Airbnb LISTINGS PRICE PREDICTION

Post Graduate Program in Data Science Engineering

Location: **Bangalore** Batch: **PGPDSE-FT Nov21**

Submitted by

Chirayu Kale,
Pawan Balijireddi,
Praveen NM,
V Raghvendra,
Vuppili Mourya

Mentored by

Mr. Ashish Sharma

Table of Contents

Project Details

OVERVIEW

Airbnb is a home-sharing platform that allows home-owners and renters (‘hosts’) to put their properties (‘listings’) online, so that guests can pay to stay in them. Hosts are expected to set their own prices for their listings. Although Airbnb and other sites provide some general guidance, there are currently no free and accurate services which help hosts price their properties using a wide range of data points.

The objective of this Capstone project is to predict the prices for listings which will help the ‘hosts’ to determine the prices for their listings and the guests to choose to an appropriate accommodation according to their need and budget.

BUSINESS PROBLEM STATEMENT

1. Business Problem Understanding

As of now, there are various paid third-party pricing software available in the market, but generally the hosts are required to put in their 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.

Airbnb pricing is important to get right, particularly in big cities like New York where there is lots of competition and even small differences in prices can make a big difference. It is also a difficult thing to do correctly — **price too high and no one will book. Price too low and hosts be missing out on a lot of potential income.**

2. Business Objective – price prediction for Airbnb listings

- The objective is to device a model to predict the price of the listings in accordance to help the hosts determine the prices for their properties and the model will be equally helpful for the customers as well. The information on range of prices of the listings will be beneficial for the organization also to determine the areas to be endorsed on their website mobile applications, which are presently generating low revenue for the hosts and the organization.

3. **Methodology to be followed:**

CRISP-DM which stands for Cross Industry Standard Process for Data Mining is a methodology created to help shape data mining projects. It describes the different phases/tasks involved in the project and provides an overview of data mining life cycle.

❖ **Business Understanding** - It focuses on determining the business requirements/objectives and understanding what outcome to achieve. Also determine the business units being affected. Convert this business problem into a data mining problem and carve out an initial plan.

• Determine the business objectives: Understand what is needed to be accomplished for the customer.

• Assess situation: Determine resources availability, project requirements, assess risks and contingencies, and conduct a cost-benefit analysis.

• Determine data mining goals: Convert business problem to a data mining problem and recognize the data mining problem type such as classification, regression or clustering, etc.

• Produce a project plan: Devise a step-to-step plan for executing the project.

❖ **Data understanding** - This phase starts with collecting the data and then examining the data for its surface properties like data format, number of records, etc. The next step is to better understand the data by understanding each attribute and perform basic statistics on them. Understand the relationship between different attributes. Determine the quality of data by checking the missing values, outliers, duplicates, etc.

• Collect initial data: Acquire the data and load it into the analysis tool to be used.

• Describe data: Examine the data and document its surface properties like data format, number of records, or field identities. Understand the meaning of each attribute and attribute value in business terms. For each attribute, compute basic statistics so as to get a higher-level understanding.

• Explore data: Find insights from the data. Query it, visualize it, and identify relationships among the data.

• Verify data quality: Identify special values, missing attributes and null data. Determine how clean/dirty is the data.

- **Data Pre-Processing** - This stage, which is often referred to as data wrangling, has the objective to develop the final data set for EDA and modelling. Covers all activities to construct the final dataset from the initial raw data. Some of the tasks include table, record and attribute selection as well as transformation and cleaning of data for modelling tools.

• Select data: Determine which attributes/features will be used and document reasons for inclusion/exclusion.

• Clean data: Correct, impute and remove the improper data.

• Extract data: Derive new attributes from the existing ones

• Integrate data: Create features by combining data from multiple sources.

• Format data: Re-format data as necessary. For example, convert string values to numeric values so as to perform mathematical operations.

- **Modelling -** In this stage we build and assess different models built using various techniques from the training dataset.

• Select modelling technique: Determine the algorithms to be used to model the data based on the business requirement.

• Generate test design: In order to build and test the model, we need to divide the dataset into training and testing data set. In this step we divide the data into train and test data set.

• Build model: Based on the modelling technique selected, build the model on the input data set.

• Assess model: Compare the results of different models based on confusion matrix. The outcome of this step frequently leads to model tuning iterations until the best model is found.

- **Evaluation -** Evaluate the models and review the steps executed to construct the model to be certain it properly achieves the business objectives.

• Evaluate results: Understand the data mining results and check how impactful they are in achieving the data mining goal. Select appropriate model based on confusion matrix.

• Review process: Review the work accomplished and make sure that nothing was overlooked and all steps were properly executed. Summarize the findings and correct anything if needed.
• Determine next steps: Based on the previous three tasks, determine whether to proceed to deployment, iterate further, or initiate new projects.

Linear Regression – OLS

Conclusions

By implementing the resultant models built using the above methods, we can suggest the prices for the properties listed and can determine the trends that fluctuate the pricings.

TOPIC SURVEY

1. Problem understanding

Pricing a property and evaluating the proposed price for a property are challenges that, respectively, owners and customers of Airbnb rentals face on a daily basis. This project aims to create a model for predicting the price of an Airbnb listing using property specifications, owner information, and customer reviews for the listing. Owners and customers can use the resulting model to estimate the expected value of an Airbnb listing. Regression models will be trained and tuned on a dataset of Airbnb listings from New York city, and the resulting models will be compared in terms of Mean Squared Error, Mean Absolute Error, and R2 score.

2. Current solution to the problem

As of now, there are various paid third-party pricing software available in the market, but generally the hosts are required to put in their 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. We aim to create a more reliable model which will be beneficial for both the hosts, customers and the organization.

3. Proposed solution to the problem

In order to the price prediction for listed properties, in this project we will several listing features to try and predict price, we will see how the factors like neighborhood and other feature have an impact on the pricing. This will allow our model to ascertain some factors like hosts, room type or the reviews to be the main drivers of the fluctuations in the pricing.

To implement the said approach, we carry out Regression analysis models to predict the appropriate day the ticket has to be allocated to an associate from the date of creation.

Regression – OLS

CRITICAL ASSESSMENT OF TOPIC SURVEY

Find the key area, gaps identified in the topic survey where the project can add value to the customers and business.

1. What key gaps are you trying to solve?

On the Airbnb accommodation booking platform, the guests get to choose three types of accommodation: an entire house/ apartment, a private room (often with shared facilities), or a shared room. Costs saving, value for money, and a drive for community are confirmed as motivators for the use of such P2P accommodation

The mutual review system of hosts and guests is seen as the foundation of trust in Airbnb transactions, even though precisely the reciprocity of the system is considered to undermine its reliability. The users refer to this study are the hosts and their properties as their listings. Each host is associated with a set of attributes including a photo, a personal statement, their listings, guest reviews of their properties, and Airbnb certified contact information. Similarly, each listing displays attributes including location, price, a brief description, photos, capacity, availability, check-in and checkout times, cleaning fees and security deposits. Airbnb describes itself as “a trusted community marketplace for people to list, discovers, and book unique accommodation around the world". Prospective hosts list their spare rooms or apartments on the Airbnb platform; establish their own nightly, weekly or monthly price; and offer accommodation to the guests. Airbnb derives revenue from both guests and hosts for this service: guests pay a 9% to 12% service fee for each reservation they make, depending on the length of their stay, and hosts pay a 3% service fee to cover the cost of processing payments.

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 techniques to aid both the property owners and the customers with price evaluation given minimal available information about the property. Features of the rentals, owner characteristics, and the customer reviews can be the potential predictors in the study, and a range of methods from regression can be used for creating the prediction model. Some of the questions we aim to address through this project are, how do prices of listings vary by location? What localities in NYC are rated highly by guests? How does the demand for Airbnb rentals fluctuate across the year? What are the different types of properties in NYC? Do they vary by neighborhood?

There are various factors which can be the key influencers in determining the pricing for the listings, currently our dataset facilitates with the below mentioned features which we classify as independent variables of our project.

Data Dictionary

Sr. No.	Variable	Datatype	Description
1	id	integer	Airbnb's unique identifier for the listing
2	listing_url	text	
3	scrape_id	bigint	Inside Airbnb "Scrape" this was part of
4	last_scraped	datetime	UTC. The date and time this listing was "scraped".
5	name	text	Name of the listing
6	description	text	Detailed description of the listing

7	neighborhood_overview	text	Host's description of the neighbourhood
8	picture_url	text	URL to the Airbnb hosted regular sized image for the listing
9	host_id	integer	Airbnb's unique identifier for the host/user
10	host_url	text	The Airbnb page for the host
11	host_name	text	Name of the host. Usually just the first name(s).
12	host_since	date	The date the host/user was created. For hosts that are Airbnb guests this could be the date they registered as a guest.
13	host_location	text	The host's self reported location
14	host_about	text	Description about the host
15	host_response_time		
16	host_response_rate		
17	host_acceptance_rate		That rate at which a host accepts booking requests.
18	host_is_superhost	boolean [t=true; f=false]	
19	host_thumbnail_url	text	
20	host_picture_url	text	
21	host_neighbourhood	text	
22	host_listings_count	text	The number of listings the host has (per Airbnb calculations)
23	host_total_listings_count	text	The number of listings the host has (per Airbnb calculations)
24	host_verifications		
25	host_has_profile_pic	boolean [t=true; f=false]	
26	host_identity_verified	boolean [t=true; f=false]	
27	neighbourhood	text	
28	neighbourhood_cleansed	text	The neighbourhood as geocoded using the latitude and longitude against neighborhoods as defined by open or public digital shapefiles.
29	neighbourhood_group_cleansed	text	The neighbourhood group as geocoded using the latitude and longitude against neighborhoods as defined by open or public digital shapefiles.
30	latitude	numeric	Uses the World Geodetic System (WGS84) projection for latitude and longitude.

31	longitude	numeric	Uses the World Geodetic System (WGS84) projection for latitude and longitude.
32	property_type	text	Self selected property type. Hotels and Bed and Breakfasts are described as such by their hosts in this field
33	room_type	text	[Entire home/apt|Private room|Shared room|Hotel] All homes are grouped into the following three room types: Entire place Private room Shared room Entire place Entire places are best if you're seeking a home away from home. With an entire place, you'll have the whole space to yourself. This usually includes a bedroom, a bathroom, a kitchen, and a separate, dedicated entrance. Hosts should note in the description if they'll be on the property or not (ex: "Host occupies first floor of the home"), and provide further details on the listing. Private rooms Private rooms are great for when you prefer a little privacy, and still value a local connection. When you book a private room, you'll have your own private room for sleeping and may share some spaces with others. You might need to walk through indoor spaces that another host or guest may occupy to get to your room. Shared rooms Shared rooms are for when you don't mind sharing a space with others. When you book a shared room, you'll be sleeping in a space that is shared with others and share the entire space with other people. Shared rooms are

			popular among flexible travelers looking for new friends and budget-friendly stays.
34	accommodates	integer	The maximum capacity of the listing
35	bathrooms	numeric	The number of bathrooms in the listing
36	bathrooms_text	string	The number of bathrooms in the listing. On the Airbnb web-site, the bathrooms field has evolved from a number to a textual description. For older scrapes, bathrooms is used.
37	bedrooms	integer	The number of bedrooms
38	beds	integer	The number of bed(s)
39	amenities	json	
40	minimum_nights	integer	minimum number of night stay for the listing (calendar rules may be different)

41	maximum_nights	integer	maximum number of night stay for the listing (calendar rules may be different)
42	minimum_minimum_nights	integer	the smallest minimum_night value from the calender (looking 365 nights in the future)
43	maximum_minimum_nights	integer	the largest minimum_night value from the calender (looking 365 nights in the future)
44	minimum_maximum_nights	integer	the smallest maximum_night value from the calender (looking 365 nights in the future)
45	maximum_maximum_nights	integer	the largest maximum_night value from the calender (looking 365 nights in the future)
46	minimum_nights_avg_ntm	numeric	the average minimum_night value from the calender (looking 365 nights in the future)
47	maximum_nights_avg_ntm	numeric	the average maximum_night value from the calender (looking 365 nights in the future)
48	calendar_updated	date	
49	has_availability	boolean	[t=true; f=false]
50	availability_30	integer	avaliability_x. The availability of the listing x days in the future as determined by the calendar. Note a listing may not be available because it has been booked by a guest or blocked by the host.
51	availability_60	integer	avaliability_x. The availability of the listing x days in the future as determined by the calendar. Note a listing may not be available because it has been booked by a guest or blocked by the host.
52	availability_90	integer	avaliability_x. The availability of the listing x days in the future as determined by the calendar. Note a listing may not be available because it has been booked by a guest or blocked by the host.
53	availability_365	integer	avaliability_x. The availability of the listing x days in the future as determined by the calendar. Note a listing may not be available because it has been booked by a guest or blocked by the host.
54	calendar_last_scraped	date	
55	number_of_reviews	integer	The number of reviews the listing has

56	number_of_reviews_ltm	integer	The number of reviews the listing has (in the last 12 months)
57	number_of_reviews_l30d	integer	The number of reviews the listing has (in the last 30 days)
58	first_review	date	The date of the first/oldest review
59	last_review	date	The date of the last/newest review
60	review_scores_rating		
61	review_scores_accuracy		
62	review_scores_cleanliness		
63	review_scores_checkin		
64	review_scores_communication		
65	review_scores_location		
66	review_scores_value		
67	license	text	The licence/permit/registration number
68	instant_bookable	boolean	[t=true; f=false]. Whether the guest can automatically book the listing without the host requiring to accept their booking request. An indicator of a commercial listing.
69	calculated_host_listings_count	integer	The number of listings the host has in the current scrape, in the city/region geography.
70	calculated_host_listings_count_entire_homes	integer	The number of Entire home/apt listings the host has in the current scrape, in the city/region geography
71	calculated_host_listings_count_private_rooms	integer	The number of Private room listings the host has in the current scrape, in the city/region geography
72	calculated_host_listings_count_shared_rooms	integer	The number of Shared room listings the host has in the current scrape, in the city/region geography
73	reviews_per_month	numeric	The number of reviews the listing has over the lifetime of the listing

The dependent variable is:

Sr. No.	**Variable**	**Datatype**	**Description**
1	price	currency	daily price in local currency

EXPLORATORY DATA ANALYSIS

DATA PRE-PROCESSING

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data pre-processing task.

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

Data shape and dimension:

The data consists of 74 variables and 38277 rows.

```
# dimensions of data
df.shape
```

```
(38277, 74)
```

Treating Anomalies in the data:

The target variable ‘price’ contains symbols like ‘$’ and ‘,’, and the 'host_acceptance_rate' variable contains symbols like ‘%’ and ‘,’. Both the variables were treated for the anomalies. Initially some of the variables in the dataset were assigned an improper datatype. The datatypes for all these variables were converted to suitable ones, to make them fit for the analysis.

Missing values treatment:

The next step of data pre-processing is to handle missing data in the datasets. If our dataset contains some missing data, then it may create a huge problem for our machine learning model. Hence it is necessary to handle missing values present in the dataset.

The data contains missing values in both the numerical and categorical variables. For the categorical variables, the mode imputation method is used to fill the missing values. For the numerical variables, the missing values have been treated with median and mean based on the presence of the outliers and the distribution of the data.

	total	Percentage
host_response_rate	17193	44.917313
host_acceptance_rate	16486	43.070251
host_is_superhost	34	0.088826
host_total_listings_count	34	0.088826
host_identity_verified	34	0.088826
neighbourhood_group_cleansed	0	0.000000
room_type	0	0.000000
accommodates	0	0.000000
bathrooms	38277	100.000000
bathrooms_text	107	0.279541
bedrooms	3975	10.384826
beds	2405	6.283147
price	0	0.000000
minimum_nights	0	0.000000
maximum_nights	0	0.000000
minimum_minimum_nights	18	0.047026
maximum_minimum_nights	18	0.047026
minimum_maximum_nights	18	0.047026
maximum_maximum_nights	18	0.047026
minimum_nights_avg_ntm	18	0.047026
maximum_nights_avg_ntm	18	0.047026

	total	Percentage
calendar_updated	38277	100.000000
availability_30	0	0.000000
availability_60	0	0.000000
availability_90	0	0.000000
availability_365	0	0.000000
number_of_reviews	0	0.000000
number_of_reviews_ltm	0	0.000000
review_scores_rating	9504	24.829532
review_scores_accuracy	10116	26.428403
review_scores_cleanliness	10105	26.399666
review_scores_checkin	10123	26.446691
review_scores_communication	10112	26.417953
review_scores_location	10126	26.454529
review_scores_value	10127	26.457141
license	38276	99.997387
instant_bookable	0	0.000000
calculated_host_listings_count	0	0.000000
calculated_host_listings_count_entire_homes	0	0.000000
calculated_host_listings_count_private_rooms	0	0.000000
calculated_host_listings_count_shared_rooms	0	0.000000
reviews_per_month	9504	24.829532

The variables with more than 75% missing values are dropped before further processing.

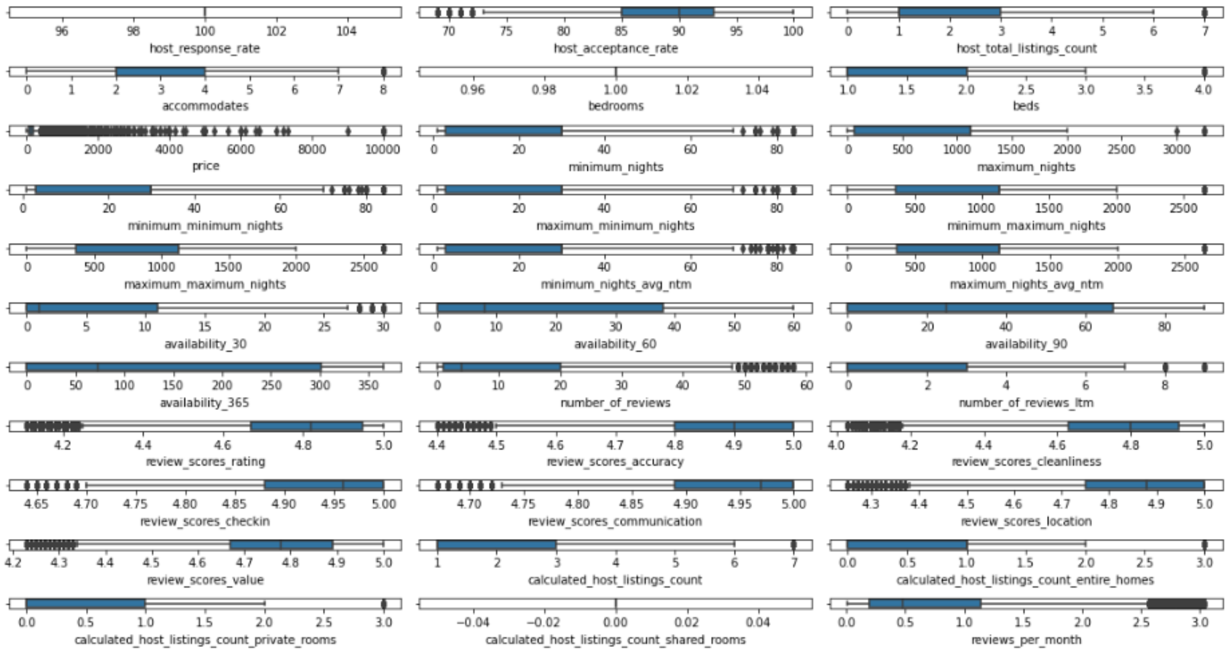
```
missing_perc[missing_perc['Percentage'] > 75.00]
```

	total	Percentage
bathrooms	38277	100.000000
calendar_updated	38277	100.000000
license	38276	99.997387

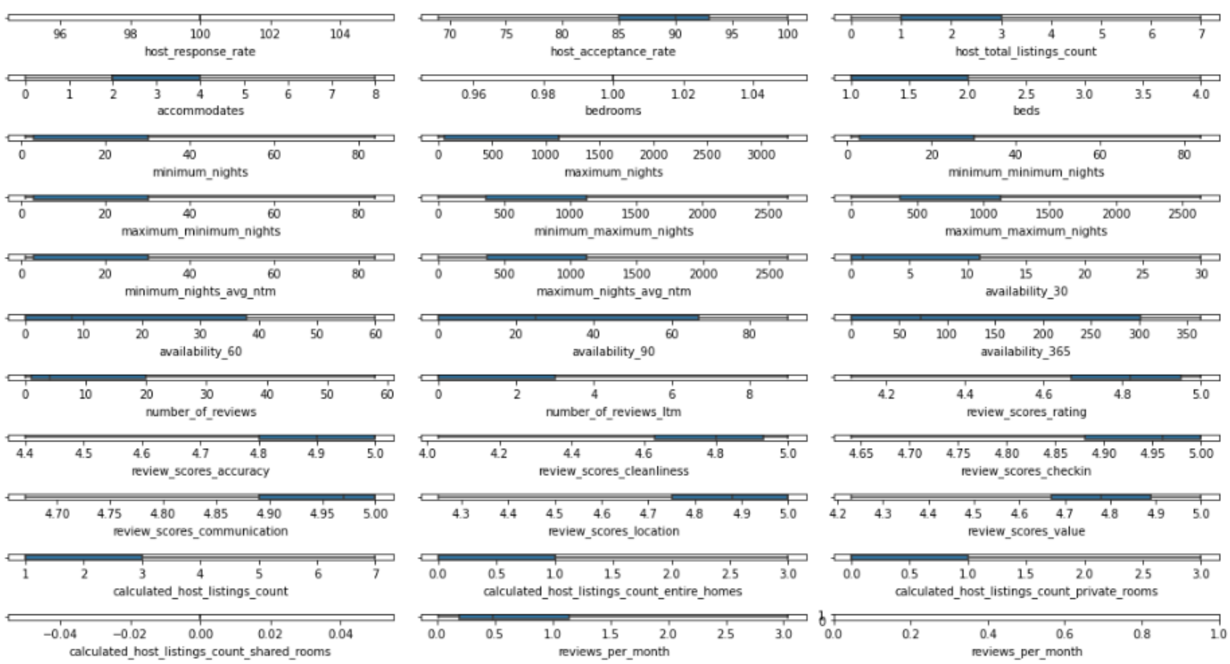
Treating outliers/missing values:

Checking and removal of outliers is important because presence of outliers can lead us to make incorrect conclusions by leading us to believe that the central tendencies are the correct representatives of the real-world scenario.

We have done caping of outliers using the Winsorization method for our dataset.



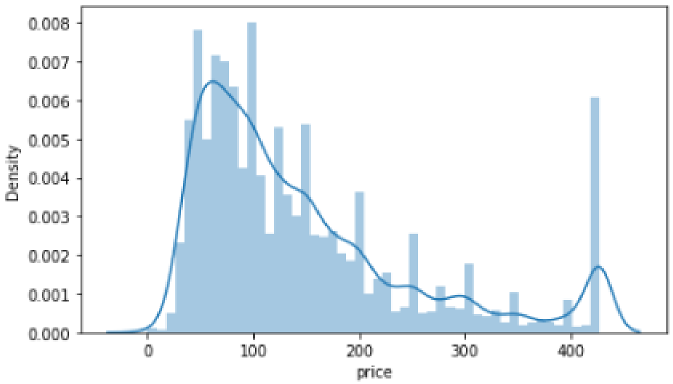
Boxplot of numerical variables before winsorization treatment.



Boxplot of numerical variables after winsorization treatment.

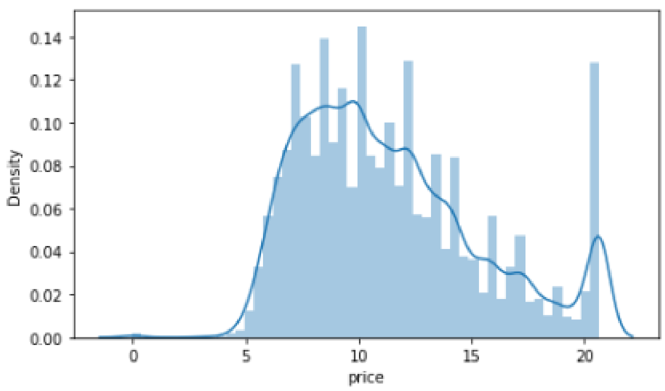
Exploratory Data Analysis:

Univariate Analysis: We plot the distribution curve to study the variation of the numerical data for our target variable 'price'.



Here, we can see the data is skewed towards right, and the skewness is 1.30

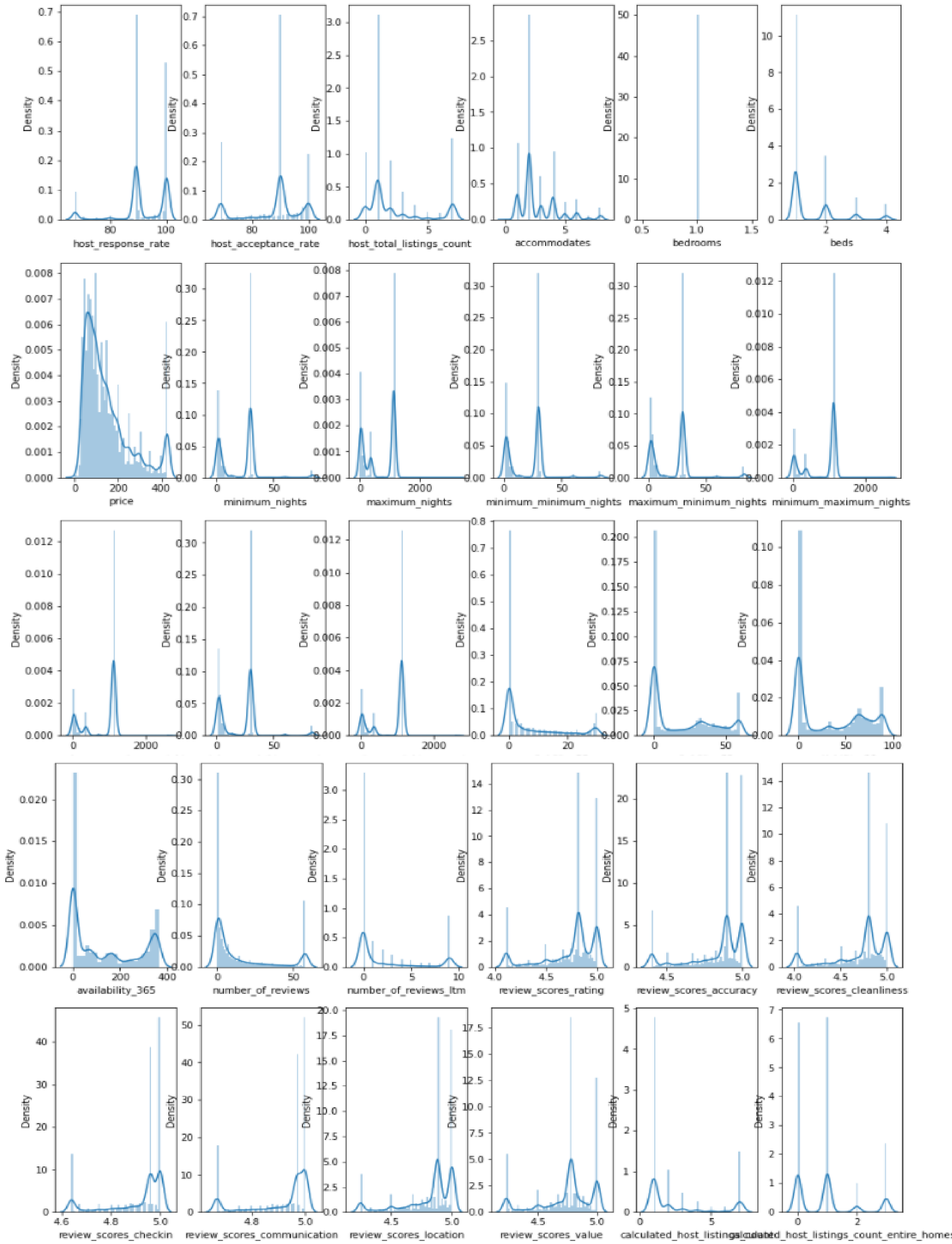
After doing square root transformation of the data, we get the following distribution.

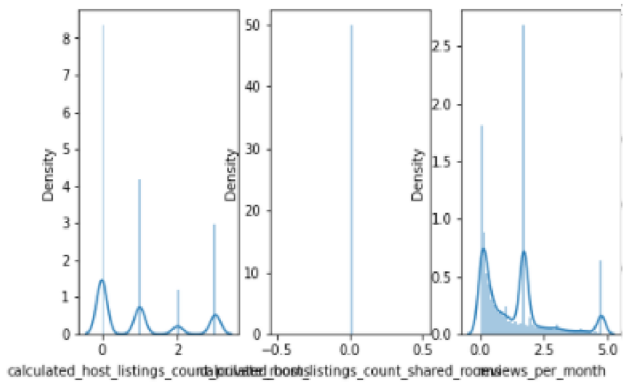


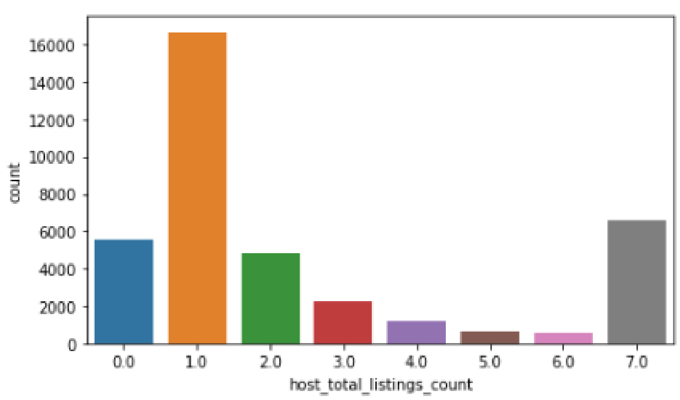
Here we see that, the skewness has been reduced. Now the skewness is reduced to 0.70

For Categorical Variables – We plot a combination of bar graph and pie chart to understand the distribution of categorical data in the dataset.

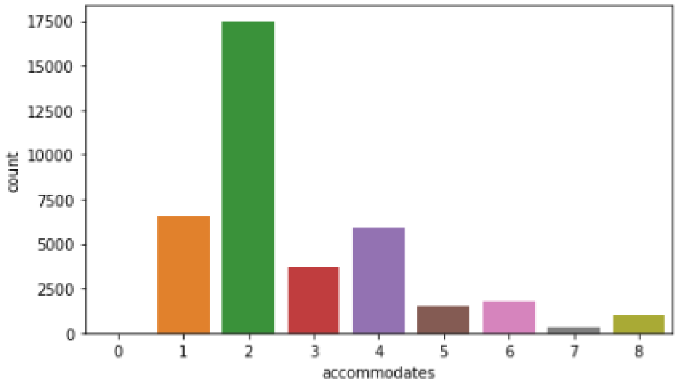
The distribution plot for the numeric variables is given on the next page.


host_response_rate
host_acceptance_rate
host_total_listings_count
accommodates
bedrooms
beds
price
minimum_nights
maximum_nights
minimum_minimum_nights
maximum_minimum_nights
minimum_maximum_nights
availability_365
number_of_reviews
number_of_reviews_ltm
review_scores_rating
review_scores_accuracy
review_scores_cleanliness
review_scores_checkin
review_scores_communication
review_scores_location
review_scores_value
calculated_host_listings_count
calculated_host_listings_count_entire_homes
Density

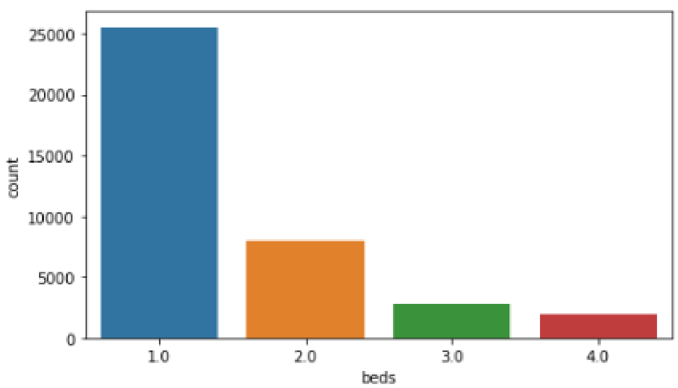




The count plot of 'host_total_listings_count' shows that the majority of the hosts have only one property listed with Airbnb



The properties listed with accommodation for two people certainly have a greater number as compared to the other numbers of accommodations.

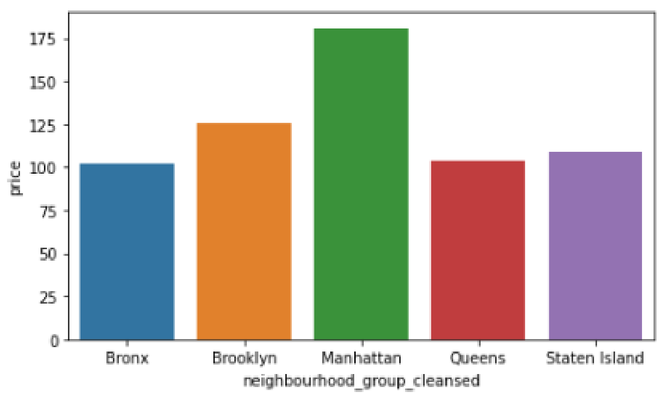


Bi-variate Analysis: We plot various charts to study the variation of price with respect to the independent features.

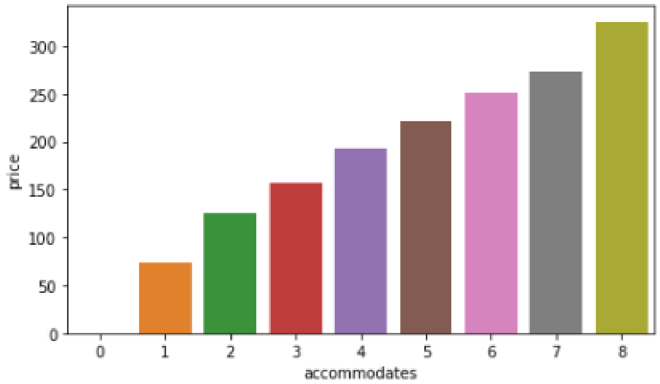
The area wise average pricing for the room_types is given below.

neighbourhood_group_cleansed	room_type	
Bronx	Entire home	147.004545
	Hotel room	0.000000
	Private room	73.452681
	Shared room	59.034483
Brooklyn	Entire home	175.088458
	Hotel room	120.555556
	Private room	74.264416
	Shared room	60.052910
Manhattan	Entire home	211.239301
	Hotel room	273.204188
	Private room	130.683826
	Shared room	109.709016
Queens	Entire home	158.629864
	Hotel room	171.111111
	Private room	68.208003
	Shared room	86.880734
Staten Island	Entire home	143.407609
	Private room	71.571429
	Shared room	40.000000

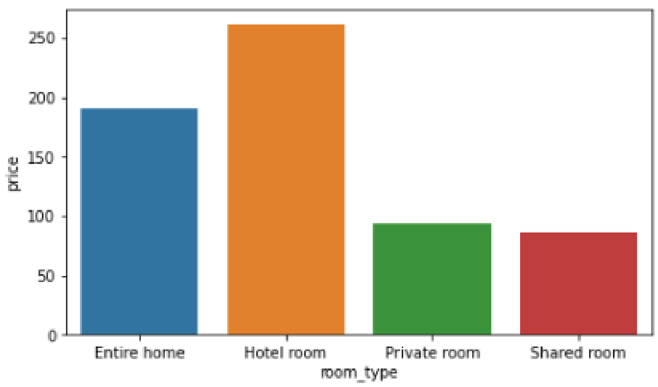
The maximum number of properties have one bed in their property.



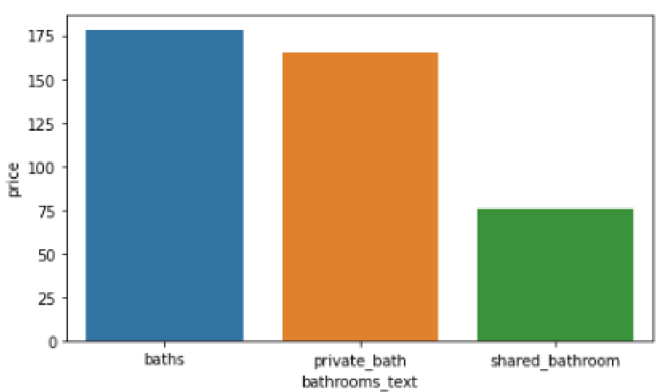
The properties located in Manhattan are costlier as compared to the properties located elsewhere in New York. Manhattan area is followed by Brooklyn, and the rest of the three areas Bronx, Queens, and Staten Island have same range of pricing.



The price also varies gradually with respect to the capacity of the listed property. As the capacity increases so, does the price, and vice-versa.

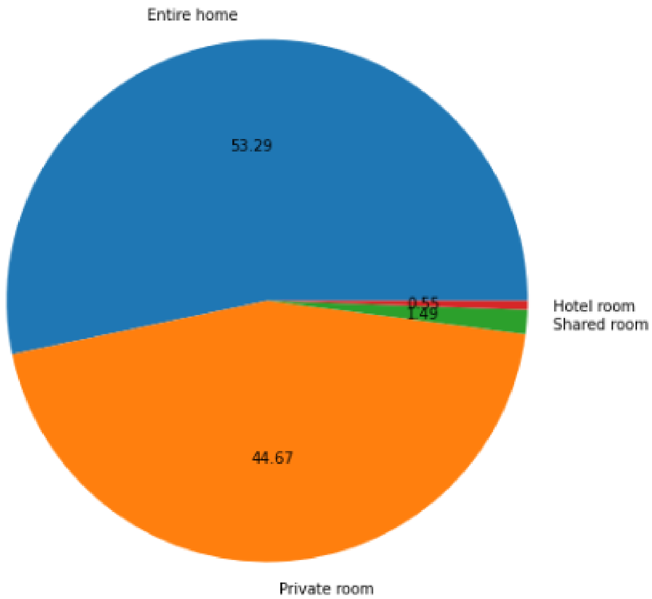


The 'room_type' as Hotel room has the highest mean price followed by Entire home, Private room, and Shared room.

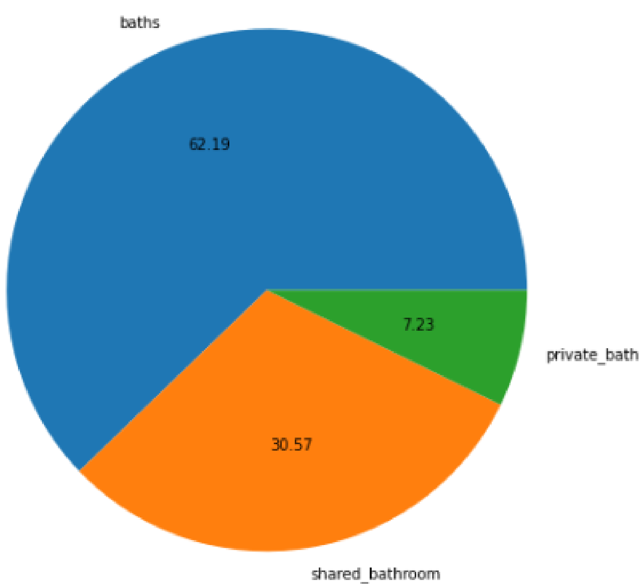


The properties having bathrooms have a high mean price as compared to the ones having shared bathrooms.

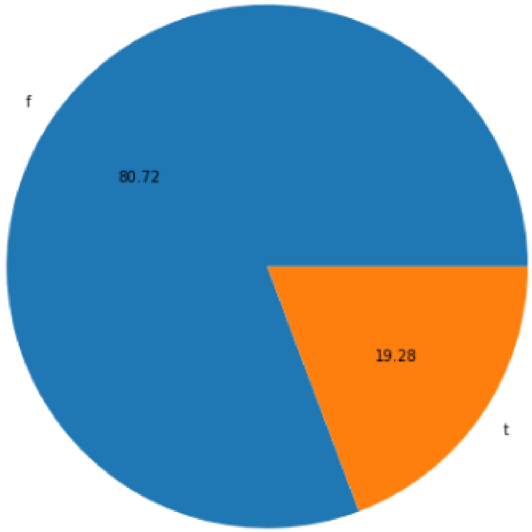
Univariate Analysis Interpretations:



The properties listed as ‘Entire home’ and ‘Private room’ have nearly equal and the highest share amongst the properties listed. The ‘Hotel room’ and ‘Shared room’ have the very negligible presence amongst the listed properties.

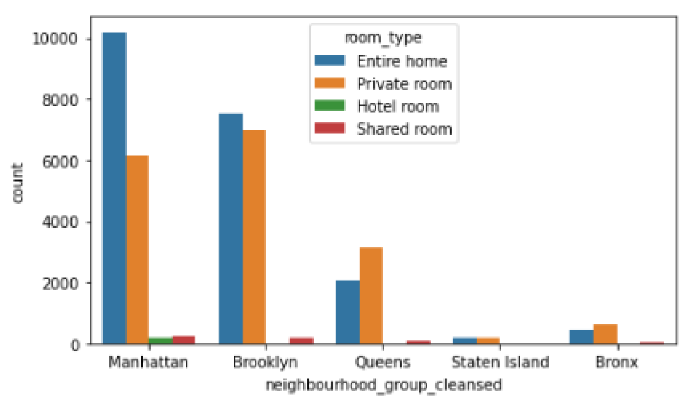


The properties having bathrooms are huge in percentage as compared to the ones having shared bathrooms.



The above pie-chart shows us the two types of hosts present on Airbnb. The one is ‘super host’ and the other is ‘non-super host’. From the pie-chart we can see that only 19.28% hosts on Airbnb are super hosts.

room_type	Entire home	Hotel room	Private room	Shared room
neighbourhood_group_cleansed				
Bronx	440	1	634	29
Brooklyn	7529	9	6989	189
Manhattan	10188	191	6158	244
Queens	2056	9	3149	109
Staten Island	184	0	168	1



The above table and plot give the numeric view of the properties listed area wise.

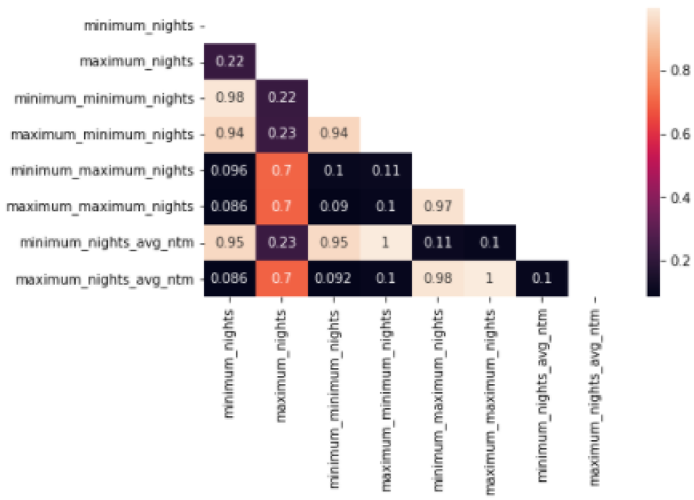
Multi-collinearity in the variables:

To check multi-collinearity in the variables, we use heatmap.

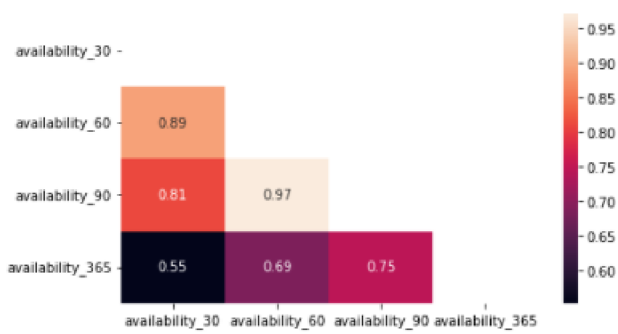
Heat-Map - Pearson Correlation Matrix

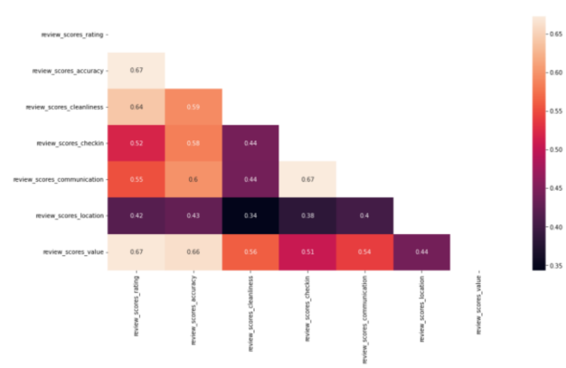
(Assumption : For the Pearson correlation, both variables should be normally distributed. Other assumptions include linearity and homoscedasticity)

It gives a measure of how much two numeric variables are linearly correlated. It tries to obtain a best fit line between two numeric variables and how close the points are to a fitted line.



The above heatmap shows that there is multi-collinearity between the variables concerned with minimum_nights and maximum_nights.



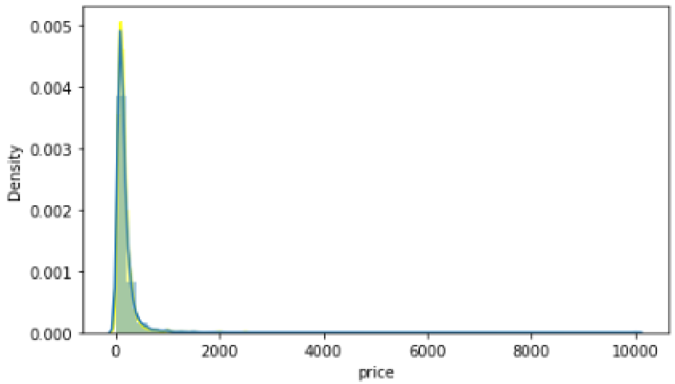


The variables depicting the availability and reviews also show multi-collinearity.

Statistical Tests:

2 Sample t-test:

The two-sample *t*-test (also known as the independent samples *t*-test) is a method used to test whether the unknown population means of two groups are equal or not.




```
H0 : mu(host_is_superhost_true) = Mu(host_is_superhost_false)
H1 : mu(host_is_superhost_true) != Mu(host_is_superhost_false)
```

```
print(stats.ttest_ind(df_dom , df_dum))
print('p_value greater than 0.5 so fail to reject null')
print('Means are not varying much')
```

```
Ttest_indResult(statistic=1.4083683776346778, pvalue=0.1590301476571377)
p_value greater than 0.5 so fail to reject null
Means are not varying much
```

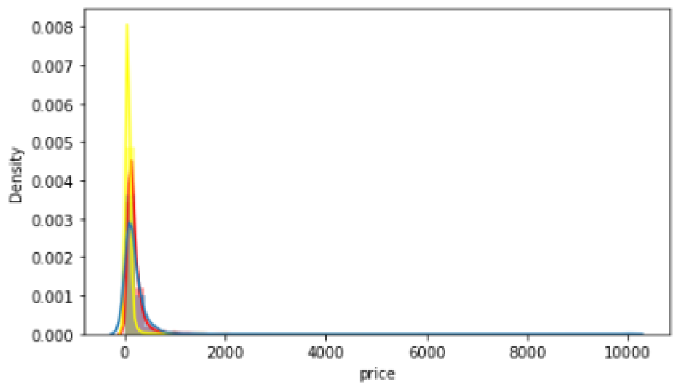
We got p-value > 0.05, hence ‘host_is_superhost’ variable and target variable are independent.

We perform the One Way ANNOVA test for the variables.

Below we have shown the ANNOVA test for "neighbourhood_group_cleansed" vs target variable.

```
H0 : All the sample means are equal
H1 : Atleast one sample mean is different
```

```
P_value 1.218925448838764e-289
p_val is less than significance level so reject null All the sample means are equal
Accept alternate Atleast one sample mean is different
```



The above graph shows distribution of 'price' for the five neighborhood groups.

The above output shows that the p-value is less than 0.05, thus we reject the null hypothesis and conclude that the average price of all neighbour_group_cleansed (i.e. is different locations) is not the same, some of the neighbour cleansed group categories may have significant effect in predicting the price.

```
(df[df['instant_bookable']=='f']['price']).mean()
```

```
139.853865211303
```

```
df[df['instant_bookable']=='t']['price'].mean()
```

```
162.45942835219643
```

```
stats.f_oneway(df[df['instant_bookable']=='f']['price'],df[df['instant_bookable']=='t']['price'])
```

```
F_onewayResult(statistic=359.1399131895222, pvalue=1.0057653201709963e-79)
```

Here we can see the p_value which is less than the 0.05 we can simply say instant_bookable columns average prices are different that we can check above as well, hence, with different categories price is changing.

Numerical vs numerical variables.

```
H0 : Varaibles are not correlated
H1 : Variables are correlated
```

	p_values	features
0	0.000001	host_response_rate
1	0.000000	host_acceptance_rate
2	0.000000	host_total_listings_count
3	0.000000	accommodates
4	NaN	bedrooms
5	0.000000	beds
6	0.000000	minimum_nights
7	0.007795	maximum_nights
8	0.000000	minimum_minimum_nights
9	0.000000	maximum_minimum_nights
10	0.007760	minimum_maximum_nights
11	0.130950	maximum_maximum_nights
12	0.000000	minimum_nights_avg_ntm
13	0.664886	maximum_nights_avg_ntm
14	0.000000	availability_30
15	0.000000	availability_60
16	0.000000	availability_90
17	0.000000	availability_365
18	0.000000	number_of_reviews
19	0.000000	number_of_reviews_ltm
20	0.000000	review_scores_rating
21	0.000032	review_scores_accuracy
22	0.000000	review_scores_cleanliness
23	0.006979	review_scores_checkin
24	0.000000	review_scores_communication
25	0.000000	review_scores_location
26	0.003938	review_scores_value
27	0.007183	calculated_host_listings_count
28	0.000000	calculated_host_listings_count_entire_homes
29	0.000000	calculated_host_listings_count_private_rooms
30	NaN	calculated_host_listings_count_shared_rooms
31	0.000000	reviews_per_month

P_values of Pearson test for all numeric variables

Interpretation: p_values of most of the features are less than significance level.

Hence, we reject null, the features are correlated with the target variable.

BASE MODEL

We have selected Linear Regression as our base model.

OLS Regression Results

Dep. Variable:	price	R-squared:	0.572
Model:	OLS	Adj. R-squared:	0.571
Method:	Least Squares	F-statistic:	849.9
Date:	Thu, 12 May 2022	Prob (F-statistic):	0.00
Time:	23:17:22	Log-Likelihood:	-26641.
No. Observations:	26793	AIC:	5.337e+04
Df Residuals:	26750	BIC:	5.372e+04
Df Model:	42		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
host_response_rate	0.0004	0.001	0.726	0.468	-0.001	0.002
host_acceptance_rate	0.0036	0.001	6.963	0.000	0.003	0.005
host_total_listings_count	0.0244	0.004	6.733	0.000	0.017	0.032
accommodates	0.2026	0.004	45.396	0.000	0.194	0.211
bedrooms	-1.6301	0.128	-12.749	0.000	-1.881	-1.379
beds	0.0433	0.008	5.300	0.000	0.027	0.059
minimum_nights	-0.0035	0.002	-2.012	0.044	-0.007	-8.96e-05
maximum_nights	0.0001	1.2e-05	11.887	0.000	0.000	0.000
minimum_minimum_nights	-0.0215	0.002	-11.978	0.000	-0.025	-0.018
maximum_minimum_nights	-0.0079	0.003	-2.807	0.005	-0.013	-0.002
minimum_maximum_nights	-0.0005	4.68e-05	-11.198	0.000	-0.001	-0.000
maximum_maximum_nights	0.0006	0.000	5.251	0.000	0.000	0.001
minimum_nights_avg_ntm	0.0208	0.003	6.547	0.000	0.015	0.027
maximum_nights_avg_ntm	-0.0002	0.000	-1.301	0.193	-0.000	8.61e-05
availability_30	0.0186	0.001	16.494	0.000	0.016	0.021
availability_60	-0.0047	0.001	-4.024	0.000	-0.007	-0.002
availability_90	0.0037	0.001	5.977	0.000	0.002	0.005
availability_365	-6.514e-05	4.57e-05	-1.426	0.154	-0.000	2.44e-05
number_of_reviews	-0.0018	0.000	-5.222	0.000	-0.002	-0.001
number_of_reviews_ltm	-0.0104	0.003	-3.956	0.000	-0.016	-0.005
review_scores_rating	0.2593	0.029	9.037	0.000	0.203	0.315
review_scores_accuracy	-0.1193	0.040	-2.996	0.003	-0.197	-0.041
review_scores_cleanliness	0.1933	0.022	8.838	0.000	0.150	0.236
review_scores_checkin	-0.1260	0.056	-2.239	0.025	-0.236	-0.016
review_scores_communication	0.0730	0.062	1.179	0.239	-0.048	0.194
review_scores_location	0.5354	0.025	21.027	0.000	0.485	0.585
review_scores_value	-0.3161	0.030	-10.521	0.000	-0.375	-0.257
calculated_host_listings_count	-0.0180	0.005	-3.752	0.000	-0.027	-0.009
calculated_host_listings_count_entire_homes	-0.1001	0.010	-9.797	0.000	-0.120	-0.080
calculated_host_listings_count_private_rooms	-0.0611	0.011	-5.607	0.000	-0.082	-0.040
calculated_host_listings_count_shared_rooms	7.74e-16	7.42e-17	10.430	0.000	6.29e-16	9.2e-16
reviews_per_month	-0.0431	0.008	-5.070	0.000	-0.060	-0.026
host_is_superhost_t	0.0196	0.012	1.631	0.103	-0.004	0.043
host_identity_verified_t	0.0185	0.011	1.685	0.092	-0.003	0.040
neighbourhood_group_cleansed_Brooklyn	0.3049	0.024	12.466	0.000	0.257	0.353
neighbourhood_group_cleansed_Manhattan	0.7054	0.025	28.577	0.000	0.657	0.754
neighbourhood_group_cleansed_Queens	0.0869	0.026	3.369	0.001	0.036	0.138
neighbourhood_group_cleansed_Staten Island	-0.1119	0.048	-2.318	0.020	-0.207	-0.017
room_type_Hotel room	-1.3307	0.058	-22.860	0.000	-1.445	-1.217
room_type_Private room	-0.6094	0.022	-28.025	0.000	-0.652	-0.567
room_type_Shared room	-0.8630	0.039	-22.138	0.000	-0.939	-0.787
bathrooms_text_private_bath	0.3904	0.021	18.696	0.000	0.349	0.431
bathrooms_text_shared_bathroom	-0.2073	0.016	-13.017	0.000	-0.238	-0.176
instant_bookable_t	0.1034	0.010	10.191	0.000	0.084	0.123
constant	-1.6301	0.128	-12.749	0.000	-1.881	-1.379

Omnibus:	6090.559	Durbin-Watson:	2.004
Prob(Omnibus):	0.000	Jarque-Bera (JB):	246759.391
Skew:	-0.296	Prob(JB):	0.00
Kurtosis:	17.856	Cond. No.	1.16e+16

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 6.4e-22. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

Interpretations:

The base model has the RSquared value of 0.572

Durbin Watson is 2.004 (equal to 2), hence, there is no auto correlation

Prob(Jarque_Bera) < 0.5, hence, errors are not following normal distribution

Cond. No. is greater than 1000, this implies that the variables have severe multicollinearity.

Model Evaluation:

```
base_model.rsquared
```

```
0.572
```

```
base_model.rsquared_adj
```

```
0.571
```

Our base model (linear regression) is 57.2% efficient in predicting the target variable.

We will try to improve the RSquare by trying different tree based regression models and doing hyper-parameter tuning.

Model Performance:

```
print('train rmse :' , get_train_rmse(base_model))
print('test rmse :' , get_test_rmse(base_model))
print('Model doesnot have overfitting')
```

```
train rmse : 0.654022
test rmse : 0.660636
Model doesnot have overfitting
```

```
mape_train = round(mape(ytrain['price'], train_pred),4)

# print the MAPE for the training set
print("Mean Absolute Percentage Error (MAPE) on training set: ", mape_train)
```

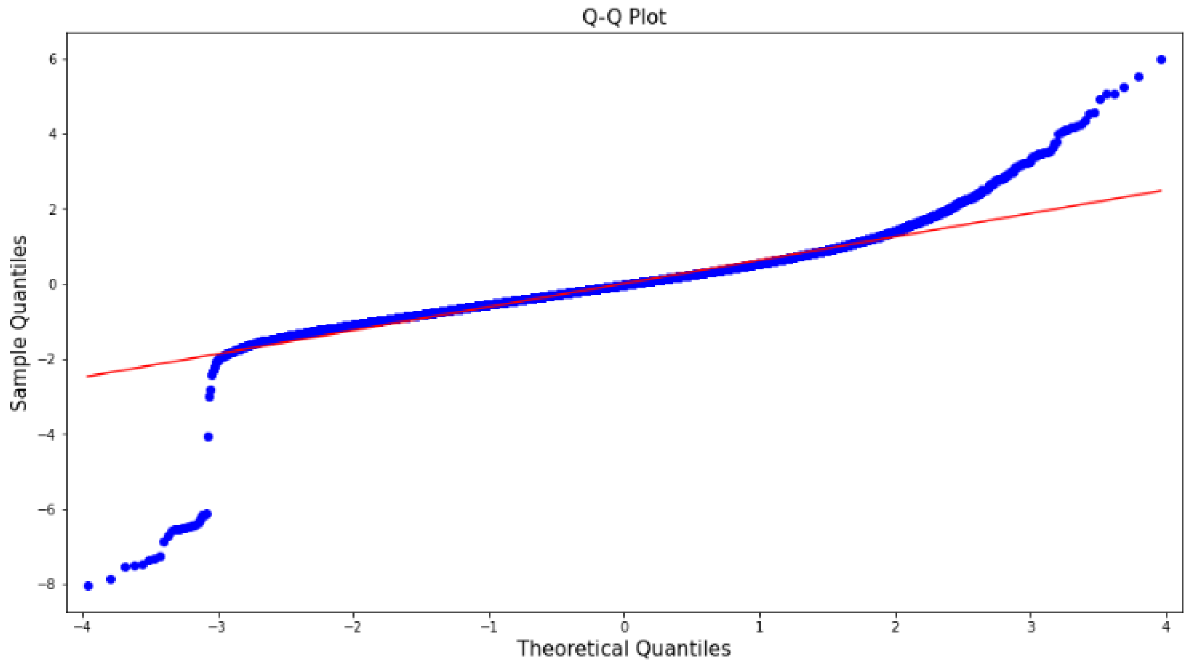
```
Mean Absolute Percentage Error (MAPE) on training set:  198.0052
```

```
mape_test = round(mape(ytest['price'], test_pred),4)

# print the MAPE for the training set
print("Mean Absolute Percentage Error (MAPE) on test set: ", mape_test)
```

```
Mean Absolute Percentage Error (MAPE) on test set:  197.7984
```

From the above values we can say that our model is not over-fitted.



Q-Q Plot

The Shapiro Wilk test output is:

```
stat, p_value = shapiro(base_model.resid)

print('Test statistic:', stat)
print('P-Value:', p_value)
```

```
Test statistic: 0.9087713956832886
P-Value: 0.0
```

From the above test we can see that the p-value is 0.0 (less than 0.05) and from the Q-Q Plot points are away from the normal line, thus we can say that the residuals are not normally distributed.

Lasso Regression Model:

The word "LASSO" stands for Least Absolute Shrinkage and Selection Operator. It is a statistical formula for the regularisation of data models and feature selection.

Lasso regression makes least significant features' slope as zero.

Model Evaluation:

```
print('RSquared without tuning:')
print('RSquared Train:' , train_score(lasso_model))
print('RSquared Test:' , test_score(lasso_model))
```

```
RSquared without tuning:
RSquared Train: 0.04656058609491076
RSquared Test: 0.0460088802914026
```

```
print('RSquared after tuning:')
print('RSquared Train:' , train_score(lasso_tune))
print('RSquared Test:' , test_score(lasso_tune))
```

```
RSquared after tuning:
RSquared Train: 0.571617374472559
RSquared Test: 0.5650707490975982
```

Our Lasso Regression model is 56.5% efficient in predicting the target variable.

Model Performance:

```
print('RSquared after tuning:')
print('RSquared Train:' , train_score(lasso_tune))
print('RSquared Test:' , test_score(lasso_tune))
```

```
RSquared after tuning:
RSquared Train: 0.571617374472559
RSquared Test: 0.5650707490975982
```

```
print('RMSE after tuning:')
print('train_rmse' , get_train_rmse(lasso_tune))
print('test_rmse' , get_test_rmse(lasso_tune))
```

```
RMSE after tuning:
train_rmse 0.654022
test_rmse 0.660635
```

From the above values we can say that our model is not over-fitted.

Ridge Regression Model:

Ridge regression is a model tuning method that is used to analyse any data that suffers from multicollinearity. This method performs L2 regularization. When the issue of multicollinearity occurs, least-squares are unbiased, and variances are large, this results in predicted values being far away from the actual values.

Model Evaluation:

```
print('RSquared without tuning:')
print('RSquared Train:' , train_score(ridge_model))
print('RSquared Test:' , test_score(ridge_model))
```

```
RSquared without tuning:
RSquared Train: 0.5716164907723695
RSquared Test: 0.5650995151396561
```

```
print('RSquared after tuning:')
print('RSquared Train:' , train_score(ridge_tune))
print('RSquared Test:' , test_score(ridge_tune))
```

```
RSquared after tuning:
RSquared Train: 0.5716164907723695
RSquared Test: 0.5650995151396561
```

Our Ridge Regression model is 56.5% efficient in predicting the target variable.

Model Performance:

```
print('RMSE without tuning:')
print('train_rmse :' , get_train_rmse(ridge_model))
print('test_rmse :' , get_test_rmse(ridge_model))
```

```
RMSE without tuning:
train_rmse : 0.654023
test_rmse : 0.660613
```

```
print('RMSE after tuning:')
print('train_rmse' , get_train_rmse(ridge_tune))
print('test_rmse' , get_test_rmse(ridge_tune))
```

```
RMSE after tuning:
train_rmse 0.654023
test_rmse 0.660613
```

From the above values we can say that our model is not over-fitted.

Decision Tree Regressor Model:

Decision tree regressor observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output. Continuous output means that the output/result is not discrete, i.e., it is not represented just by a discrete, known set of numbers or values.

Model Evaluation:

```
print('RSquared without tuning:')
print('RSquared Train:' , train_score(dt_model))
print('RSquared Test:', test_score(dt_model))
```

```
RSquared without tuning:
RSquared Train: 0.9769047307905014
RSquared Test: 0.3982984768497163
```

```
print('RSquared after tuning:')
print('RSquared Train:' , train_score(tuned_model))
print('RSquared Test:', test_score(tuned_model))
```

```
RSquared after tuning:
RSquared Train: 0.6821752873043108
RSquared Test: 0.5996124732652096
```

Our Decision tree regressor is 59.96% efficient in predicting the target variable.

Model Performance:

```
print('RMSE without tuning:')
print('train_rmse :' , get_train_rmse(dt_model))
print('test_rmse :' , get_test_rmse(dt_model))
```

```
RMSE without tuning:
train_rmse : 0.151858
test_rmse : 0.777039
```

```
print('RMSE after tuning:')
print('train_rmse :' , get_train_rmse(tuned_model))
print('test_rmse :' , get_test_rmse(tuned_model))
```

```
RMSE after tuning:
train_rmse : 0.56334
test_rmse : 0.633859
```

From the above values we can say that our model was over-fitted, but after hyper-parameter tuning we have solved the problem of over-fit.

Random Forest Regressor Model:

Every decision tree has high variance, but when we combine all of them together in parallel then the resultant variance is low as each decision tree gets perfectly trained on that particular sample data and hence the output doesn't depend on one decision tree but multiple decision trees. In the case of a classification problem, the final output is taken by using the majority voting classifier. In the case of a regression problem, the final output is the mean of all the outputs.

Model Evaluation:

```
print('RSquared without tuning:')
print('RSquared Train:', r2_score(ytrain , pred_train))
print('RSquared Test:' , r2_score(ytest, pred))
```

```
RSquared without tuning:
RSquared Train: 0.9356960382723759
RSquared Test: 0.6671694361909977
```

```
print('RSquared after tuning:')
print("RSquared Train:", train_score(rt_tuned_model))
print('RSquared Test:', test_score(rt_tuned_model))
```

```
RSquared after tuning:
RSquared Train: 0.8612665523771245
RSquared Test: 0.6744953063009462
```

Our Random Forest regressor is 67.45% efficient in predicting the target variable.

Model Performance:

```
print('RMSE without tuning:')
print('Train rmse :' , get_train_rmse(rt_model))
print('Test rmse :' , get_test_rmse(rt_model))
```

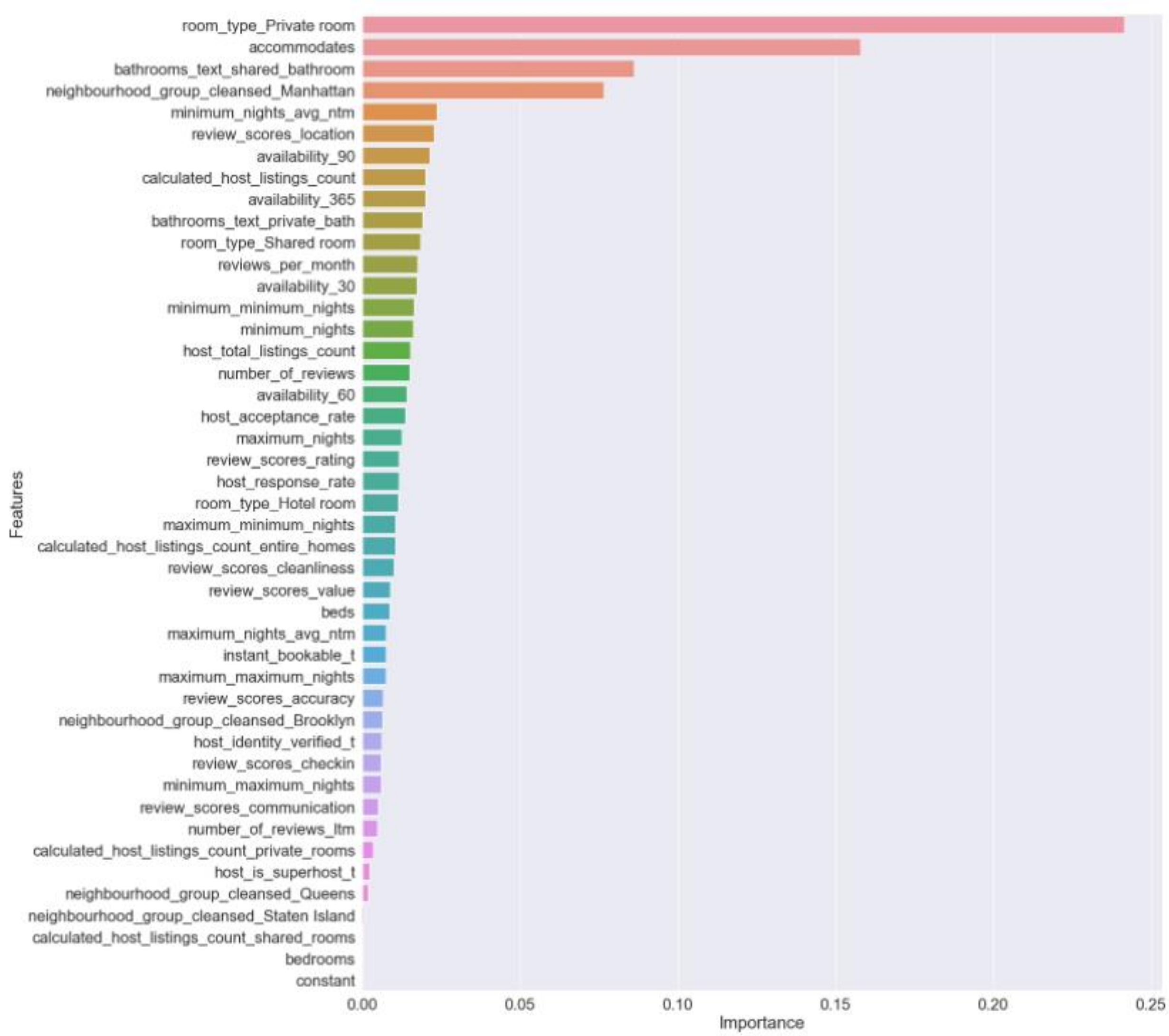
```
RMSE without tuning:
Train rmse : 0.253393
Test rmse : 0.577915
```

```
print('RMSE after tuning:')
print('Train rmse :', get_train_rmse(rt_tuned_model))
print('Test rmse :', get_test_rmse(rt_tuned_model))
```

```
RMSE after tuning:
Train rmse : 0.372192
Test rmse : 0.57152
```

From the above values we can say that our tuned model is less over-fitted than the non-tuned model.

Here, will have a look at the import features that impact the model prediction the most after tuned random forest model.



We can see that the four features, 'room_type_Private room', 'accommodates', 'bathrooms_text_shared_bathroom', and 'neighbourhood_group_cleansed_Manhattan' have a very significant impact on the prediction of the target variable.

ADABoost Regressor Model:

Model Evaluation:

```
print('RSquared without tuning:')
print('RSquared Train:' , train_score(ada_model))
print('RSquared Test:', test_score(ada_model))
```

```
RSquared without tuning:
RSquared Train: 0.9176260206655766
RSquared Test: 0.6437362336947372
```

```
print('RSquared after tuning:')
print('RSquared Train:' , train_score(ada_model_tuned))
print('RSquared Test:', test_score(ada_model_tuned))
```

```
RSquared after tuning:
RSquared Train: 0.6959069492951739
RSquared Test: 0.6267204269584645
```

Our ADABoost regressor is 62.67% efficient in predicting the target variable.

Model Performance:

```
print('RMSE without tuning:')
print('Train rmse :' , get_train_rmse(ada_model))
print('Test rmse :' , get_test_rmse(ada_model))
```

```
RMSE without tuning:
Train rmse : 0.286795
Test rmse : 0.597913
```

```
print('RMSE after tuning:')
print('Train rmse :' , get_train_rmse(ada_model_tuned))
print('Test rmse :' , get_test_rmse(ada_model_tuned))
```

```
RMSE after tuning:
Train rmse : 0.551036
Test rmse : 0.612026
```

From the above values we can say that our original model was over-fitted and after hyper-parameter tuning we are not able to solve the problem of over-fit to a large extent.

Gradient Boosting Regressor Model:

Model Evaluation:

```
print('RSquared without tuning:')
print('RSquared Train:' , train_score(gbm_model))
print('RSquared Test:', test_score(gbm_model))
```

```
RSquared without tuning:
RSquared Train: 0.6630087475618877
RSquared Test: 0.635331164786566
```

```
print('RSquared after tuning:')
print('RSquared Train:' , train_score(gbm_tuned))
print('RSquared Test:', test_score(gbm_tuned))
```

```
RSquared after tuning:
RSquared Train: 0.7380257311465931
RSquared Test: 0.6753345137174563
```

Our Gradient Boosting regressor is 67.53% efficient in predicting the target variable.

Model Performance:

```
print('RMSE without tuning:')
print('Train rmse :' , get_train_rmse(gbm_model))
print('Test rmse :' , get_test_rmse(gbm_model))
```

```
RMSE without tuning:
Train rmse : 0.580077
Test rmse : 0.604925
```

```
print('RMSE after tuning:')
print('Train rmse :' , get_train_rmse(gbm_tuned))
print('Test rmse :' , get_test_rmse(gbm_tuned))
```

```
RMSE after tuning:
Train rmse : 0.511453
Test rmse : 0.570782
```

From the above values we can say that our original model was over-fitted and after hyper-parameter tuning the overfit is slightly increased, but the overall efficiency of the model is increased.

XGBoost Regressor Model:

Model Evaluation:

```
print('RSquared without tuning:')
print('RSquared Train:' , train_score(xgb_model))
print('RSquared Test:', test_score(xgb_model))
```

```
RSquared without tuning:
RSquared Train: 0.8231306035838207
RSquared Test: 0.6793934696950417
```

```
print('RSquared after tuning:')
print('RSquared Train:' , train_score(xgb_tuned))
print('RSquared Test:', test_score(xgb_tuned))
```

```
RSquared after tuning:
RSquared Train: 0.7924463047908064
RSquared Test: 0.6824934916600406
```

Our XGBoost regressor is 68.25% efficient in predicting the target variable.

Model Performance:

```
print('RMSE without tuning:')
print('Train rmse :' , get_train_rmse(xgb_model))
print('Test rmse :' , get_test_rmse(xgb_model))
```

```
RMSE without tuning:
Train rmse : 0.420245
Test rmse : 0.567203
```

```
print('RMSE after tuning:')
print('Train rmse :' , get_train_rmse(xgb_tuned))
print('Test rmse :' , get_test_rmse(xgb_tuned))
```

```
RMSE after tuning:
Train rmse : 0.455241
Test rmse : 0.564454
```

From the above values we can say that our original model was over-fitted and after hyper-parameter tuning we are not able to solve the problem of over-fit to a large extent.

LightGBM Regressor Model:

Model Evaluation:

```
print('RSquared without tuning:')
print('RSquared Train:' , train_score(lgb_model))
print('RSquared Test:', test_score(lgb_model))
```

```
RSquared without tuning:
RSquared Train: 0.744283889442167
RSquared Test: 0.6800638872119851
```

```
print('RSquared after tuning:')
print('RSquared Train:' , train_score(lgb_model_tuned))
print('RSquared Test:', test_score(lgb_model_tuned))
```

```
RSquared after tuning:
RSquared Train: 0.6982843870170306
RSquared Test: 0.6634572242021717
```

Our LightGBM regressor is 66.34% efficient in predicting the target variable.

Model Performance:

```
print('RMSE without tuning:')
print('Train rmse :' , get_train_rmse(lgb_model))
print('Test rmse :' , get_test_rmse(lgb_model))
```

```
RMSE without tuning:
Train rmse : 0.505307
Test rmse : 0.56661
```

```
print('RMSE after tuning:')
print('Train rmse :' , get_train_rmse(lgb_model_tuned))
print('Test rmse :' , get_test_rmse(lgb_model_tuned))
```

```
RMSE after tuning:
Train rmse : 0.548878
Test rmse : 0.581129
```

From the above values we can say that our original model was over-fitted and after hyper-parameter tuning we are slightly able to solve the problem of over-fit.

Interpretations/Conclusions:

	Model Name	R-Squared Train	R-Squared Test	RMSE Train	RMSE Test
0	Linear Regression	0.571600	0.565100	0.654000	0.660600
1	Lasso Regression	0.046600	0.046000	0.571600	0.565100
2	Lasso Regression Tuned	0.571600	0.565100	0.654000	0.660600
3	Ridge Regression	0.571600	0.565100	0.654000	0.660600
4	Ridge Regression Tuned	0.571600	0.565100	0.654000	0.660600
5	Decision Tree Regressor	0.976900	0.398200	0.151800	0.777000
6	Decision Tree Regressor Tuned	0.682200	0.599600	0.563300	0.633800
7	Random Forest Regressor	0.935700	0.667200	0.253400	0.577900
8	Random Forest Regressor Tuned	0.861300	0.674500	0.372200	0.571500
9	ADABoost Regressor	0.917600	0.643700	0.286800	0.597900
10	ADABoost Regressor Tuned	0.695900	0.626700	0.551000	0.612000
11	Gradient Boosting Regressor	0.663000	0.635300	0.580100	0.604900
12	Gradient Boosting Regressor Tuned	0.738000	0.675300	0.511400	0.570800
13	XGBoost Regressor	0.823100	0.679400	0.420200	0.567200
14	XGBoost Regressor Tuned	0.792400	0.682500	0.455200	0.564400
15	LightGBM Regressor	0.744300	0.680100	0.505300	0.566600
16	LightGBM Regressor Tuned	0.698300	0.663400	0.548900	0.581100

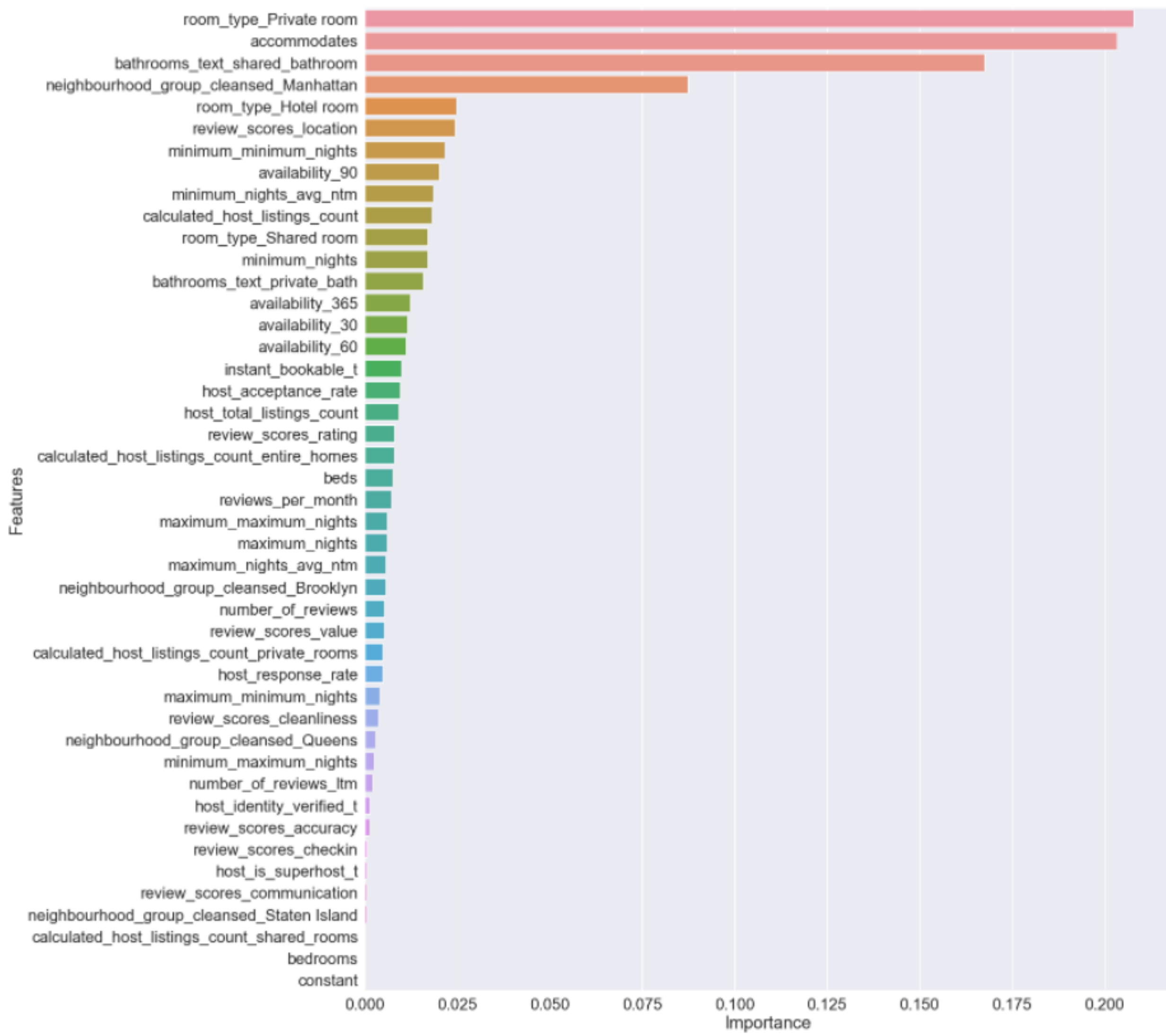
The detailed process undergone in achieving the results are seen in the jupyter notebook attached with the report. From initial observation it can be seen that, linear regression model is giving the RSquared value of 0.579 though RMSE value for test and train are not having much any difference. In the we can see there is strong multi collinearity between the column by looking at the condition no.

There are so many columns which has negative coefficients with unit decrease in these there will be unit increase in price i.e., they are inversely proportional with change proportional to modulus of coefficient.

We have tried all the various regressors available like bagging and boosting regressors, and we have found that Gradient Boosting regressor is giving us the most significant model. Therefore, we have concluded Gradient Boosting regressor as our final model for the project.

Model Selection:

Our main model is Gradient Boosting regressor (tuned) model since we got 73% accuracy in train and 67% on test. Although we are getting more train accuracy in different models but test score got decreased devastatingly, so it leads to overfitting issues, since the difference is low and accuracy is high, we are selecting Gradient Boosting as our final model



Based upon the feature importance:

1. We can see that the four features, 'room_type', ‘accommodates', 'bathrooms_text', and ‘neighbourhood_group_cleansed' have a very significant impact on the prediction of the target variable.

2. We can say that ‘room_type' and ‘accomodates', ‘neighbourhood_group_cleansed' are contributing more in price prediction.

3. In ‘room_type’ we have 4 types, viz., ‘hotel_room’, ‘shared_room’, ‘private_room’ and ‘entire_home’. ‘private_room’ is contributing more in predicting the price.

4. ‘accomodates’: The number of people amongst the property can be shared.

    - ‘accomodates’ are more significant since, increase in accomodates increases the ‘price’.

    - ‘accomodates’ has a positive correlation with the target variable ‘price’.

5. ‘neighbourhood_group_cleansed’ (location):

    - Location wise, Manhattan is contributing the most to the model, Staten Island has the least contribution, almost negligible.

    - Price varies according to location. Manhattan has the costliest properties followed by Brooklyn. Staten Island has the cheapest properties.

    - Basic amenities are also more available in the properties located in Manhattan followed by Brooklyn.

6. 'bathrooms_text’ (type of bathrooms).

    - Shared_bathrooms are contributing more and people are opting for listings with shared bathrooms.

    - Listings with private bathrooms are too costly and are less in number.

7. 'availability’ (number of days the property shows as available).

    - Listings are available with different number of days

    - All the 'availability’ features are highly correlated since they are all equally contributing.

8. ’review’

    - 'review_scores_rating' and 'review_scores_accuracy' are more significant than compared to 'review_scores_communication'.

9. Number of 'beds' have less significance compared to the number of ‘bathrooms’.

10. ‘host_identity’, ‘bedrooms’, and 'review_scores_communication' are less significant features.

Buissness interpretation:

1. We can see that the four features, 'room_type', 'accommodates', 'bathrooms_text', and 'neighbourhood_group_cleansed' have a very significant impact on the prediction of the target variable 'price'.
2. In 'room_type' if it is 'hotel room', then the hosts can list their property at a higher price.
3. Location wise Manhattan has the costliest properties followed by Brooklyn.
4. As the number of accommodates increases, so thus increases the price.
5. Private bathrooms are costlier, and we have limited number of listings having them. This can be the reason of them being costly.
6. The 'super host' status of the host, or the number of properties listed by him don't have much impact in the price prediction.
7. The number of reviews any property has, does not have any impact on the pricing.
8. Most number of listings have capacity for 2 accommodates.

Limitations:

1. The base models could have been better tuned considering the computational capacity of the systems on which the models were built.
2. The anomalies and outliers in the data too many. This consumes very significant time of the project.

Future Work:

1. We successfully built the most significant model possible for the dataset. We will be further going forward with deployment of our model.
2. We will explore more cloud-based options to do better hyper-parameter tuning and increasing the significance of the model.