CAPSTONE PROJECT

Host optimal rent price prediction

GROUP-7

Mentored by:

Vibha Santhanam

Submitted by

Akash G

Gautham U

Mahima

Sai Mohan P

Sibe S

Table of Contents:

Industry review

1.1 Overview

The industry of hospitality and domain focuses on broad group of services that includes Lodging, transportation theme parks, cruise lines as well as additional fields inside the tourism industry. Under the lodging there are several sectors which includes hotels, motels, resorts and bed and breakfasts.

Bread and breakfasts (B&B) are like a private rooms or homes rented out by the hosts for the customers to stay in for night and a morning breakfast.

1.2 Business problem statement

One challenge that every hosts face is determining the optimal nightly rent price. In many areas, renters aka. hosts are presented with a good selection of listings and can filter by criteria like price, number of bedrooms, room type, and more. The amount a host can charge is ultimately tied to market prices.

Although there are general guidance which hosts should follow for pricing, there are no easy way to access methods to determine the optimal price to rent out a space.

There are other models available on the market where one method could be to find a few listings that are similar to the place that will be up for rent, average the listed prices and set our price to this calculated average price. But the market is dynamic, so we would want to update the price frequently and this method can become tedious.

To identify the growth of the business using a public dataset from Airbnb.,, Analyzing and building the model on how it will perform based on the current business situation and to build a optimal pricing system for the hosts who would opt when they post their properties for rent at the best price.

Our approach of analysis will include Exploratory Data Analysis (EDA) done on the public dataset we obtained, which may involve in data cleansing, visualization for further analysis of each variable or attribute.

Moving further, we plan to develop a supervised learning model based on regression to predict the price based on the historic data of the rental price, service charge with the aid of various regression methods or algorithms.

In summary, we are going to create a model that helps in predicting the optimal prices with a public dataset available.

1.3 Topic Survey in Brief

1.3.1 Problem understanding

The prime challenge that hosts face is determining the optimal nightly rent price. As the industry grows higher each year, there is frequent conversion happening. With the help of existing data obtained from Airbnb dataset we can predict how the model works with business statement, optimal pricing and new feature or update added if needed.

1.3.2 Current solution to the problem

Currently the companies like AirBnB follows many pricing strategies to fix the prices for the listings available. The company currently uses a smart pricing tool where the pricings are based on seasonal demands, special events and daily trends. Some companies follow RevPAR (**revenue per available rental**) method which considers your average daily rate (ADR) as well as your occupancy rate (OR).

1.3.3 Proposed solution to the problem

Our priority is to build a model which would predict the pricing quoted by the hosts and the plan is to formulize the model in accordance with the ongoing market and economy.

1.3.4 Critical assessment of topic survey

Summary

1. The hosts requires a new and optimized pricing strategy to stay ahead of the curve and stabilize in the current market.
2. Hence we plan on aiding them to come up with such a strategy by building a regression model for predicting the best price they would place in the future depending upon certain constrictions.

Opportunities in the current market

1. Smartphones and Internet has made a huge necessity which can be considered as a greater opportunity for the company to provide improvised web and mobile applications for customers with attractive advertisements and offers.
2. A very dynamic policy can be framed with the hosts to avoid poor hosts, scam, unfriendly environments which would increase maximum productivity.
3. Some KPI’s such as occupancy rate, operation costs, average income can be trimmed or expanded based on the quarterly market basis.

2 Literature review

Over the past decade, online bookings have grown in leaps and bounds. Online markets in the United States, Western Europe, Japan, Australia, New Zealand and Singapore have matured over the past decade.

The online travel industry is composed of a variety of companies that offer differing rates, costs, and locations for accommodations, tour packages. The online travel industry offers general consumers a convenient, quick and easy way to book their traveling needs and wants.

Hotels are considered one of the keys to tourism industry. Setting the proper price for hotels has always been a mystery for the decision makers because of its direct relationship with the demand for hotels. Thus, in the current market under competitive conditions an optimal pricing system is to be decided.

The goal of price optimization is to help hosts who share their homes/rooms/cabin through OTA (Online Travel Agencies) set the optimal price for their listings. In contrast to conventional pricing problems, where pricing strategies are applied to a large quantity of identical listings, there are no identical listings in online travel services, because each listing on these platforms offer unique value and experiences to the guests.

There are many factors affecting the pricing quoted by the hosts, which depend on the demography, instant bookability, reviews, customer preference, room/property type etc. which are considered some of the key factors for analysis and coming up with a model to predict the optimal price correlating with the host’s quotation.

3. Dataset and Domain

3.1 Data Dictionary

The public dataset obtained from AirBnB contains several variables based on the host, location, regulations, reviews, price, availability etc.

Data Source: link

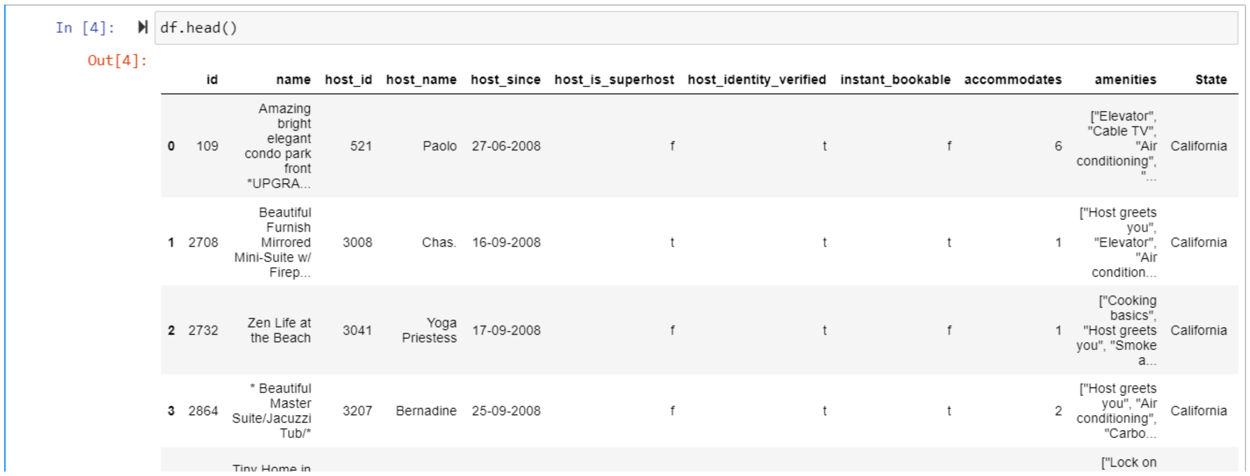
Below are the features available in the dataset:

Features	**Data type**	**Description**
id	integer	Airbnb's unique identifier for the listing
name	text	Name of the listing
host_id	integer	Airbnb's unique identifier for the host.
host_name	text	Name of the host.
host_since	date	The date the host was created. For hosts that are Airbnb guests this could be the date they registered as a guest.
host_is_superhost	boolean	To check if the host is a superhost or not.
host_identity_verified	boolean	To check if the host is identified or not.
instant_bookable	boolean	To check whether the guest can automatically book the listing without the host requiring to accept their booking request. An indicator of a commercial listing.
accommodates	integer	The maximum capacity of the listing
amenities	object	The amenities provided by the host
State	object	State of the host and residence.

neighbourhood_group	object	Name of the neighbourhood group.
neighbourhood	object	Name of the neighbourhood.
latitude	float	Uses the World Geodetic System (WGS84) projection for latitude and longitude.
longitude	float	Uses the World Geodetic System (WGS84) projection for latitude and longitude.
property_type	object	Type of the property as such by their hosts.
room_type	object	Type of the room as described as such by their hosts.
bathrooms	float	The number of bathrooms in the listing
bedrooms	float	The number of bedrooms in the listing
beds	float	The number of beds in the room.
price	object	daily price in local currency
minimum_nights	integer	Minimum number of night stay for the listing.
maximum_nights	integer	Maximum number of night stay for the listing.
number_of_reviews	integer	The number of reviews made for the listing.
last_review	date	The date where the last review was made.
reviews_per_month	integer	The number of reviews the listing has over the lifetime of the listing
calculated_host_listings_count	integer	The number of listings the host has made in the region.
availability_365	integer	The availability of the listing for 365 days posted by the host.

4 Data Exploration (EDA):

4.1 Variable categorization (count of numeric and categorical):

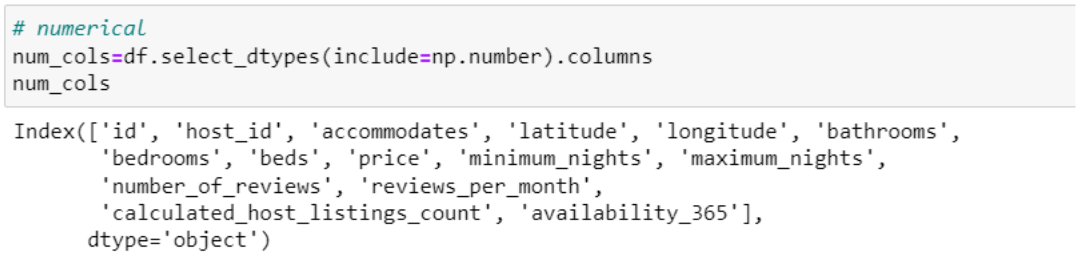


```
In [4]:  df.head()
```

Out[4]:

	id	name	host_id	host_name	host_since	host_is_superhost	host_identity_verified	instant_bookable	accommodates	amenities	State
0	109	Amazing bright elegant condo park front *UPGRA...	521	Paolo	27-06-2008	f	t	f	6	["Elevator", "Cable TV", "Air conditioning", "...	California
1	2708	Beautiful Furnish Mirrored Mini-Suite w/ Firep...	3008	Chas.	16-09-2008	t	t	t	1	["Host greets you", "Elevator", "Air condition...	California
2	2732	Zen Life at the Beach	3041	Yoga Priestess	17-09-2008	f	t	f	1	["Cooking basics", "Host greets you", "Smoke a...	California
3	2864	* Beautiful Master Suite/Jacuzzi Tub/*	3207	Bernadine	25-09-2008	f	t	t	2	["Host greets you", "Air conditioning", "Carbo...	California
		Tiny Home in								["Lock on	

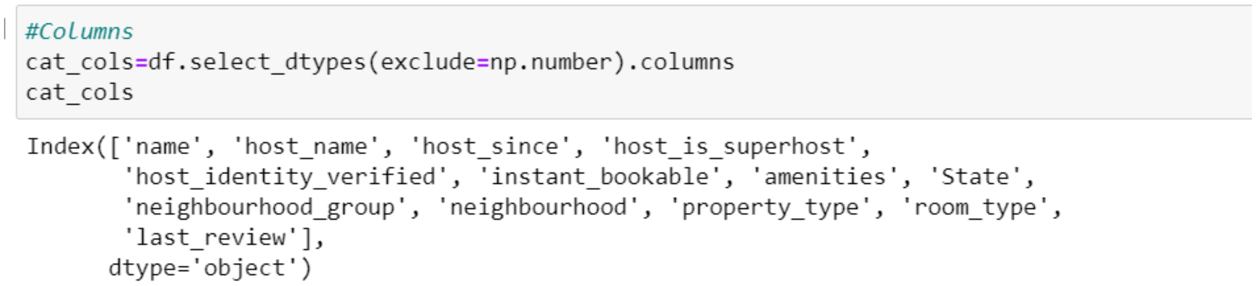
4.1.1 Numerical Columns:



```
# numerical
num_cols=df.select_dtypes(include=np.number).columns
num_cols
```

```
Index(['id', 'host_id', 'accommodates', 'latitude', 'longitude', 'bathrooms',
       'bedrooms', 'beds', 'price', 'minimum_nights', 'maximum_nights',
       'number_of_reviews', 'reviews_per_month',
       'calculated_host_listings_count', 'availability_365'],
      dtype='object')
```

4.1.2 Categorical Columns:

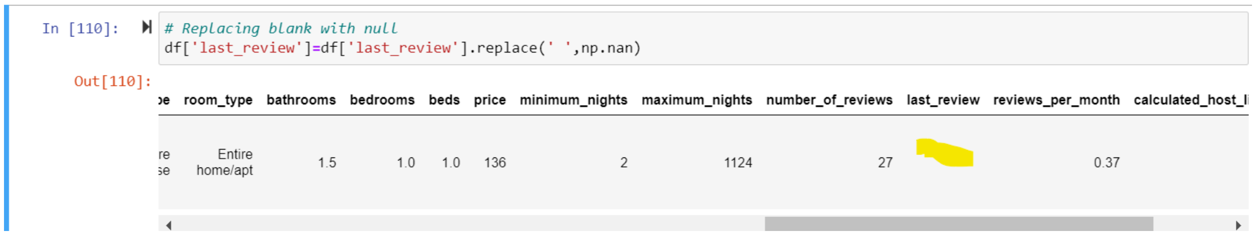


```
#Columns
cat_cols=df.select_dtypes(exclude=np.number).columns
cat_cols
```

```
Index(['name', 'host_name', 'host_since', 'host_is_superhost',
       'host_identity_verified', 'instant_bookable', 'amenities', 'State',
       'neighbourhood_group', 'neighbourhood', 'property_type', 'room_type',
       'last_review'],
      dtype='object')
```

4.2 Data Cleaning:

4.2.1 Identifying blank values in the data:

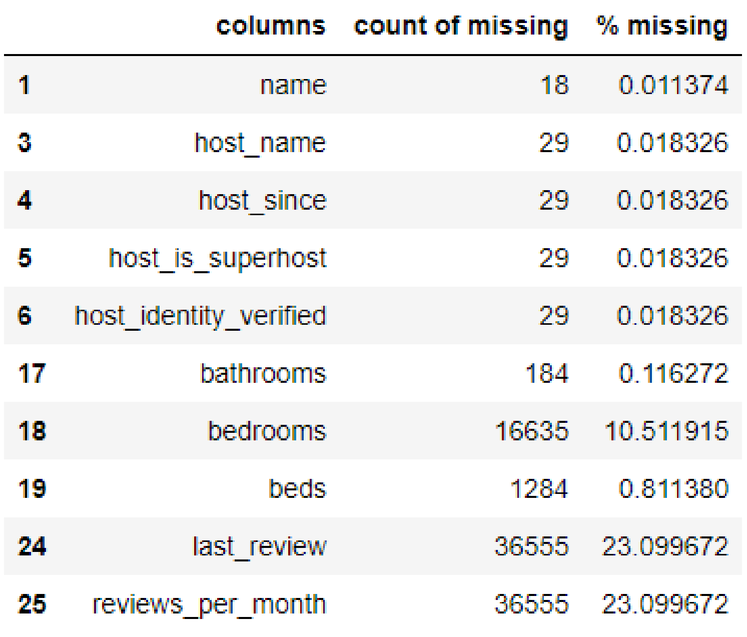


In [110]:
```
# Replacing blank with null
df['last_review']=df['last_review'].replace(' ',np.nan)
```

Out[110]:

›e	room_type	bathrooms	bedrooms	beds	price	minimum_nights	maximum_nights	number_of_reviews	last_review	reviews_per_month	calculated_host_l
re se	Entire home/apt	1.5	1.0	1.0	136	2	1124	27		0.37	

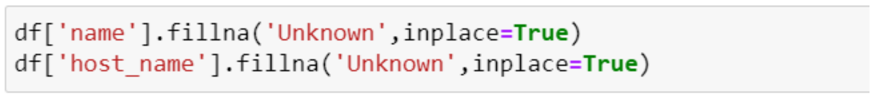
4.2.2 Null Values in the dataset:



	columns	count of missing	% missing
1	name	18	0.011374
3	host_name	29	0.018326
4	host_since	29	0.018326
5	host_is_superhost	29	0.018326
6	host_identity_verified	29	0.018326
17	bathrooms	184	0.116272
18	bedrooms	16635	10.511915
19	beds	1284	0.811380
24	last_review	36555	23.099672
25	reviews_per_month	36555	23.099672

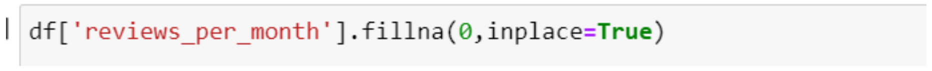
4.2.3 Treatment of Null Values:

- The null values in the name, host_name columns are imputed as "Unknown" as we don't have much data to support their information.



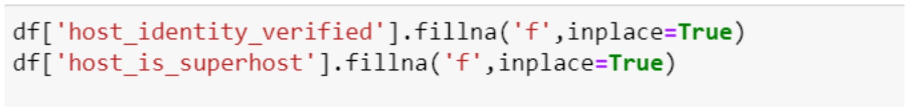
```
df['name'].fillna('Unknown',inplace=True)
df['host_name'].fillna('Unknown',inplace=True)
```

- The reviews_per_month null values are imputed with 0 as there was no total reviews have identified.



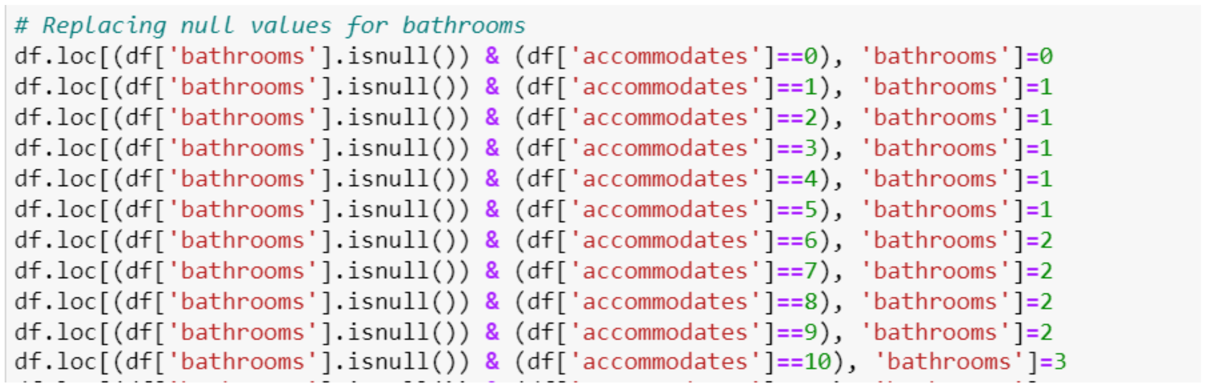
```
df['reviews_per_month'].fillna(0,inplace=True)
```

- The host_is_superhost, host_identity_verified columns are imputed with ‘f’ (false) for null values as they don’t have enough data to support the description of the respective columns.

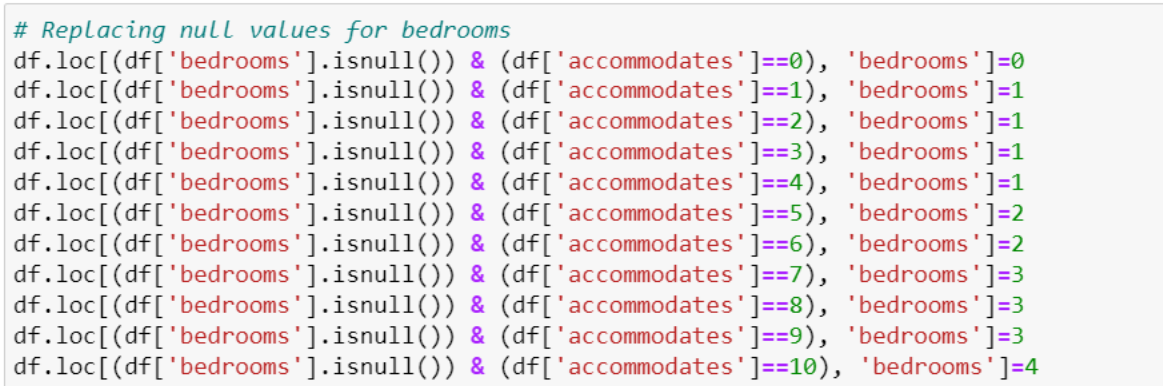


```
df['host_identity_verified'].fillna('f',inplace=True)
df['host_is_superhost'].fillna('f',inplace=True)
```

- The null values for bathrooms and bedrooms are filled based on the number of accommodates the particular listing can occupy.

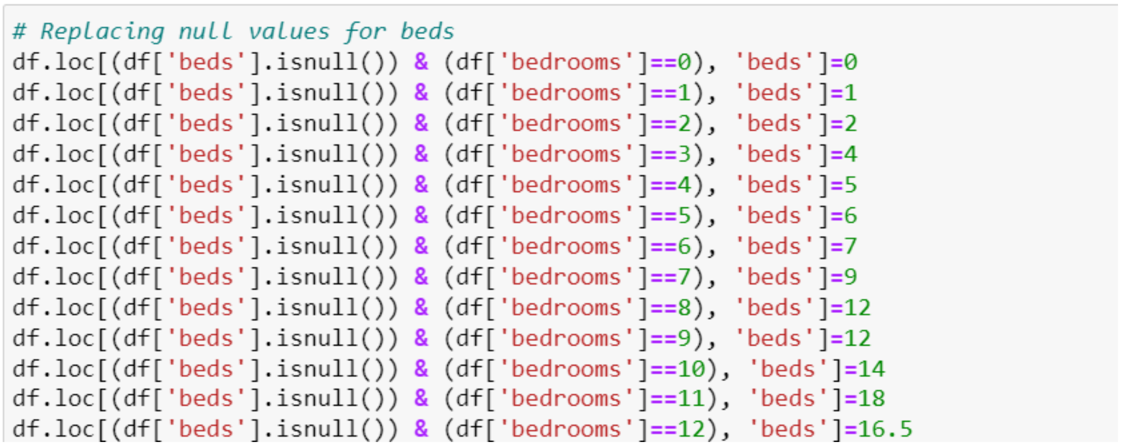


```
# Replacing null values for bathrooms
df.loc[(df['bathrooms'].isnull()) & (df['accommodates']==0), 'bathrooms']=0
df.loc[(df['bathrooms'].isnull()) & (df['accommodates']==1), 'bathrooms']=1
df.loc[(df['bathrooms'].isnull()) & (df['accommodates']==2), 'bathrooms']=1
df.loc[(df['bathrooms'].isnull()) & (df['accommodates']==3), 'bathrooms']=1
df.loc[(df['bathrooms'].isnull()) & (df['accommodates']==4), 'bathrooms']=1
df.loc[(df['bathrooms'].isnull()) & (df['accommodates']==5), 'bathrooms']=1
df.loc[(df['bathrooms'].isnull()) & (df['accommodates']==6), 'bathrooms']=2
df.loc[(df['bathrooms'].isnull()) & (df['accommodates']==7), 'bathrooms']=2
df.loc[(df['bathrooms'].isnull()) & (df['accommodates']==8), 'bathrooms']=2
df.loc[(df['bathrooms'].isnull()) & (df['accommodates']==9), 'bathrooms']=2
df.loc[(df['bathrooms'].isnull()) & (df['accommodates']==10), 'bathrooms']=3
```



```
# Replacing null values for bedrooms
df.loc[(df['bedrooms'].isnull()) & (df['accommodates']==0), 'bedrooms']=0
df.loc[(df['bedrooms'].isnull()) & (df['accommodates']==1), 'bedrooms']=1
df.loc[(df['bedrooms'].isnull()) & (df['accommodates']==2), 'bedrooms']=1
df.loc[(df['bedrooms'].isnull()) & (df['accommodates']==3), 'bedrooms']=1
df.loc[(df['bedrooms'].isnull()) & (df['accommodates']==4), 'bedrooms']=1
df.loc[(df['bedrooms'].isnull()) & (df['accommodates']==5), 'bedrooms']=2
df.loc[(df['bedrooms'].isnull()) & (df['accommodates']==6), 'bedrooms']=2
df.loc[(df['bedrooms'].isnull()) & (df['accommodates']==7), 'bedrooms']=3
df.loc[(df['bedrooms'].isnull()) & (df['accommodates']==8), 'bedrooms']=3
df.loc[(df['bedrooms'].isnull()) & (df['accommodates']==9), 'bedrooms']=3
df.loc[(df['bedrooms'].isnull()) & (df['accommodates']==10), 'bedrooms']=4
```

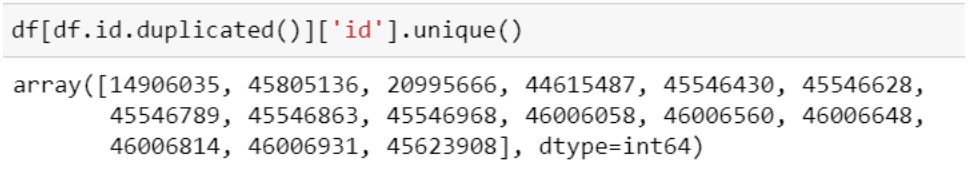
- The null values for beds are filled with respect to the number of bedrooms in the particular listings.



```
# Replacing null values for beds
df.loc[(df['beds'].isnull()) & (df['bedrooms']==0), 'beds']=0
df.loc[(df['beds'].isnull()) & (df['bedrooms']==1), 'beds']=1
df.loc[(df['beds'].isnull()) & (df['bedrooms']==2), 'beds']=2
df.loc[(df['beds'].isnull()) & (df['bedrooms']==3), 'beds']=4
df.loc[(df['beds'].isnull()) & (df['bedrooms']==4), 'beds']=5
df.loc[(df['beds'].isnull()) & (df['bedrooms']==5), 'beds']=6
df.loc[(df['beds'].isnull()) & (df['bedrooms']==6), 'beds']=7
df.loc[(df['beds'].isnull()) & (df['bedrooms']==7), 'beds']=9
df.loc[(df['beds'].isnull()) & (df['bedrooms']==8), 'beds']=12
df.loc[(df['beds'].isnull()) & (df['bedrooms']==9), 'beds']=12
df.loc[(df['beds'].isnull()) & (df['bedrooms']==10), 'beds']=14
df.loc[(df['beds'].isnull()) & (df['bedrooms']==11), 'beds']=18
df.loc[(df['beds'].isnull()) & (df['bedrooms']==12), 'beds']=16.5
```

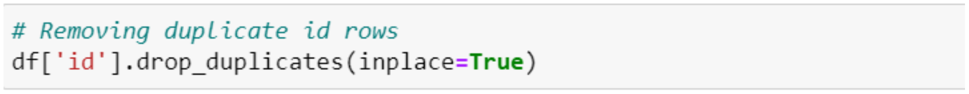
4.3 Removing duplicate records:

Host id is an unique identifier column, there shouldn't be any duplicate records in the data set. So we are finding those duplicated records and removing them.



```
df[df.id.duplicated()]['id'].unique()
```

```
array([14906035, 45805136, 20995666, 44615487, 45546430, 45546628,
       45546789, 45546863, 45546968, 46006058, 46006560, 46006648,
       46006814, 46006931, 45623908], dtype=int64)
```



```
# Removing duplicate id rows
df['id'].drop_duplicates(inplace=True)
```

4.4 Outlier Treatment:

The interquartile range is a measure of where the "middle fifty" is in a data set. Where a range is a measure of where the beginning and end are in a set, an interquartile range is a measure of where the bulk of the values lie. That's why it's preferred over many other measures of spread (i.e. the average or median) when reporting things like school performance or SAT scores.

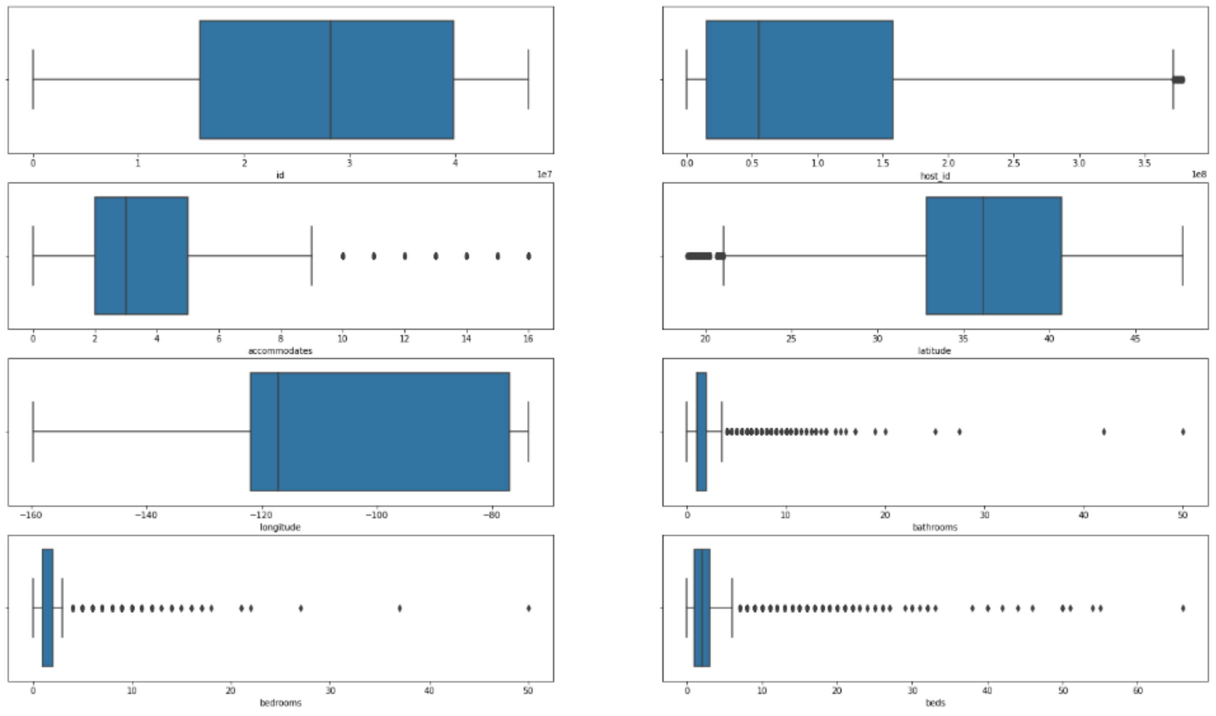
The interquartile range formula is the first quartile subtracted from the third quartile:

IQR = Q3 – Q1.

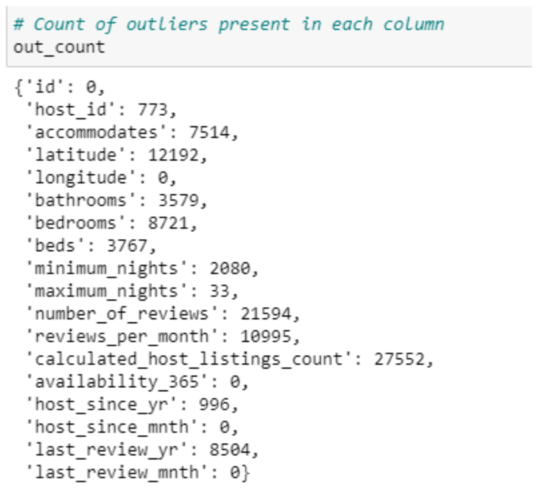
Upper limit= Q3 + 1.5 * IQR

Lower limit = Q1 - 1.5 * IQR

Hence any values below the lower limit or above the upper limit were capped or removed based on each feature’s variation.



Outliers found in each columns:

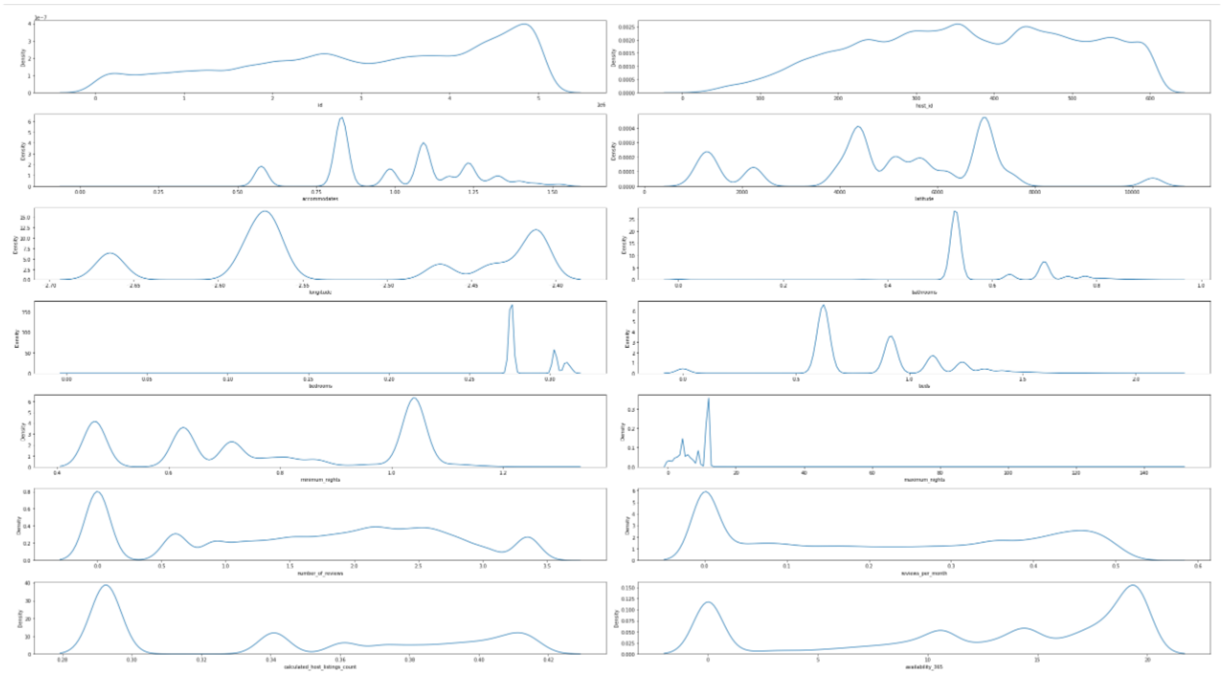


```
# Count of outliers present in each column
out_count
```

```
{'id': 0,
 'host_id': 773,
 'accommodates': 7514,
 'latitude': 12192,
 'longitude': 0,
 'bathrooms': 3579,
 'bedrooms': 8721,
 'beds': 3767,
 'minimum_nights': 2080,
 'maximum_nights': 33,
 'number_of_reviews': 21594,
 'reviews_per_month': 10995,
 'calculated_host_listings_count': 27552,
 'availability_365': 0,
 'host_since_yr': 996,
 'host_since_mnth': 0,
 'last_review_yr': 8504,
 'last_review_mnth': 0}
```

5 Relationship between the variables:

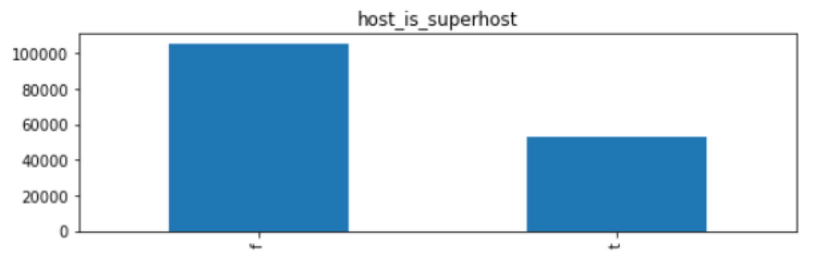
5.1 Univariate Analysis on Numerical Variables:



From the above image we can see that the numerical columns are skewed.

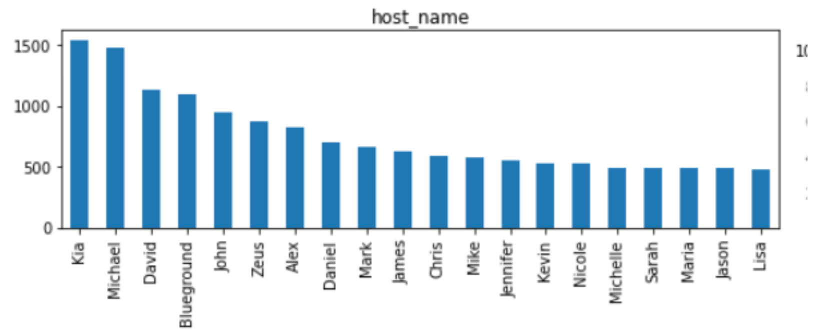
5.2 Univariate Analysis on Categorical Columns:

5.2.1 Count of host is super host column :



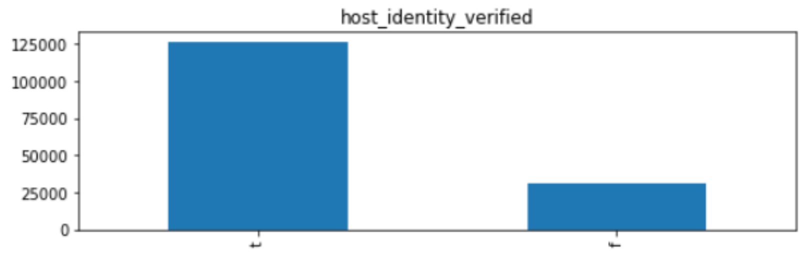
Majority of the hosts are not super hosts.

5.2.2 Count of host name :



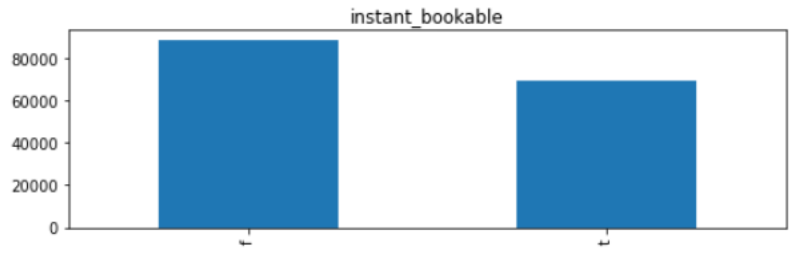
Kia has the highest number of listings in Seattle, followed by Micheal and David.

5.2.3 Count of host identity verified column:



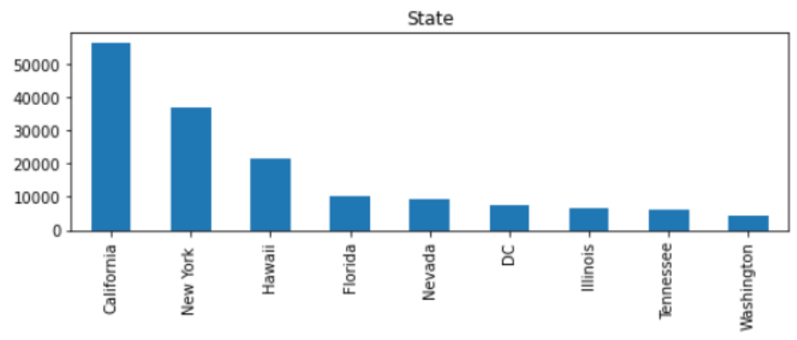
Inference: Majority of the hosts are verified.

5.2.4 Count of instant bookable column:



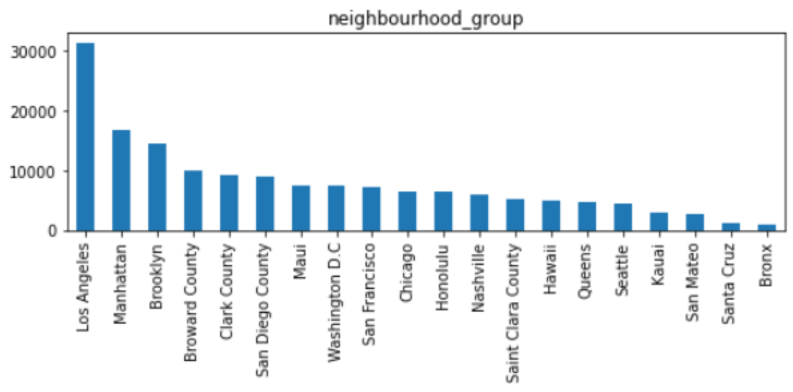
Instant bookable option is not available in most of the listings.

5.2.5 Count of State:



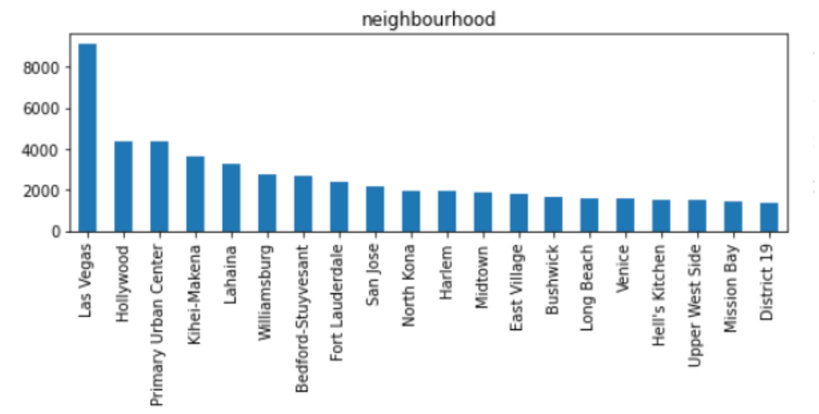
California has the highest number of listings followed by New york.

5.2.6 Count of neighbourhood group column:



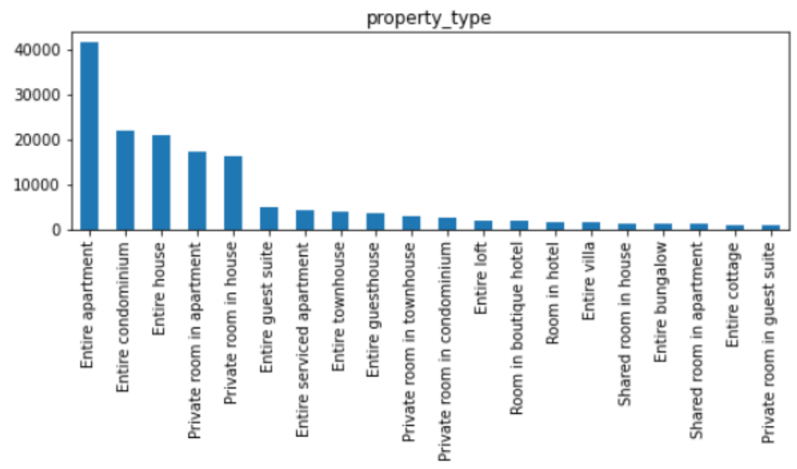
Los Angeles has the highest number of listing in the neighbourhood group.

5.2.7 Count of neighbourhood column :



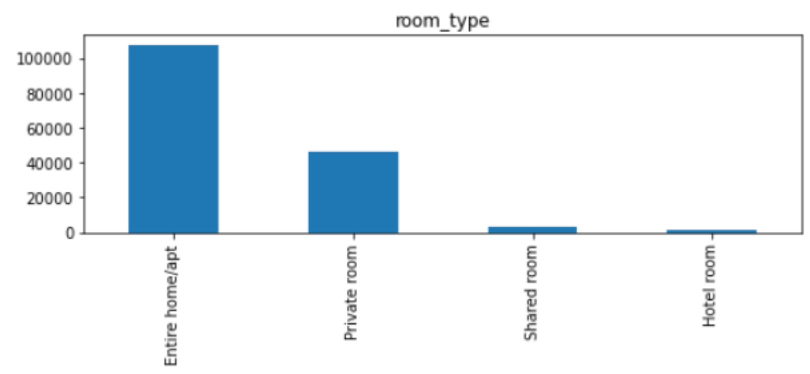
Los Vegas has the highest number of listing in the neighbourhood.

5.2.8 Count of property type column :



Entire Appartment has the highest count in the property listings based on property type.

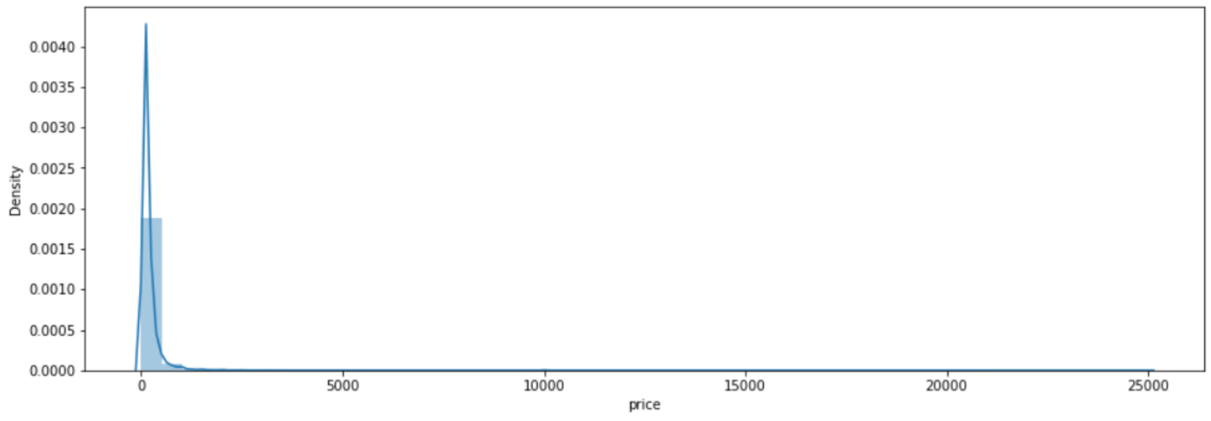
5.2.9 Count of room type column :



Entire home/Apt has the highest count in the property listings based on room type.

5.3 Target variable Analysis :

- Target variable : Price

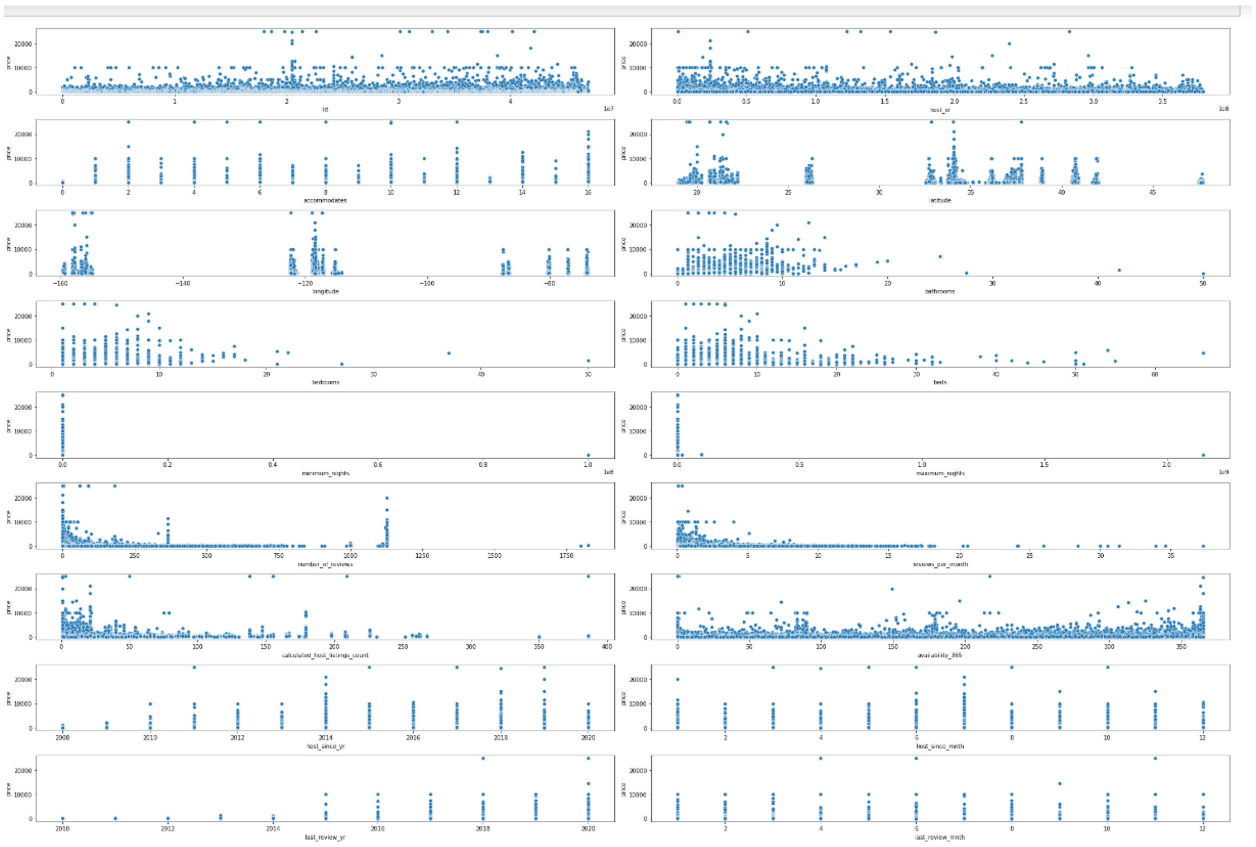


```
Skewness of price variable:  22.880891758896077
```

We can see that the target column price is having a positive skewed distribution, which infers the presence of outliers in the data.

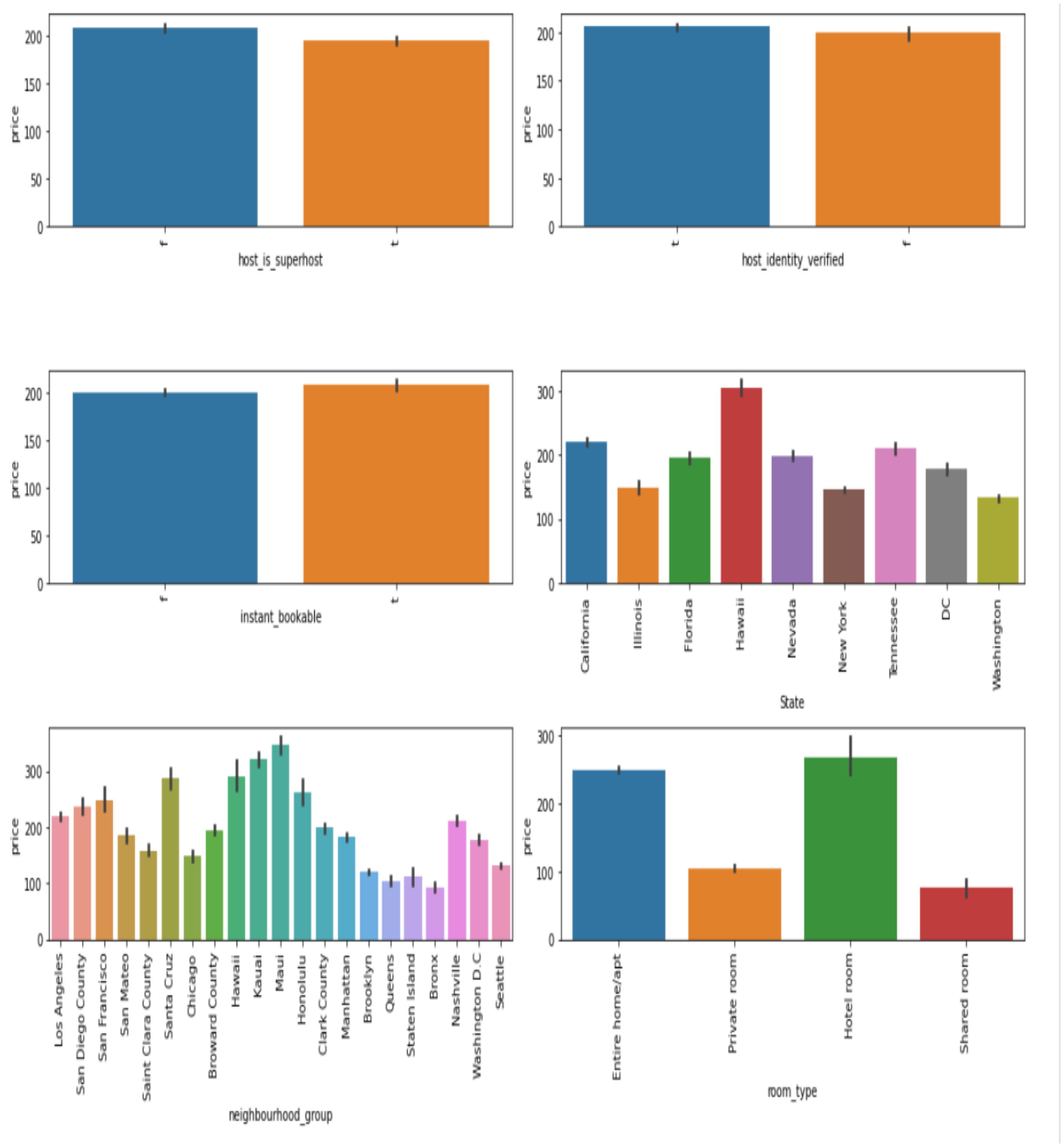
5.4 Bivariate Analysis based on target variable :

5.4.1 Relationship between numerical variables and target variable:



From the scatterplots we can see that there isn’t much relationship between the target variable price and the other numerical columns in the dataset.

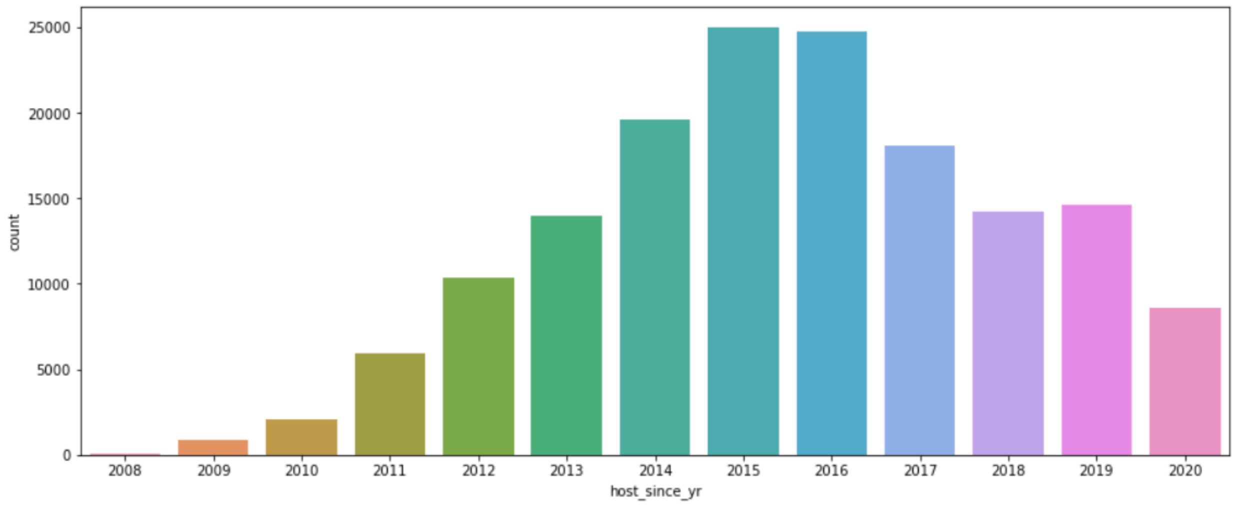
5.4.2 Relationship between target and categorical columns :



- Host_is_superhost: Average prices are almost equal for both superhosts and non-superhosts.
- Host_is_verified: Average prices are almost equal for the listings which are verified and unverified.

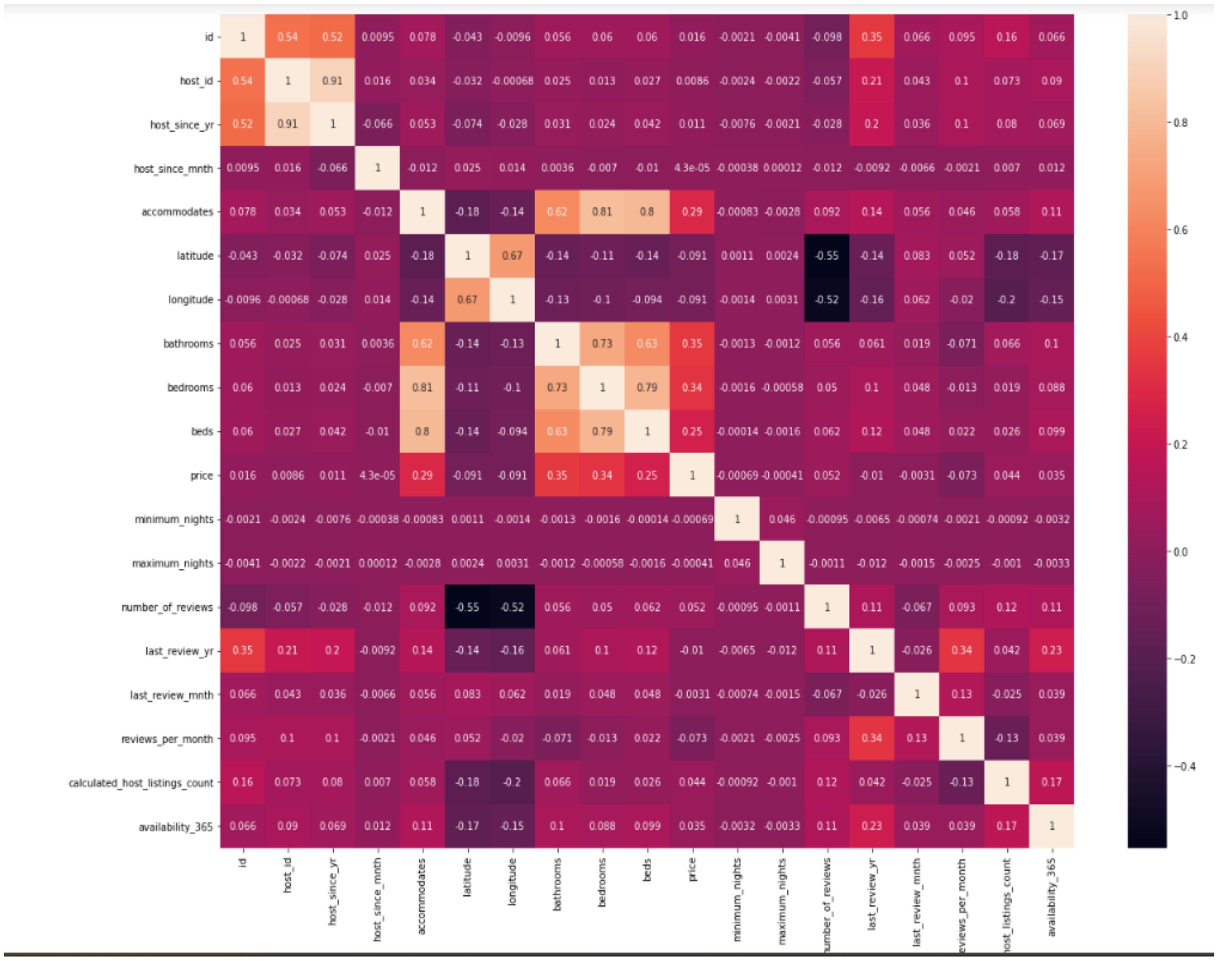
- Instant_bookable: Average prices are almost equal whether there is a availability of instant booking or not.
- State: Average price in Hawaii is greater than compared to other states.
- Neighbourhood_group: Average price in Maui neighbour hood group is greater than compared to others.
- Room_type: Average prices in hotel rooms and Entire home/apt are almost equal and higher than the other room types.

5.4.3 Number of new listings each year:



New hosts are increasing each year until 2016 and starts decreasing from the year 2017.

5.5 Correlation:



	id	host_id	host_since_yr	host_since_mnth	accommodates	latitude	longitude	bathrooms	bedrooms	beds	price	minimum_nights	maximum_nights	umber_of_reviews	last_review_yr	last_review_mnth	eviews_per_month	ost_listings_count	availability_365
id	1	0.54	0.52	0.0095	0.078	-0.043	-0.0096	0.056	0.06	0.06	0.016	-0.0021	-0.0041	-0.098	0.35	0.066	0.095	0.16	0.066
host_id	0.54	1	0.91	0.016	0.034	-0.032	-0.00068	0.025	0.013	0.027	0.0086	-0.0024	-0.0022	-0.057	0.21	0.043	0.1	0.073	0.09
host_since_yr	0.52	0.91	1	-0.066	0.053	-0.074	-0.028	0.031	0.024	0.042	0.011	-0.0076	-0.0021	-0.028	0.2	0.036	0.1	0.08	0.069
host_since_mnth	0.0095	0.016	-0.066	1	-0.012	0.025	0.014	0.0036	-0.007	-0.01	4.3e-05	-0.00038	0.00012	-0.012	-0.0092	-0.0066	-0.0021	0.007	0.012
accommodates	0.078	0.034	0.053	-0.012	1	-0.18	-0.14	0.62	0.81	0.8	0.29	-0.00083	-0.0028	0.092	0.14	0.056	0.046	0.058	0.11
latitude	-0.043	-0.032	-0.074	0.025	-0.18	1	0.67	-0.14	-0.11	-0.14	-0.091	0.0011	0.0024	-0.55	-0.14	0.083	0.052	-0.18	-0.17
longitude	-0.0096	-0.00068	-0.028	0.014	-0.14	0.67	1	-0.13	-0.1	-0.094	-0.091	-0.0014	0.0031	-0.52	-0.16	0.062	-0.02	-0.2	-0.15
bathrooms	0.056	0.025	0.031	0.0036	0.62	-0.14	-0.13	1	0.73	0.63	0.35	-0.0013	-0.0012	0.056	0.061	0.019	-0.071	0.066	0.1
bedrooms	0.06	0.013	0.024	-0.007	0.81	-0.11	-0.1	0.73	1	0.79	0.34	-0.0016	-0.00058	0.05	0.1	0.048	-0.013	0.019	0.088
beds	0.06	0.027	0.042	-0.01	0.8	-0.14	-0.094	0.63	0.79	1	0.25	-0.00014	-0.0016	0.062	0.12	0.048	0.022	0.026	0.099
price	0.016	0.0086	0.011	4.3e-05	0.29	-0.091	-0.091	0.35	0.34	0.25	1	-0.00069	-0.00041	0.052	-0.01	-0.0031	-0.073	0.044	0.035
minimum_nights	-0.0021	-0.0024	-0.0076	-0.00038	-0.00083	0.0011	-0.0014	-0.0013	-0.0016	-0.00014	-0.00069	1	0.046	-0.00095	-0.0065	-0.00074	-0.0021	-0.00092	-0.0032
maximum_nights	-0.0041	-0.0022	-0.0021	0.00012	-0.0028	0.0024	0.0031	-0.0012	-0.00058	-0.0016	-0.00041	0.046	1	-0.0011	-0.012	-0.0015	-0.0025	-0.001	-0.0033
number_of_reviews	-0.098	-0.057	-0.028	-0.012	0.092	-0.55	-0.52	0.056	0.05	0.062	0.052	-0.00095	-0.0011	1	0.11	-0.067	0.093	0.12	0.11
last_review_yr	0.35	0.21	0.2	-0.0092	0.14	-0.14	-0.16	0.061	0.1	0.12	-0.01	-0.0065	-0.012	0.11	1	-0.026	0.34	0.042	0.23
last_review_mnth	0.066	0.043	0.036	-0.0066	0.056	0.083	0.062	0.019	0.048	0.048	-0.0031	-0.00074	-0.0015	-0.067	-0.026	1	0.13	-0.025	0.039
reviews_per_month	0.095	0.1	0.1	-0.0021	0.046	0.052	-0.02	-0.071	-0.013	0.022	-0.073	-0.0021	-0.0025	0.093	0.34	0.13	1	-0.13	0.039
calculated_host_listings_count	0.16	0.073	0.08	0.007	0.058	-0.18	-0.2	0.066	0.019	0.026	0.044	-0.00092	-0.001	0.12	0.042	-0.025	-0.13	1	0.17
availability_365	0.066	0.09	0.069	0.012	0.11	-0.17	-0.15	0.1	0.088	0.099	0.035	-0.0032	-0.0033	0.11	0.23	0.039	0.039	0.17	1

- On comparing with the target variable there isn't much correlation between the variables.
- But there is multi collinearity is present between the columns accommodates , beds, bathroom and bedrooms
- Also multi-collinearity between host id and Host since year
- Latitude and Longitude has moderately strong multi collinearity between them.

6. Statistical significance of variables:

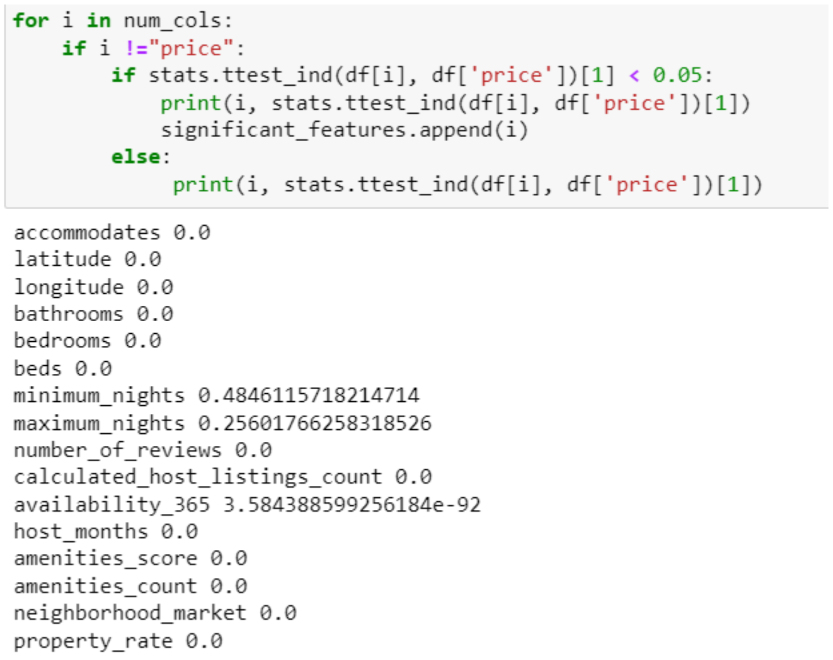
H0 (Null Hypothesis): Both groups have equal mean indicating that they are insignificant.

H1 (Alternate Hypothesis): Both groups do not have equal mean indicating that they are significant.

If p-value < 0.05 reject the null hypothesis that is that the means are not equal and the features are significant.

6.1 Significance test for numerical variables using ttest:

t-test is a statistical test that is used to compare the means of two groups. It is often used in hypothesis testing to determine whether a process or treatment actually has an effect on the population of interest, or whether two groups are different from one another.



```
for i in num_cols:
    if i !="price":
        if stats.ttest_ind(df[i], df['price'])[1] < 0.05:
            print(i, stats.ttest_ind(df[i], df['price'])[1])
            significant_features.append(i)
        else:
             print(i, stats.ttest_ind(df[i], df['price'])[1])
```

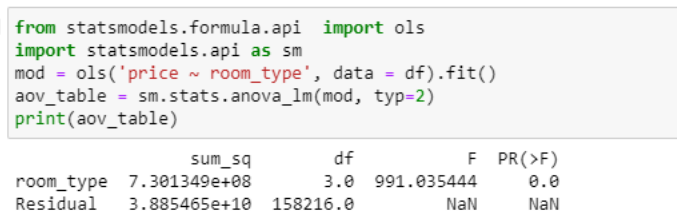
```
accommodates 0.0
latitude 0.0
longitude 0.0
bathrooms 0.0
bedrooms 0.0
beds 0.0
minimum_nights 0.4846115718214714
maximum_nights 0.25601766258318526
number_of_reviews 0.0
calculated_host_listings_count 0.0
availability_365 3.584388599256184e-92
host_months 0.0
amenities_score 0.0
amenities_count 0.0
neighborhood_market 0.0
property_rate 0.0
```

After performing the t-test we have identified the significant numerical variables as follows,

'accommodates', 'latitude', 'longitude', 'bathrooms', 'bedrooms', 'beds', 'number_of_reviews', 'reviews_per_month', 'calculated_host_listings_count', 'availability_365', 'host_months','Property_rate'

6.2 Anova test to find significance between categorical columns:

6.2.1 Anova test on Room type column:

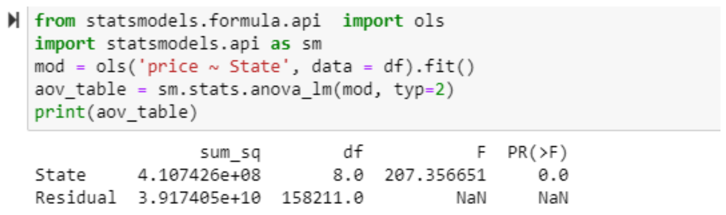


```
from statsmodels.formula.api  import ols
import statsmodels.api as sm
mod = ols('price ~ room_type', data = df).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print(aov_table)
```

```
                 sum_sq        df           F  PR(>F)
room_type  7.301349e+08       3.0  991.035444     0.0
Residual   3.885465e+10  158216.0         NaN     NaN
```

Since P-value is lesser than 0.05, room_type is a significant variable

6.2.2 Anova test on State column:

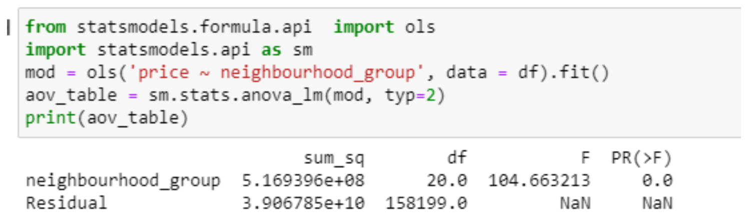


```
from statsmodels.formula.api  import ols
import statsmodels.api as sm
mod = ols('price ~ State', data = df).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print(aov_table)
```

```
                sum_sq        df           F  PR(>F)
State     4.107426e+08       8.0  207.356651     0.0
Residual  3.917405e+10  158211.0         NaN     NaN
```

Since P-value is lesser than 0.05, State is a significant variable

6.2.3 Anova Test on neighbourhood group:

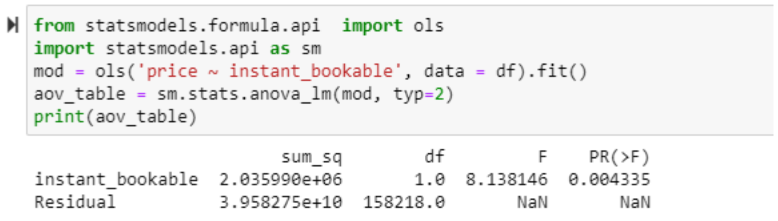


```
from statsmodels.formula.api  import ols
import statsmodels.api as sm
mod = ols('price ~ neighbourhood_group', data = df).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print(aov_table)
```

```
                           sum_sq        df           F  PR(>F)
neighbourhood_group  5.169396e+08      20.0  104.663213     0.0
Residual             3.906785e+10  158199.0         NaN     NaN
```

Since P-value is lesser than 0.05, neighbourhood_group is a significant variable

6.2.4 Anova test on instant bookable column:

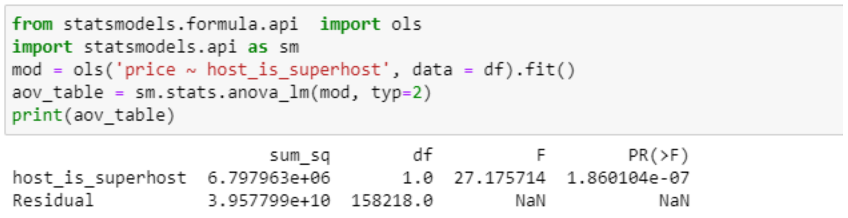


```
from statsmodels.formula.api  import ols
import statsmodels.api as sm
mod = ols('price ~ instant_bookable', data = df).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print(aov_table)
```

```
                        sum_sq        df         F    PR(>F)
instant_bookable  2.035990e+06       1.0  8.138146  0.004335
Residual          3.958275e+10  158218.0       NaN       NaN
```

Since P-value is lesser than 0.05, instant_bookable is a significant variable.

6.2.5 Anova Test on Is super host column:



```
from statsmodels.formula.api  import ols
import statsmodels.api as sm
mod = ols('price ~ host_is_superhost', data = df).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
print(aov_table)
```

```
                         sum_sq        df          F        PR(>F)
host_is_superhost  6.797963e+06       1.0  27.175714  1.860104e-07
Residual           3.957799e+10  158218.0        NaN           NaN
```

Since P-value is lesser than 0.05, host_is_superhost is a significant variable.

7. Feature Engineering

7.1 Amenities Score

Word Cloud:

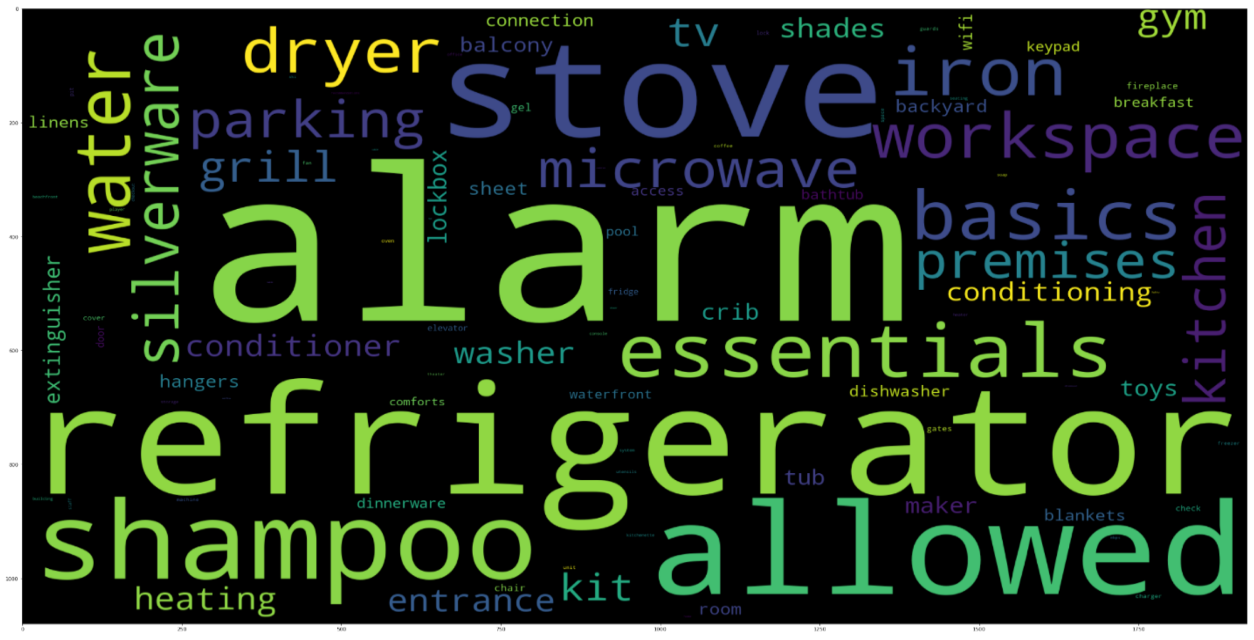
Word Cloud is a great way to represent text data. The size and color of each word that appears in the word cloud indicate it's frequency or importance.

A word cloud (also known as a tag cloud) is a visual representation of words. Cloud creators are used to highlight popular words and phrases based on frequency and relevance. They provide you with quick and simple visual insights that can lead to more in-depth analyses.

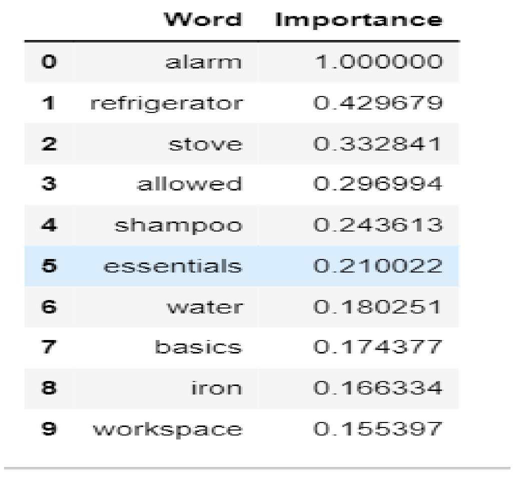
Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance .The dataset used for generating word cloud is collected from UCI Machine Learning Repository. It consists of YouTube comments on videos of popular artists.

7.1.1 Word cloud on Amenities column:

The word cloud is applied on amenities column to identify the most commonly available amenities in the whole listings.



The top 10 features identified from the amenities are,



	Word	Importance
0	alarm	1.000000
1	refrigerator	0.429679
2	stove	0.332841
3	allowed	0.296994
4	shampoo	0.243613
5	essentials	0.210022
6	water	0.180251
7	basics	0.174377
8	iron	0.166334
9	workspace	0.155397

By taking the Importance value of each amenities and making a cumulative sum of them to get a scoring column.

7.2 Regions column

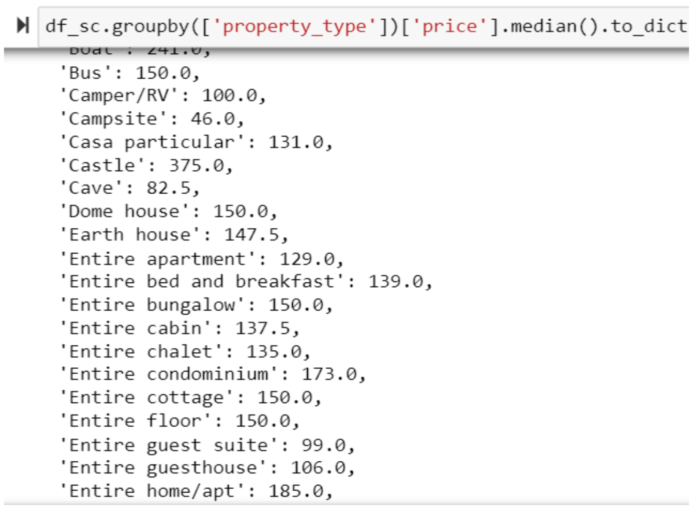
We divide the states based on the US regions.

- California - West
- Illinois - Middle West
- Florida - South East
- Hawaii - West
- Nevada - West
- New York - North East
- Tennessee - South East
- Washington - West
- DC – West

So the states are divided into 4 major regions, so that we can predict price based on weightages provided by each regions

7.3 Property Rate:

Based on the property type obtaining reasonable price for each category from the target variable. Below are the sample of property type and their respective property rates.



```
df_sc.groupby(['property_type'])['price'].median().to_dict
 'Boat': 241.0,
 'Bus': 150.0,
 'Camper/RV': 100.0,
 'Campsite': 46.0,
 'Casa particular': 131.0,
 'Castle': 375.0,
 'Cave': 82.5,
 'Dome house': 150.0,
 'Earth house': 147.5,
 'Entire apartment': 129.0,
 'Entire bed and breakfast': 139.0,
 'Entire bungalow': 150.0,
 'Entire cabin': 137.5,
 'Entire chalet': 135.0,
 'Entire condominium': 173.0,
 'Entire cottage': 150.0,
 'Entire floor': 150.0,
 'Entire guest suite': 99.0,
 'Entire guesthouse': 106.0,
 'Entire home/apt': 185.0,
```

7.4 Property Budget

Based on the property type the budgets are categorized into four major divisions.

- Low
- Moderate
- High
- Luxury

7.5 Host Rank

Based on the number of years the hosts has rented their properties we have identified their ranks into three categories

- Less than 4 years are categorized as **Novice**
- 4 to 8 years are categorized as **Middle Term**
- More than 8 years are categorized as **Long term**

7.6 Amenities Count

The number of amenities provided by the listings are calculated and created as a new column. This may provide a weighted advantage over the price prediction of the host listings.

7.7 Redundant columns removing:

The following columns are not going to bring much of an influence in predicting the price of the listings, they are being removed.

name, host_name, id, host_id, State, neighbourhood_group, amenities, reviews_per_month, host_since_yr, host_since_mnth, property_type, neighbourhood

7.8 One Hot Encoding:

As there are a smaller number of unique categories present in the following columns we are moving forward with the One hot encoding.

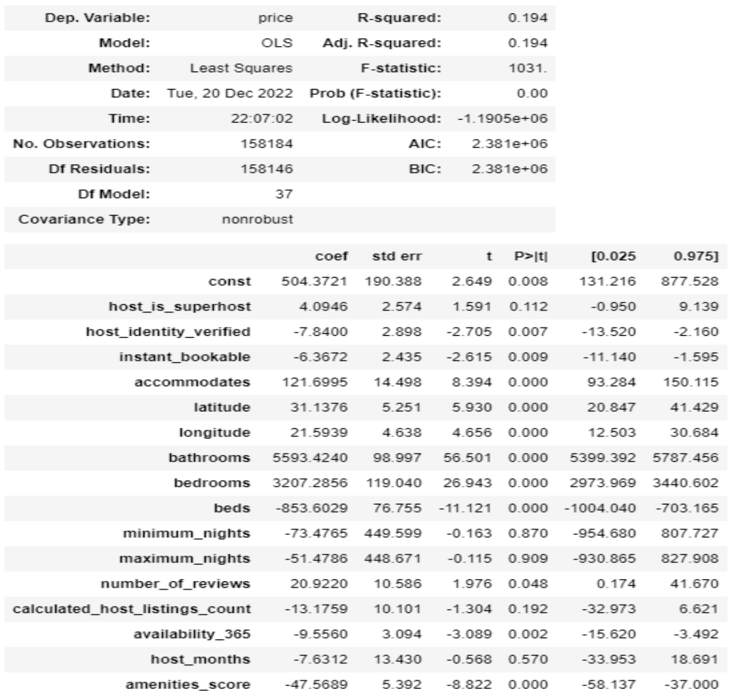
host_is_superhost, host_identity_verified, instant_bookable

8. Model Building:

8.1 Linear Regression Model (Base Model)

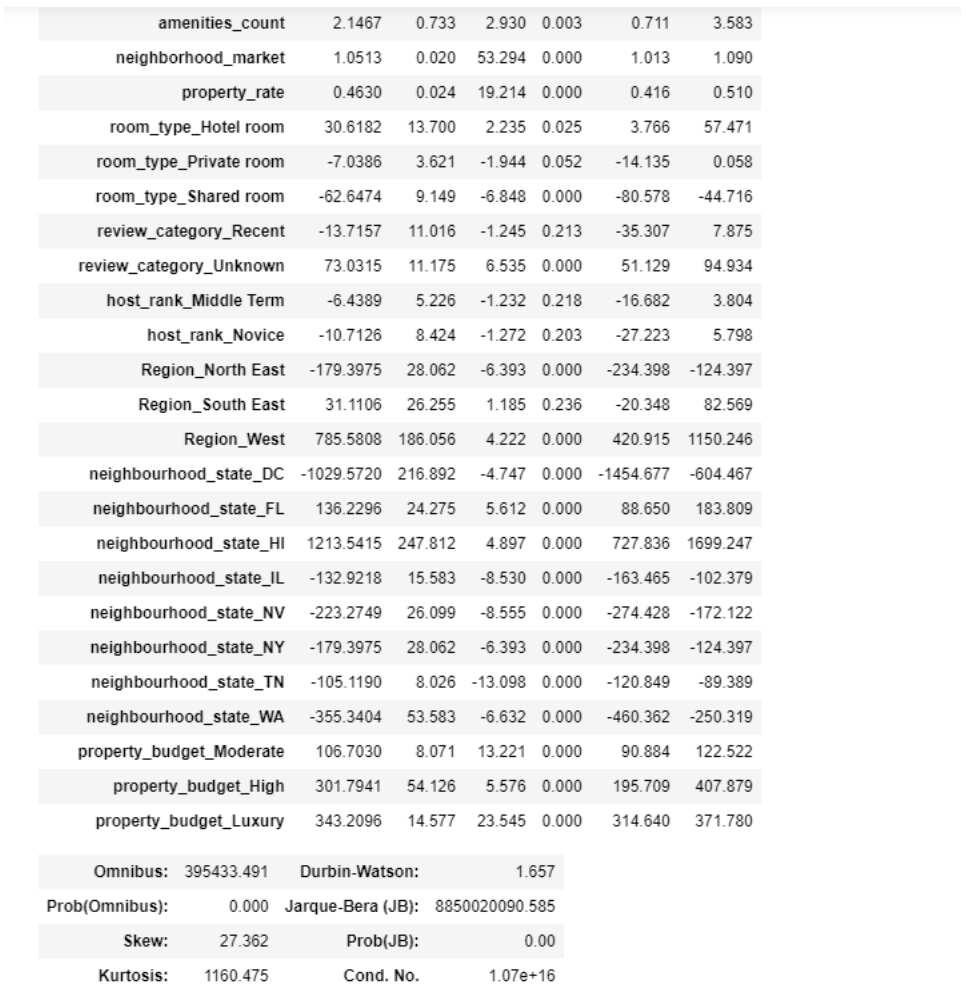
Regression analysis is a powerful statistical method that allows you to examine the relationship between two or more variables of interest. In predictive analytics it can be used to predict a future numerical value of a variable. Linear regression is most suitable for linear data. But it’s very sensitive toward outliers in the data points. The outliers in the data can have a significant impact on the model. There are other non-parametric models available in which normality and homoscedasticity are not required.

But Linear Regression creates a base simple model from which we could able to understand the dataset better. Below is the summary attained through Linear Model.



Dep. Variable:	price	R-squared:	0.194
Model:	OLS	Adj. R-squared:	0.194
Method:	Least Squares	F-statistic:	1031.
Date:	Tue, 20 Dec 2022	Prob (F-statistic):	0.00
Time:	22:07:02	Log-Likelihood:	-1.1905e+06
No. Observations:	158184	AIC:	2.381e+06
Df Residuals:	158146	BIC:	2.381e+06
Df Model:	37		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	504.3721	190.388	2.649	0.008	131.216	877.528
host_is_superhost	4.0946	2.574	1.591	0.112	-0.950	9.139
host_identity_verified	-7.8400	2.898	-2.705	0.007	-13.520	-2.160
instant_bookable	-6.3672	2.435	-2.615	0.009	-11.140	-1.595
accommodates	121.6995	14.498	8.394	0.000	93.284	150.115
latitude	31.1376	5.251	5.930	0.000	20.847	41.429
longitude	21.5939	4.638	4.656	0.000	12.503	30.684
bathrooms	5593.4240	98.997	56.501	0.000	5399.392	5787.456
bedrooms	3207.2856	119.040	26.943	0.000	2973.969	3440.602
beds	-853.6029	76.755	-11.121	0.000	-1004.040	-703.165
minimum_nights	-73.4765	449.599	-0.163	0.870	-954.680	807.727
maximum_nights	-51.4786	448.671	-0.115	0.909	-930.865	827.908
number_of_reviews	20.9220	10.586	1.976	0.048	0.174	41.670
calculated_host_listings_count	-13.1759	10.101	-1.304	0.192	-32.973	6.621
availability_365	-9.5560	3.094	-3.089	0.002	-15.620	-3.492
host_months	-7.6312	13.430	-0.568	0.570	-33.953	18.691
amenities_score	-47.5689	5.392	-8.822	0.000	-58.137	-37.000



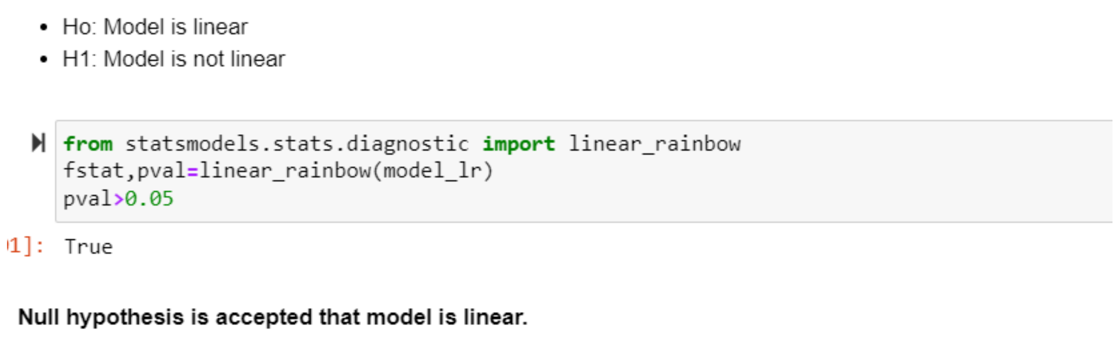
amenities_count	2.1467	0.733	2.930	0.003	0.711	3.583
neighborhood_market	1.0513	0.020	53.294	0.000	1.013	1.090
property_rate	0.4630	0.024	19.214	0.000	0.416	0.510
room_type_Hotel room	30.6182	13.700	2.235	0.025	3.766	57.471
room_type_Private room	-7.0386	3.621	-1.944	0.052	-14.135	0.058
room_type_Shared room	-62.6474	9.149	-6.848	0.000	-80.578	-44.716
review_category_Recent	-13.7157	11.016	-1.245	0.213	-35.307	7.875
review_category_Unknown	73.0315	11.175	6.535	0.000	51.129	94.934
host_rank_Middle Term	-6.4389	5.226	-1.232	0.218	-16.682	3.804
host_rank_Novice	-10.7126	8.424	-1.272	0.203	-27.223	5.798
Region_North East	-179.3975	28.062	-6.393	0.000	-234.398	-124.397
Region_South East	31.1106	26.255	1.185	0.236	-20.348	82.569
Region_West	785.5808	186.056	4.222	0.000	420.915	1150.246
neighbourhood_state_DC	-1029.5720	216.892	-4.747	0.000	-1454.677	-604.467
neighbourhood_state_FL	136.2296	24.275	5.612	0.000	88.650	183.809
neighbourhood_state_HI	1213.5415	247.812	4.897	0.000	727.836	1699.247
neighbourhood_state_IL	-132.9218	15.583	-8.530	0.000	-163.465	-102.379
neighbourhood_state_NV	-223.2749	26.099	-8.555	0.000	-274.428	-172.122
neighbourhood_state_NY	-179.3975	28.062	-6.393	0.000	-234.398	-124.397
neighbourhood_state_TN	-105.1190	8.026	-13.098	0.000	-120.849	-89.389
neighbourhood_state_WA	-355.3404	53.583	-6.632	0.000	-460.362	-250.319
property_budget_Moderate	106.7030	8.071	13.221	0.000	90.884	122.522
property_budget_High	301.7941	54.126	5.576	0.000	195.709	407.879
property_budget_Luxury	343.2096	14.577	23.545	0.000	314.640	371.780

Omnibus:	395433.491	Durbin-Watson:	1.657
Prob(Omnibus):	0.000	Jarque-Bera (JB):	8850020090.585
Skew:	27.362	Prob(JB):	0.00
Kurtosis:	1160.475	Cond. No.	1.07e+16

8.2 Checking Assumptions of Linear Regression

- Linearity
- Normality
- Heteroscedasticity
- Autocorrelation of errors
- Multicollinearity

8.2.1 Linearity



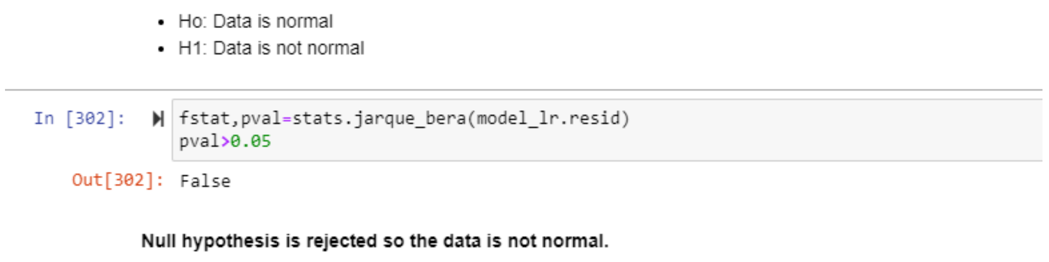
- Ho: Model is linear
- H1: Model is not linear

```
from statsmodels.stats.diagnostic import linear_rainbow
fstat,pval=linear_rainbow(model_lr)
pval>0.05
```

```
1]: True
```

Null hypothesis is accepted that model is linear.

8.2.2 Normality

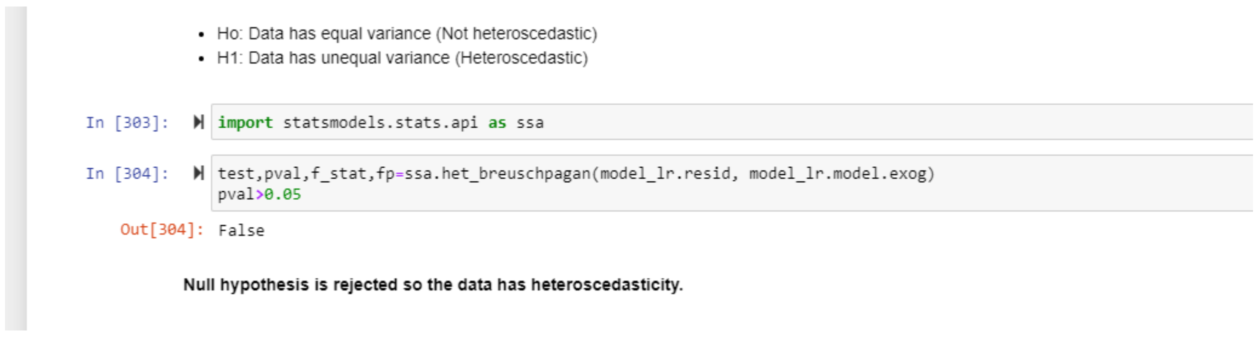


- Ho: Data is normal
- H1: Data is not normal

```
In [302]:  fstat,pval=stats.jarque_bera(model_lr.resid)
           pval>0.05

Out[302]:  False
```

Null hypothesis is rejected so the data is not normal.

8.2.3 Homoscedasticity

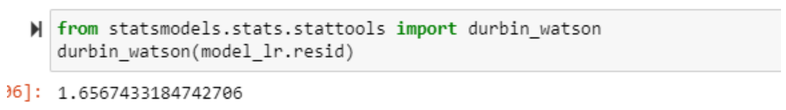


- Ho: Data has equal variance (Not heteroscedastic)
- H1: Data has unequal variance (Heteroscedastic)

```
In [303]:  import statsmodels.stats.api as ssa

In [304]:  test,pval,f_stat,fp=ssa.het_breuschpagan(model_lr.resid, model_lr.model.exog)
           pval>0.05

Out[304]:  False
```

Null hypothesis is rejected so the data has heteroscedasticity.

8.2.4 Auto Correlation of errors

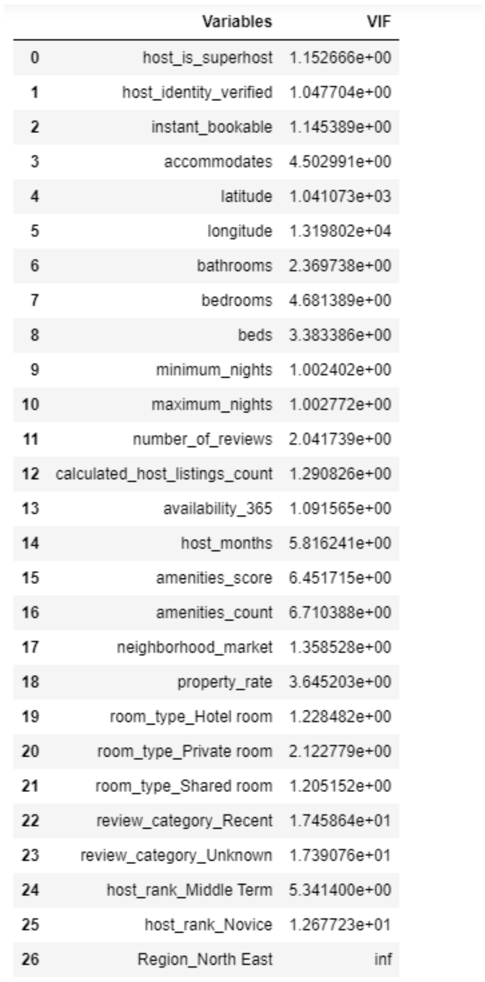


```
from statsmodels.stats.stattools import durbin_watson
durbin_watson(model_lr.resid)
```

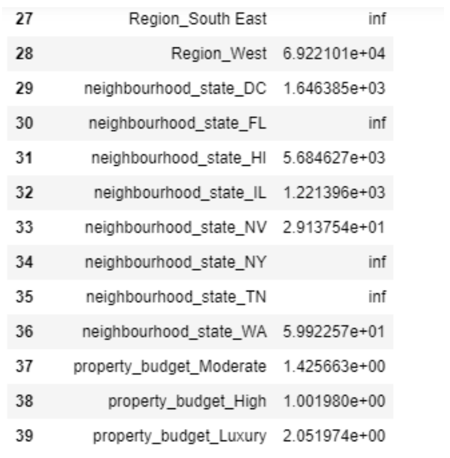
```
)6]: 1.6567433184742706
```

This is an ideal value denoting there is no autocorrelation of errors.

8.2.5 MultiCollinearity



	Variables	VIF
0	host_is_superhost	1.152666e+00
1	host_identity_verified	1.047704e+00
2	instant_bookable	1.145389e+00
3	accommodates	4.502991e+00
4	latitude	1.041073e+03
5	longitude	1.319802e+04
6	bathrooms	2.369738e+00
7	bedrooms	4.681389e+00
8	beds	3.383386e+00
9	minimum_nights	1.002402e+00
10	maximum_nights	1.002772e+00
11	number_of_reviews	2.041739e+00
12	calculated_host_listings_count	1.290826e+00
13	availability_365	1.091565e+00
14	host_months	5.816241e+00
15	amenities_score	6.451715e+00
16	amenities_count	6.710388e+00
17	neighborhood_market	1.358528e+00
18	property_rate	3.645203e+00
19	room_type_Hotel room	1.228482e+00
20	room_type_Private room	2.122779e+00
21	room_type_Shared room	1.205152e+00
22	review_category_Recent	1.745864e+01
23	review_category_Unknown	1.739076e+01
24	host_rank_Middle Term	5.341400e+00
25	host_rank_Novice	1.267723e+01
26	Region_North East	inf



27	Region_South East	inf
28	Region_West	6.922101e+04
29	neighbourhood_state_DC	1.646385e+03
30	neighbourhood_state_FL	inf
31	neighbourhood_state_HI	5.684627e+03
32	neighbourhood_state_IL	1.221396e+03
33	neighbourhood_state_NV	2.913754e+01
34	neighbourhood_state_NY	inf
35	neighbourhood_state_TN	inf
36	neighbourhood_state_WA	5.992257e+01
37	property_budget_Moderate	1.425663e+00
38	property_budget_High	1.001980e+00
39	property_budget_Luxury	2.051974e+00

Region , neighbourhood states , review category, latitude , longitude all columns are having multi collinearity

8.3 Regularization Models :

8.3.1 Ridge Regression:

Ridge Regression is used for the analysis of multicollinearity in multiple regression data. It is preferred when a data set consists of a higher number of predictor variables and also preferred when there is a multicollinearity found in the data set.

Observations:

Root Mean Squared Error (RMSE): 473.288

Mean Absolute Error (MAE): 108.105

Mean Absolute Percentage Error (MAPE): 0.675

8.3.2 Lasso Regression:

Lasso is a modification of a linear regression, where the model is penalized for the sum of absolute values of weights. Hence the absolute values of weights will be reduced and many will tend to zero. It is a regularization method used to reduce the overfitting of the model.

Observations:

Root Mean Squared Error (RMSE): 516.698

Mean Absolute Error (MAE): 153.057

Mean Absolute Percentage Error (MAPE): 1.301

8.3.3 Elastic Net Regression

Elastic Net combines the feature elimination from lasso and feature coefficient reduction from the Ridge model to improve the model’s predictions.

Observations:

Root Mean Squared Error (RMSE): 516.698

Mean Absolute Error (MAE): 153.057

Mean Absolute Percentage Error (MAPE): 1.301

8.4 Robust Models

8.4.1 Huber regression

Huber Regression is an example of a robust regression algorithm that assigns less weight to observations identified as outliers.

Observations:

Root Mean Squared Error (RMSE): 492.551

Mean Absolute Error (MAE): 99.227

Mean Absolute Percentage Error (MAPE): 0.435

8.4.2 RANSAC

Random Sample Consensus (RANSAC) regression is a non-deterministic algorithm that tries to separate the training data into inliers (which may be subject to noise) and outliers. Then, it estimates the final model only using the inliers.

Observations:

Root Mean Squared Error (RMSE): 512.587

Mean Absolute Error (MAE): 121.409

Mean Absolute Percentage Error (MAPE): 0.606

8.4.3 Theil-Sen Regression

Theil-Sen regression is a non-parametric regression method, which means that it makes no assumption about the underlying data distribution. In short, it involves fitting multiple regression models on subsets of the training data and then aggregating the coefficients at the last step.

Observations:

Root Mean Squared Error (RMSE): 854.409

Mean Absolute Error (MAE): 279.742

Mean Absolute Percentage Error (MAPE): 2.699

8.5 Decision Trees:

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches and a leaf node represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

Observations:

Root Mean Squared Error (RMSE): 579.908

Mean Absolute Error (MAE): 110.160

Mean Absolute Percentage Error (MAPE): 0.551

8.6 Ensemble Models

8.6.1 Random Forest:

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of over fitting to their training set.

Observations:

Root Mean Squared Error (RMSE): 431.587

Mean Absolute Error (MAE): 84.664

Mean Absolute Percentage Error (MAPE): 0.46

8.6.2 Bagging

Bagging is also known as bootstrap aggregation which means that the ensemble learning method is commonly used to reduce variance within a noisy dataset. In bagging, a random sample of data in training set is selected with replacement and outputting the class that is the mode of the classes (classification) or mean prediction (regression)

Observations:

Root Mean Squared Error (RMSE): 452.632

Mean Absolute Error (MAE): 91.281

Mean Absolute Percentage Error (MAPE): 0.492

8.6.3 Ada Boost:

Adaptive Boosting, or most commonly known AdaBoost, is a Boosting algorithm. This algorithm uses the method to correct its predecessor. Thus, at every new predictor the focus is more on the complicated cases more than the others.

It fits a sequence of weak learners on different weighted training data. It starts by predicting the original data set and gives equal weight to each observation. If prediction is incorrect using the first learner, then it gives higher weight to observation which have been predicted incorrectly. Being an 24 iterative process, it continues to add learner(s) until a limit is reached in the number of models or accuracy.

Observations:

Root Mean Squared Error (RMSE): 858.941

Mean Absolute Error (MAE): 678.790

Mean Absolute Percentage Error (MAPE): 7.528

8.6.4 Gradient Boosting

Gradient Boosting regression calculates the difference between the current prediction and the known target value. This difference is called residual. After that the Gradient boosting trains a weak model that maps feature to that residual.

Observations:

Root Mean Squared Error (RMSE): 858.945

Mean Absolute Error (MAE): 678.790

Mean Absolute Percentage Error (MAPE): 7.528

8.6.5 XG Boost:

XG Boost or Extreme Gradient Boosting is an advanced implementation of the Gradient Boosting. This algorithm has high predictive power and is ten times faster than any other gradient boosting techniques. Moreover, it includes a variety of regularization which reduces overfitting and improves overall performance. XG Boost has an in-built routine to handle missing values.

Observations:

Root Mean Squared Error (RMSE): 399.131

Mean Absolute Error (MAE): 88.544

Mean Absolute Percentage Error (MAPE): 0.503

8.6.6 Stacking Model:

Model Stacking is a way to improve model predictions by combining the outputs of multiple models and running them through another machine learning model called meta-learner.

For our stacking model we have considered Random Forest, Huber Regressor and XG Boost models as base model and the final model is based on random forest.

Observations:

Root Mean Squared Error (RMSE): 401.633

Mean Absolute Error (MAE): 87.313

Mean Absolute Percentage Error (MAPE): 0.469

8.6.7 Random Forest Tuned:

Using Grid Search CV we passed certain parameters and identified the best parameters as follows.

- **max_depth**: None
- **max_features**: 0.2
- **min_samples_leaf**: 1
- **min_samples_split**: 2
- **n_estimators**: 300

From the above best parameters identified we have fitted a random forest model and o -bserved below.

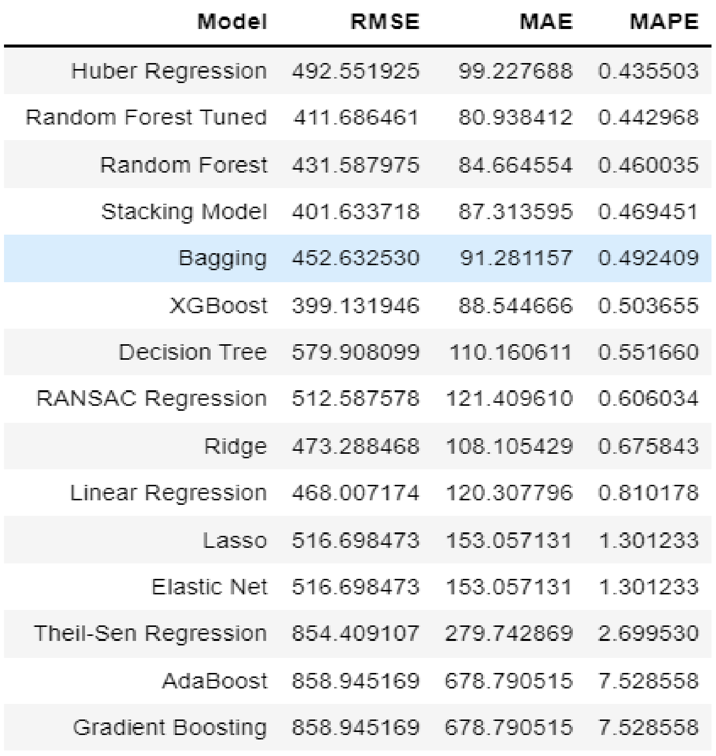
Observations:

Root Mean Squared Error (RMSE): 411.686

Mean Absolute Error (MAE): 80.938

Mean Absolute Percentage Error (MAPE): 0.442

9. Comparison of models:



Model	RMSE	MAE	MAPE
Huber Regression	492.551925	99.227688	0.435503
Random Forest Tuned	411.686461	80.938412	0.442968
Random Forest	431.587975	84.664554	0.460035
Stacking Model	401.633718	87.313595	0.469451
Bagging	452.632530	91.281157	0.492409
XGBoost	399.131946	88.544666	0.503655
Decision Tree	579.908099	110.160611	0.551660
RANSAC Regression	512.587578	121.409610	0.606034
Ridge	473.288468	108.105429	0.675843
Linear Regression	468.007174	120.307796	0.810178
Lasso	516.698473	153.057131	1.301233
Elastic Net	516.698473	153.057131	1.301233
Theil-Sen Regression	854.409107	279.742869	2.699530
AdaBoost	858.945169	678.790515	7.528558
Gradient Boosting	858.945169	678.790515	7.528558

1. A total of 15 base models are built for prediction.

2. Since we tend to proceed model building with outliers we prioritize the metrics MAE and MAPE.

3. The reason is because MAE and MAPE deal with the median score which is robust to outliers compared to RMSE which is sensitive to them.

4. The top three models on inferring the MAE and MAPE score are Random forest regression model, Stacking regression model and Huber regression Model.

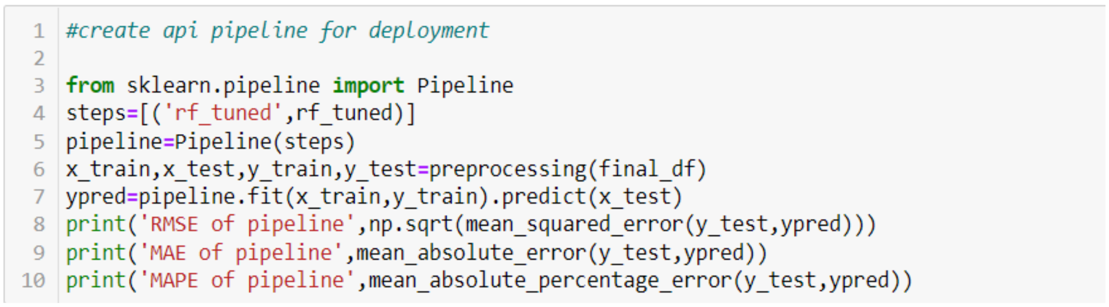
5. Out of these three we found Random forest model to be promising and providing a generalized and ideal MAPE value 0.45 approx.

6. On tuning the Random forest model through GridSearchCV we rebuild and find the MAPE valueto be 0.43 approx which shows a minute improvement.

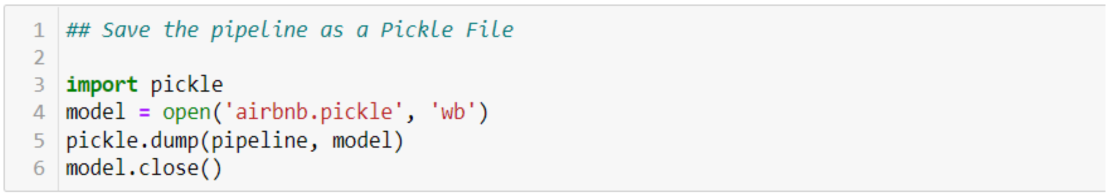
10. Model deployment

As we have trained our model, now we need to use that model to predict the price with the details taken from the new users. We will deploy our model and will design a web application where the user will input all the attribute values and the data will be given into the model, based on the training given to the model, the model will predict the optimal rent price for the listings with the details fed into the system.

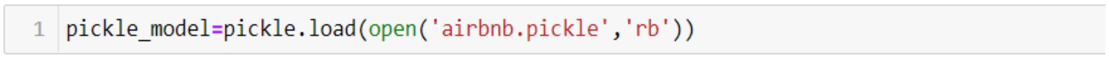
For our model deployment we have created a pipeline for the api using python pipeline library and have loaded the model as a pickle file into the pipeline.



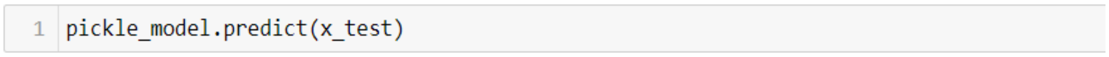
```python
#create api pipeline for deployment

from sklearn.pipeline import Pipeline
steps=[('rf_tuned',rf_tuned)]
pipeline=Pipeline(steps)
x_train,x_test,y_train,y_test=preprocessing(final_df)
ypred=pipeline.fit(x_train,y_train).predict(x_test)
print('RMSE of pipeline',np.sqrt(mean_squared_error(y_test,ypred)))
print('MAE of pipeline',mean_absolute_error(y_test,ypred))
print('MAPE of pipeline',mean_absolute_percentage_error(y_test,ypred))
```



```python
## Save the pipeline as a Pickle File

import pickle
model = open('airbnb.pickle', 'wb')
pickle.dump(pipeline, model)
model.close()
```

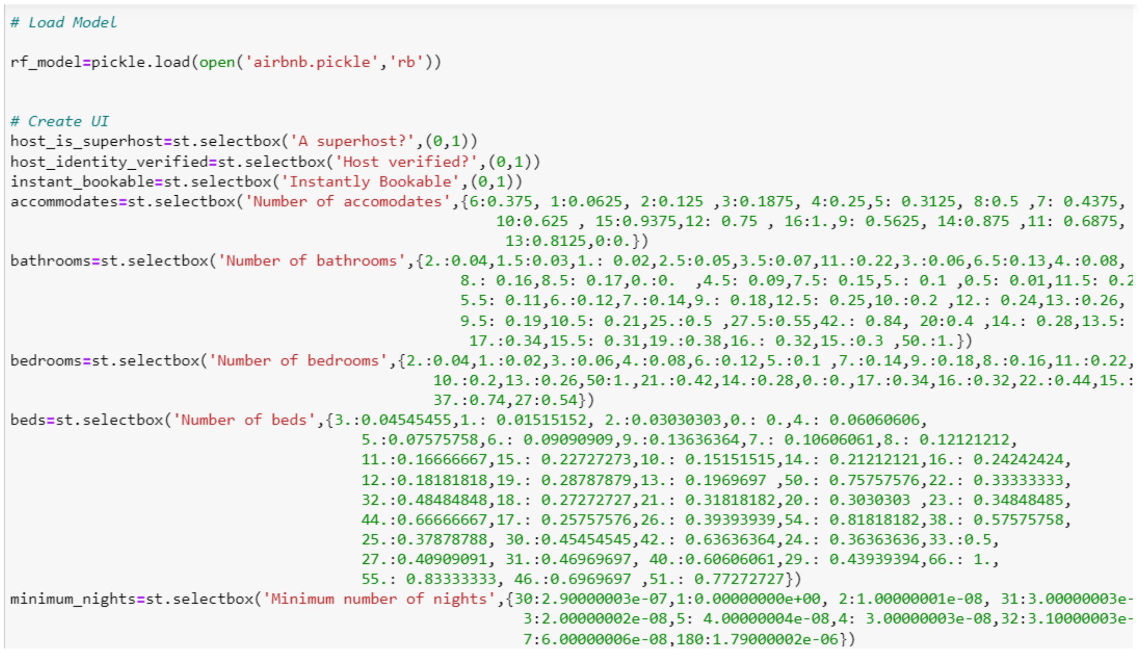


```python
pickle_model=pickle.load(open('airbnb.pickle','rb'))
```

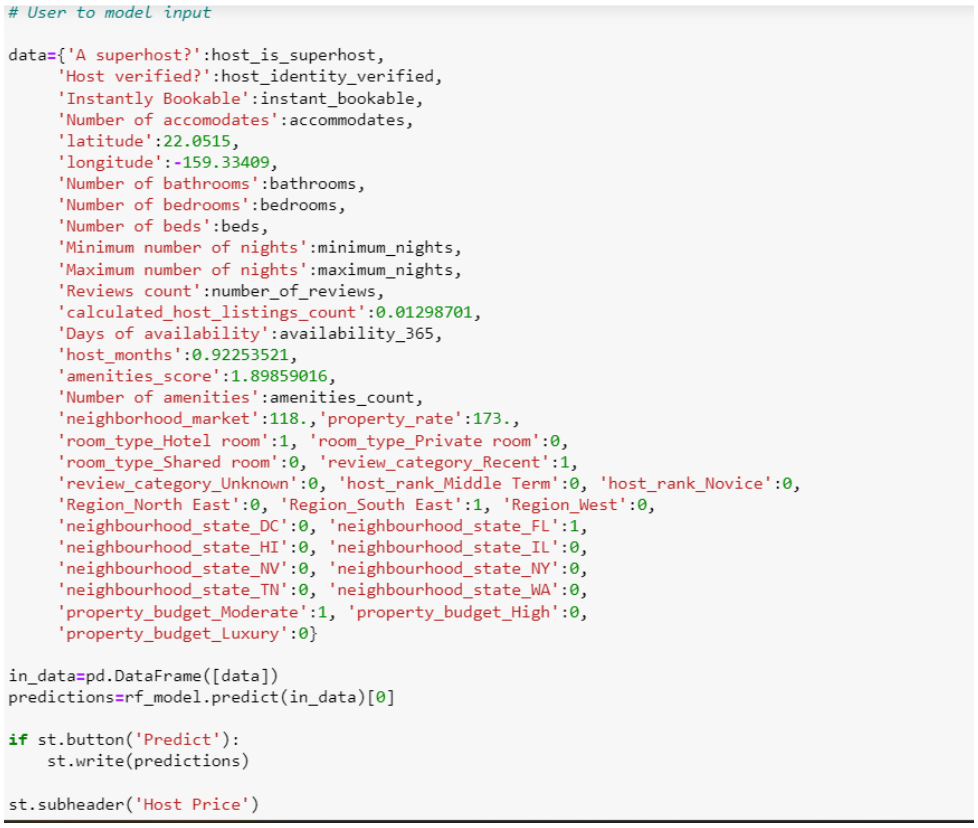


```python
pickle_model.predict(x_test)
```

Streamlit is an open source python library which helps in creating and sharing custom web apps for machine learning and data science. Using that we have created our API.

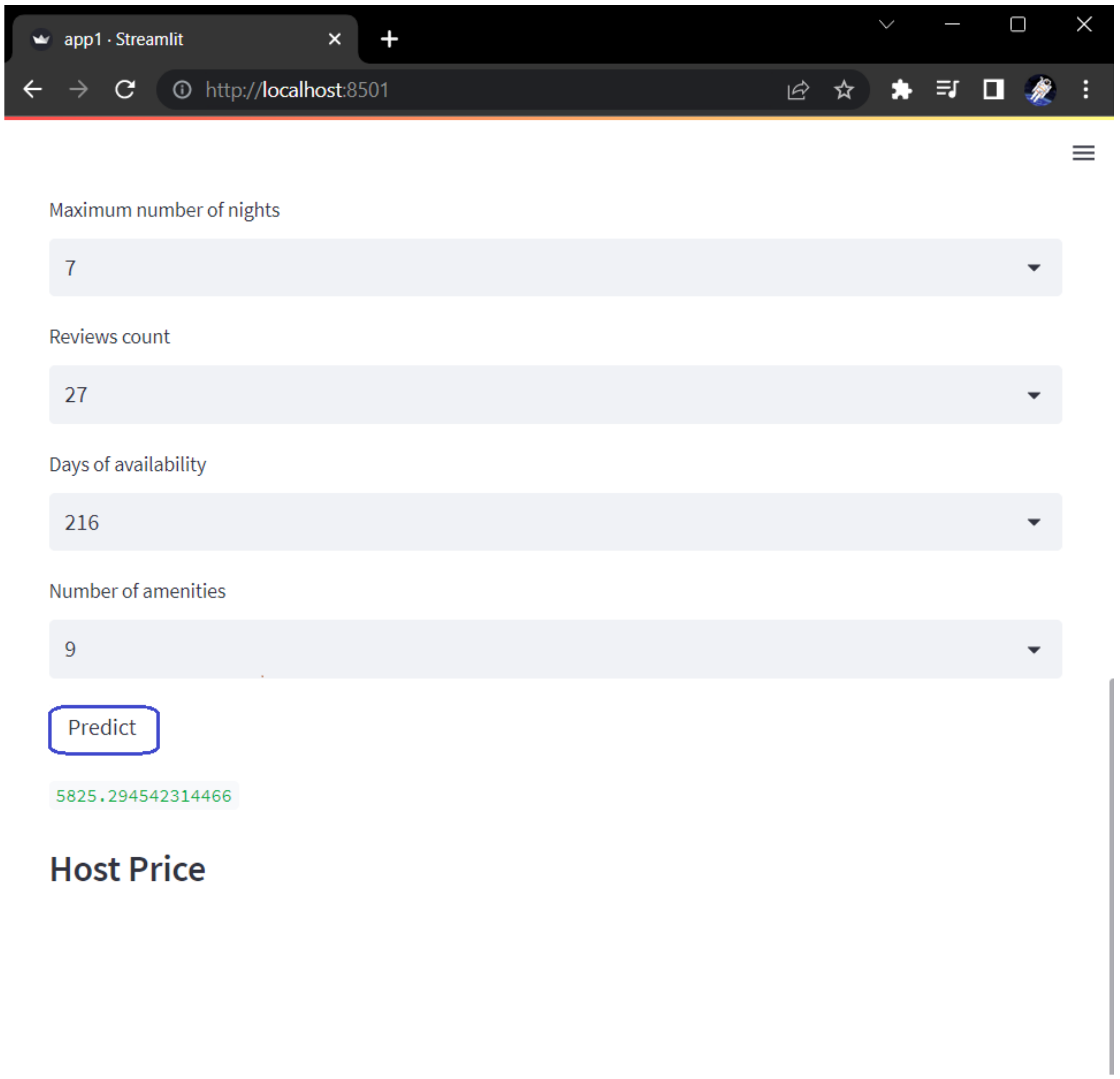


```
# Load Model

rf_model=pickle.load(open('airbnb.pickle','rb'))


# Create UI
host_is_superhost=st.selectbox('A superhost?',(0,1))
host_identity_verified=st.selectbox('Host verified?',(0,1))
instant_bookable=st.selectbox('Instantly Bookable',(0,1))
accommodates=st.selectbox('Number of accomodates',{6:0.375, 1:0.0625, 2:0.125 ,3:0.1875, 4:0.25,5: 0.3125, 8:0.5 ,7: 0.4375,
                                                    10:0.625 , 15:0.9375,12: 0.75 , 16:1.,9: 0.5625, 14:0.875 ,11: 0.6875,
                                                     13:0.8125,0:0.})
bathrooms=st.selectbox('Number of bathrooms',{2.:0.04,1.5:0.03,1.: 0.02,2.5:0.05,3.5:0.07,11.:0.22,3.:0.06,6.5:0.13,4.:0.08,
                                                8.: 0.16,8.5: 0.17,0.:0.  ,4.5: 0.09,7.5: 0.15,5.: 0.1 ,0.5: 0.01,11.5: 0.2
                                                5.5: 0.11,6.:0.12,7.:0.14,9.: 0.18,12.5: 0.25,10.:0.2 ,12.: 0.24,13.:0.26,
                                                9.5: 0.19,10.5: 0.21,25.:0.5 ,27.5:0.55,42.: 0.84, 20:0.4 ,14.: 0.28,13.5:
                                                 17.:0.34,15.5: 0.31,19.:0.38,16.: 0.32,15.:0.3 ,50.:1.})
bedrooms=st.selectbox('Number of bedrooms',{2.:0.04,1.:0.02,3.:0.06,4.:0.08,6.:0.12,5.:0.1 ,7.:0.14,9.:0.18,8.:0.16,11.:0.22,
                                             10.:0.2,13.:0.26,50:1.,21.:0.42,14.:0.28,0.:0.,17.:0.34,16.:0.32,22.:0.44,15.:
                                             37.:0.74,27:0.54})
beds=st.selectbox('Number of beds',{3.:0.04545455,1.: 0.01515152, 2.:0.03030303,0.: 0.,4.: 0.06060606,
                                     5.:0.07575758,6.: 0.09090909,9.:0.13636364,7.: 0.10606061,8.: 0.12121212,
                                     11.:0.16666667,15.: 0.22727273,10.: 0.15151515,14.: 0.21212121,16.: 0.24242424,
                                     12.:0.18181818,19.: 0.28787879,13.: 0.1969697 ,50.: 0.75757576,22.: 0.33333333,
                                     32.:0.48484848,18.: 0.27272727,21.: 0.31818182,20.: 0.3030303 ,23.: 0.34848485,
                                     44.:0.66666667,17.: 0.25757576,26.: 0.39393939,54.: 0.81818182,38.: 0.57575758,
                                     25.:0.37878788, 30.:0.45454545,42.: 0.63636364,24.: 0.36363636,33.:0.5,
                                     27.:0.40909091, 31.:0.46969697, 40.:0.60606061,29.: 0.43939394,66.: 1.,
                                     55.: 0.83333333, 46.:0.6969697 ,51.: 0.77272727})
minimum_nights=st.selectbox('Minimum number of nights',{30:2.90000003e-07,1:0.00000000e+00, 2:1.00000001e-08, 31:3.00000003e-
                                                         3:2.00000002e-08,5: 4.00000004e-08,4: 3.00000003e-08,32:3.10000003e-
                                                         7:6.00000006e-08,180:1.79000002e-06})
```



```
# User to model input

data={'A superhost?':host_is_superhost,
     'Host verified?':host_identity_verified,
     'Instantly Bookable':instant_bookable,
     'Number of accomodates':accommodates,
     'latitude':22.0515,
     'longitude':-159.33409,
     'Number of bathrooms':bathrooms,
     'Number of bedrooms':bedrooms,
     'Number of beds':beds,
     'Minimum number of nights':minimum_nights,
     'Maximum number of nights':maximum_nights,
     'Reviews count':number_of_reviews,
     'calculated_host_listings_count':0.01298701,
     'Days of availability':availability_365,
     'host_months':0.92253521,
     'amenities_score':1.89859016,
     'Number of amenities':amenities_count,
     'neighborhood_market':118.,'property_rate':173.,
     'room_type_Hotel room':1, 'room_type_Private room':0,
     'room_type_Shared room':0, 'review_category_Recent':1,
     'review_category_Unknown':0, 'host_rank_Middle Term':0, 'host_rank_Novice':0,
     'Region_North East':0, 'Region_South East':1, 'Region_West':0,
     'neighbourhood_state_DC':0, 'neighbourhood_state_FL':1,
     'neighbourhood_state_HI':0, 'neighbourhood_state_IL':0,
     'neighbourhood_state_NV':0, 'neighbourhood_state_NY':0,
     'neighbourhood_state_TN':0, 'neighbourhood_state_WA':0,
     'property_budget_Moderate':1, 'property_budget_High':0,
     'property_budget_Luxury':0}

in_data=pd.DataFrame([data])
predictions=rf_model.predict(in_data)[0]

if st.button('Predict'):
    st.write(predictions)

st.subheader('Host Price')
```

With the backend imputation of models are done using streamlit we have deployed our final API as follows. So when a new user comes and fill up the criteria it will help them in predicting the optimal price based on the data they have filled up in the webpage. Below is the sample snippet taken .



Maximum number of nights

7

Reviews count

27

Days of availability

216

Number of amenities

9

```
5825.294542314466
```

Host Price

11. Business Interpretation

With the model created we can identify the important features which impacts the price of the listings directly. So focusing on those features may help in predicting the optimal price more accurate.

1. The type of amenities plays a major role in price of the listings. So the listings having variety of most important amenities puts the market price of those listing higher.
2. Based on the property type whether the property is an apartment, barn , boat house, Hotel rooms or shared rooms the price range varies.Entire room/apt has the higher mean price across all property types compared to a private room and shared room.
   The private room and Entire home/apt represent the core of the business, with prices in different ranges.
3. Neighbourhood also helps in prediction of price range for the listings. For example , Hawaii is an tourist attraction spot so there are high chances of prices being high whereas less common neighbourhoods are having lower price range for listings.
4. The older the host property helps in defining their reputation, so that those properties tend to have higher price range.
5. The number of accommodates also a major feature in price range. The more the number of accommodates the higher the price for that listings and vice versa.

12. Conclusion

The final model that we have adopted will help predict the optimal price for the new host listings any added into the system. Although the model predicts an optimal price there is still need of improvement of the models to increase the performances.

A bunch of different variables that are not included could play an major role in increasing the performance. For example, given the importance of customer reviews of the listing in determining price, a better understanding of the reviews could very likely improve the prediction. One could use Sentiment Analysis. A score between very negative sentiment and very positive sentiment can be assigned to each review per listing property. The scores are then averaged across all the reviews associated with that listing and the final scores can be included as a new feature.

We could also include image quality as a feature. Using deep learning and supervised learning analyses on an Airbnb panel dataset, researchers found that units with verified photos (taken by Airbnb’s photographers) generate additional revenue per year on average. So adding Image features may provide an increased advantage in the price prediction.

References :

- https://www.susanacharm.com/airbnb-project
- https://iide.co/case-studies/swot-analysis-of-airbnb/
- https://www.hostfully.com/blog/airbnb-pricing-strategies/
- https://developer.nvidia.com/blog/dealing-with-outliers-using-three-robust-linear-regression-models/
- https://practicaldatascience.co.uk/machine-learning/how-to-save-and-load-machine-learning-models-using-pickle