# Airbnb: Where to Stay in Amsterdam?

## IST 652: Scripting for Data Analysis

## Final Project Report

By Group 9:
Rahul Rathod
Yash Kapadia

# 1. Introduction

Amsterdam is the capital and most popular city of the Netherlands. It is colloquially referred to as the "Venice of the North", attributed by the large number of canals which form a UNESCO World Heritage Site. Amsterdam is the heaven of art because of its high-density distribution of museums and art galleries. It has large amount of collections of Vincent Willem van Gogh and Rembrandt Harmen zoon van Rijn. Amsterdam is also famous of its open culture to sex and cannabis. According to https://www.dw.com/en/how-amsterdam-is-fighting-mass-tourism/a-47806959, there were 19 million tourists visiting Amsterdam in 2019 which brings the high demands of hotels.

Airbnb is an online marketplace for arranging or offering lodging, primarily homestays, or tourism experiences (https://en.wikipedia.org/wiki/Airbnb). More and more people choose to stay in a local house when they are travelling. The prices of the houses vary a lot depending on the location, the size, the service or the surroundings of the houses.

# 2. Project Goal

Airbnb renting has always been interesting. Some spots can be rented in higher prices, whereas others can only be rented in very low prices. Suppose you would like to be a host in Amsterdam - in order to be a successful one, it is a good idea to investigate the host listing data from airbnb and see what are the major factors that influence the room prices. In this project, we are going to do this job for those who want to be a successful host!

Airbnb has seen a meteoric growth since its inception in 2008 with the number of rentals listed on its website growing exponentially each year. Airbnb has successfully disrupted the traditional hospitality industry as more and more travellers, not just the ones who are looking for a bang for their buck but also business travellers' resort to Airbnb as their premier accommodation provider. Are you planning your trip to Amsterdam and searching for a place to stay on Airbnb? Before diving into thousands of listings posted online, you might be interested in some of the high-level information about the Amsterdam Airbnb market.

## 2.1. Major Business Questions

- What is the number of new listings posted on Airbnb for Amsterdam, on a year to year basis?
- What is the distribution of listings in Amsterdam?
- What is the distribution of prices in Amsterdam Airbnb listings?
- What is the distribution of price in Amsterdam Airbnb listings?

- Which factors of the property could affect the rental price on Airbnb?
- What is the average price according to the location of the neighbourhood?
- What is the average price according to the time of the year?

# 3. Analysis

## 3.1. About the data

The dataset consists of 6 structured csv files and 1 semi structured json file. The files are downloaded from insideairbnb.com and give a snapshot of the Amsterdam situation on February 14th, 2020. The 'listings' file contains all the advertisements in Amsterdam. The listings_details file contains additional variables. The calendar has 365 records for each listing. It specifies the whether the listing is available on a particular day (365 days ahead), and the price on that day. In addition, a reviews file is available that contains reviews about particular listings provided by travellers, and shapefile of the neighbourhoods in Amsterdam. We have also used neighbourhood.geojson which is a dictionary that contains coordinates of neighbourhoods.

The Airbnb data of Amsterdam was retrieved from Kaggle (http://insideairbnb.com/get-the-data.html).The dataset includes the following files:

1. calendar.csv: The calendar has 365 records for each listing. It specifies the whether the listing is available on a particular day (365 days ahead), and the price on that day.
2. listings.csv: A listing is basically an advertisement. This file holds the most useful variables that can be used visualizations.
3. listings_details.csv: This file holds the same variables as the listing file plus 80 additional variables.
4. neighbourhood.csv: Simple file with the Dutch names of the neighbouhoods
5. reviews.csv: This is a simple file that can be used to count the number of reviews by listing (for a specific period).
6. reviews_details.csv: This file holds the full details of all reviews, and can also be used for instance for text mining.
7. neighbourhoods.geojson: This is the shape file that can be used in conjunction with interactive maps (such as Leaflet for R of the Python folium package).

## 3.2. Data Preprocessing

### 3.2.1. Data Preparation for Exploratory Data Analysis

The data preparation steps include:

- Changing the data types of the required columns.
- Dropping columns with NA values.
- Filtering out rows which are not required.

- Merging of listings and listings_details csv files for data analysis.

### 3.2.2. Fetching Amsterdam Neighbourhoods Coordinates

We are using neighbourhoods.geojson to get the longitude and latitude of the center of each neighbourhood.

In [15]:
```
import geopandas as gpd
geo_ams = gpd.read_file('neighbourhoods.geojson')
geo_ams.head()
```

Out[15]:

| | neighbourhood | neighbourhood_group | geometry |
|---|---|---|---|
| 0 | Bijlmer-Oost | None | MULTIPOLYGON Z (((4.99167 52.32444 43.06929, 4... |
| 1 | Noord-Oost | None | MULTIPOLYGON Z (((5.07916 52.38865 42.95663, 5... |
| 2 | Noord-West | None | MULTIPOLYGON Z (((4.93072 52.41161 42.91539, 4... |
| 3 | Oud-Noord | None | MULTIPOLYGON Z (((4.95242 52.38983 42.95411, 4... |
| 4 | IJburg - Zeeburgereiland | None | MULTIPOLYGON Z (((5.03906 52.35458 43.01664, 5... |

In [16]:
```
geo_ams["longitude"] = geo_ams.centroid.x
geo_ams["latitude"] = geo_ams.centroid.y
geo_ams.drop('neighbourhood_group', axis=1, inplace=True)
geo_ams.head()
```

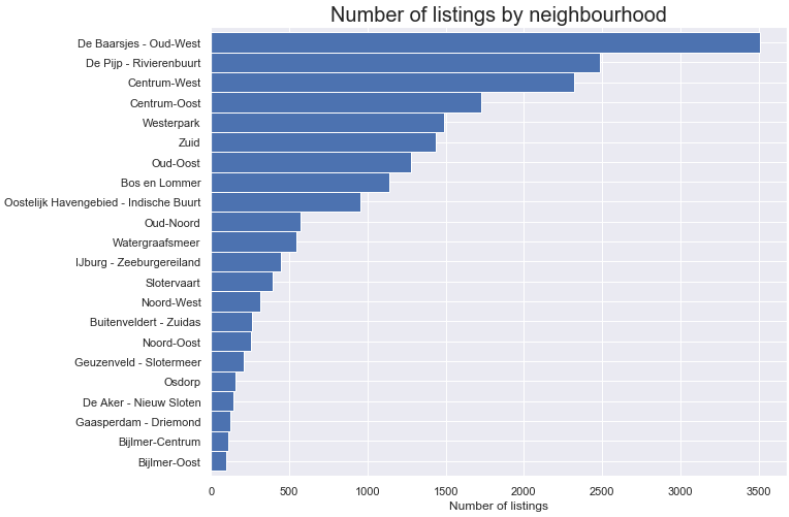
Out[16]:

| | neighbourhood | geometry | longitude | latitude |
|---|---|---|---|---|
| 0 | Bijlmer-Oost | MULTIPOLYGON Z (((4.99167 52.32444 43.06929, 4... | 4.977317 | 52.320279 |
| 1 | Noord-Oost | MULTIPOLYGON Z (((5.07916 52.38865 42.95663, 5... | 5.003523 | 52.398339 |
| 2 | Noord-West | MULTIPOLYGON Z (((4.93072 52.41161 42.91539, 4... | 4.894589 | 52.415306 |
| 3 | Oud-Noord | MULTIPOLYGON Z (((4.95242 52.38983 42.95411, 4... | 4.910489 | 52.394068 |
| 4 | IJburg - Zeeburgereiland | MULTIPOLYGON Z (((5.03906 52.35458 43.01664, 5... | 4.998377 | 52.360899 |

## 3.3. Exploratory Data Analysis

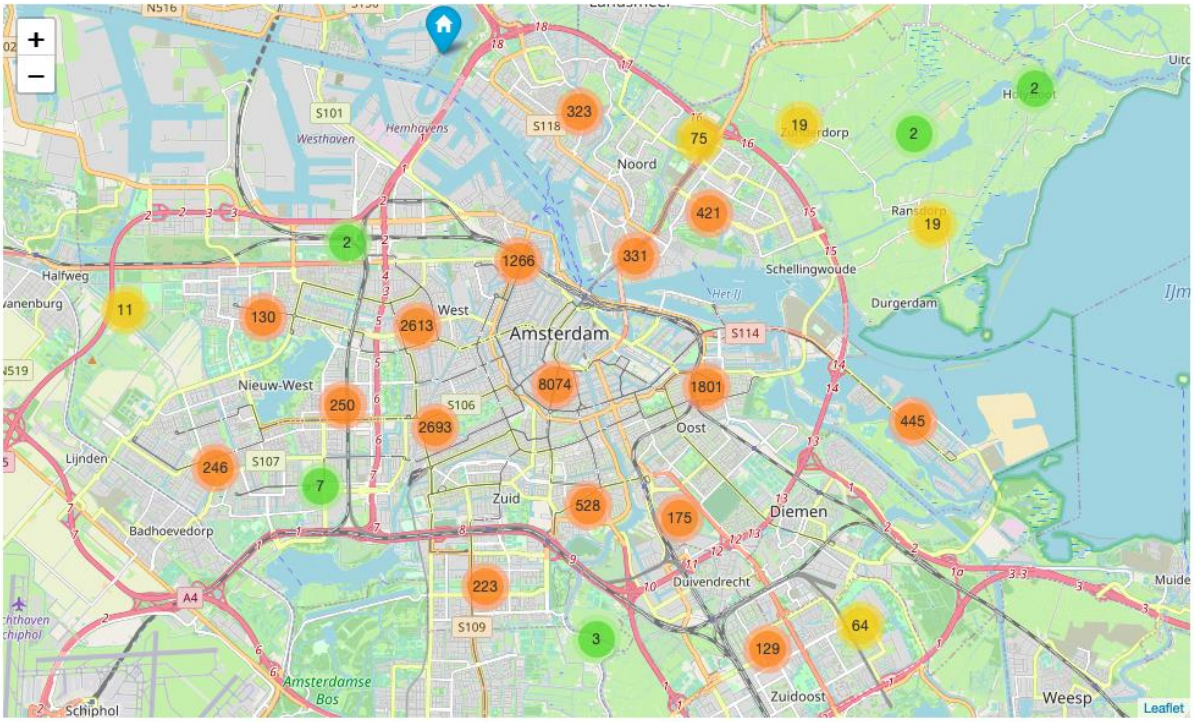
### Numbers of listings in each neighbourhood of Amsterdam?

Neighbourhood "De Baarsjes - Oud-West" holds most listings, and altogether eight neigbourhoods have over one thousand listings.



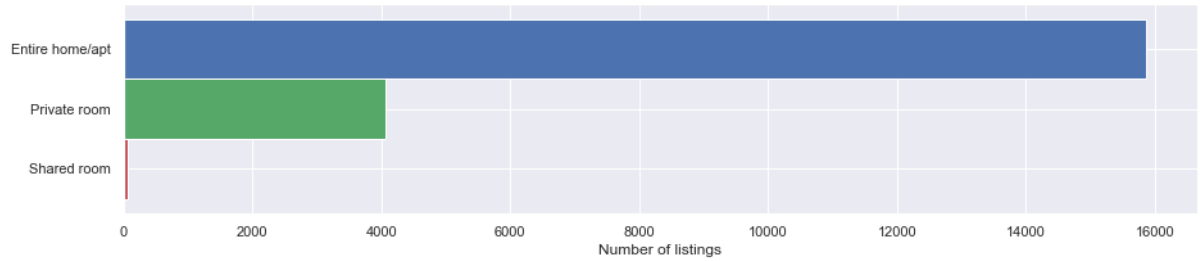

## Most Number of Listings in Amstedam

This is an interactive map built using folium map library which shows the number of listings as clusters and we can drill down to a particular are to see the number of listings. Below, you can see that most listings are in the centre of the city.



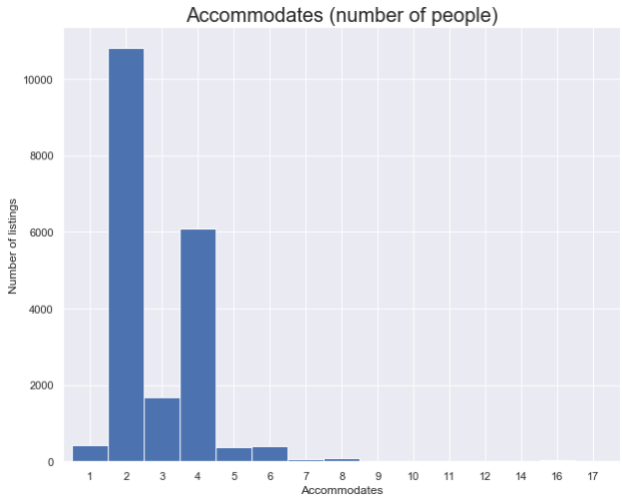

## Room types

The room type is very important in Amsterdam, because Amsterdam has a rule that Entire homes/apartments can only be rented out via Airbnb for a maximum of 60 days a year. Below, we can see that this restriction applies to most of the listings.



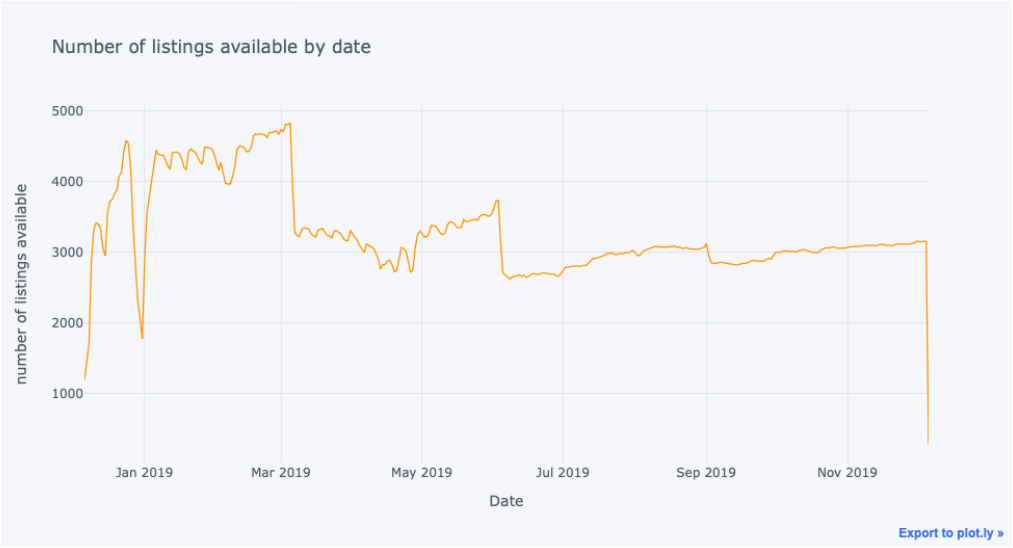

## Accommodates (number of people)

There are more number of listings for 2 people. Airbnb uses a maximum of 16 guests per listing. However, Amsterdam has an additional restriction. Due to fire hazard considerations and also taking possible noisy group into account, owners are only allowed to rent their property to groups with a maximum of 4 people. This actually means that the listings that indicate that the maximum number of people is above 4 are breaking this rule!



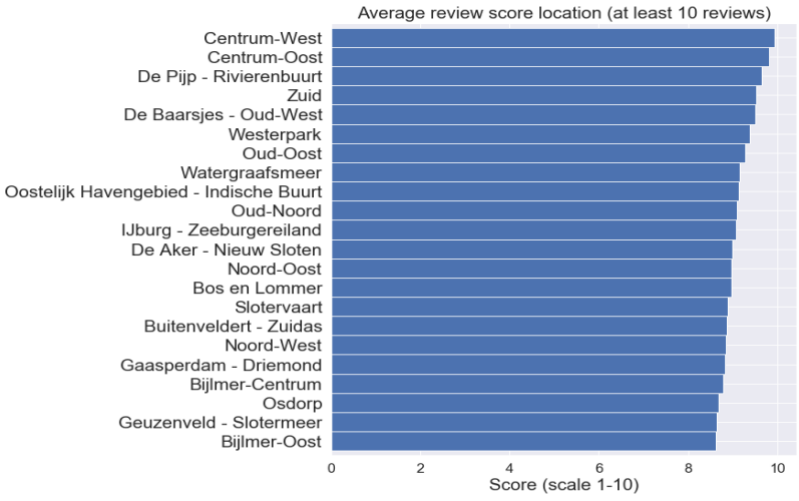

## Availability over Time

In order to analyze the number of listings available for the customers we have plotted the graph for number of listings available for each date in the year 2019. Below, we see that up to three months ahead, there are generally more accomodations available than further into the future. Reasons for this might be that hosts are more actively updating their calendars in this timeframe. This graph is interactive, and hovering over the points will show you a tooltip with the "number of available listings" and "weekday" by date.




## Average Review Scores for Neighbourhood

In this graph, we have explored the average reviews for the neighbourhoods where the number of reviews is greater than or equal to 10. Below we see that the central neighbourhoods, which were generally also the most expensive, generally also score higher on location review score.



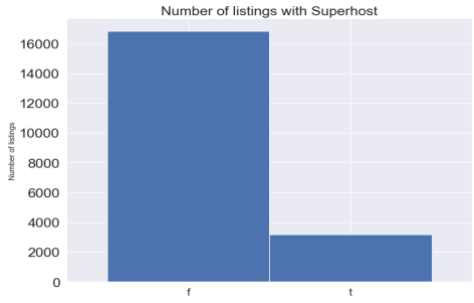

## Super host

As a Super host, you'll have more visibility, earning potential, and exclusive rewards. It's our way of saying thank you for your outstanding hospitality. How to become a Super host:

- Every 3 months, we check if you meet the following criteria. If you do, you'll earn or keep your Super host status.
- Super hosts have a 4.8 or higher average overall rating based on reviews from at least 50% of their Airbnb guests in the past year.

- Super hosts have hosted at least 10 stays in the past year or, if they host longer-term reservations, 100 nights over at least 3 stays.
- Super hosts have no cancellations in the past year, unless there were extenuating circumstances.
- Super hosts respond to 90% of new messages within 24 hours.

Below, we can see that only a small portion of the listings in Amsterdam do have a host who is Super host

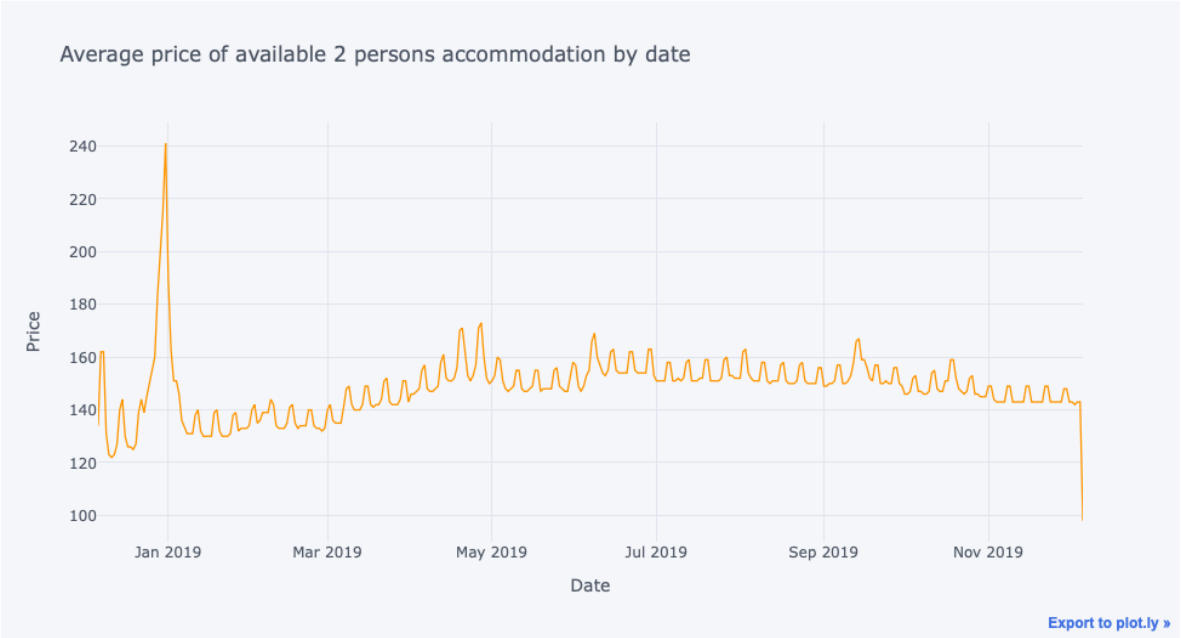



## Price Analysis

### Time series analysis - price by date

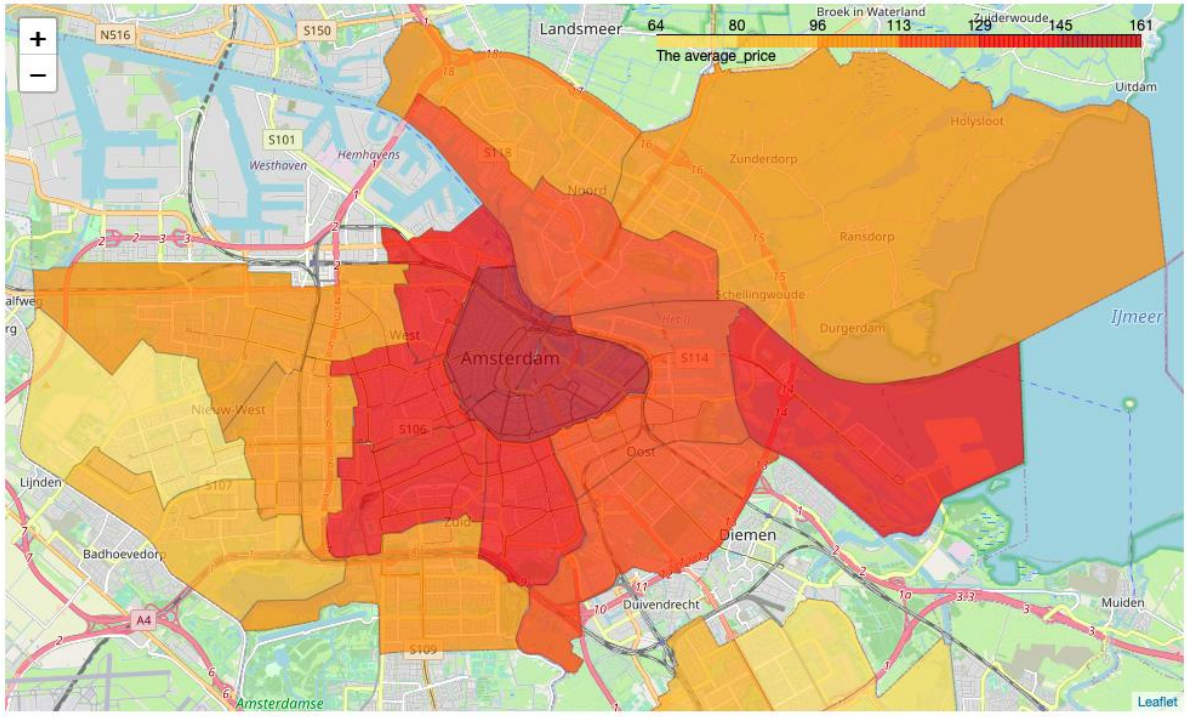
Here we have merged the calendar data and listing_details data in order to find patterns in the average price of the listings over time.

Below, you see the average price of all 2-persons accomodations flagged as available by date. The peak of average price 240 Euro is at December 31st, and the cyclical pattern is due to higher prices in weekends. This graph is interactive, and hovering over the points will show you a tooltip with the average price and weekday by date.



## Distribution of price on the map

We can see that it is true that the properties locating in Centrum-Oost and Centrum-West are more expensive than the others. It is worthy to mention these two neighbourhoods are the places where many famous places locate such as Anne Frank house, Red light district, Amsterdam museum. Also, it is interesting to see that IJburg - Zeeburgereiland also shows the high price even though it is not very close to the center. A reason could be the nice seaside scenery when living on the island.




## 3.4. Predictive Analytics

### Which factors of the property could affect the rental price on Airbnb?

Which factors of the property could affect the rental price on Airbnb? The answer of this question can be useful for people from Amsterdam who wants to start renting their properties on Airbnb, or tourists who are going to visit Amsterdam and want to estimate the cost of accommodation.

#### 3.4.1. Data Preparation for Regression and Statistical Analysis

The data preparation steps include:

1. **Data Cleaning**: There are 96 columns in the airbnb data. But there are columns not valuable for our analysis such as listing_url, scrape_id. So, we can firstly filter some columns that are not describing the property.

2. **Drop the unavailable samples**: We would like to focus on the properties that are available regularly. So, we can remove the properties are not available in the past recent months.

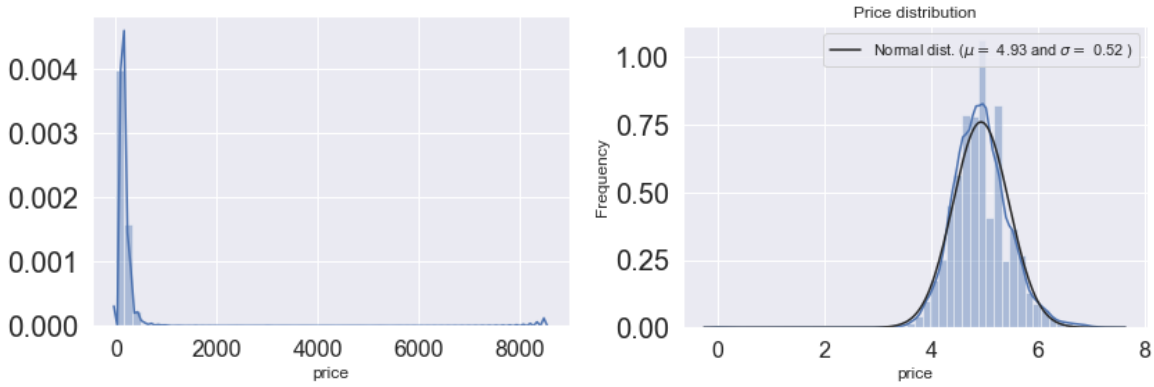
3. **Drop the samples missing information:** Also, we would like to focus on the valid property only which means the property has a valid price and reviews from previous guests to verify the authenticity.

4. **Examine and Dealing with missing values**
    a. Drop the feature having too many missing values: More than 97% of properties do not have the square feet recorded. So, we remove the square_feet from the data.
    b. Fill the missing sub-reviews by corresponding review_scores_rating: If a review score for a sub-catogery is missing, we fill it by its average review score which is review_score_rating.
    c. Fill the missing bathrooms, bedrooms, beds by 0: If the values of bathrooms, bedrooms, beds are missing, we can understand there is no bathroom, bedroom, bed in this property. So, we fill them by 0.
    **d.** Fill the missing cleaning fee by mode

5. **Convert the type of some features:** We can see there are some features related to price that should be numerical rather than categorical, such as price, cleaning_fee, extra_people. That is because the money data is recorded as a string in the format of '50$'. So we convert these money-related features to numbers.

6. **Normalizing the distribution of price**: examine the normality of the price distribution

    We can see that the distribution of price is highly skewed. Also, there are several outlier when the price is above 2000. We firstly remove outliers and then correct the distribution of the price by log function




7. **Getting dummy categorical features:** We used one hot encoder to encode the categorical features.
8. **Merging all the features**: Merged the features with dummy features by IDs of properties and Drop the duplicate categorical features that have already been encoded
9. **Drop overfitting features:** Drop the features that have more than 95% same values for all the data.

### 3.4.2. Regression Modelling

Given the information of the property, we would like to predict its price. In this section, we are going to fit our data into multiple linear regression model and find the top 10 features affecting the price

To evaluate the performance of models, we firstly separate the data into training set and test set. The training set will be used to select and train the model, and the test set will be used to test the performance of the model.

**Model Fitting**

```
regression_model= regressor.fit(X_train, y_train)
print('linear', rmse(y_train, regression_model.predict(X_train)))
```

```
linear 0.35573067449207396
```

The above rmsle score shows our model can accurately predict the price given the information data of the property. Once we have the trained model, more importantly, we would like to know which factor effect the price most. This can be done by examining the coef_ attribute of linear regression model which represent the importance scores of all features.

**Top 10 Factors influencing the price**

| | Importance |
|---|---|
| room_type_Entire home/apt | 0.531568 |
| neighbourhood_cleansed_Centrum-West | 0.399425 |
| neighbourhood_cleansed_Centrum-Oost | 0.365117 |
| neighbourhood_cleansed_De Pijp - Rivierenbuurt | 0.224000 |
| neighbourhood_cleansed_Zuid | 0.195987 |
| room_type_Private room | 0.184985 |
| neighbourhood_cleansed_De Baarsjes - Oud-West | 0.180298 |
| neighbourhood_cleansed_Westerpark | 0.174293 |
| neighbourhood_cleansed_Oud-Oost | 0.134577 |
| accommodates | 0.110956 |

According to the above table, there is no surprise that the room type, neighbourhood and number of accommodates are top 3 most important features to the price. Bigger house can accommodate more people therefore can be more expensive. Other related factors are the number of beds and the number of bathrooms. Location is definitely another important

factor to the price. The review scores for location can reflect the neighbourhood the property locates and the how convenience the public transportation nearby. Property locating in the centrum neighbourhood can be more expensive than the others. This can be verified by the previous visualization of the medians of the prices of different neighbourhoods.

# 4. Conclusion and Recommendations

In this project, we have analysed the Airbnb houses in Amsterdam using the Airbnb public data. According to the our analysis, the price of the proper on Airbnb in Amsterdam mostly varies based on the number of accommodates/the number of bedrooms, the location and the extra fee (cleaning fee/fee for extra people). Bigger house and house close to city centre or public transportation can be more expensive than the other. And the price can be higher if the extra fee is included in the price. Also, properties that locate near nightlife spots can be more expensive.

**Here are my recommendations**:

If you want to rent your house in Amsterdam on Airbnb:

- If you have a house in the popular neighbourhoods, then congratulations, you can rent your house in a good price.
- Ask for reviews from your guests. People trust on the previous review than your description and pictures. Showing on time when checking-in, being honest and providing good service (such as allowing extra guest) can really helpful on gaining higher review scores.

If you are going to visit Amsterdam and want to live in a local house:

- The more bedrooms and bathrooms in the house, the price can be more expensive. If you can find more travelling partners who don't mind sharing bedrooms or bathrooms with you, the average price per person can be very fair.
- It is always expensive to live in the centre. If you really care about living close to centre, I recommend to live in the neighbourhood of Oud-West which has the most number of houses but the median price ranks the 5th expensive neighbourhood. If you have an empty wallet, considering the mature public transportation network of Amsterdam, I recommend to live in the neighbourhood far away from the centre. Even if you live the suburb, a less than 30-minutes train can take you to the centre.
- Places having many restaurants and shopping places nearby can be expensive. You can choose other places if you don't necessarily eat or shop near the place you live.
- If you are a night owl and a lover to night clubs of Amsterdam, unfortunately, you might need to prepare more budgets on accommodation.

There are might be other factors can affect the price such as the decoration of the property and the service the host can provide. Unfortunately, we can't go through the influence of all of possible factors due to the unavailability of the data. However, I believe that our analysis is extremely valuable for the hosts from Amsterdam who wants to define the price of their proprieties or tourists who want to have an estimate of the accommodation cost based on their requirements and budgets. This analysis can also be applied to any other city in the world.

# 5. Program Description:

The program for the data analysis is structured in a way that it starts from importing the required libraries, data cleaning and exploration, generating visualizations, fitting data into linear model to do predictions and generating top 10 features that influence the price of airbnb listing.

Here the complete description:

### Step 1- Importing Libraries

```
# importing the required packages for the analysis
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
import folium
from folium.plugins.marker_cluster import MarkerCluster
from folium.plugins import FastMarkerCluster
import plotly
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected=True)
import cufflinks as cf
cf.go_offline()
cf.set_config_file(offline=False, world_readable=True)
sns.set()
from scipy import stats
from scipy.stats import norm
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression,Lasso, Ridge
import geopandas as gpd
import warnings
warnings.filterwarnings('ignore')
```

### Step 2- Loading the data

```
# importing the required data files in to pandas dataframe
calendar = pd.read_csv('calendar.csv')
listings = pd.read_csv('listings.csv')
listing_details = pd.read_csv('listings_details.csv')
neighbourhood = pd.read_csv('neighbourhoods.csv')
review = pd.read_csv('reviews.csv')
reviews_details = pd.read_csv('reviews_details.csv')
```

### Step 3- Data Cleaning

```
calendar['date'] = pd.to_datetime(calendar['date'], errors ='coerce')

listings['last_review'] = pd.to_datetime(listings['last_review'], errors ='coerce')

listings = listings[listings['name'].notna()]

listings = listings[listings['host_name'].notna()]

listings = listings.drop('neighbourhood_group',axis='columns')

listing_details =
listing_details[['id','property_type','accommodates','first_review','review_scores_value',


'review_scores_cleanliness','review_scores_location','review_scores_accuracy',

                'review_scores_communication','review_scores_checkin',

                'review_scores_rating','maximum_nights',

                'listing_url','host_is_superhost','host_about',

                'host_response_time','host_response_rate',

                'street','weekly_price','monthly_price','market']]

final_listings = listings.merge(listing_details,on='id')
```

### Step 4- Exploratory Data Analysis

```
# Number of Listings by Neighbourhood
feq=final_listings['neighbourhood'].value_counts().sort_values(ascending=True)
feq.plot.barh(figsize=(10, 8), width=1)
plt.title("Number of listings by neighbourhood", fontsize=20)
```

```
plt.xlabel('Number of listings', fontsize=12)
plt.show()
# Map showing the number of listings in Amsterdam

lats2018 = final_listings['latitude'].tolist()
lons2018 = final_listings['longitude'].tolist()
locations = list(zip(lats2018, lons2018))

map1 = folium.Map(location=[52.3680, 4.9036], zoom_start=11.5)
FastMarkerCluster(data=locations).add_to(map1)
map1
freq = final_listings['room_type']. value_counts().sort_values(ascending=True)
freq.plot.barh(figsize=(15, 3), width=1, color = ["r","g","b"])
plt.xlabel('Number of listings', fontsize=12)
plt.show()
accom=final_listings['accommodates'].value_counts().sort_index()
accom.plot.bar(figsize=(10, 8), color='b', width=1, rot=0)
plt.title("Accommodates (number of people)", fontsize=20)
plt.ylabel('Number of listings', fontsize=12)
plt.xlabel('Accommodates', fontsize=12)
plt.show()
sum_available = calendar[calendar.available ==
"t"].groupby(['date']).size().to_frame(name= 'available').reset_index()
sum_available['weekday'] = sum_available['date'].dt.day_name()
sum_available = sum_available.set_index('date')

sum_available.iplot(y='available', mode = 'lines', xTitle = 'Date', yTitle = 'number of listings
available',\
        text='weekday', title = 'Number of listings available by date')
fig = plt.figure(figsize=(20,10))
plt.rc('xtick', labelsize=16)
plt.rc('ytick', labelsize=20)

ax1 = fig.add_subplot(121)
feq = final_listings[final_listings['number_of_reviews']>=10]
feq1 =
feq.groupby('neighbourhood')['review_scores_location'].mean().sort_values(ascending=Tr
ue)
ax1=feq1.plot(color='b', width=1, kind='barh')
plt.title("Average review score location (at least 10 reviews)", fontsize=20)
plt.xlabel('Score (scale 1-10)', fontsize=20)
plt.ylabel("")
feq=final_listings['host_is_superhost'].value_counts()
feq.plot.bar(figsize=(10, 8), width=1, rot=0)
```

```
plt.title("Number of listings with Superhost", fontsize=20)
plt.ylabel('Number of listings', fontsize=12)
plt.show()
```

Step 4- Data Analysis for Airbnb listing price

```
listings.index.name = "listing_id"
calendar = pd.merge(calendar, final_listings[['id','accommodates']], left_on = "listing_id", right_on = 'id', how = "left")
calendar.sample(15)
calendar['date'] = pd.to_datetime(calendar['date'], errors ='coerce')
calendar['price'] = calendar['price'].astype(str)
calendar['price'] = calendar['price'].apply(lambda x:x.lstrip('$'))
calendar['price'] = calendar['price'].apply(lambda x:x.replace(',',''))
calendar['price'] = calendar['price'].astype(float)
average_price = calendar[(calendar.available == "t") & (calendar.accommodates == 2)].groupby(['date'])[['price']].mean().astype(np.int64).reset_index()
average_price['weekday'] = average_price['date'].dt.day_name()
average_price = average_price.set_index('date')

average_price.iplot(y='price', mode='lines', xTitle='Date', yTitle='Price',
   text='weekday', title='Average price of available 2 persons accommodation by date')
```

## Step 5- Predictive Analytics

```
#examine the taget variable price
listings_details['price'].describe()
sns.set_style('darkgrid')
sns.distplot(listings_details['price'])
#we can see that target values is highley skewed
#removing outliers that are above 2000
listings_details = listings_details[listings_details['price']<2000]
def correct_dist(price):
   price = np.log1p(price)

   #Check the new distribution
   sns.distplot(price , fit=norm);

   # Get the fitted parameters used by the function
   (mu, sigma) = norm.fit(price)
   print( '\n mu = {:.2f} and sigma = {:.2f}\n'.format(mu, sigma))

   #Now plot the distribution
   plt.legend(['Normal dist. ($\mu=$ {:.2f} and $\sigma=$ {:.2f} )'.format(mu, sigma)],
```

```
            loc='best')
    plt.ylabel('Frequency')
    plt.title('Price distribution')

    #Get also the QQ-plot
    plt.show()
    return price

listings_details['corrected_price'] = correct_dist(listings_details['price'])

#creating dummy variables for all categorical objects
category_features = list(listings_details.dtypes[listings_details.dtypes == 'object'].index)
category_features.remove('name')
category_features
def one_hot_encode(data, columns):
    onehot = pd.get_dummies(data[columns])
    onehot['id'] = data['id']
    # move id column to the first column
    fixed_columns = [onehot.columns[-1]] + list(onehot.columns[:-1])
    onehot = onehot[fixed_columns]
    return onehot
#creating dummies for categorical features
dummy = one_hot_encode(listings_details,category_features)
dummy.head()
#mergeing the encoded df with original df
final_df = pd.merge(listings_details,dummy)
#droping the categorical features
final_df.drop(category_features, axis=1, inplace=True)
#dropping irrelevant columns
y = final_df['corrected_price']
X = final_df.drop(['id', 'name', 'latitude', 'longitude', 'price','corrected_price'], axis=1)
X.shape
#removing overfitting variables
overfit = []
for i in X.columns:
    counts = X[i].value_counts()
    zeros = counts.iloc[0]
    if zeros / len(X) * 100 > 95:
        overfit.append(i)
overfit
X.drop(overfit, axis=1, inplace=True)
```

## Step 6- Regression Modelling

```
#splitting data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
regressor = LinearRegression()
# Root Mean Squared Logarithmic Error ,   RMSE
def rmse(y, y_pred):
   return np.sqrt(mean_squared_error(y, y_pred))
regression_model= regressor.fit(X_train, y_train)
print('linear', rmse(y_train, regression_model.predict(X_train)))
linear_model_list = {'Importance': regression_model}
feature_importance = []
for model_name, model in linear_model_list.items():
   feature_importance.append(model.coef_)
```

**Step 7- Feature Importance**

```
feature_importance = pd.DataFrame(feature_importance, columns=X_train.columns)
feature_importance.index = list(linear_model_list.keys())
feature_Importance =feature_importance.T
top5_factors =
feature_Importance.sort_values(by=['Importance'],ascending=False).head(10)
top5_factors
```