Project Name: AirBnb Data Analysis

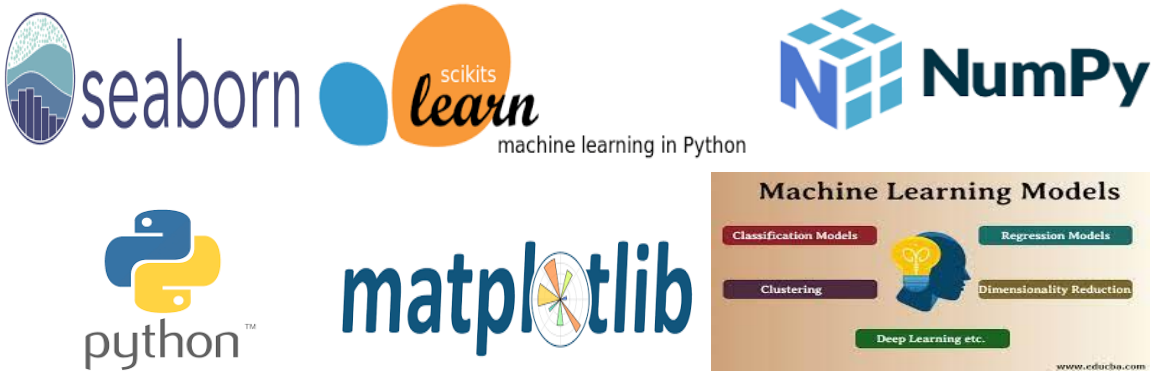
- Md Abdul Mazed Siddiki
- Madzella Yannick
- Mohammed Danish Mustafa

General Description:

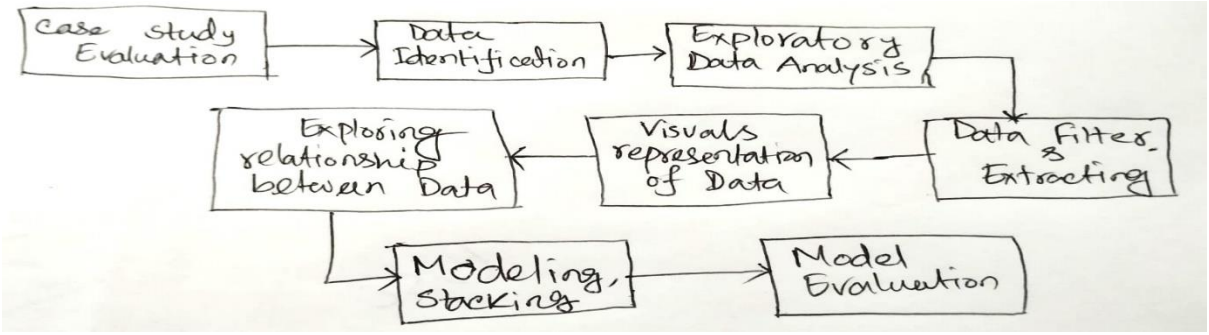
Since 2008, guests and hosts have used Airbnb to expand on travelling possibilities and present more unique, personalized way of experiencing the world. This dataset describes the listing activity and metrics in Paris, France. The content of this data file includes all needed information to find out more about hosts, geographical availability, necessary metrics to make predictions and draw conclusions. The objective of the project is to perform data visualization techniques to understand the insight of the data. This project aims to apply Exploratory Data Analysis (EDA) and using Machine Learning models to predict prices.

Tools used:

For this project we used Python-Numpy, Pandas, Seaborn, Matplotlib, Excel, and jupyter notebook to build the whole framework.



Design Details:



Dataset:

The dataset had information regarding the reviews with respect to listing id. This data has all the information regarding the listings. It had Host name, location, neighbourhood, price, review score and number of reviews, latitude, longitude, room type.

The features in the dataset can be described as follows:

```
#   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              52725 non-null  int64
 1   name                            52725 non-null  object
 2   host_id                         52725 non-null  int64
 3   host_name                       52679 non-null  object
 4   neighbourhood_group             0 non-null      float64
 5   neighbourhood                   52725 non-null  object
 6   latitude                        52725 non-null  float64
 7   longitude                       52725 non-null  float64
 8   room_type                       52725 non-null  object
 9   price                           52725 non-null  int64
 10  minimum_nights                  52725 non-null  int64
 11  number_of_reviews               52725 non-null  int64
 12  last_review                     38216 non-null  object
 13  reviews_per_month               38217 non-null  float64
 14  calculated_host_listings_count  52725 non-null  int64
 15  availability_365                52725 non-null  int64
```

Exploratory Data Analysis (EDA) –

EDA Using Python, which is an approach to analyse the datasets to summarize their main characteristics in form of visual methods. EDA is nothing but a data exploration technique to understand various aspects of the data. The main aim of EDA is to obtain confidence in a data to an extent where we are ready to engage a machine learning model. EDA is important to analyse the data it's a first steps in data analysis process. Exploratory data analysis helps us to finding the errors, discovering data, mapping out data structure, finding out anomalies. Exploratory data analysis is important for business process because we are preparing dataset for deep through analysis that will detect your business problem.

Steps Involved in EDA: -

- Data Sourcing
- Data Cleaning and filtering
- Data analysis with visualisation

Sourcing:

Data Sourcing is the process of gathering data from multiple sources as external or internal data collection. There are two major kind of data which can be classified according to the source: 1. Public data 2. Private data

Data Cleaning and filter:

After collecting the data, the next step is data cleaning. Data cleaning means that you get rid of any information that doesn't need to be there and clean up by mistake. Data Cleaning is the process of clean the data to improve the quality of the data for further data analysis and building a machine learning model. The benefit of data cleaning is that all the incorrect and irrelevant data is gone and we get the good quality of data which will help in improving the accuracy.

Data analysis with visualisation:

Visualisation is the presentation of the data in the graphical or visual form to understand the data more clearly. Visualisation is easy to understand the data. Easily analyse the data and summarize it. Easily understand the features of the data. Help to find the trend or pattern of the data. Help to get meaningful insights from the data.

Important Charts for Visualisation:

1.Histogram

2.Bar Chart

3.Box plot

4.pie chart

5.Heatmap

6.Scatter plot

7.Line chart etc.

Data Exploration:

Checking the first 4 rows of the dataset and the dataset consist of 52725 observations (rows) and 16 features (columns).

1. EXPLORATORY DATA ANALYSIS

```
In [7]: listings.shape
Out[7]: (52725, 16)
```

```
In [4]: listings.head()
```

Out[4]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_re
0	4867396	Appartement 60m2 Rue Legendre 75017	9703910	Matthieu	NaN	Batignolles-Monceau	48.888800	2.320466	Entire home/apt	60	1	
1	7704653	Appart au pied de l'arc de triomphe	35777602	Claire	NaN	Batignolles-Monceau	48.876636	2.293724	Entire home/apt	200	1	
2	2725029	Nice appartment in Batignolles	13945253	Vincent	NaN	Batignolles-Monceau	48.883836	2.321031	Entire home/apt	80	3	
3	9337509	Charming flat near Batignolles	5107123	Julie	NaN	Batignolles-Monceau	48.892360	2.322338	Entire home/apt	60	2	
4	12928158	Spacious bedroom near the centre of Paris	51195601	Daniele	NaN	Batignolles-Monceau	48.889417	2.298321	Private room	50	1	

Data type:

In a dataset, data types refer to the specific format of the data values that are stored in each column or variable. Common data types include:

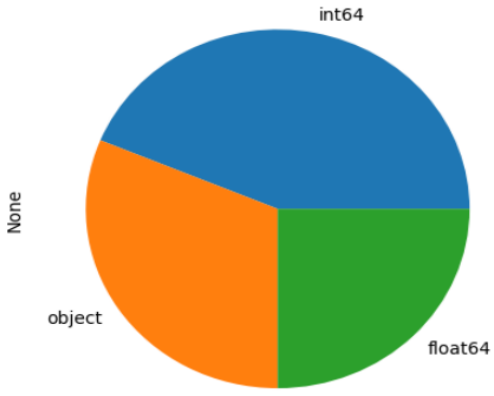
- Numeric data types: These include integers and floating-point numbers. Integers are whole numbers, while floating-point numbers have decimal places.
- Character data types: These include strings of text, such as names, addresses, and other alphanumeric values.
- Date/time data types: These are used to store dates and times in a specific format, such as YYYY-MM-DD for dates and HH:MM:SS for times.
- Boolean data types: These can have only two values, true or false, and are often used for logical comparisons.

It is important to know the data types of the variables in a dataset because it affects how the data can be analysed and manipulated. For example, numeric variables can be used in mathematical calculations, while character variables cannot. In addition, some statistical methods and machine learning algorithms require specific data types to work correctly.

In our dataset has following datatype

```
In [10]: # Visualization of data types proportion
         listings.dtypes.value_counts().plot.pie()
```

```
Out[10]: <AxesSubplot:ylabel='None'>
```



Data Cleaning:

Wrong data can skew and distort analysis results. Data input into Big Data analyses can be disorganized without any evidence of validity. Its complexity can more make it difficult to come at a set of proper validation constraints. This phase sets often complex validation rules and eliminates any known wrong data.

First we checked missing data in our dataset using isna() function. We show here percentage of nan values in each column and then keep the column with percentages of nan value less than 80% .

```
In [14]: # Percentage of nan values in each column
         (listings.isna().sum()/listings.shape[0]).sort_values(ascending=True)
```

```
Out[14]: id                                0.000000
         name                              0.000000
         host_id                           0.000000
         neighbourhood                     0.000000
         latitude                          0.000000
         longitude                         0.000000
         room_type                         0.000000
         price                             0.000000
         minimum_nights                    0.000000
         number_of_reviews                 0.000000
         calculated_host_listings_count    0.000000
         availability_365                  0.000000
         host_name                         0.000872
         reviews_per_month                 0.275164
         last_review                       0.275183
         neighbourhood_group               1.000000
         dtype: float64
```

Notice : Variables 'host_name', 'reviews_per_month', 'last_review' and 'neighbourhood_group' contain nan values

```
In [15]: # Let's keep columns with percentage of nan values less than 0.8
         listings = listings[listings.columns[(listings.isna().sum()/listings.shape[0]) < 0.8]]
```

Now we have deleted unnecessary column from dataset. We have deleted 'last_review' column because we don't want to work with time series.

```
In [21]: #  Delete 1 variables coz we dont work with time series
         listings1 = listings1.drop(['last_review'], axis = 1)
         listings1.head()
```

Out[21]:

	id	name	host_id	host_name	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	reviews_per_mor
0	4867396	Appartement 60m2 Rue Legendre 75017	9703910	Matthieu	Batignolles-Monceau	48.888800	2.320466	Entire home/apt	60	1	1	0
1	7704653	Appart au pied de l'arc de triomphe	35777602	Claire	Batignolles-Monceau	48.876636	2.293724	Entire home/apt	200	1	0	N
2	2725029	Nice appartment in Batignolles	13945253	Vincent	Batignolles-Monceau	48.883836	2.321031	Entire home/apt	80	3	1	0
3	9337509	Charming flat near Batignolles	5107123	Julie	Batignolles-Monceau	48.892360	2.322338	Entire home/apt	60	2	1	0
4	12928158	Spacious bedroom near the centre of Paris	51195601	Daniele	Batignolles-Monceau	48.889417	2.298321	Private room	50	1	2	2

Data splitting:

At this stage we want see how many columns numeric and categorical type in dataset are. Then for numeric features we must scale them for better performance and for the categorical features we use onehotencoding.

```
In [22]: # Split the data into numeric and categorical features

         numeric_features = [f for f, t in zip(listings1.columns, listings1.dtypes) if t in ['int64','float64']]

In [23]: numeric_features

Out[23]: ['id',
          'host_id',
          'latitude',
          'longitude',
          'price',
          'minimum_nights',
          'number_of_reviews',
          'reviews_per_month',
          'calculated_host_listings_count',
          'availability_365']

In [24]: categorical_features = [f for f in listings1.columns if f not in numeric_features]

In [25]: categorical_features

Out[25]: ['name', 'host_name', 'neighbourhood', 'room_type']
```

Scaling and encoding:

Scaling refers to the process of standardizing the range of features in the dataset. This is important because some algorithms are sensitive to the scale of the input features. For example, gradient descent-based algorithms work best when the features are on a similar scale. Scaling techniques include standardization, where the features are transformed to have zero mean and unit variance, and normalization, where the features are scaled to have a minimum and maximum value.

Encoding, on the other hand, refers to the process of converting categorical data into numerical data that can be used in machine learning algorithms. Categorical data is data that represents groups or categories, such as colors or types of animals. Some machine learning algorithms can only handle numerical data, so encoding is necessary to make categorical data usable. Common encoding techniques include one-hot encoding, where each category is represented as a binary vector, and ordinal encoding, where categories are represented by an integer value based on their order.

Both scaling and encoding are important pre-processing steps in machine learning, as they can improve the performance of algorithms and make the data more suitable for analysis.

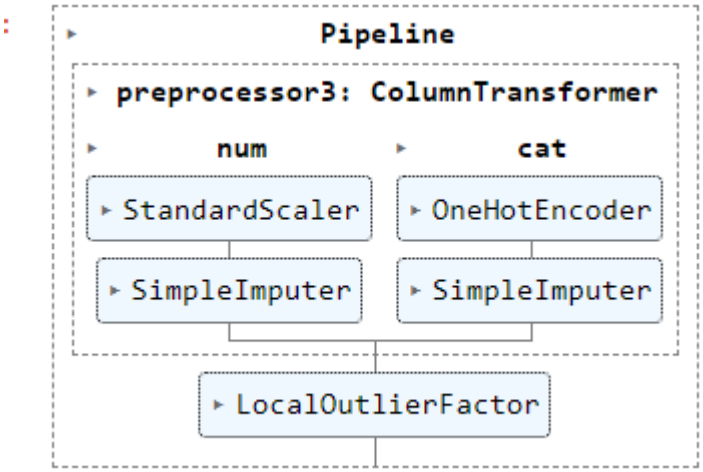
In our project we use StandardScaler, RobustScaler for scaling numerical data and OneHotEncoder for categorical data.

```
In [32]: # Build ML pipeline for Local Outlier Factor model so as to identify outliers
numeric_transformer = Pipeline(steps=[
                                       ('normal', (StandardScaler())),
                                       ('imputer', SimpleImputer(strategy='median'))
                           ])
categorical_transformer = Pipeline(steps=[
                                        ('encoder', OneHotEncoder(handle_unknown='ignore', sparse=True)),
    ('imputer2', SimpleImputer(strategy='most_frequent'))
                           ])
```

Outlier Detection:

Outlier detection is an important technique in machine learning, as outliers can have a significant impact on the accuracy and performance of models. Outliers can skew the distribution of data, leading to overfitting or underfitting of models.

In our project we use LOF outlier algorithm. Local Outlier Factor (LOF) is a density-based outlier detection method. It measures the local deviation of the density of a data point with respect to its neighbours. The LOF algorithm is a popular unsupervised learning technique used to identify anomalies and outliers in a dataset. We put outlier in Pipeline.



We got 14 outliers from the dataset which is very less comparing the dataset. So, we don't need to remove outlier from dataset. It won't effect on accuracy.

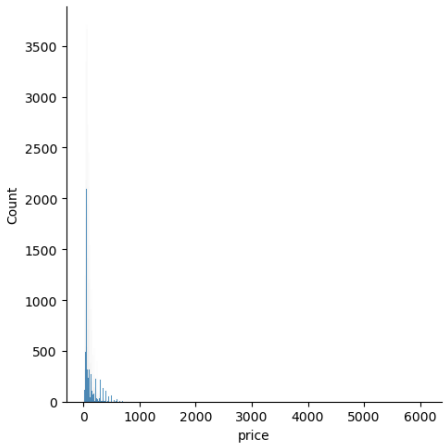
```
outliers_df['outliers'].value_counts()
```

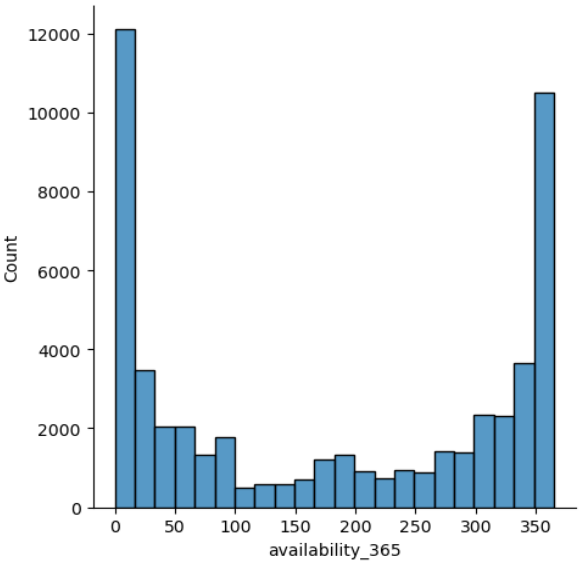
```
 1     52711
-1        14
Name: outliers, dtype: int64
```

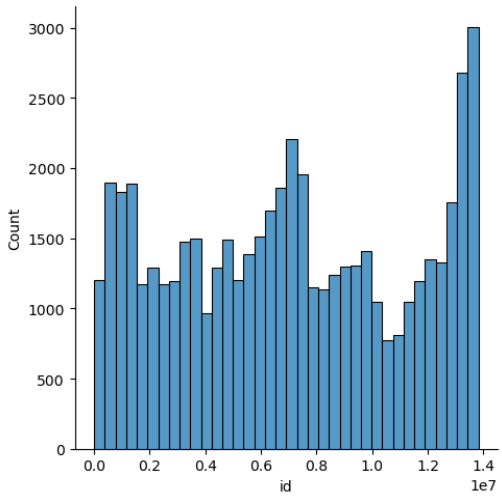
Notice : This dataframe contains 14 outliers

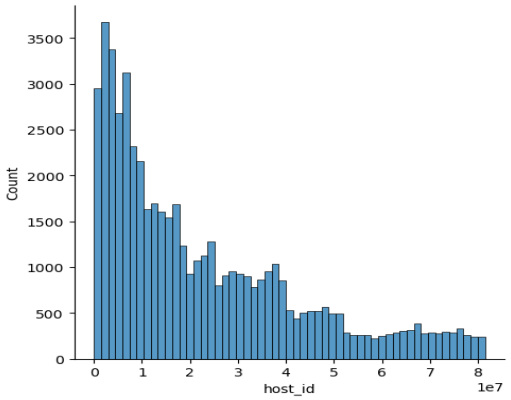
Data visualization:

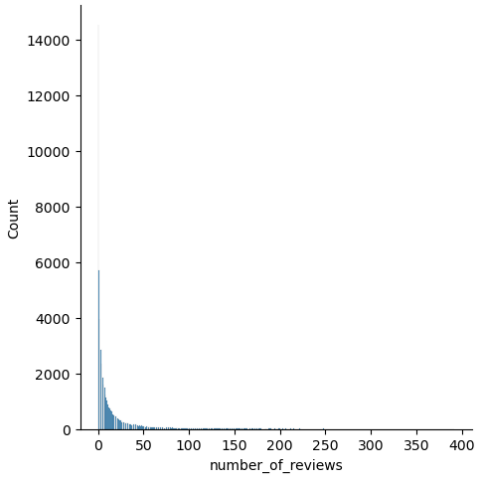
Here is some histograms representation of quantitative variables with respect to count.

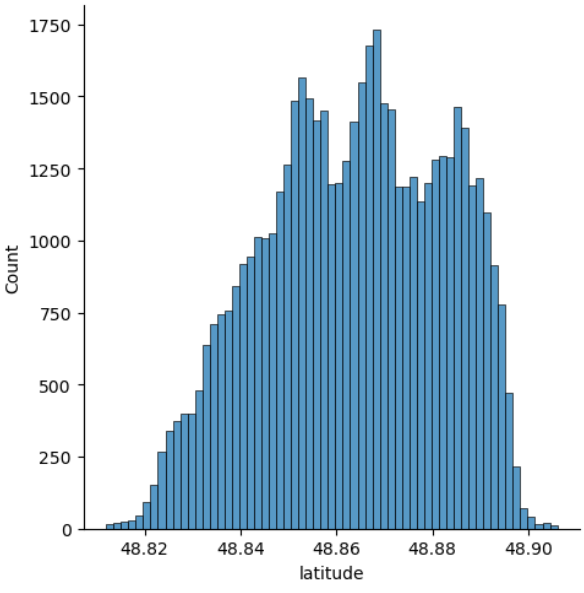


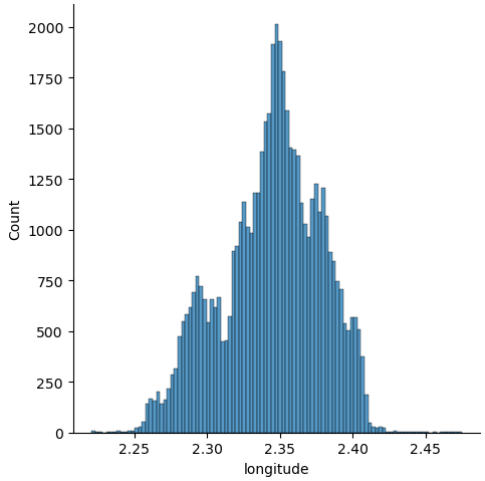












Number of apartments per owner. Here host name is owner name and name is the apartment name.

In [54]:
```
new_df.head()
```

Out[54]:

	host_name	name
0	Matthieu	Appartement 60m2 Rue Legendre 75017
1	Claire	Appart au pied de l'arc de triomphe
2	Vincent	Nice appartment in Batignolles
3	Julie	Charming flat near Batignolles
4	Daniele	Spacious bedroom near the centre of Paris

[57]:
```
df.count()
```

:[57]:

host_name	name
(EMAIL HIDDEN)	23
(URL HIDDEN)A	3
(Url Hidden)HIDDENHIDDEN	1
1 Rue Affre	1
2691lilou	1
...	...
수연	2
수일	2
예빈(Ye-Bin)	1
이효은	1
정인	1

Below table shows renting price per city quarter. There are three types of price mentioned min= minimum, max= maximum and mean= average price.

```
]: df2 = new_df2.groupby([ 'neighbourhood' ]).agg([ 'min' , 'max' , 'mean' ])

]: df2

]:
```

	price		
	min	max	mean
neighbourhood			
Batignolles-Monceau	17	2004	88.625590
Bourse	21	6081	118.325639
Buttes-Chaumont	10	1002	66.539683
Buttes-Montmartre	14	3306	74.784066
Entrepôt	10	451	81.292556
Gobelins	12	1129	74.245876
Hôtel-de-Ville	19	3608	135.476923
Louvre	12	852	138.712094
Luxembourg	20	1874	145.288035
Ménilmontant	13	1352	66.239832
Observatoire	0	2004	84.961179
Opéra	15	1253	99.646253

Correlation between variables:

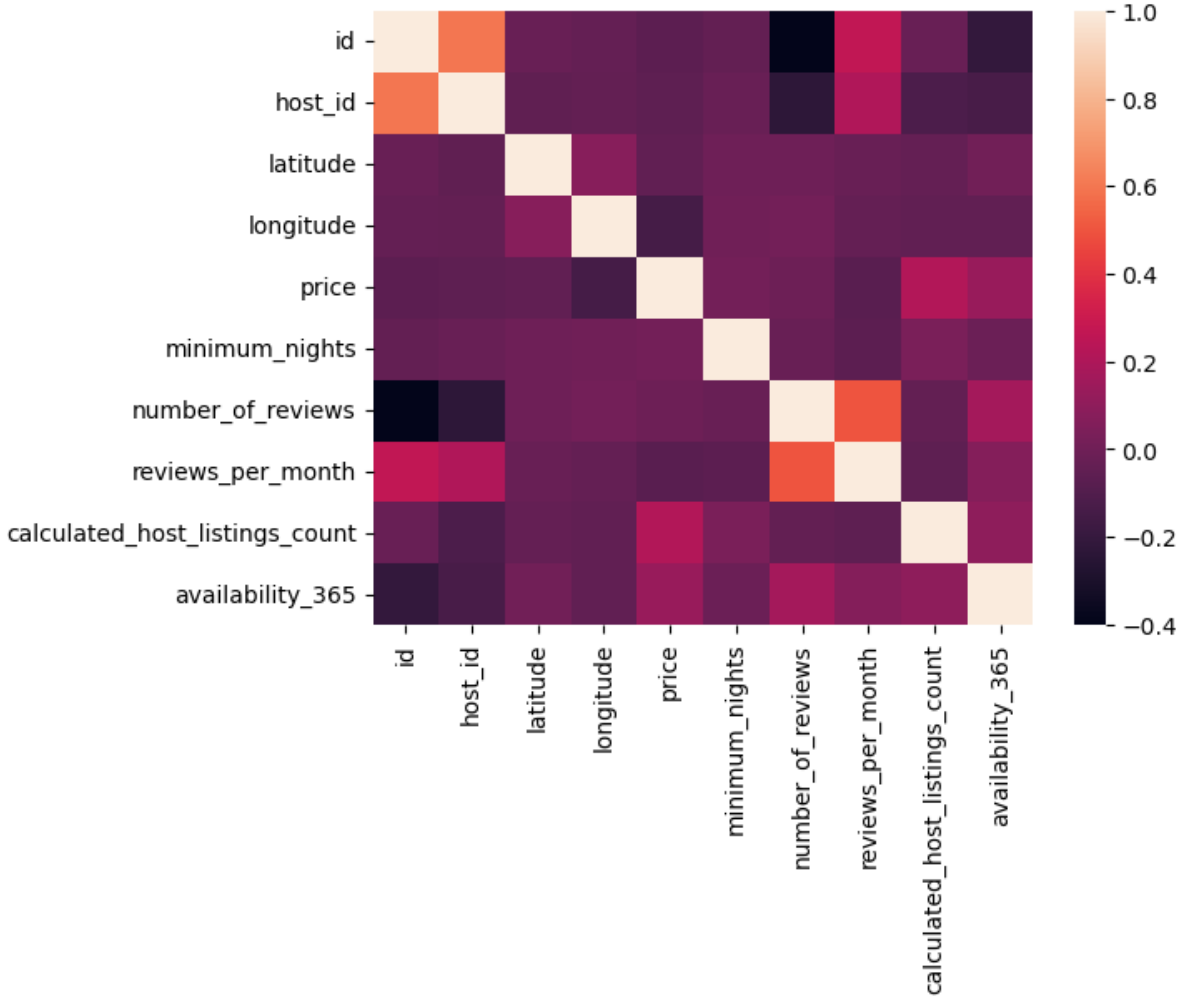
Correlation refers to the statistical relationship between two or more variables. Specifically, correlation measures how much one variable change when another variable changes.

Correlation is commonly used in machine learning to understand the relationship between input features and the target variable. For example, if we have a dataset with multiple features and a target variable, we may want to calculate the correlation between each feature and the target variable to determine which features are most important for predicting the target variable.

In [45]:
```
# Pairwise correlation between quantitative variables
listings.corr()
```

Out[45]:

	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_li
id	1.000000	0.596453	-0.028063	-0.037022	-0.065670	-0.044603	-0.401893	0.263105	
host_id	0.596453	1.000000	-0.051608	-0.045682	-0.061263	-0.027518	-0.233452	0.211031	
latitude	-0.028063	-0.051608	1.000000	0.078139	-0.048125	-0.004173	-0.005561	-0.029362	
longitude	-0.037022	-0.045682	0.078139	1.000000	-0.147700	-0.002011	0.010360	-0.035627	
price	-0.065670	-0.061263	-0.048125	-0.147700	1.000000	0.009766	-0.010473	-0.077854	
minimum_nights	-0.044603	-0.027518	-0.004173	-0.002011	0.009766	1.000000	-0.027143	-0.064564	
number_of_reviews	-0.401893	-0.233452	-0.005561	0.010360	-0.010473	-0.027143	1.000000	0.497596	
reviews_per_month	0.263105	0.211031	-0.029362	-0.035627	-0.077854	-0.064564	0.497596	1.000000	
calculated_host_listings_count	-0.026747	-0.122364	-0.035810	-0.047441	0.214818	0.030978	-0.045405	-0.060358	
availability_365	-0.215381	-0.135731	0.007411	-0.048662	0.130332	-0.014827	0.169609	0.065999	



Modelling:

In this project we use 4 different models to predict price. For train the model first we have to split the dataset and use a ML pipeline.

2. Modeling

In [61]:
```
# listings1 DataFrame does not contains last_review variable which is a date
```

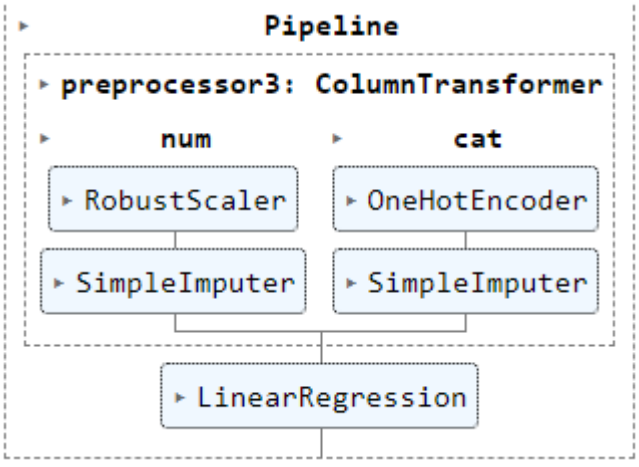
In [62]:
```
# Split the data intro train and test set
X = listings1.drop('price', axis =1)
y= listings1['price']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
 X, y, test_size=0.20, random_state=2022)
```

Building pipeline for each model

]:
```
numeric_transformer = Pipeline(steps=[

                                    ('normal', (RobustScaler())),

                                    ('imputer', SimpleImputer(strategy='mean'))

                        ])

categorical_transformer = Pipeline(steps=[

                                     ('encoder', OneHotEncoder(handle_unknown='ignore', sparse=True)),
    ('imputer2', SimpleImputer(strategy='most_frequent'))

                        ])


preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, num_features),
        ('cat', categorical_transformer, cat_features)])
```

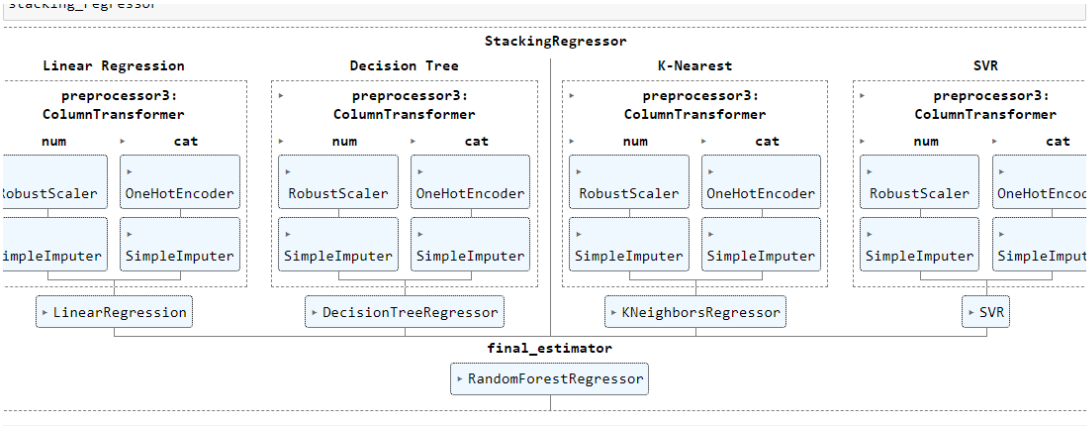
Pipeline for liner regression

```
LR_pipeline = Pipeline(steps=[
                                ('preprocessor3', preprocessor),
    ('predictor1', LinearRegression())

                                ])
LR_pipeline
```



Pipeline for stacking regression.

Out[78]:



Model performance

```
# evaluate model performance
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print('Mean Squared Error:', mse)
print('Root Mean Squared Error:', rmse)
print('Mean Absolute Error:', mae)
print('R-squared:', r2)
```

```
Mean Squared Error: 6148.291424997629
Root Mean Squared Error: 78.41104147374672
Mean Absolute Error: 40.08979231863442
R-squared: 0.16071541730480965
```

Conclusion:

The objective of the project is to perform data analysis techniques to understand the insight of the data. This project aims to apply Exploratory Data Analysis (EDA) and Machine Learning models to understanding the data.