AirBnB New York Analysis

Finn Bartels

1/8/2021

Introduction

In this report we will have a look on **AirBnB** data of New York from 2019. This will include several informations about listings, their hosts, conditions and reviews.

1. Overview
    - A look on our data
    - Goals
2. Analysis and methods
    - Data cleaning and exploration
    - Data visualization
    - Modeling approach
3. Results
4. Conclusion

1. Overview

A look on our data

For this project a dataset of AirBnB listings in New York from https://www.kaggle.com/ was used which is originally from the website http://insideairbnb.com/.
I worked with data from 2019 including informations about New York based AirBnB listings. You can find the data in this github.

```
AirData <- read.csv("AirBnB_NYC.csv")
```

There are several informations about an offering like the name of the offering, host informations, in which neighbourhood the accommodation is, the price per night, amount of reviews and the availability for the year.

We can see the columns and values here:

```
str(AirData)
```

```
## 'data.frame':    48895 obs. of  16 variables:
##  $ id                            : int  2539 2595 3647 3831 5022 5099 5121 5178 5203 5238 ...
##  $ name                          : chr  "Clean & quiet apt home by the park" "Skylit Midtown Castle"
##  $ host_id                       : int  2787 2845 4632 4869 7192 7322 7356 8967 7490 7549 ...
##  $ host_name                     : chr  "John" "Jennifer" "Elisabeth" "LisaRoxanne" ...
##  $ neighbourhood_group           : chr  "Brooklyn" "Manhattan" "Manhattan" "Brooklyn" ...
##  $ neighbourhood                 : chr  "Kensington" "Midtown" "Harlem" "Clinton Hill" ...
##  $ latitude                      : num  40.6 40.8 40.8 40.7 40.8 ...
##  $ longitude                     : num  -74 -74 -73.9 -74 -73.9 ...
##  $ room_type                     : chr  "Private room" "Entire home/apt" "Private room" "Entire home/
##  $ price                         : int  149 225 150 89 80 200 60 79 79 150 ...
##  $ minimum_nights                : int  1 1 3 1 10 3 45 2 2 1 ...
##  $ number_of_reviews             : int  9 45 0 270 9 74 49 430 118 160 ...
##  $ last_review                   : chr  "2018-10-19" "2019-05-21" "" "2019-07-05" ...
##  $ reviews_per_month             : num  0.21 0.38 NA 4.64 0.1 0.59 0.4 3.47 0.99 1.33 ...
##  $ calculated_host_listings_count: int  6 2 1 1 1 1 1 1 1 4 ...
##  $ availability_365              : int  365 355 365 194 0 129 0 220 0 188 ...
```

Goals

With this report I want to work with different analysis and methods to predict whether a listed accommodation was in Manhattan or in another neighbourhood group. For this purpose I ran logistic regression, a decision tree and a random forest.

2. Analysis and methods

Data cleaning and exploration

First step is deleting columns host_name and last_review as we don't need them. Next step is cleaning the data of NA values. We can see NA values in "reviews_per_month", so we change these values to 0.
This step has not much influence on the distribution between the neighbourhod_groups as we can see at the distribution of NA values (revDist):

```
##   n()
## 1   0
```

```
##   n()
## 1   0
```

```
##   n()
## 1   0
```

```
##   n()
## 1   0
```

```
##   n()
## 1   0
```

```
##   n()
## 1   0

##   n()
## 1   0

##   n()
## 1   0

##   n()
## 1   0

##   n()
## 1   0

##   n()
## 1   0

##     n()
## 1 10052

##   n()
## 1   0

##   n()
## 1   0

## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 5 x 3
##   neighbourhood_group revMean revDist
##   <chr>                 <dbl>   <dbl>
## 1 Bronx                 1.48    0.134
## 2 Brooklyn              1.05    0.173
## 3 Manhattan             0.977   0.238
## 4 Queens                1.57    0.123
## 5 Staten Island         1.58    0.100
```

Some columns had to be factorized or changed to date.

```
AirData$neighbourhood_group <- as.factor(AirData$neighbourhood_group)
AirData$neighbourhood <- as.factor(AirData$neighbourhood)
AirData$room_type <- as.factor(AirData$room_type)
```

As we want to use different methods to predict an offering to be located in manhattan or not, we will add this to a column:

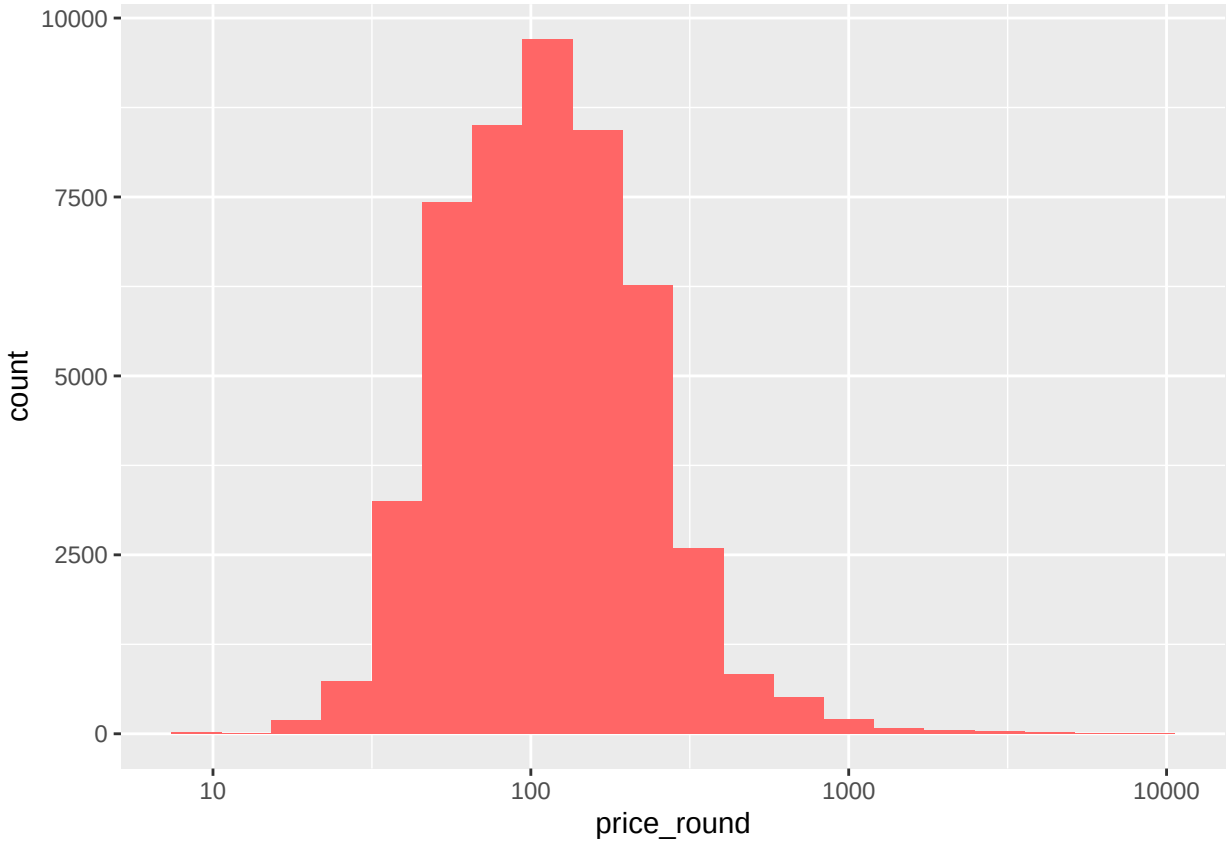
```
AirData <- AirData %>%
  mutate(manhattan=ifelse(neighbourhood_group=="Manhattan",
                          "manhattan",
                          "not_manhattan"))
AirData$manhattan <- as.factor(AirData$manhattan)
```

We will review some of the most important values for the models we will work with.
Let us have a look at the distribution of the prices:

```
AirData %>%
  mutate(price_round = round(price, digits=-1)) %>%
  group_by(price_round) %>%
  ggplot(aes(price_round)) +
  geom_histogram(bins = 20, fill = "#FF6666") +
  scale_x_continuous(trans="log10")
```

```
## Warning: Transformation introduced infinite values in continuous x-axis
```

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




```
quantile(AirData$price, probs=c(0.01, 0.10, 0.25, 0.50, 0.75, 0.90, 0.99))
```

```
##  1% 10% 25% 50% 75% 90% 99%
##  30  49  69 106 175 269 799
```

You can see that there are 0.5% of the data under 26 Dollars and 0.5% of the data over 1000 Dollars. We define the values over 1000 Dollars as outliers and remove them from the dataframe.
Looking at the offers <= 26 Dollars we can see that some listings are at 0 Dollar. We define these as outliers too.

```r
AirData %>%
  filter(price <= 26) %>%
  group_by(price) %>%
  arrange(price) %>%
  summarize(n())
```

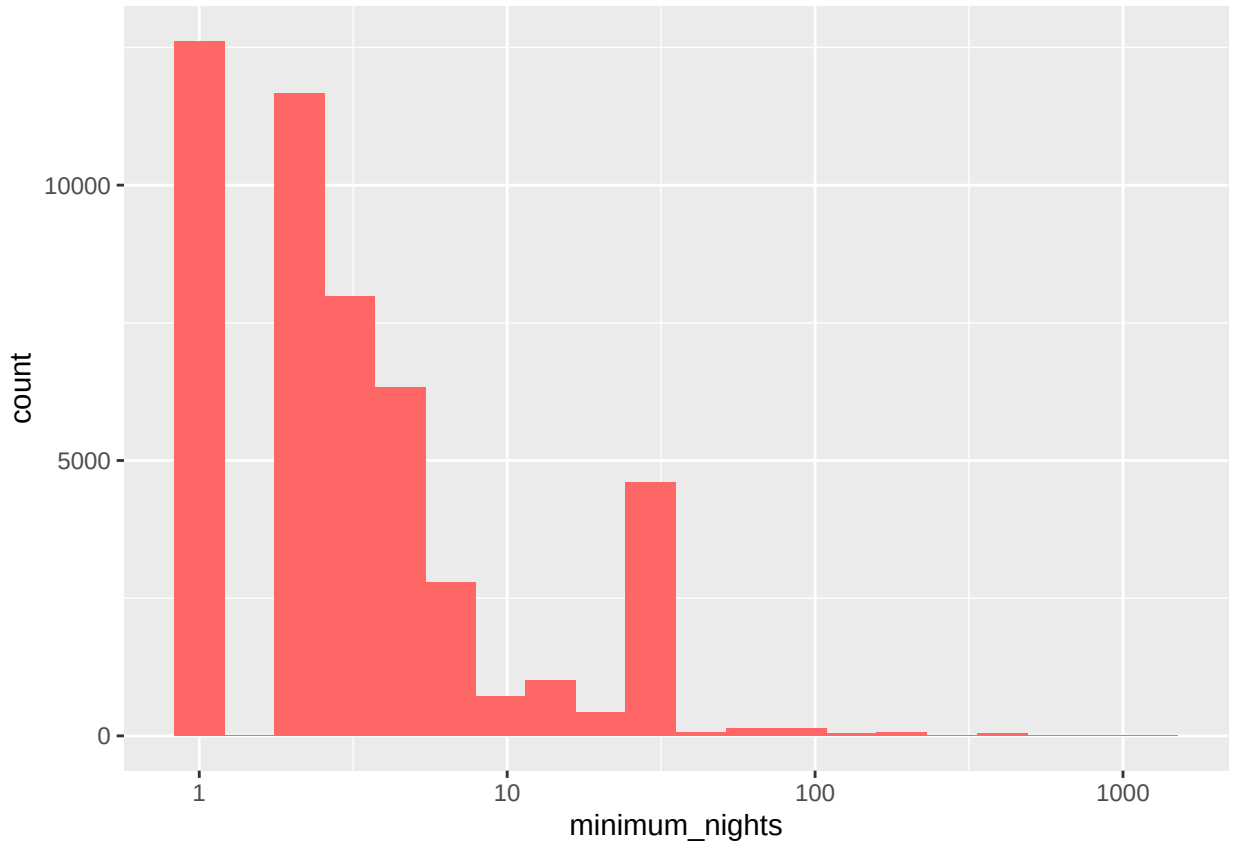
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 16 x 2
##    price `n()`
##    <int> <int>
##  1     0    11
##  2    10    17
##  3    11     3
##  4    12     4
##  5    13     1
##  6    15     6
##  7    16     6
##  8    18     2
##  9    19     4
## 10    20    33
## 11    21     6
## 12    22    16
## 13    23     7
## 14    24    13
## 15    25   100
## 16    26    24
```

```r
AirData <- AirData %>%
  filter(price > 0 & price <= 1000)
```

We also shorten the minimum nights to maximum 365 nights, defining more than one year as outliers:

```r
AirData %>%
  ggplot(aes(minimum_nights)) +
  geom_histogram(bins = 20, fill = "#FF6666") +
  scale_x_log10()
```



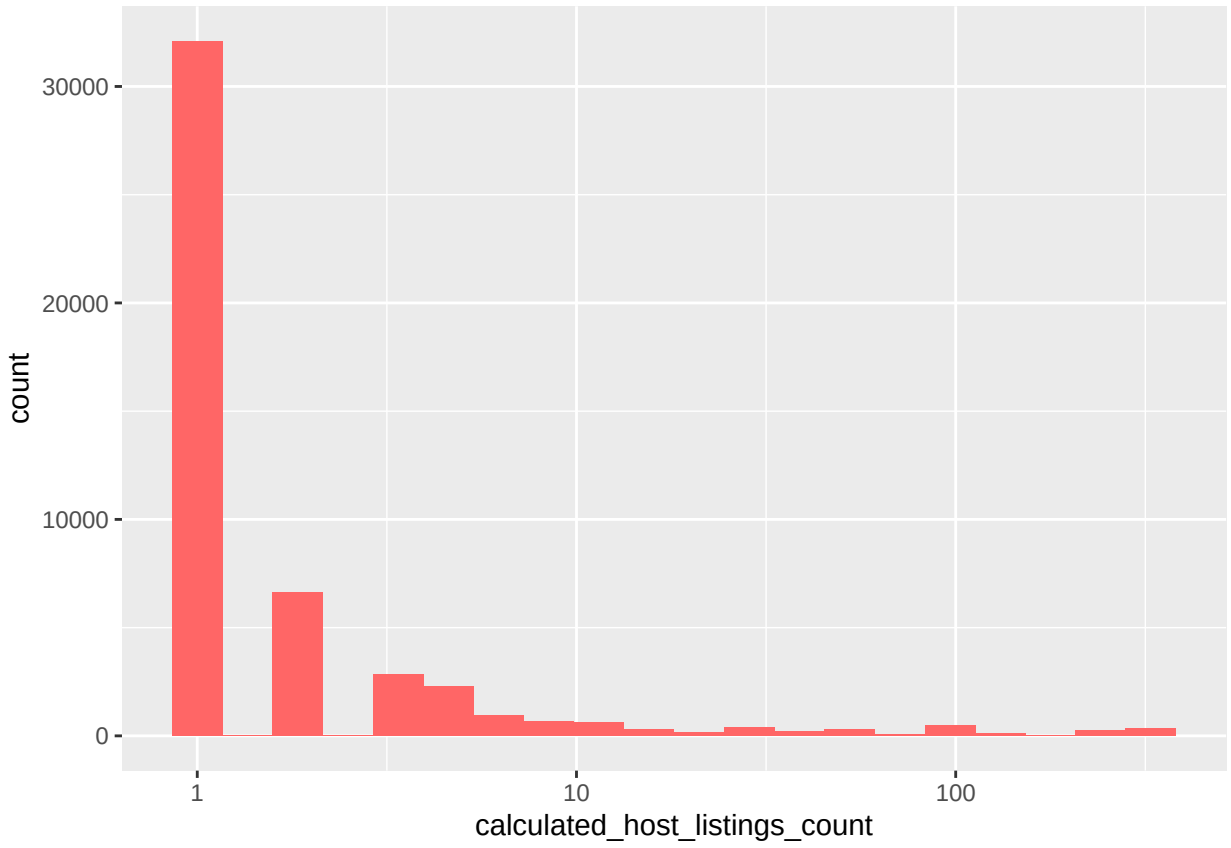

```
quantile(AirData$minimum_nights, probs=c(0.005, 0.10, 0.25, 0.50, 0.75, 0.90, 0.995))
```

```
##  0.5%   10%   25%   50%   75%   90% 99.5%
##     1     1     1     3     5    28    90
```

```
AirData <- AirData %>%
  filter(minimum_nights < 365)
```

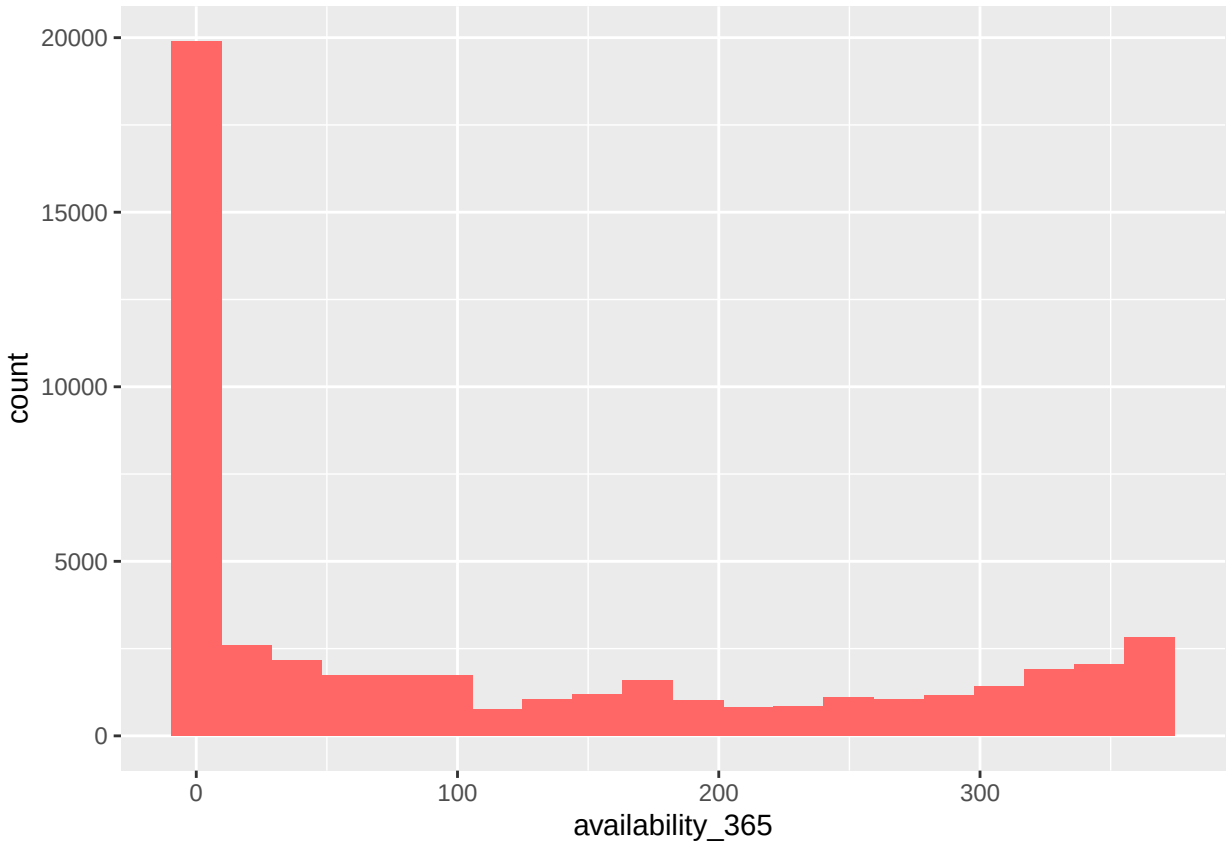
The calculated host listings tell us how many listings each host has:

```
AirData %>%
  ggplot(aes(calculated_host_listings_count)) +
  geom_histogram(bins = 20, fill = "#FF6666") +
  scale_x_log10()
```



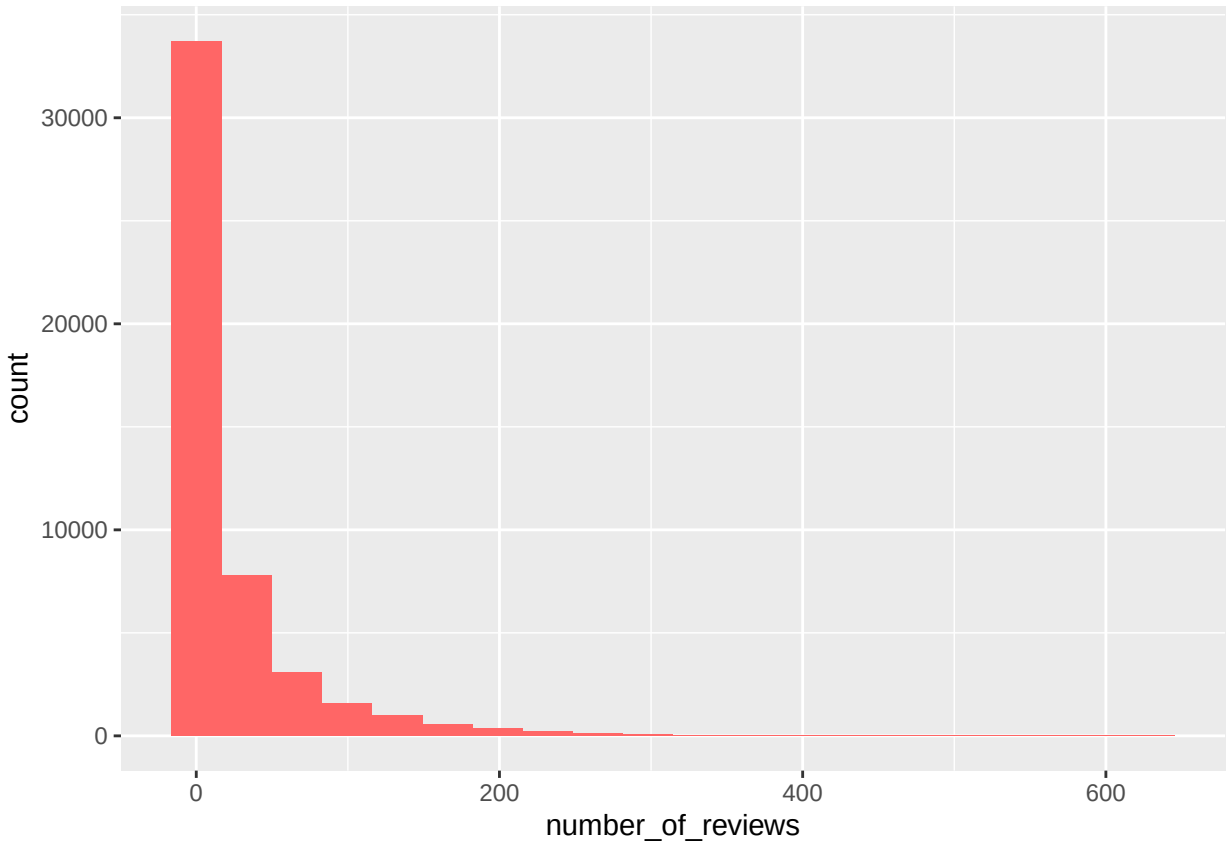
We notice that most apartments have a small number of days when the listing is available for booking.

```
AirData %>%
  ggplot(aes(availability_365)) +
  geom_histogram(bins = 20, fill = "#FF6666")
```



Also the number of reviews is near 0 for most listings:

```
AirData %>%
  group_by(number_of_reviews) %>%
  ggplot(aes(number_of_reviews)) +
  geom_histogram(bins = 20, fill = "#FF6666")
```



For the next steps we set a seed and split the data into train and test set with a distribution of 80/20. I have chosen this split because the original dataset is large enough to work with a test set of 20% and forcing an accurate variance too. Furthermore the results get more accurate by training a larger train set instead of training only 50%.

```
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
```

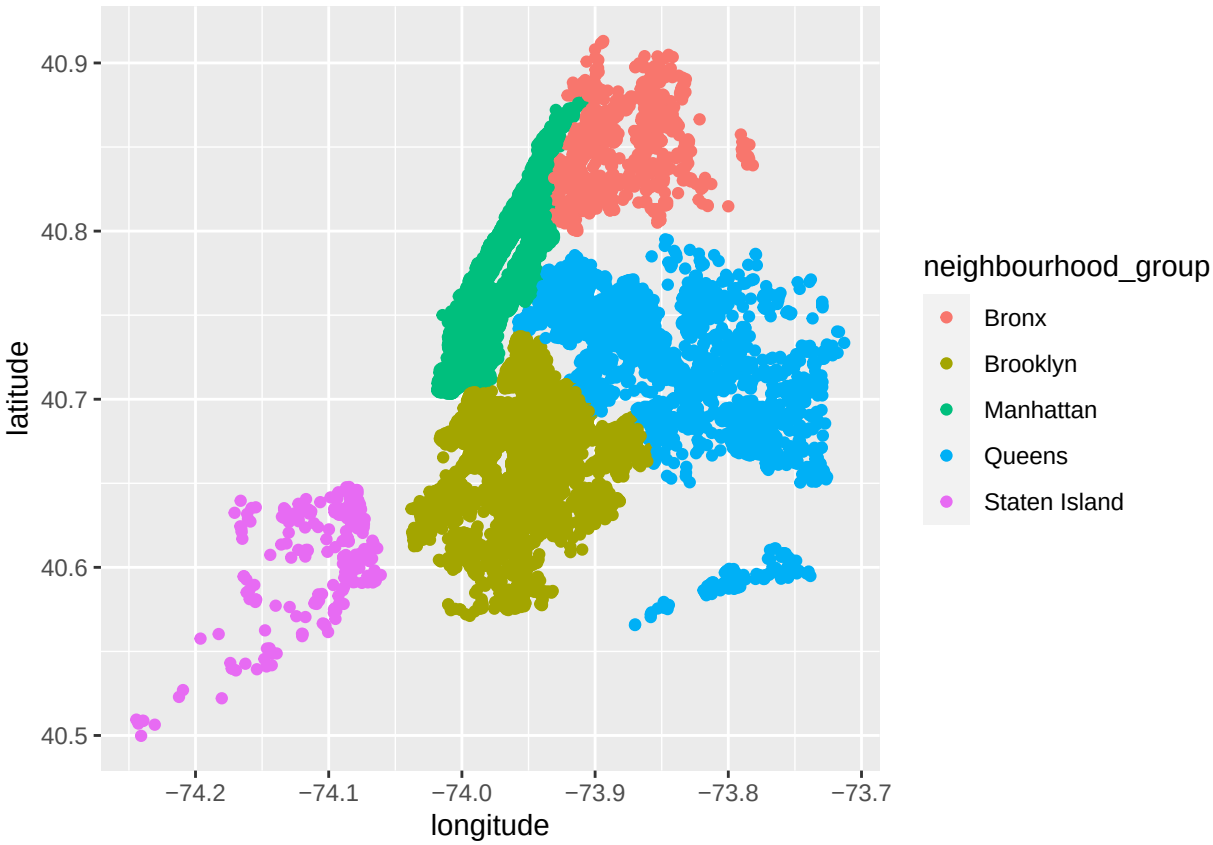
```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
test_index <- createDataPartition(y = AirData$neighbourhood_group, times = 1, p = 0.2, list = FALSE)
train_set <- AirData[-test_index,]
test_set <- AirData[test_index,]
```

data visualization

Now let's get some insights of the data.
We will group by the Neighbourhood_group and plot the longitude and latitude.

```
train_set %>%
  ggplot(aes(longitude, latitude, col=neighbourhood_group)) +
  geom_point()
```



We can see that the offers are perfectly grouped inside their neighbourhoods and that we have no outliers geographically.

The offerings are not evenly distributed on the neighbourhood_groups

```
train_set %>%
  group_by(neighbourhood_group) %>%
  summarize(offerings=n())
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 5 x 2
##   neighbourhood_group offerings
##   <fct>                   <int>
## 1 Bronx                     870
## 2 Brooklyn                16020
## 3 Manhattan               17173
## 4 Queens                   4523
## 5 Staten Island             296
```

just like the mean prices, while the median price is much lower than the mean price:

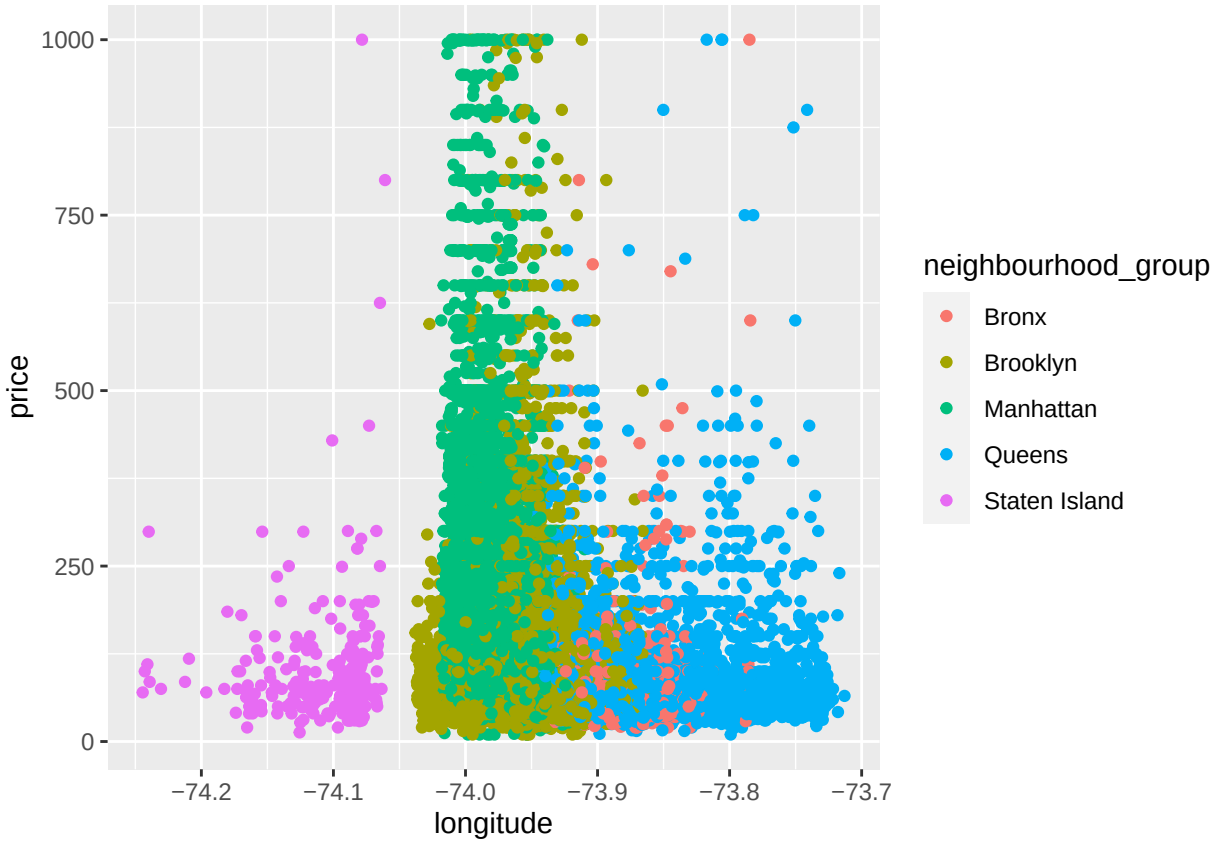
```
train_set %>%
  group_by(neighbourhood_group) %>%
  summarize(mean_price=mean(price), med_price=median(price))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 5 x 3
##   neighbourhood_group mean_price med_price
##   <fct>                     <dbl>     <dbl>
## 1 Bronx                      86.4        65
## 2 Brooklyn                  117.         90
## 3 Manhattan                 180.        150
## 4 Queens                     96.1        74
## 5 Staten Island              98.4        75
```

We can also notice the differences in price by plotting price for longitudes:

```
train_set %>%
  ggplot(aes(longitude, price, col=neighbourhood_group)) +
  geom_point()
```



Using smoothing we obtain a plot that shows us how the price behaves in relation to the longitude (the two vertical lines mark the begin and end of manhattan horizontally):

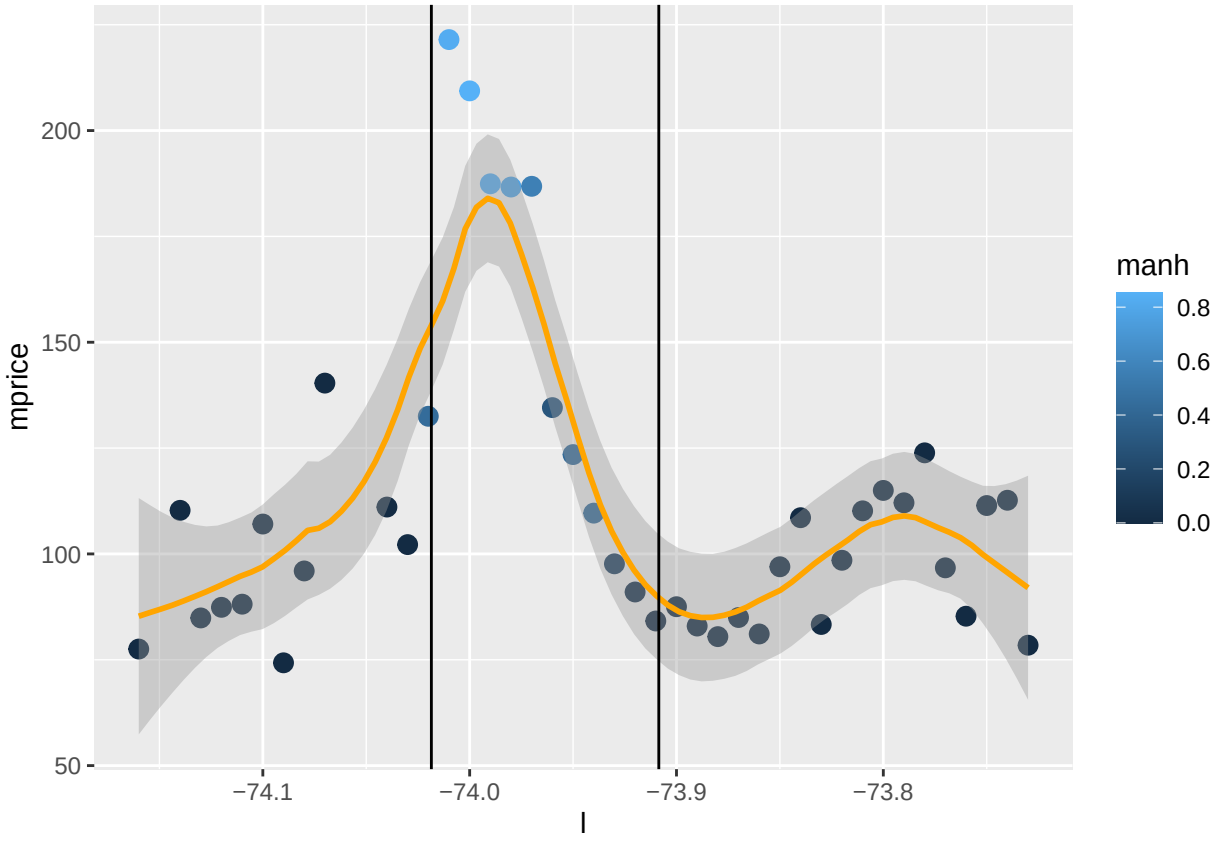
```
train_set_mprice <- train_set %>%
  mutate(l=round(longitude, digits=2)) %>%
  group_by(l) %>%
  summarize(n=n(),
            mprice=mean(price),
            manh=mean(neighbourhood_group=="Manhattan")) %>%
  filter(n>=10)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
min_Manh_long <- min(train_set %>% filter(neighbourhood_group=="Manhattan") %>% select(longitude))
max_Manh_long <- max(train_set %>% filter(neighbourhood_group=="Manhattan") %>% select(longitude))

train_set_mprice %>%
  ggplot(aes(l, mprice, col=manh)) +
  geom_point(size=3) +
  geom_smooth(color="orange",
              span = 0.25,
              method = "loess",
              method.args = list(degree=1)) +
  geom_vline(xintercept = min_Manh_long) +
  geom_vline(xintercept = max_Manh_long)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



There are three different room types: Apartment, private room and shared room.
In every neighbourhood except of Manhattan there are more private room offerings than apartment offerings and in every neighbourhood group there are only a few shared room offerings:

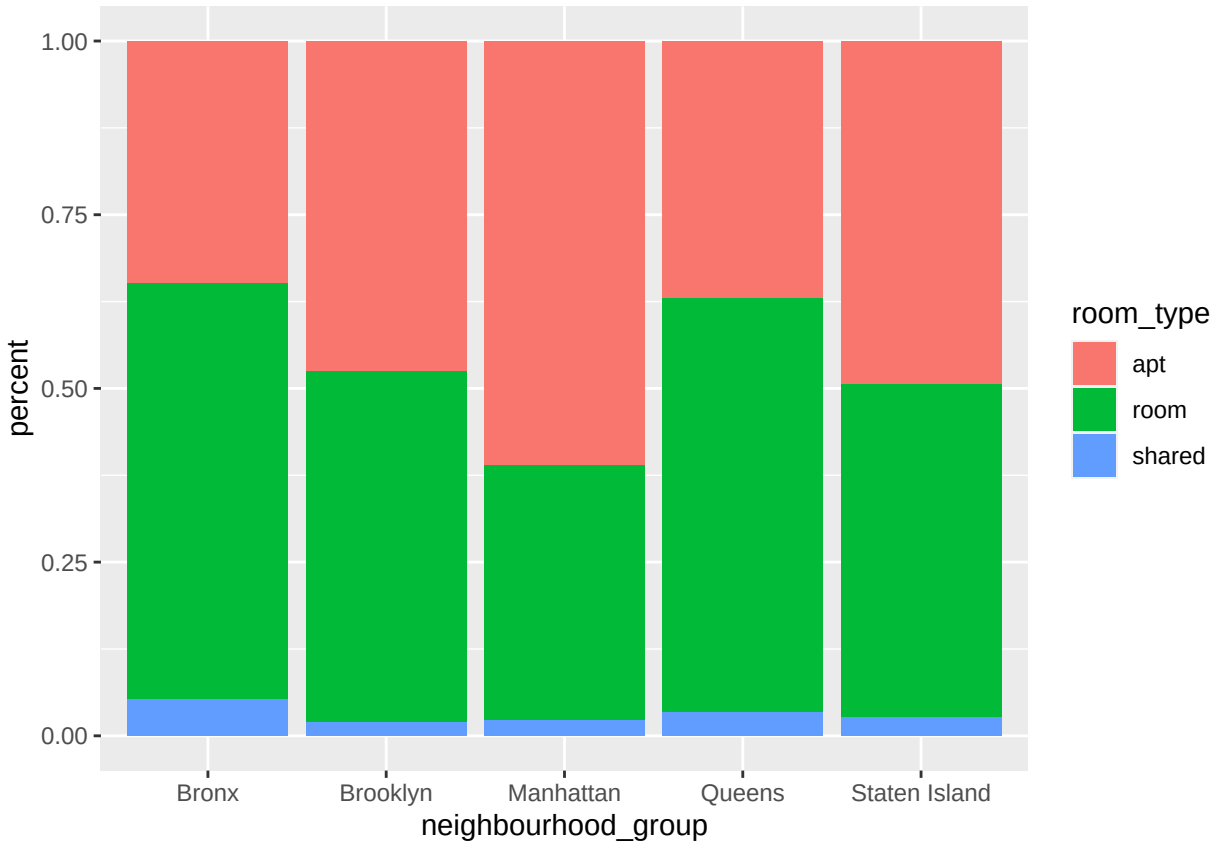
```
train_set %>%
  group_by(neighbourhood_group) %>%
  summarize(room=mean(room_type=="Private room"),
            apt=mean(room_type=="Entire home/apt"),
            shared=mean(room_type=="Shared room"))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 5 x 4
##   neighbourhood_group  room   apt shared
##   <fct>               <dbl> <dbl>  <dbl>
## 1 Bronx               0.599 0.348 0.0529
## 2 Brooklyn            0.506 0.474 0.0200
## 3 Manhattan           0.369 0.609 0.0220
## 4 Queens              0.597 0.369 0.0334
## 5 Staten Island       0.480 0.493 0.0270
```

```
train_set %>%
  group_by(neighbourhood_group) %>%
  summarize(room=mean(room_type=="Private room"),
            apt=mean(room_type=="Entire home/apt"),
            shared=mean(room_type=="Shared room")) %>%
  gather(room_type, percent, room:shared) %>%
  ggplot(aes(x=neighbourhood_group, y=percent, fill=room_type)) +
  geom_bar(position = "fill", stat="identity")
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



Manhattan listings have the highest rate of minimum nights of all neighbourhood group:

```
train_set %>%
  group_by(neighbourhood_group) %>%
  summarize(min_nights=mean(minimum_nights))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 5 x 2
##   neighbourhood_group min_nights
##   <fct>                     <dbl>
## 1 Bronx                      4.29
## 2 Brooklyn                   5.65
## 3 Manhattan                  8.00
## 4 Queens                     4.82
## 5 Staten Island              3.44
```

In addition to that, the most listings of hosts with several listings are in Manhattan:

```
train_set %>%
  group_by(neighbourhood_group) %>%
  summarize(mean(calculated_host_listings_count))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 5 x 2
##   neighbourhood_group `mean(calculated_host_listings_count)`
##   <fct>                                                <dbl>
## 1 Bronx                                                 2.19
## 2 Brooklyn                                              2.29
## 3 Manhattan                                            12.9
## 4 Queens                                                4.06
## 5 Staten Island                                         2.43
```

It is to be assumed that there were not many private hosts in Manhattan, who offered their own apartments rather commercial providers.

Looking on the availability

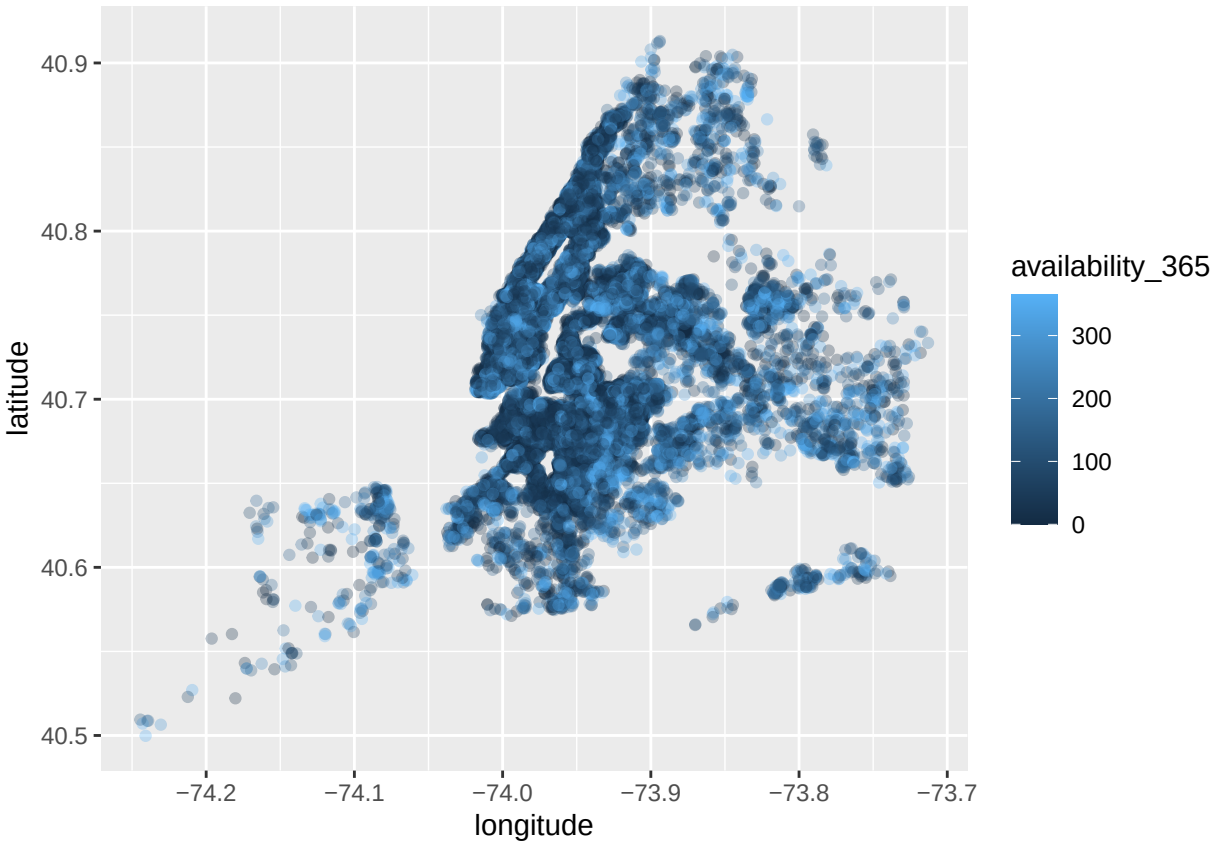
```
train_set %>%
  group_by(neighbourhood_group) %>%
  summarize(min_nights=round(mean(availability_365), digits=0))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 5 x 2
##   neighbourhood_group min_nights
##   <fct>                    <dbl>
## 1 Bronx                      164
## 2 Brooklyn                   100
## 3 Manhattan                  112
## 4 Queens                     144
## 5 Staten Island              204
```

from the geographical sight:

```
train_set %>%
  ggplot(aes(longitude, latitude, color=availability_365)) +
  geom_point(alpha=0.3)
```



We can see that Manhattan and Brooklyn had the lowest availability in 2019. An effect maybe caused by the central location of those two neighbourhoods.

Modeling approach

With the insight from the data visualization we can suspect to have some good fundamentals for the modeling approaches to predict whether a listing is in Manhattan or in another neighbourhood. For this purpose we use several predictors from the train_set:

We will use selected information as predictors like price, room type, minimum nights, number of reviews, host listings count, availability and host_id.

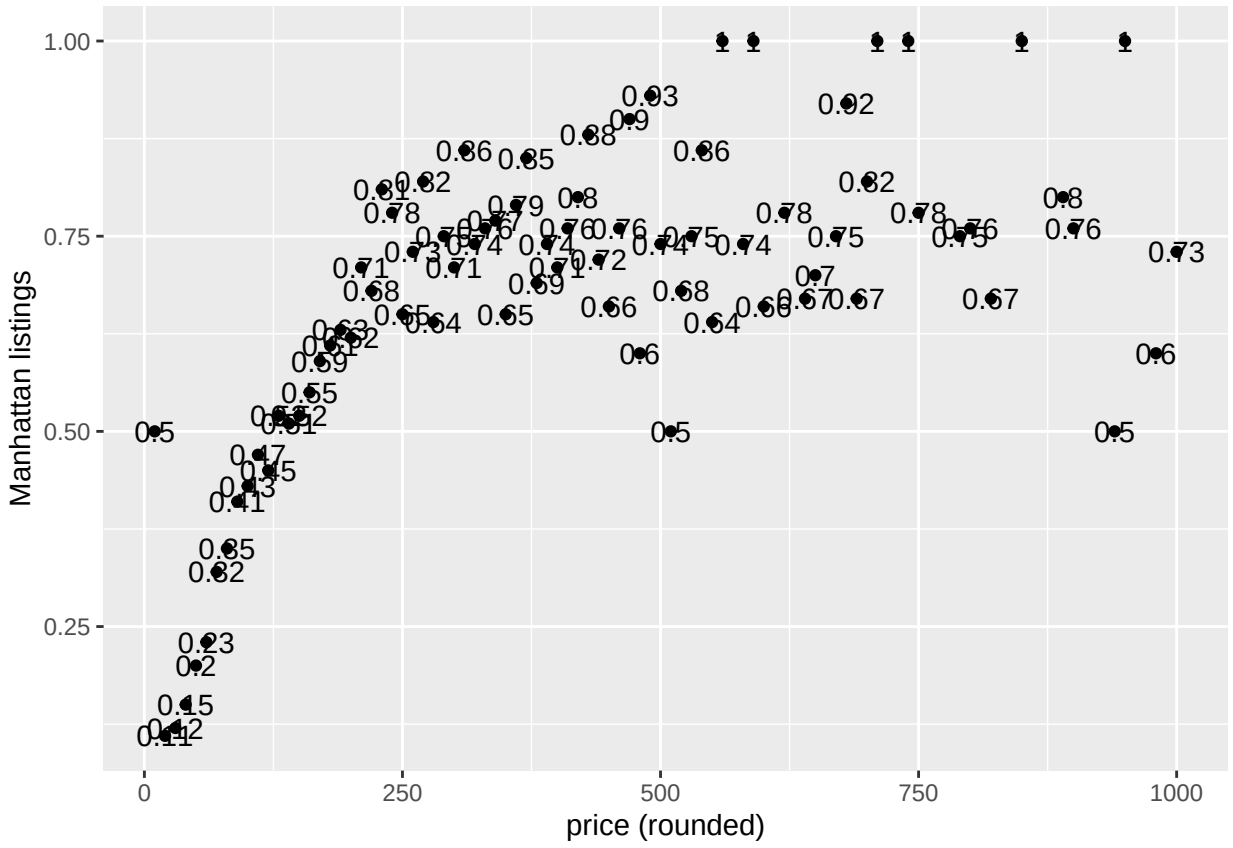
It is not beneficial to use all of the variables found in the dataset. Because we wont work with word detection the names of listings wont be used in the modeling approaches. Moreover we wont use the neighbourhood and longitude and latitude because those would already have too much predictive power and would only predict one measure of location by another measure of location. These approaches would bring an accuracy of around 99%-100%, which would not make sense for our goal of using several predictors to predict the right categorical outcome of Manhattan or not Manhattan.

Logistic regression Getting started with the first model we will use logistic regression because the outcome we force to predict is categorical data.
For the next observations we round the prices (e.g. 12 -> 10) and group them. Groupings with less than 3 listings get erased to prevent the analysis from outliers. At the proportion of Manhattan listings per price we can see some kind of a curve where the proportion of Manhattan listings increases by price and stands on a solid high level from 250 Dollars on.

```
#filter rounded price groupings with n()>=5
glm_pre <- train_set %>%
  mutate(price_round = round(price, digits=-1)) %>%
  group_by(price_round) %>%
  mutate(n=n()) %>%
  filter(n>=3) %>%
  ungroup() %>%
  select(-price_round, -n)

glm_pre %>%
  mutate(price_round = round(price, digits=-1)) %>%
  group_by(price_round) %>%
  summarize(n=n(), proportion_manhattan=mean(neighbourhood_group=="Manhattan")) %>%
  mutate(proportion_manhattan = round(proportion_manhattan, digits=2)) %>%
  ggplot(aes(price_round, proportion_manhattan)) +
  geom_point() +
  geom_text(aes(label=proportion_manhattan)) +
  xlab("price (rounded)") +
  ylab("Manhattan listings")
```

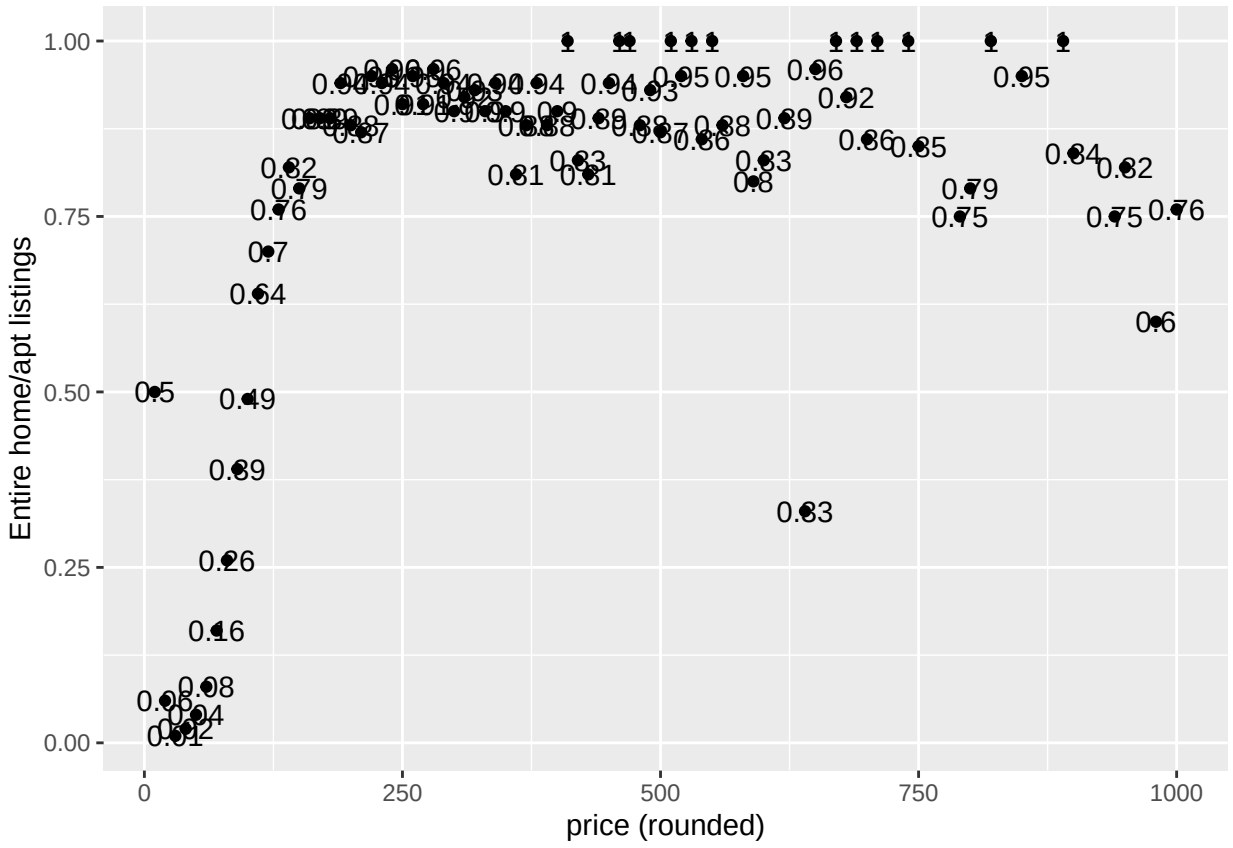
```
## `summarise()` ungrouping output (override with `.groups` argument)
```



We can see a similar effect observing the room type **Apartment** per price. This room type is significantly high for the neighbourhoods of Manhattan as we mentioned before.

```
glm_pre %>%
  mutate(price_round = round(price, digits=-1)) %>%
  group_by(price_round) %>%
  summarize(n=n(), mrt=mean(room_type=="Entire home/apt")) %>%
  mutate(mrt = round(mrt, digits=2)) %>%
  ggplot(aes(price_round, mrt)) +
  geom_point() +
  geom_text(aes(label=mrt)) +
  xlab("price (rounded)") +
  ylab("Entire home/apt listings")
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



Starting with the logistic regression I used the glm function to train whether a listing is in Manhattan or not with several predictors, then predicting the data of the testing set and finally projecting the confusion matrix and its accuracy:

```
glm_fit <- glm_pre %>%
  mutate(y = as.numeric(manhattan == "manhattan")) %>%
  glm(y ~ host_id +
        room_type +
        price +
        minimum_nights +
        number_of_reviews +
        reviews_per_month +
        calculated_host_listings_count +
        availability_365
        , data=., family = "binomial")
p_hat_logit <- predict(glm_fit, newdata = test_set, type = "response")
y_hat_logit <- ifelse(p_hat_logit > 0.5, "manhattan", "not_manhattan") %>% factor
confusionMatrix(y_hat_logit, test_set$manhattan)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      manhattan not_manhattan
##   manhattan          1675           774
```

```
##   not_manhattan      2619          4655
##
##                Accuracy : 0.651
##                  95% CI : (0.6415, 0.6605)
##     No Information Rate : 0.5584
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2592
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.3901
##             Specificity : 0.8574
##          Pos Pred Value : 0.6840
##          Neg Pred Value : 0.6400
##              Prevalence : 0.4416
##          Detection Rate : 0.1723
##    Detection Prevalence : 0.2519
##       Balanced Accuracy : 0.6238
##
##        'Positive' Class : manhattan
##
```

```
glm_acc <- confusionMatrix(y_hat_logit, test_set$manhattan)$overall[["Accuracy"]]
```

The accuracy for this model is at 0.6547. The specificity is much higher than the sensitivity. This shows that the most listings outside of Manhattan were predicted correctly while more than half of the listings from Manhattan were predicted to be outside of Manhattan.

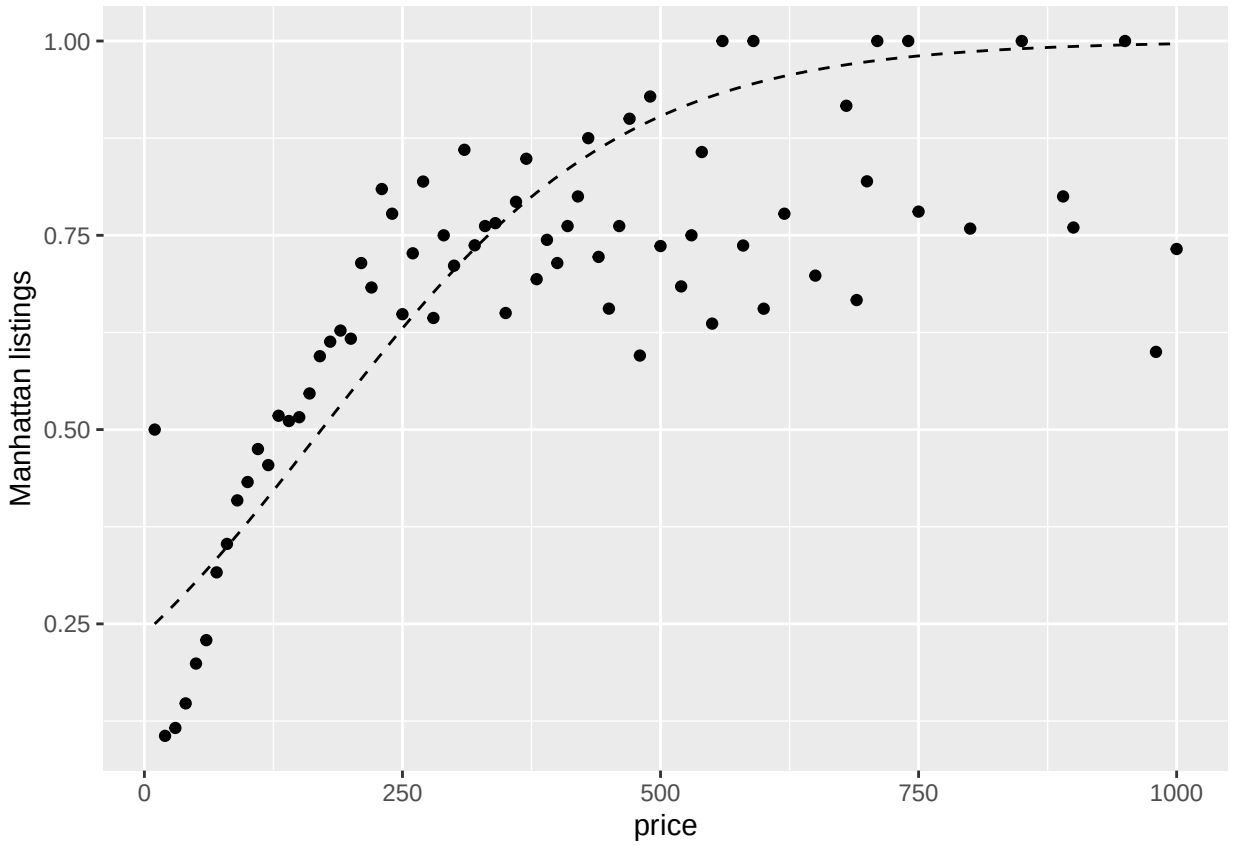
Comparing that model with the logistic regression but only with price as the predictor we get an accuracy close to the model before:

If we now draw a logistic curve on the proportion of Manhattan for prices we get a way more accurate fit than with a simple line:

```
tmp <- glm_pre %>%
  mutate(x = round(price, digits=-1)) %>%
  group_by(x) %>%
  filter(n() >= 5) %>%
  summarize(prop = mean(manhattan == "manhattan"))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
logistic_curve <- data.frame(x = seq(min(tmp$x), max(tmp$x))) %>%
  mutate(p_hat = plogis(glm_fit$coef[1] + glm_fit$coef[2]*x))
tmp %>%
  ggplot(aes(x, prop)) +
  geom_point() +
  geom_line(data = logistic_curve, mapping = aes(x, p_hat), lty = 2) +
  xlab("price") +
  ylab("Manhattan listings")
```



The proportion of Manhattan listings by price seems to show a limited growth.

K-nearest neighbours I already told before that some attributes wont be taken into account because they would just predict a location measure by another location measure. To show this we will, just for investigation purposes, run the knn-model with longitude and latitude as predictors for the neighbourhood prediction:

```
m_knn <- train_set %>%
  train(manhattan~longitude+latitude,
        method="knn",
        data=.,
        tuneGrid = data.frame(k = seq(5))
        #trControl=control
  )
m_p_hat_knn <- predict(m_knn, test_set, type = "raw")
confusionMatrix(m_p_hat_knn, test_set$manhattan)$overall["Accuracy"]
```

```
##  Accuracy
## 0.9998972
```

This model shows that we are close to predict every listing in the test set correctly. So lets continue.

Decision tree To improve the accuracy of predicting a listing that we already got from the logistic regression and to find out decisive predictors the next model I have chosen is the decision tree. An advantage of the decision trees is their good visual interpretability:

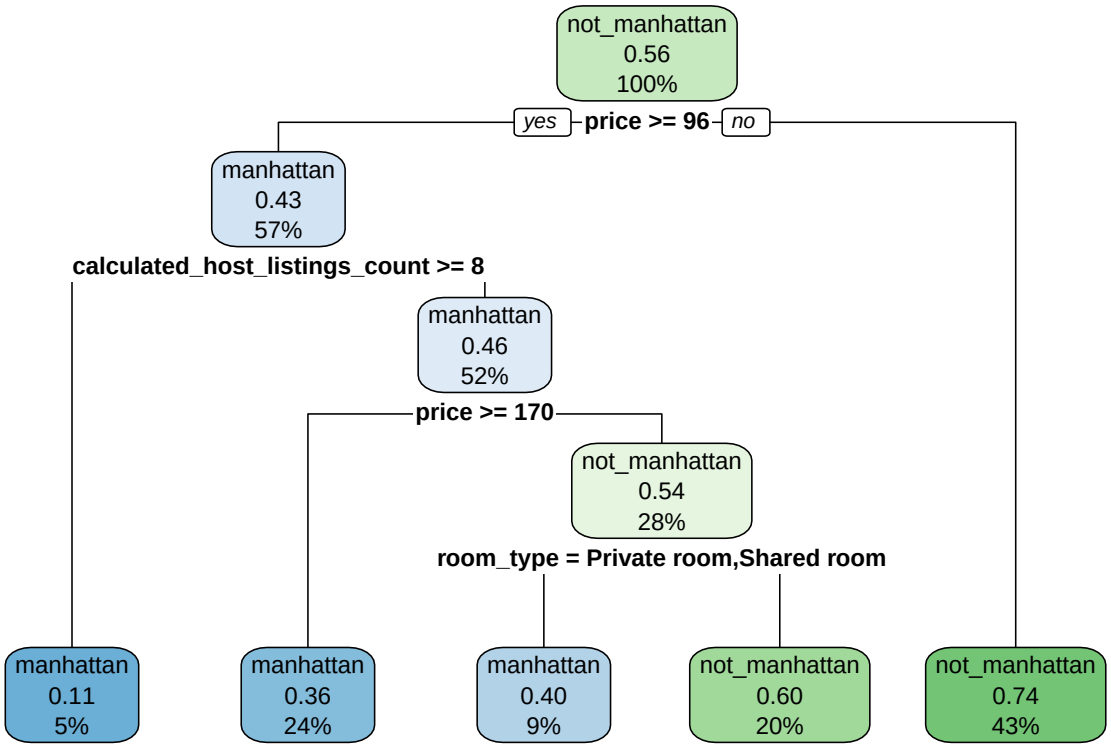
```
dec_fit <- train_set %>%
  select(host_id,
         room_type,
         price,
         minimum_nights,
         number_of_reviews,
         calculated_host_listings_count,
         availability_365,
         manhattan
  ) %>%
  rpart(manhattan ~ ., data=., model=TRUE)
fit_pred <- predict(dec_fit, test_set, type="class")
confusionMatrix(table(fit_pred, test_set$manhattan))
```

```
## Confusion Matrix and Statistics
##
##
## fit_pred        manhattan not_manhattan
##   manhattan          2347          1263
##   not_manhattan      1947          4166
##
##                Accuracy : 0.6699
##                  95% CI : (0.6604, 0.6792)
##     No Information Rate : 0.5584
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3193
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.5466
##             Specificity : 0.7674
##          Pos Pred Value : 0.6501
##          Neg Pred Value : 0.6815
##              Prevalence : 0.4416
##          Detection Rate : 0.2414
##    Detection Prevalence : 0.3713
##       Balanced Accuracy : 0.6570
##
##        'Positive' Class : manhattan
##
```

```
dt_acc <- confusionMatrix(table(fit_pred, test_set$manhattan))$overall[["Accuracy"]]
```

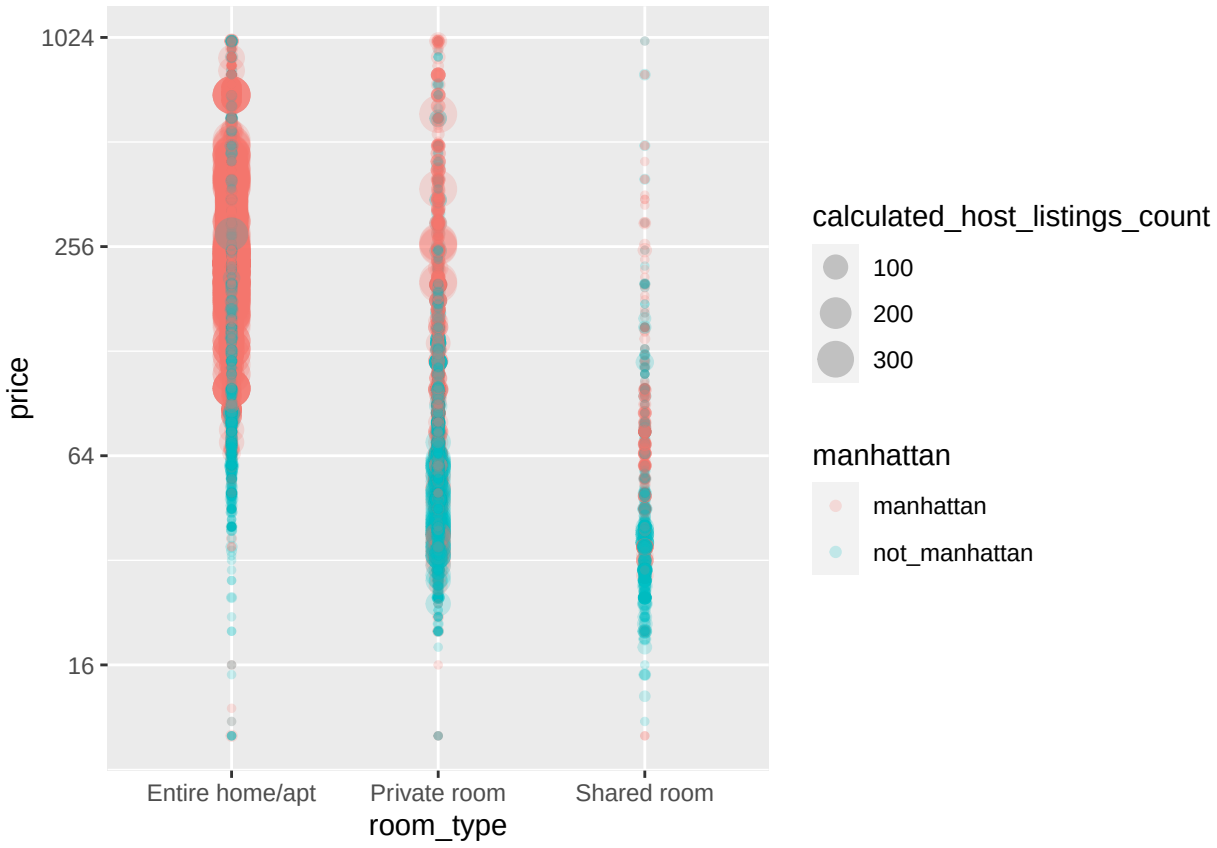
You can see a slight improvement over the previous model. Now the accuracy is at 0.6775.
This is how the Decision tree looks like:

```
rpart.plot(dec_fit)
```



The tree uses price, host listings count and room type as the predictors to get the best result to decide whether the listing is in Manhattan or in another neighbourhood.
The decisions from the tree can be reviewed in a plot that contains those predictors:

```
glm_pre %>%
  ggplot(aes(room_type, price, col=manhattan, size=calculated_host_listings_count)) +
  geom_point(alpha=0.2)+
  scale_y_continuous(trans="log2")
```



Here we can observe the branches we already had in the decision tree. Most listings <96 Dollars are not in Manhattan. Most listings >=96 Dollars and with a high host listings count are Manhattan based. Those with a price >=164 Dollars are mostly Manhattan based.

Random forests Trying to improve the prediction performance of the previous modeling approach we use the random forest as an average of multiple decision trees.

```
forest_fit <- train_set %>%
  select(host_id,
         room_type,
         price,
         minimum_nights,
         number_of_reviews,
         calculated_host_listings_count,
         availability_365,
         manhattan
  ) %>%
  randomForest(manhattan ~ ., data=., ntree=500)
forest_p_hat <- predict(forest_fit, test_set, type="response")
confusionMatrix(forest_p_hat, test_set$manhattan)
```

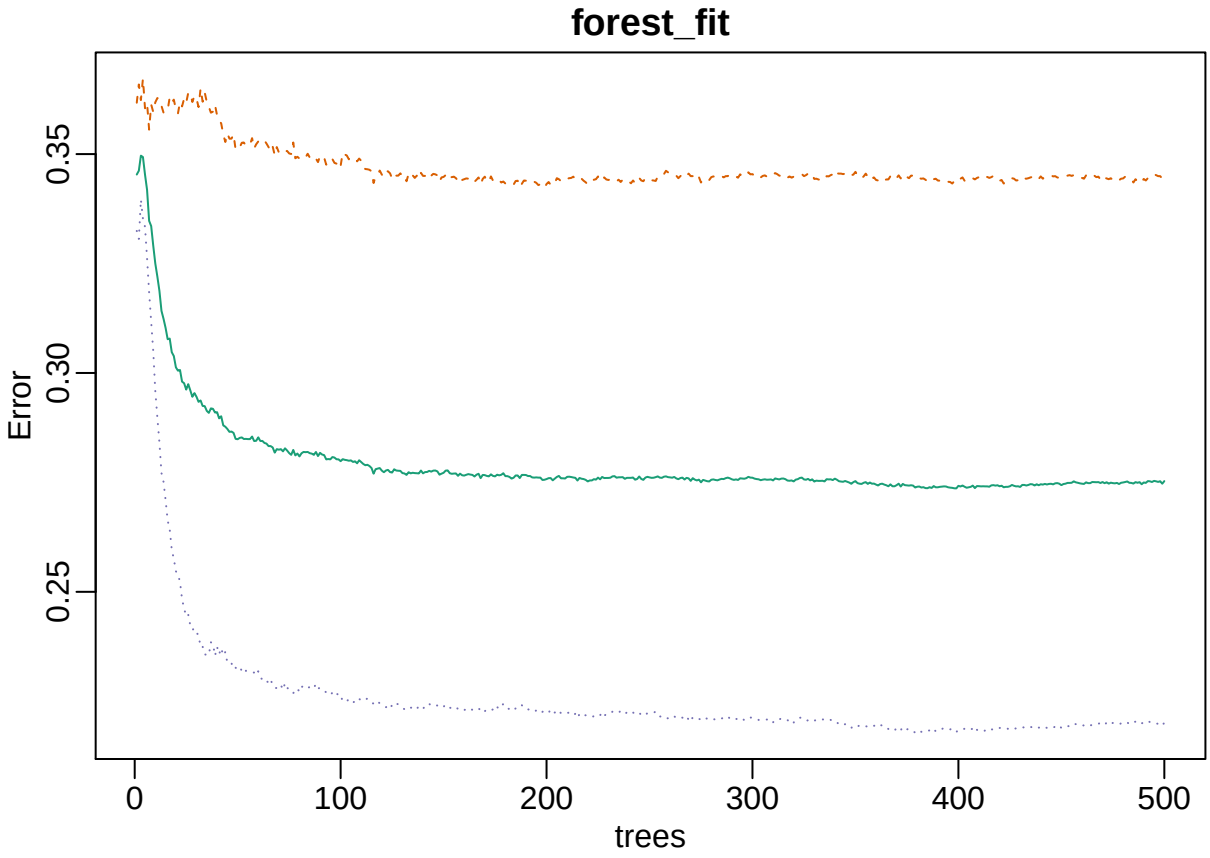
```
## Confusion Matrix and Statistics
##
```

```
##                Reference
## Prediction      manhattan not_manhattan
##   manhattan          2730          1200
##   not_manhattan      1564          4229
##
##                Accuracy : 0.7157
##                  95% CI : (0.7066, 0.7247)
##     No Information Rate : 0.5584
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4184
##
##  Mcnemar's Test P-Value : 5.035e-12
##
##             Sensitivity : 0.6358
##             Specificity : 0.7790
##          Pos Pred Value : 0.6947
##          Neg Pred Value : 0.7300
##              Prevalence : 0.4416
##          Detection Rate : 0.2808
##    Detection Prevalence : 0.4042
##       Balanced Accuracy : 0.7074
##
##        'Positive' Class : manhattan
##
```

```
rf_acc <- confusionMatrix(forest_p_hat, test_set$manhattan)$overall["Accuracy"]
```

With a random forest of 500 trees the accuracy rises to 0.7159.

```
rafalib::mypar()
plot(forest_fit)
```



As we can see from the previous plot the Out-of-bag error is at 27.55%, the class error of manhattan is at 34.80% and the class error of not_manhattan is at 21.82% if we use 500 trees. It seems not necessary to increase the ntree-parameter because after 100 trees there is only marginal variation on each error.

3. Results

As we can see from the confusion matrix of each modeling approach we get differently accurate results:

Method	Accuracy
glm accuracy	0.6510336
glm accuracy (only price)	0.6398231
decision tree accuracy	0.6698550
random forest accuracy	0.7157256

The random forest approach brings the highest accuracy. ## 4. Conclusion
From the different analysis and modelings we now have an insight on how much the different informations about AirBnB listings have an influence on how to predict in which neighbourhood group a listing is. In different methods we've already seen that the price is a recommended predictor for the neighbourhood prediction.
The highest overall accuracy is at 0.7167. This means that around every fourth listing will be predicted incorrectly to be in a neighbourhood group in which it is actually not.

As we've seen there are some significant differences between the neighbourhood groups in New York, especially between Manhattan and other neighbourhoods. As a next step we could for example run analysis and models to predict the prices of listings and analyse whether signs of gentrification are recognizable in neighbourhoods.