Capstone Project

US-Airbnb Open Data

Airbnb listings (Price Prediction) 2021

airbnb
Yourself Be everyone else is taken
Think and Wonder, Wonder and Think

Mentored By: - **Ms. Anjana Agrawal**

Submitted By: **Pulkit Kohli**
Pankaj Nagar
Sunil Nagar
Meghansh Satyajit
Devendra Singh
Yashvir Verma

ABSTRACT:

Airbnb’s unique approach towards lodging is part of the “sharing economy. One can find unique home-stays with customized experiences across the world instead of the usual way of booking hotels. From places to crash on your backpacking trip through Europe, to a place to stay for a month during your internship in Los Angeles, Airbnb offers a multitude of options for a traveler.

The aim of our preliminary analysis and data exploration is to evaluate the actions taken by the visitors and the hosts based on their environment and predicting the visitor’s staying price. Hoping to find insights such as which hosts are the busiest and why the challenge is to track Airbnb’s price index, rate fluctuations and accommodation pricing trends based on location popularity, rating, number of rooms and other attributes.

Keywords: Airbnb, Lodging, Visitors, Machine Learning, Predictions, Location Popularity, Ratings, business, internet, regression analysis, world, hotels and accommodations

Acknowledgements

At the outset, we are indebted to our mentor Ms. Anjana Agrawal for her time, valuable inputs and guidance. Her experience, support and structured thought process guided us to be on the right track towards completion of this project.

We also thank all the course faculty of the DSE program for providing us with a strong foundation in various concepts of analytics & machine learning.

Last but not the least, we would like to sincerely thank our respective families for giving us the necessary support, space and time to complete this project.

We certify that the work done by us for conceptualizing and completing this project is original and authentic.

Table Of Contents

Chapter 1: Project Overview

Background & Need for Study

Airbnb, Inc. is an online marketplace for seeking and/or offering lodging, primarily homestays and tourism experiences. It records price shifts from over 650K hosts operating in more than 81K cities and offers alternative accommodations around the world in its Hotel Price Index. The challenge is to track rate fluctuations and accommodation pricing trends based on location popularity, rating, no. of rooms and other similar attributes.

Approach

The data was extracted from Kaggle playground web platform. After processing the dataset and cleaning the inconsistencies, the numerical and categorical features used in the Airbnb price prediction model is generated. Various regression algorithms are used to predict the host-price based on a set of independent variables like neighborhood-geographical data, the day of the week, room type, availability and the nights occupied. The predictive models are also used to identify the variables that strongly influence the conversion using variable importance and probabilistic approaches. The models are evaluated using relevant model performance measures to arrive at the most robust models for prediction.

Key Learnings:

1. Different hosts and areas.
2. Which hosts are the busiest and the reason behind the same
3. Noticeable difference in traffic among different areas and the reason behind the same
4. Predictions based on locations, prices, reviews, etc.

What Ticks Airbnb?

Some key overviews:

1. Hosts are expected to set their own prices/price-range. Airbnb does offer a basic level of insight upon this strategy but that is often not enough as we found out in our study.

2. Paid third party pricing software is available, but generally you are required to put in your own expected average nightly price ('base price'), and the algorithm will vary the daily price around that base price on each day depending on day of the week, seasonality, how far away the date is, and other factors.

3. Airbnb pricing is important to get right, particularly in big cities like NYC where there is lots of competition and even small differences in prices can make a big difference.

4. This project aims to solve this problem, by using machine learning to predict the price for an Airbnb listing. This project will analyze Airbnb listings to better understand how different attributes such as bedrooms, location, house type amongst others can be used to accurately predict the price of a new listing that is optimal in terms of the host's profitability yet affordable to their guests. This project is intended to be helpful to the internal pricing tools that Airbnb provides to its hosts. Furthermore, additional analysis is performed to ascertain the likelihood of a listing's availability for potential guests to consider while making a booking.

Problem Statement:

Pricing a rental property on Airbnb is a challenging task for the owner as it determines the number of customers for the place. On the other hand, customers have to evaluate an offered price with minimal knowledge of an optimal value for the property. This project aims to develop a reliable price prediction model using machine learning algorithms to aid both the property owners and the customers with price evaluation given minimal available information about the property. In this project, we will build a price prediction model of Airbnb listings and make comparisons between different methods.

Chapter 2: About the Dataset

{Usability: 10.0 License: CCO: Public Domain

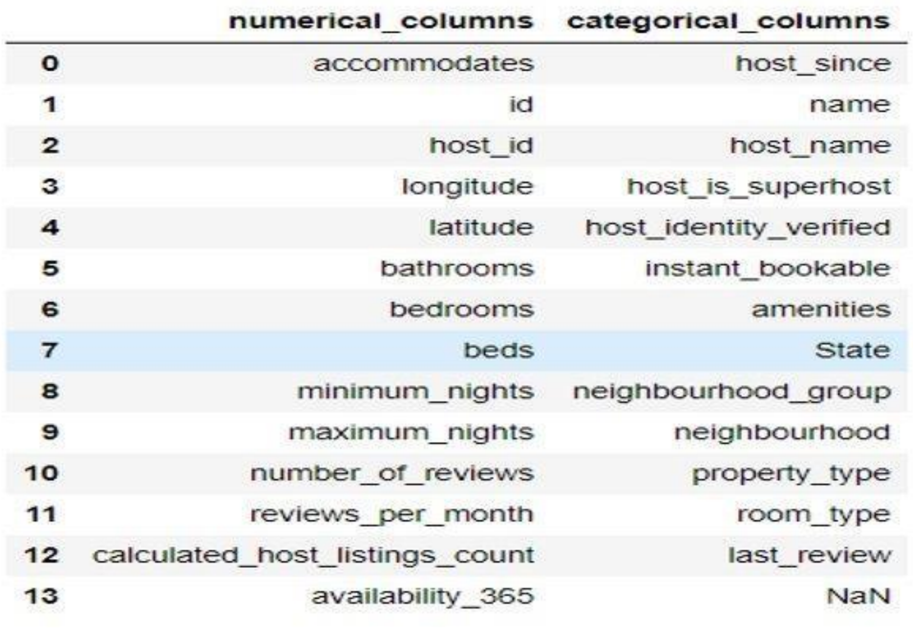
Tags: Business, Analysis Number of records: 158249

Number of features: 27 + 1 target variable}

Feature Name	**Feature Description**
price	Daily price of the listing in local currency
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
host_since	The date the host/user was created
host_is_superhost	If the host is also the superhost or not?
host_identity_verified	Is the identity of the host verified or not?
instant_bookable	Can the property(listing) be booked instantly?
accommodates	maximum capacity of the listing
amenities	Amenities that are being provided by the host
State	State in which the listing is present
neighbourhood_group	Group in which the neighbourhood lies
neighbourhood	The name of the neighbourhood
latitude	Latitude of the listing
longitude	Longitude of the listing
property_type	Self-selected property type. Hotels and/or Bed and Breakfasts.
room_type	Type of room being provided by the listing
bathrooms	Number of bathrooms in the listing
bedrooms	Number of bedrooms in the listing
beds	Number of bed(s) in the listing
minimum_nights	Minimum no. Of nights for booking
maximum_nights	Maximum no. of nights stay for booking
number_of_reviews	No. Of reviews over the listing’s lifetime
last_review	The date of the last/newest review
reviews_per_month	Average reviews per month on listing
calculated_host_listings_count	total number of listings by host
availability_365	Number of days in year the listing is available for rent

This public dataset is part of Airbnb, and the original source can be found on their website and Kaggle's playground datasets.

This data file includes all needed information to find out more about hosts, geographical availability, necessary metrics to make predictions and draw conclusions.

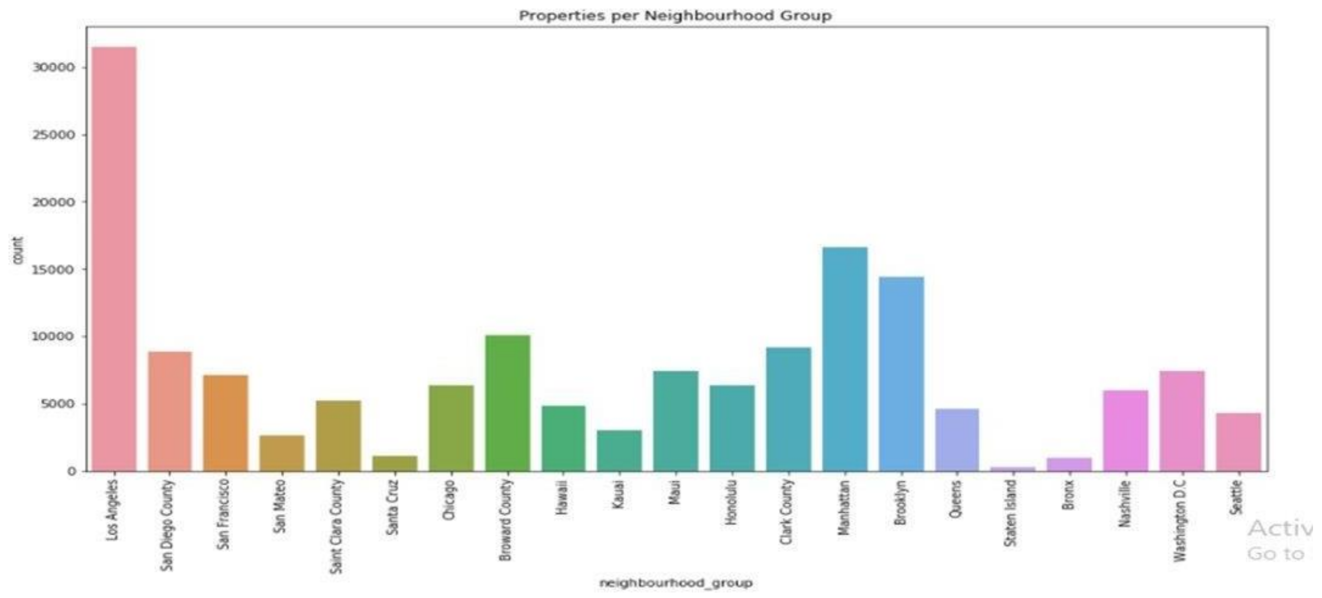


	numerical_columns	categorical_columns
0	accommodates	host_since
1	id	name
2	host_id	host_name
3	longitude	host_is_superhost
4	latitude	host_identity_verified
5	bathrooms	instant_bookable
6	bedrooms	amenities
7	beds	State
8	minimum_nights	neighbourhood_group
9	maximum_nights	neighbourhood
10	number_of_reviews	property_type
11	reviews_per_month	room_type
12	calculated_host_listings_count	last_review
13	availability_365	NaN

There are 15 numerical columns and 12 categorical columns.

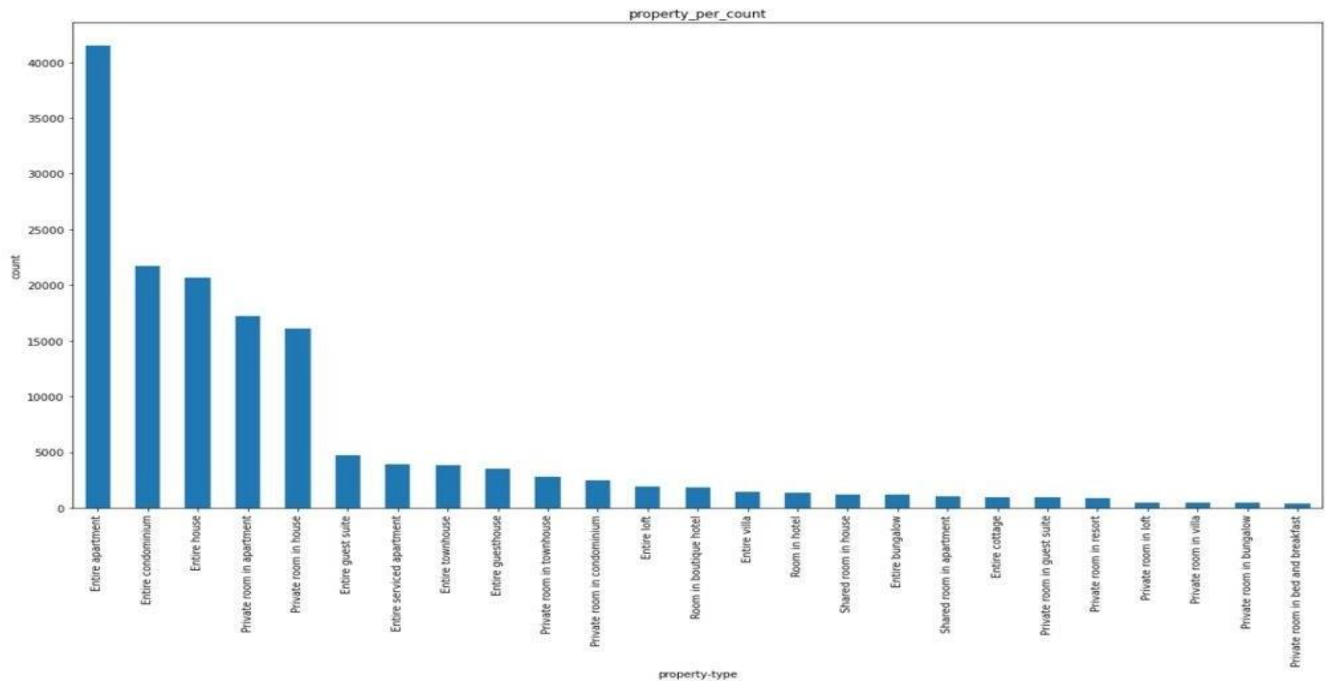
About the feature columns (univariate analysis)

1. Number of Airbnb listings in the neighborhood group: -



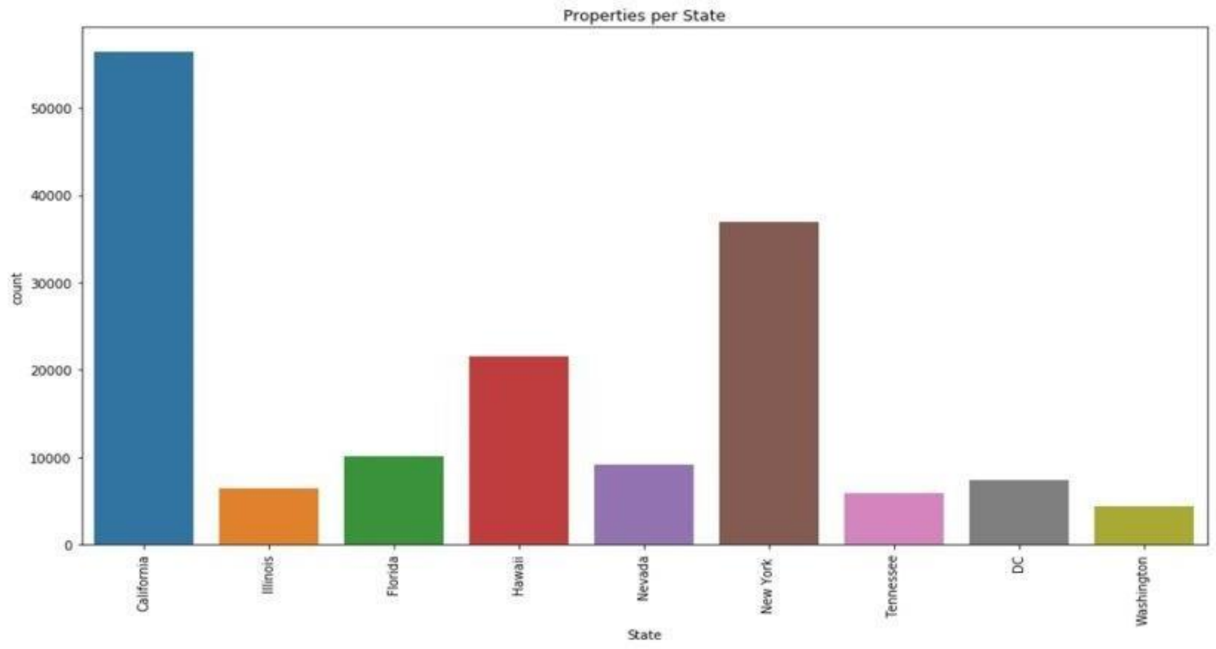
Insight: Most properties are located in Los Angeles,
Bronx, Staten Island, Santa Cruz have minimum number of properties

2. Hierarchy in property types: - Showing top property types: -



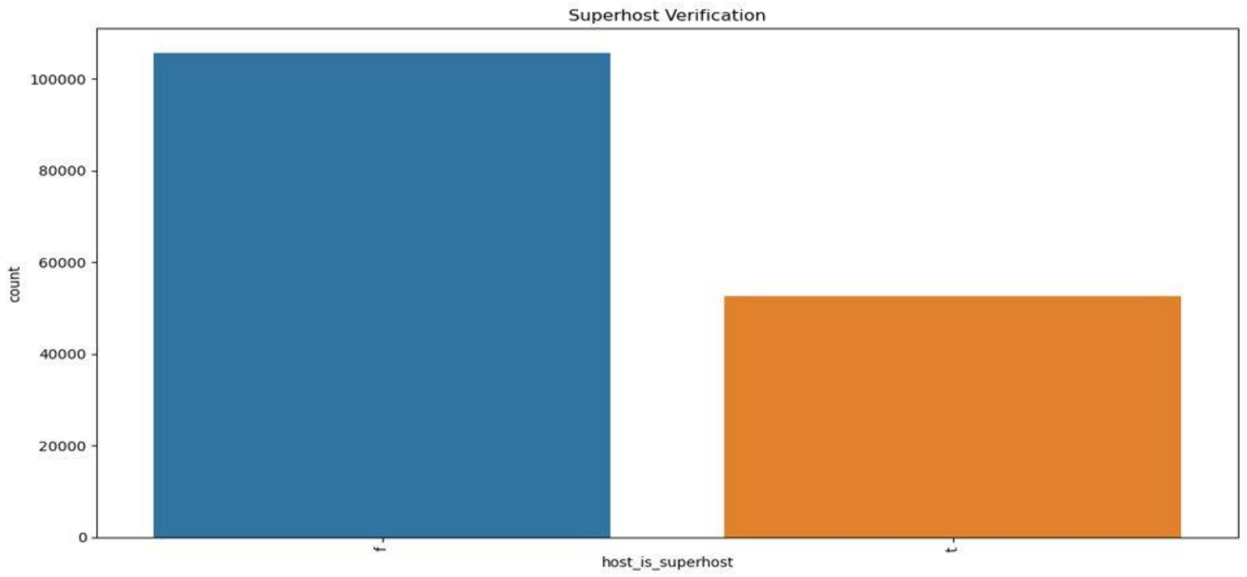
Insight: Most people offer entire apartments and very few hosts offer private rooms in bed and breakfasts.

3. State-wise distribution:-



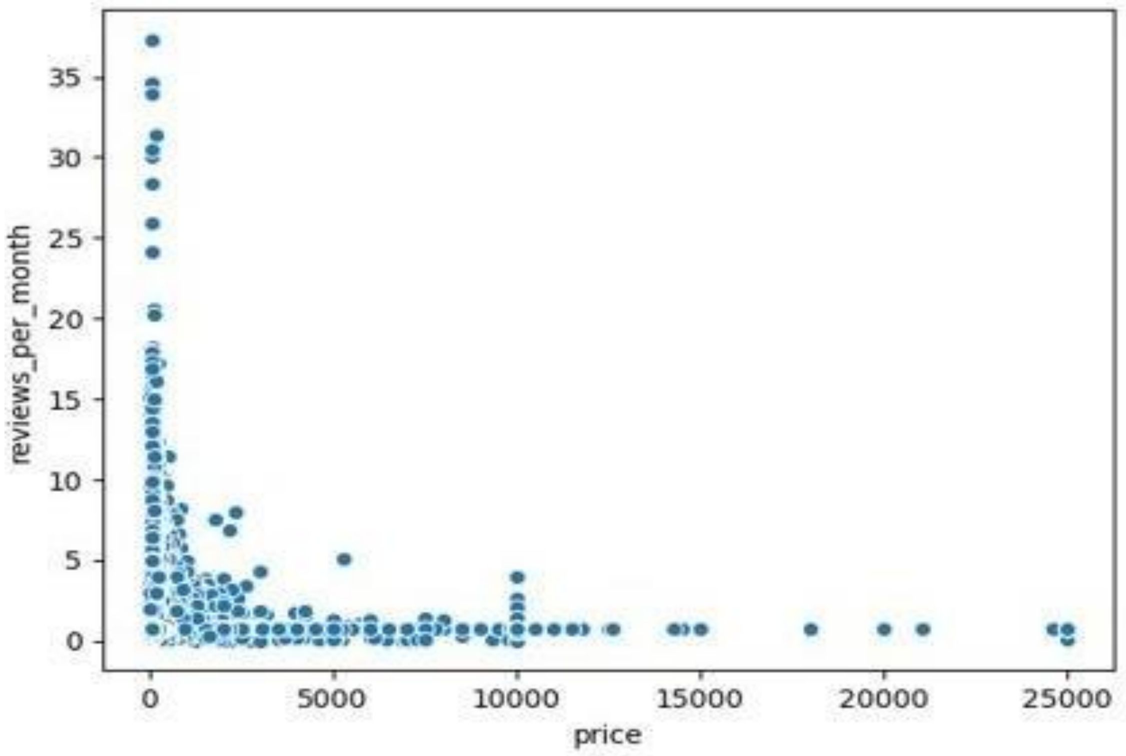
Insight: California has the highest number of listings while Washington has the lowest amongst all popular states.

4. Host v/s Super-host:-

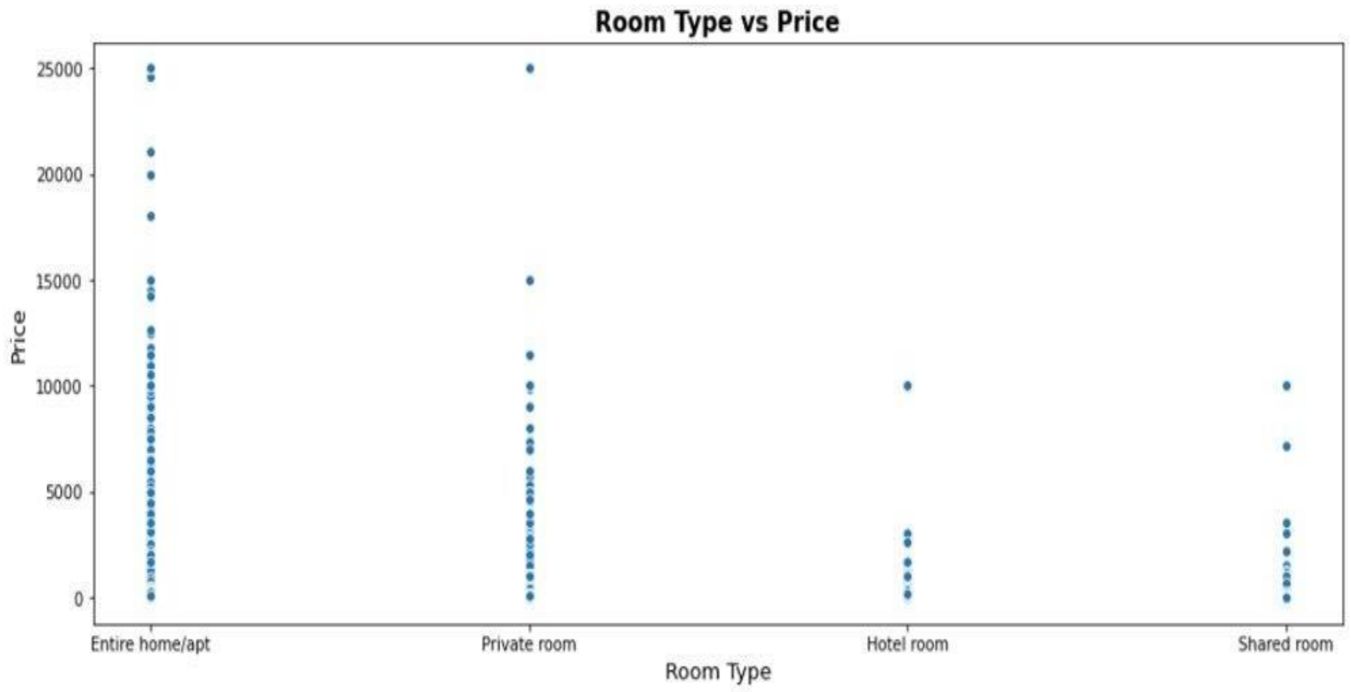


Insight: Verification has been done for most of the listings and there are more super-hosts than normal hosts.

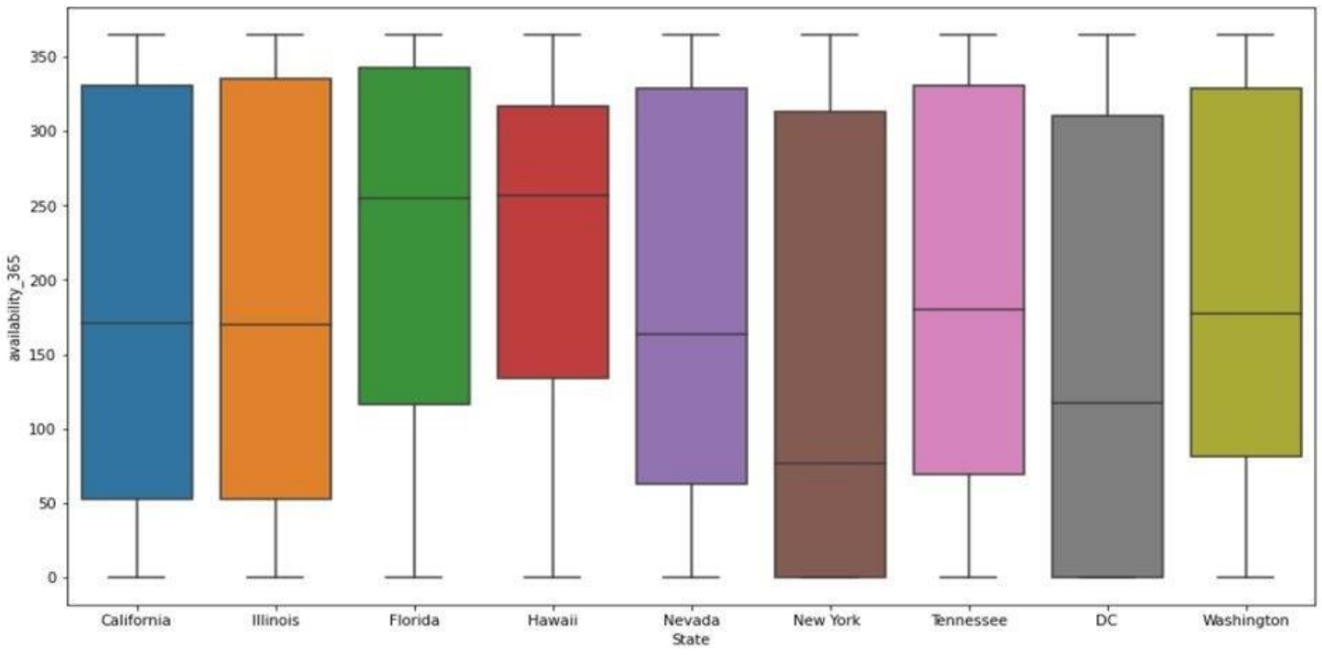
Bi-variate & Multi-variate Analysis: -



Insight: Those who pay less reviewed more and those who pay more reviewed less

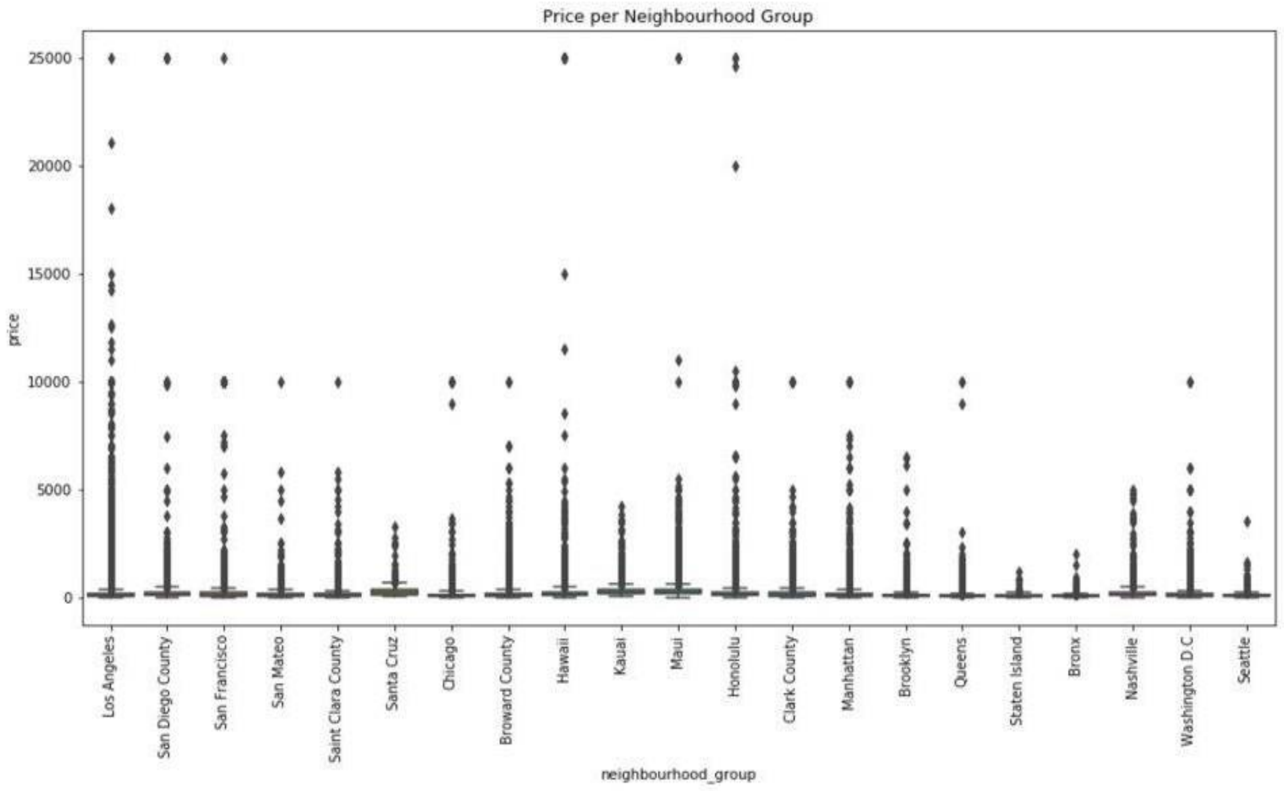


Insight: Entire homes/apartments have a higher price range.

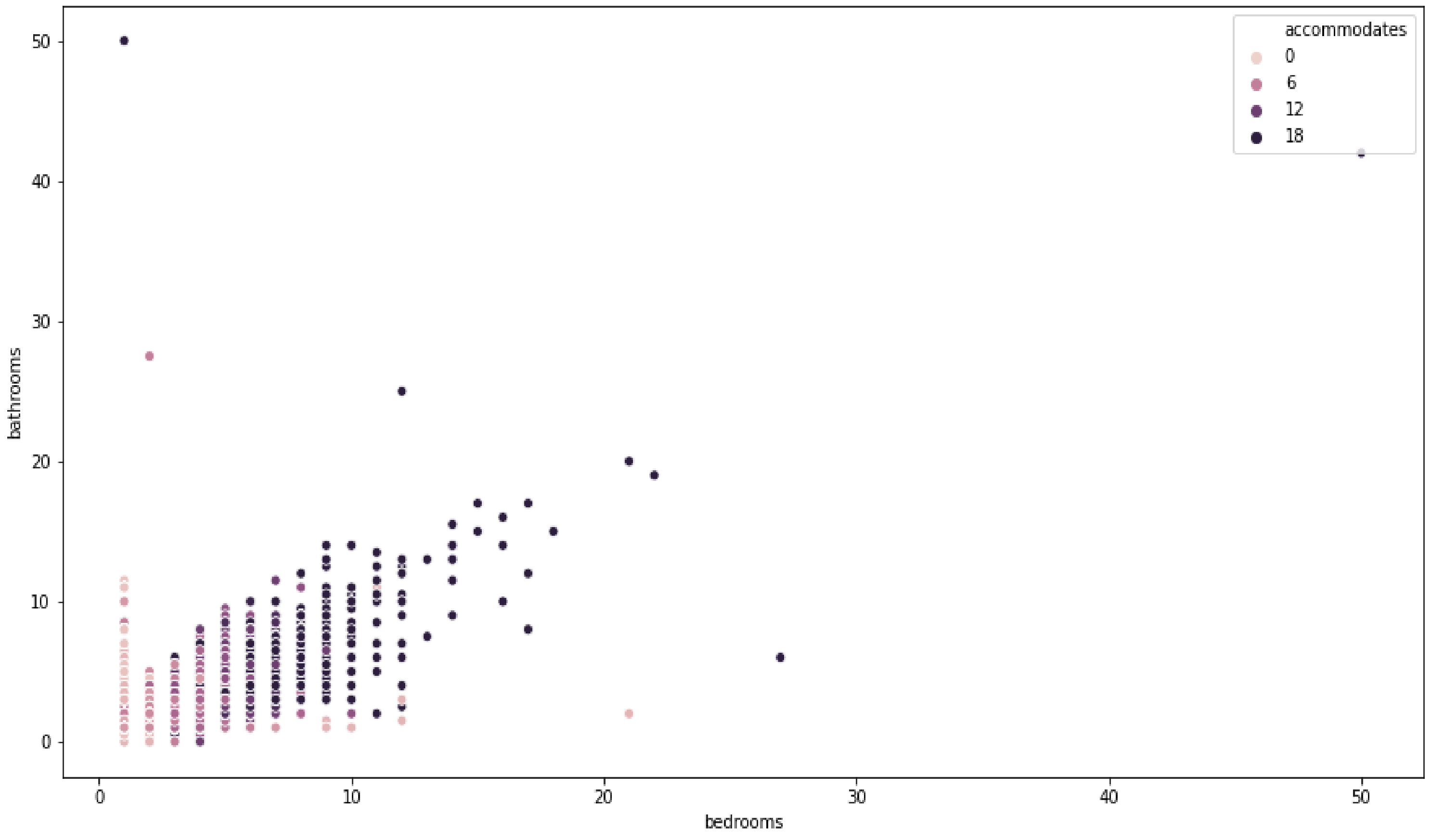


Insight: The state of New York has less availability of listings in a year

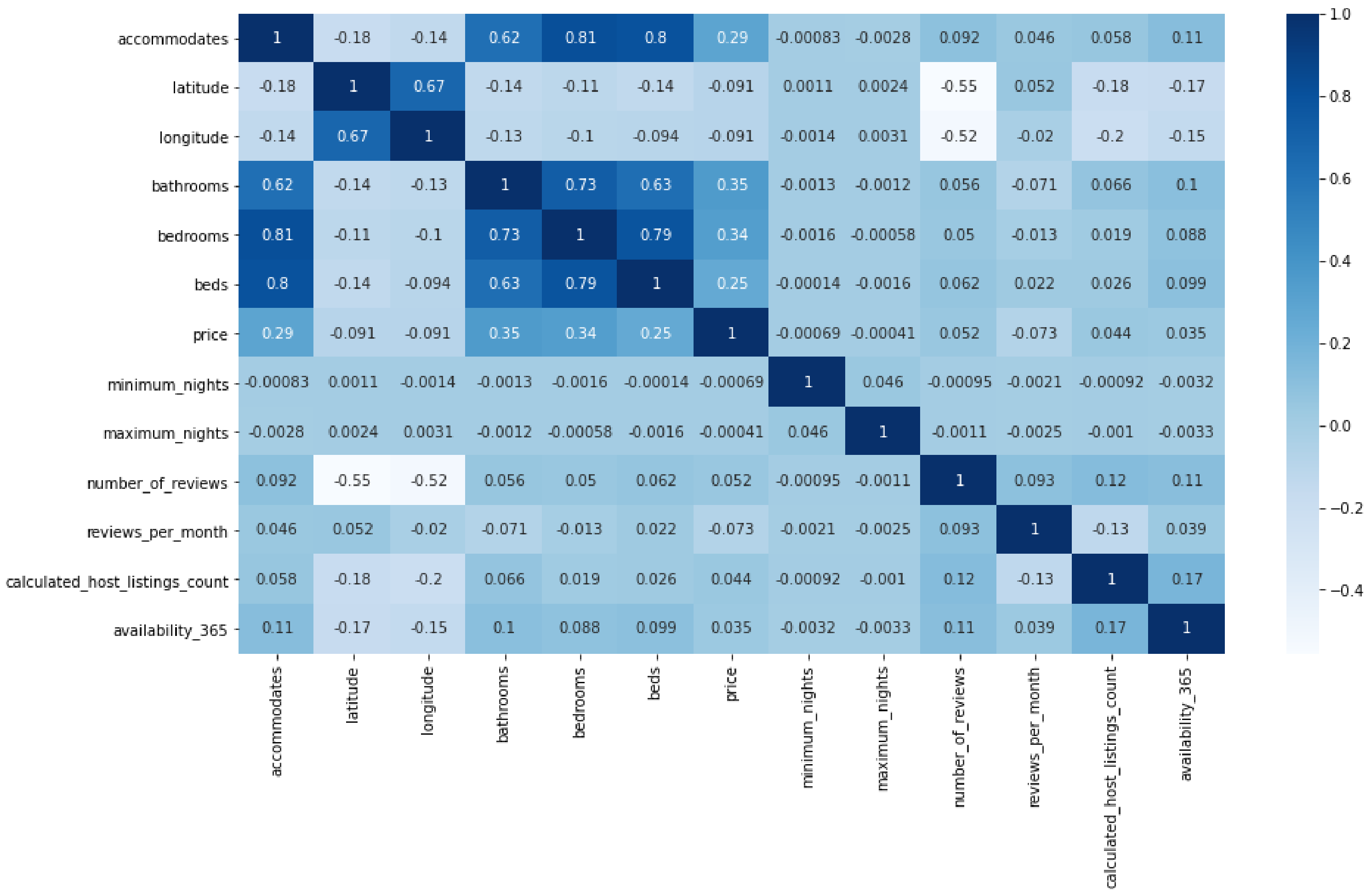
Neighborhood v/s Price: -



Insight: Listings in Los Angeles are on the more expensive side compared to listings in the state of Staten Island

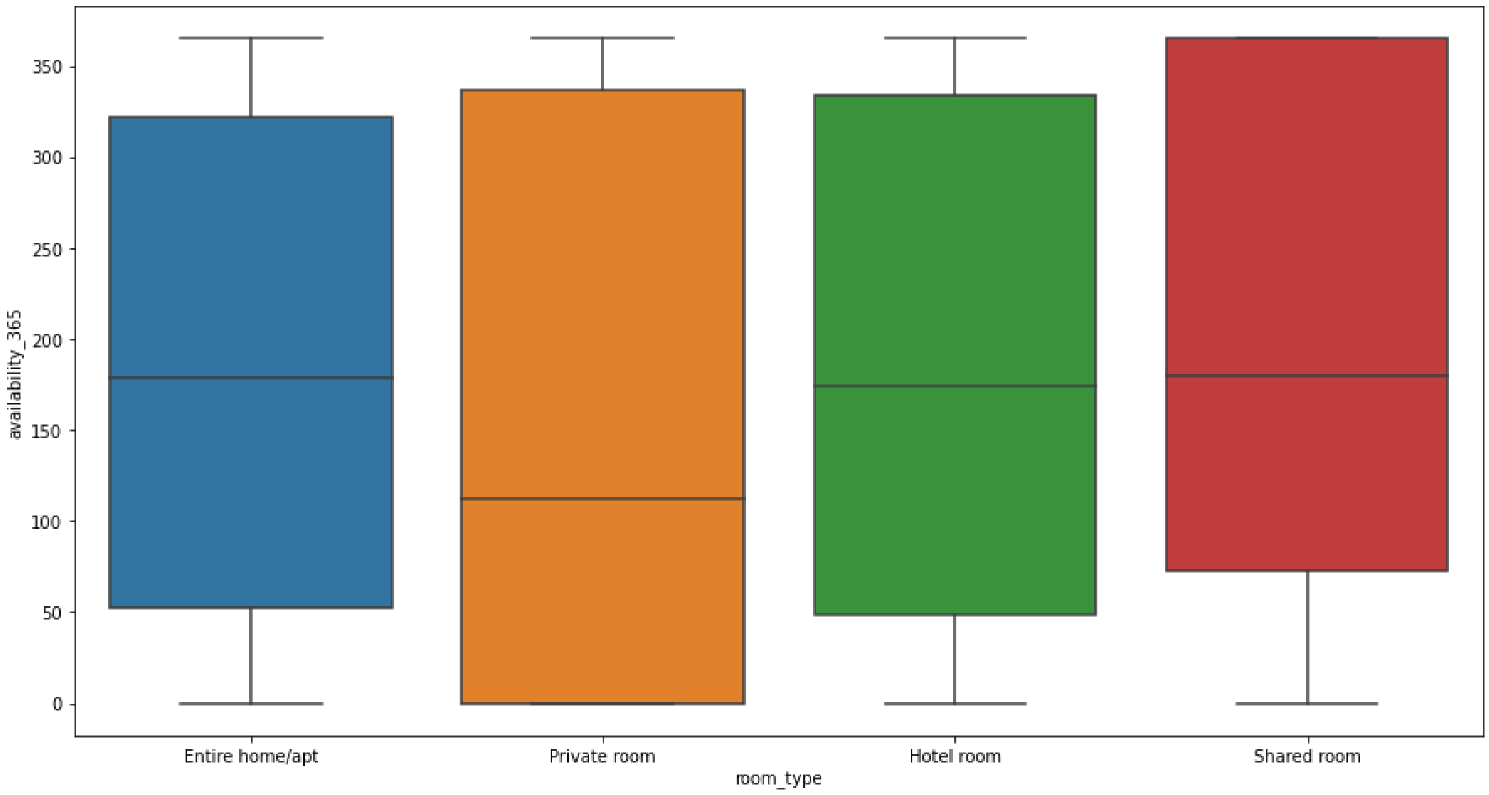


Insight: The listings which have more bathrooms have more bedrooms and can accommodate more people.

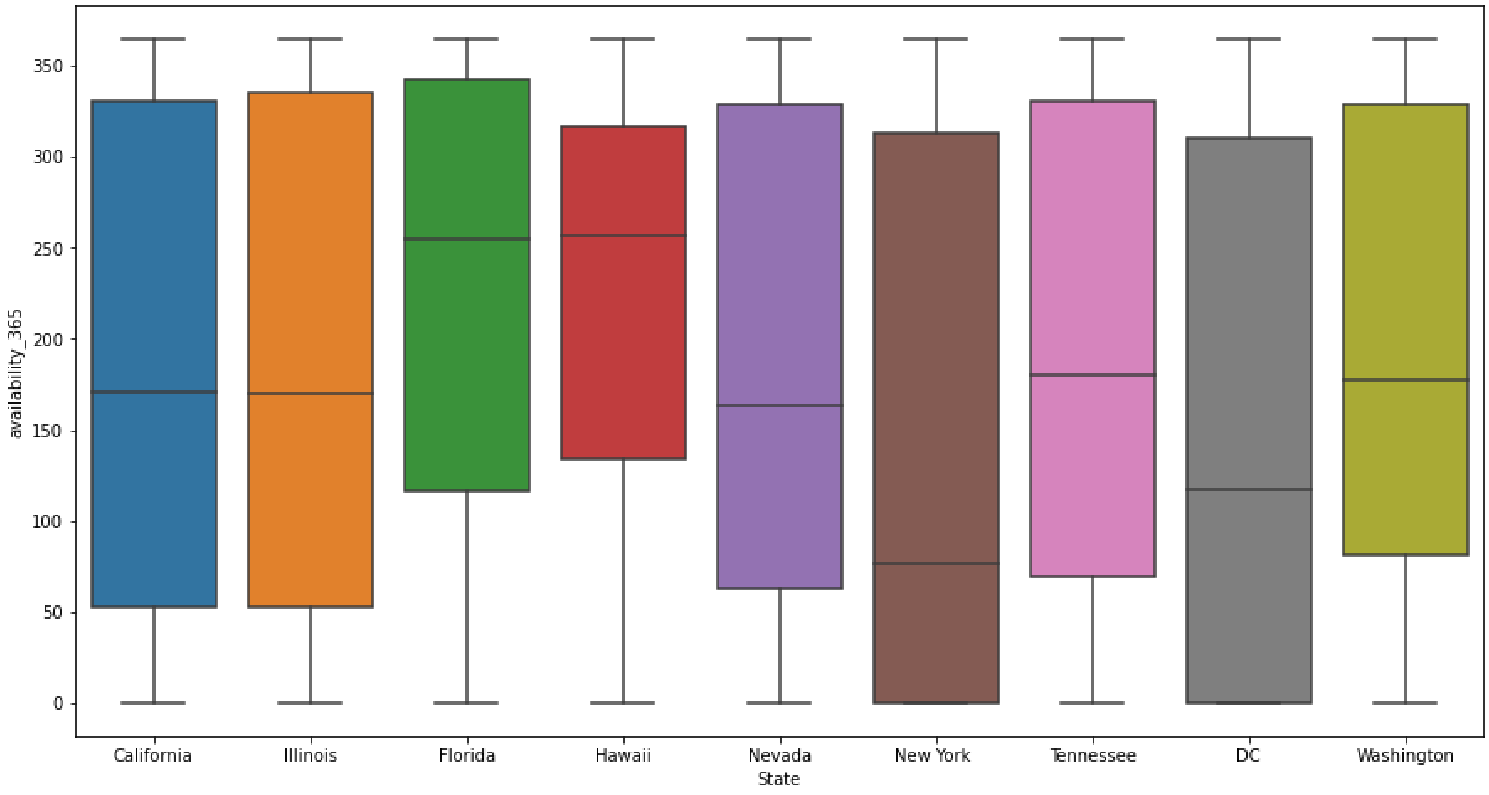


	accommodates	latitude	longitude	bathrooms	bedrooms	beds	price	minimum_nights	maximum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availability_365
accommodates	1	-0.18	-0.14	0.62	0.81	0.8	0.29	-0.00083	-0.0028	0.092	0.046	0.058	0.11
latitude	-0.18	1	0.67	-0.14	-0.11	-0.14	-0.091	0.0011	0.0024	-0.55	0.052	-0.18	-0.17
longitude	-0.14	0.67	1	-0.13	-0.1	-0.094	-0.091	-0.0014	0.0031	-0.52	-0.02	-0.2	-0.15
bathrooms	0.62	-0.14	-0.13	1	0.73	0.63	0.35	-0.0013	-0.0012	0.056	-0.071	0.066	0.1
bedrooms	0.81	-0.11	-0.1	0.73	1	0.79	0.34	-0.0016	-0.00058	0.05	-0.013	0.019	0.088
beds	0.8	-0.14	-0.094	0.63	0.79	1	0.25	-0.00014	-0.0016	0.062	0.022	0.026	0.099
price	0.29	-0.091	-0.091	0.35	0.34	0.25	1	-0.00069	-0.00041	0.052	-0.073	0.044	0.035
minimum_nights	-0.00083	0.0011	-0.0014	-0.0013	-0.0016	-0.00014	-0.00069	1	0.046	-0.00095	-0.0021	-0.00092	-0.0032
maximum_nights	-0.0028	0.0024	0.0031	-0.0012	-0.00058	-0.0016	-0.00041	0.046	1	-0.0011	-0.0025	-0.001	-0.0033
number_of_reviews	0.092	-0.55	-0.52	0.056	0.05	0.062	0.052	-0.00095	-0.0011	1	0.093	0.12	0.11
reviews_per_month	0.046	0.052	-0.02	-0.071	-0.013	0.022	-0.073	-0.0021	-0.0025	0.093	1	-0.13	0.039
calculated_host_listings_count	0.058	-0.18	-0.2	0.066	0.019	0.026	0.044	-0.00092	-0.001	0.12	-0.13	1	0.17
availability_365	0.11	-0.17	-0.15	0.1	0.088	0.099	0.035	-0.0032	-0.0033	0.11	0.039	0.17	1

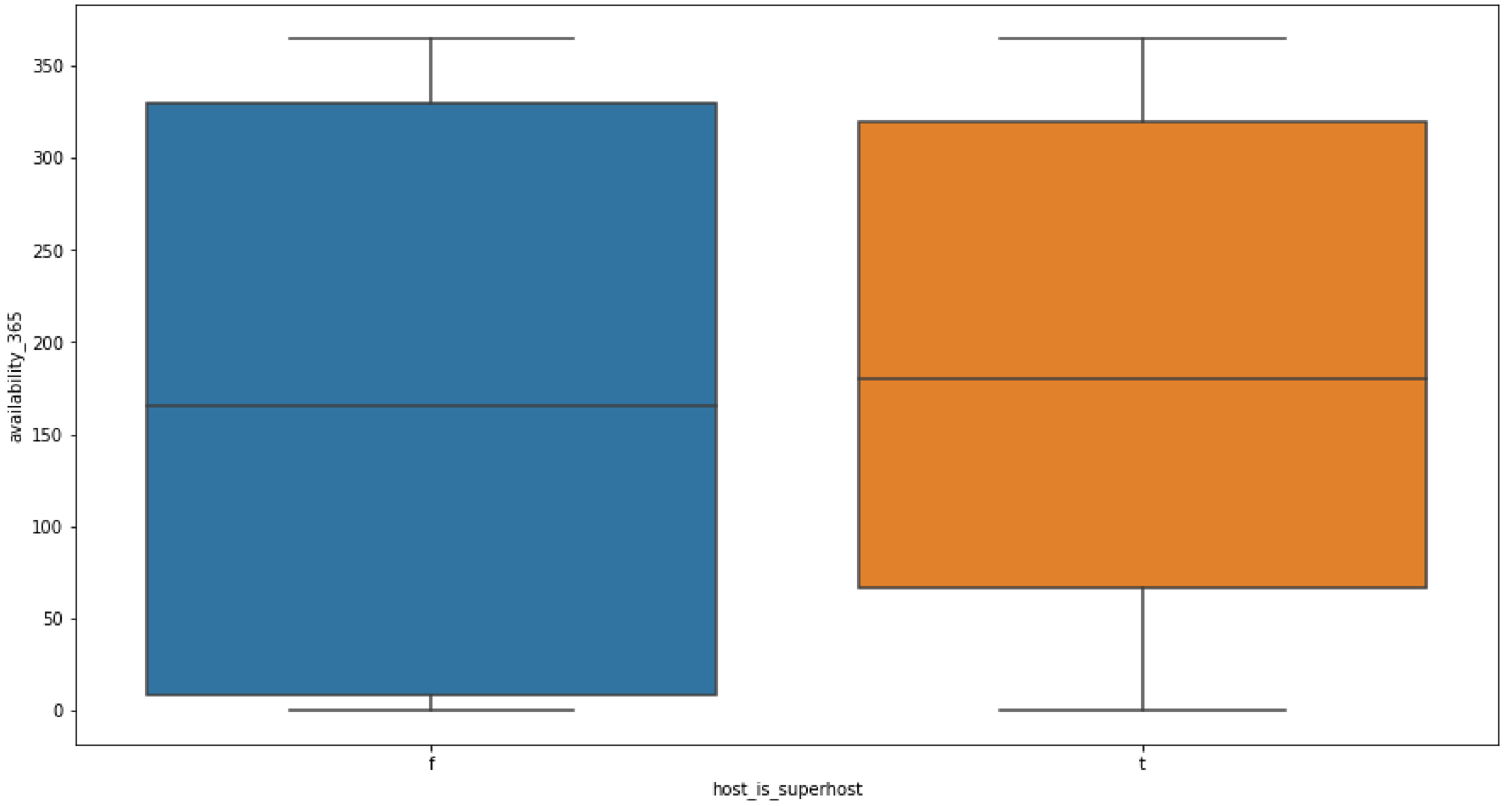
Insight: Accommodates have the highest positive co-relation with bedrooms



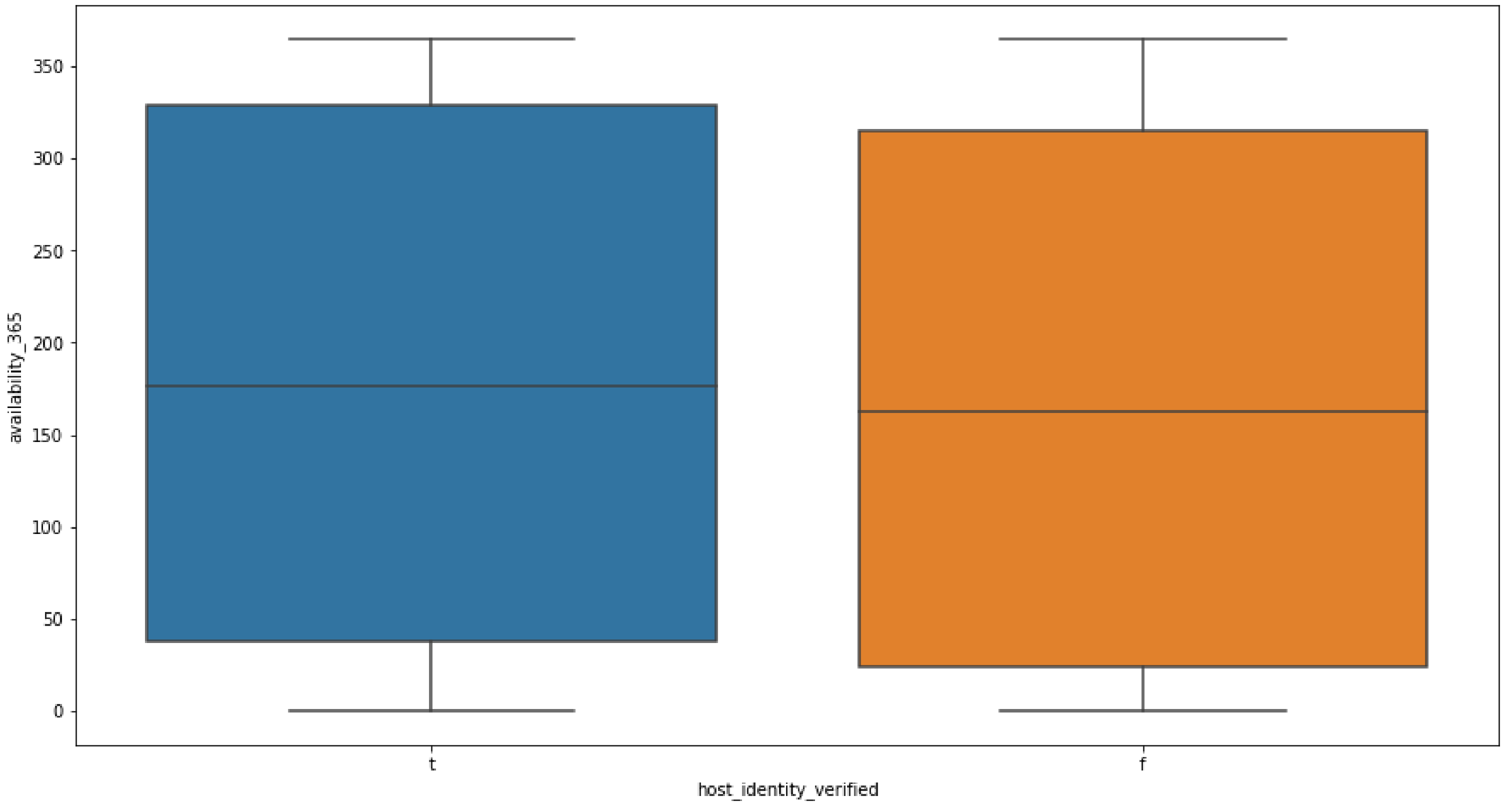
Insight: Private rooms have less availability within a year



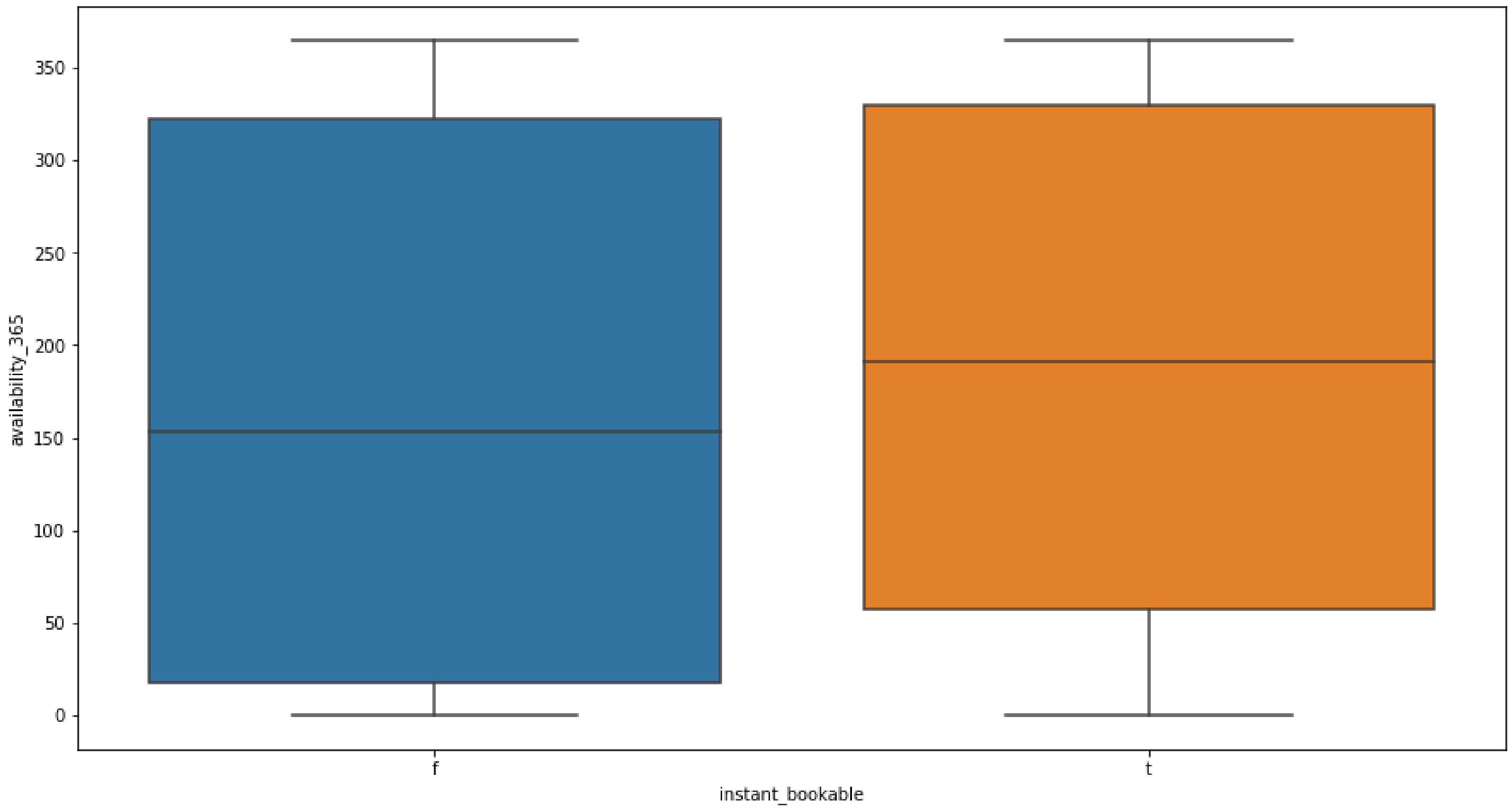
Insight: Listings in New York are less available within a year and listings in Florida and Hawaii have more availability



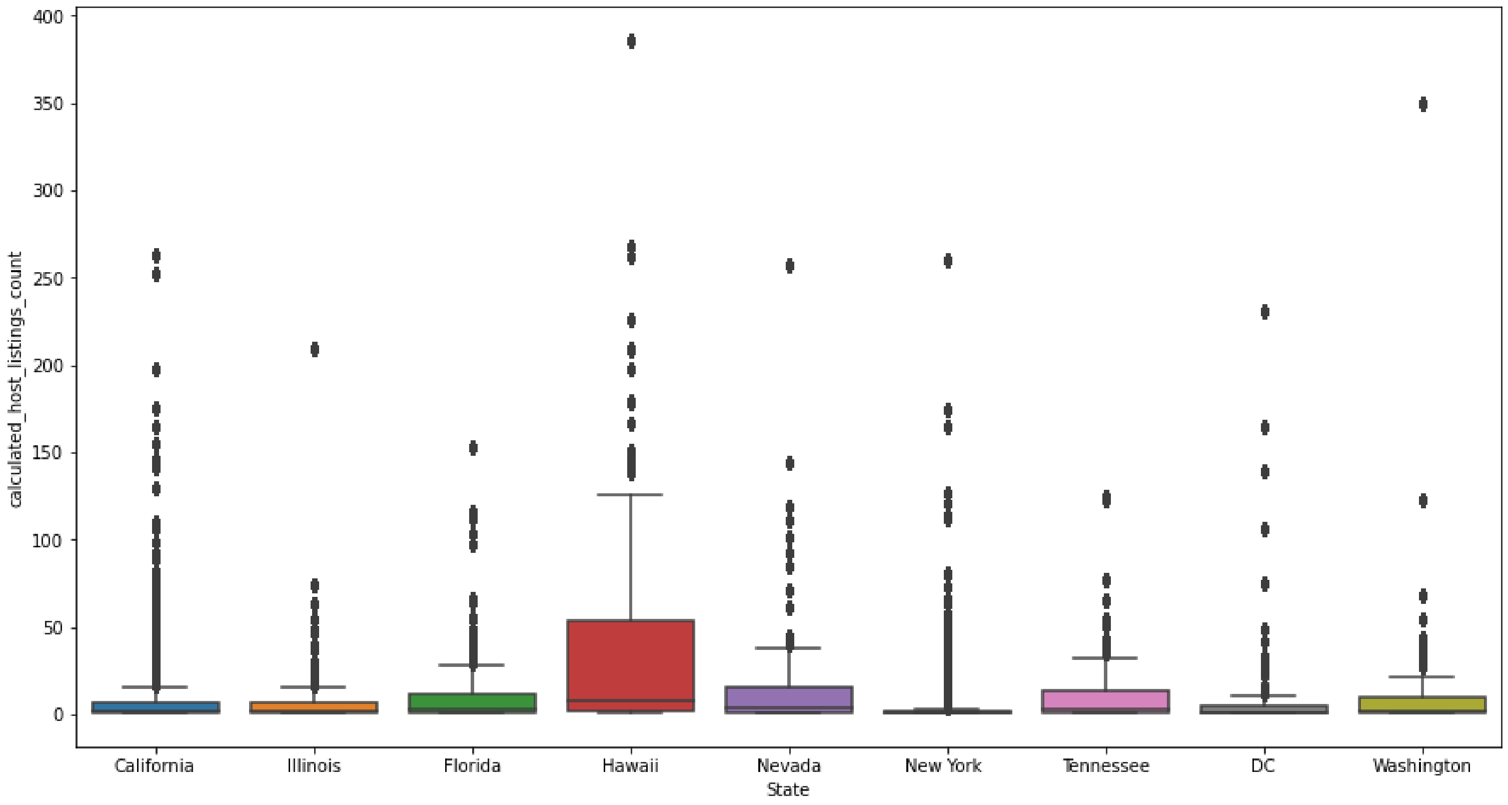
Insight: Average availability in a year is almost the same even if the host is a super host or not



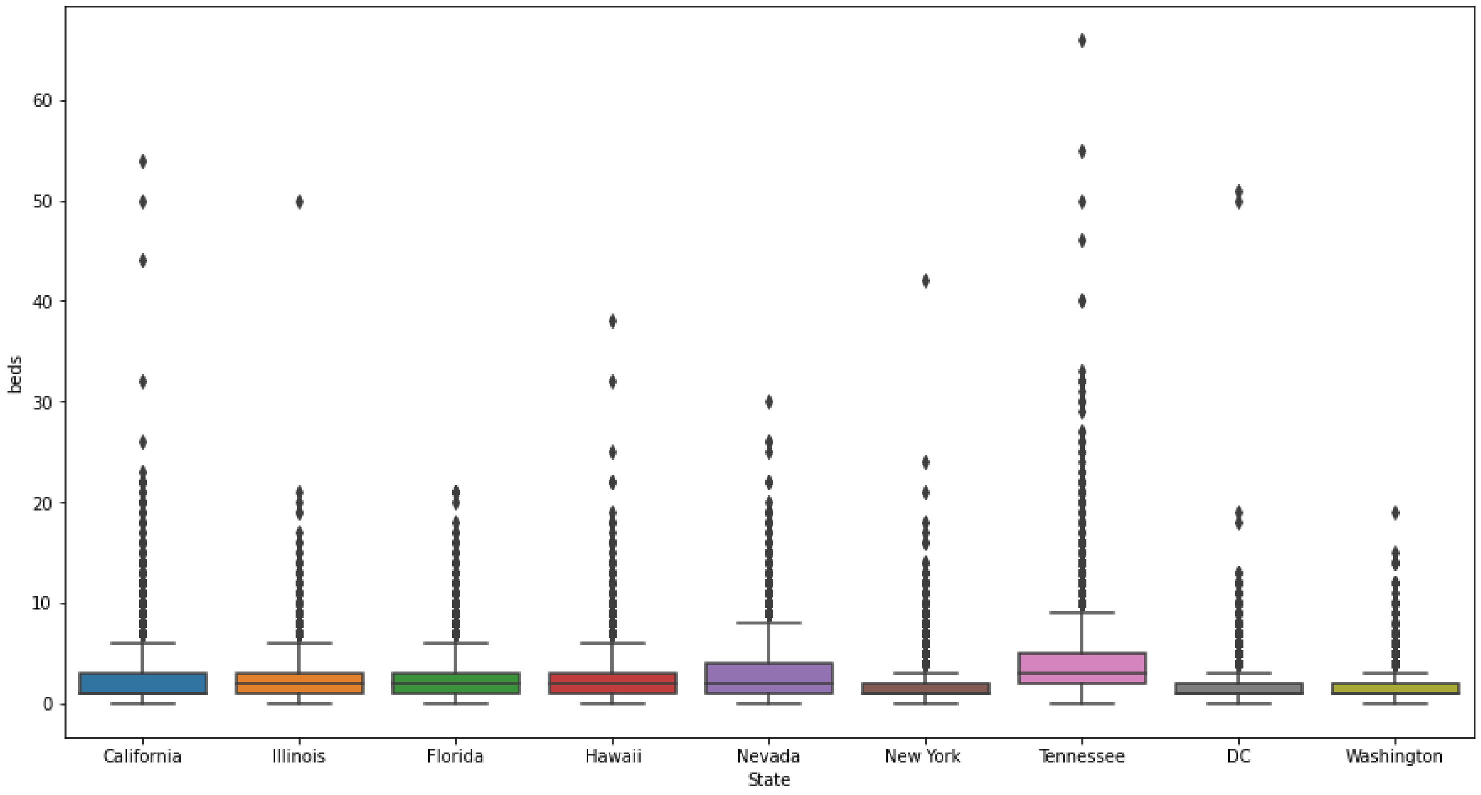
Insight: Average availability is almost the same even if the host's identity is verified or not



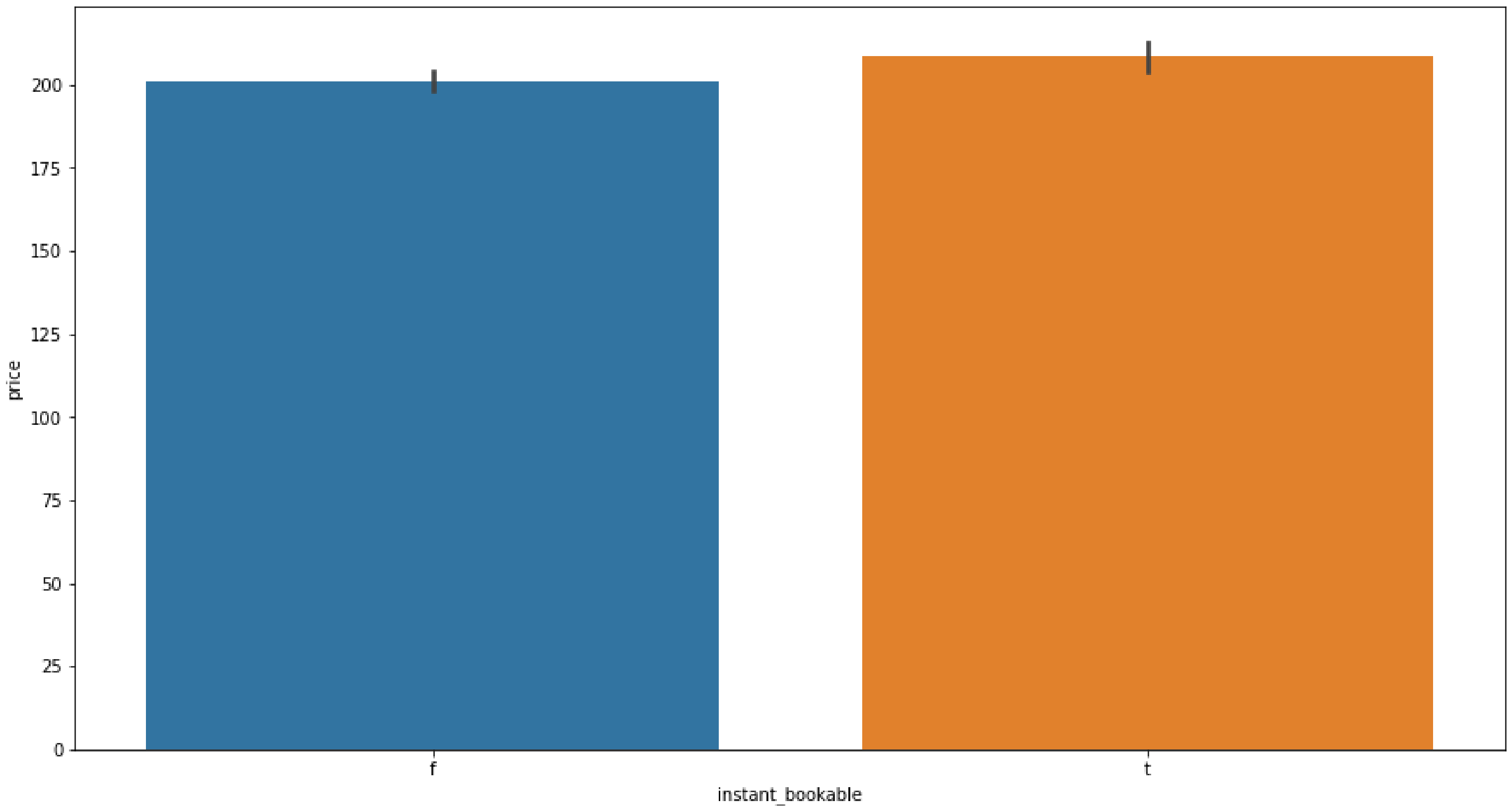
Insight: Listings that can be booked instantly had more availability in a year



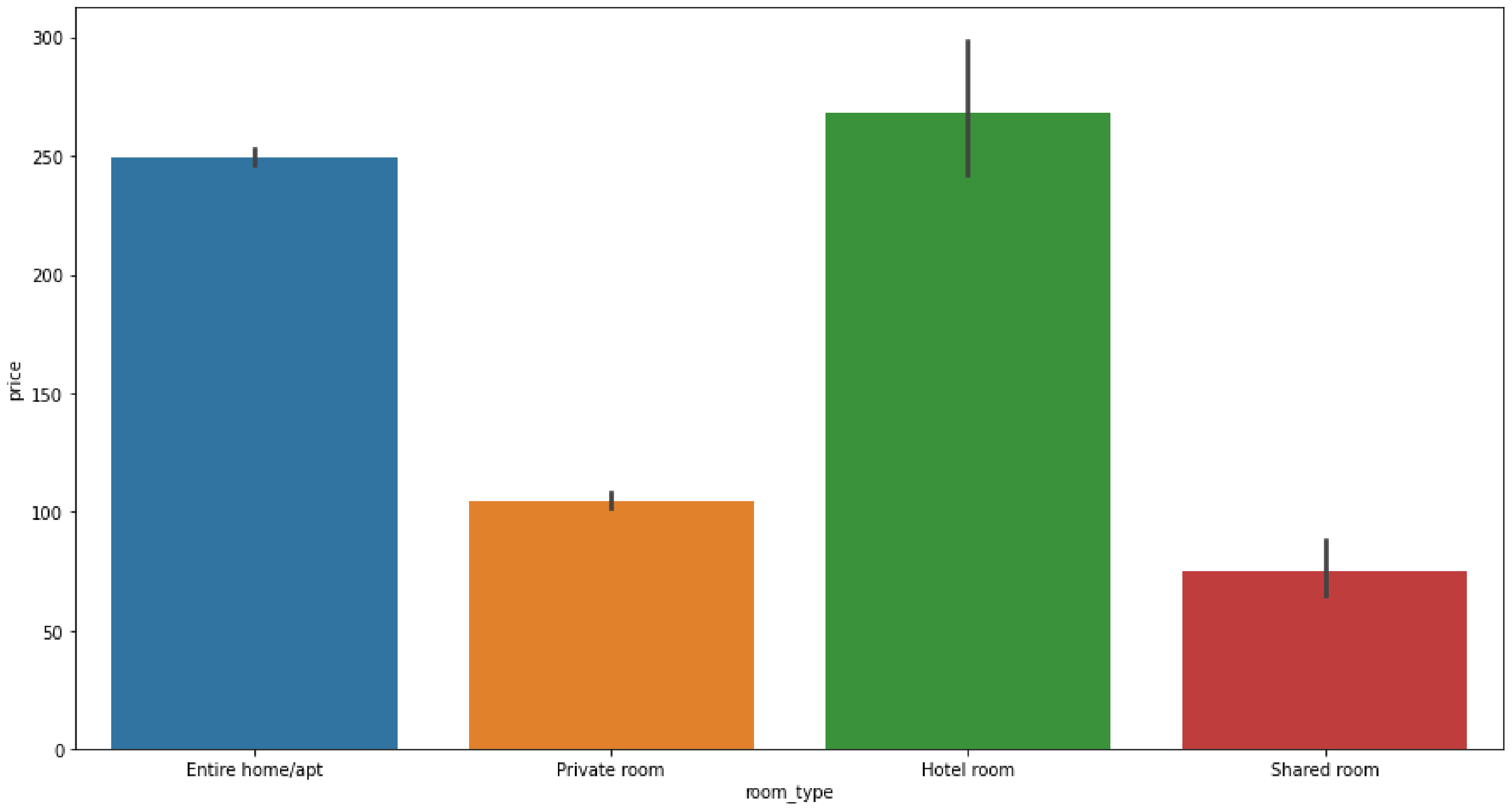
Insight: Hawaii is the state which has the highest count of host listings



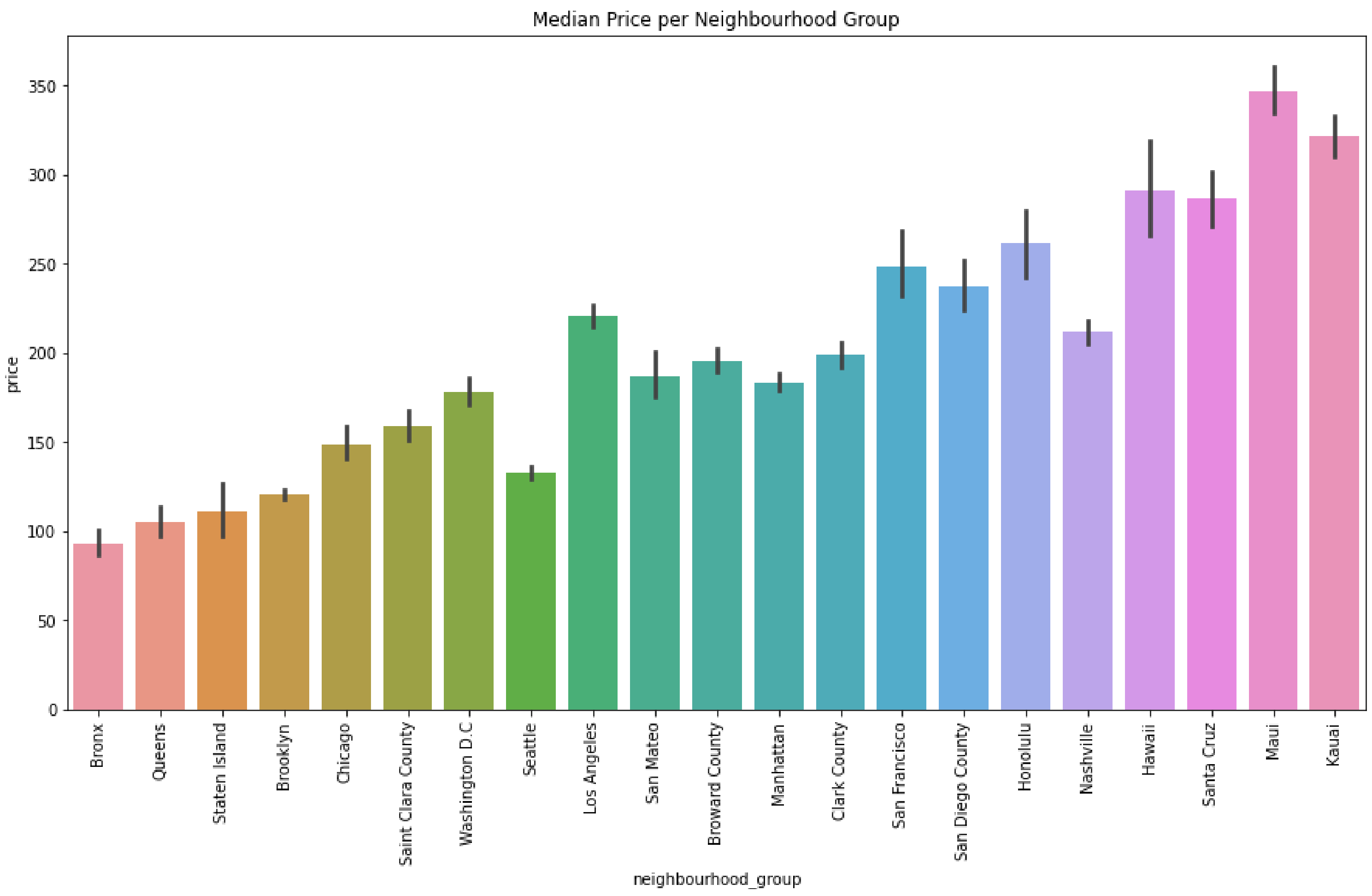
Insight: Tennesse is the state which has listings with highest number of beds



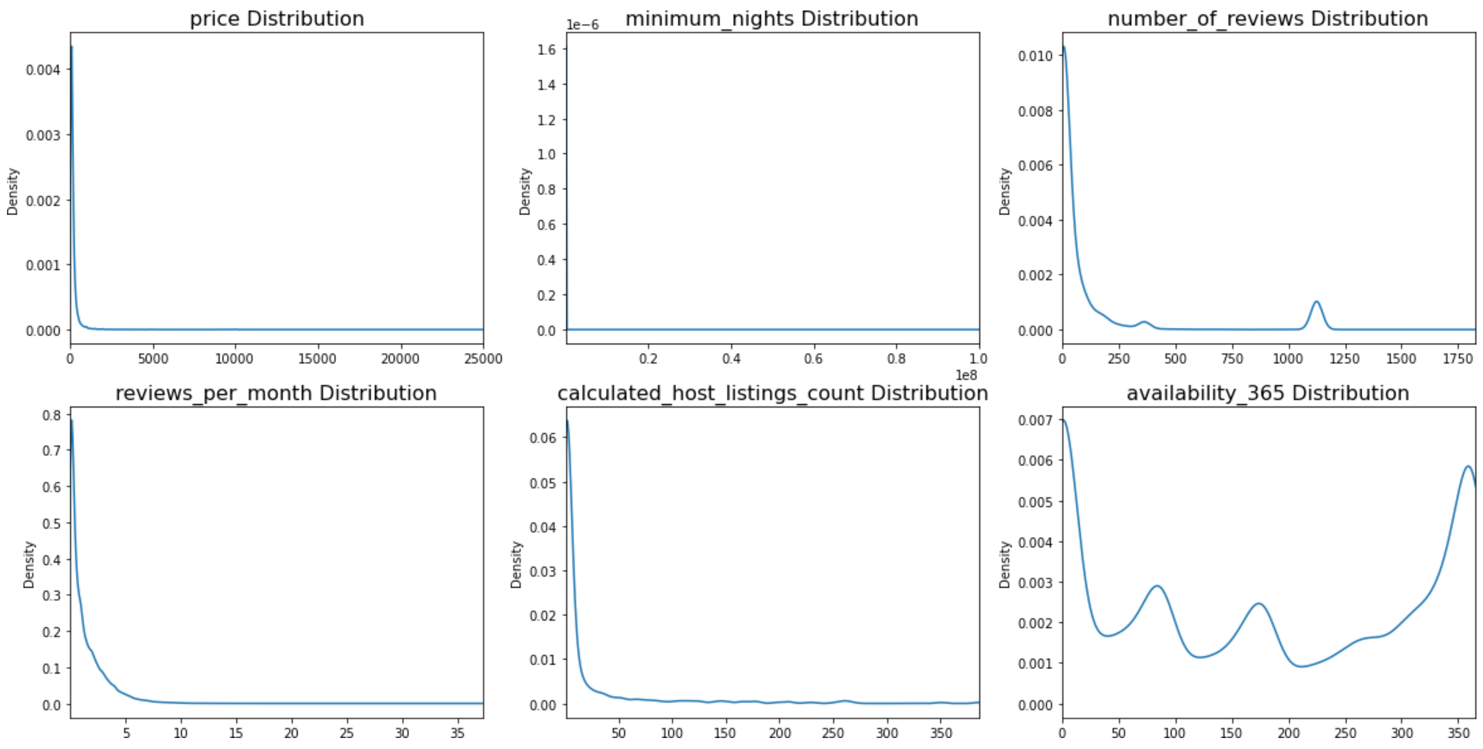
Insight :Listings that can be booked instantly have a higher price



Insight: Hotel rooms tend to have a bit higher prices than other room types

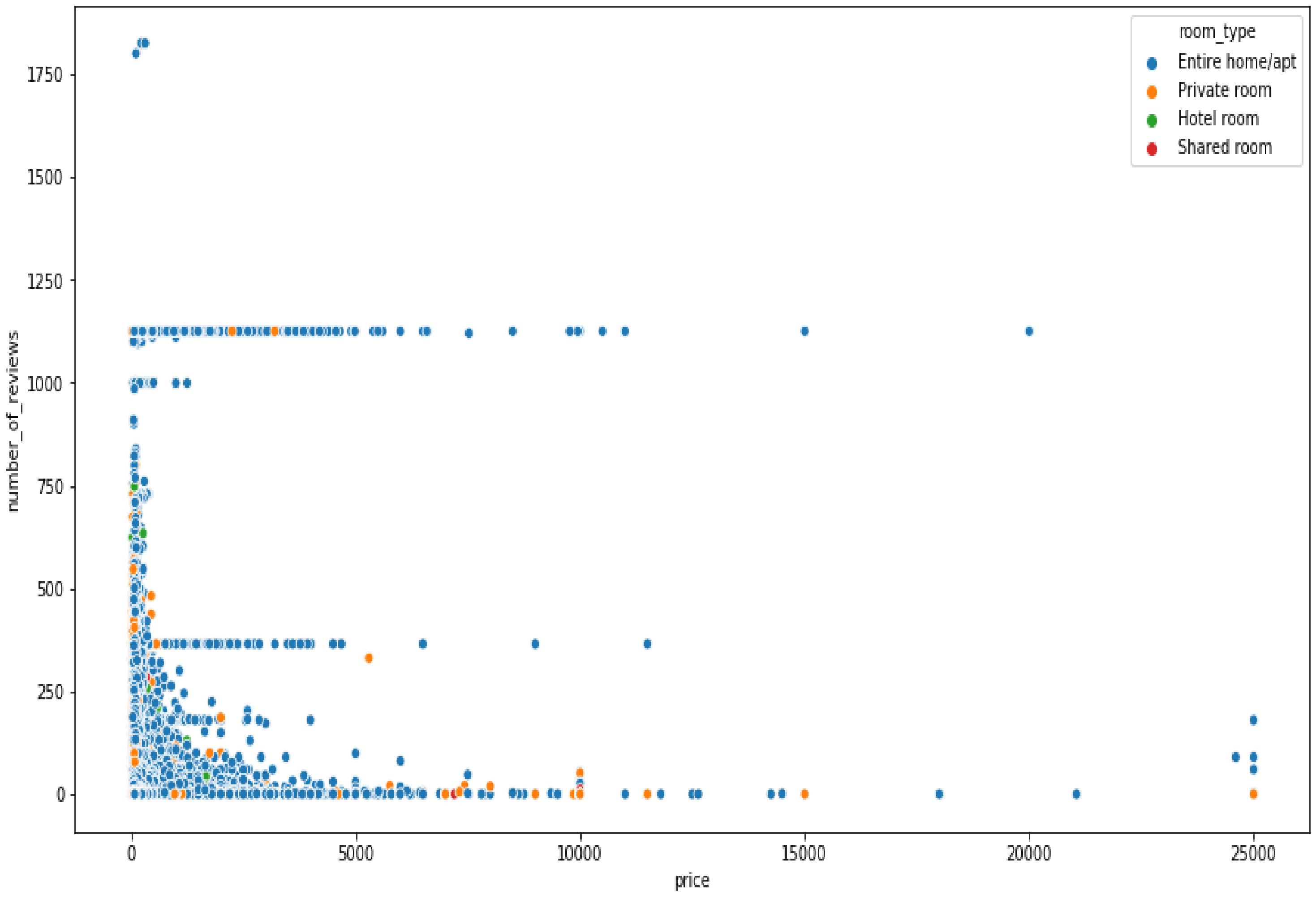


Insight: The median price per neighbourhood is highest in Maui and lowest in Bronx.

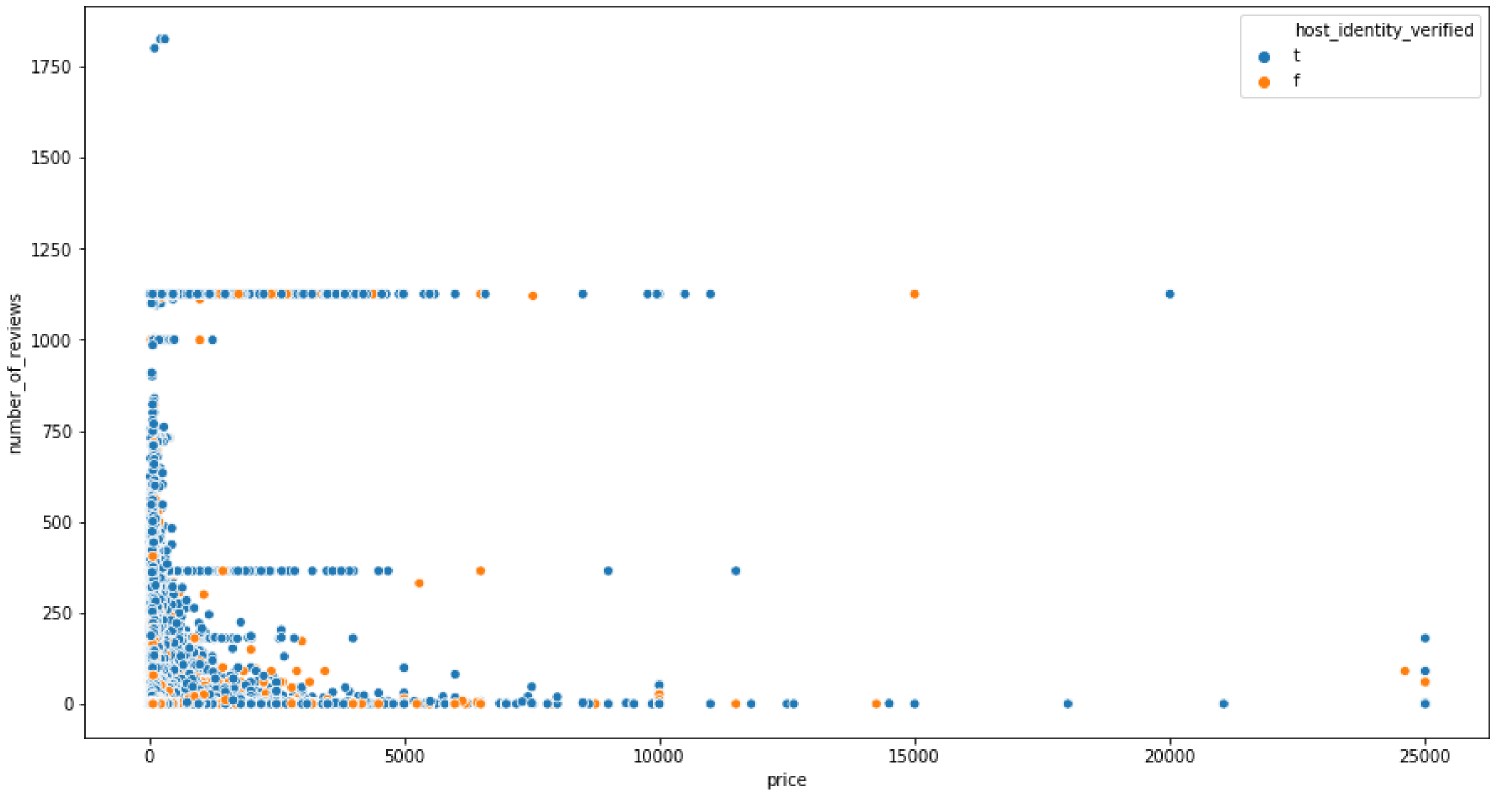


Numerical – Numerical Analysis

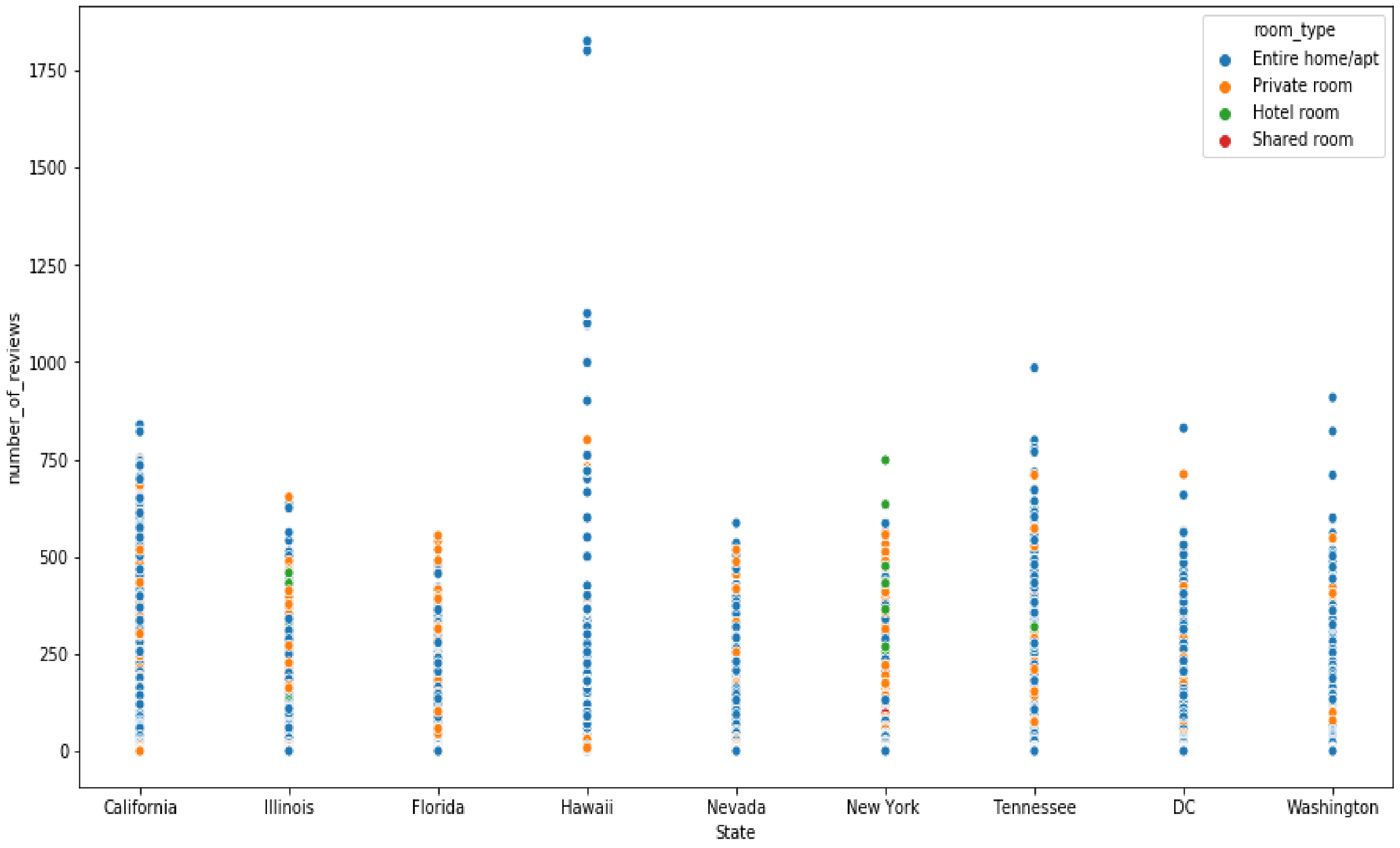
Insight: Most of the numerical columns are not normally distributed and are a bit left skewed.



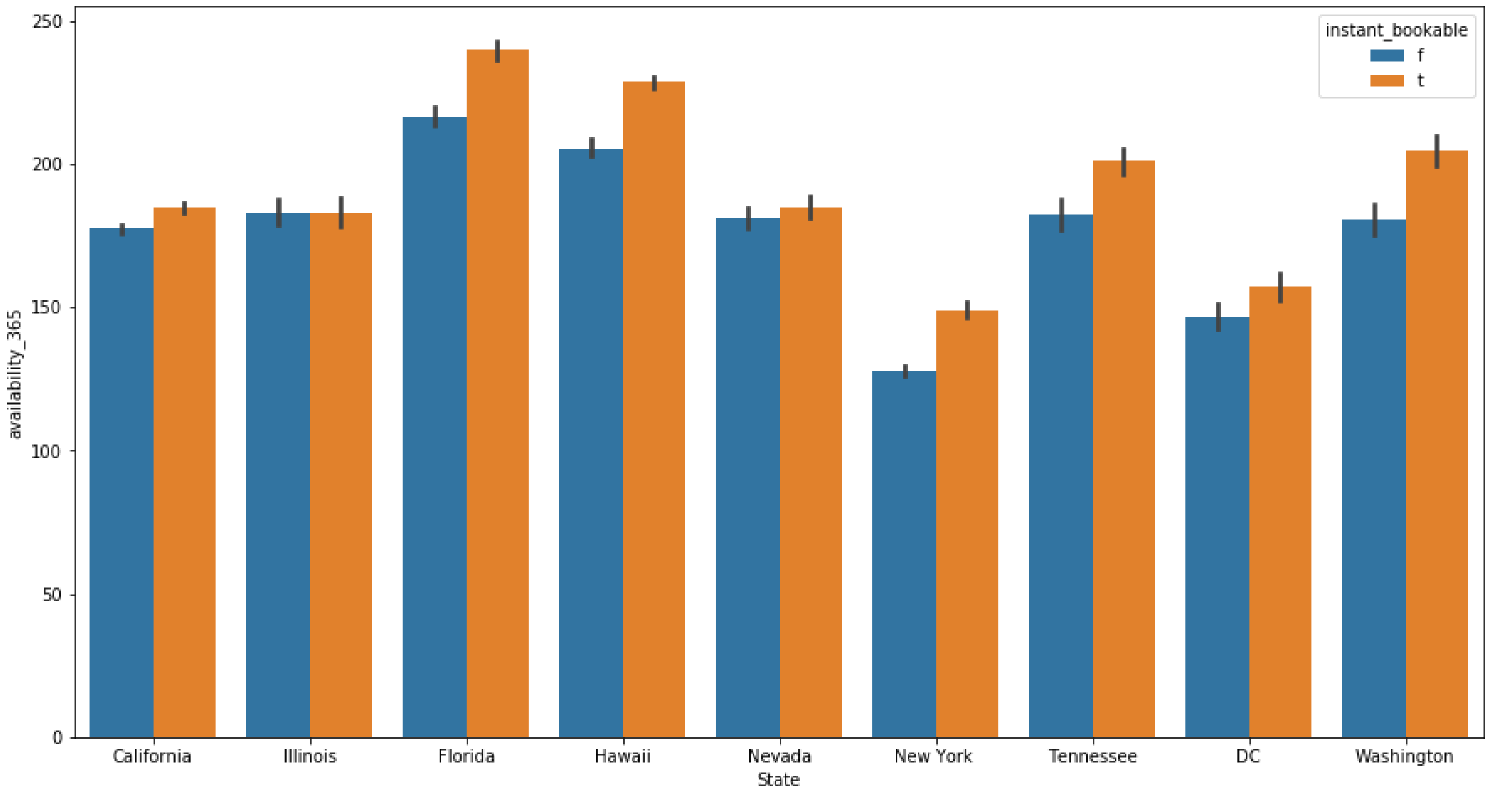
Insight: The Entire home/apt room type is reviewed more and also has a higher price



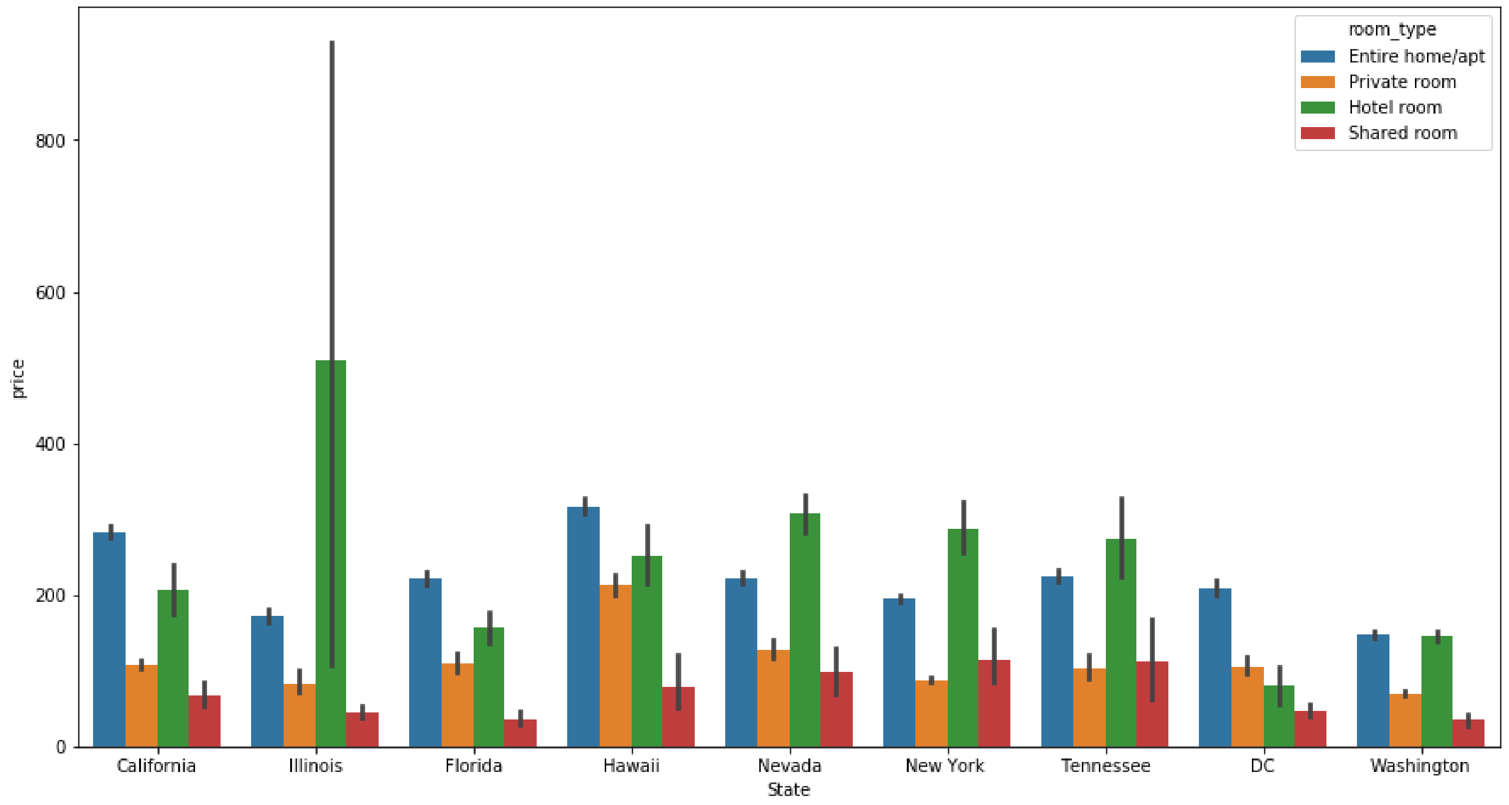
Insight: Listings whose host's identity is verified have more number of reviews and price



Insight: The Entire home/apt room type has a greater number of reviews in most of the states



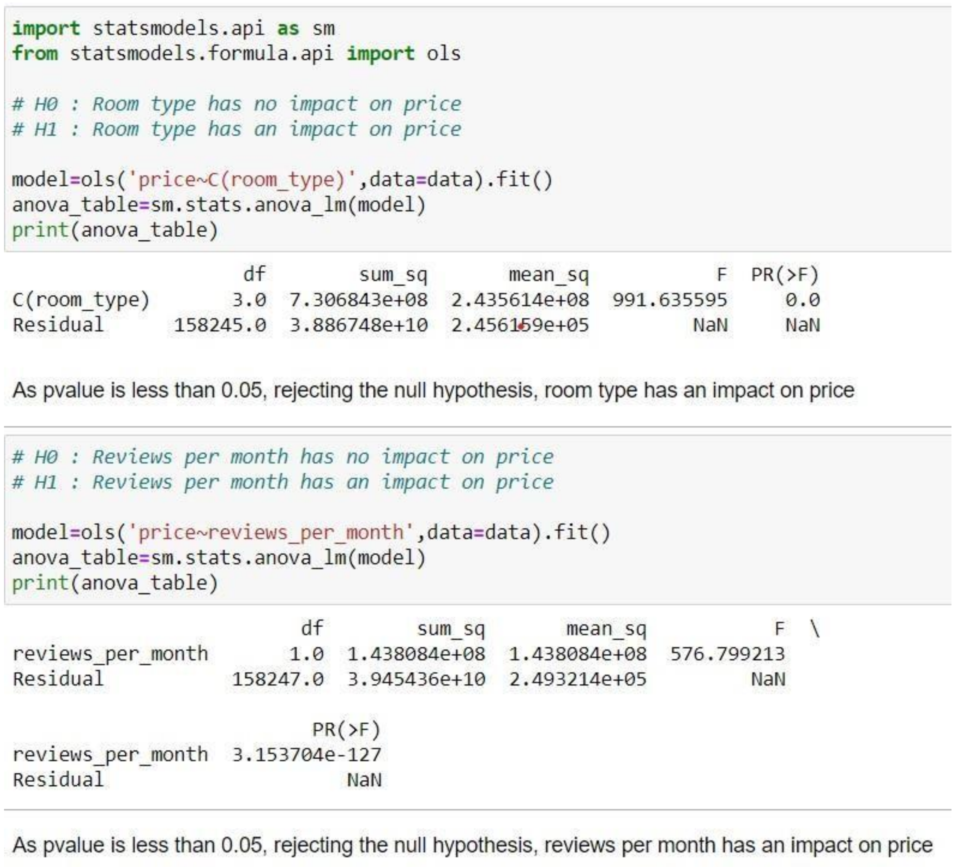
Insight: For almost all the states Most of the properties that can be booked instantly are available for a greater number of days in a year



Insight: Hotel rooms and Entire home/apt in most of the states tend to have a bit higher prices than other room types

Chapter 3: Statistical Analysis & Insights

With Target Variable Price:-



```
import statsmodels.api as sm
from statsmodels.formula.api import ols

# H0 : Room type has no impact on price
# H1 : Room type has an impact on price

model=ols('price~C(room_type)',data=data).fit()
anova_table=sm.stats.anova_lm(model)
print(anova_table)
```

```
                    df        sum_sq       mean_sq           F  PR(>F)
C(room_type)       3.0  7.306843e+08  2.435614e+08  991.635595     0.0
Residual      158245.0  3.886748e+10  2.456159e+05         NaN     NaN
```

As pvalue is less than 0.05, rejecting the null hypothesis, room type has an impact on price

```
# H0 : Reviews per month has no impact on price
# H1 : Reviews per month has an impact on price

model=ols('price~reviews_per_month',data=data).fit()
anova_table=sm.stats.anova_lm(model)
print(anova_table)
```

```
                         df        sum_sq       mean_sq           F  \
reviews_per_month       1.0  1.438084e+08  1.438084e+08  576.799213
Residual           158247.0  3.945436e+10  2.493214e+05         NaN

                          PR(>F)
reviews_per_month  3.153704e-127
Residual                     NaN
```

As pvalue is less than 0.05, rejecting the null hypothesis, reviews per month has an impact on price

```
# H0 : Number of reviews has no impact on price
# H1 : Number of reviews has an impact on price

model=ols('price~number_of_reviews',data=data).fit()
anova_table=sm.stats.anova_lm(model)
print(anova_table)
```

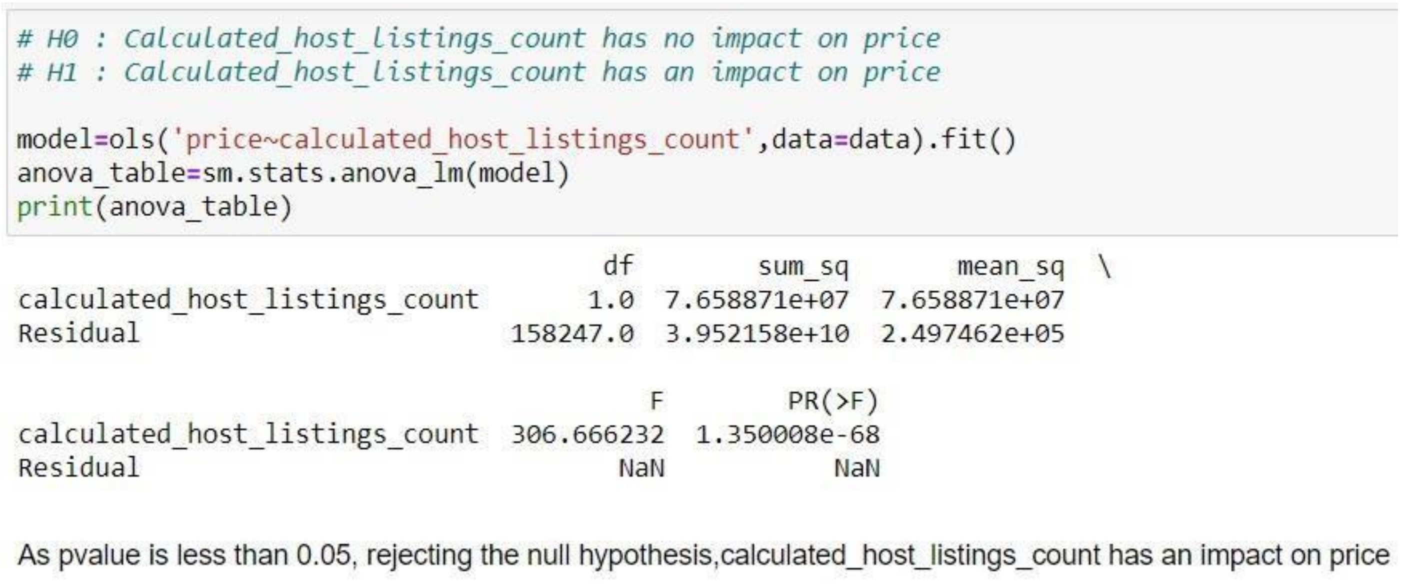
```
                         df        sum_sq       mean_sq           F  \
number_of_reviews       1.0  1.090283e+08  1.090283e+08  436.914921   
Residual           158247.0  3.948914e+10  2.495412e+05         NaN   

                         PR(>F)  
number_of_reviews  6.874075e-97  
Residual                    NaN  
```

As pvalue is less than 0.05, rejecting the null hypothesis, number of reviews has an impact on price

```
# H0 : Availability_365 has no impact on price
# H1 : Availability_365 has an impact on price

model=ols('price~availability_365',data=data).fit()
anova_table=sm.stats.anova_lm(model)
print(anova_table)
```

```
                        df        sum_sq       mean_sq           F  \
availability_365       1.0  4.783786e+07  4.783786e+07  191.406693   
Residual          158247.0  3.955033e+10  2.499278e+05         NaN   

                        PR(>F)  
availability_365  1.662127e-43  
Residual                   NaN  
```

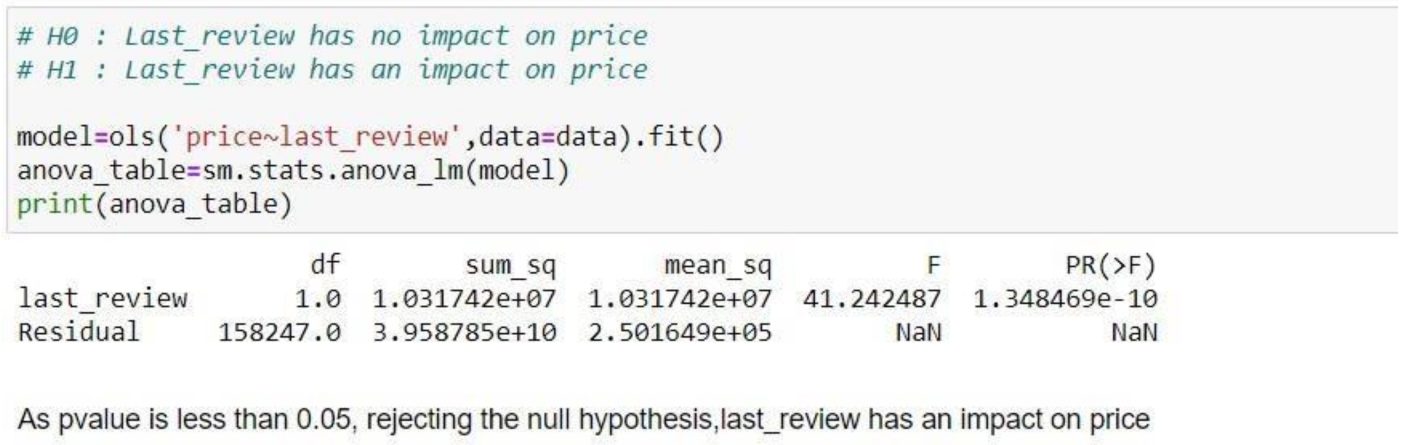
As pvalue is less than 0.05, rejecting the null hypothesis, availability_365 has an impact on price



```
# H0 : Calculated_host_listings_count has no impact on price
# H1 : Calculated_host_listings_count has an impact on price

model=ols('price~calculated_host_listings_count',data=data).fit()
anova_table=sm.stats.anova_lm(model)
print(anova_table)
```

```
                                      df        sum_sq       mean_sq  \
calculated_host_listings_count       1.0  7.658871e+07  7.658871e+07
Residual                        158247.0  3.952158e+10  2.497462e+05

                                         F        PR(>F)
calculated_host_listings_count  306.666232  1.350008e-68
Residual                               NaN           NaN
```

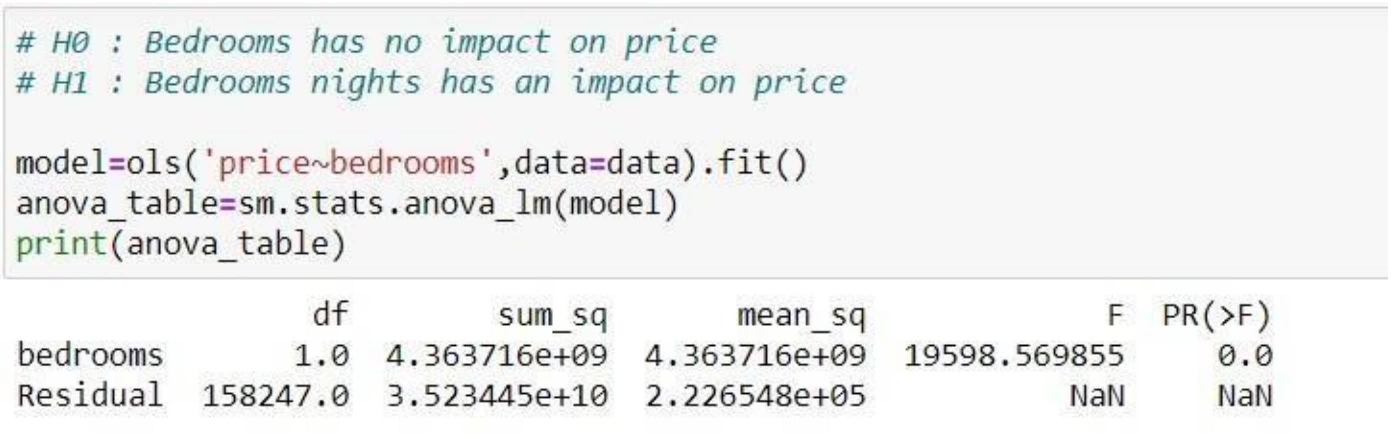
As pvalue is less than 0.05, rejecting the null hypothesis,calculated_host_listings_count has an impact on price



```
# H0 : Last_review has no impact on price
# H1 : Last_review has an impact on price

model=ols('price~last_review',data=data).fit()
anova_table=sm.stats.anova_lm(model)
print(anova_table)
```

```
                    df        sum_sq       mean_sq          F        PR(>F)
last_review        1.0  1.031742e+07  1.031742e+07  41.242487  1.348469e-10
Residual      158247.0  3.958785e+10  2.501649e+05        NaN           NaN
```

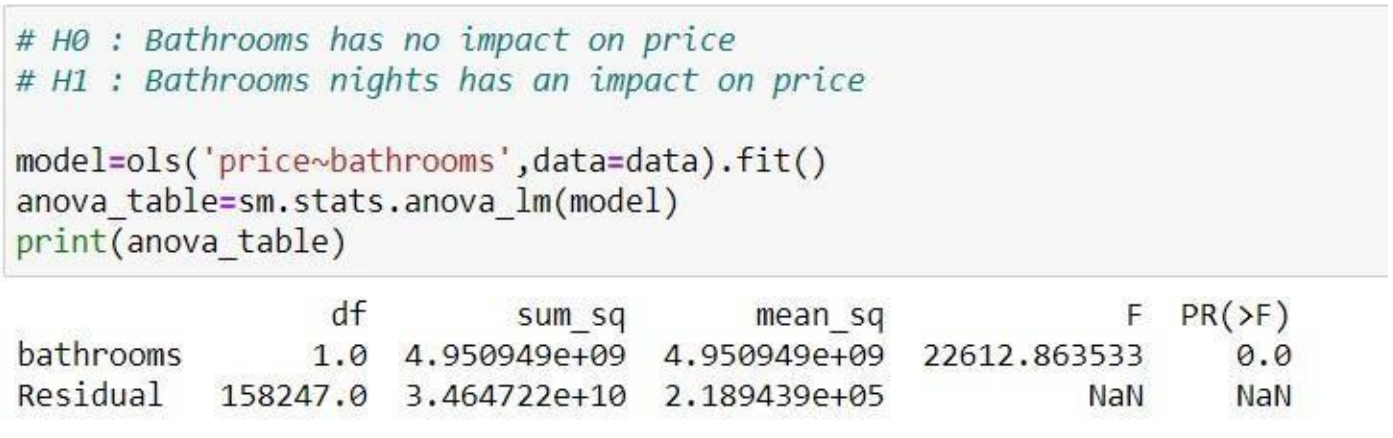
As pvalue is less than 0.05, rejecting the null hypothesis,last_review has an impact on price



```
# H0 : Bedrooms has no impact on price
# H1 : Bedrooms nights has an impact on price

model=ols('price~bedrooms',data=data).fit()
anova_table=sm.stats.anova_lm(model)
print(anova_table)
```

```
                df        sum_sq       mean_sq             F  PR(>F)
bedrooms       1.0  4.363716e+09  4.363716e+09  19598.569855     0.0
Residual  158247.0  3.523445e+10  2.226548e+05           NaN     NaN
```

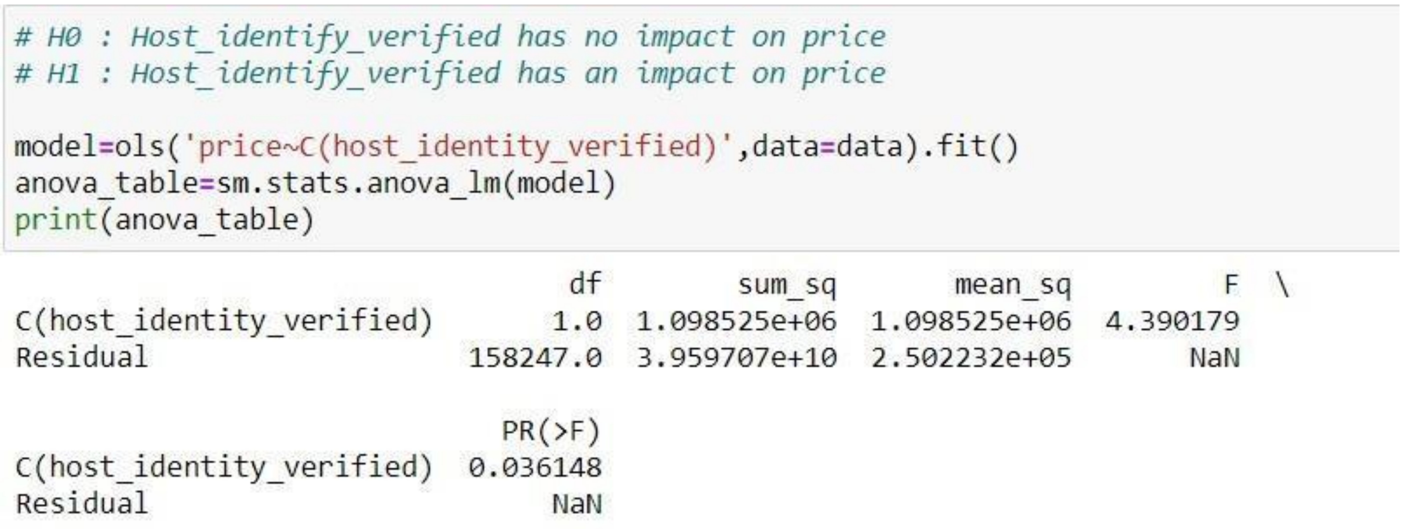
As pvalue is less than 0.05, rejecting the null hypothesis, bedrooms has an impact on price



```
# H0 : Bathrooms has no impact on price
# H1 : Bathrooms nights has an impact on price

model=ols('price~bathrooms',data=data).fit()
anova_table=sm.stats.anova_lm(model)
print(anova_table)
```

```
                 df        sum_sq       mean_sq             F  PR(>F)
bathrooms       1.0  4.950949e+09  4.950949e+09  22612.863533     0.0
Residual   158247.0  3.464722e+10  2.189439e+05           NaN     NaN
```

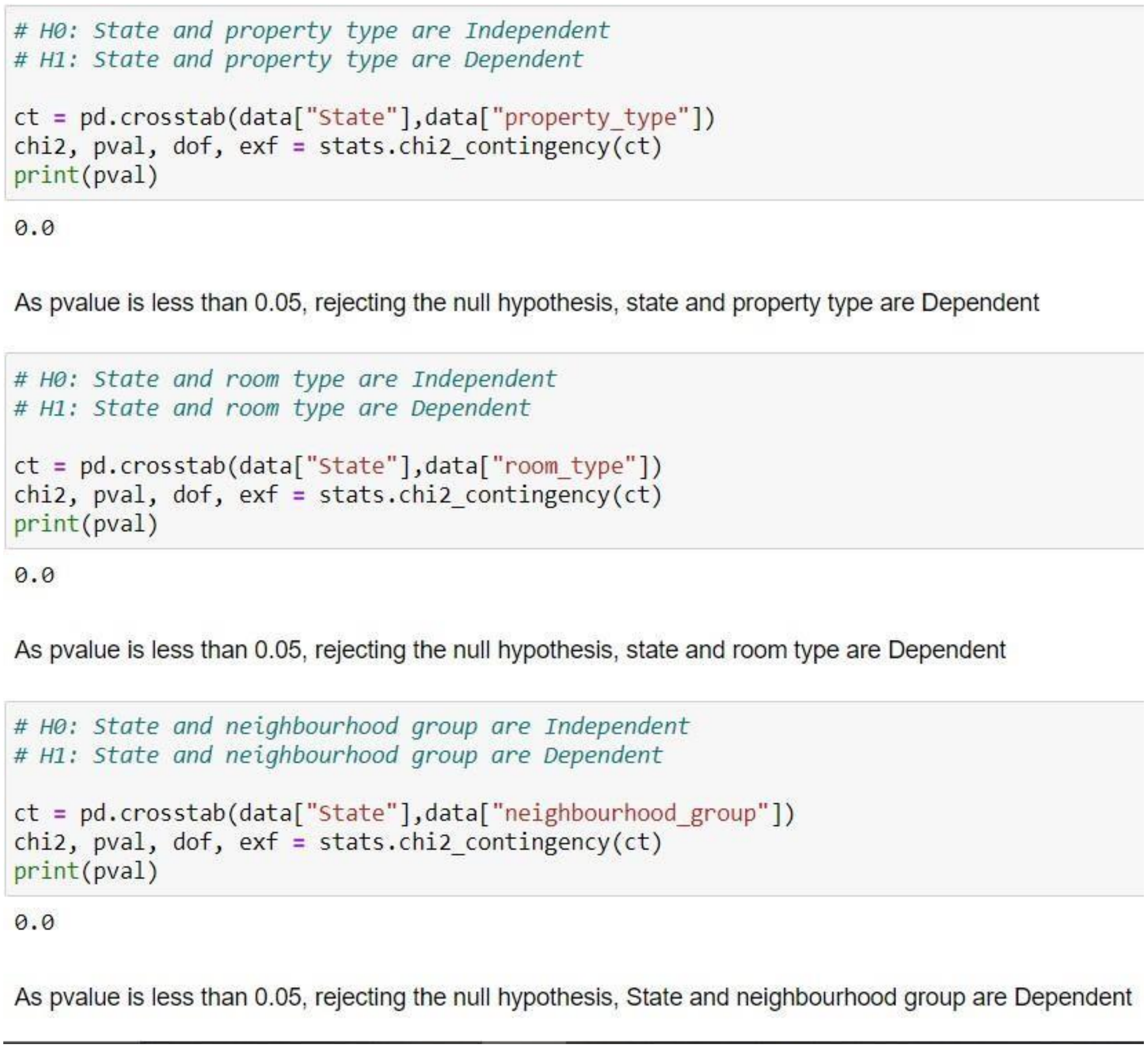
As pvalue is less than 0.05, rejecting the null hypothesis, bathrooms has an impact on price



```
# H0 : Host_identify_verified has no impact on price
# H1 : Host_identify_verified has an impact on price

model=ols('price~C(host_identity_verified)',data=data).fit()
anova_table=sm.stats.anova_lm(model)
print(anova_table)
```

```
                                df        sum_sq       mean_sq         F  \
C(host_identity_verified)      1.0  1.098525e+06  1.098525e+06  4.390179   
Residual                  158247.0  3.959707e+10  2.502232e+05       NaN   

                             PR(>F)  
C(host_identity_verified)  0.036148  
Residual                        NaN  
```

As pvalue is less than 0.05, rejecting the null hypothesis, host_identity_verified has an impact on price



```python
# H0: State and property type are Independent
# H1: State and property type are Dependent

ct = pd.crosstab(data["State"],data["property_type"])
chi2, pval, dof, exf = stats.chi2_contingency(ct)
print(pval)
```

```
0.0
```

As pvalue is less than 0.05, rejecting the null hypothesis, state and property type are Dependent

```python
# H0: State and room type are Independent
# H1: State and room type are Dependent

ct = pd.crosstab(data["State"],data["room_type"])
chi2, pval, dof, exf = stats.chi2_contingency(ct)
print(pval)
```

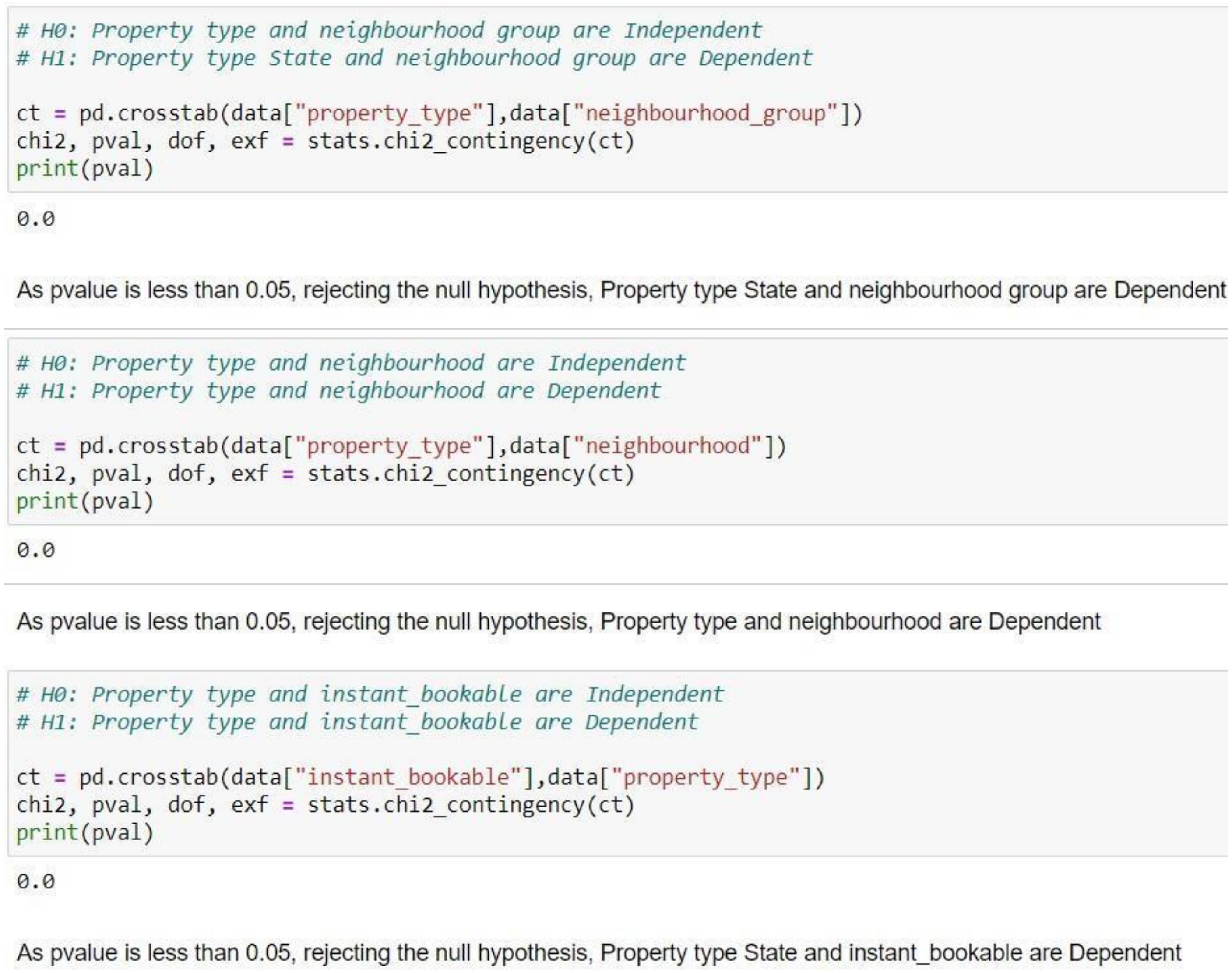
```
0.0
```

As pvalue is less than 0.05, rejecting the null hypothesis, state and room type are Dependent

```python
# H0: State and neighbourhood group are Independent
# H1: State and neighbourhood group are Dependent

ct = pd.crosstab(data["State"],data["neighbourhood_group"])
chi2, pval, dof, exf = stats.chi2_contingency(ct)
print(pval)
```

```
0.0
```

As pvalue is less than 0.05, rejecting the null hypothesis, State and neighbourhood group are Dependent



```python
# H0: Property type and neighbourhood group are Independent
# H1: Property type State and neighbourhood group are Dependent

ct = pd.crosstab(data["property_type"],data["neighbourhood_group"])
chi2, pval, dof, exf = stats.chi2_contingency(ct)
print(pval)
```

```
0.0
```

As pvalue is less than 0.05, rejecting the null hypothesis, Property type State and neighbourhood group are Dependent

```python
# H0: Property type and neighbourhood are Independent
# H1: Property type and neighbourhood are Dependent

ct = pd.crosstab(data["property_type"],data["neighbourhood"])
chi2, pval, dof, exf = stats.chi2_contingency(ct)
print(pval)
```

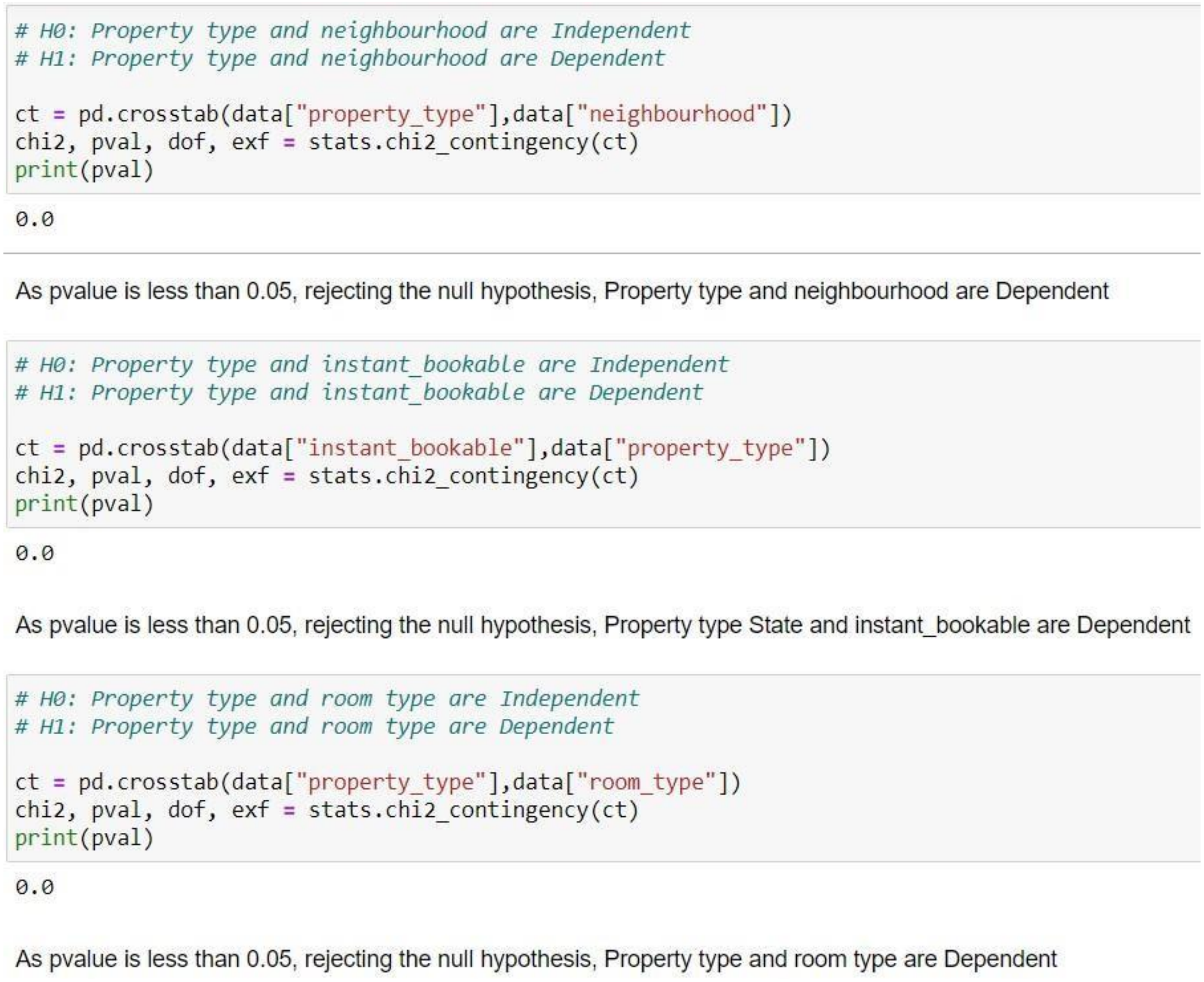
```
0.0
```

As pvalue is less than 0.05, rejecting the null hypothesis, Property type and neighbourhood are Dependent

```python
# H0: Property type and instant_bookable are Independent
# H1: Property type and instant_bookable are Dependent

ct = pd.crosstab(data["instant_bookable"],data["property_type"])
chi2, pval, dof, exf = stats.chi2_contingency(ct)
print(pval)
```

```
0.0
```

As pvalue is less than 0.05, rejecting the null hypothesis, Property type State and instant_bookable are Dependent



```python
# H0: Property type and neighbourhood are Independent
# H1: Property type and neighbourhood are Dependent

ct = pd.crosstab(data["property_type"],data["neighbourhood"])
chi2, pval, dof, exf = stats.chi2_contingency(ct)
print(pval)
```

```
0.0
```

As pvalue is less than 0.05, rejecting the null hypothesis, Property type and neighbourhood are Dependent

```python
# H0: Property type and instant_bookable are Independent
# H1: Property type and instant_bookable are Dependent

ct = pd.crosstab(data["instant_bookable"],data["property_type"])
chi2, pval, dof, exf = stats.chi2_contingency(ct)
print(pval)
```

```
0.0
```

As pvalue is less than 0.05, rejecting the null hypothesis, Property type State and instant_bookable are Dependent

```python
# H0: Property type and room type are Independent
# H1: Property type and room type are Dependent

ct = pd.crosstab(data["property_type"],data["room_type"])
chi2, pval, dof, exf = stats.chi2_contingency(ct)
print(pval)
```

```
0.0
```

As pvalue is less than 0.05, rejecting the null hypothesis, Property type and room type are Dependent

Based on the above graphs and statistical tests which were done for all the necessary columns, we can draw the following few inferences: -

- Most properties are located in Los Angeles.
- Bronx, Staten Island, Santa Cruz have minimum number of properties
- Most people offer entire apartments and very few hosts offer private rooms in bed and breakfasts.
- California has the highest number of listings while Washington has the lowest amongst all popular states.
- Verification has been done for most of the listings and there are more super-hosts than normal hosts.
- Those who pay less reviewed more and those who pay more reviewed less.
- Entire homes/apartments have a higher price range.

- The state of New York has less availability of listings in a year.
- Listings in Los Angeles are on the more expensive side compared to listings in the state of Staten Island
- Private rooms have less availability within a year as compared to entire apartments, hotels and shared rooms.
- Listings in New York are less available within a year and listings in Florida and Hawaii have more availability.
- Average availability in a year is almost the same even if the host is a super host or not.
- Average availability is almost the same even if the host's identity is verified or not.
- Hawaii is the state which has the highest count of host listings.
- Tennesse is the state which has listings with highest number of beds.
- Listings that can be booked instantly have a higher price.
- Hotel rooms tend to have a bit higher prices than other room types.
- The Entire home/apt room type is reviewed more and also has a higher price range as compared to shared rooms, private rooms and hotel room types.
- Listings whose host's identity is verified, have more number of reviews and a higher price range.
- The Entire home/apt room type has a greater number of reviews in most of the states.
- For almost all the states most of the properties that can be booked instantly are available for a greater number of days in a year.
- Hotel rooms and Entire home/apt in most of the states tend to have a bit higher prices than other room types.
- Room type, reviews per month, reviews per month, availability_365, last review, maximum nights, minimum nights, no. Of bedrooms, bathrooms and beds, whether the host is a superhost or not, whether the host’s ID has been verified or not, whether the listing is instantly bookable or not, state and neighborhood group, all have a noticeable impact on the price for a listing.
- Host_is_superhost and host_identity_verified are dependent.
- State and property type are dependent.
- State and room type are dependent.
- State and neighborhood group are dependent.
- State and neighborhood are dependent.
- Property Type and whether the listing is instant_bookable are dependent.
- Property type and neighborhood groups are dependent.
- Neighborhood group and neighborhood are dependent.

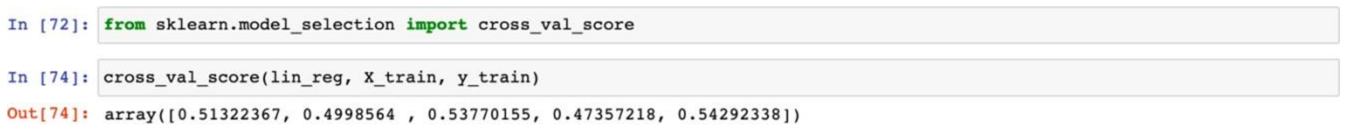
Chapter 4 : Evaluation Techniques & Metrics:

Evaluation Techniques

While training a model is a key step, how the model generalizes on unseen data is an equally important aspect that should be considered in every machine learning pipeline. We need to know whether it actually works and, consequently, if we can trust its predictions.

Cross-validation:

Cross-validation is a technique that involves partitioning the original observation dataset into a training set, used to train the model, and an independent set used to evaluate the analysis.The most common cross-validation technique is k-fold cross-validation.

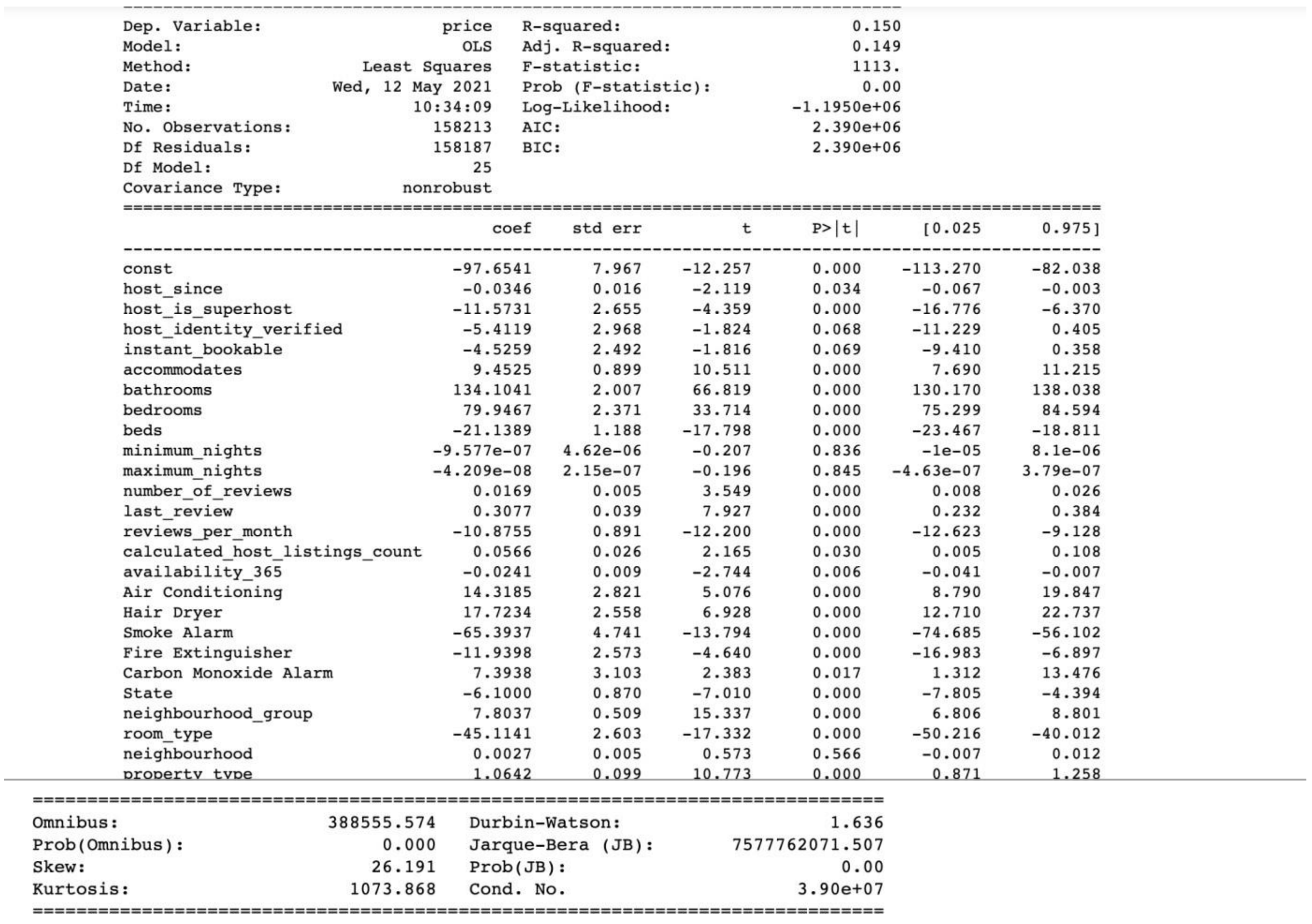


```
In [72]: from sklearn.model_selection import cross_val_score

In [74]: cross_val_score(lin_reg, X_train, y_train)
Out[74]: array([0.51322367, 0.4998564 , 0.53770155, 0.47357218, 0.54292338])
```

Evaluation Metrics

1. RMSE (Root Mean Square Error) : It represents the sample standard deviation of the differences between predicted values and observed values (called residuals).
2. Adjusted R^2 : Just like R^2, adjusted R^2 also shows how well terms fit a curve or line but adjusts for the number of terms in a model.

Chapter 5 : Base Model Building: -



```
==============================================================================
Dep. Variable:                  price   R-squared:                       0.150
Model:                            OLS   Adj. R-squared:                  0.149
Method:                 Least Squares   F-statistic:                     1113.
Date:                Wed, 12 May 2021   Prob (F-statistic):               0.00
Time:                        10:34:09   Log-Likelihood:            -1.1950e+06
No. Observations:              158213   AIC:                         2.390e+06
Df Residuals:                  158187   BIC:                         2.390e+06
Df Model:                          25
Covariance Type:            nonrobust
==================================================================================================
                                     coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------------
const                            -97.6541      7.967    -12.257      0.000    -113.270     -82.038
host_since                        -0.0346      0.016     -2.119      0.034      -0.067      -0.003
host_is_superhost                -11.5731      2.655     -4.359      0.000     -16.776      -6.370
host_identity_verified            -5.4119      2.968     -1.824      0.068     -11.229       0.405
instant_bookable                  -4.5259      2.492     -1.816      0.069      -9.410       0.358
accommodates                       9.4525      0.899     10.511      0.000       7.690      11.215
bathrooms                        134.1041      2.007     66.819      0.000     130.170     138.038
bedrooms                          79.9467      2.371     33.714      0.000      75.299      84.594
beds                             -21.1389      1.188    -17.798      0.000     -23.467     -18.811
minimum_nights                 -9.577e-07   4.62e-06     -0.207      0.836       -1e-05     8.1e-06
maximum_nights                 -4.209e-08   2.15e-07     -0.196      0.845   -4.63e-07    3.79e-07
number_of_reviews                  0.0169      0.005      3.549      0.000       0.008       0.026
last_review                        0.3077      0.039      7.927      0.000       0.232       0.384
reviews_per_month                -10.8755      0.891    -12.200      0.000     -12.623      -9.128
calculated_host_listings_count     0.0566      0.026      2.165      0.030       0.005       0.108
availability_365                  -0.0241      0.009     -2.744      0.006      -0.041      -0.007
Air Conditioning                  14.3185      2.821      5.076      0.000       8.790      19.847
Hair Dryer                        17.7234      2.558      6.928      0.000      12.710      22.737
Smoke Alarm                      -65.3937      4.741    -13.794      0.000     -74.685     -56.102
Fire Extinguisher                -11.9398      2.573     -4.640      0.000     -16.983      -6.897
Carbon Monoxide Alarm              7.3938      3.103      2.383      0.017       1.312      13.476
State                             -6.1000      0.870     -7.010      0.000      -7.805      -4.394
neighbourhood_group                7.8037      0.509     15.337      0.000       6.806       8.801
room_type                        -45.1141      2.603    -17.332      0.000     -50.216     -40.012
neighbourhood                      0.0027      0.005      0.573      0.566      -0.007       0.012
property_type                      1.0642      0.099     10.773      0.000       0.871       1.258
==============================================================================
Omnibus:                   388555.574   Durbin-Watson:                   1.636
Prob(Omnibus):                  0.000   Jarque-Bera (JB):       7577762071.507
Skew:                          26.191   Prob(JB):                         0.00
Kurtosis:                    1073.868   Cond. No.                     3.90e+07
==============================================================================
```

As we can see in the above summary graph, R2 is 0.150 . Seeing the Durbin-Watson value which is close to 2 we can conclude that there is no auto-corelation present.

As we can see that the probability of F statistic(0.00) is less than 0.05 so we can conclude that our model is significant.

Seeing the p values of different features we know about the significant features.

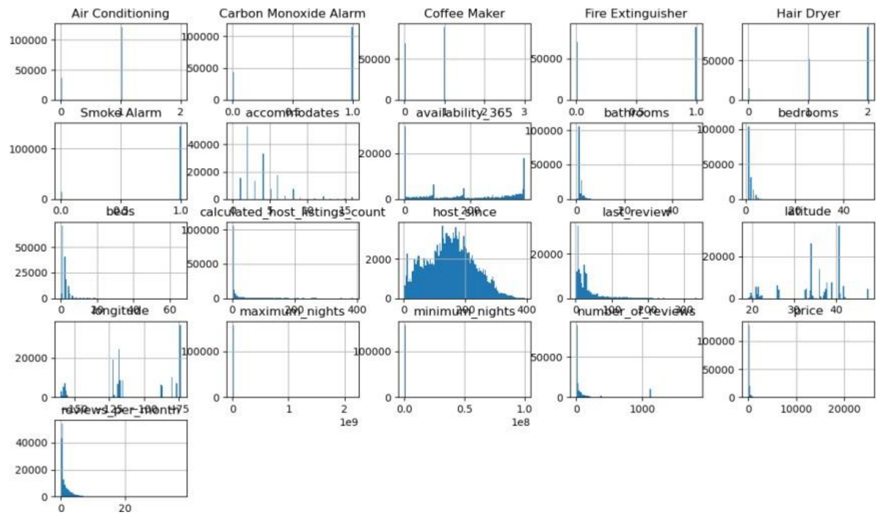
Chapter 6 : Feature Engineering

Feature engineering is the process of using domain knowledge of the data to create features that makes more sense and easier for a machine learning algorithms This input data comprise features, which are usually in the form of structured columns. Algorithms require features with some specific characteristics to work properly .According to a survey in Forbes, data scientists spend 80% of their time on data preparation:

According to Luca Massaron , a data scientist and marketing research director for packt publication ,

"The features you use influence more than anything else the result. No algorithm alone, to my knowledge, can supplement the information gain given by correct feature engineering." – Luca Massaron

Variables distribution before feature engineering:



We applied feature engineering on :

Numerical Variables:

1. host_since: Transformed using log transformation
2. minimum_nights: Transformed using log transformation
3. maximum_nights: Transformed using log transformation
4. number_of_reviews: Transformed using log transformation
5. reviews_per_month: Transformed using log transformation
6. last_review: Transformed using log transformation
7. calculated_host_listings_count: Transformed using log transformation

Categorical Variables:

1) State: Ordinal Encoding based on the aggregate price of the listing.

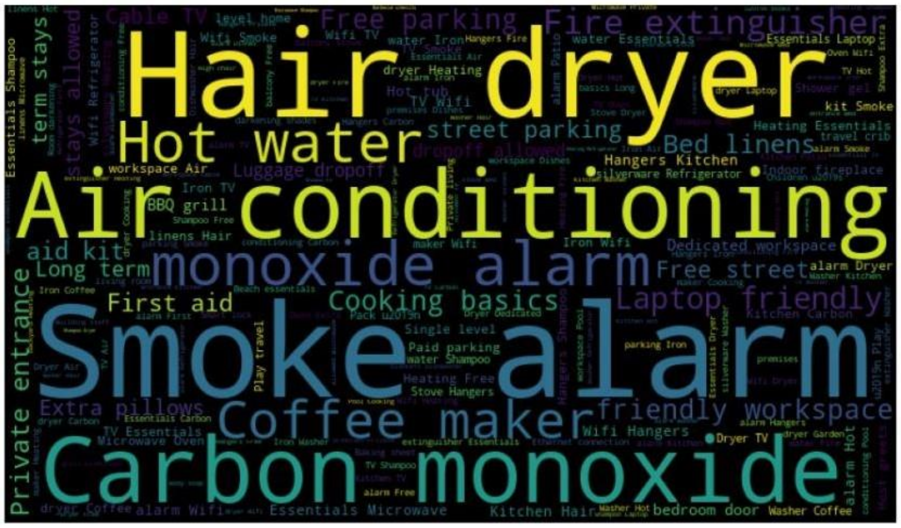
2) Room_type: Ordinal Encoding based on the aggregate price of the listing.

3) neighbourhood_group: Ordinal Encoding based on the aggregate price of the listing.

- After performing feature engineering there is a significant decrease in the skewness and kurtosis of the variables which leads to increase in our accuracy from 0.151 to 0.50.

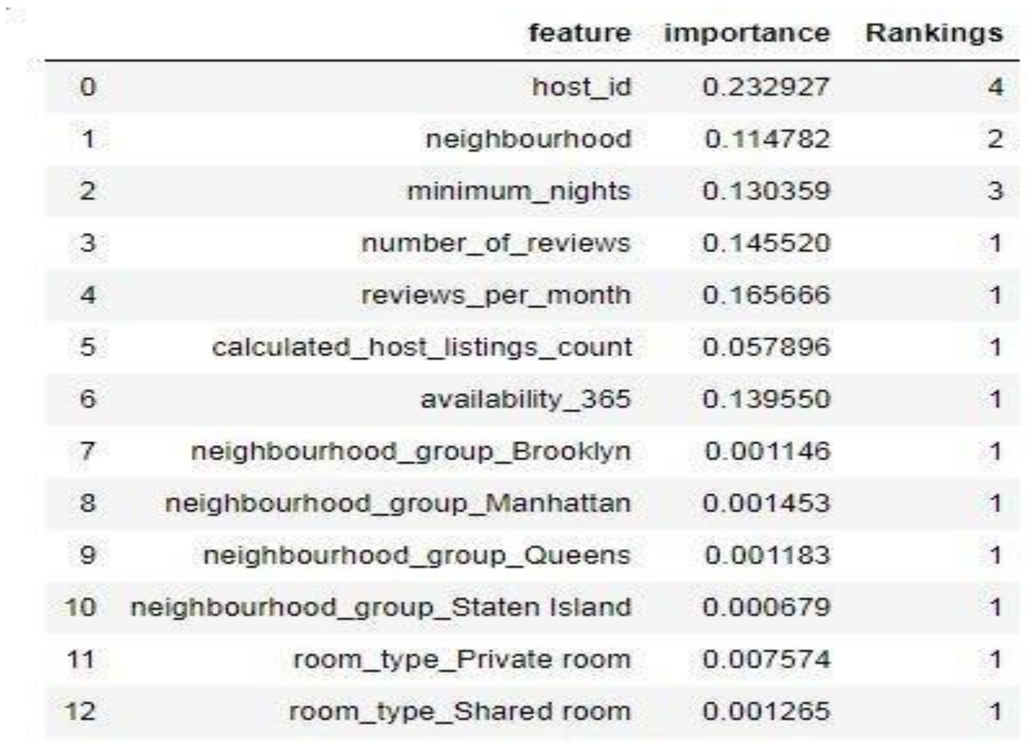
- With the help of feature engineering we attained the normal distribution for most of the features.

For amenities column , we created a word cloud and found out which are the top 5 commonly used amenities in the listings:



Using the top 5 amenities we created separate features:
1)Smoke Alarm
2) Air Conditioning
3) Hair dryer
4) Fire Extinguisher
5) Coffee Maker

Feature ranking:-



	feature	importance	Rankings
0	host_id	0.232927	4
1	neighbourhood	0.114782	2
2	minimum_nights	0.130359	3
3	number_of_reviews	0.145520	1
4	reviews_per_month	0.165666	1
5	calculated_host_listings_count	0.057896	1
6	availability_365	0.139550	1
7	neighbourhood_group_Brooklyn	0.001146	1
8	neighbourhood_group_Manhattan	0.001453	1
9	neighbourhood_group_Queens	0.001183	1
10	neighbourhood_group_Staten Island	0.000679	1
11	room_type_Private room	0.007574	1
12	room_type_Shared room	0.001265	1

Rfe_score : Feature ranking with recursive feature elimination.

Given an external estimator that assigns weights to features (e.g., the coefficients of a linear model), the goal of recursive feature elimination (RFE) is to select features by recursively considering smaller and smaller sets of features.

Importance: An extra-trees regressor.

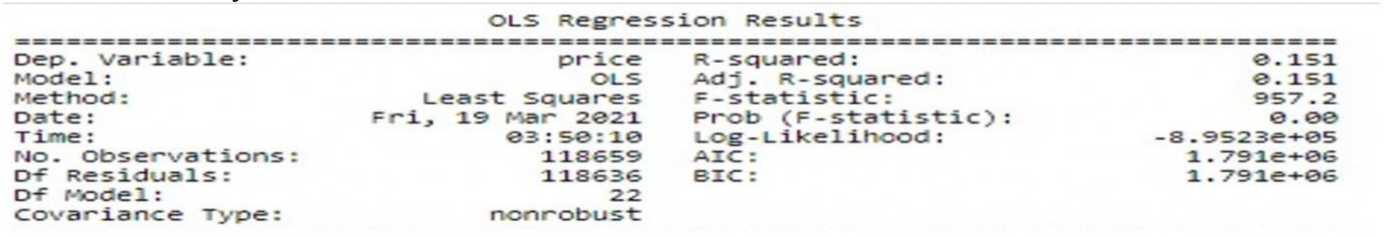
This class implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improvethe predictive accuracy and control over-fitting.

Chapter 7 : Assumptions for modelling

Model : A simple linear regressor

OLS Model (Ordinary Least Square)

Model Summary :



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.151
Model:                            OLS   Adj. R-squared:                  0.151
Method:                 Least Squares   F-statistic:                     957.2
Date:                Fri, 19 Mar 2021   Prob (F-statistic):               0.00
Time:                        03:50:10   Log-Likelihood:            -8.9523e+05
No. Observations:              118659   AIC:                         1.791e+06
Df Residuals:                  118636   BIC:                         1.791e+06
Df Model:                          22
Covariance Type:            nonrobust
```

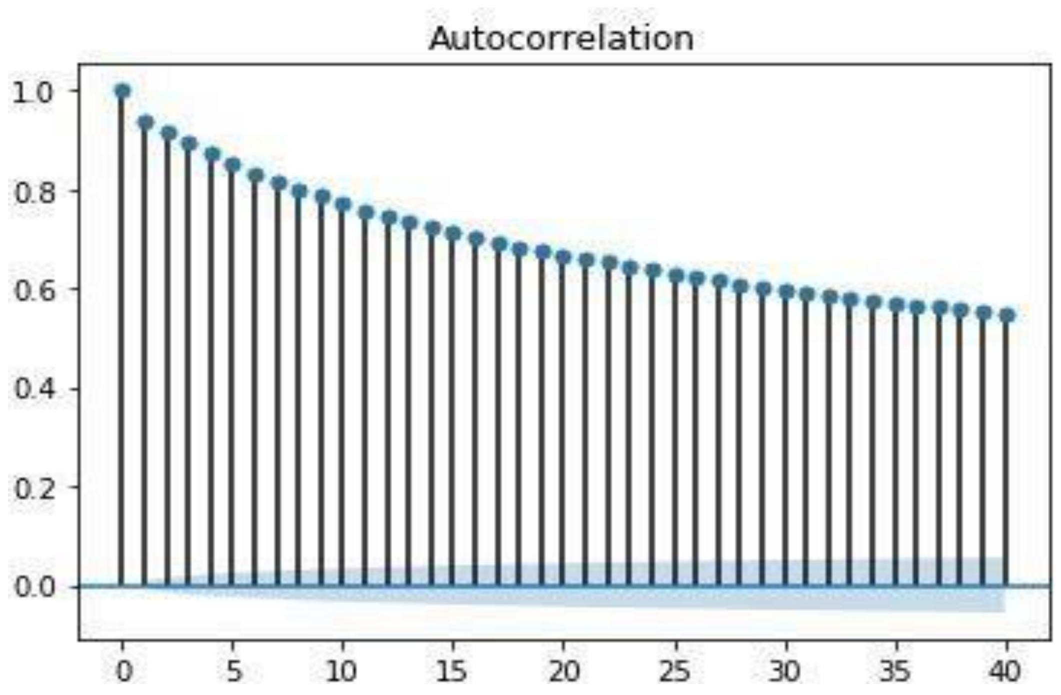
1. Method : Least squares : sum of squares of residues
2. DF Model : Degree of freedom (number of features -1)
3. Covariance type : Standard Errors assume that the covariance matrix of the errors is correctly specified.
4. R-squared: proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model.
5. Adjusted R-squared : The adjusted R-squared is a modified version of R-squared that has been adjusted for the number of predictors in the model. The adjusted R-squared increases only if the new term improves the model more than would be expected by chance.
6. F-statistic and prob F-statistic: The F value is the ratio of the mean regression sum of squares divided by the mean error sum of squares. Its value will range from zero to an arbitrarily large number. The value of Prob(F) is the probability that the null hypothesis for the full model is true. Our model failed!!

Assumptions:

Assumption : 1

Linear regression analysis requires that there is little or no autocorrelation in the data. Autocorrelation occurs when the residuals are not independent from each other. In other words when the value of y(x+1) is not independent from the value of y(x).

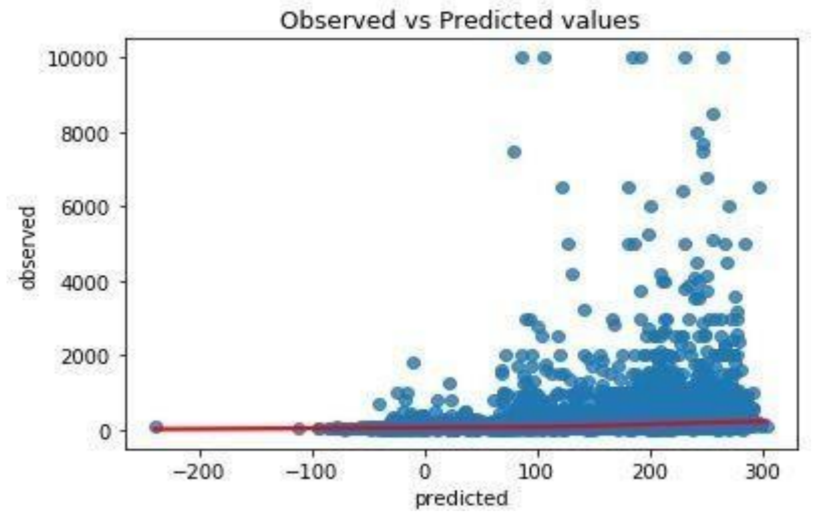
While a scatterplot allows you to check for autocorrelation, you can test the linear regression model for autocorrelation with the Durbin-Watson test. Durbin-Watson's d tests the null hypothesis that the residuals are not linearly auto-correlated. While d can assume values between 0 and 4, values around 2 indicate no autocorrelation. As a rule of thumb values of $1.5 < d < 2.5$ show that there is no auto-correlation in the data. However, the Durbin-Watson test only analyses linear autocorrelation and only between direct neighbours,which are first order effect.

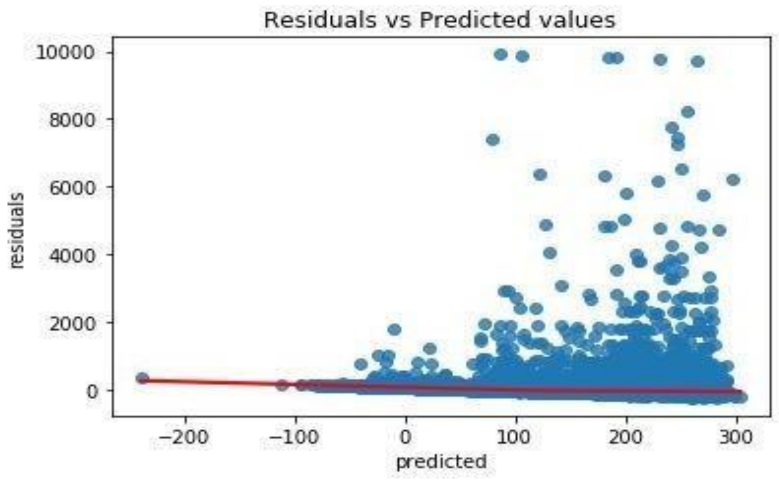


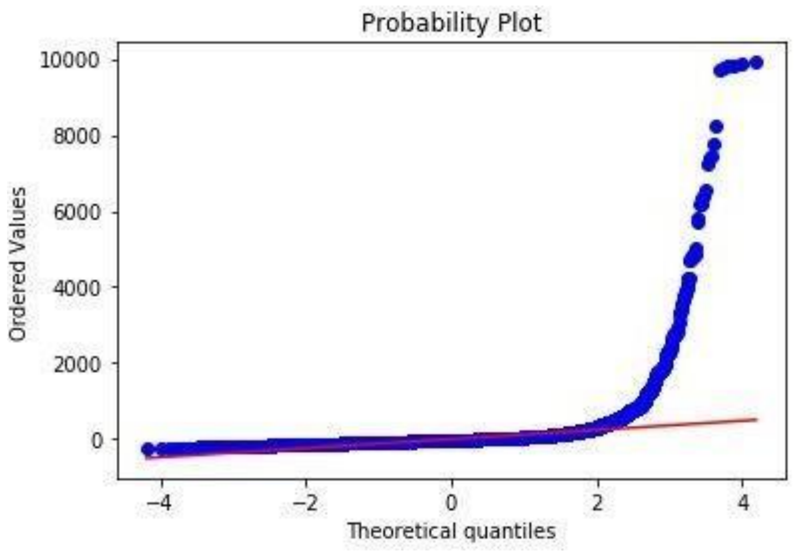
Highly positive autocorr. Durbin watson : 0.092

Assumption : 2

Linear regression needs the relationship between the independent and dependent variables to be linear. It is also important to check for outliers since linear regression is sensitive to outlier effects. The linearity assumption can best be tested with scatter plots,



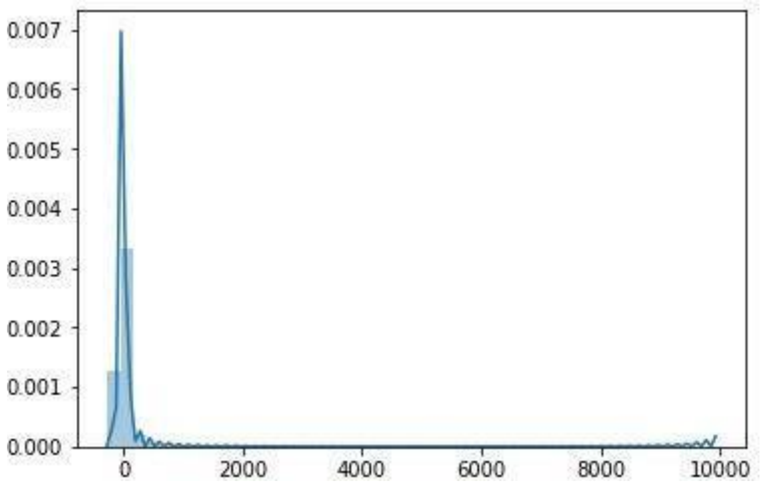




The data is highly non-linear

Assumption : 3

The linear regression analysis requires all variables to be multivariate normal. This assumption can best be checked with a histogram or a Q-Q-Plot. Normality can be checked with a goodness of fit test, e.g., the Kolmogorov-Smirnov test. When the data is not normally distributed a non-linear transformation (e.g., log-transformation) might fix this issue.



The data is not normal.

Assumption 4:

The assumption of the linear regression analysis is homoscedasticity. The scatter plot is good way to check whether the data are homoscedastic (meaning the residuals are equal across the regression line).

The Goldfeld-Quandt Test can also be used to test for heteroscedasticity. The test splits the data into two groups and tests to see if the variances of the residuals are similar across the groups. If homoscedasticity is present, a non-linear correction might fix the problem.

The F-statistic from Goldfeld-Quandt Test is 277.159 and p-value is 0.00 which makesthe data heteroscedastic

Assumption 5:

linear regression assumes that there is little or no multicollinearity in the data. Multicollinearity occurs when the independent variables are too highly correlated with each other.

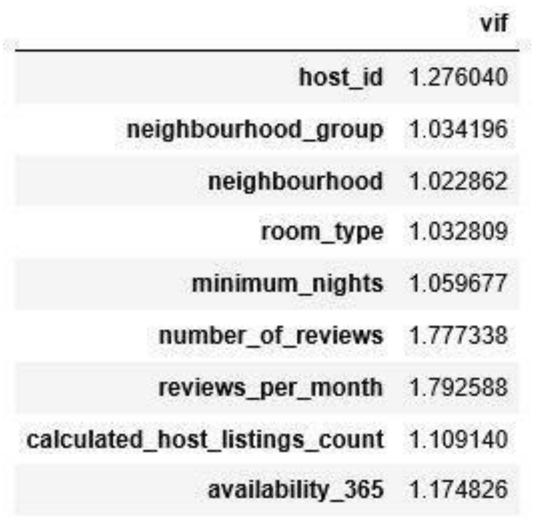
Multicollinearity may be tested with three central criteria:

1. Correlation matrix – when computing the matrix of Pearson's Bivariate Correlation among all independent variables the correlation coefficients need to be smaller than 1.

2. Tolerance – the tolerance measures the influence of one independent variable on all other independent variables; the tolerance is calculated with an initial linear regression analysis. Tolerance is defined as $T = 1 – R^2$ for these first step regression analysis. With $T < 0.1$ there might be multicollinearity in the data and with $T < 0.01$ there certainly is.

3. Variance Inflation Factor (VIF) – the variance inflation factor of the linear regression is defined as VIF = 1/T. With VIF > 5 there is an indication that multicollinearity may be present; with VIF > 10 there is certainly multicollinearity among the variables.

If multicollinearity is found in the data, centering the data (that is deducting the mean of the variable from each score) might help to solve the problem. However, the simplest way to address the problem is to remove independent variables with high VIF values.



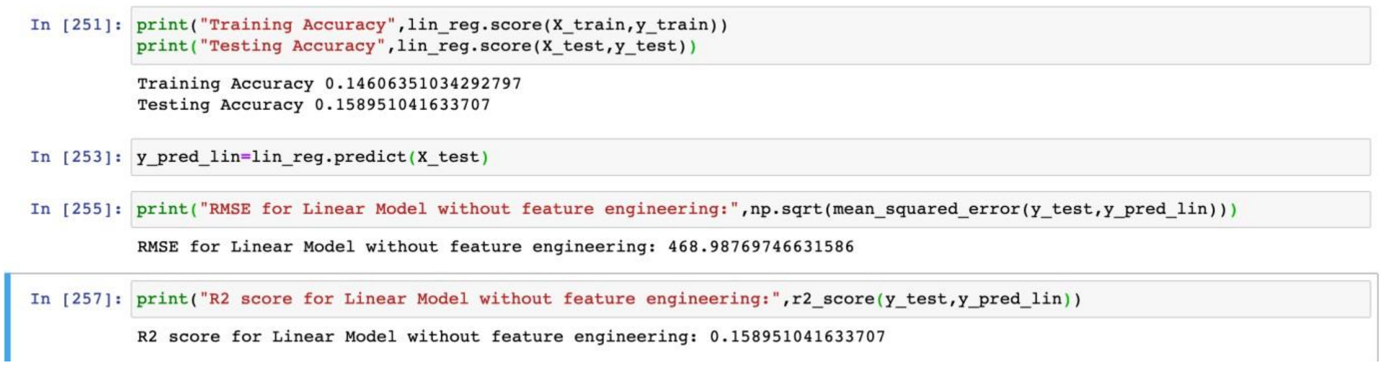
	vif
host_id	1.276040
neighbourhood_group	1.034196
neighbourhood	1.022862
room_type	1.032809
minimum_nights	1.059677
number_of_reviews	1.777338
reviews_per_month	1.792588
calculated_host_listings_count	1.109140
availability_365	1.174826

All the vif scores represent that there is no multicollinearity among the features.

Chapter 8 : Different Models Implementation

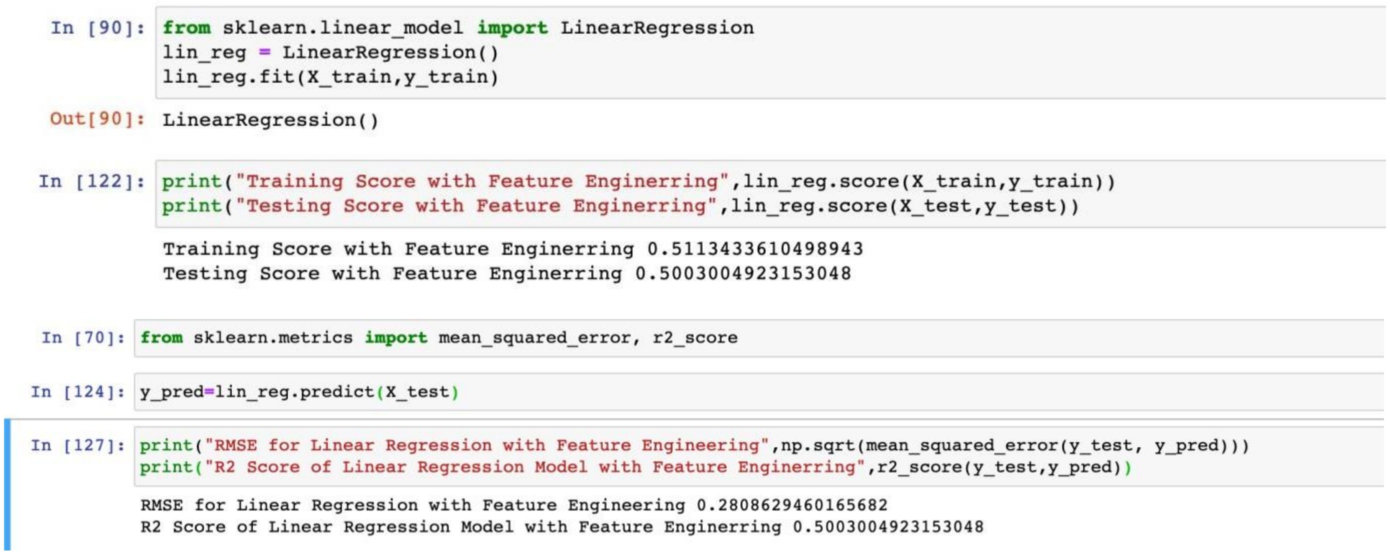
MODELS:

1) Linear Regression without Feature Engineering:



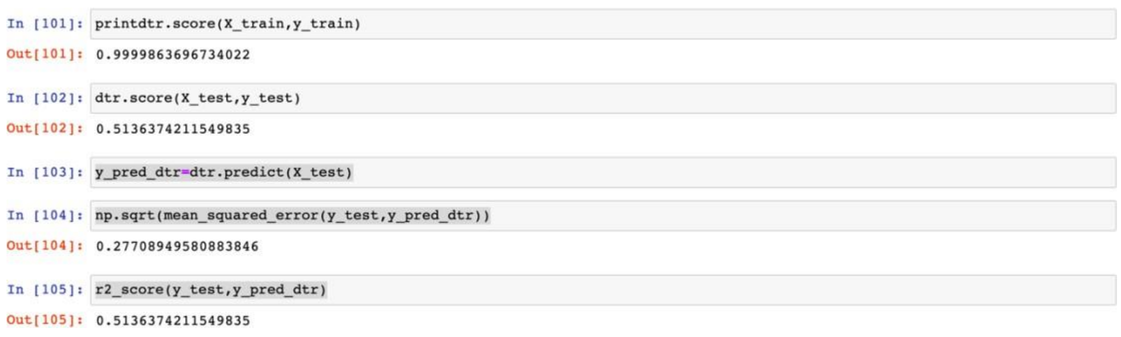
```
In [251]: print("Training Accuracy",lin_reg.score(X_train,y_train))
          print("Testing Accuracy",lin_reg.score(X_test,y_test))

          Training Accuracy 0.14606351034292797
          Testing Accuracy 0.158951041633707

In [253]: y_pred_lin=lin_reg.predict(X_test)

In [255]: print("RMSE for Linear Model without feature engineering:",np.sqrt(mean_squared_error(y_test,y_pred_lin)))

          RMSE for Linear Model without feature engineering: 468.98769746631586

In [257]: print("R2 score for Linear Model without feature engineering:",r2_score(y_test,y_pred_lin))

          R2 score for Linear Model without feature engineering: 0.158951041633707
```

2) Linear Regression with Feature Engineering:



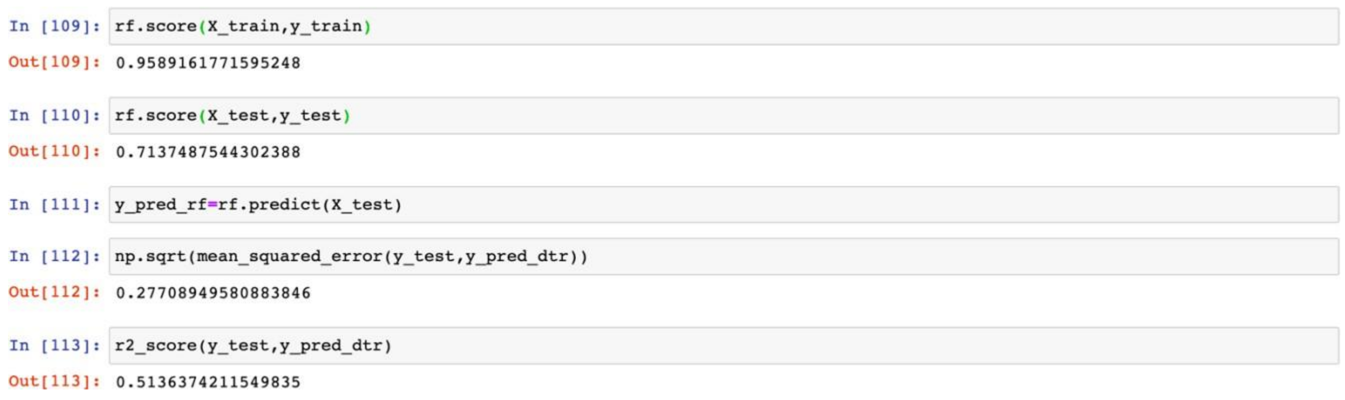
```
In [90]: from sklearn.linear_model import LinearRegression
         lin_reg = LinearRegression()
         lin_reg.fit(X_train,y_train)

Out[90]: LinearRegression()

In [122]: print("Training Score with Feature Enginerring",lin_reg.score(X_train,y_train))
          print("Testing Score with Feature Enginerring",lin_reg.score(X_test,y_test))

          Training Score with Feature Enginerring 0.5113433610498943
          Testing Score with Feature Enginerring 0.5003004923153048

In [70]: from sklearn.metrics import mean_squared_error, r2_score

In [124]: y_pred=lin_reg.predict(X_test)

In [127]: print("RMSE for Linear Regression with Feature Engineering",np.sqrt(mean_squared_error(y_test, y_pred)))
          print("R2 Score of Linear Regression Model with Feature Enginerring",r2_score(y_test,y_pred))

          RMSE for Linear Regression with Feature Engineering 0.2808629460165682
          R2 Score of Linear Regression Model with Feature Enginerring 0.5003004923153048
```

3)Decision Tree Regressor:



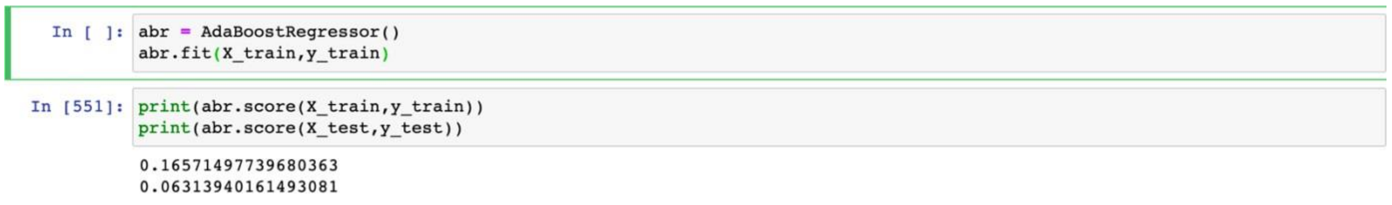
```
In [101]: printdtr.score(X_train,y_train)
Out[101]: 0.9999863696734022

In [102]: dtr.score(X_test,y_test)
Out[102]: 0.5136374211549835

In [103]: y_pred_dtr=dtr.predict(X_test)

In [104]: np.sqrt(mean_squared_error(y_test,y_pred_dtr))
Out[104]: 0.27708949580883846

In [105]: r2_score(y_test,y_pred_dtr)
Out[105]: 0.5136374211549835
```

4)Random Forest Regerssor



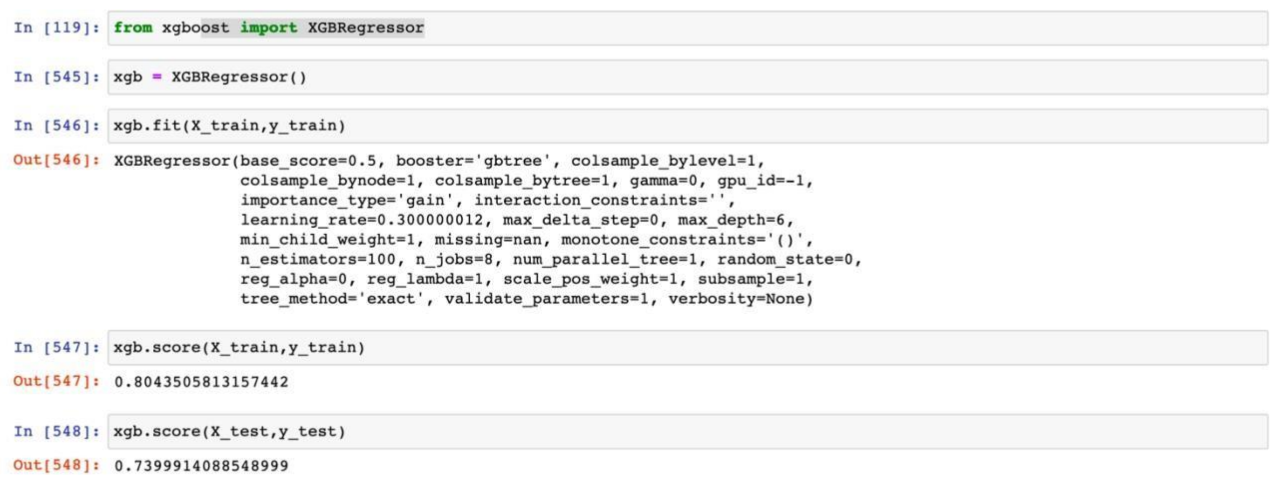
```
In [109]: rf.score(X_train,y_train)
Out[109]: 0.9589161771595248

In [110]: rf.score(X_test,y_test)
Out[110]: 0.7137487544302388

In [111]: y_pred_rf=rf.predict(X_test)

In [112]: np.sqrt(mean_squared_error(y_test,y_pred_dtr))
Out[112]: 0.27708949580883846

In [113]: r2_score(y_test,y_pred_dtr)
Out[113]: 0.5136374211549835
```

5)Ada Boost Regressor



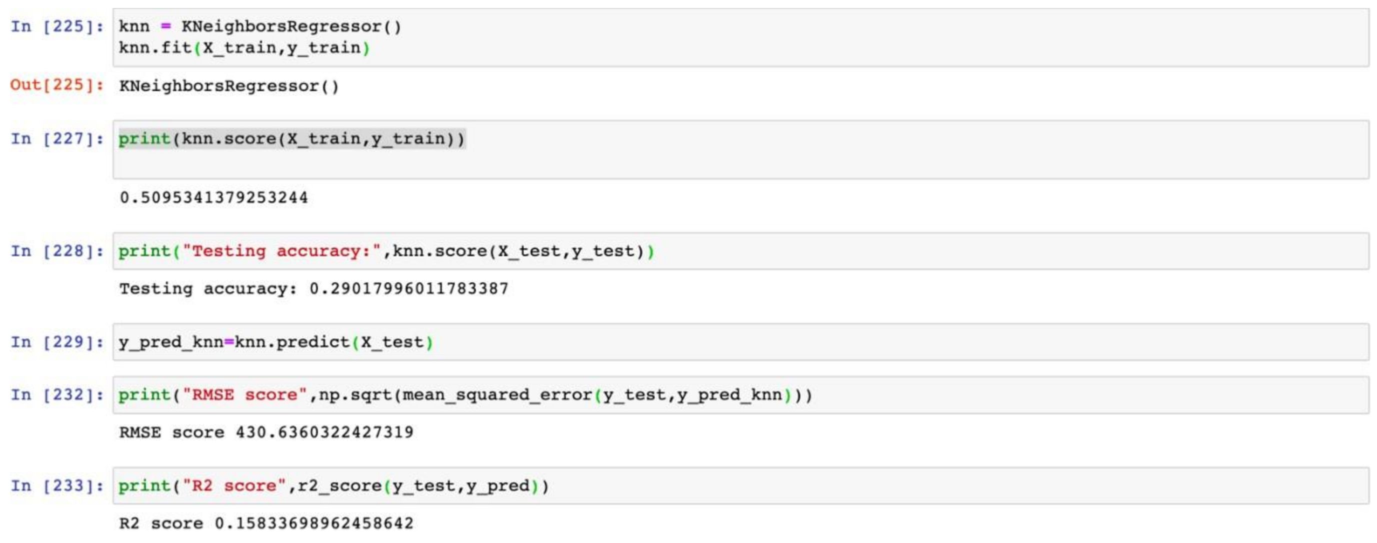
```
In [ ]: abr = AdaBoostRegressor()
        abr.fit(X_train,y_train)

In [551]: print(abr.score(X_train,y_train))
          print(abr.score(X_test,y_test))

          0.16571497739680363
          0.06313940161493081
```

6)XG Boost Regressor



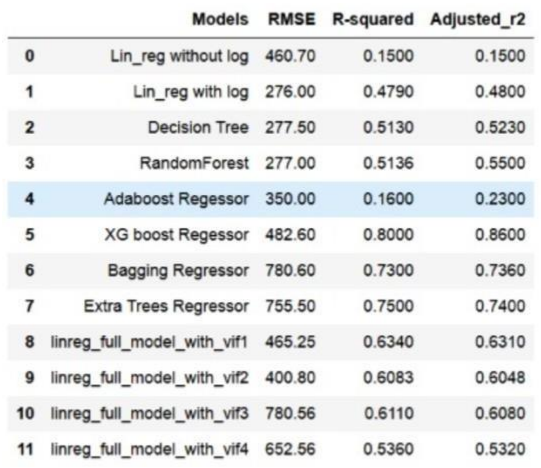
```
In [119]: from xgboost import XGBRegressor

In [545]: xgb = XGBRegressor()

In [546]: xgb.fit(X_train,y_train)

Out[546]: XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                       colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
                       importance_type='gain', interaction_constraints='',
                       learning_rate=0.300000012, max_delta_step=0, max_depth=6,
                       min_child_weight=1, missing=nan, monotone_constraints='()',
                       n_estimators=100, n_jobs=8, num_parallel_tree=1, random_state=0,
                       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
                       tree_method='exact', validate_parameters=1, verbosity=None)

In [547]: xgb.score(X_train,y_train)

Out[547]: 0.8043505813157442

In [548]: xgb.score(X_test,y_test)

Out[548]: 0.7399914088548999
```

7) K-Neighbours Regressor



```
In [225]: knn = KNeighborsRegressor()
          knn.fit(X_train,y_train)

Out[225]: KNeighborsRegressor()

In [227]: print(knn.score(X_train,y_train))

          0.5095341379253244

In [228]: print("Testing accuracy:",knn.score(X_test,y_test))

          Testing accuracy: 0.29017996011783387

In [229]: y_pred_knn=knn.predict(X_test)

In [232]: print("RMSE score",np.sqrt(mean_squared_error(y_test,y_pred_knn)))

          RMSE score 430.6360322427319

In [233]: print("R2 score",r2_score(y_test,y_pred))

          R2 score 0.15833698962458642
```

Chapter 9 : Comparison and selection of model Results

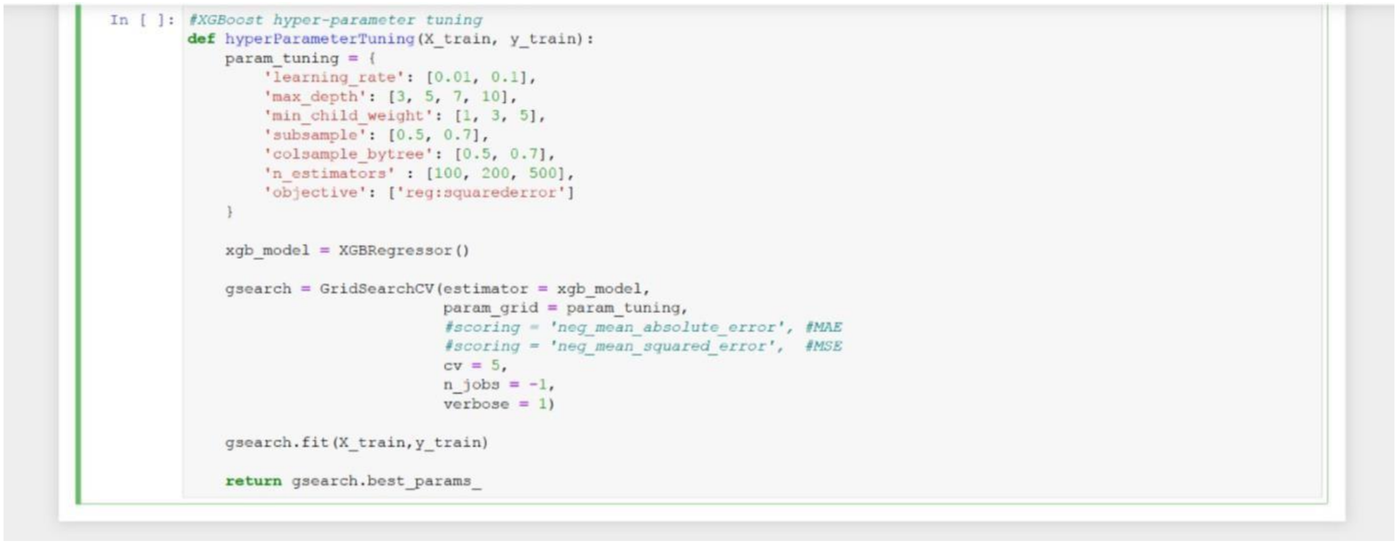


	Models	RMSE	R-squared	Adjusted_r2
0	Lin_reg without log	460.70	0.1500	0.1500
1	Lin_reg with log	276.00	0.4790	0.4800
2	Decision Tree	277.50	0.5130	0.5230
3	RandomForest	277.00	0.5136	0.5500
4	Adaboost Regessor	350.00	0.1600	0.2300
5	XG boost Regessor	482.60	0.8000	0.8600
6	Bagging Regressor	780.60	0.7300	0.7360
7	Extra Trees Regressor	755.50	0.7500	0.7400
8	linreg_full_model_with_vif1	465.25	0.6340	0.6310
9	linreg_full_model_with_vif2	400.80	0.6083	0.6048
10	linreg_full_model_with_vif3	780.56	0.6110	0.6080
11	linreg_full_model_with_vif4	652.56	0.5360	0.5320

Model Name	**R2**	**RMSE**
Linear Regression without Feature Engineering	0.151	0.461
Linear Regression with Feature Engineering	0.503	0.280
Decision Tree Regressor	0.513	0.277
Random Forest Regressor	0.713	0.212
Ada Boost Regressor	0.06	0.985
XG Boost Regressor	0.739	0.202
K Neighbours Regressor	0.158	0.430

As we can see in the above table, XG Boost Regressor is performing better from other models
With a test accuracy of 73.9 % and RMSE 0.280 which is less as compared to the other models.

Chapter 10 : Hyper parameter tuning



```
In [ ]: #XGBoost hyper-parameter tuning
        def hyperParameterTuning(X_train, y_train):
            param_tuning = {
                'learning_rate': [0.01, 0.1],
                'max_depth': [3, 5, 7, 10],
                'min_child_weight': [1, 3, 5],
                'subsample': [0.5, 0.7],
                'colsample_bytree': [0.5, 0.7],
                'n_estimators' : [100, 200, 500],
                'objective': ['reg:squarederror']
            }

            xgb_model = XGBRegressor()

            gsearch = GridSearchCV(estimator = xgb_model,
                                   param_grid = param_tuning,
                                   #scoring = 'neg_mean_absolute_error', #MAE
                                   #scoring = 'neg_mean_squared_error',  #MSE
                                   cv = 5,
                                   n_jobs = -1,
                                   verbose = 1)

            gsearch.fit(X_train,y_train)

            return gsearch.best_params_
```

From using the best model XG boost(compared to the other models) and performing hyper parameter turning ,we got the following best parameters:

1) Learning rate:0.01
2) Max depth:10
3) Min_child_weight: 0.5

4) n_estimators:500

5) subsample:0.5
6) colsample_bytree=0.7

We created a XG boostmodel with best hyper tuned parameters

And got an accuracy of 79% on the test data with RMSE of 0.140.

Chapter 11 : Conclusion

1. We feel that some features are missing to accurately predict the price of the bnbs.Such as most bnb charge an extra $25 /day per person ..
2. Since we don't have information on the booking date . Hard assumptions such as booking_date = date of review - nights stayed , we are afraid that it might increase the cost of the model as it directly affects the number of weekends and holidays during the guest's stay.
3. Price model varies between hosts and superhosts. Since our dataset is missing the rating its difficult cluster the hosts and the superhosts. However we can cluster them on the other bases.
4. Airbnb gives all their hosts have full control on their pricing model such as price of the bnb on the weekend , or a holiday ,management charges , damaged cause to property etc. which could mean each record in our dataset could very possibly have a unique custom pricing model with different coefficients attached to the feature columns. However activity of superhosts can be predicted with more confidence as they try to maintain the same model , varying only with the neighbourhood .

What next ?

1. With the help of latitude and longitude ,we can find out the listings which are closer to some posh locations where the prices are high.
2. Clustering neighbourhood in terms of how costly the neighbourhood is.
3. Trying different types of regressors such as extra-tree regressor .
4. Since it was not possible to run automated hyperparameter tuning due to the size of the dataset and the computational power of my personal laptop, there may be a set of hyperparameters that would be optimal for the models above, which may increase predictive power or lower RMSE. Other models were also explored such as XGBoost which gave the best accuracy.

Citations:

1) Kaggle Courses: "Introduction to Machine Learning" and "Intermediate Machine Learning". Code is referenced from here.
2) https://www.kaggle.com/thomaskonstantin/u-s-airbnb-analysis-and-price-prediction. Code is referenced from this notebook for the purposes of data analysis.
3) https://www.kaggle.com/ojaswagarg/nlp-to-get-over-90-regression-score. Code is referenced from this notebook for the purposes of data analysis.
4) https://machinelearningmastery.com/rfe-feature-selection-in-python/. Code is referenced from here for running recursive feature elimination.
5) "Machine Learning for Absolute Beginners" by Oliver Theobald