AirBNB Listings - USA

PROGRAMMING FOR DATA SCIENCE (BUAN 6349.5U1)

GROUP NUMBER: 13

- **LIKITH VINAYAKA GIRIDHAR (LXG210034)**
- **MUAAZ ALI SYED (MXS220139)**

Airbnb Listings - USA

Abstract

- As an Airbnb host, when you manage multiple listings in various cities across the USA. The goal is to optimize the performance of these listings to attract more guests and achieve higher occupancy rates.
- The number of reviews a listing receives is a crucial indicator of its popularity and guest satisfaction. Positive reviews can attract more potential guests and improve the overall reputation of your listings.
- With data points including neighbhorhood, room_type, price, number_of_reviews, reviews_per_month as well as details on number of reviews in the past 12 months.
- To achieve this, we build a machine learning model that can predict the number of reviews a listing has received in the last 12 months.

Data Source and Description

We have taken the data source from Kaggle (https://www.kaggle.com/datasets/tamle507/airbnb-listings-usa). The data set booking information for reviews hosts have received for their listings on Airbnb with various attributes such as neighbourhood_group, neighbourhood, room_type, minimum_nights, price, reviews_per_month, last_review, number_of_reviews_ltm etc

Field	Description
id	Airbnb's unique identifier for the listing
name	Name of the listing
host_id	Airbnb's unique identifier for the host/user
host_name	Name of the host. Usually just the first name(s)
neighbourhood_group	The neighbourhood group as geocoded using the latitude and longitude against neighborhoods as defined by open or public digital shapefiles
neighbourhood	Location of the listing
latitude	Latitude of the listing (uses World Geodetic System (WGS84) projection)
longitude	Longitude of the listing (uses World Geodetic System (WGS84) projection)
room_type	Room type of the listing: 'Entire home/apt', 'Private room', 'Shared room', 'Hotel'
price	Daily price of the listing in the local currency
minimum_nights	Minimum number of night stay of the listing (calendar rules may differ)
number_of_reviews	The total number of reviews of the listing
last_review	The date of the last (newest) review of the listing
reviews_per_month	The total number of reviews per month of the listing
calculated_host_listings_count	The total number of listings the host has in the city/region area
availability_365	The availability of the listing 365 days in advance (determined by the calendar); please note that a listing may not be available because it has either been booked by a guest or blocked by the host.
number_of_reviews_ltm	The total number of reviews the listing has in the last 12 months
license	The license/permit/registration number
state	State of the listing abbreviated (added in)
city	City/County of the listing (added in)

Goal of the Project:

The goal of the project is to predict number of reviews received by a listing in the last 12 months. This can help the host to manage multiple listings in various cities across the USA. The goal is to optimize the performance of these listings to attract more guests and achieve higher occupancy rates using Regression that we will perform. We hoped that the analysis with the data exploration and model building will answer the following questions:

Data Exploration – We will use summary statistics and visualization tools to examine the following questions:

- Which listings have the highest/lowest number of reviews in the last 12 months? How much do guests pay for a room per night?
- Which hosts on average have listings with the highest/lowest number of reviews in the last 12 months?
- Relationship between Number of listings and neighbourhood group

- Does a listing's price correlate with its number of reviews in the last 12 months?
- Which listing room types on median have the highest/lowest number of reviews in the last 12 months?
- Do listings with indicated licenses on median differ from listings without indicated licenses in their numbers of reviews in the last 12 months?
- Distribution of room types
- Average price by room type
- Reviews per month by neighbourhood group
- Average availability by room type
- Average minimum nights by room type
- Relationship between two variables: "Total Number of Reviews" and "Number of Reviews in Last 12 Months"

Prediction Model – We will apply different Regression models to predict the number of reviews in the last 12 months

1. Data Understanding and Preparation: In this context, we can perform data cleaning, exploring, normalization, removing/fixing missing values, feature engineering, etc.
2. Modelling: We will utilize regression algorithms to predict the number of reviews in the last 12 months by utilizing significant attributes extracted through data exploration.
3. Validation: Validate the model performance on the test data, and generate performance metrics.
4. Conclusion: What insights can we gain from the prediction model? And how can we use the model to assist hosts?

Understanding our data:

- The target variable number_of_reviews_ltm which represents the number of reviews a host has received in the last 12 months
- The remaining variables provide additional details about the hosts, including neighbourhood_group, neighbourhood, latitude, longitude, room_type, price, minimum_nights, number_of_reviews, last_review, reviews_per_month, calculated_host_listings_count, availability_365, number_of_reviews_ltm, license, state, city and more.
- Some of the variables have numerical values, while others are categorical with various levels of categories.
- The dataset layout provides a tabular format with rows representing individual hosts and columns representing attributes of the hosts.
- The dataset could be useful for analyzing patterns in hotel bookings and predicting future bookings, as well as identifying factors that contribute to cancellations.

Exploring the Dataset:

1) Which listings have the highest/lowest number of reviews in the last 12 months?

Subset the data to include only consider mature listings which will be defined as having number_of_reviews in total >= 10

1. Mature listings with the highest number of reviews in the past year tend to be situated in popular tourist areas like Broadway Show, and they are often managed by commercial entities such as Sahara Las Vegas. These listings commonly offer 'Private room' accommodations and have shorter minimum stay requirements.

2. On the other hand, mature listings with the lowest number of reviews in the last 12 months are typically found in residential neighborhoods like Beverly Hills Apartment. They are usually hosted by

individuals, like David, and predominantly offer 'Entire home/apt' rentals. Additionally, these listings have longer minimum stay requirements.

2) Which hosts on average have listings with the highest/lowest number of reviews in the last 12 months?

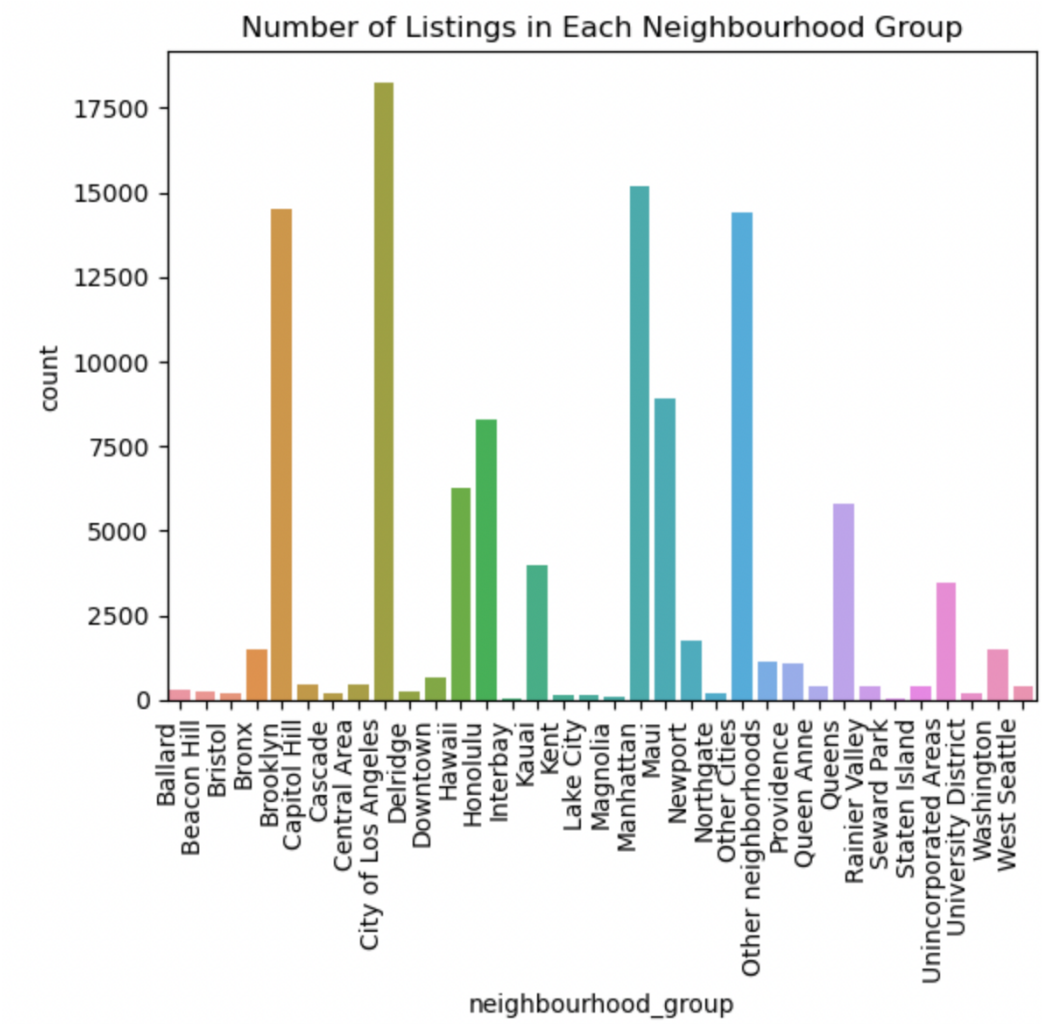
Let's only consider mature listings which will be defined as having number_of_reviews in total >= 10.On an average a guest pays around 80-120 Euros per night.

1. On average, hosts with mature listings that receive the highest number of reviews in the last 12 months are primarily commercial entities, like Hotel Pepper Tree. In contrast, hosts with mature listings receiving the lowest number of reviews are predominantly individuals, such as Jason.

2. The significant contrast in the minimum nights required between these two host types supports the observations mentioned in the first insight.

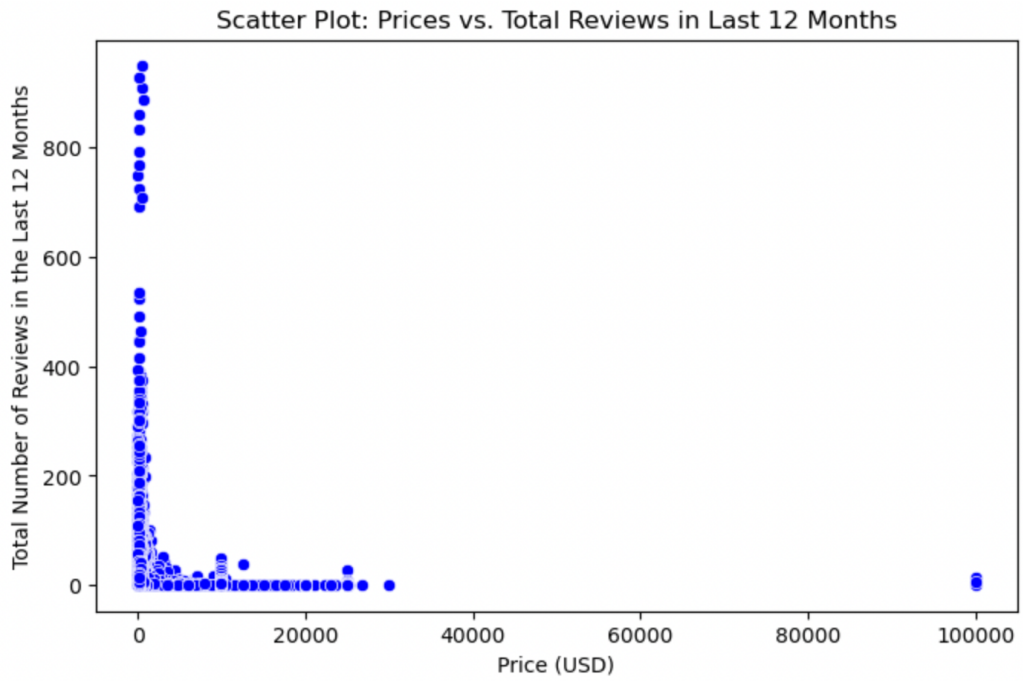
3) Relationship between Number of listings and neighbourhood group

We can see most of the listings are in the neighbhohood of major cities – Brooklyn, City of Los Angeles, Hawaii, Honolulu, Manhattan, Maui, Queens, Washington etc



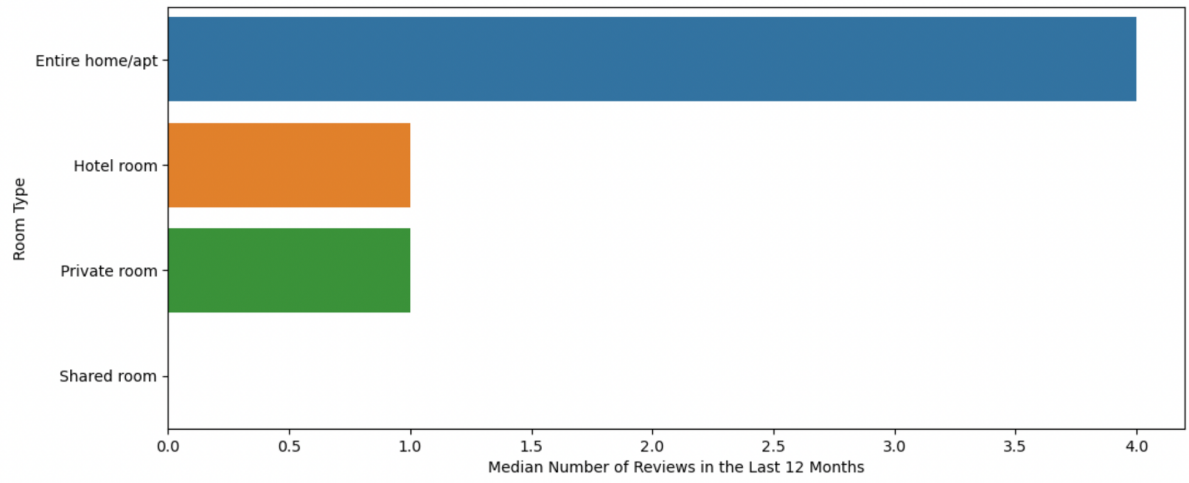
4) Does a listing's price correlate with its number of reviews in the last 12 months?

Based on the plots we can see that the listings with higher prices have lower number of reviews in the last 12 months.



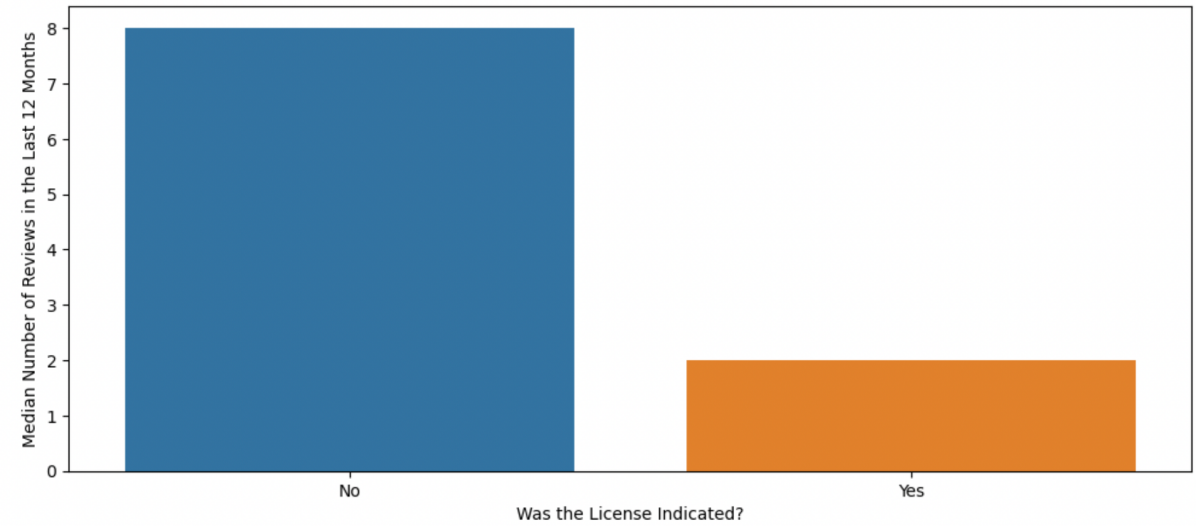
5) Which listing room types on median have the highest/lowest number of reviews in the last 12 months?

Based on the bar chart, on median, 'Entire home/apt' room types have the highest number of reviews in the last 12 months while 'Shared room' room types have the lowest number of reviews in the last 12 months.



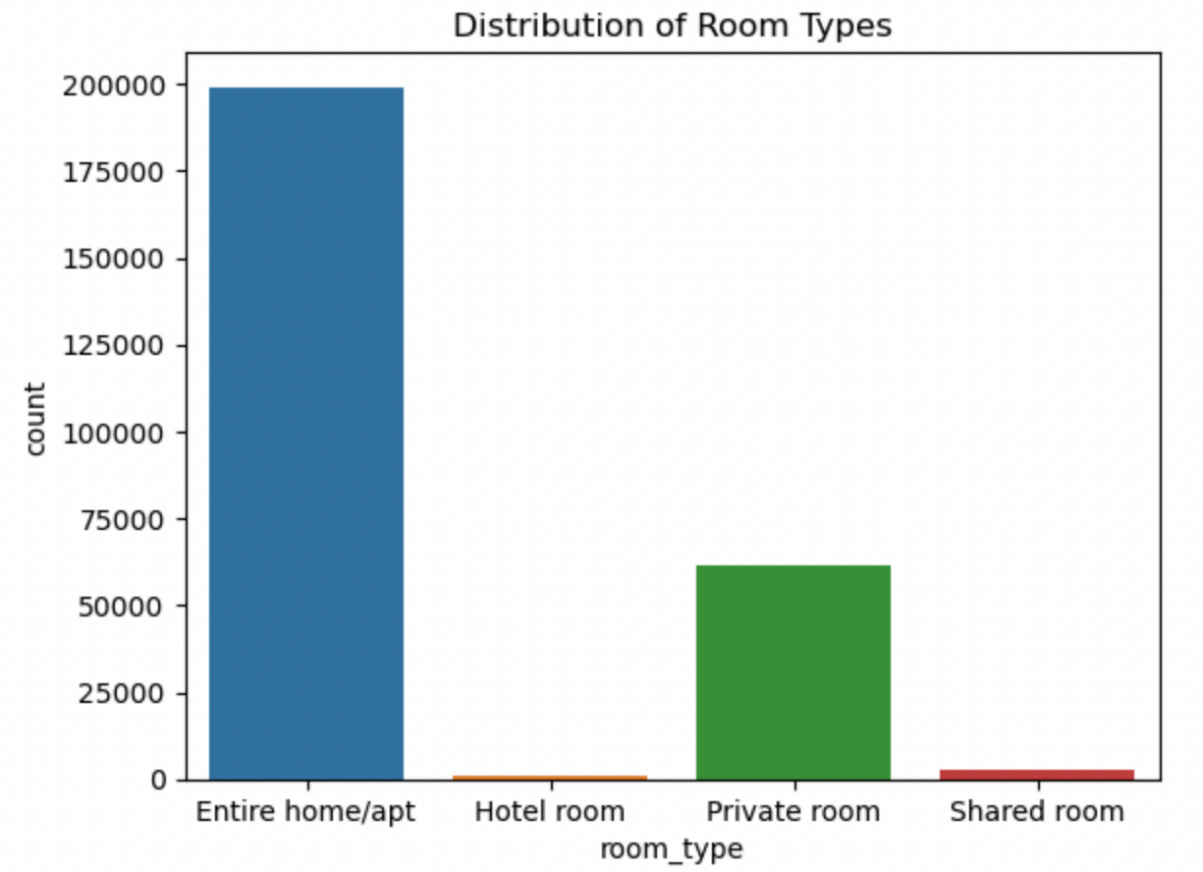
6) Do listings with indicated licenses on median differ from listings without indicated licenses in their numbers of reviews in the last 12 months?

We can notice that the listings without indicated licenses have higher number of reviews in the last 12 months than listings with indicated licenses.



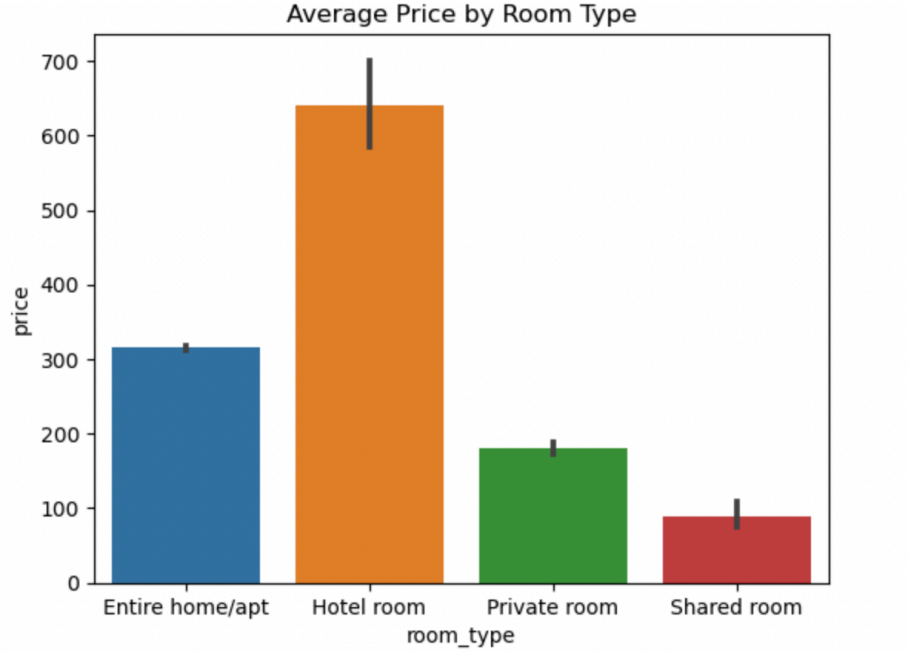
7) Distribution of room types

We can see majority of the listings are Entire homes or apartments followed by Private room meanwhile Hotel rooms and shared rooms are the least listed accomodations on Airbnb



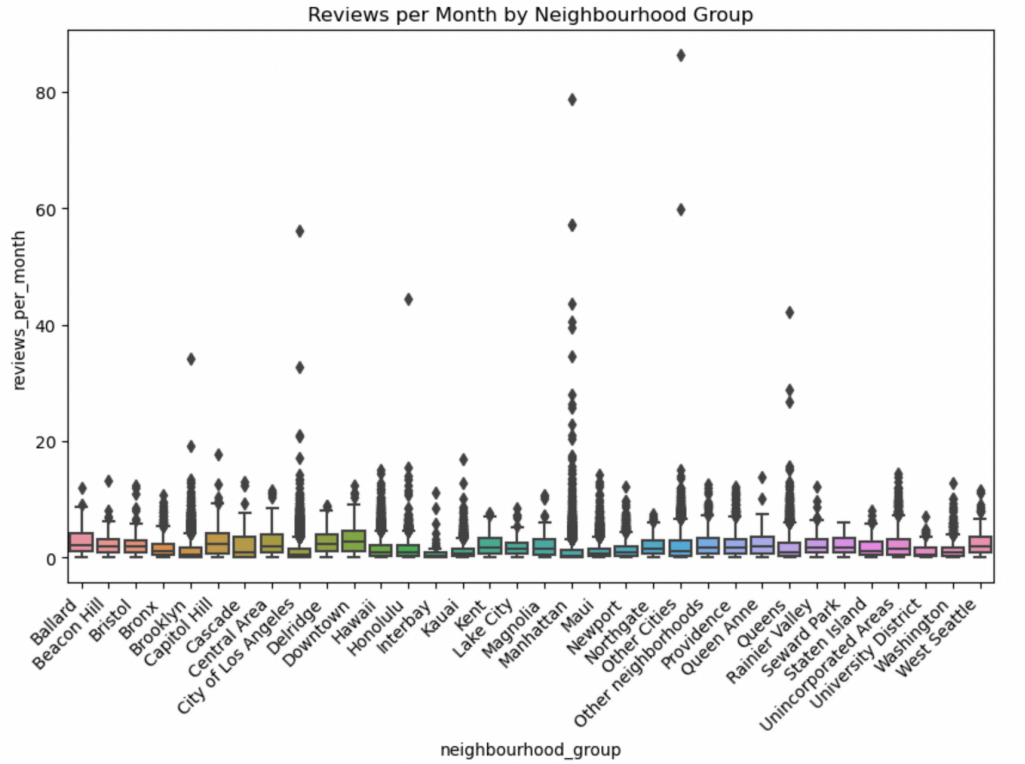
8) Average price by room type

The average price for Hotel rooms is the highest followed by Entire homes and apartments while the price is least for shared rooms



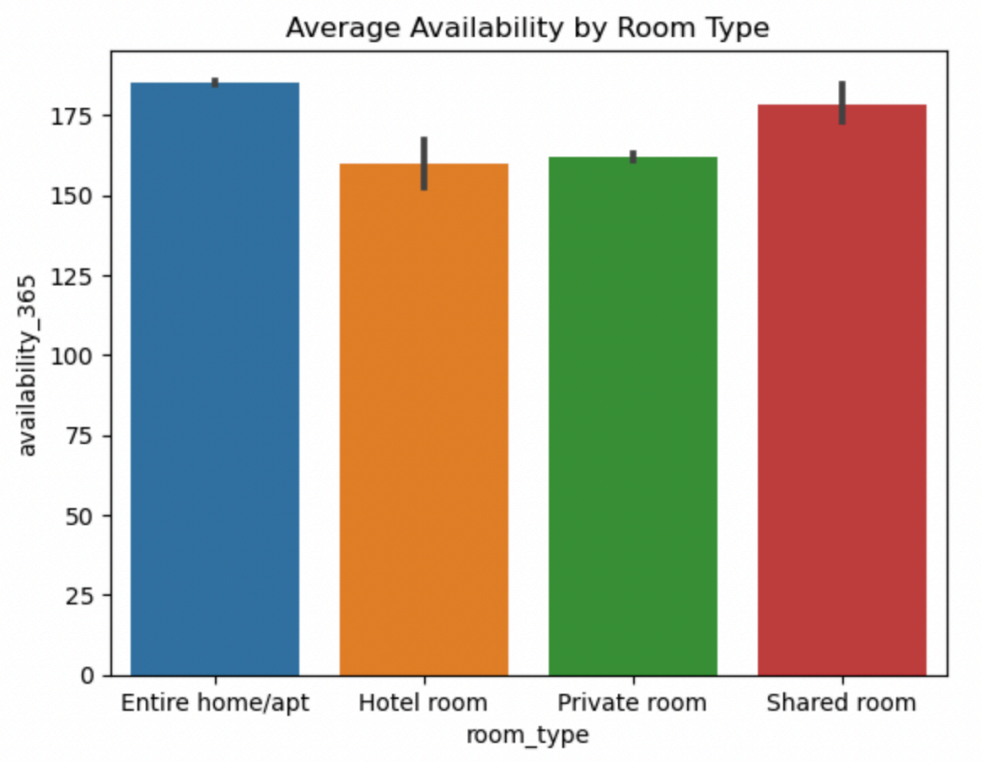
9) Reviews per month by neighbourhood group

The mean number of reviews per month remains the same for most of the neighbourhoods. But the neighbhohood of major cities – Brooklyn, City of Los Angeles, Hawaii, Honolulu, Manhattan, Maui, Queens, Washington etc have an extremely high number of outliers.



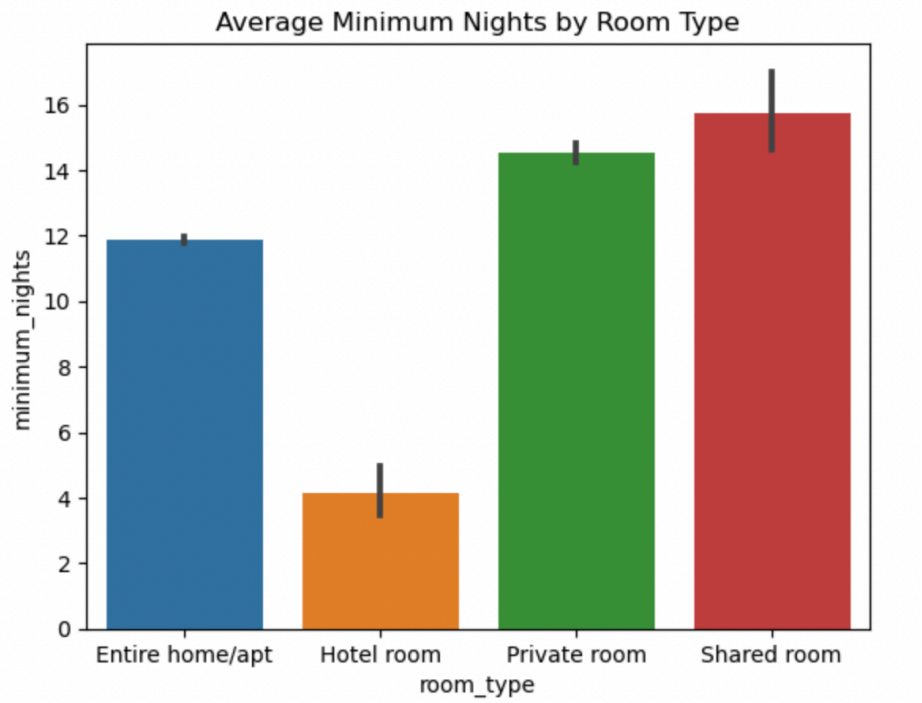
10) Average availability by room type

It is interesting to note that the average availability of all the room types 365 days in advance are the same



11) Average minimum nights by room type

It is interesting to note that average minimum nights by room type is highest for shared room followed by private room with hotel rooms having the lowest value

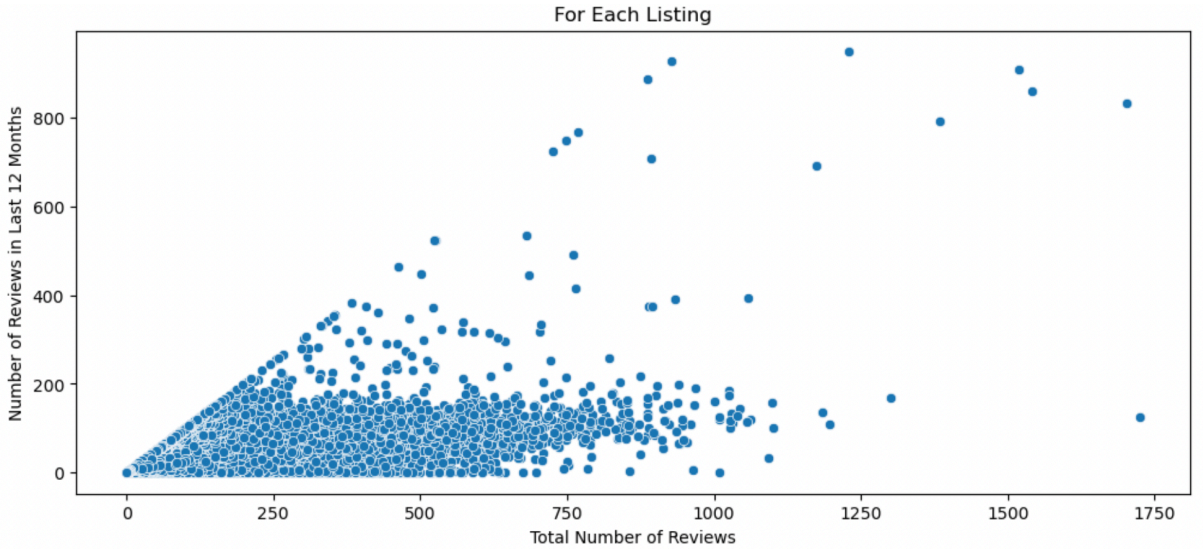


12) Relationship between two variables: "Total Number of Reviews" and "Number of Reviews in Last 12 Months"

The scatter plot helps us visualize the relationship between the total number of reviews a listing has received and the number of reviews it received in the last 12 months.

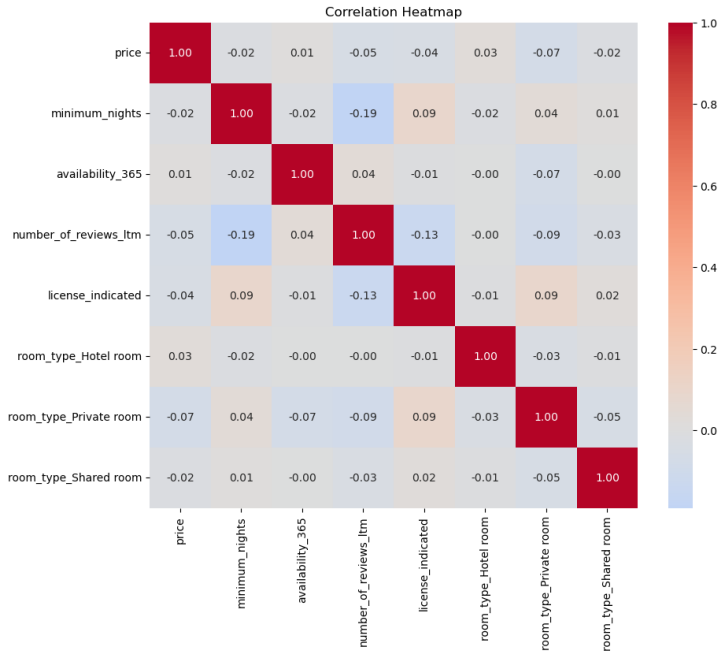
The points that are distributed in a linear or non-linear manner, it may suggest a correlation or relationship between the two variables.

The points are concentrated around a certain region, it may indicate a common behavior among listings in terms of review patterns.



13) Heatmap Plot of the Correlation Matrix

No two attributes have high correlation as observed from the heatmap



Data Cleaning and preparation for model building:

Dropping missing values, feature extraction and feature engineering:

- Drop neighbourhood_group since we have city
- Create license_indicated whether its 1 if a listing indicates a license or 0 if a listing doesn't indicate a license; although license is mostly null, it does correlate with number_of_reviews_ltm
- Drop last_review and reviews_per_month which should allow our model to generalize to new listings
- Finally, let's drop all null observations since they'll account for less than 0.3% of the total.
- Let's drop id and host_id since they're just generic identifiers
- Let's drop host_name since we have name which contains more information on the listing
- Let's drop latitude and longitude since they're just coordinates
- Dropping number_of_reviews should allow us to generalize to new listings
- Dropping neighbourhood, state and city should allow us to generalize to new USA neighbourhood, states and cities
- Dropping calculated_host_listings_count should allow us to generalize to new hosts
- Let's keep name in case we need to use text classification instead.
- Let's create (and drop the input fields to avoid multicollinearity): Dummy variables for room_type with 'Entire home/apt' as the default case
- To get a clearer picture, let's drop all observations with price = 0.

Regression Model Performance Metrics:

In [67]:

```
# Create DataFrame with the R2 values and 95% confidence intervals
r2_df = pd.DataFrame(data=reg_dic.values(), index=reg_dic.keys()) \
            .apply(['mean', 'std'], axis='columns') \
            .apply([lambda row: row['mean'] - 2*row['std'],
                    lambda row: row['mean'] + 2*row['std']], axis='columns')

# Assign column names for R2 values and 95% confidence intervals
r2_df.columns = ['R2_95CI_Lower', 'R2_95CI_Upper']

# Sort the DataFrame in descending order based on R2_95CI_Lower values
r2_df_sorted = r2_df.sort_values(['R2_95CI_Lower', 'R2_95CI_Upper'], ascending=False)

# Print the R2 values with their 95% confidence intervals
for model, row in r2_df_sorted.iterrows():
    print(f"Model: {model}")
    print(f"R2_95CI_Lower: {row['R2_95CI_Lower']:.4f}")
    print(f"R2_95CI_Upper: {row['R2_95CI_Upper']:.4f}\n")

```

```
Model: Random Forest Regressor
R2_95CI_Lower: 0.0961
R2_95CI_Upper: 0.1256

Model: K Neighbors Regressor
R2_95CI_Lower: 0.0892
R2_95CI_Upper: 0.1252

Model: Ridge Regression
R2_95CI_Lower: 0.0507
R2_95CI_Upper: 0.0648

Model: Linear Regression
R2_95CI_Lower: 0.0507
R2_95CI_Upper: 0.0649

Model: Lasso Regression
R2_95CI_Lower: 0.0316
R2_95CI_Upper: 0.0417
```

In [69]:
```python
from sklearn.metrics import mean_squared_error, mean_absolute_error

# Dictionary to store performance metrics (MSE and MAE) for each model
performance_metrics = {}

# Calculate performance metrics (MSE and MAE) for each model
for model_name, model in regression_models.items():
    y_pred = model.predict(X_test_scaled.drop('name', axis='columns'))
    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)

    performance_metrics[model_name] = {'Mean Squared Error (MSE)': mse, 'Mean Absolute Error (MAE)': mae}

# Print the performance metrics for each model
for model_name, metrics in performance_metrics.items():
    print(f"Model: {model_name}")
    print(f"Mean Squared Error (MSE): {metrics['Mean Squared Error (MSE)']:.4f}")
    print(f"Mean Absolute Error (MAE): {metrics['Mean Absolute Error (MAE)']:.4f}\n")

```

```
Model: Linear Regression
Mean Squared Error (MSE): 367.8673
Mean Absolute Error (MAE): 13.0653

Model: Ridge Regression
Mean Squared Error (MSE): 367.8673
Mean Absolute Error (MAE): 13.0653

Model: Lasso Regression
Mean Squared Error (MSE): 377.6506
Mean Absolute Error (MAE): 13.4398

Model: K Neighbors Regressor
Mean Squared Error (MSE): 350.9676
Mean Absolute Error (MAE): 11.1235

Model: Random Forest Regressor
Mean Squared Error (MSE): 354.4606
Mean Absolute Error (MAE): 10.8191
```

Hyperparameter Tuning using GridSearchCV to check for best parameters and R2 Score:

Hyperparameter tuning using GridSearchCV

In [61]:
```python
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import GridSearchCV

linear_regression = LinearRegression()

# Define parameter grid for GridSearchCV
param_grid = {}

# Perform GridSearchCV
grid_search_linear = GridSearchCV(linear_regression, param_grid, cv=5)
grid_search_linear.fit(X_train_scaled.drop('name', axis='columns'), y_train)

# Get best parameters and best score
print("Best Parameters:", grid_search_linear.best_params_)
print("Best R2 Score:", grid_search_linear.best_score_)

```

```
Best Parameters: {}
Best R2 Score: 0.05777703678941008
```

In [62]:
```python
from sklearn.linear_model import Ridge

ridge_regression = Ridge()

# Define parameter grid for GridSearchCV
param_grid = {
    'alpha': [0.1, 1, 10, 100]
}

# Perform GridSearchCV
grid_search_ridge = GridSearchCV(ridge_regression, param_grid, cv=5)
grid_search_ridge.fit(X_train_scaled.drop('name', axis='columns'), y_train)

# Get best parameters and best score
print("Best Parameters:", grid_search_ridge.best_params_)
print("Best R2 Score:", grid_search_ridge.best_score_)

```

```
Best Parameters: {'alpha': 10}
Best R2 Score: 0.05777755538711497
```

In [63]:
```python
from sklearn.linear_model import Lasso

lasso_regression = Lasso()

# Define parameter grid for GridSearchCV
param_grid = {
    'alpha': [0.1, 1, 10, 100]
}

# Perform GridSearchCV
grid_search_lasso = GridSearchCV(lasso_regression, param_grid, cv=5)
grid_search_lasso.fit(X_train_scaled.drop('name', axis='columns'), y_train)

# Get best parameters and best score
print("Best Parameters:", grid_search_lasso.best_params_)
print("Best R2 Score:", grid_search_lasso.best_score_)

```

```
Best Parameters: {'alpha': 0.1}
Best R2 Score: 0.05654805693174181
```

In [64]:
```python
from sklearn.neighbors import KNeighborsRegressor

k_neighbors_regressor = KNeighborsRegressor()

# Define parameter grid for GridSearchCV
param_grid = {
    'n_neighbors': [3, 5, 7, 9],
    'weights': ['uniform', 'distance']
}

# Perform GridSearchCV
grid_search_knn = GridSearchCV(k_neighbors_regressor, param_grid, cv=5)
grid_search_knn.fit(X_train_scaled.drop('name', axis='columns'), y_train)

# Get best parameters and best score
print("Best Parameters:", grid_search_knn.best_params_)
print("Best R2 Score:", grid_search_knn.best_score_)

```

```
Best Parameters: {'n_neighbors': 9, 'weights': 'uniform'}
Best R2 Score: 0.16146390577720177
```

In [65]:
```python
from sklearn.ensemble import RandomForestRegressor

random_forest_regressor = RandomForestRegressor()

# Define parameter grid for GridSearchCV
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10]
}

# Perform GridSearchCV
grid_search_rf = GridSearchCV(random_forest_regressor, param_grid, cv=5)
grid_search_rf.fit(X_train_scaled.drop('name', axis='columns'), y_train)

# Get best parameters and best score
print("Best Parameters:", grid_search_rf.best_params_)
print("Best R2 Score:", grid_search_rf.best_score_)

```

```
Best Parameters: {'max_depth': 10, 'min_samples_split': 10, 'n_estimators': 200}
Best R2 Score: 0.24156757224429987
```

Based on the provided R2 values, R2 95% confidence intervals, Mean Squared Error (MSE), and Mean Absolute Error (MAE) for each model, we can infer the following:

1. **Random Forest Regressor:**
    - R2 (Coefficient of Determination) is approximately 0.1132, with a 95% confidence interval ranging from 0.1039 to 0.1225.
    - MSE is approximately 354.4606.
    - MAE is approximately 10.8191.
    - After performing GridSearchCV we get Best Parameters: {'max_depth': 10, 'min_samples_split': 10, 'n_estimators': 200} Best R2 Score: 0.24141915019094543
2. **K Neighbors Regressor:**
    - R2 is approximately 0.1072, with a 95% confidence interval ranging from 0.0892 to 0.1252.
    - MSE is approximately 350.9676.
    - MAE is approximately 11.1235.
    - After performing GridSearchCV we get the Best Parameters: {'n_neighbors': 9, 'weights': 'uniform'} Best R2 Score: 0.16146390577720177

3. **Ridge Regression:**
    - R2 is approximately 0.0578, with a 95% confidence interval ranging from 0.0507 to 0.0648.
    - MSE is approximately 367.8673.
    - MAE is approximately 13.0653.
    - After performing GridSearchCV we get Best Parameters: {'alpha': 10} Best R2 Score: 0.05777755538711497
4. **Linear Regression:**
    - R2 is approximately 0.0578, with a 95% confidence interval ranging from 0.0507 to 0.0649.
    - MSE is approximately 367.8673.
    - MAE is approximately 13.0653.
    - After performing GridSearchCV we get Best R2 score to be: 0.05777703678941008
5. **Lasso Regression:**
    - R2 is approximately 0.0367, with a 95% confidence interval ranging from 0.0316 to 0.0417.
    - MSE is approximately 377.6506.
    - MAE is approximately 13.4398.
    - After performing GridhSearchCV we obtain Best Parameters: {'alpha': 0.1} Best Best R2 Score: 0.05654805693174181

Inference:

Based on the performance metrics and R2 values, the two most suitable models appear to be the **Random Forest Regressor** and the **K Neighbors Regressor**. These models have the highest R2 values, indicating better overall fit to the data. Moreover, they both exhibit lower MSE and MAE values, which suggest better predictive accuracy compared to the other regression models.

Between these two models, the **Random Forest Regressor** has slightly better performance metrics (lower MSE and MAE) compared to the **K Neighbors Regressor**. Therefore, the **Random Forest Regressor** might be the preferred choice among the tested models for predicting the number of reviews in the last 12 months in the given dataset.

The above selection of Random Forest Regressor has further been solidified by concurrence from hyperparameter tuning using GridSearchCV.

Scope for future work:

- We have done Hyper Parameter Tuning using GridSearchCV and next to improve model performance we can try RandomizedSearchCV
- Try to extrapolate this model to predict the reviews for the next 12 months
- Try converting the problem to a classification problem and observe the performance of classification models