
    Min      1Q  Median      3Q     Max 
-0.4449 -0.2589 -0.1688  0.0239  3.9904 

Coefficients:
Estimate Std. Error t value Pr(>|t|) 
(Intercept) -319.2018  111.6378 -2.866 0.00449 ** 
atitude     2.9447  0.9328  3.157 0.00178 ** 
longitude   -2.6943  1.2248 -2.200 0.02867 * 
... 
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.52 on 271 degrees of freedom
Multiple R-squared:  0.03881, Adjusted R-squared:  0.03174 
F-statistic: 5.474 on 2 and 271 DF, p-value: 0.004671

[1] "***** n2-r3 *****"
Call:
lm(formula = price ~ ., data = trains[[sub]])

Residuals:
    Min      1Q  Median      3Q     Max 
-0.7627 -0.3930 -0.2213  0.0386  4.6007 

Coefficients:
Estimate Std. Error t value Pr(>|t|) 
(Intercept) -1148.519  330.424 -3.476 0.000583 *** 
atitude     -1.1862  2.025  0.554 0.4045 * 
longitude   -13.226  3.517 -3.761 0.000202 *** 
... 
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7717 on 314 degrees of freedom
Multiple R-squared:  0.05117, Adjusted R-squared:  0.04513 
F-statistic: 8.467 on 2 and 314 DF, p-value: 0.000262

[1] "***** n3-r3 *****"
Call:
lm(formula = price ~ ., data = trains[[sub]])

Residuals:
    Min      1Q  Median      3Q     Max 
-0.5401 -0.2593 -0.1545  0.0127  5.0126 

Coefficients:
Estimate Std. Error t value Pr(>|t|) 
(Intercept) -101.808  90.766 -1.122 0.2641 
atitude     4.512  2.232  2.022 0.0433 * 
longitude   1.322  1.312  0.853 0.3941 

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6257 on 126 degrees of freedom
Multiple R-squared:  0.03195, Adjusted R-squared:  0.01659 
F-statistic: 2.08 on 2 and 126 DF, p-value: 0.1293

[1] "***** n4-r3 *****"
all:
mformula = price ~ ., data = trains[[sub]]

Residuals:
    1      2      4      5      7      8 
0.153761 -0.025443 -0.290715  0.569389 -0.397762 -0.009229 

Coefficients:
Estimate Std. Error t value Pr(>|t|) 
Intercept) -2919.25  3307.89 -0.883  0.442 
atitude     46.13   21.25  2.171  0.118 
ongitude   -14.11   37.85 -0.373  0.734 

Residual standard error: 0.444 on 3 degrees of freedom
Multiple R-squared:  0.6387, Adjusted R-squared:  0.3979 
F-statistic: 2.652 on 2 and 3 DF, p-value: 0.2172

[1] "***** n5-r3 *****"
all:
nformula = price ~ ., data = trains[[sub]]

Residuals:
    Min      1Q  Median      3Q     Max 
0.40836 -0.14433 -0.05595  0.06429  1.04848 

Coefficients:
Estimate Std. Error t value Pr(>|t|) 
(Intercept) 77.152  186.703  0.413  0.682 
atitude     1.712  2.084  0.822  0.417 
ongitude   2.004  1.707  1.174  0.248 

Residual standard error: 0.2919 on 36 degrees of freedom
Multiple R-squared:  0.1093, Adjusted R-squared:  0.05986 
F-statistic: 2.21 on 2 and 36 DF, p-value: 0.1244

```

Figure 8: Linear Regression output filtering by Manhattan and Entire home/Apartment

### 3.2.2 DECISON TREES RESULTS

#### Decison tree without filters

For the entire dataset the MSE has a value of 0.605 (Figure 9) Important to be notice is the fact that the neigborhood group variable have not been used in the construction of the tree. Moreover, what really influence the price is the type of room (Figure 10).

```
[1] "===== all ====="
```

```
Regression tree:
tree(formula = price ~ ., data = trains[[sub]])
Variables actually used in tree construction:
[1] "room_type" "longitude" "latitude"
Number of terminal nodes: 5
Residual mean deviance: 0.6053 = 17360 / 28680
Distribution of residuals:
Min. 1st Qu. Median Mean 3rd Qu. Max.
-2.2750 -0.4143 -0.1493 0.0000 0.2142 4.8720
[1] 0.6053303
```

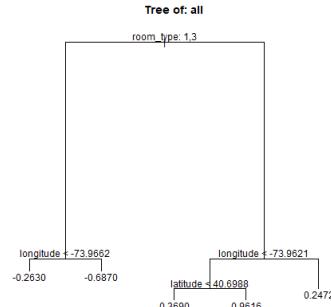


Figure 9: Decision Tree results on the entire dataset

Figure 10: Tree generated for the entire dataset

#### Decison tree for specific Neigborhood group

For the specific neighborhood group decision tree models give output similar MSE results with respect to the Linear Regression. Also in this case, the latitude for Manhattan has not been used in the model to predict the price (Figure 11).

```
[1] "===== 1 ====="
Regression tree:
tree(formula = price ~ ., data = trains[[sub]])
Number of terminal nodes: 4
Residual mean deviance: 0.4618 = 6876 / 14890
Distribution of residuals:
Min. 1st Qu. Median Mean 3rd Qu. Max.
-1.9220 -0.3540 -0.1268 0.0000 0.1630 4.8710
[1] 0.4395665

[1] "===== 2 ====="
Regression tree:
tree(formula = price ~ ., data = trains[[sub]])
Variables actually used in tree construction:
[1] "room_type" "longitude"
Number of terminal nodes: 4
Residual mean deviance: 0.7757 = 12140 / 15650
Distribution of residuals:
Min. 1st Qu. Median Mean 3rd Qu. Max.
-2.2950 -0.5138 -0.1934 0.0000 0.3072 4.7680
[1] 0.7964149

[1] "===== 3 ====="
Regression tree:
tree(formula = price ~ ., data = trains[[sub]])
Variables actually used in tree construction:
[1] "room_type"
Number of terminal nodes: 5
Residual mean deviance: 0.3588 = 1514 / 4220
Distribution of residuals:
Min. 1st Qu. Median Mean 3rd Qu. Max.
-1.4870 -0.3017 -0.1276 0.0000 0.1564 4.9280
[1] 0.3699898

[1] "===== 4 ====="
Regression tree:
tree(formula = price ~ ., data = trains[[sub]])
Number of terminal nodes: 11
Residual mean deviance: 0.2805 = 73.77 / 263
Distribution of residuals:
Min. 1st Qu. Median Mean 3rd Qu. Max.
-1.60400 -0.25800 -0.08494 0.00000 0.14370 2.82700
[1] 0.5173908

[1] "===== 5 ====="
Regression tree:
tree(formula = price ~ ., data = trains[[sub]])
Variables actually used in tree construction:
[1] "room_type"
Number of terminal nodes: 2
Residual mean deviance: 0.3482 = 281.7 / 809
Distribution of residuals:
Min. 1st Qu. Median Mean 3rd Qu. Max.
-1.0760 -0.2764 -0.1060 0.0000 0.1212 5.0060
[1] 0.4073877
```

Figure 11: Decision tree results for specific neighborhood group

### **Decison tree for specific Neighboorhood group and room type**

For neighborhood group and room price Decision tree construction was not possible for all the subset. For example for shared room for Bronx were not possible to plot the tree since the model had only one node. MSE are similar to Linear Regression but for this type of subset perform slightly worse. This is due to the lack of information of some of these subsets (Figure 12).

```

[1] "===== n1-r1 ====="
Regression tree:
tree(formula = price ~ ., data = trains[[sub]])
Number of terminal nodes: 4
Residual mean deviance: 0.1783 = 1198 / 6720
Distribution of residuals:
Min. 1st Qu. Median Mean 3rd Qu. Max.
-0.75440 -0.22280 -0.07515 0.00000 0.11800 4.94600
[1] 0.1691098

[1] "===== n1-r2 ====="
Regression tree:
tree(formula = price ~ ., data = trains[[sub]])
Number of terminal nodes: 5
Residual mean deviance: 0.7349 = 4583 / 6236
Distribution of residuals:
Min. 1st Qu. Median Mean 3rd Qu. Max.
-1.7400 -0.5756 -0.1893 0.00000 0.2752 4.3100
[1] 0.7179003

[1] "===== n1-r3 ====="
Regression tree:
tree(formula = price ~ ., data = trains[[sub]])
Number of terminal nodes: 4
Residual mean deviance: 0.1571 = 42.25 / 269
Distribution of residuals:
Min. 1st Qu. Median Mean 3rd Qu. Max.
-0.87700 -0.16430 -0.08477 0.00000 0.02883 4.06200
[1] 0.3043294

[1] "===== n2-r1 ====="
Regression tree:
tree(formula = price ~ ., data = trains[[sub]])
Number of terminal nodes: 8
Residual mean deviance: 0.3902 = 2048 / 5248
Distribution of residuals:
Min. 1st Qu. Median Mean 3rd Qu. Max.
-2.970 -0.3301 -0.1267 0.00000 0.1243 4.8870
[1] 0.3882155

[1] "===== n2-r2 ====="
Regression tree:
tree(formula = price ~ ., data = trains[[sub]])
Number of terminal nodes: 3
Residual mean deviance: 0.9637 = 8040 / 8343
Distribution of residuals:
Min. 1st Qu. Median Mean 3rd Qu. Max.
-3.3220 -0.6882 -0.2203 0.00000 0.4045 3.8560
[1] 1.017449

[1] "===== n2-r3 ====="
Regression tree:
tree(formula = price ~ ., data = trains[[sub]])
Number of terminal nodes: 10
Residual mean deviance: 0.4952 = 151.5 / 306
Distribution of residuals:
Min. 1st Qu. Median Mean 3rd Qu. Max.
-2.6830 -0.2992 -0.1347 0.00000 0.0506 3.5570
[1] 0.4992158

[1] "===== n3-r1 ====="
Regression tree:
tree(formula = price ~ ., data = trains[[sub]])
Number of terminal nodes: 5
Residual mean deviance: 0.163 = 69.6 / 427
Distribution of residuals:
Min. 1st Qu. Median Mean 3rd Qu. Max.
-1.35200 -0.21400 -0.06405 0.00000 0.12680 3.81700
[1] 0.1611146

[1] "===== n3-r2 ====="
Regression tree:
tree(formula = price ~ ., data = trains[[sub]])
Number of terminal nodes: 8
Residual mean deviance: 0.6961 = 169.1 / 243
Distribution of residuals:
Min. 1st Qu. Median Mean 3rd Qu. Max.
-2.2490 -0.4638 -0.1854 0.00000 0.2079 3.7900
[1] 0.7622148

[1] "===== n3-r3 ====="
Regression tree:
tree(formula = price ~ ., data = trains[[sub]])
Variables actually used in tree construction:
[1] "latitude"
Number of terminal nodes: 6
Residual mean deviance: 0.1029 = 3.5 / 34
Distribution of residuals:
Min. 1st Qu. Median Mean 3rd Qu. Max.
-0.52340 -0.15660 -0.04544 0.00000 0.04203 0.92870
[1] 0.04892947

```

Figure 12: Decision tree results for specific neighbourhood group and room type

### 3.2.3 RANDOM FOREST RESULTS

#### Random Forest on the entire dataset

Random Forest on the entire dataset outputs an explained variance of 42% which is acceptable and also the MSE is lower with respect to the models before.

```
[1] "===== all ====="
Call:
randomForest(formula = price ~ ., data = trains[[sub]])
  Type of random forest: regression
  Number of trees: 500
  No. of variables tried at each split: 1

  Mean of squared residuals: 0.5754847
  % Var explained: 42.18
[1] "MSE: 0.578656097325552"
```

Figure 13: Random Forest results on the entire dataset

#### Random Forest for specific Neighbourhood group

Random Forest on specific neighborhood outputs overall MSE lower than the last two models. Explained variance are acceptable: for Brooklyn and Manhattan is approximately around 40%, 37% for Queens and about 25% for Staten Island and Bronx.

```
[1] "===== 1 ====="
Call:
randomForest(formula = price ~ ., data = trains[[sub]])
  Type of random forest: regression
  Number of trees: 500
  No. of variables tried at each split: 1

  Mean of squared residuals: 0.4407074
  % Var explained: 40.45
[1] "MSE: 0.418992779004395"

[1] "===== 2 ====="
Call:
randomForest(formula = price ~ ., data = trains[[sub]])
  Type of random forest: regression
  Number of trees: 500
  No. of variables tried at each split: 1

  Mean of squared residuals: 0.72859
  % Var explained: 37.38
[1] "MSE: 0.749899501079486"
-- --
[1] "===== 3 ====="
Call:
randomForest(formula = price ~ ., data = trains[[sub]])
  Type of random forest: regression
  Number of trees: 500
  No. of variables tried at each split: 1

  Mean of squared residuals: 0.3474919
  % Var explained: 34.37
[1] "MSE: 0.352243050175804"

[1] "===== 4 ====="
Call:
randomForest(formula = price ~ ., data = trains[[sub]])
  Type of random forest: regression
  Number of trees: 500
  No. of variables tried at each split: 1

  Mean of squared residuals: 0.3650865
  % Var explained: 23.27
[1] "MSE: 0.484112832735206"

[1] "===== 5 ====="
Call:
randomForest(formula = price ~ ., data = trains[[sub]])
  Type of random forest: regression
  Number of trees: 500
  No. of variables tried at each split: 1

  Mean of squared residuals: 0.3513488
  % Var explained: 24.72
[1] "MSE: 0.396692012991184"
```

Figure 14: Random Forest results for specific neighborhood group

## Random Forest for specific Neighbourhood group and room type

Random Forest for specific neighbourhood and room type does perform poorly since in some cases explained variance is negative. This could be due to the fact of overfitting, but in this case is more probable the problem is underfitting since in some specific case there is a low number of information.

```
[1] "===== n1-r1 ====="
Call:
randomForest(formula = price ~ ., data = trains[[sub]])
  Type of random forest: regression
  Number of trees: 500
No. of variables tried at each split: 1
  Mean of squared residuals: 0.1885908
  % Var explained: 2.21
[1] "MSE: 0.17859959929437"

[1] "===== n1-r2 ====="
Call:
randomForest(formula = price ~ ., data = trains[[sub]])
  Type of random forest: regression
  Number of trees: 500
No. of variables tried at each split: 1
  Mean of squared residuals: 0.8016605
  % Var explained: 1.5
[1] "MSE: 0.768164722382269"

[1] "===== n1-r3 ====="
Call:
randomForest(formula = price ~ ., data = trains[[sub]])
  Type of random forest: regression
  Number of trees: 500
No. of variables tried at each split: 1
  Mean of squared residuals: 0.1849337
  % Var explained: 4.42
[1] "MSE: 0.25323097709792"

[1] "===== n4-r1 ====="
Call:
randomForest(formula = price ~ ., data = trains[[sub]])
  Type of random forest: regression
  Number of trees: 500
No. of variables tried at each split: 1
  Mean of squared residuals: 0.1289314
  % Var explained: -4.61
[1] "MSE: 0.229843237622339"

[1] "===== n4-r2 ====="
Call:
randomForest(formula = price ~ ., data = trains[[sub]])
  Type of random forest: regression
  Number of trees: 500
No. of variables tried at each split: 1
  Mean of squared residuals: 0.7779035
  % Var explained: -15.23
[1] "MSE: 0.71075187666878"

[1] "===== n4-r3 ====="
Call:
randomForest(formula = price ~ ., data = trains[[sub]])
  Type of random forest: regression
  Number of trees: 500
No. of variables tried at each split: 1
  Mean of squared residuals: 0.01837979
  % Var explained: 75.47
[1] "MSE: 0.304203607022342"

[1] "===== n2-r1 ====="
Call:
randomForest(formula = price ~ ., data = trains[[sub]])
  Type of random forest: regression
  Number of trees: 500
No. of variables tried at each split: 1
  Mean of squared residuals: 0.4015436
  % Var explained: 23.75
[1] "MSE: 0.373153844259809"

[1] "===== n2-r2 ====="
Call:
randomForest(formula = price ~ ., data = trains[[sub]])
  Type of random forest: regression
  Number of trees: 500
No. of variables tried at each split: 1
  Mean of squared residuals: 0.9846148
  % Var explained: 5.71
[1] "MSE: 1.05375292497579"

[1] "===== n2-r3 ====="
Call:
randomForest(formula = price ~ ., data = trains[[sub]])
  Type of random forest: regression
  Number of trees: 500
No. of variables tried at each split: 1
  Mean of squared residuals: 0.7182231
  % Var explained: -9.47
[1] "MSE: 0.565461676852938"

[1] "===== n5-r1 ====="
Call:
randomForest(formula = price ~ ., data = trains[[sub]])
  Type of random forest: regression
  Number of trees: 500
No. of variables tried at each split: 1
  Mean of squared residuals: 0.2107108
  % Var explained: -15.87
[1] "MSE: 0.134361647516686"

[1] "===== n5-r2 ====="
Call:
randomForest(formula = price ~ ., data = trains[[sub]])
  Type of random forest: regression
  Number of trees: 500
No. of variables tried at each split: 1
  Mean of squared residuals: 0.902801
  % Var explained: -11.48
[1] "MSE: 0.661500485887416"

[1] "===== n5-r3 ====="
Call:
randomForest(formula = price ~ ., data = trains[[sub]])
  Type of random forest: regression
  Number of trees: 500
No. of variables tried at each split: 1
  Mean of squared residuals: 0.198373
  % Var explained: -39.67
[1] "MSE: 0.0567952387666339"
```

Figure 15: Random Forest results for specific neighbourhood group and room type

### 3.2.4 RANGER RANDOM FOREST RESULTS

Ranger Random Forest is known to be computationally light with respect to the classic Random Forest. In fact, for the tuning part there were not problem in running it for the entire dataset.

#### Ranger on the entire dataset

Ranger outputs for the entire dataset are consistent. We have a  $R^2$  of 0.47 and OOB error of.

```
[1] "===== all ====="
Ranger result

Call:
ranger(price ~ ., data = trains[[sub]], write.forest = TRUE,      classification = F)

Type:                         Regression
Number of trees:                500
Sample size:                     28689
Number of independent variables: 4
Mtry:                           2
Target node size:               5
Variable importance mode:       none
Splitrule:                      variance
OOB prediction error (MSE):    0.548326
R squared (OOB):                0.4568823
[1] "MSE: 0.542787782904249"
```

Figure 16: Ranger Random Forest results on the entire dataset

#### Ranger for specific Neighbourhood group

```
[1] "===== 1 ====="
Ranger result

Call:
ranger(price ~ ., data = trains[[sub]], write.forest = TRUE,      classification = F)

Type:                         Regression
Number of trees:                500
Sample size:                     14894
Number of independent variables: 3
Mtry:                           1
Target node size:               5
Variable importance mode:       none
Splitrule:                      variance
OOB prediction error (MSE):    0.4346521
R squared (OOB):                0.415184
[1] "MSE: 0.423576642914145"

[1] "===== 2 ====="
Ranger result

Call:
ranger(price ~ ., data = trains[[sub]], write.forest = TRUE,      classification = F)

Type:                         Regression
Number of trees:                500
Sample size:                     15657
Number of independent variables: 3
Mtry:                           1
Target node size:               5
Variable importance mode:       none
Splitrule:                      variance
OOB prediction error (MSE):    0.7304504
R squared (OOB):                0.3714795
[1] "MSE: 0.736503144251037"

[1] "===== 3 ====="
Ranger result

Call:
ranger(price ~ ., data = trains[[sub]], write.forest = TRUE,      classification = F)

Type:                         Regression
Number of trees:                500
Sample size:                     41224
Number of independent variables: 3
Mtry:                           1
Target node size:               5
Variable importance mode:       none
Splitrule:                      variance
OOB prediction error (MSE):    0.3459867
R squared (OOB):                0.3468607
[1] "MSE: 0.351308553169474"

[1] "===== 4 ====="
Ranger result

Call:
ranger(price ~ ., data = trains[[sub]], write.forest = TRUE,      classification = F)

Type:                         Regression
Number of trees:                500
Sample size:                     275
Number of independent variables: 3
Mtry:                           1
Target node size:               5
Variable importance mode:       none
Splitrule:                      variance
OOB prediction error (MSE):    0.4478808
R squared (OOB):                0.1898886
[1] "MSE: 0.253040394051809"

[1] "===== 5 ====="
Ranger result

Call:
ranger(price ~ ., data = trains[[sub]], write.forest = TRUE,      classification = F)

Type:                         Regression
Number of trees:                500
Sample size:                     811
Number of independent variables: 3
Mtry:                           1
Target node size:               5
Variable importance mode:       none
Splitrule:                      variance
OOB prediction error (MSE):    0.3959889
R squared (OOB):                0.2091952
[1] "MSE: 0.270598776462825"
```

Figure 17: Ranger Random Forest results for specific neighbourhood group

## Ranger for specific Neighboorhood group and room type

The dataset filtered by neighborhood and room type gives as result a  $R^2$  of 0.25.

```
[1] "***** n1-r1 *****"
Ranger result
Call:
ranger(price ~ ., data = trains[[sub]], write.Forest = TRUE, classification = F)
Type: Regression
Number of trees: 500
Sample size: 6324
Number of independent variables: 2
Mtry: 1
Target node size: 5
Variable importance mode: none
OOS prediction error (MSE): 0.188609
R squared (OOS): 0.0175011
[1] "MSE: 0.188609338342419"

[1] "***** n1-r2 *****"
Ranger result
Call:
ranger(price ~ ., data = trains[[sub]], write.Forest = TRUE, classification = F)
Type: Regression
Number of trees: 500
Sample size: 6348
Number of independent variables: 2
Mtry: 1
Target node size: 5
Variable importance mode: variance
OOS prediction error (MSE): 0.779768
R squared (OOS): 0.02391908
[1] "MSE: 0.805082950328269"

[1] "***** n1-r3 *****"
Ranger result
Call:
ranger(price ~ ., data = trains[[sub]], write.Forest = TRUE, classification = F)
Type: Regression
Number of trees: 500
Sample size: 274
Number of independent variables: 2
Mtry: 1
Target node size: 5
Variable importance mode: none
OOS prediction error (MSE): 0.271288
R squared (OOS): 0.02942318
[1] "MSE: 0.086880072194648"

[1] "***** n4-r1 *****"
Ranger result
Call:
ranger(price ~ ., data = trains[[sub]], write.Forest = TRUE, classification = F)
Type: Regression
Number of trees: 500
Sample size: 2105
Number of independent variables: 2
Mtry: 1
Target node size: 5
Variable importance mode: none
OOS prediction error (MSE): 0.175069
R squared (OOS): 0.1016101
[1] "MSE: 0.14701396762348"

[1] "***** n4-r2 *****"
Ranger result
Call:
ranger(price ~ ., data = trains[[sub]], write.Forest = TRUE, classification = F)
Type: Regression
Number of trees: 500
Sample size: 113
Number of independent variables: 2
Mtry: 1
Target node size: 5
Variable importance mode: variance
OOS prediction error (MSE): 0.5325196
R squared (OOS): -0.1367216
[1] "MSE: 0.0719037733593"

[1] "***** n4-r3 *****"
Ranger result
Call:
ranger(price ~ ., data = trains[[sub]], write.Forest = TRUE, classification = F)
Type: Regression
Number of trees: 500
Sample size: 113
Number of independent variables: 2
Mtry: 1
Target node size: 5
Variable importance mode: none
OOS prediction error (MSE): 0.1159859
R squared (OOS): -0.2385456
[1] "MSE: 0.378764441358849"

[1] "***** n2-r1 *****"
Ranger result
Call:
ranger(price ~ ., data = trains[[sub]], write.Forest = TRUE, classification = F)
Type: Regression
Number of trees: 500
Sample size: 2259
Number of independent variables: 2
Mtry: 1
Target node size: 5
Variable importance mode: variance
OOS prediction error (MSE): 0.4099668
R squared (OOS): 0.2328405
[1] "MSE: 0.378499211515247"

[1] "***** n2-r2 *****"
Ranger result
Call:
ranger(price ~ ., data = trains[[sub]], write.Forest = TRUE, classification = F)
Type: Regression
Number of trees: 500
Sample size: 2348
Number of independent variables: 2
Mtry: 1
Target node size: 5
Variable importance mode: variance
OOS prediction error (MSE): 0.4099668
R squared (OOS): 0.2328405
[1] "MSE: 0.378499211515247"

[1] "***** n3-r1 *****"
Ranger result
Call:
ranger(price ~ ., data = trains[[sub]], write.Forest = TRUE, classification = F)
Type: Regression
Number of trees: 500
Sample size: 2242
Number of independent variables: 2
Mtry: 1
Target node size: 5
Variable importance mode: none
Splitrule: variance
OOS prediction error (MSE): 0.7178005
R squared (OOS): 0.06295275
[1] "MSE: 0.18240649924091"

[1] "***** n3-r2 *****"
Ranger result
Call:
ranger(price ~ ., data = trains[[sub]], write.Forest = TRUE, classification = F)
Type: Regression
Number of trees: 500
Sample size: 1384
Number of independent variables: 2
Mtry: 1
Target node size: 5
Variable importance mode: none
Splitrule: variance
OOS prediction error (MSE): 0.7111906
R squared (OOS): 0.05403314
[1] "MSE: 0.656202123989"

[1] "***** n3-r3 *****"
Ranger result
Call:
ranger(price ~ ., data = trains[[sub]], write.Forest = TRUE, classification = F)
Type: Regression
Number of trees: 500
Sample size: 119
Number of independent variables: 2
Mtry: 1
Target node size: 5
Variable importance mode: none
Splitrule: variance
OOS prediction error (MSE): 0.5746933
R squared (OOS): 0.4190039
[1] "MSE: 0.19834985747352"

[1] "***** n5-r1 *****"
Ranger result
Call:
ranger(price ~ ., data = trains[[sub]], write.Forest = TRUE, classification = F)
Type: Regression
Number of trees: 500
Sample size: 291
Number of independent variables: 2
Mtry: 1
Target node size: 5
Variable importance mode: variance
OOS prediction error (MSE): 0.7350826
R squared (OOS): -0.05847549
[1] "MSE: 0.884606420981124"

[1] "***** n5-r2 *****"
Ranger result
Call:
ranger(price ~ ., data = trains[[sub]], write.Forest = TRUE, classification = F)
Type: Regression
Number of trees: 500
Sample size: 40
Number of independent variables: 2
Mtry: 1
Target node size: 5
Variable importance mode: variance
OOS prediction error (MSE): 0.1958126
R squared (OOS): -0.3441889
[1] "MSE: 0.093473289643726"
```

Figure 18: Ranger Random Forest results for specific neighborhood group and room type

### **3.2.5 NEURAL NETWORK RESULTS**

**Neural Networks on the entire dataset**

**Neural Networks for specific Neighboorhood group**

**Neural Networks for specific Neighboorhood group and room type**

### **3.2.6 K-MEANS RESULTS**

### **3.2.7 PRINCIPAL COMPONENT ANALYSIS RESULTS**

### 3.3 DISCUSSION

#### 3.3.1 LINEAR REGRESSION DISCUSSION

Linear regression model gives interesting results for the non-filtered dataset and also for the filtered by neighbourhood group. All variable results rejected by Null hypothesis, so the model depends on all the selected variables. Latitude and longitude are correlated with the target, room type is strongly positive correlated with the price and neighbourhood group does not seem to have a great contribution in the prediction of the price.

#### 3.3.2 DECISION TREE DISCUSSION

The performance with respect to the other models are not the best, but acceptable. The prediction results are not also very precised for the filtered neighbourhood and room type. Also, the plots of the predicted value are not so consistent since the values are divided in category which correspond to the leaves that are not so strong with respect to the other model predictions.

#### 3.3.3 RANDOM FOREST DISCUSSION

Random forest outputs consistent results and performs better than linear regression and decision tree. Parameters tuning does not give big improvement in performance and also are computationally expensive.

#### 3.3.4 RANGER RANDOM FOREST DISCUSSION

Results of Ranger are the best with respect to the previous models. Also the tuning part was fast and computationally cheaper than the classic Random Forest model but does not give great improvements in performance.

#### 3.3.5 NEURAL NETWORK DISCUSSION

#### 3.3.6 K-MEANS DISCUSSION

# 4

---

## CONCLUSION

From the result the method with the most higher accuracy is the Random Forest method... while the worst are ....

Moreover, Random Forest method is also the worst in term of computation time for the tuning part since it takes for a configuration with 4 core, more or less 1 hour to tune the parameters.

### 4.1 LINEAR REGRESSION

### 4.2 DECISION TREE

Decision tree are one of the most used model in the Machine Learning world since are very familiar to human users and can be easily plotted.

### 4.3 RANDOM FOREST

Random Forest is an ensemble method which use a combination of decision tree to get the prediction.

#### 4.3.1 RANGER RANDOM FOREST

Ranger Random Forest is a computationally light model which results are very close the classical Random Forest.

# 5

---

## APPENDIX

The code is in the file "project-code.Rmd" and is attached to this file.