Amsterdam Airbnb Listings

HarvardX Capstone Project

08/January/2021

Dino Vukelić

Executive Summary

Analysis will highlight specific elements of the "Amsterdam Airbnb Listing" Dataset from 12 December, 2020 which features 18,522 listings with 17 variables, and available for download on [insideairbnb] (http://insideairbnb.com/get-the-data.html).

In this report for the Capstone course of the HarvardX Professional Certificate in Data Science (PH125.9x), we will begin by examining the unrefined data and perform any necessary tidying of the dataset. We will then determine which features provide most insight into price prediction and remove those that only serve to add noise. Data visualization and exploratory data analysis will then be performed on the tidied dataset to inform our modelling.

Several machine learning algorithms will be applied to improve our RMSE (the root mean square error) from our baseline to our final model.

Methods and Analysis

First we begin with data preparation and loading the data from insideairbnb. After the data has been loaded successfully, we will proceed to clean the data by checking it for NAs and other errors such as misspellings or incorrect categorizations or associations.

Once the NAs have been removed and the data tidied, we proceed to our exploratory data analysis where we examine correlations between the price and the other features. Visualizations will illuminate these connections and confirm our hypotheses.

Modelling will involve the partitioning of our dataset "Airbnb" into training, validation, and test sets. In this report, the "validation" set will be used in conjunction with the training set. This is in contrast to the terminology used in the MovieLens Project, but more in line with contemporary machine learning conventions. We will start with a baseline model based on the median price of the dataset and move on to more advanced algorithms including linear models, elastic net regression, regression tree,kNN, and BAG all tuning with the training set "airbnb_train". Models will be evaluated based on which produces the lowest RMSE.

Mathematically, the RMSE is the standard deviation of the prediction errors (residuals) and is used to measure the difference between observed and predicted values. The advantage of using the RMSE is that its unit is the same as the unit being measured (in this case the price in EUR). The model that produces the lowest RMSE on the validation set will then be run on the test set.

Data Preparation and Required Packages

We will begin by loading the following libraries: tidyverse, readr, data.table, icesTAF, caret, lubridate, glmnet, scales, stringr, dplyr, ggmap, ggcorrplot, treemapify, rpart, nnet, formatR, rmarkdown, and knitr with the "pacman" package. (If a package below is missing, p_load will automatically download it from CRAN).

```
if(!require(pacman)) install.packages("pacman", repos = "http://cran.us.r-project.org")
library(pacman)
pacman::p_load(tidyverse, readr, data.table, icesTAF, caret, lubridate,
               ggthemes, ggplot2, glmnet, scales, stringr, dplyr, ggmap, ggcorrplot,
               treemapify, rpart, nnet, formatR, rmarkdown, knitr)
```

Data Preparation

Download the Dataset:

```
if(!dir.exists("Airbnb")) mkdir("Airbnb")
if(!file.exists("./Airbnb/Amsterdam.csv"))
download.file("https://raw.githubusercontent.com/DinoVukelic/Airbnb/main/Amsterdam.csv
", "./Airbnb/Amsterdam.csv")
```

Read the Data:

```
suppressWarnings(Airbnb <-read_csv("./Airbnb/Amsterdam.csv"))
```

Set the number of significant digits to 4

```
options(digits = 4)
```

Preliminary Data Exploration and Cleaning

Check dimensions of dataset:

```
dim(Airbnb)
```

There are 18522 observations and 17 features

Check the classes of the features:

```
str(Airbnb)
```

We note that there 17 features of numeric, character and logical classes. We also notice that there are already visible NA values in the "neighbourhood_group" feature.

Dating Cleaning

Coerce Airbnb tibble into a data frame:

```
Airbnb <-as.data.frame(Airbnb)
class(Airbnb)
```

Confirm that the data is tidy and that there are no NAs:

```
sum(is.na(Airbnb))
```

There are 23360 NAs in the dataset.

Check which features have NAs:

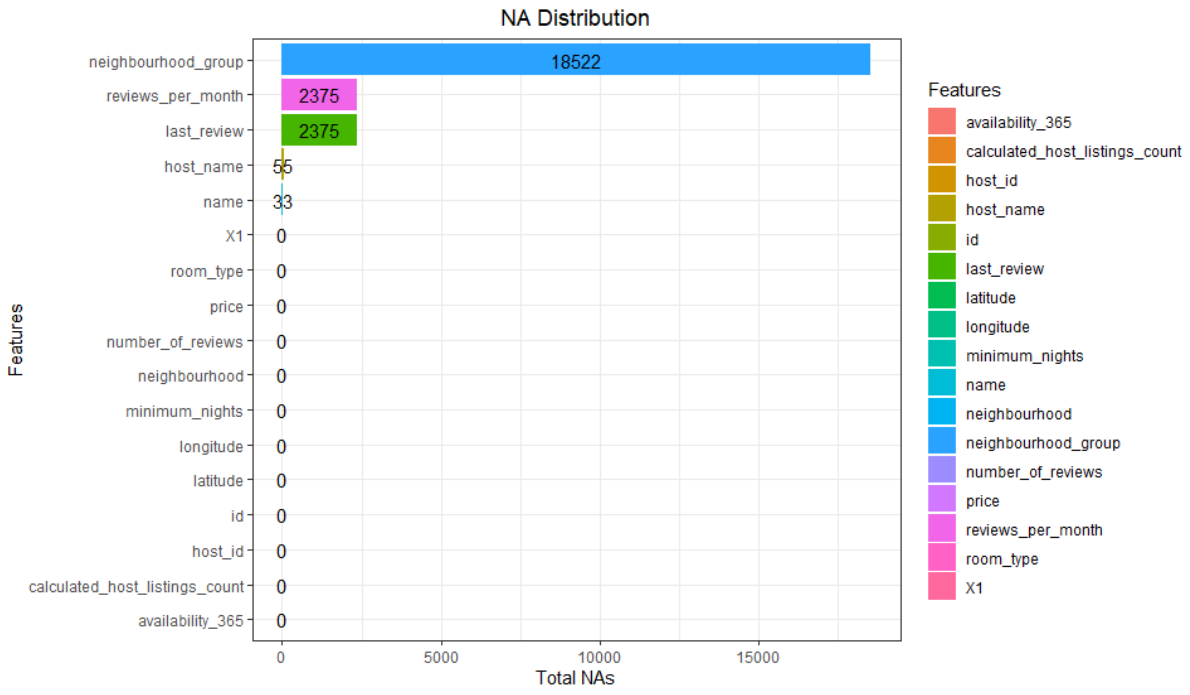
```
colSums(is.na(Airbnb))
```

We find that neighbourhood_group has 18522 NAs, last_review has 2375 NAs, reviews_per_month has 2375 NAs, host_name has 55 NAs, while name has 33 NAs.

Create a dataframe na_bar to plot the NAs:

```
na_vis <- data.frame(t(colSums(is.na(Airbnb))))
na_bar <- data.frame(Features = names(na_vis),totals=colSums(na_vis))
```

Let's observe the NA distribution visually:



The visualization confirms there are only five features with missing values and that the vast majority of observations are present. After confirming that there are only 5 features with NAs, (neighbourhood_group, last_review, reviews_per_month, host_name and name) we proceed to clean the dataset. As overall_satisfaction has 1473 NAs, assigning a value of 0 will significantly skew the ratings negatively. Therefore we will fill the values with the mean to provide for more accurate predictive values.

Convert NAs to the mean value:

```
Airbnb$reviews_per_month[is.na(Airbnb$reviews_per_month)] <-
mean(Airbnb$reviews_per_month, na.rm = TRUE)
```

The mean is roughly 0.63.

```
mean(Airbnb$reviews_per_month)
```

Confirm the absence of NAs in this feature:

```
head(Airbnb$reviews_per_month)
```

For the host_name, name, last_review and neighbourhood_group feature, there are only 20,985 NAs and so we set them to zero because they will not be needed for price prediction.

```
Airbnb <-Airbnb %>% replace_na(list(host_name = 0 , name = 0, last_review = 0,
neighbourhood_group = 0))
```

Now confirm the absence of any NAs in the dataset:

```
sum(is.na(Airbnb))
```

Feature Exploration and Selection

```
names(Airbnb)
```

There are 17 features and those less related to price prediction will be dropped to refine our EDA and Modelling Focus:

Feature: "X1"

```
head(Airbnb$X1)
```

"X1" is simply a numerical list for the dataset.

Features: "id" & "host_id"

```
head(Airbnb$id)
head(Airbnb$host_id)
```

"id" & "host_id" are arbitrary numbers assigned to identify rooms and hosts.

Feature: "neighbourhood"

Check for unique values of the feature "neighbourhood"

```
Airbnb %>% select(neighbourhood) %>% distinct()
```

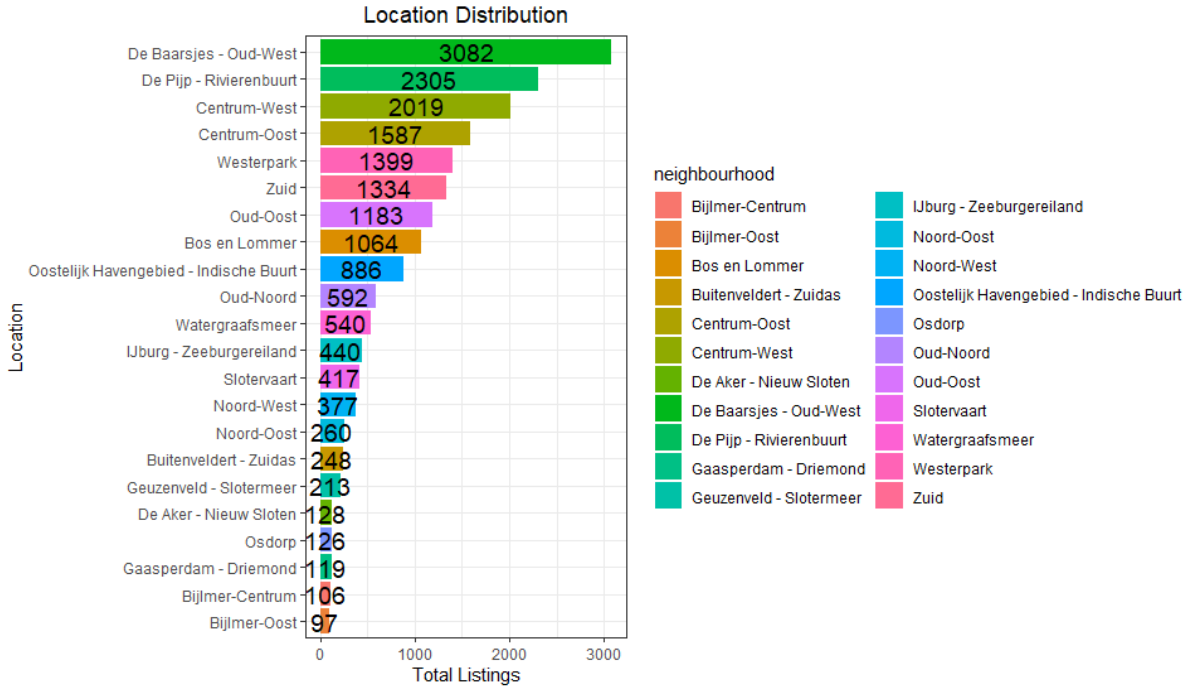
Note that there are 22 values that represent Amsterdam neighbourhoods and city parts.

numbers of listings:

```
city_list <-Airbnb %>% group_by(neighbourhood) %>% summarize(listing_sum = n())
%>%
  arrange(-listing_sum)
city_list
```

It is clear the vast majority of listings are in De Baarsjes - Oud-West (3082).

Let's explore a data visualization to confirm this:



Features: "X1, id, host_id, host_name,last_review, neighbourhood_group,reviews_per_month, name, calculated_host_listings_count"

The "X1, id, host_id, host_name,last_review, neighbourhood_group,reviews_per_month, name, calculated_host_listings_count" features will not be needed for price prediction therefore those features will be removed.

```
head(Airbnb$X1)
head(Airbnb$id)
head(Airbnb$host_id)
head(Airbnb$host_name)
```

```
head(Airbnb$last_review)
head(Airbnb$neighbourhood_group)
head(Airbnb$reviews_per_month)
head(Airbnb$name)
head(Airbnb$calculated_host_listings_count)
```

Feature: "currency"

The "currency" feature will be dropped as all rates are in Euros.

```
Airbnb %>% select(price) %>% distinct()
```

Create the cleaned dataset

Remove the above mentioned features:

```
Airbnb <-Airbnb %>% select(-c(X1, id, host_id, host_name,last_review,
neighbourhood_group,reviews_per_month,
                              name, calculated_host_listings_count))
```

Confirm the features have been tidied and reordered with only 8 features:

```
names(Airbnb)
```

Check the first few values of the cleaned dataset:

```
head(Airbnb)
```

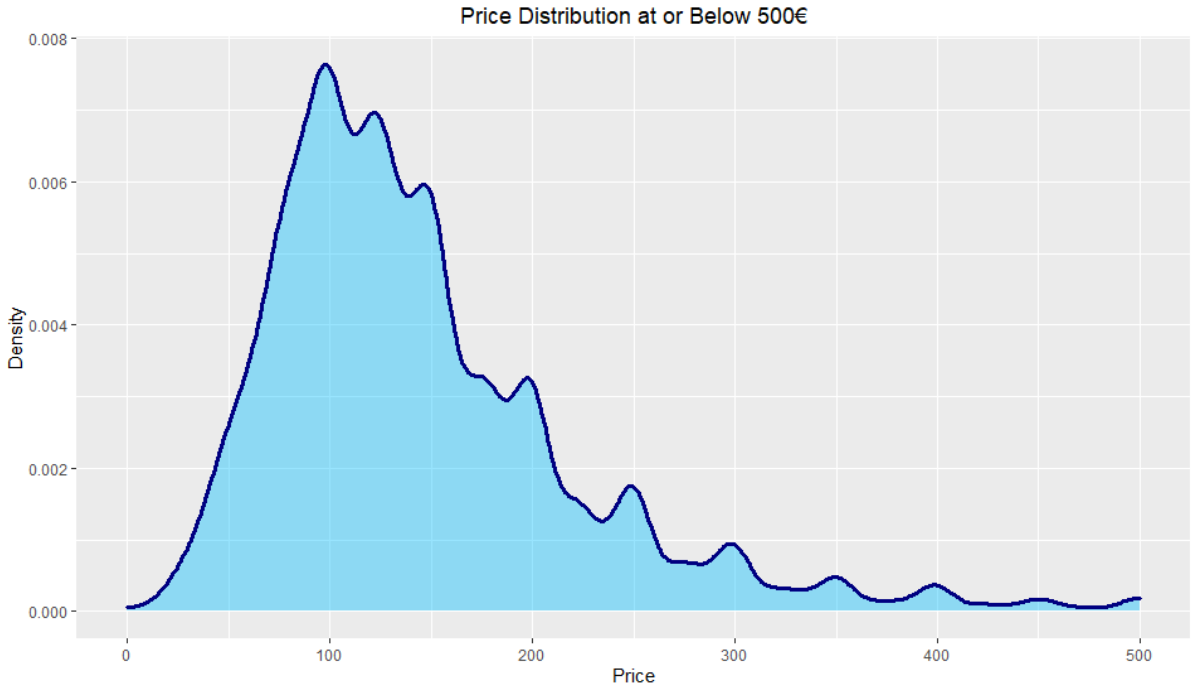
Explanatory Data Analysis

Now we will begin analyzing the features of our dataset to inform our price prediction modelling approach.

Density Plot of Price Distribution at or below 500€:

First we explore a plot of the price distribution at or below 500€ to determine the most common price ranges.

Density Plot of Price Distribution at or below 500€:



The plot reveals a significant portion of the prices are below 200€/night.

Geographical Scatterplot of Prices in Amsterdam:

Let's use the ggmap package to load a map of Amsterdam and visualize which areas are more expensive than others.

Create the map using stamenmap:

```
amsterdam_map <- get_stamenmap(bbox = c(left = 4.72, bottom = 52.25,
                                        right = 5.06, top = 52.46),
                               zoom = 11, maptype = "terrain")
```

Note that the quantiles and price range will influence the pricing scale color:

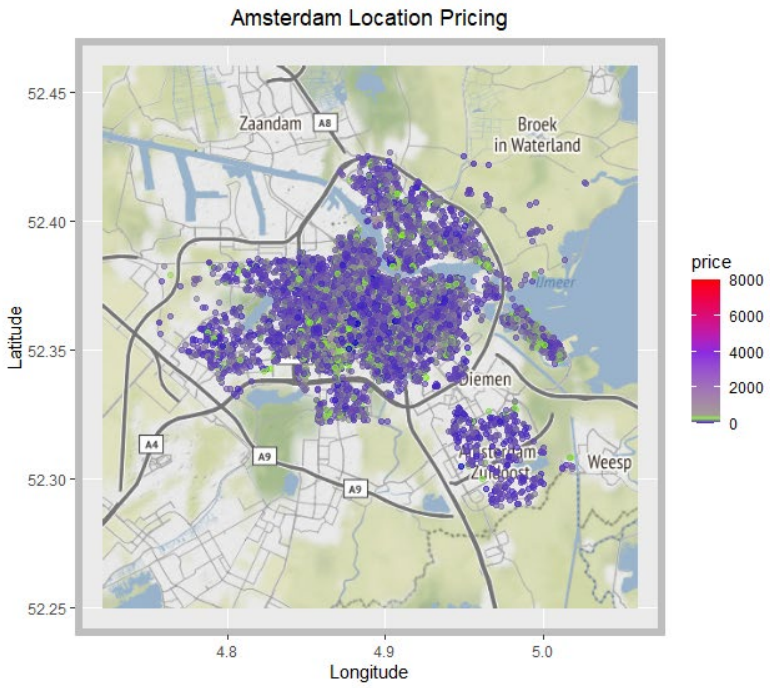
```
summary(Airbnb$price)
```

According to the above summary, we know that Q3 of the IQR was 180€, so let's determine what percentage of prices are less than or equal to 500€.

```
quantile(Airbnb$price)
sum(Airbnb$price <=500)/length(Airbnb$price)
```

~ 98.7% of listings are less than or equal to 500€, therefore we filter our price to remove outliers.

Visualize a "Heatmap" of Amsterdam with all listing prices included:

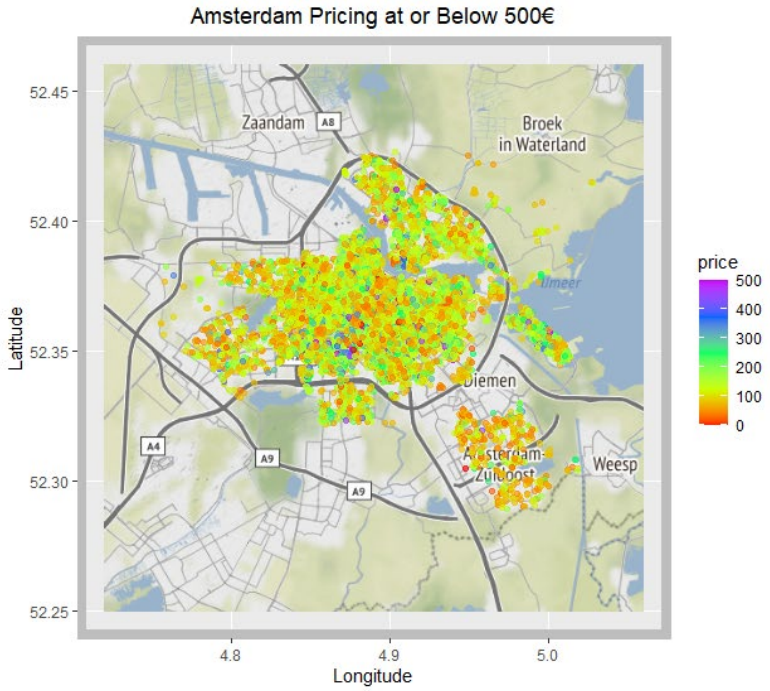


It does not appear that one area is significantly more expensive than another and that most prices are less than 500€, though outliers may be skewing our data, so let's attempt to refine our heatmap by filtering the data.

Filter the dataframe airbnb_map:

```
Airbnb_map <-Airbnb %>% filter(price <=500)
```

Visualize a "Heatmap" of Amsterdam with all listing prices included:



The visualization suggests the vast majority of listings in Amsterdam are between 50€ - 200€ / night and are relatively concentrated in Oostelijk Havengebied - Indische Buurt, Centrum-Oost, Centrum-West, De Pijp - Rivierenbuurt, De Baarsjes - Oud-West, Bos en Lommer, Westerpark, Oud-Oost, Noord-West, Watergraafsmeer, IJburg - Zeeburgereiland and Noord-Oost.

Confirm the percentage of listings between 50€ - 200€ per night:

```
sum(Airbnb$price >= 50 & Airbnb$price <= 200)/length(Airbnb$price)
```

Nearly 80% of listings range from 50€ - 200€/night.

Treemap:

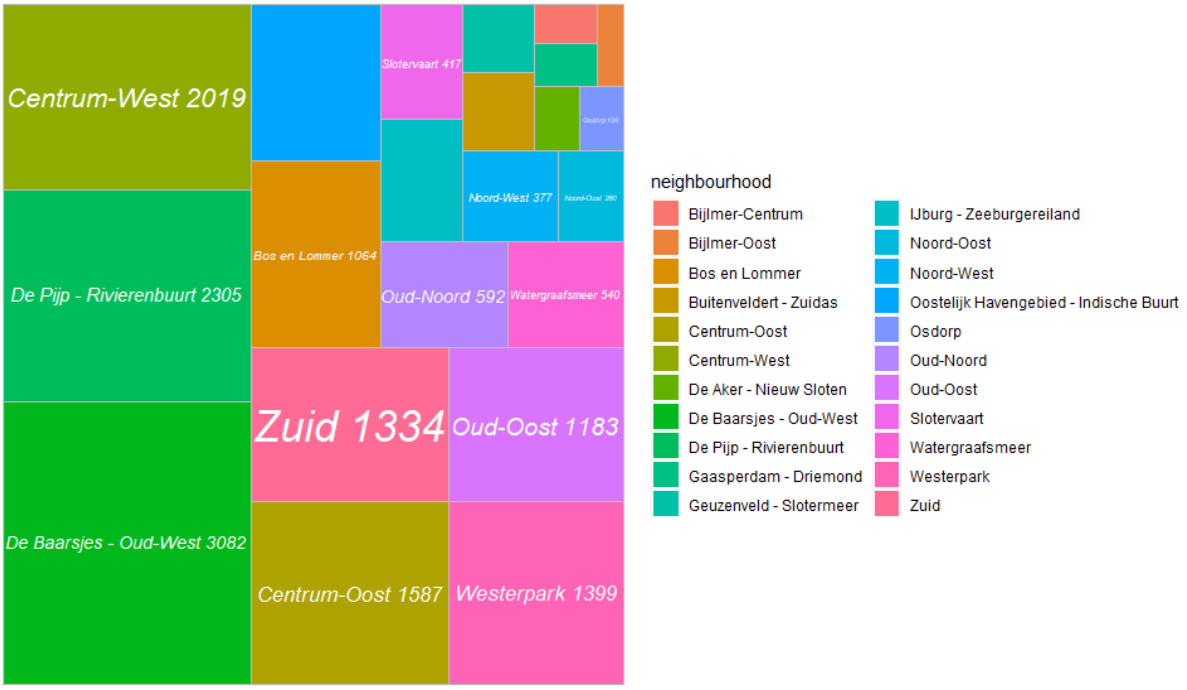
Arrange the neighbourhood by minimum_nights in a dataframe for the Treemap:

```
neighbourhood_distribution <-Airbnb %>% group_by(neighbourhood) %>%
summarize(minimum_nights = n()) %>%
  arrange(-minimum_nights)
```

Add column "tmlab" for Treemap labels:

```
neighbourhood_distribution <-neighbourhood_distribution %>%
  unite("tmlab", neighbourhood:minimum_nights, sep = " ", remove = FALSE)
```

Plot a Treemap to visualize the distribution of listings by neighbourhood:

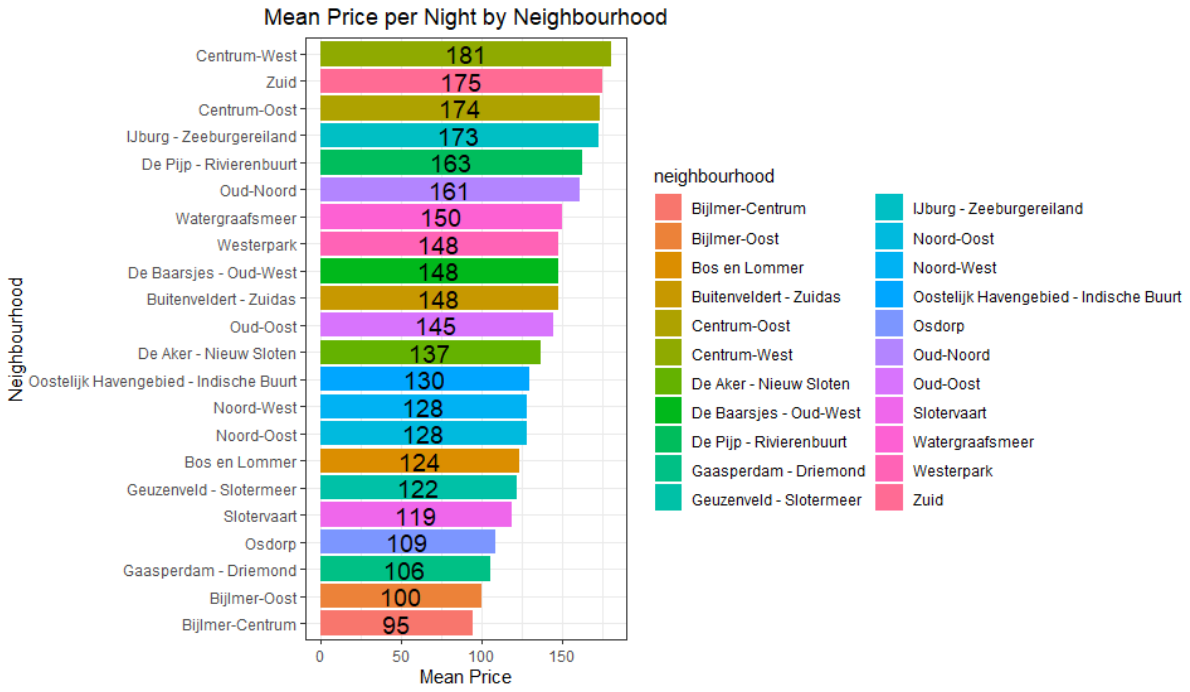


The Treemap confirms the overwhelming number of listings in Oostelijk Havengebied - Indische Buurt, Centrum-Oost, Centrum-West, De Pijp - Rivierenbuurt, De Baarsjes - Oud-West, Bos en Lommer, Westerpark, Oud-Oost, Noord-West, Watergraafsmeer, IJburg - Zeeburgereiland and Noord-Oost compared to the other neighbourhoods.

Coerce the "mean_price" to integer:

```
neighbourhood_price$mean_price <-as.integer(neighbourhood_price$mean_price)
```

Plot the Visualization:



According to this visualization, it would seem that location (latitude & longitude) is highly correlated with the price. Let's replot the visualization with the Mean Price & Sum of Listings by neighbourhood.

Visualize Mean Price & Sum of Listings by neighbourhood:

Create dataframe "neighbourhood_comp" with a percentage column:

```
neighbourhood_comp <-neighbourhood_price %>%
  mutate(percentage = sprintf("%0.3f",(minimum_nights/sum(minimum_nights)*100)))
```

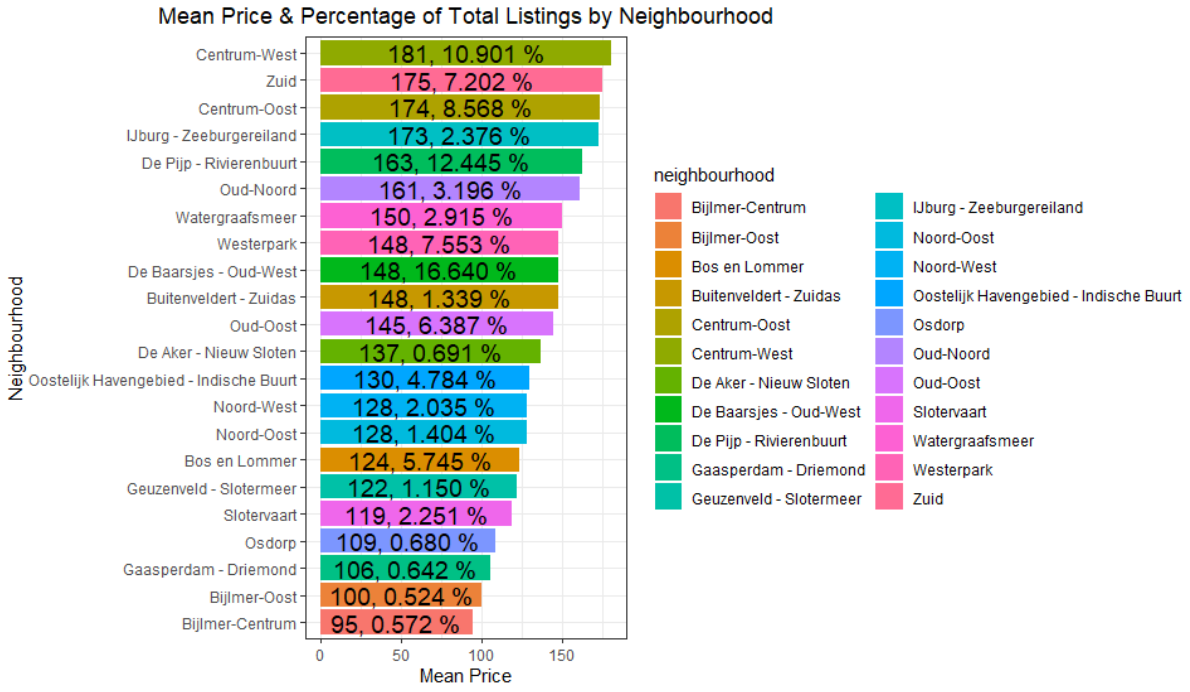
Add the % symbol to the percentage feature:

```
neighbourhood_comp$percentage <- paste(neighbourhood_comp$percentage, "%")
```

Combine the mean price & percentage values into one column:

```
neighbourhood_comp <-neighbourhood_comp %>%
  unite("citylab", mean_price, percentage, sep = ", ", remove = FALSE)
```

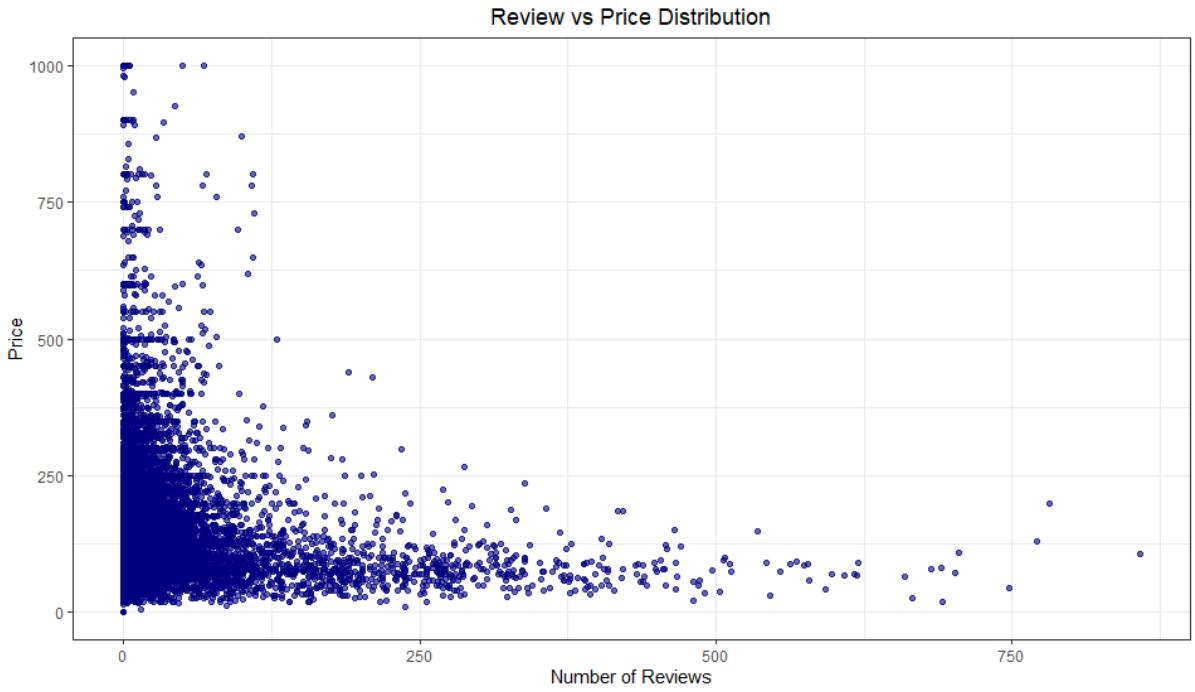
Plot the visualization with Mean Price & Percentage of Total Listings:



We learn that 10.9% of listings are in Centrum-West with the highest mean price of 181€/Night. While highest listing of 16.64% is in De Baarsjes-Oud-West with the mean price of 148€/Night.The lower percentage of total listings in these neighbourhoods at the lower price range, does not indicate that they will have higher amount of listings.Therefore the conclusion here would be that lower the mean price is the lower the profits are.

Feature: "reviews_sum"

Now let's compare the total number of reviews and different price ranges:

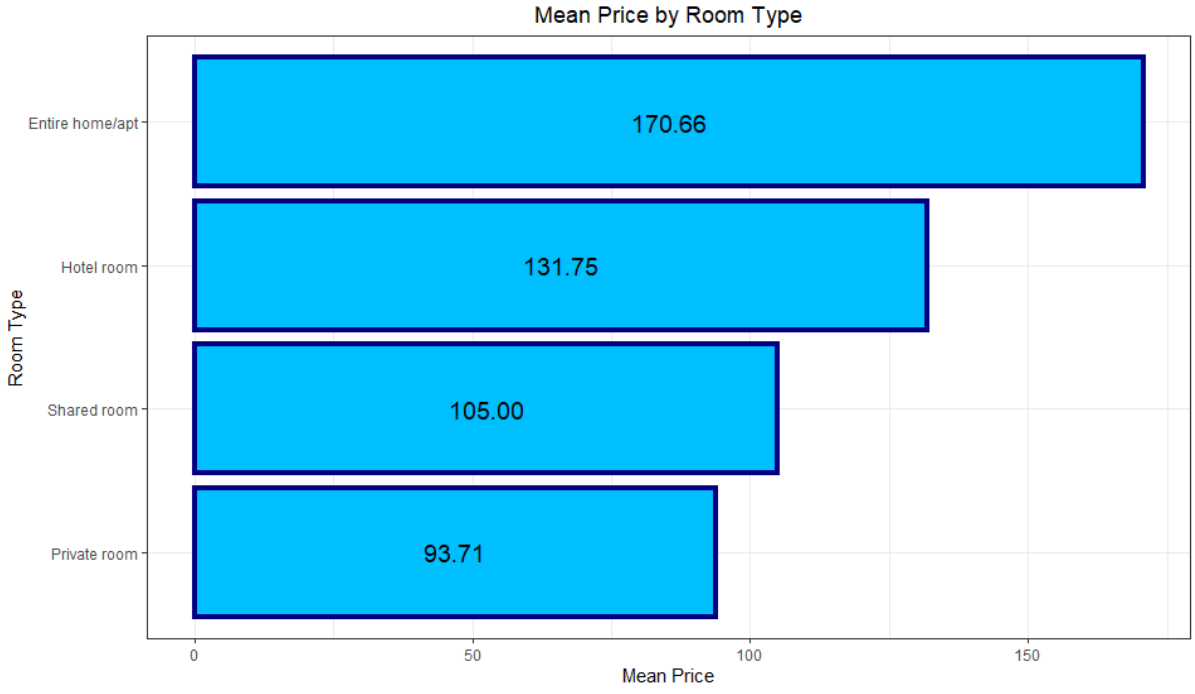


The visualization shows us that there are generally fewer reviews for listings with higher prices (above ~300€). We can infer that this is because there are fewer stays at these higher-priced listings.

Feature: "room_type"

There are 3 different room types: Entire home / Apartment, Private Room, & Shared Room.

Let's explore the relationship between room type and average price:



We notice that entire home/apartment has significantly higher mean price of 170.66€ than hotel, shared and private room. Private room has the lowest mean price of 93.71€. Interestingly, people in Amsterdam are more inclined to rent the entire home/apartment no matter the price. And we can see that the market started to exploit that room type by having a noticeably higher mean price for entire home/apartment.

EDA Conclusion

The Density Plot of Price Distribution at or below 500€ starts us off by highlighting important relationships of the most common price ranges,which are later confirmed by visualizations and quantitative analysis. Additionally, we learn location (using latitude and longitude), and reviews sum all have certain potential to increase the accuracy of our predictive models if included in our formula. "Room_type", an important feature, could be vectorized into a quantitative scale, though for the purposes of the following regression models, will be revisited in our conclusion.

Modelling

Partition Airbnb Combined & Test Sets for the Final Model:

airbnb_combined will be used to fit the Final Model as it is equal to the sum of the training and validation sets and equal to 90% of the total data. airbnb_test, comprised of the remaining 10% of the data, will be used as the test set for the final model.

```
set.seed(123, sample.kind = "Rounding")
test_index <- createDataPartition(y = Airbnb$price, times = 1, p = 0.1, list = F)
Airbnb_combined <- Airbnb[-test_index,]
Airbnb_test <- Airbnb[test_index,]
```

Remove test_index:

```
rm(test_index)
```

Split Training and Validation Sets from airbnb_combined to train our models:

airbnb_train will constitute 80% and validation the remaining 20% of airbnb_combined.

```
set.seed(123, sample.kind = "Rounding")
test_index <- createDataPartition(y = Airbnb_combined$price, times = 1,
                                  p = 0.2, list = F)
Airbnb_train <- Airbnb_combined[-test_index,]
validation <- Airbnb_combined[test_index,]
```

Remove test_index once again:

```
rm(test_index)
```

The Loss Function / RMSE is defined as follows

Results will be based on calculating the RMSE on the test set. The Root Mean Square Error or RMSE weighs large errors more heavily and tends to be the preferred measurement of error in regression models when large errors are undesirable.

```
RMSE <-function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

Baseline Model: Median Model

This model will use the median price of the training set to calculate a "baseline" RMSE as a benchmark.

```
Airbnb_train_median <-median(Airbnb_train$price)
```

Table the Results

```
MM_RMSE <-RMSE(validation$price, Airbnb_train_median)
results_table <-tibble(Model_Type = "Baseline Median", RMSE = MM_RMSE) %>%
  mutate(RMSE = sprintf("%0.2f", RMSE))
knitr::kable(results_table)
```

The Baseline Model achieves an RMSE of 105.15.

Formula:

Vectorize the optimal formula that will be used for most Models:

The formula has been determined by the above EDA as well as experimentation on the training models. Additionally, the formula will reduce lines of code.

```
Airbnb_form <-price ~ number_of_reviews + latitude + longitude + minimum_nights
```

Linear Model:

```
lm_Airbnb <-lm(Airbnb_form, data = Airbnb_train)
```

Create the prediction

```
lm_preds <-predict(lm_Airbnb, validation)
```

Table the Results

```
LM_RMSE <-RMSE(validation$price, lm_preds)
results_table <-tibble(Model_Type = c("Baseline Median", "Linear"),
                       RMSE = c(MM_RMSE, LM_RMSE)) %>%
```

```
  mutate(RMSE = sprintf("%0.2f", RMSE))
knitr::kable(results_table)
```

The Linear Model vastly improves upon the Baseline Model with an RMSE of 101.88.

Elastic Net Regression with glmnet:

This model introduces regularization on a generalized linear model using penalized maximum likelihood and tuning optimal alpha and lambda parameters.

Set the seed for reproducibility:

```
set.seed(123, sample.kind = "Rounding")
train_enr <- train(Airbnb_form, data = Airbnb_train, method = "glmnet",
                   preProcess = c("center", "scale"),
                   tuneLength = 10, trace = F)
```

Confirm the optimal alpha and lambda parameters

```
train_enr$bestTune
```

alpha = 0.4, and lambda = 11.1

Create the Prediction:

```
elastic_preds <-predict(train_enr, validation)
```

Table the Results

```
ENR_RMSE <-RMSE(validation$price, elastic_preds)
results_table <-tibble(Model_Type = c("Baseline Median", "Linear",
```

```
                                "Elastic Net Regression"),
                    RMSE = c(MM_RMSE,LM_RMSE, ENR_RMSE)) %>%
  mutate(RMSE = sprintf("%0.2f", RMSE))
knitr::kable(results_table)
```

The Elastic Net with Regression underperforms the Linear Model with an RMSE of 101.92.

Regression Tree Model with rpart:

Set the seed for reproducibility:

```
set.seed(123, sample.kind = "Rounding")
train_rpart <- train(Airbnb_form, method = "rpart", data = Airbnb_train,
                     tuneGrid = data.frame(cp = seq(0, 0.05, len = 25)),
                     preProcess = c("center", "scale"))
```

Check the bestTune to find the final complexity parameter used for the model:

```
train_rpart$bestTune
```

The final cp is 0.05.

Create the Prediction:

```
rt_preds <-predict(train_rpart, validation)
```

Table the Results:

```
RT_RMSE <-RMSE(validation$price, rt_preds)
```

```
results_table <-tibble(Model_Type = c("Baseline Median", "Linear",
                                      "Elastic Net Regression",
                                      "Regression Tree"),
                       RMSE = c(MM_RMSE,LM_RMSE, ENR_RMSE, RT_RMSE)) %>%
  mutate(RMSE = sprintf("%0.2f", RMSE))
knitr::kable(results_table)
```

The Regression Tree model underperforms the both the Linear Model and the Elastic Net Regression Model with an RMSE of 102.66.

"Bagging Tree" -- Bootstrap Aggregating Model:

Set the seed for reproducibility:

```
set.seed(123, sample.kind = "Rounding")
train_bag <-train(Airbnb_form, data = Airbnb_train, method = "treebag",
                  importance = T, tuneLength = 10,
                  preProcess = c("center","scale"))
```

Create the Prediction

```
bag_preds <-predict(train_bag, validation)
```

Table the Results

```
BAG_RMSE <-RMSE(validation$price, bag_preds)
results_table <-tibble(Model_Type = c("Baseline Median", "Linear",
                                      "Elastic Net Regression",
                                      "Regression Tree","BAG"),
                       RMSE = c(MM_RMSE,LM_RMSE, ENR_RMSE, RT_RMSE,
                                BAG_RMSE)) %>%
  mutate(RMSE = sprintf("%0.2f", RMSE))
```

```
knitr::kable(results_table)
```

The BAG Tree Model underperforms the Linear, Elastic Net Regression, and Regression Tree Model with an RMSE of 103.74

Final Model

kNN Model:

Set the seed for reproducibility:

```
set.seed(123, sample.kind = "Rounding")
train_knn <- train(Airbnb_form, method = "knn", data = Airbnb_train,
                   tuneLength = 5, preProcess = c("center","scale"))
```

Find best value for k:

```
train_knn$bestTune
```

k = 13 is the final value used for the model.

Create the Prediction

```
knn_preds <- predict(train_knn, validation)
```

Table the Results

Model Type	RMSE
Baseline Median	105.15
Linear	101.88
Elastic Net Regression	101.92
Regression Tree	102.66
BAG	103.74
kNN	105.25

The kNN Model underperforms all Models with an RMSE of 105.25

Conclusion

The Linear Model is slightly better than the Baseline Model with a final RMSE of 101.88 on the test set -- a 3.10% increase in predictive performance. Elastic Net Regression has only 0.04% underperformed the RMSE of Linear model. The worst model was kNN with RMSE of 105.25 that is only 0.10 RMSE difference between Baseline Median of 105.15 RMSE. (All Models were tested on the test set though only the best performing Model (Linear Model) was selected for the final report).

In general, machine learning algorithm where the predicted output is continuous and has a constant slope was the most successful (Linear).