UNIVERSITÀ DEGLI STUDI DI MILANO

LA STATALE

Department od 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

List of Figures

1

Abstract

The aim of the project is to analyse, develop prediction models and define clusters of the data from ”New York City Airbnb Open Data” Kaggle competition.
One part of the study 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 Networks

For each of these, a comparison between the Mean Square Error of all methods has been made to highlight which have the best performance. Also, for training all models, multiple subsets of the dataset has been applied: filter by neighbourhood group and filter by neighbourhood group and room type. In this way, is possible to give an overview of the result for each case.

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

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

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. This could be useful 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 line with his/her preference choice, or for a host point of view, where given the position and other information he/she could get a suggestion of the 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 between the houses with different characteristics. For a user point of view could be useful to have information about available houses similar to those booked in the past. This could be also useful after the booking for a suggestion analysis having the information about last booked houses in New York or houses in similar cities around the world. Another important analysis, could be determine which are the most important features that describe better the data.

2.2 Algorithms

2.2.1 Linear Regression

Linear regression is a linear approach to modelling 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 represent decisions visually and explicitly 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, modelled 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

PCA produces a low-dimensional representation of a dataset. It finds a sequence of linear combinations of the variables that have maximal variance, and are mutually uncorrelated. Apart from producing derived variables for use in supervised learning problems, PCA also serves as a tool for data visualization.

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 selected only 5 of these variables: price, latitude, longitude, neighbourhood_group and room_type. It is reasonable to select them to predict the prices and to obtain different clusters. The position and the neighbourhood are important since if a property is positioned near the city centre 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$. Price has been filtered with a value greater than 15$ since it is not possible to rent a house for free. Moreover, a luxury house cost a lot per day, but these values cannot 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$. The reason is that the third quantile has a price distribution of 175$ 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 last 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 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.

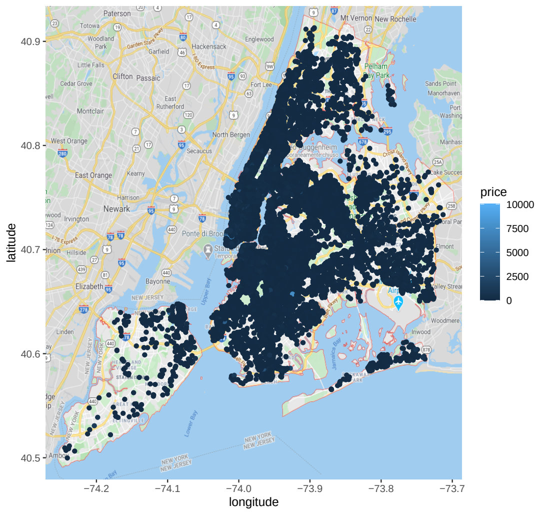


Figure 2: Distribution of all houses in NY colored by prices

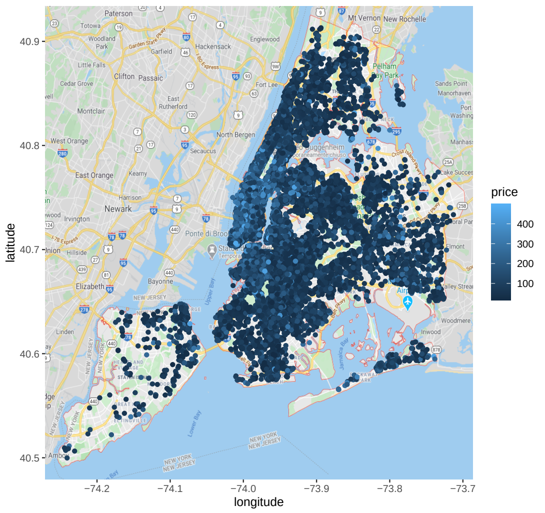


Figure 3: Distribution of all houses in NY of price between 15$ and 500$ per day

3.1.2 Data Cleaning and Pre-processing

The dataset has approximately 48.000 rows for each column and due to the large amount of data, an important part of the work is related to the pre-processing. 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 have being rescaled in cluster analysis for 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 and room type could impact the prices in different way for each singular case, different subsets (Figure 4) of the original dataset have been define and to then try different methods of forecast. This is because a customer should choose which of the different neighbourhood and room type is interested in and not only been generalised to the all New York city houses. Models have been built for different scenarios : 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 all different scenarios could be computationally expensive for large dataset, but in this case the training proceeded without any problem. [1] Computational problem may emerge

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

for the case of hyper parametrisation tuning, even using multiprocessing and multithreading technics. For semplicity, in this project no model tuning has been made.
Also, for clustering have been filter every scenario, but not for the case of Hierarchical Clustering because otherwise it would generate a unreadable dendrogram. For this reason, it has been calculated the mean of all neighbourhood and room type price case.

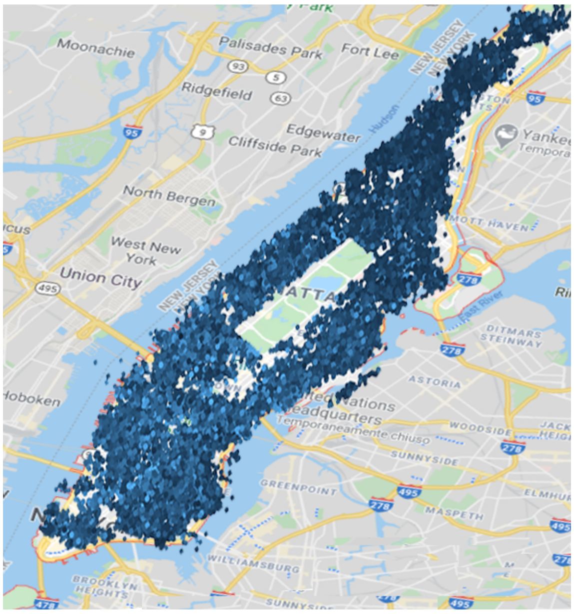


Figure 4: Approximated distribution map of all houses in Manhattan

3.2 Results

The study results are for the different groups that depend on the different subset of the dataset: entire dataset, filtering by neighbourhood and filtering by neighbourhood and room type.

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 5). The Mean Square Error (Figure 37) has a value of 0.6 which is acceptable. All the neighbourhood groups have a positive effect on the price apart for the 4^{th} one. This could be because Staten Island does not have expensive houses compared to the other groups.

Latitude and longitude have a slight negative effect on the price. Entire apartment has a slight positive effect while the shared house is negative. This could be because entire house will cost more than a shared room, so the price will be greater 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 5: Linear Regression output for the entire dataset

Linear Regression for specific neighbourhood group

Models give different results for the specific neighbourhood group (Figure 6 [2]). For the first three (Brooklyn, Manhattan and Queens) R^2 are approximately over 30%. For the MSE, the values 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. 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 confidence interval.

For the last two groups (Staten Island and Bronx) all variables do not have any significance in the response variable apart for the Entire Apartment dummy that has a positive effect.

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

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

Residuals:
    Min      1Q  Median      3Q     Max 
-1.8259 -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       5.14455    0.20932  24.577  < 2e-16 ***
longitude     -4.66795    0.22881 -20.401  < 2e-16 ***
room_type2     0.96345    0.01137  84.701  < 2e-16 ***
room_type3    -0.17115    0.03990  -4.289  1.8e-05 ***
---
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: < 2.2e-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
```

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

Residuals:
    Min      1Q  Median      3Q     Max 
-0.9120 -0.3348 -0.1459  0.2172  3.6564 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -30.688887 144.852858  -0.212    0.832    
latitude      0.734021   1.511517   0.486    0.628    
longitude    -0.001178   1.331628  -0.001    0.999    
room_type2    0.739498   0.078599   9.408   <2e-16 ***
room_type3    0.159594   0.292213   0.546    0.585    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6387 on 269 degrees of freedom
Multiple R-squared:  0.2493,	Adjusted R-squared:  0.2382 
F-statistic: 22.34 on 4 and 269 DF,  p-value: 6.126e-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 6: Linear Regression results for neighbourhood group filter

[2] In order 1 to 5 are: Brooklyn, Manhattan, Queens, Staten Island and Bronx

Linear Regression for specific neighbourhood group and room type

Adding an additional filter for the room type, models give acceptable results in some cases and not in others. (Figure 7 [3]) For the larger subset the model confirms the significance of the variables, apart for latitude in Manhattan such as happened in the model using the whole 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 because 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 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.

[3] In order n1 to n5 are: Brooklyn, Manhattan, Queens, Staten Island and Bronx and r1 to r3: Private room, Entire apartment and Shared room

```
[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     0.1990   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] "========== 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     0.4123   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: 0.8696 on 6238 degrees of freedom
Multiple R-squared:  0.07123,	Adjusted R-squared:  0.07093 
F-statistic: 239.2 on 2 and 6238 DF,  p-value: < 2.2e-16
```

```
[1] "========== n1-r3 =========="

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

Residuals:
    Min      1Q  Median      3Q     Max 
-0.4449 -0.2589 -0.1681  0.0239  3.9904 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)   
(Intercept) -319.9018   111.6378  -2.866  0.00449 **
latitude       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:  0.03883,	Adjusted R-squared:  0.03174 
F-statistic: 5.474 on 2 and 271 DF,  p-value: 0.004671
```

```
[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) -673.3536    81.3847  -8.274   <2e-16 ***
latitude      -0.5113     0.4675  -1.094    0.274    
longitude     -9.3809     0.8670 -10.820   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6845 on 5252 degrees of freedom
Multiple R-squared:  0.1039,	Adjusted R-squared:  0.1036 
F-statistic: 304.5 on 2 and 5252 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) -888.5411    89.7697  -9.898   <2e-16 ***
latitude      -0.9338     0.5683  -1.643      0.1    
longitude    -12.5366     0.9398 -13.339   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.9986 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] "========== 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 0.000581 ***
latitude        4.160      2.025   2.054 0.040769 *  
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: 0.0002622
```

```
[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   0.4346  
longitude    -0.3707     0.1963  -1.888   0.0591 .
---
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] "========== 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) 31.86387   28.50462   1.118    0.264
latitude    -0.65874    0.61449  -1.072    0.284
longitude    0.06699    0.48209   0.139    0.889

Residual standard error: 0.8305 on 1381 degrees of freedom
Multiple R-squared:  0.001545,	Adjusted R-squared:  9.897e-05 
F-statistic: 1.068 on 2 and 1381 DF,  p-value: 0.3438
```

```
[1] "========== n3-r3 =========="

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

Residuals:
    Min      1Q  Median      3Q     Max 
-0.5401 -0.2595 -0.1545  0.0127  5.0126 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)  
(Intercept) -101.808     90.766  -1.122   0.2641  
latitude       4.512      2.232   2.022   0.0453 *
longitude      1.122      1.312   0.855   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-r1 =========="

all:
m(formula = price ~ ., data = trains[[sub]])

esiduals:
    Min      1Q  Median      3Q     Max 
).4759 -0.2744 -0.1064  0.1763  2.6581 

oefficients:
           Estimate Std. Error t value Pr(>|t|)
Intercept)  88.5017   140.5864   0.630    0.530
atitude     -1.8712     1.5119  -1.238    0.218
ongitude     0.1791     1.3315   0.135    0.893

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

```
1] "========== n4-r2 =========="

all:
m(formula = price ~ ., data = trains[[sub]])

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

oefficients:
            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

esidual standard error: 0.8287 on 110 degrees of freedom
ultiple R-squared:  0.009854,	Adjusted R-squared:  -0.00814
-statistic: 0.5474 on 2 and 110 DF,  p-value: 0.58
```

```
1] "========== n4-r3 =========="

all:
m(formula = price ~ ., data = trains[[sub]])

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

oefficients:
           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

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

```
[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
latitude     -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] "========== n5-r2 =========="

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

Residuals:
    Min      1Q  Median      3Q     Max 
-1.1884 -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
latitude      1.6975     2.1824   0.778    0.437
longitude     0.2177     1.6531   0.132    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] "========== n5-r3 =========="

all:
m(formula = price ~ ., data = trains[[sub]])

esiduals:
     Min       1Q   Median       3Q      Max 
0.40836 -0.14435 -0.05595  0.06429  1.04848 

oefficients:
           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

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

Figure 7: Linear Regression output filtering by neighbourhood group and room type

3.2.2 Decison Trees Results

Decison tree without filters

For the entire dataset, the MSE has a value of 0.605 (Figure 8). To be notice is the fact that the neighbourhood groups variable have not been used in the construction of the tree. Moreover, what really influence the price is the type of room since this decision makes the first split of the tree brenches and get a positive (scaled) price or not (Figure 9).

```
[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 8: Decision Tree results on the entire dataset

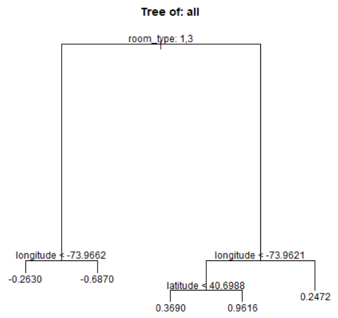


Figure 9: Tree generated for the entire dataset

Decision tree for specific neighbourhood group

For the specific neighbourhood group decision tree models 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 10).

```
[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]])
Number of terminal nodes:  4
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 10: Decision tree results for specific neighbourhood group

Decison tree for specific neighbourhood group and room type

For neighbourhood group and room price Decision tree construction was not possible for all the subsets. 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 slighly worse. This could be due to the lack of information. (Figure 11). All variables (latitude and longitude) have been used in each model.

```
[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  0.0000  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.
-2.5970 -0.3301 -0.1267  0.0000  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.
-2.3220 -0.6882 -0.2203  0.0000  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  0.0000  0.0506  3.5570
[1] 0.4992158


[1] "========== n5-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] "========== n5-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  0.0000  0.2079  3.7900
[1] 0.7622148


[1] "========== n5-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.
-0.52540 -0.15660 -0.04544  0.00000  0.04203  0.92870
[1] 0.04892947
```

```
[1] "========== n3-r1 =========="

Regression tree:
tree(formula = price ~ ., data = trains[[sub]])
Number of terminal nodes:  3
Residual mean deviance:  0.1738 = 389.1 / 2239
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.85200 -0.22640 -0.08418  0.00000  0.11440  4.94200
[1] 0.1600965


[1] "========== n3-r2 =========="

Regression tree:
tree(formula = price ~ ., data = trains[[sub]])
Number of terminal nodes:  4
Residual mean deviance:  0.6597 = 910.4 / 1380
Distribution of residuals:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-1.9190 -0.5029 -0.2189  0.0000  0.2876  4.1550
[1] 0.6590635


[1] "========== n3-r3 =========="

Regression tree:
tree(formula = price ~ ., data = trains[[sub]])
Number of terminal nodes:  6
Residual mean deviance:  0.2555 = 31.43 / 123
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-1.81800 -0.15550 -0.07597  0.00000  0.05964  3.40800
[1] 0.2989787


[1] "========== n4-r1 =========="

Regression tree:
tree(formula = price ~ ., data = trains[[sub]])
Number of terminal nodes:  12
Residual mean deviance:  0.08816 = 9.962 / 113
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.56420 -0.14540 -0.04604  0.00000  0.12440  1.31700
[1] 0.2339327


[1] "========== n4-r2 =========="

Regression tree:
tree(formula = price ~ ., data = trains[[sub]])
Number of terminal nodes:  6
Residual mean deviance:  0.5117 = 54.76 / 107
Distribution of residuals:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-1.7900 -0.3757 -0.1485  0.0000  0.3059  2.4700
[1] 0.7737648


[1] "========== n4-r3 =========="

Regression tree:
tree(formula = price ~ ., data = trains[[sub]])
Variables actually used in tree construction:
character(0)
Number of terminal nodes:  1
Residual mean deviance:  0.09365 = 0.3746 / 4
Distribution of residuals:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.2931 -0.1908 -0.1795  0.0000  0.3317  0.3317
Not possible to plot tree:  n4-r3[1] 0.5818964
```

Figure 11: 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% and a MSE of 0.57 which is acceptable. Number of trees used is 500, which is the default setting for the model.

```
[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 12: Random Forest results on the entire dataset

Random Forest for specific neighbourhood group

Random Forest on specific neighbourhood outputs overall acceptable MSE. Explained variance for Brooklyn and Manhattan is approximately around 40%, 37% for Queens and about 25% for Staten Island and Bronx. Also in this case, Manhattan MSE is higher than Brooklyn.

```
[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 13: Random Forest results for specific neighbourhood 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 plausible the case that the model does not fit the data in a adequate way due to the lack 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: 0.253230897709792"

[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: 0.229843237622333"

[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: 0.304203607022542"
```

```
[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: 0.373153844299809"

[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: 0.0567952387866339"
```

```
[1] "========== n3-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: 0.1710755
                    % Var explained: 5.65
[1] "MSE: 0.162824504191009"

[1] "========== n3-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: 0.7120831
                    % Var explained: -3.14
[1] "MSE: 0.647736155624821"

[1] "========== n3-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: 0.4336741
                    % Var explained: -9.8
[1] "MSE: 0.21647963409082"
```

Figure 14: 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 cheaper with respect to the classic Random Forest. Also for this case, the number of tree used is 500.

Ranger on the entire dataset

Ranger outputs for the entire dataset are consistent, a R^2 of 0.45 and a MSE of 0.54 which is the lowest compared to the other models.

```
[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 15: Ranger Random Forest results on the entire dataset

Ranger for specific neighbourhood group

For the specific neighbourhood group Ranger gives different results.
R^2 value greater than 0.30 for Brooklyn, Manhattan and Queens and approximately 0.2 for Bronx and Staten Island. Also in this case, MSE of Manhattan is higher with respect to Brooklyn. **Ranger for specific neighbourhood group and room type**

The dataset filtered by neighbourhood and room type outputs negative R^2 as in the case of Random Forest for the shared type of room (not for Brooklyn) and for all rooms in Bronx and Staten Island.

```
[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):       0.4348521
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:                      4224
Number of independent variables:  3
Mtry:                             1
Target node size:                 5
Variable importance mode:         none
Splitrule:                        variance
OOB prediction error (MSE):       0.3458987
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):       0.3959689
R squared (OOB):                  0.2091952
[1] "MSE: 0.270598776462825"
```

Figure 16: Ranger Random Forest results for specific neighbourhood group

3.2.5 Neural Network Results

Neural Network on the entire dataset has been run for 30 epochs. For simplicity, it has been used a single architecture for all the subsets. It has been used three dense layers of respectively 32, 16 and 1 units and Relu activation function.

Neural Networks on the entire dataset

For the entire dataset, the results are acceptable but not in line with random forest and the other regression methods. The model MAE and loss decrease during the epochs and it stabilise both for validation and training at the 15th epoch.

```
[1] "========== n1-r1 =========="
Ranger result

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

Type:                             Regression
Number of trees:                  500
Sample size:                      6724
Number of independent variables:  2
Mtry:                             1
Target node size:                 5
Variable importance mode:         none
Splitrule:                        variance
OOB prediction error (MSE):       0.1858609
R squared (OOB):                  0.01750151
[1] "MSE: 0.183555338342419"


[1] "========== n1-r2 =========="
Ranger result

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

Type:                             Regression
Number of trees:                  500
Sample size:                      6241
Number of independent variables:  2
Mtry:                             1
Target node size:                 5
Variable importance mode:         none
Splitrule:                        variance
OOB prediction error (MSE):       0.7797858
R squared (OOB):                  0.02281908
[1] "MSE: 0.805082950282569"


[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
Splitrule:                        variance
OOB prediction error (MSE):       0.2712858
R squared (OOB):                  0.02842518
[1] "MSE: 0.0886830072194648"
```

```
[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
OOB prediction error (MSE):       0.1718005
R squared (OOB):                  0.05295275
[1] "MSE: 0.162340649924091"


[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
OOB prediction error (MSE):       0.7131906
R squared (OOB):                  -0.03403514
[1] "MSE: 0.6565262152989"


[1] "========== n3-r3 =========="
Ranger result

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

Type:                             Regression
Number of trees:                  500
Sample size:                      129
Number of independent variables:  2
Mtry:                             1
Target node size:                 5
Variable importance mode:         none
Splitrule:                        variance
OOB prediction error (MSE):       0.4190039
R squared (OOB):                  -0.0526597
[1] "MSE: 0.198349635747252"
```

```
[1] "========== n2-r1 =========="
Ranger result

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

Type:                             Regression
Number of trees:                  500
Sample size:                      5255
Number of independent variables:  2
Mtry:                             1
Target node size:                 5
Variable importance mode:         none
Splitrule:                        variance
OOB prediction error (MSE):       0.4009668
R squared (OOB):                  0.2328408
[1] "MSE: 0.378149521519247"


[1] "========== n2-r2 =========="
Ranger result

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

Type:                             Regression
Number of trees:                  500
Sample size:                      8346
Number of independent variables:  2
Mtry:                             1
Target node size:                 5
Variable importance mode:         none
Splitrule:                        variance
OOB prediction error (MSE):       1.035405
R squared (OOB):                  0.04343341
[1] "MSE: 0.976143348838547"


[1] "========== n2-r3 =========="
Ranger result

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

Type:                             Regression
Number of trees:                  500
Sample size:                      317
Number of independent variables:  2
Mtry:                             1
Target node size:                 5
Variable importance mode:         none
Splitrule:                        variance
OOB prediction error (MSE):       0.5746933
R squared (OOB):                  -0.07013213
[1] "MSE: 0.741990634474665"
```

Neural Networks for specific neighbourhood group

For the specific group, the models output different results. The most populous neighbourhood such as Brooklyn and Manhattan and Queens have a decreasing MAE and loss trend. Only for Manhattan, the validation loss seems to increase approximately after the 18th epochs. For the remaining neighbourhood they have a decreasing training and validation MAE and loss, while the validation remains stable for all the epochs.

```
[1] "========== n4-r1 =========="
Ranger result

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

Type:                             Regression
Number of trees:                  500
Sample size:                      125
Number of independent variables:  2
Mtry:                             1
Target node size:                 5
Variable importance mode:         none
Splitrule:                        variance
OOB prediction error (MSE):       0.1750849
R squared (OOB):                  0.1016101
[1] "MSE: 0.134701396762548"


[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:         none
Splitrule:                        variance
OOB prediction error (MSE):       0.5325296
R squared (OOB):                  -0.1367216
[1] "MSE: 1.07190377323593"


[1] "========== n4-r3 =========="
Ranger result

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

Type:                             Regression
Number of trees:                  500
Sample size:                      5
Number of independent variables:  2
Mtry:                             1
Target node size:                 5
Variable importance mode:         none
Splitrule:                        variance
OOB prediction error (MSE):       0.1159859
R squared (OOB):                  -0.2385456
[1] "MSE: 0.576764441356849"
```

```
[1] "========== n5-r1 =========="
Ranger result

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

Type:                             Regression
Number of trees:                  500
Sample size:                      431
Number of independent variables:  2
Mtry:                             1
Target node size:                 5
Variable importance mode:         none
Splitrule:                        variance
OOB prediction error (MSE):       0.162747
R squared (OOB):                  -0.103441
[1] "MSE: 0.217155374495409"


[1] "========== n5-r2 =========="
Ranger result

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

Type:                             Regression
Number of trees:                  500
Sample size:                      251
Number of independent variables:  2
Mtry:                             1
Target node size:                 5
Variable importance mode:         none
Splitrule:                        variance
OOB prediction error (MSE):       0.7250826
R squared (OOB):                  -0.05847549
[1] "MSE: 0.88460642091214"


[1] "========== n5-r3 =========="
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:         none
Splitrule:                        variance
OOB prediction error (MSE):       0.1958126
R squared (OOB):                  -0.3441889
[1] "MSE: 0.0526173289643726"
```

Figure 17: Ranger Random Forest results for specific neighbourhood group and room type

Neural Networks for specific neighbourhood group and room type

For the different type of room and neighbourhood in some cases results are not satisfactory while in other cases are acceptable.

For the private room results for Brooklyn, Manhattan and Queens presents a decreasing training MAE and loss and a stable validation MAE and loss during the epochs. For Brooklyn validation MAE and loss remains nearby 0, for Manhattan instead, are higher compared to the training. For the last two neighbourhood starts with a very high loss and MAE train and validation, while during the epochs the value converge to a lower value near 0.5.

For the entire apartment type, for Brooklyn training MAE and loss have a decreasing trend, while validation after 12th epochs increases and get value higher than the starting point. For Manhattan instead, both train and validation decreasing trend, in particular slightly for the validation. For Queens validation MAE increases till epoch 18th and then decrease until it reaches almost the training MAE at the 30th epochs; while for the loss it continuously increase and decrease for all the epochs. For Bronx and State Island decreasing training MAE and loss, while a stable validation MAE.

For Shared rooms similar trends of validation and train for the Brooklyn, Manhattan and Queens, starting from a high value. For State Island and Bronx results are decreases very fast but reaching high values of MAE and loss. For Staten Island, train and validation start from a loss value higher than 1000, which is very high considering the dimension of this subsets.

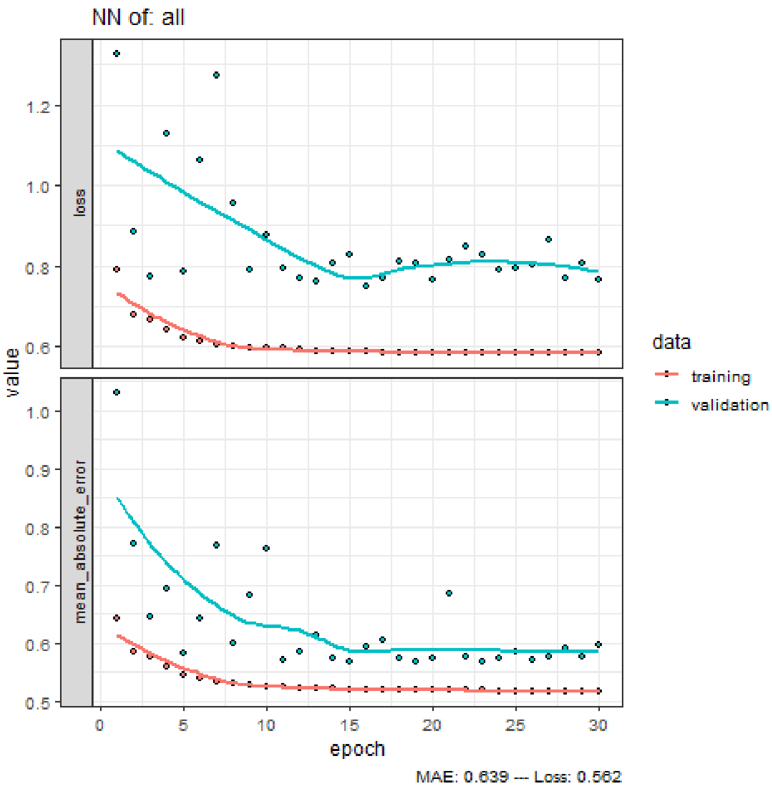


Figure 18: Neural Networks results on the entire dataset

3.2.6 K-means Results

K-means algorithm has been run on all type of subsets. Moreover, since the variables are not all numerical, it has been used a specific library called "clustMixType" which computes k-prototypes clustering for mixed-type data. It uses Euclidean distance for numerical variables and simple match coefficient. Clusters have has been computed only using k = 5 since changing the value for all the different subset will be computationally expensive.

As shown in the cluster map of the entire dataset (Figure 21) Manhattan is divided in the north and south part where this last one also includes a part of Brooklyn. This should be the case of the most expensive houses in the entire NY city. All cluster is a a partition of price zone in the entire city. As shown in the Figure 22 the distributions of prices and room type differ in the different clusters.

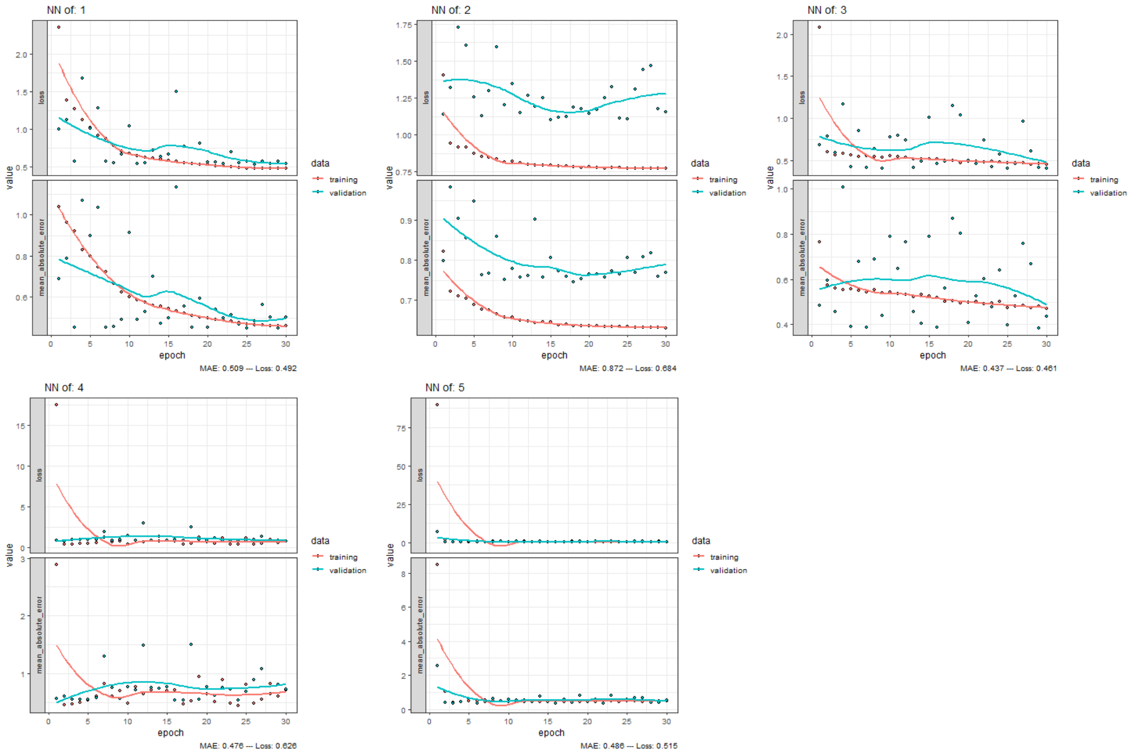
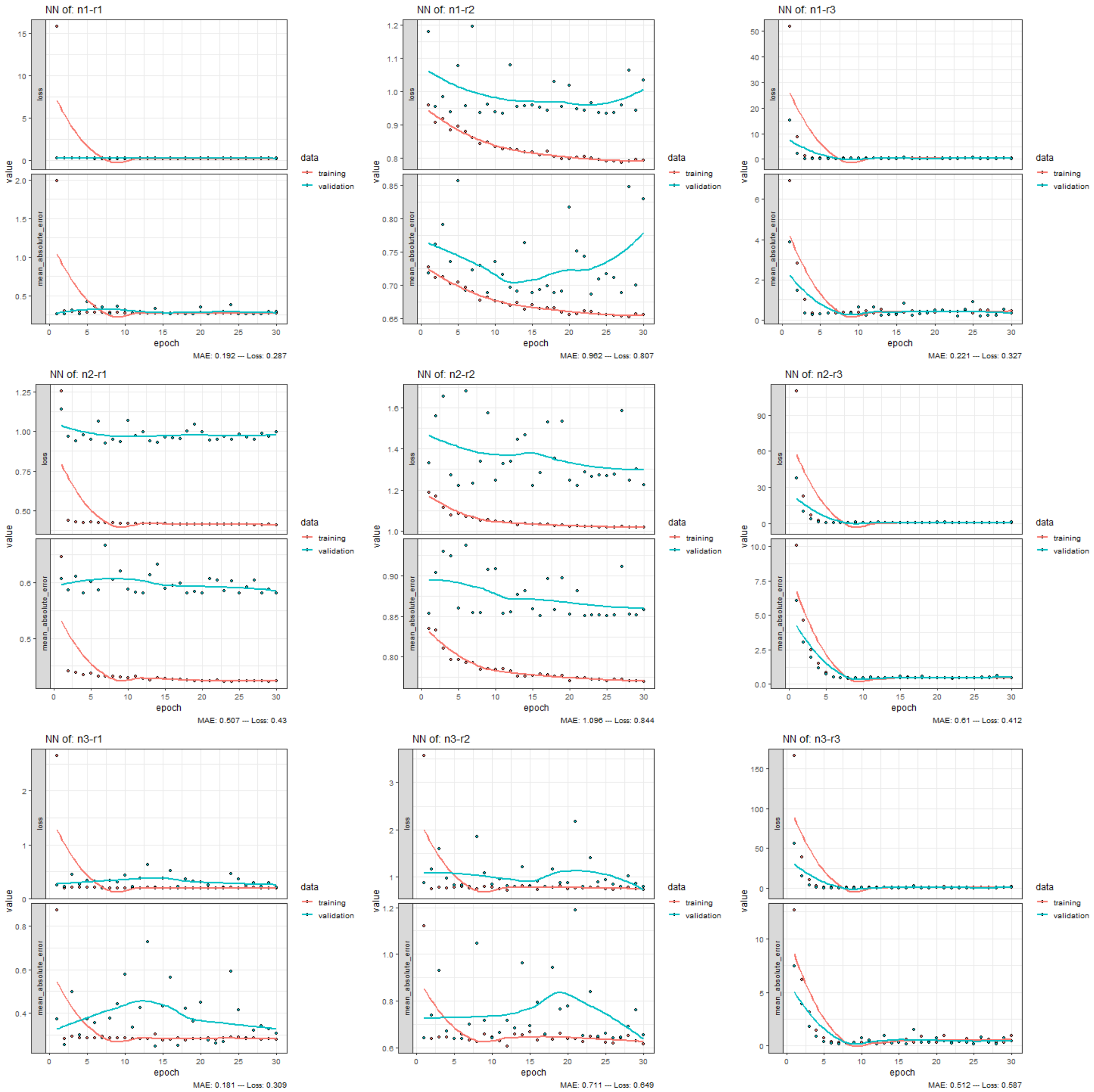


Figure 19: Neurala Networks results for specific neighbourhood group and room type

The maps plot clusters for the single neighbourhood (Figure 23) also shows how the data are distributed in the maps.

- Brooklyn has 3 main visible clusters.
- Manhattan has 2 clusters for north and south part.
- Queens instead has 4 visible cluster depending on the nearness to the seaside, Manhattan/Brooklyn and inland.
- Staten Island has spars cluster in the north-east and one in the south-west part.
- Bronx has 4 clusters visible in which 2 are mixed and the other two depends on the nearness to Manhattan since the more the houses get closer to Manhattan, the higher will be the cost.



Clusters for specific neighbourhood group and room type have a similar shape changing the room type.

- Brooklyn:
  - Private room has 4 visible clusters distinguishable from the others based on the position.
  - Entire Apartment has 4 visible cluster but less distinguishable with respect to the private rooms. One cluster has a distribution of small number of points for all the neighbourhood.
  - Shared room has 4 visible cluster, in the south is split in two parts. To be notice

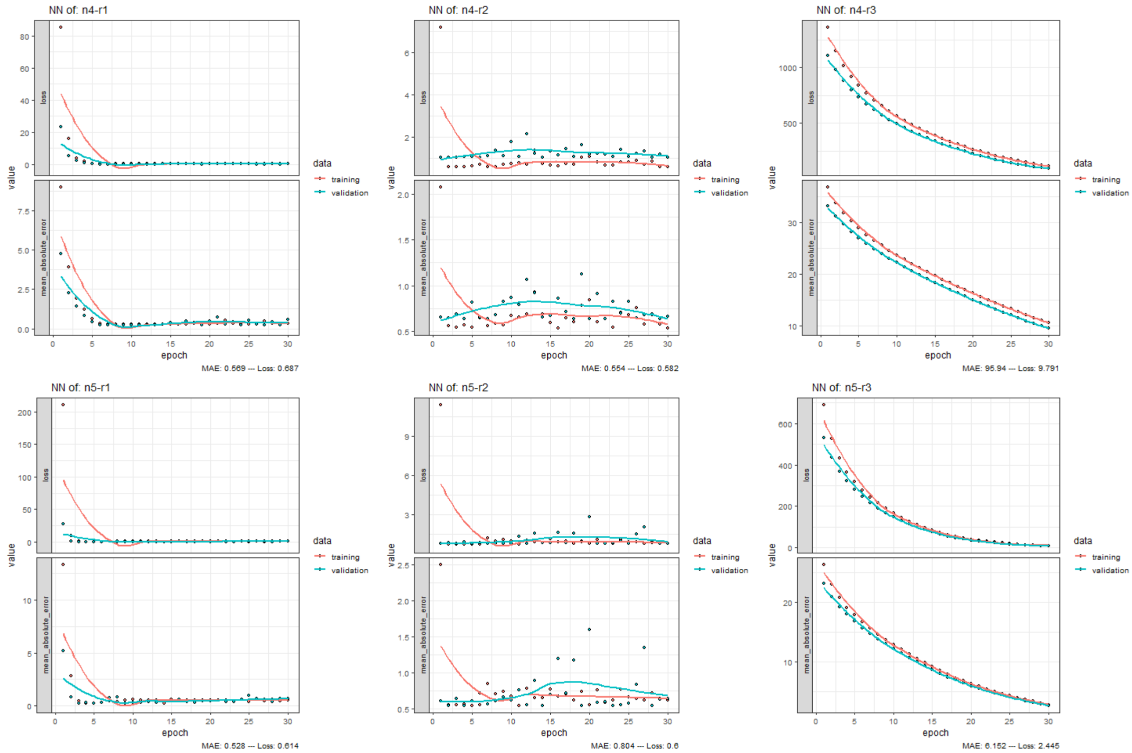


Figure 20: Neural Networks results for specific neighbourhood group and room type

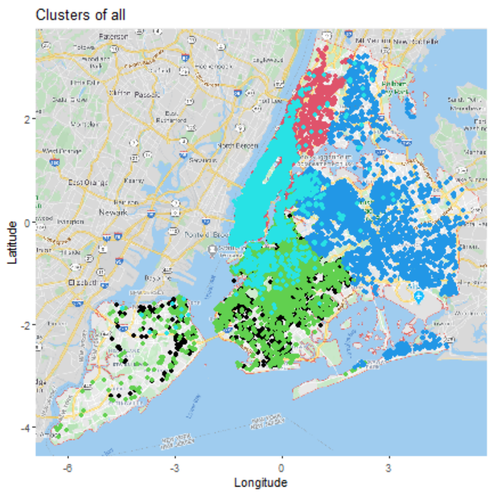


Figure 21: K-means on entire dataset

```
$`1`
   neighbourhood_group    latitude          longitude                    room_type       price
 Brooklyn      :9676    Min.   :-4.1808   Min.   :-6.3324   Private room     :1211   Min.   :-1.32482
 Manhattan     :   0    1st Qu.:-1.1986   1st Qu.:-0.6183   Entire home/apt:8758    1st Qu.:-0.37060
 Queens        : 102    Median :-0.8324   Median :-0.1418   Shared room     : 168   Median :-0.02981
 Staten Island: 359     Mean   :-0.9193   Mean   :-0.2816                           Mean   : 0.05813
 Bronx         :   0    3rd Qu.:-0.5292   3rd Qu.: 0.1727                           3rd Qu.: 0.49274
                        Max.   : 0.1840   Max.   : 3.3742                           Max.   : 2.48069

$`2`
   neighbourhood_group    latitude          longitude                      room_type         price
 Brooklyn      :9417    Min.   :-2.9821   Min.   :-0.965260   Private room     :11458   Min.   :-1.3248
 Manhattan     :   0    1st Qu.:-0.8932   1st Qu.:-0.005302   Entire home/apt: 1100    1st Qu.:-0.9386
 Queens        :3548    Median :-0.6099   Median : 0.425144   Shared room     :  407   Median :-0.8136
 Staten Island:    0    Mean   :-0.5983   Mean   : 0.786263                            Mean   :-0.7107
 Bronx         :   0    3rd Qu.:-0.2839   3rd Qu.: 1.050479                            3rd Qu.:-0.5864
                        Max.   : 1.0520   Max.   : 5.162639                            Max.   : 3.0487

$`3`
   neighbourhood_group    latitude           longitude                     room_type        price
 Brooklyn      :   0    Min.   :-0.04711   Min.   :-1.00744   Private room     :6250   Min.   :-1.3135
 Manhattan     :5666    1st Qu.: 0.88825   1st Qu.:-0.04856   Entire home/apt:1228    1st Qu.:-0.8704
 Queens        :1031    Median : 1.44008   Median : 0.19348   Shared room     : 279   Median :-0.6432
 Staten Island:    0    Mean   : 1.44737   Mean   : 0.29681                          Mean   :-0.5758
 Bronx         :1060    3rd Qu.: 1.84607   3rd Qu.: 0.52681                          3rd Qu.:-0.3706
                        Max.   : 3.36320   Max.   : 3.67901                          Max.   : 2.0149

$`4`
   neighbourhood_group    latitude          longitude                    room_type       price
 Brooklyn      : 765    Min.   :-2.7939   Min.   :-3.7011   Private room     : 397   Min.   :1.163
 Manhattan     :3940    1st Qu.:-0.1479   1st Qu.:-0.9388   Entire home/apt:4473    1st Qu.:1.538
 Queens        : 165    Median : 0.2254   Median :-0.6877   Shared room     :  25   Median :1.913
 Staten Island:    7    Mean   : 0.2373   Mean   :-0.5541                           Mean   :2.235
 Bronx         :  18    3rd Qu.: 0.6149   3rd Qu.:-0.2613                           3rd Qu.:2.765
                        Max.   : 2.8482   Max.   : 4.5849                           Max.   :4.185

$`5`
   neighbourhood_group    latitude          longitude                    room_type       price
 Brooklyn      :    0   Min.   :-0.4834   Min.   :-1.4459   Private room     :2851   Min.   :-1.3248
 Manhattan     :11271   1st Qu.:-0.0347   1st Qu.:-0.9164   Entire home/apt:8944    1st Qu.:-0.2456
 Queens        :  786   Median : 0.3240   Median :-0.7059   Shared room     : 266   Median : 0.2087
 Staten Island:     0   Mean   : 0.3886   Mean   :-0.5745                           Mean   : 0.1786
 Bronx         :    4   3rd Qu.: 0.7004   3rd Qu.:-0.3107                           3rd Qu.: 0.6063
                        Max.   : 2.2091   Max.   : 1.7230                           Max.   : 1.2879
```

Figure 22: Distribution of each cluster

that there less a smaller shared rooms on the east part with respect to the entire apartment.

- Manhattan :

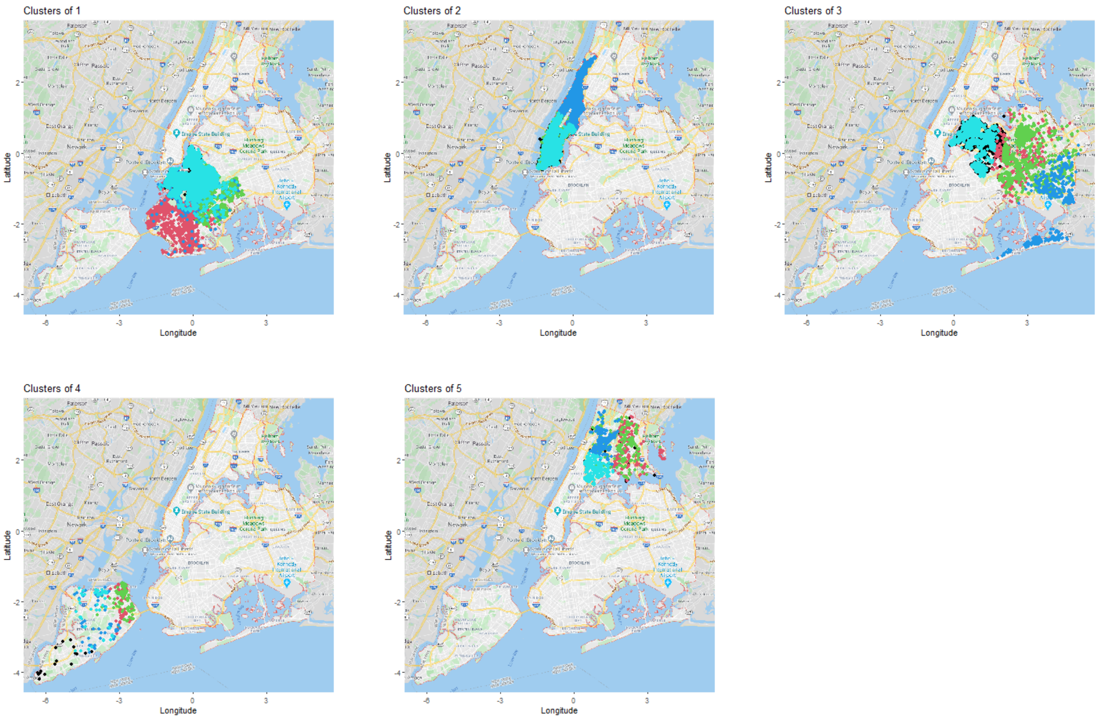
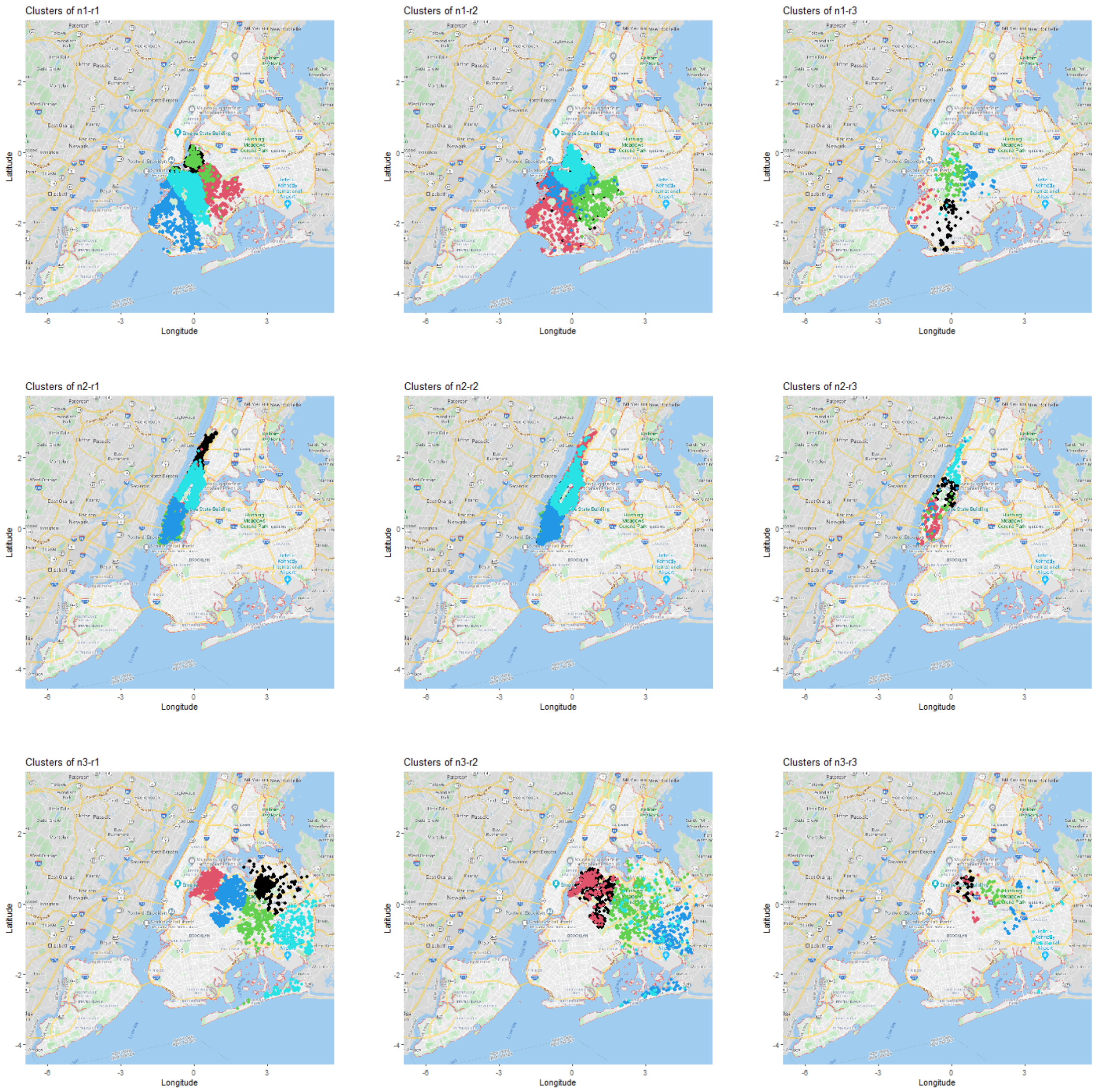


Figure 23: K-means for specific neighbourhood group

  - Private room is divided in three main clusters based on the position. This could be due to the fact that prices change in between downtown, midtown and uptown.
  - Entire apartment has similar shape as the private room, but the uptown cluster is smaller.
  - Shared room has similar shape but are less distributed in the downtown part.

- Queens:
  - Private room has a distinct shape of the 5 clusters based on the zones.
  - Entire apartment room has a mix shape of cluster in the zone nearby Brooklyn/-Manhattan and near the airport zone (south east)
  - Shared room clusters are not so much and more distributed in the inland and nearby Brooklyn/Manhattan



- Staten Island:
  - Private room are visible 4 clusters that are based on the position.
  - Entire apartment 3 greater and visible clusters, one near the border of New Jersey and Queens, one on the east coast and one more in the inland.
  - Shared room are very small number, and clustering is not very useful in this case since it is not possible to see more than 2 clusters from 4 points. Which is not very informative.
- Bronx:
  - Private room are visible 4 clusters depending on the nearness to Manhattan and

the position since the prices will change based on the neighbour.

- Entire apartment 4 clusters visible and a shape similar to the private room but with less points.
- Shared Room 4 clusters visible but low number of points.

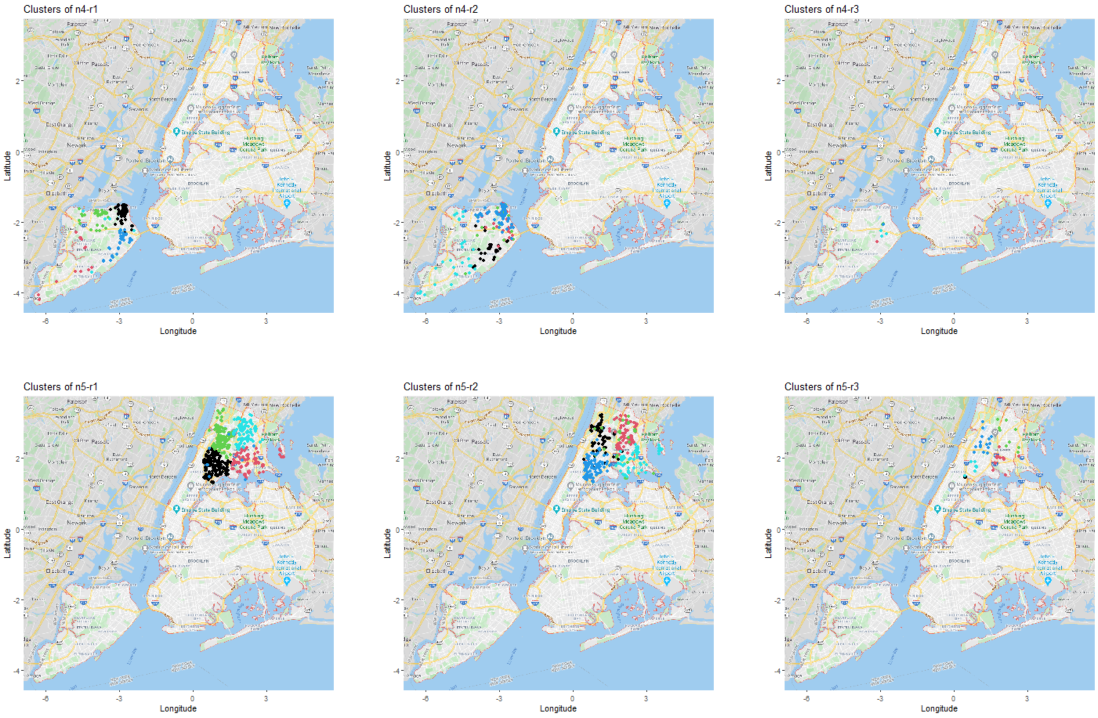


Figure 24: K-means for specific neighbourhood group and room type

Hierarchical Cluster Analysis

For this method the dataset has been changed with respect to the other. The reason is that using Hierarchical Clustering on the entire dataset will output a dendrogram in which for each house it creates a branch. This type of dendrogram is not informative. For this reason, the dataset has been aggregate using the mean of the prices depending on the case of neighbourhood or neighbourhood and room type. To compute the distances has been used the Gower measure.

As shown in Figure 25 the dendrogram is generated from the dataset that has been aggregate for mean of the neighbourhood and also for the type of room obtaining the mean of the prices for each case. From Figure 26 is possible to see how giving a certain high of 0.6 the coloured branches are those that could be consider in the same cluster.

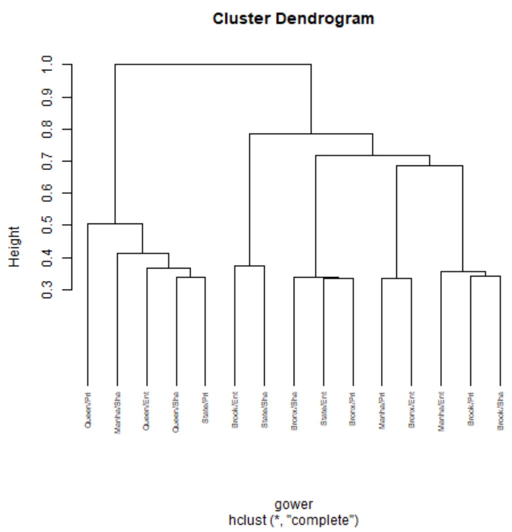


Figure 25: Hierarchical Clustering

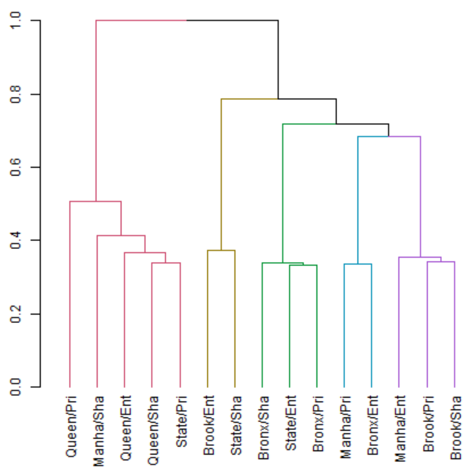


Figure 26: Hierarchical Clustering coloured

From Figure 27 is possible to see the dendrogram for the case of the mean for each neighbourhood without considering the single case for the room type. Based on the heigh level, Manhattan and Brooklyn could be consider in two different clusters, while the other three neighbourhood groups could be merged in one single cluster.

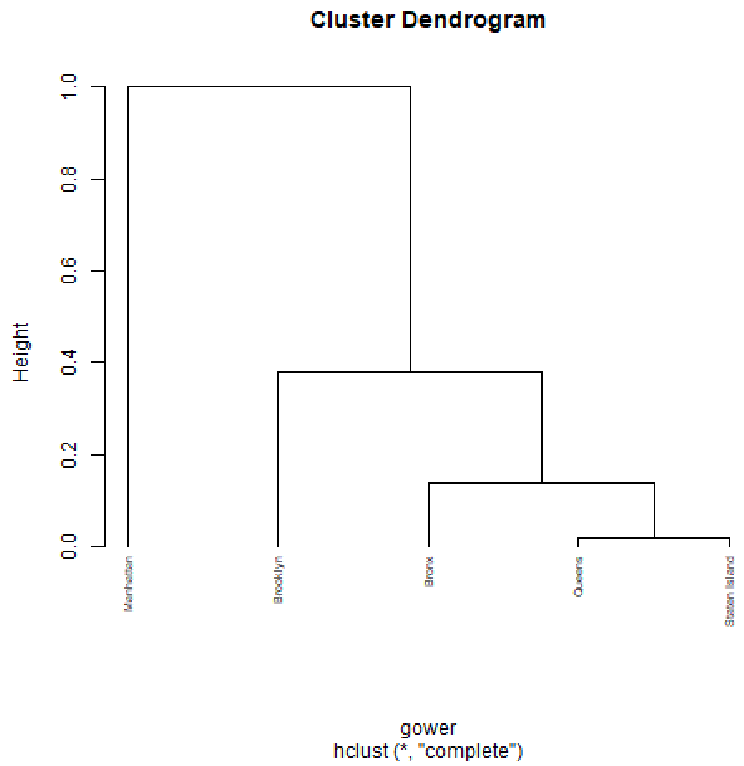


Figure 27: Hierarchical Clustering

3.2.7 Principal Component Analysis Results

PCA mixed type

PCA on mixed type of data has been applied on the entire dataset, since the neighbourhood subsets are small and there is no need to apply a dimensionality reduction. The resulting subset have only two dimensions filtering for room type and neighbourhood group.

Figure 28 shows the Eigenvalues for each dimension and the proportion of each one to the proportion and cumulative values of variance explained. Three and four dimensions correspond respectively to a cumulative variance explained of 68% and 79% which could be acceptable to get almost the explained variance of all the nine dimensions but with less dimensions.
Figure 29 shows how qualitative and quantitative variables are correlated with the different dimensions.

Factor Analysis of Mixed Data (FAMD)

	Eigenvalue	Proportion	Cumulative
dim 1	2.1303801	23.670890	23.67089
dim 2	1.8046238	20.051375	43.72227
dim 3	1.2456603	13.840670	57.56294
dim 4	1.0028761	11.143067	68.70600
dim 5	0.9971611	11.079568	79.78557
dim 6	0.9570021	10.633357	90.41893
dim 7	0.4201255	4.668061	95.08699
dim 8	0.2652652	2.947391	98.03438
dim 9	0.1769058	1.965620	100.00000

Figure 28: PCA on mixed type of data

	dim 1	dim 2	dim 3	dim 4	dim 5
latitude	-0.1972875	0.89428404	-0.2214475	-0.04100129	-0.04272996
longitude	0.7617791	0.35537824	0.4064136	0.01097103	0.01298250
price	-0.7183816	0.07289152	0.4855170	0.02253704	0.01357485

	dim 1	dim 2	dim 3	dim 4	dim 5
neighbourhood_group	0.6390071	0.871714919	0.4342793	0.5273305	0.4806940
room_type	0.3560712	0.001558062	0.3614432	0.4732362	0.5142885

Figure 29: Varible correlation with each dimension

Factor analysis is a technique that is used to reduce a large number of variables into fewer number of factors.
It has been applied on the entire dataset also in this case. From Figure 30 it is possible to see how explained variance of each dimension is changing. The results are the same of the PCA mixed type library.

Figure 31 is the scree plot that displays how much variation each principal component captures from the data.

	eigenvalue	variance.percent	cumulative.variance.percent
Dim.1	2.1303801	23.67089	23.67089
Dim.2	1.8046238	20.05138	43.72227
Dim.3	1.2456603	13.84067	57.56294
Dim.4	1.0028761	11.14307	68.70600
Dim.5	0.9971611	11.07957	79.78557

Figure 30: Eigenvalues and explained variance results

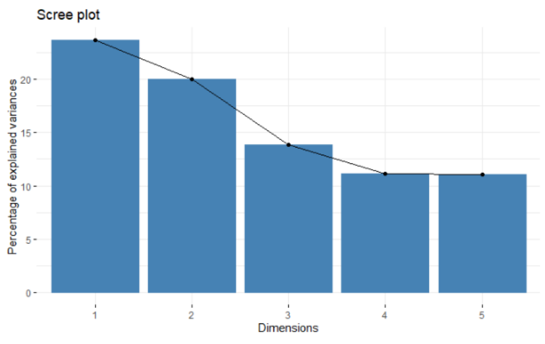


Figure 31: Scree plot

Figure 32 shows the relationship between variables, the quality of the representation of variables, as well as, the correlation between quantitative variables and the dimensions.

- price has a negative correlation with the first dimension and almost no correlation to the second one. Price also has a low quality of representation on the factor map.
- latitude has a strong correlation to the second dimensions and almost no correlation to the first one. Latitude has an high quality of representation.
- longitude has a strong correlation on the first dimension and a modest on the second one. Longitude has a medium quality of representation.

Figure 33 shows the qualitative variable contribution to each dimension:

- neighborhood group with high number of houses has the highest quality of representation, while Bronx and Staten Island has low contribution.
- room types have low contribution with respect to the neighbourhood but Entire home and Private room has more than Shared room that in the plot appears to be the worst variable in term of contribution.

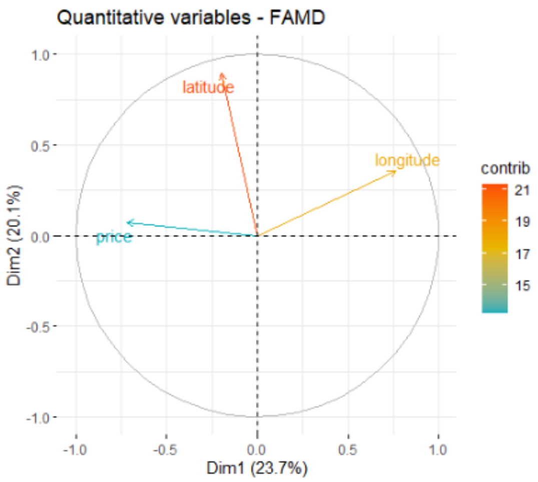


Figure 32: Quantitative variable contribution

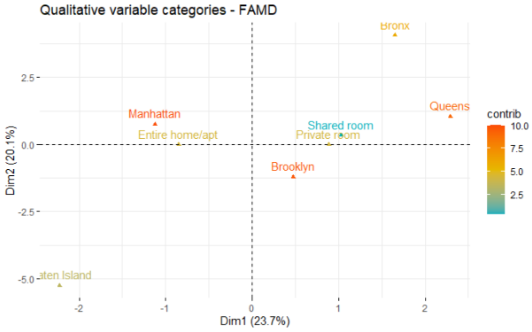


Figure 33: Qualitative variable contribution

Figure 34 shows in the first group the coordinates of the variable; the second group represent the Cos2, which is the quality of representation on the factor map; the third values are the contribution of each variables to the dimensions.

Figure 35 displays the summary of the correlation of all the variables to the principal dimensions. Latitude appears to be the one most correlated with the second dimension, while longitude and price to the first one. Neighbourhood groups appears to be the most correlated with dimension one and two. Room type is modestly correlated with dimension one and not correlated to dimension two.

```
                      Dim.1       Dim.2      Dim.3        Dim.4        Dim.5
latitude          0.03892236 0.799743946 0.04903898 0.0016811055 0.0018258497
longitude         0.58030735 0.126293692 0.16517205 0.0001203636 0.0001685452
price             0.51607212 0.005313173 0.23572672 0.0005079180 0.0001842764
neighbourhood_group 0.63900709 0.871714919 0.43427934 0.5273304673 0.4806939587
room_type         0.35607121 0.001558062 0.36144319 0.4732362202 0.5142884950
                      Dim.1        Dim.2       Dim.3        Dim.4        Dim.5
latitude          0.00151495 6.395904e-01 0.002404821 2.826116e-06 3.333727e-06
longitude         0.33675663 1.595010e-02 0.027281805 1.448739e-08 2.840749e-08
price             0.26633043 2.822981e-05 0.055567086 2.579807e-07 3.395780e-08
neighbourhood_group 0.10208251 1.899717e-01 0.047149637 6.951936e-02 5.776667e-02
room_type         0.06339335 1.213779e-06 0.065320588 1.119763e-01 1.322463e-01
                     Dim.1       Dim.2     Dim.3       Dim.4       Dim.5
latitude           1.827015 44.31638047  3.936786  0.16762844  0.18310478
longitude         27.239615  6.99833910 13.259799  0.01200184  0.01690250
price             24.224415  0.29441998 18.923837  0.05064614  0.01848011
neighbourhood_group 29.994980 48.30452324 34.863386 52.58181750 48.20624738
room_type         16.713975  0.08633722 29.016193 47.18790608 51.57526523
```

Figure 34: Coordinates,quality of representation and contribution for each variable

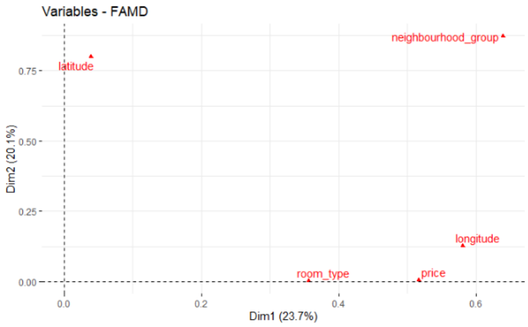


Figure 35: Summary of the correlation of all the variables to the first two dimensions

Figure 37 represents how the contribution of the variables effect each five dimension selected with an expected average value of 20%. It is possible to notice that the contributions are not uniform. Latitude does not contribute to the first dimension while price and room type for the second one. Longitude, latitude and price almost do not contribute to the fourth and fifth dimensions, this means that only qualitative variables effect these two dimensions.

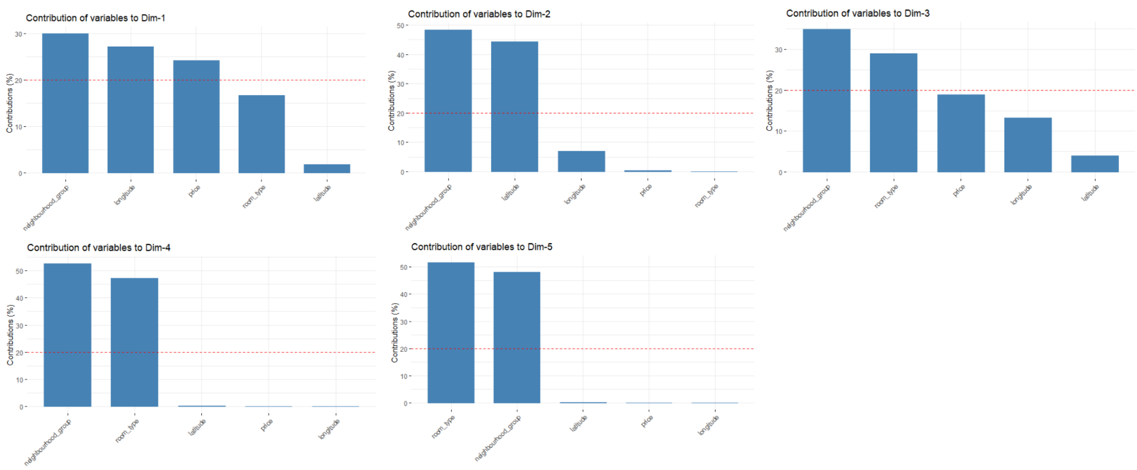


Figure 36: Contribution of all the variables to each dimension

4

Conclusion

4.1 Price prediction

Subset	Linear.Regression	Decision.Tree	Random.Forest	Ranger.Random.Forest	Neural.Networks	Best
Brooklyn	0.4351730	0.4395665	0.4183596	0.4138488	0.4861140	Ranger.Random.Forest
Manhattan	0.7714922	0.7822847	0.7318788	0.7328182	0.8628053	Random.Forest
Queens	0.3846665	0.3864543	0.3599257	0.3584391	0.3869984	Ranger.Random.Forest
Staten Island	0.4776541	0.5173908	0.4844570	0.4944066	1.3982809	Linear.Regression
Bronx	0.3429405	0.3420752	0.3271730	0.3347552	0.4838151	Random.Forest
Brooklyn - Private room	0.1858138	0.1806116	0.1928601	0.1934420	0.2288532	Decision.Tree
Brooklyn - Entire home/apt	0.7598732	0.7503308	0.8049458	0.8051254	0.8224091	Decision.Tree
Brooklyn - Shared room	0.2639579	0.2505339	0.2396742	0.2386900	0.2743721	Ranger.Random.Forest
Manhattan - Private room	0.4995290	0.4364956	0.4194580	0.4195132	0.6584851	Random.Forest
Manhattan - Entire home/apt	0.9842757	0.9937341	1.0173741	1.0180296	1.0683964	Linear.Regression
Manhattan - Shared room	0.5616574	0.5452982	0.5909831	0.5856787	0.6905749	Decision.Tree
Queens - Private room	0.1680487	0.1600965	0.1624387	0.1618092	0.1700685	Decision.Tree
Queens - Entire home/apt	0.6817353	0.6590635	0.6492862	0.6486996	0.9550785	Ranger.Random.Forest
Queens - Shared room	0.6218959	0.6218172	0.6305342	0.6338897	0.8929396	Decision.Tree
Staten Island - Private room	0.1241196	0.1802672	0.1337671	0.1338837	0.2002601	Linear.Regression
Staten Island - Entire home/apt	0.4319694	0.4368075	0.4946490	0.4930049	0.4310423	Neural.Networks
Staten Island - Shared room	0.1199132	0.2241880	0.0839624	0.2295022	0.1028918	Random.Forest
Bronx - Private room	0.2146868	0.2159434	0.2156149	0.2159356	0.6114893	Linear.Regression
Bronx - Entire home/apt	0.7956436	1.0693051	0.9520994	0.9670430	0.9997671	Linear.Regression
Bronx - Shared room	0.0690389	0.0489295	0.0460020	0.0500036	0.0699948	Random.Forest
All	0.6045975	0.5977854	0.5670718	0.5373394	0.6177393	Ranger.Random.Forest

Figure 37: Summary of the MSE for all the models for each subset of the dataset

The first objective has been accomplished.

Figure 37 is possible to see all the Mean Square Error for each model and case.
For the entire dataset, the best performance come from the Ranger Random Forest with the lowest MSE value. Neural Network are the worst, this could be due to the fact that has not been done any hyper parametrisation tuning since a single fixed architecture has been used.
For the specific neighbourhood groups, Ranger Random Forest and Random Forest has similar predictive performance and they perfomed the best with respect to the others. Only for Staten Island the Linear regression has better performance but similar to the two Random Forest.
For the specific neighbourhood groups and room type models perform differently. It is not possible to determined the best choice since each subset has different characteristics. According to all the results Random Forest could maybe be the most suitable in general but it is computationally the more expensive one, so Ranger should be a better substitute.

For future tests, tuning of the Neural Networks (in general for all the model) could increase the price prediction precision but it is require to do not ignore the computational cost.

4.2 Clusters and groups

The second objective has been accomplished.

Cluster definitions appeared to be consistent, price and position determined the main characteristics of the different group and it is possible to give a meaning to each group. Only for some specific subsets such as Staten Island and Bronx Shared room results are not consistent, but this is due to the fact that those subsets contain low information. Hierarchical Cluster Analysis give also consistent results based on the price mean for each neighbourhood group and room type. This solution is an approximation, since giving the base dataset with 48000 in input to the Hierarchical Clustering, the resulting dendrogram is unreadable since for each row it generate a branch.

Principal component analysis gives a lot of information about the variables and their contribution on the quality of the information and the correlation with the different components.

5

Appendix

The File ”project-code.Rmd” is attached and contains the code.