PREDICTION OF THE COST PER NIGHT FOR THE HOTELS IN ONLINE TRAVEL AGENT PLATFORM

Post Graduate Program in Data Science Engineering

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

Sept'21

Submitted by

Bhuvanesh

Kumaran

Mohith

Nagesh

Rajat

Varun

Mentored by

Mr. Romil Gupta

ACKNOWLEDGEMENT

Any endeavour in a specific field requires the guidance and support of many people for successful completion. The sense of achievement on completing anything remains incomplete if the people who were instrumental in its execution are not properly acknowledged. We would like to take this opportunity to verbalize our deepest sense of indebtedness to our project mentor, Mr. Romil Gupta, who was a constant pillar of support and continually provided us with valuable insights to improve upon our project and make it a success. Further, we would like to thank our parents for encouraging us and providing us a platform wherein we got an opportunity to design our own project.

DECLARATION

We hereby declare, that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

INTRODUCTION:

A hotel is a managed building or establishment, which provides guests with a place to stay overnight – on a short-term basis – in exchange for money. Due to the modern internet revolution, the reservation of a room in any hotel can be made via online by using the Online Travel Agents (OTA). These agents create search engine applications that acts as middle man between the hotel merchants and customers to process direct online reservations. An OTA acts as commission-free direct bookings agent for the customers; also provide the way to optimize the merchant's sales strategy and maximize profit. These agents take an average commission fee from merchants between 10% to 25% after each successful stay of the customers. From the article „"US Hotel gross booking by OTA with the comparison hotel applications /websites" released by „The wall Street Journal" , the country "United States of America" have started to make higher turnover from the F.Y(Financial Year)-2016. By the F.Y-2017, the gross booking by OTA agents made $34.8 billion (B) and whereas the hotel websites/apps made only $32.4B. The key advantages of having the hotel business in OTA platforms are Increased Exposure (Visibility), Drive Traffic, More Bookings, Right Guests, Better Rankings and Better Reviews.

PROBLEM STATEMENT:

1.Business Problem Understanding:

AirBnB is one such OTA platform acts as the middle man between hotels and customers to book a room; also provides lodging, homestays for vacations and tourism activities. The platform provides businesses an increased opportunity to reach out to a wider customer base and at the same time it empowers the customer by providing them with beforehand details of the type of the property, amenities, location of stay and plan out their budget for the trip accordingly. The major part of the OTA platform is to provide an accurate prediction of Cost Per Night to the customers.

2.Business Objective:

Cost Per Night: The total amount that end user has to spend for staying at a property for a single night is known as cost per night. There are various factors that determine the cost per night from property to property also depends upon features like quality of the property, its size, neighborhood, facilities/amenities and location

Travelers often budget their travel costs in order to ensure that they spend only the optimal amount and knowing cost per night beforehand would help them plan their travel efficiently. This helps the OTA platforms by providing a comparison benchmark between properties and also help in identifying the various features of a property that will determine the cost per night. This information will also help the host/owners of a property in identifying the short coming of their property and help improve their properties by matching facilities/amenities provided by their competitors.

VARIABLE CATEGORIZATION WITH DESCRIPTION:

The dataset consists of 74 variables. Out of these variables 73 are independent variables and 1 is a target variable. The variables are a mixture of both numerical and categorical type.

1. Numerical:

Sl. No	Attributes	Dtype	Desicription
1	id	int64	Airbnb's unique identifier for the listing
2	scrape_id	int64	Inside Airbnb "Scrape" this was part of
3	host_id	int64	Airbnb's unique identifier for the host/user
4	host_listings_count	float64	The number of listings the host has.
5	host_total_listings_count	float64	The number of listings the host has.
6	latitude	float64	Uses the World Geodetic System (WGS84) projection for latitude and longitude.
7	longitude	float64	Uses the World Geodetic System (WGS84) projection for latitude and longitude.
8	accommodates	int64	The maximum capacity of the listing
9	bathrooms	float64	The number of bathrooms in the listing
10	bedrooms	float64	The number of bedrooms
11	beds	float64	The number of bed(s)
12	minimum_nights	int64	minimum number of night stay for the listing.
13	maximum_nights	int64	maximum number of night stay for the listing.
14	minimum_minimum_nights	float64	the smallest minimum_night value from the calender (looking 365 nights in the future)
15	maximum_minimum_nights	float64	the largest minimum_night value from the calender (looking 365 nights in the future)
16	minimum_maximum_nights	float64	the smallest maximum_night value from the calender (looking 365 nights in the future)
17	maximum_maximum_nights	float64	the largest maximum_night value from the calender (looking 365 nights in the future)
18	minimum_nights_avg_ntm	float64	the average minimum_night value from the calender (looking 365 nights in the future)
19	maximum_nights_avg_ntm	float64	the average maximum_night value from the calender (looking 365 nights in the future)
20	calendar_updated	float64	Calendar
21	availability_30	int64	avaliability_30. The availability of the listing 30 days in the future as determined by the calendar.
22	availability_60	int64	avaliability_60. The availability of the listing 60 days in the future as determined by the calendar.

23	availability_90	int64	avaliability_90. The availability of the listing 90 days in the future as determined by the calendar.
24	availability_365	int64	avaliability_365. The availability of the listing 365 days in the future as determined by the calendar.
25	number_of_reviews	int64	The number of reviews the listing has
26	number_of_reviews_ltm	int64	The number of reviews the listing has (in the last 12 months)
27	number_of_reviews_l30d	int64	The number of reviews the listing has (in the last 30 days)
28	review_scores_rating	float64	The overall review for the stay
29	review_scores_accuracy	float64	The accuracy review score for the stay
30	review_scores_cleanliness	float64	The cleanliness review score for the stay
31	review_scores_checkin	float64	The checkin review score for the stay
32	review_scores_communication	float64	The communication review score for the stay
33	review_scores_location	float64	The location review score for the stay
34	review_scores_value	float64	The value review score for the stay
35	calculated_host_listings_count	int64	The number of listings the host has in the current scrape, in the city/region geography.
36	calculated_host_listings_count_entire_homes	int64	The number of Entire home/apt listings the host has in the current scrape, in the city/region geography
37	calculated_host_listings_count_private_rooms	int64	The number of Private room listings the host has in the current scrape, in the city/region geography
38	calculated_host_listings_count_shared_rooms	int64	The number of Shared room listings the host has in the current scrape, in the city/region geography
39	reviews_per_month	float64	The number of reviews the listing has over the lifetime of the listing

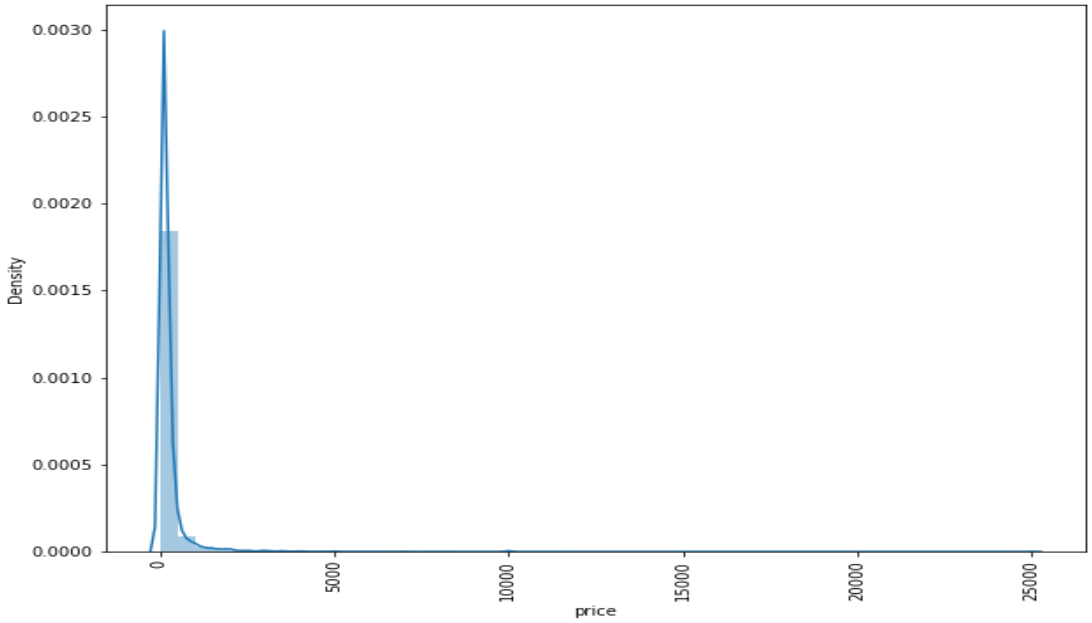
2. <u>Categorical:</u>

Sl. No	Attributes	Dtype	Desicription
1	listing_url	object	Airbnb's unique url for the listing
2	last_scraped	object	UTC. The date and time this listing was "scraped".
3	name	object	Name of the listing
4	description	object	Detailed description of the listing
5	neighborhood_overview	object	Host's description of the neighbourhood
6	picture_url	object	URL to the Airbnb hosted regular sized image for the listing
7	host_url	object	The Airbnb page for the host
8	host_name	object	Name of the host. Usually just the first name(s).
9	host_since	object	The date the host/user was created. For hosts that are Airbnb guests this could be the date they registered as a guest.
10	host_location	object	The host's self-reported location
11	host_about	object	Description about the host

12	host_response_time	object	The Time the host take to respond to a booking
13	host_response_rate	object	The rate at which a host responds booking requests.
14	host_acceptance_rate	object	The rate at which a host accepts booking requests.
15	host_is_superhost	object	[t=true; f=false]
16	host_thumbnail_url	object	Url for host thumbnail
17	host_picture_url	object	Url for host pictures
18	host_neighbourhood	object	The neighbourhood that the host is located
19	host_verifications	object	Host verification
20	host_has_profile_pic	object	Host profile picture
21	host_identity_verified	object	Host identity verified
22	neighbourhood	object	The neighbourhood as geocoded using the latitude and longitude
23	neighbourhood_cleansed	object	The neighbourhood as geocoded using the latitude and longitude against neighborhoods as defined by open or public digital shapefiles.
24	neighbourhood_group_cleansed	object	The neighborhood group as geocoded using the latitude and longitude against neighborhoods as defined by open or public digital shapefiles.
25	property_type	object	Self-selected property type. Hotels and Bed and Breakfasts are described as such by their hosts in this field
26	room_type	object	All homes are grouped into the following three-room types: Entire place Private room Shared room Entire place
27	bathrooms_text	object	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 are used.
28	amenities	object	The various consumable and facilities that come with the room
29	has_availability	object	[t=true; f=false]
30	calendar_last_scraped	object	Calendar_last_scarped
31	first_review	object	The date of the first/oldest review
32	last_review	object	The date of the last/newest review
33	license	object	The licence/permit/registration number
34	instant_bookable	object	[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.

3. Target Variable

The target variable of the above dataset is price. We have to predict the daily price for the stay.



We observe that the price values are highly right or positively skewed

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.

1. 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.

2. 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.

3. Data preparation - 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.

4. 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.

5. 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.

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.
- The data consists of 33000 rows and 74 columns. Out of these we have 35 categorical columns and the rest as numerical.

1. Variable Datatype:

- We first check the data types of each of the columns of the data.

Sl. No	Attributes	Dtype
1	id	int64
2	listing_url	object
3	scrape_id	int64
4	last_scraped	object
5	name	object
6	description	object
7	neighborhood_overview	object
8	picture_url	object
9	host_id	int64
10	host_url	object
11	host_name	object
12	host_since	object
13	host_location	object
14	host_about	object
15	host_response_time	object
16	host_response_rate	object
17	host_acceptance_rate	object
18	host_is_superhost	object
19	host_thumbnail_url	object
20	host_picture_url	object
21	host_neighbourhood	object
22	host_listings_count	float64
23	host_total_listings_count	float64
24	host_verifications	object
25	host_has_profile_pic	object
26	host_identity_verified	object
27	neighbourhood	object
28	neighbourhood_cleansed	object
29	neighbourhood_group_cleansed	object
30	latitude	float64
31	longitude	float64
32	property_type	object

33	room_type	object
34	accommodates	int64
35	bathrooms	float64
36	bathrooms_text	object
37	bedrooms	float64
38	beds	float64
39	amenities	object
40	price	object
41	minimum_nights	int64
42	maximum_nights	int64
43	minimum_minimum_nights	float64
44	maximum_minimum_nights	float64
45	minimum_maximum_nights	float64
46	maximum_maximum_nights	float64
47	minimum_nights_avg_ntm	float64
48	maximum_nights_avg_ntm	float64
49	calendar_updated	float64
50	has_availability	object
51	availability_30	int64
52	availability_60	int64
53	availability_90	int64
54	availability_365	int64
55	calendar_last_scraped	object
56	number_of_reviews	int64
57	number_of_reviews_ltm	int64
58	number_of_reviews_l30d	int64
59	first_review	object
60	last_review	object
61	review_scores_rating	float64
62	review_scores_accuracy	float64
63	review_scores_cleanliness	float64
64	review_scores_checkin	float64
65	review_scores_communication	float64
66	review_scores_location	float64
67	review_scores_value	float64
68	license	object
69	instant_bookable	object
70	calculated_host_listings_count	int64
71	calculated_host_listings_count_entire_homes	int64
72	calculated_host_listings_count_private_rooms	int64
73	calculated_host_listings_count_shared_rooms	int64
74	reviews_per_month	float64

- From here we observe that host_response_rate, host_acceptance_rate and the target variable 'price' are numerical data but are stated as object. We need to convert them to numerical data by removing special characters like %, $ and comma.

2. Removal of unwanted attribute:

- After checking the dataset in detail, we can see the several of the categorical have 100% unique variable or are URLs which will not provide any value to the predictions, hence we can drop these variables.
- Similarly numerical variable like id, host_id and scrape_id will not provide any value to the predictions; hence we can drop these variables also.

- Variables with 100% missing values such as calendar updated and bathrooms can be dropped.

3. Missing Value 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.

Sl. No	Attributes	Null Value Percentage	Dtype
1	host_response_time	32.59	object
2	host_response_rate	32.59	float64
3	host_acceptance_rate	30.97	float64
4	host_is_superhost	0.02	object
5	host_total_listings_count	0.02	float64
6	bathrooms_text	0.19	object
7	bedrooms	11.24	float64
8	beds	5.43	float64
9	minimum_minimum_nights	0.01	float64
10	maximum_minimum_nights	0.01	float64
11	minimum_maximum_nights	0.01	float64
12	maximum_maximum_nights	0.01	float64
13	minimum_nights_avg_ntm	0.01	float64
14	maximum_nights_avg_ntm	0.01	float64
15	review_scores_rating	24.03	float64
16	review_scores_accuracy	25.02	float64
17	review_scores_cleanliness	25.01	float64
18	review_scores_checkin	25.04	float64
19	review_scores_communication	25.02	float64
20	review_scores_location	25.05	float64
21	review_scores_value	25.06	float64
22	reviews_per_month	24.03	float64

- We can observe that 22 attributes have missing values in them.
- The categorical variable were imputed with proportional imputation.
- All numerical variables with missing values are highly skewed hence the best approach was to impute them with median over mean. Hence the missing value in the numerical variable were imputed with median for null value percent less than 10 and the rest of the variable was imputed with logival imputation technique.

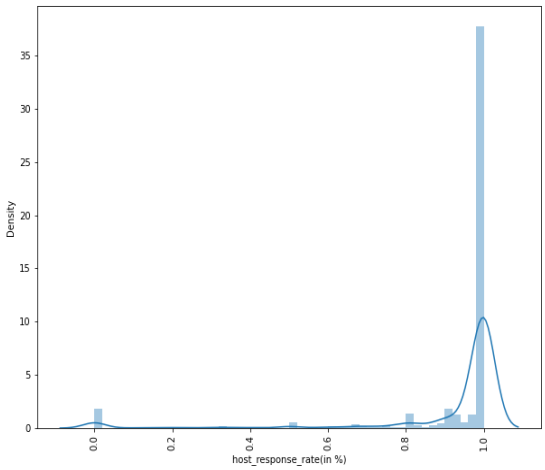
CHECK FOR OUTLIERS:

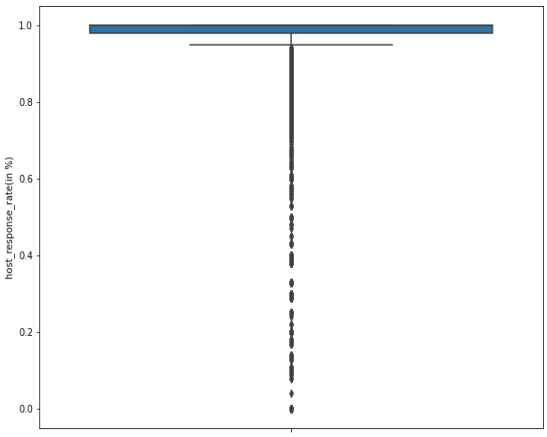
- Data has outliers present in each of the numerical columns. For making the base model, we do not perform any outlier treatment and retain all the rows present in the data.

UNIVARIATE ANALYSIS:

1.1. Numerical:

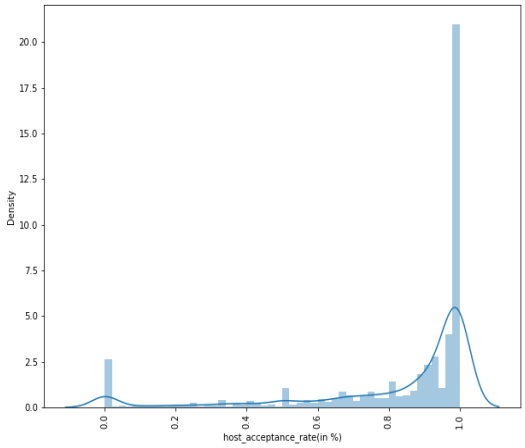
a. Host_response_rate (in %)

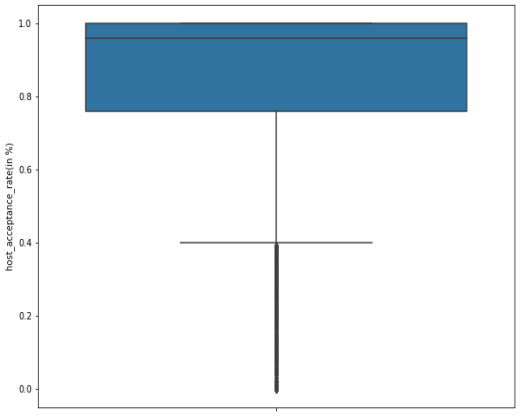




- Host response time is left skewed.
- It is leptokurtic, and has wide tail
- IQR lies at 1.0
- Outliers are present.

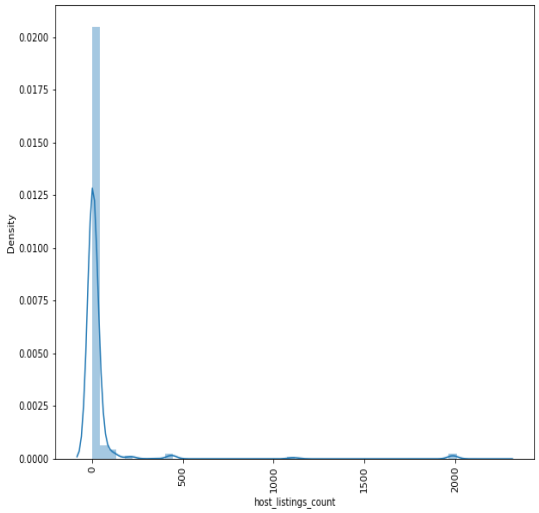
b. Host_acceptance_rate(in%)

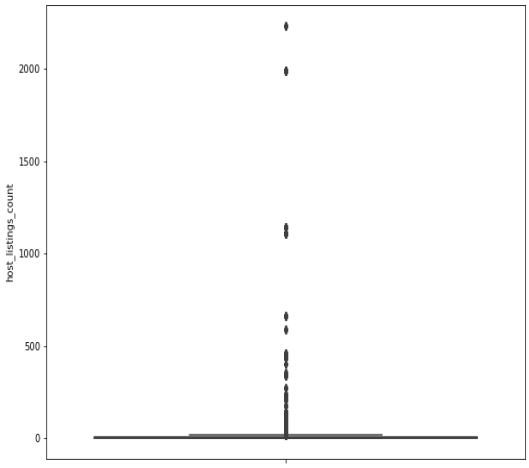




- Host acceptance rate time is left skewed.
- It is leptokurtic, and has wide tail
- IQR lies from 1.0 to 0.8.
- Outliers are present

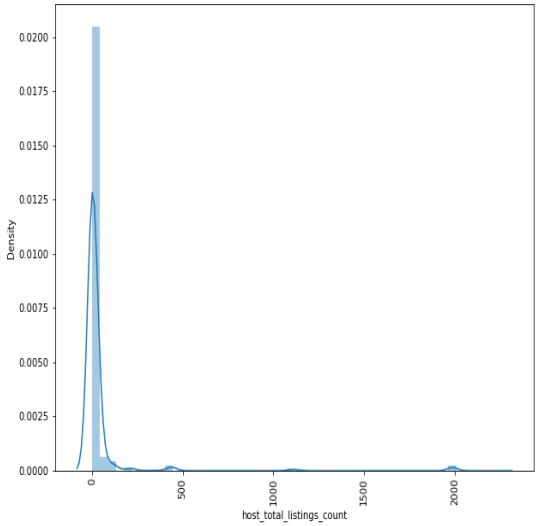
c. Host_listings_count

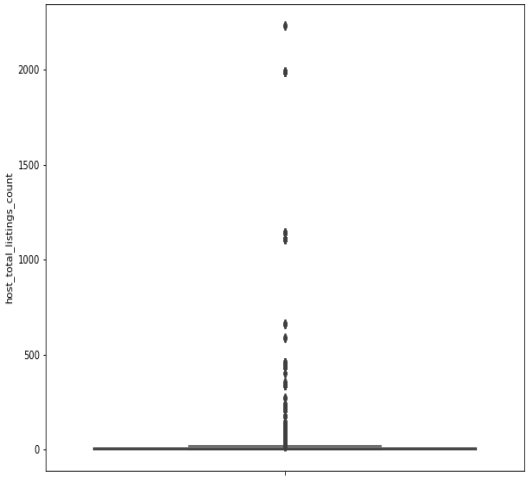




- Host listings count is right skewed.
- It is leptokurtic, and has wide tail
- IQR lies at 0.
- Outliers are present

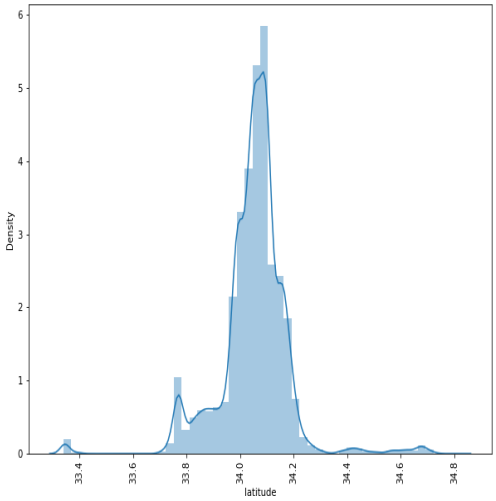
d. Host_total_listings_count

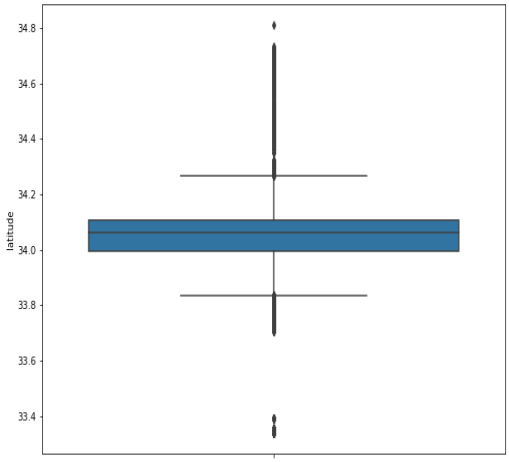




- Host total listings count is right skewed.
- it is leptokurtic, and has wide tail
- IQR lies at 0.
- Outliers are present.

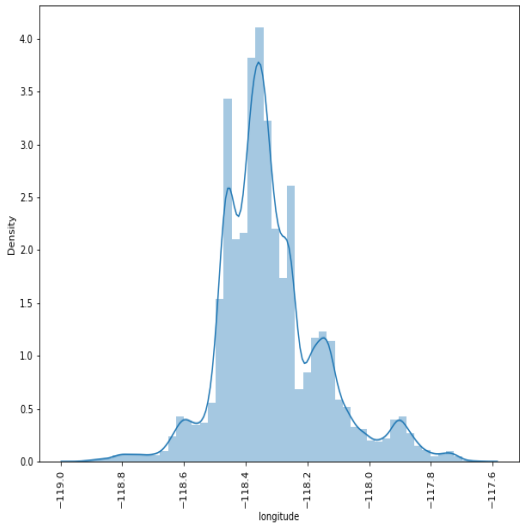
e. Latitude

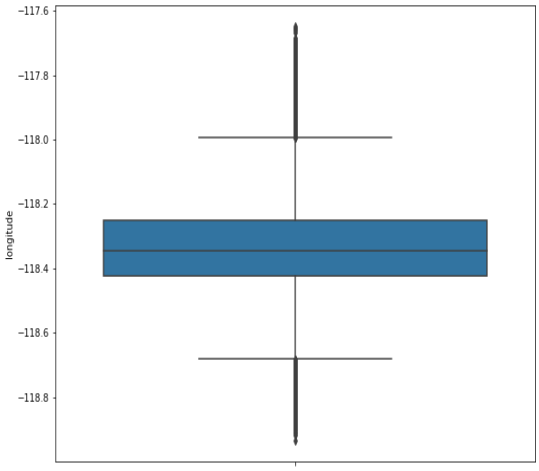




- Latitude has normal distribution.
- It is leptokurtic, and has wide tail
- IQR lies from 33.8 to 34.3.
- Outliers are present

e. Longtitude

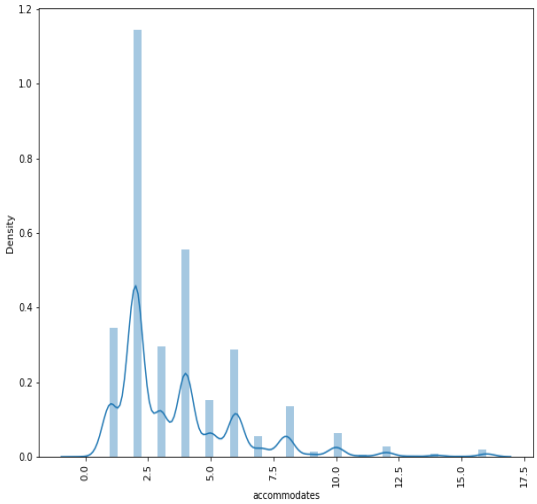


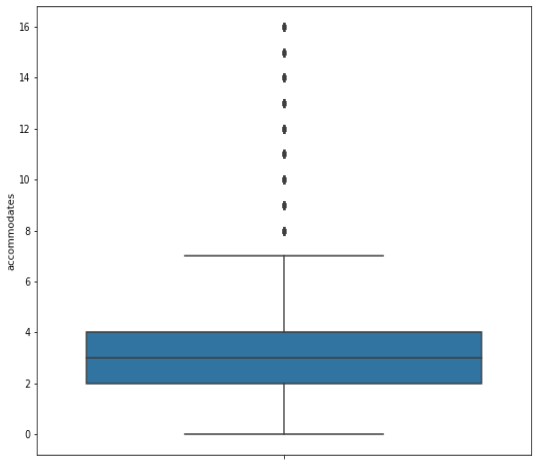


- Longitude has right skew.

- It is leptokurtic, and has wide tail
- IQR lies from -118.6 to 118.
- Outliers are present.

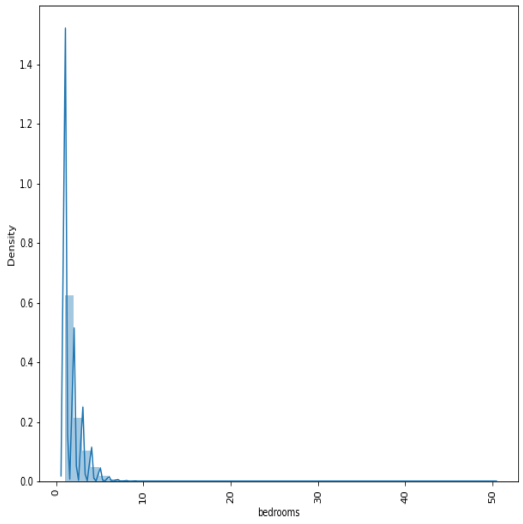
f. Accommodates

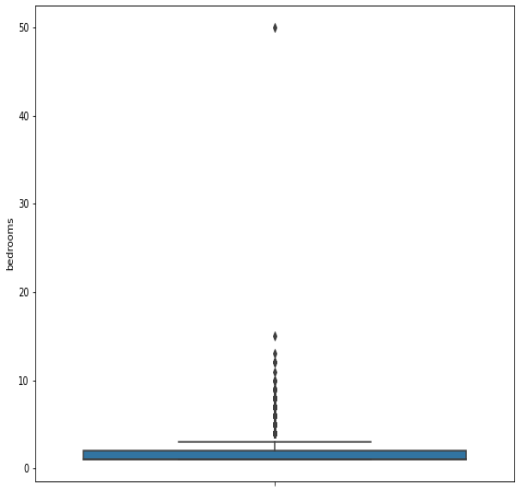




- Accommodates has right skew.
- It is leptokurtic, and has wide tail
- IQR lies from 0 to 7.
- Outliers are present.

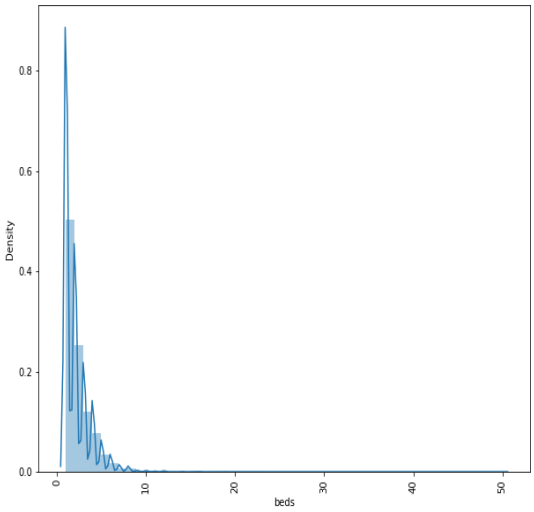
g. Bedrooms

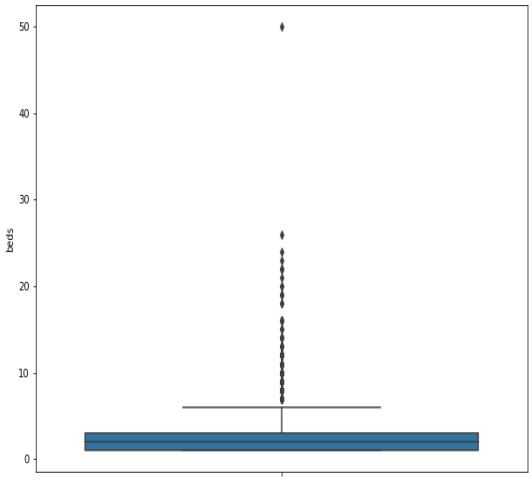




- Bedrooms has right skew.
- It is leptokurtic, and has wide tail
- IQR lies from 0 to 3.
- Outliers are present.

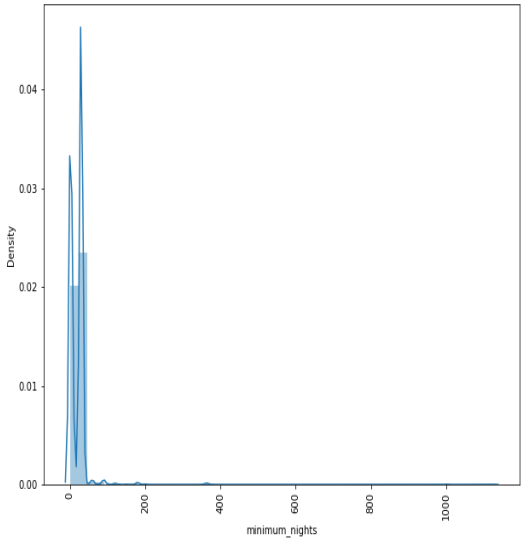
h. Beds

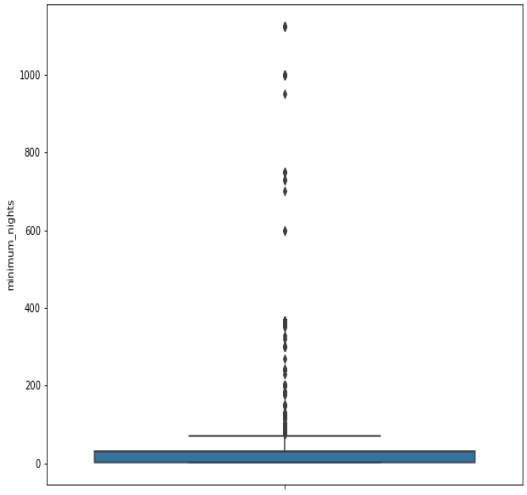




- Beds has right skew.
- It is leptokurtic, and has wide tail
- IQR lies from 0 to 7.
- Outliers are present.

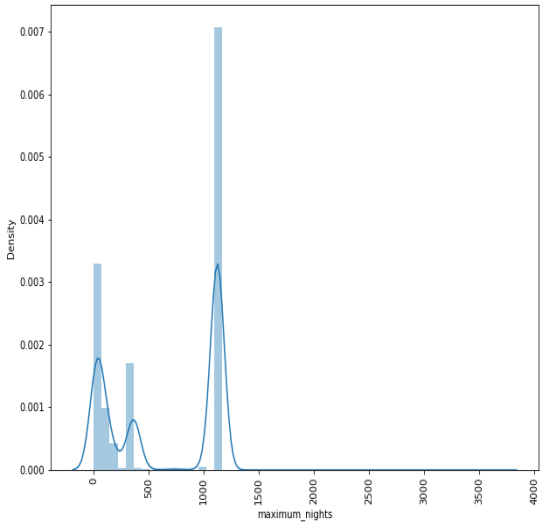
i. Minimum_nights

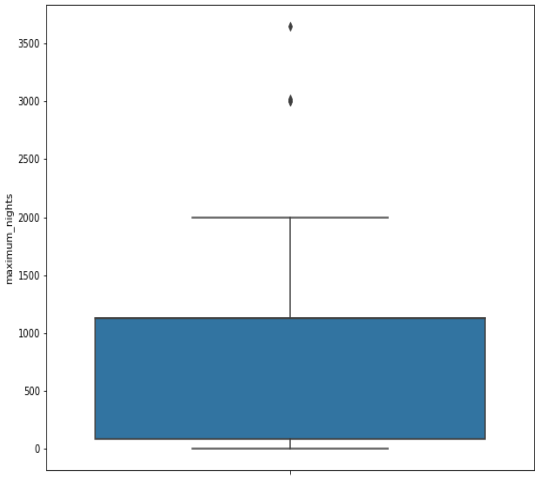




- Beds has right skew.
- It is leptokurtic, and has wide tail
- IQR lies from 0 to 100.
- Outliers are present.

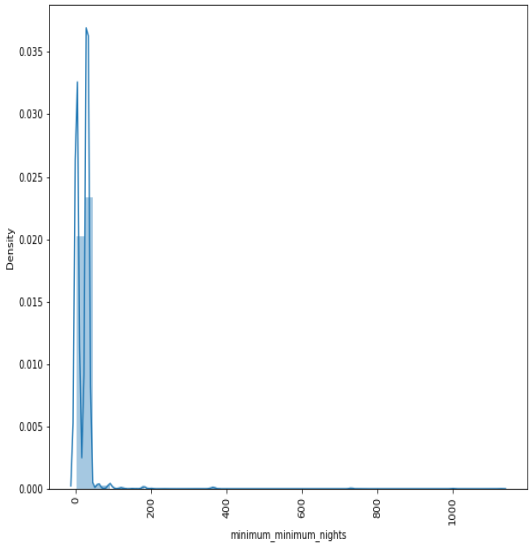
j. Maximum_nights

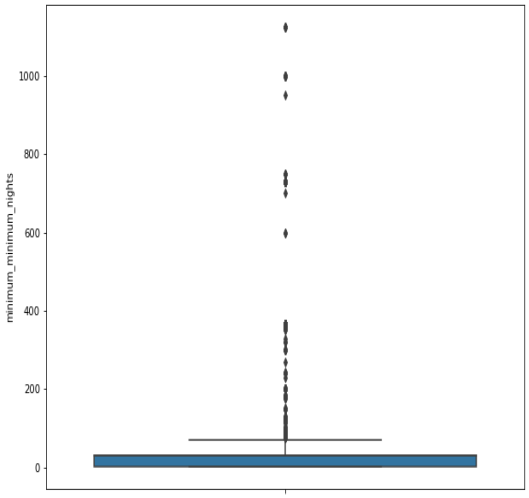




- Maximum nights is normal distributed.
- It is platykurtic, and has wide tail
- IQR lies from 0 to 1100.
- Only few outliers are present.

k. Minimum_minimum_nights

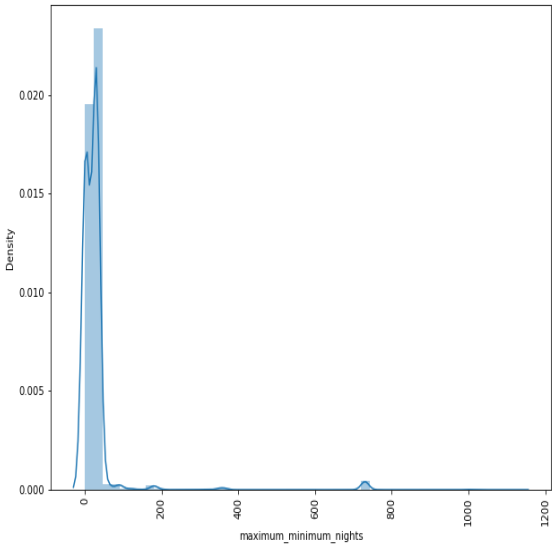


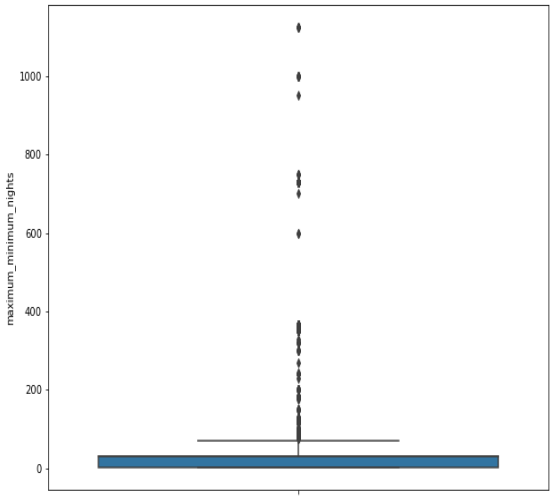


- Minimum minimum nights is right skewed.
- It is leptokurtic, and has narrow tailed.

- IQR lies from 0 to 100.
- Outliers are present.

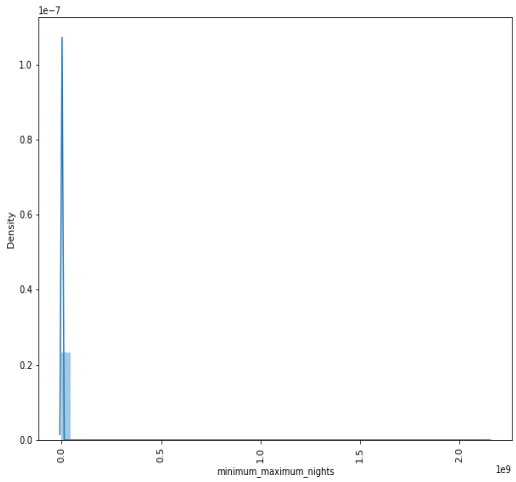
l. Maximum_minimum_nights

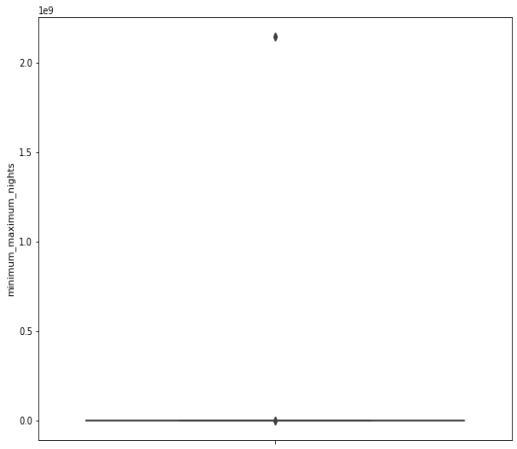




- Maximum minimum nights is right skewed.
- It is leptokurtic, and has wide tail
- IQR lies from 0 to 100.
- Outliers are present.

m. Minimum_maximum_nights

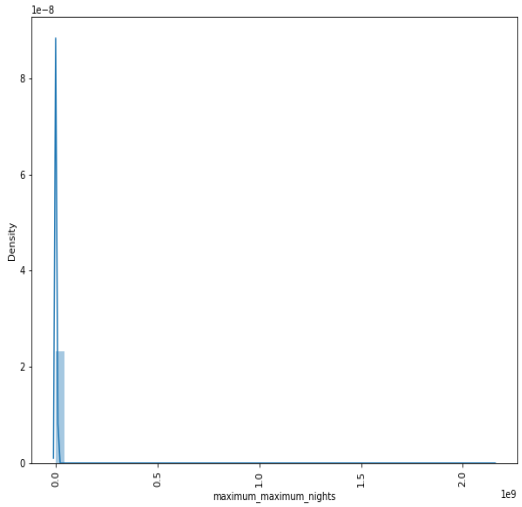


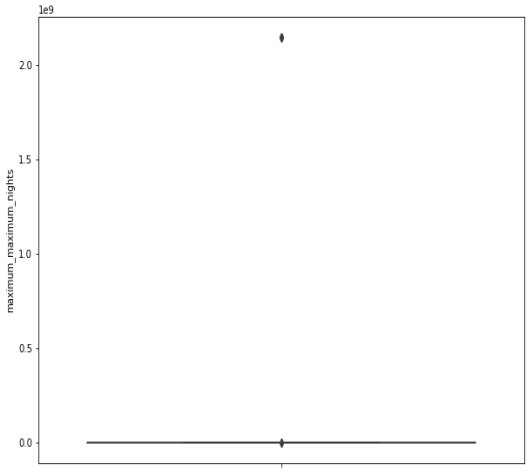


- Minimum maximum nights is right skewed.

- It is leptokurtic, and has narrow tailed.
- IQR lies at 0.
- Only few outliers are present.

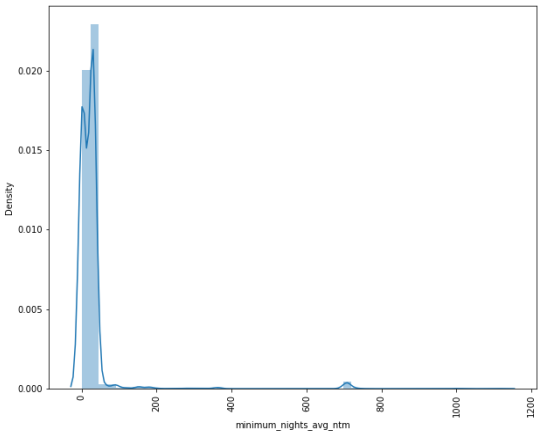
n. Maximum_maximum_nights

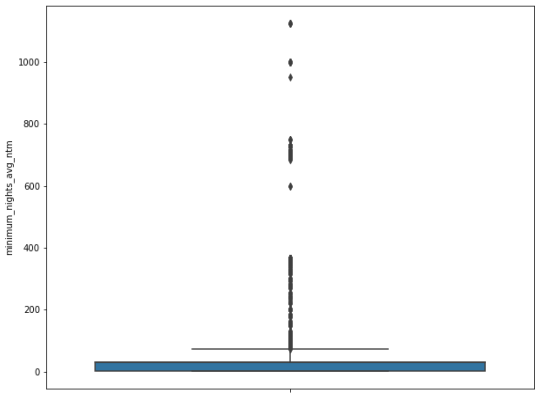




- Maximum maximum nights is right skewed.
- It is leptokurtic, and narrow tailed.
- IQR lies at 0.
- Only few outliers are present.

o. Minimum_nights_avg_ntm

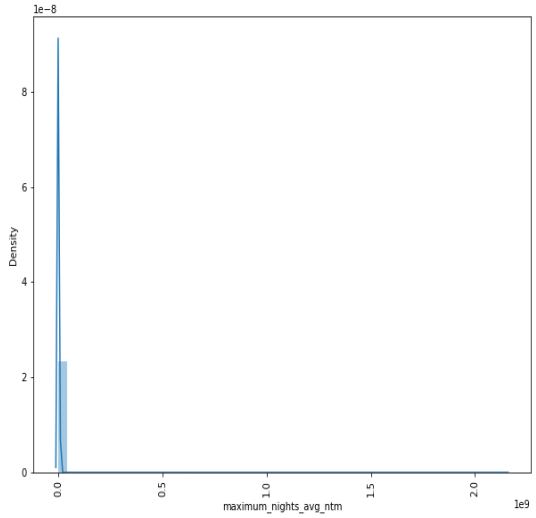


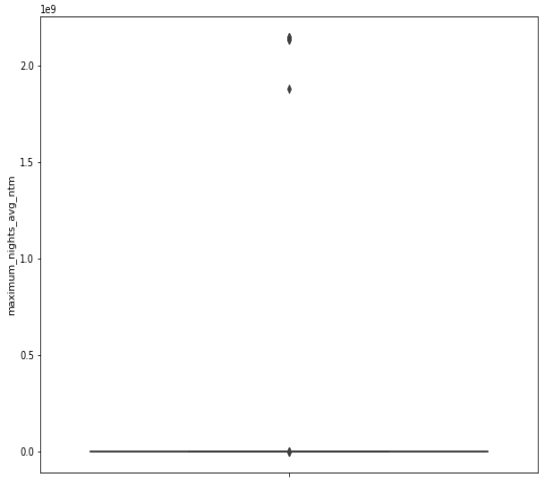


- Minimum nights avg ntm is right skewed.

- It is leptokurtic, and wide tailed.
- IQR lies from 0 to 100.
- Outliers are present.

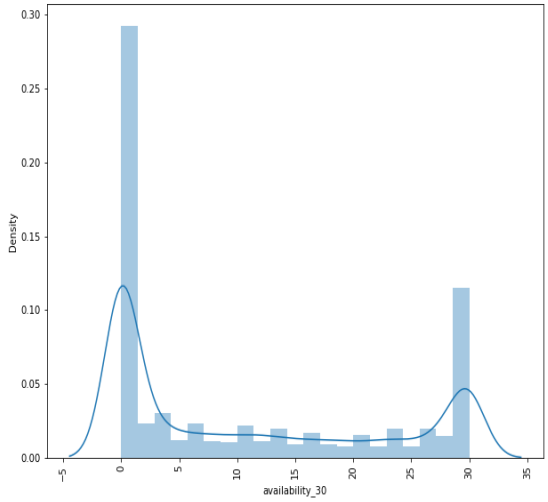
p. Maximum_nights_avg_ntm

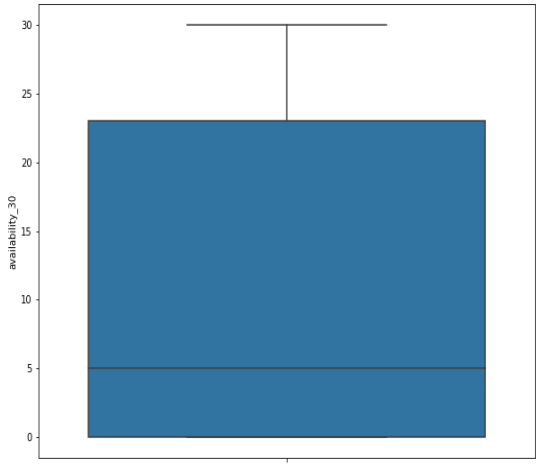




- Maximum nights avg ntm is right skewed.
- It is leptokurtic, and narrow tailed.
- IQR lies from 0.
- Only few outliers are present.

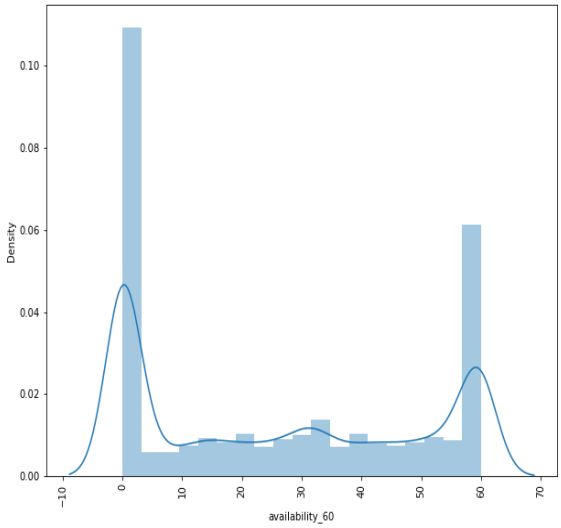
q. Availability_30

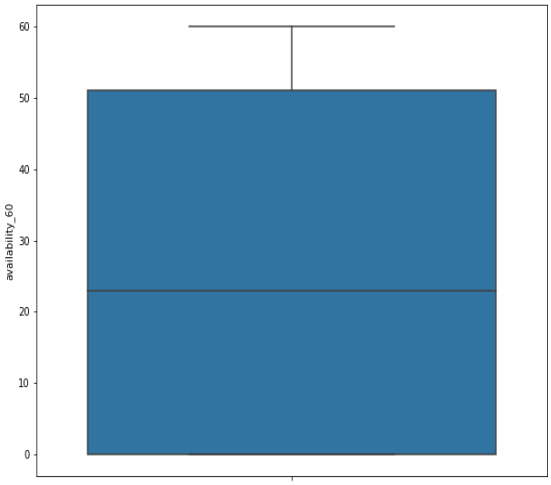




- Availability_30 is right skewed.
- It is Platykurtic, and wide tailed.
- IQR lies from 0 to 24.
- No outliers are present.

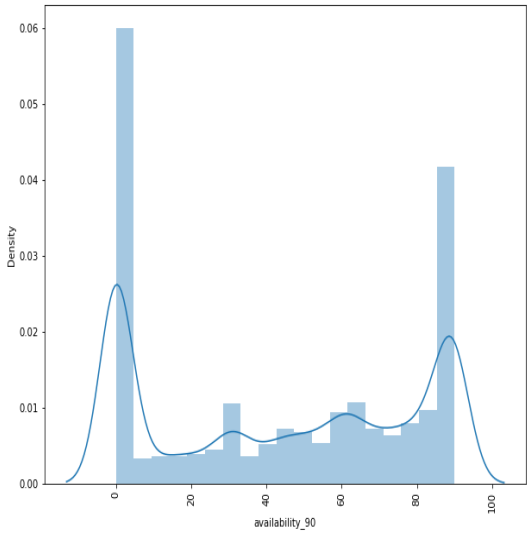
r. Availability_60

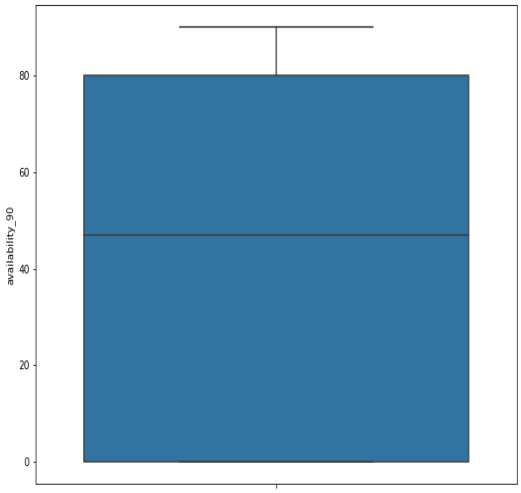




- Availability_60 is normally distributed.
- It is platykurtic, and wide tailed.
- IQR lies from 0 to 50.
- No outliers are present.

s. Availability_90

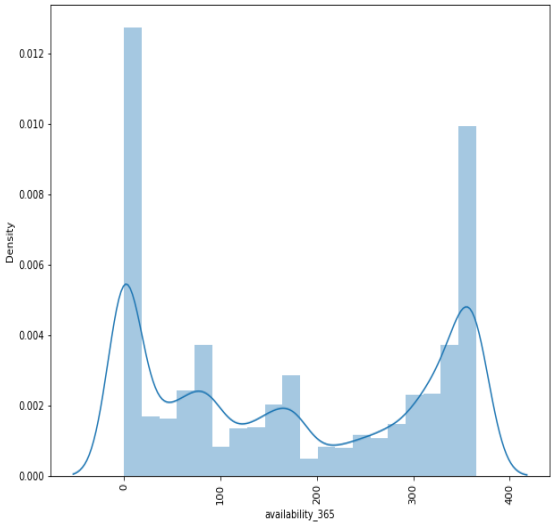


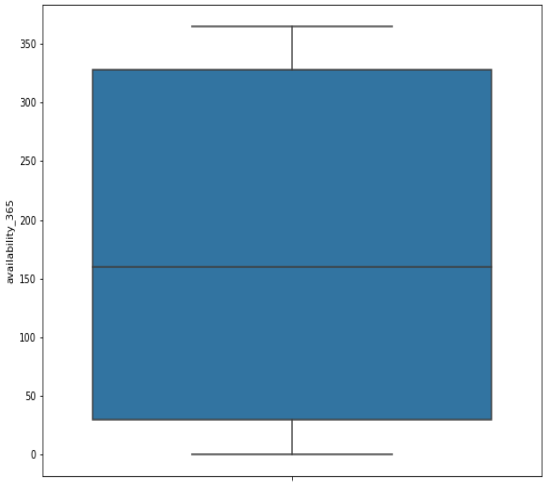


- Availability_60 is normally distributed.

- It is platykurtic, and wide tailed.
- IQR lies from 0 to 80.
- No outliers are present.

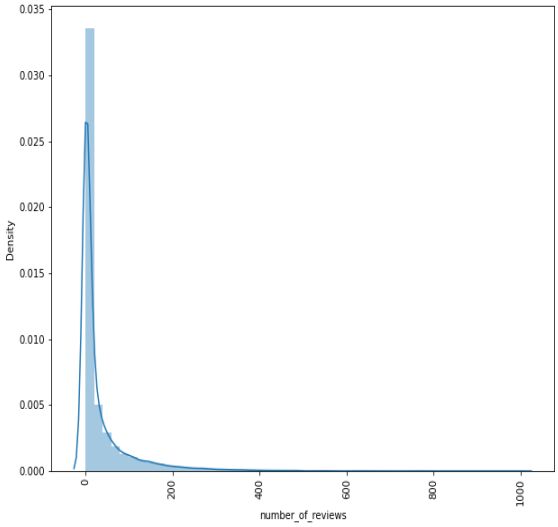
t. Availability_365

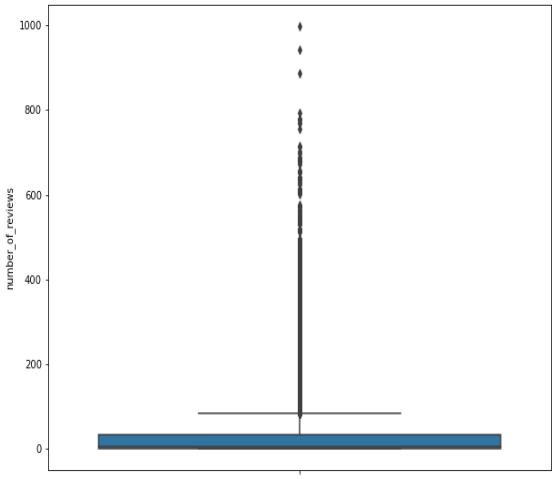




- Availability_365 is normally distributed.
- It is platykurtic, and wide tailed.
- IQR lies from 0 to 350.
- No outliers are present.

u. Number_of_reviews

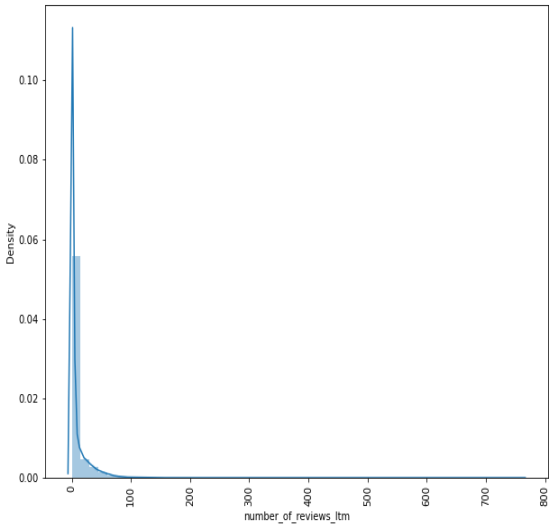


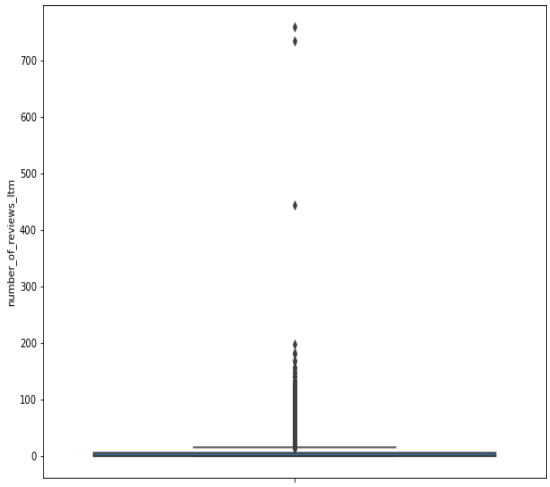


- Number of reviews is rightly skewed.
- It is Leptokurtic, and wide tailed.

- IQR lies from 0 to 100.
- Outliers are present.

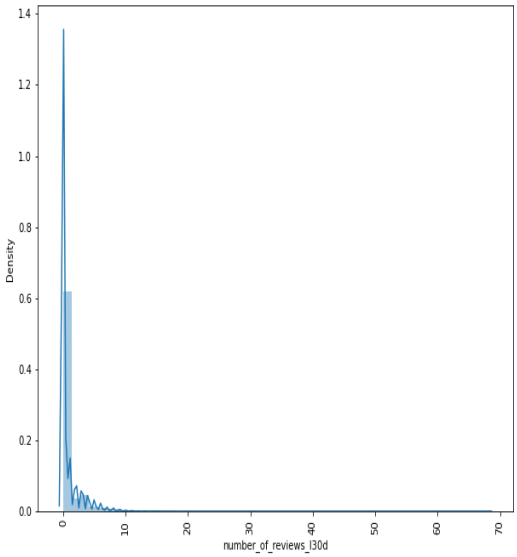
v. Number_of_reviews_itm

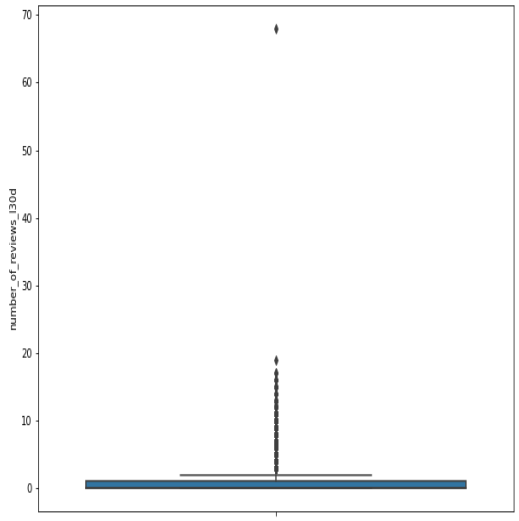




- Number of reviews itm is rightly skewed.
- It is Leptokurtic, and narrow tailed.
- IQR lies from 0 to 10.
- Outliers are present.

w. Number_of_reviews_I30d

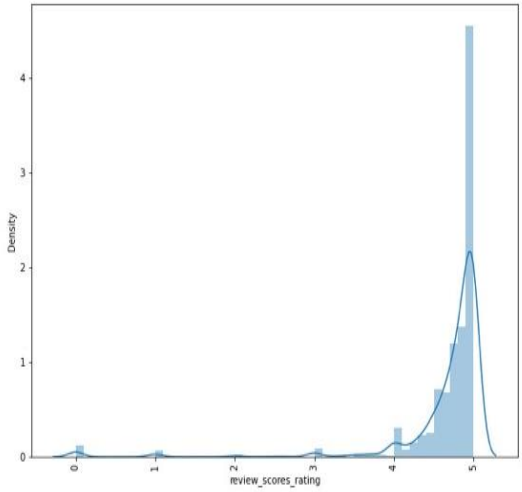


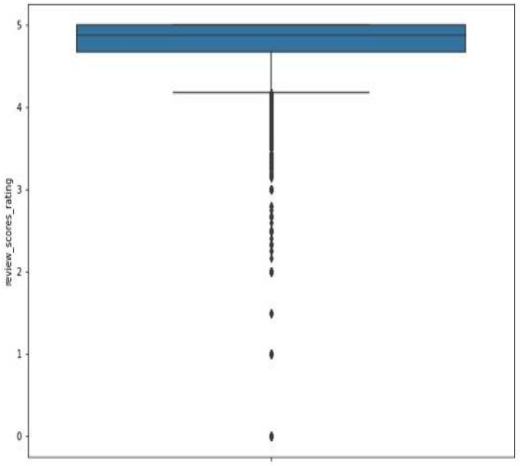


- Number of reviews itm is rightly skewed.

- It is Leptokurtic, and narrow tailed.
- IQR lies from 0 to 10.
- Outliers are present.

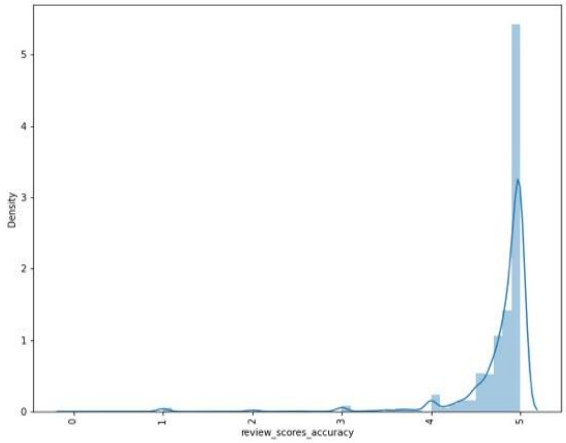
x. Review_scores_rating

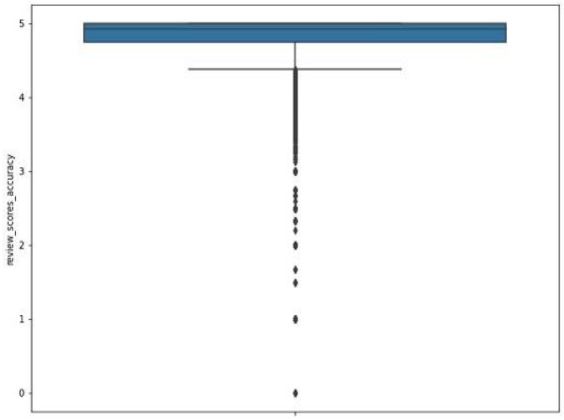




- Review scores rating is left skewed.
- It is Leptokurtic, and wide tailed.
- IQR lies from 5 to 4.
- Outliers are present.

y. Review_scores_accuracy

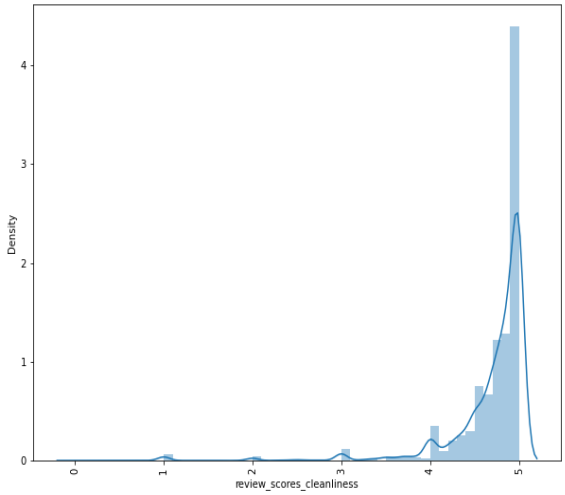


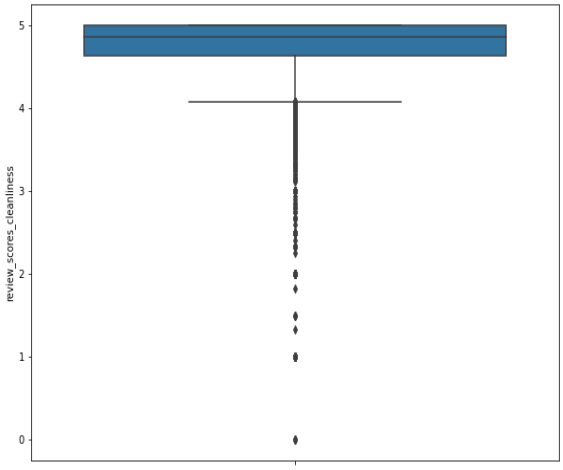


- Review scores accuracy is left skewed.
- It is Leptokurtic, and wide tailed.
- IQR lies from 5 to 4.
- Outliers are present.

-

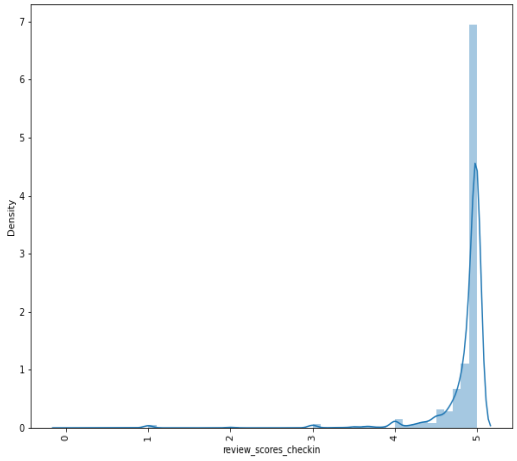
z. Review_scores_cleanliness

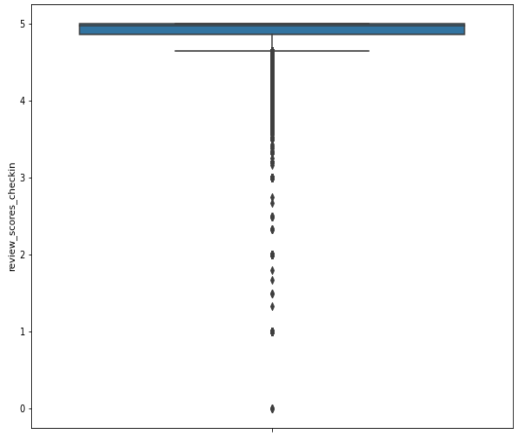




- Review scores cleanliness is left skewed.
- It is Leptokurtic, and wide tailed.
- IQR lies from 5 to 4.
- Outliers are present.

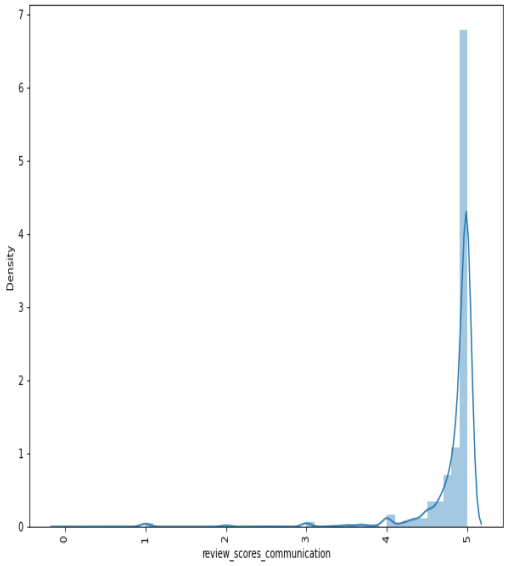
aa. Review_scores_checkin

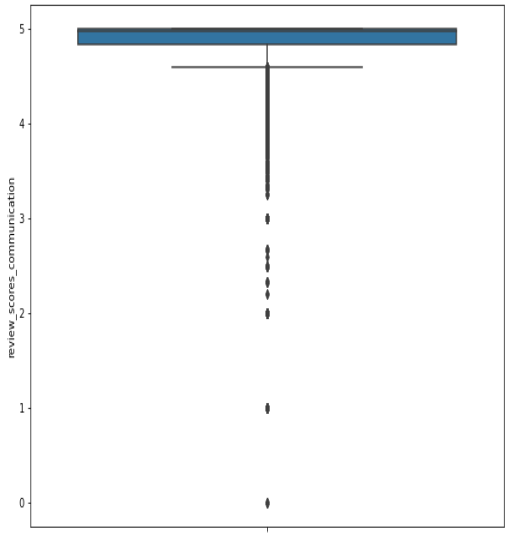




- Review scores checkin is left skewed.
- It is Leptokurtic, and wide tailed.
- IQR lies from 5 to 4.
- Outliers are present.

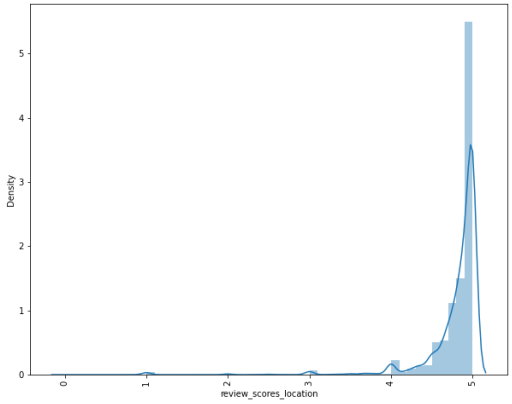
bb. Review_scores_communication

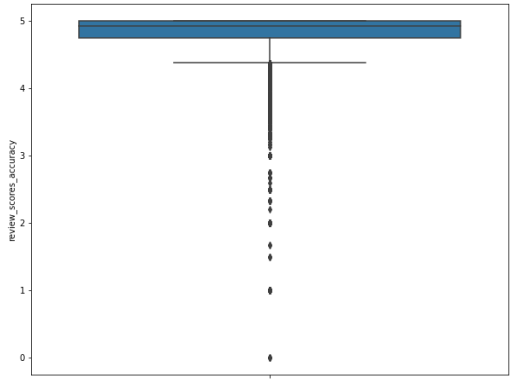




- Review scores communication is left skewed.
- It is Leptokurtic, and wide tailed.
- IQR lies from 5 to 4.
- Outliers are present.

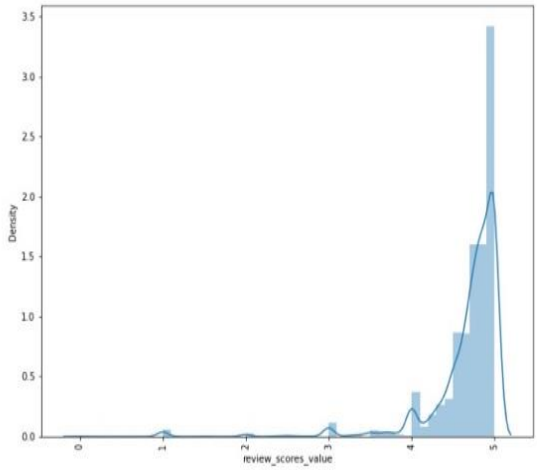
cc. Review_scores_location

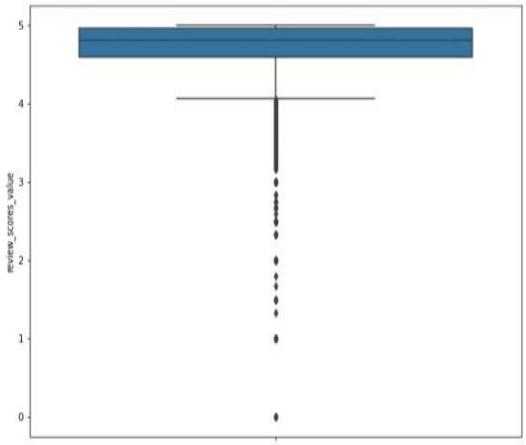




- Review scores location is left skewed.
- It is Leptokurtic, and wide tailed.
- IQR lies from 5 to 4.
- Outliers are present.

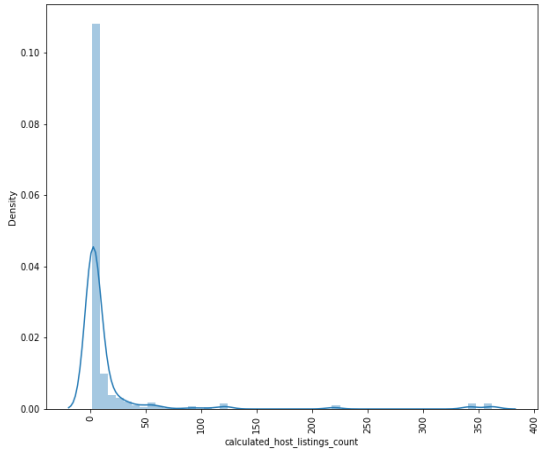
dd. Review_scores_value

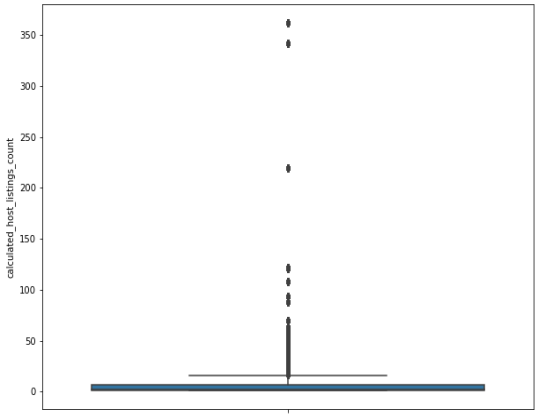




- Review scores value is left skewed.
- It is Leptokurtic, and wide tailed.
- IQR lies from 5 to 4.
- Outliers are present.

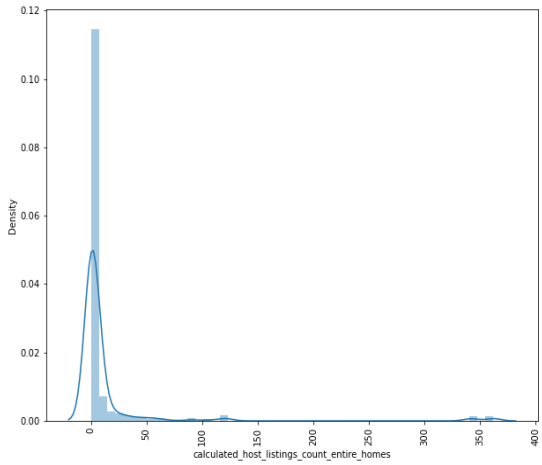
ee. Calculated_host_listings_count

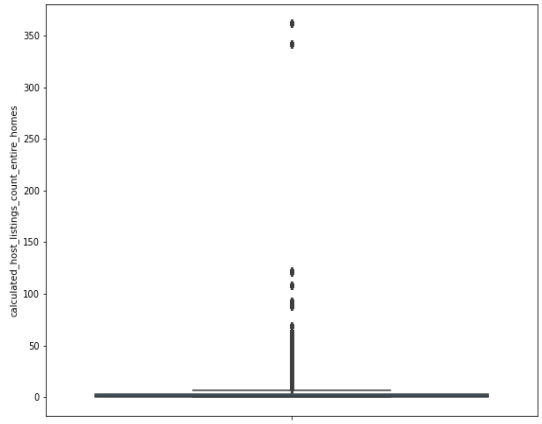




- Calculated host listings count is right skewed.
- It is Leptokurtic, and narrow tailed.
- IQR lies from 0 to 20.
- Outliers are present.

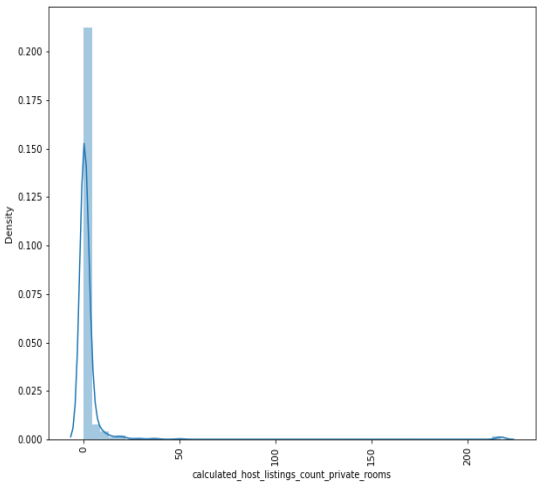
ff. Calculated_host_listings_count_entire_homes

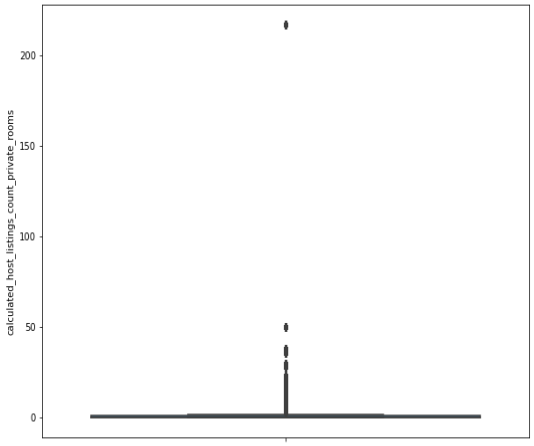




- Calculated host listings count entire homes is right skewed.
- It is Leptokurtic, and narrow tailed.
- IQR lies from 0 to 10.
- Outliers are present.

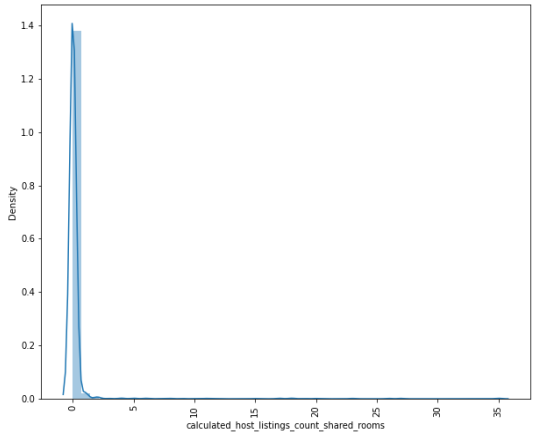
gg. Calculated_host_listings_count_private_rooms

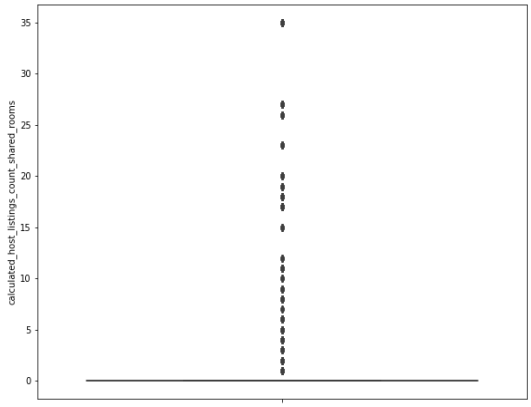




- Calculated host listings count entire homes is right skewed.
- It is Leptokurtic, and narrow tailed.
- IQR lies at 0 .
- Outliers are present.

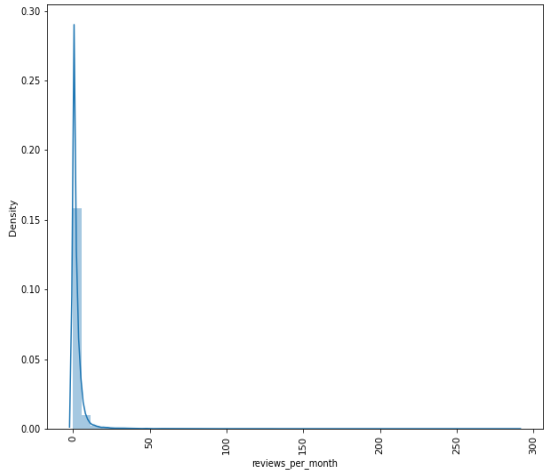
hh. Calculated_host_listings_count_shared_rooms

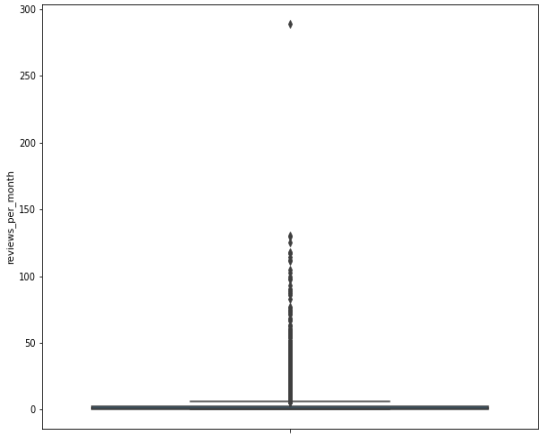




- Calculated host listings count shared rooms is right skewed.
- It is Leptokurtic, and narrow tailed.
- IQR lies at 0 .
- Outliers are present.

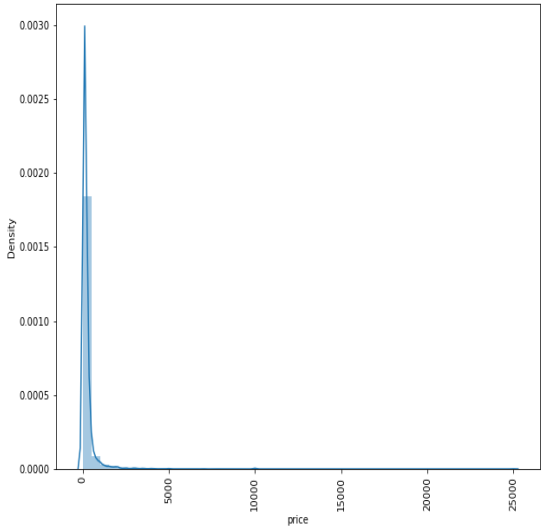
ii. Reviews_per_month

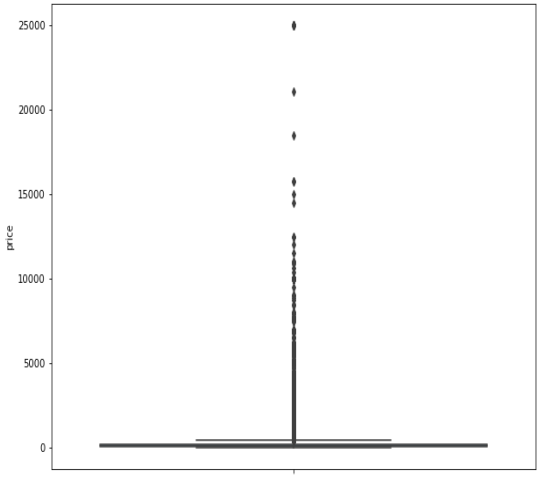




- Reviews per month is right skewed.
- It is Leptokurtic, and narrow tailed.
- IQR lies from 0 to 10 .
- Outliers are present.

jj. Price

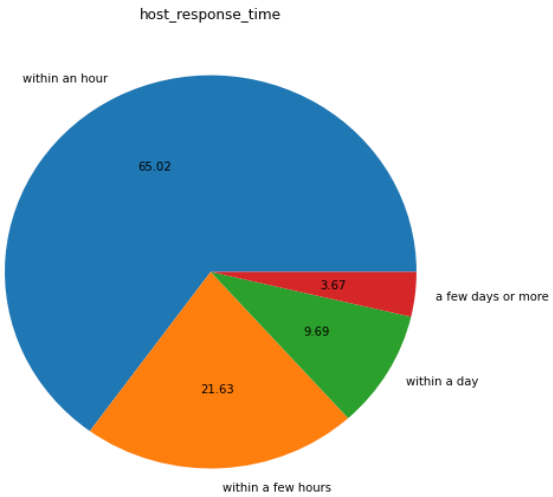




- Price is right skewed.
- It is Leptokurtic, and narrow tailed.
- IQR lies at 0 .
- Outliers are present.

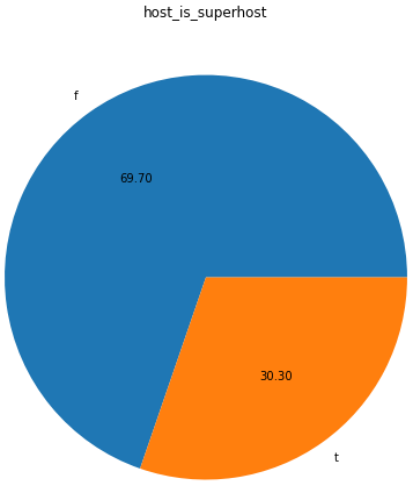
1.2 Categorical:

a. Host_response_time



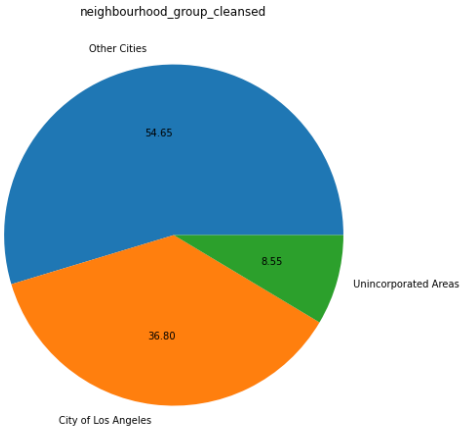
Inference: 65.02% of the host response time were handled within an hour followed 21.63% are handled within a few hours from the time of call.

b. Host_is_superhost



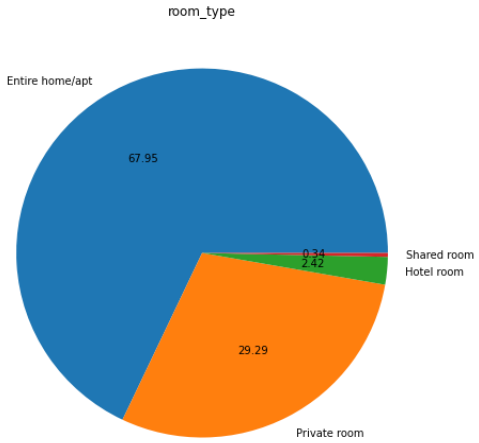
Inference: Among the host of the properties, 69.70% is super host and 30.30% are not super host.

c. Neighbourhood_group_cleanse:



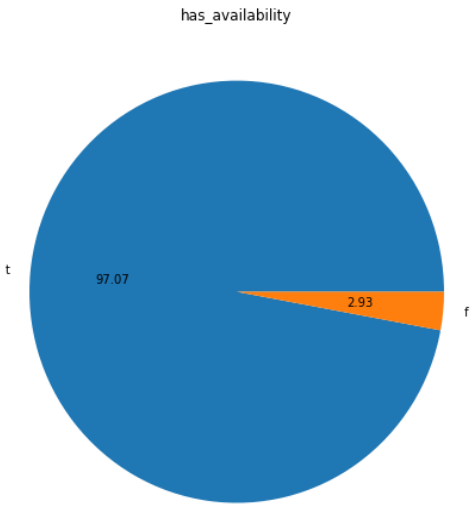
Inference: 54.65% of other cities have high percentage of neighborhood group cleansed whereas the city of Los Angeles has second highest percentage of 36.80% and 8.55% of Unincorporated Areas.

d. Room_type



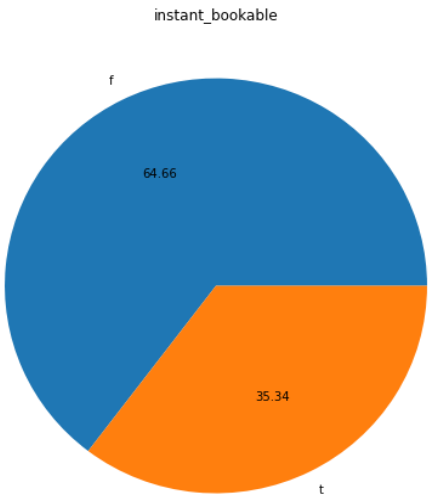
Inference: 67.95% of room type are belong to Entire home/apt, 29.29 is private room, 2.42 is hotel room and 0.34 is shared room.

e. Has_availability



Inference: When the customer-tired booking via Airbnb had the chance of booking availability of 97.07% and 2.93 of non-availability chances.

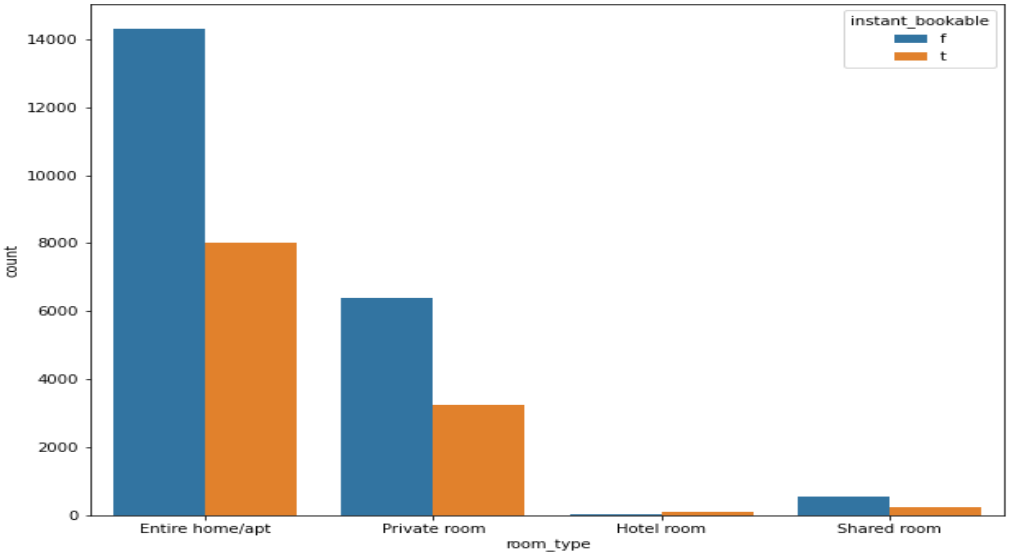
f. Instant_bookable



Inference: Customers were able to book the room at an instance basis of 64.66% and remaining 35.34% were not able to book at instance basis.

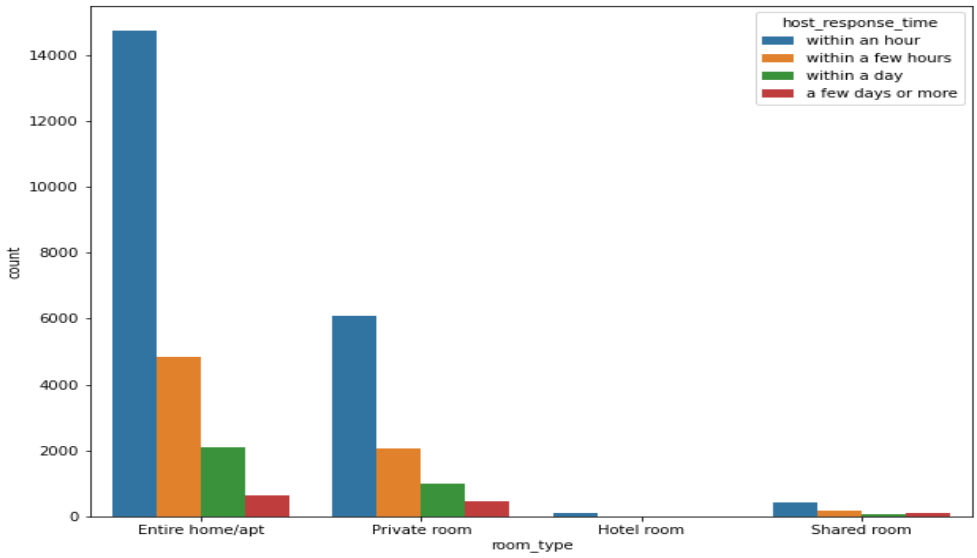
2. BIVARIATE ANALYSIS:

2. a. Room_type vs instant_bookable:



Inference: From the above bar plot, we could interpret that the category of Entire/Apt and Private Rooms are booked highly at an instant basis compare to hotel rooms and shared rooms.

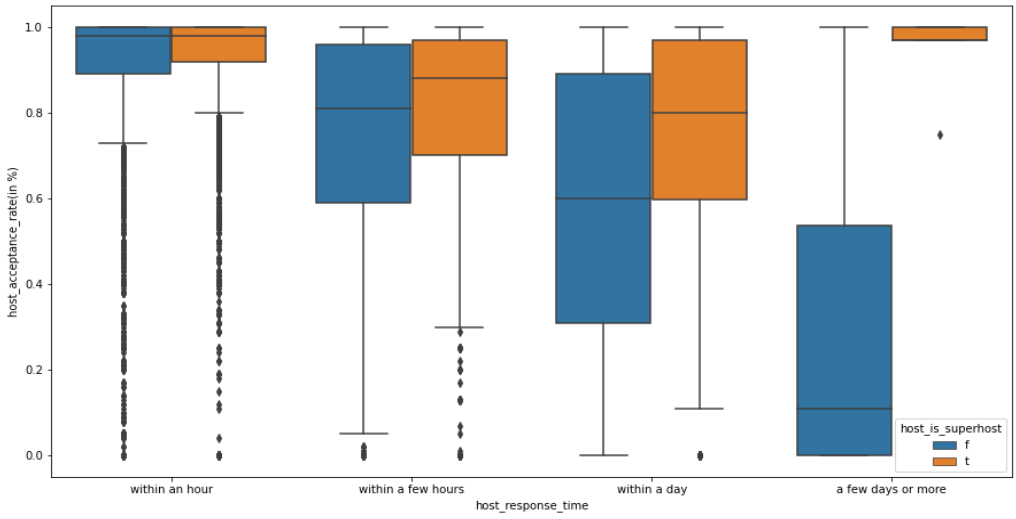
2. b. Room_type vs Host_respose_time:



Inference: From the above bar plot, we interpret that the high calls were made for entire home/apt and private rooms categories compare to hotel room and shared room. Most of these calls were handled within an hour and few hours from the time of calls.

3. MULTIVARIATE ANALYSIS:

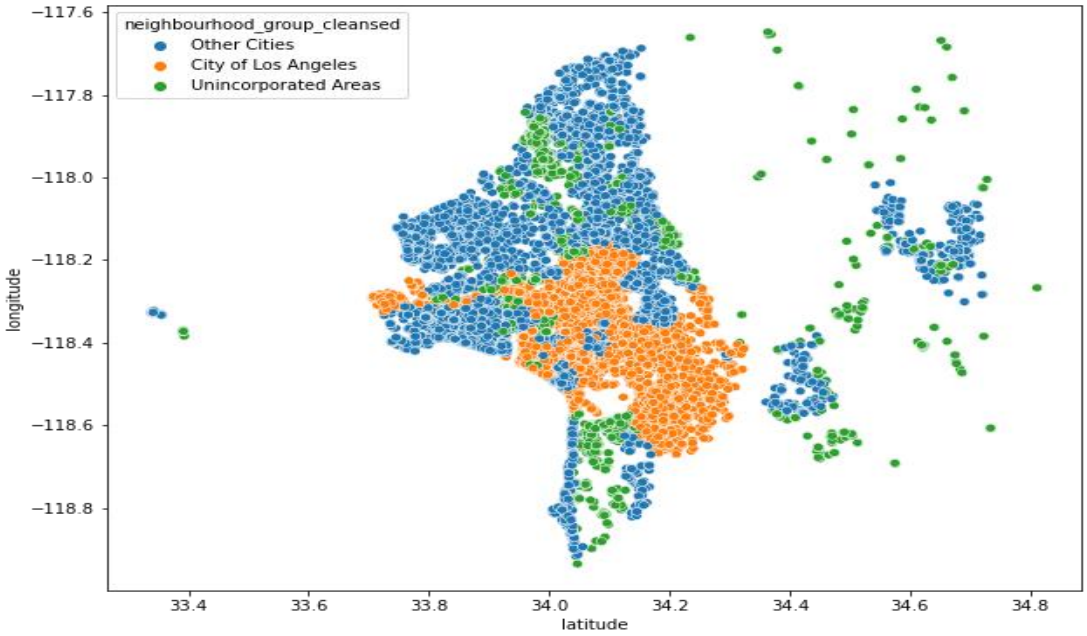
3.a. Host_acceptance_rate vs Host_response_time vs Host_is_superhost:



Inference: From the above graph, we could interpret that both Super host and non-Super host mostly took a day to accept the bookings and sometimes the super host also took few days to

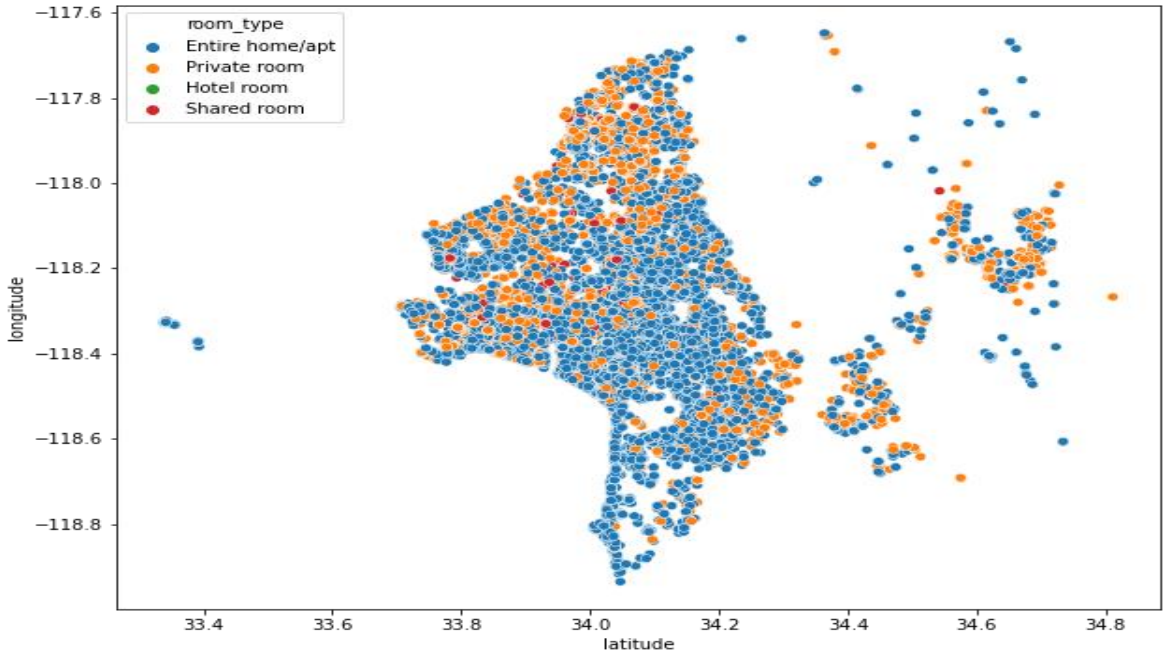
accept bookings. Whereas the acceptance rate within an hour and few hours are very less compare to with a day and few days categories.

3.b. Latitude vs Longitude vs Neighbourhood_group_cleansed:



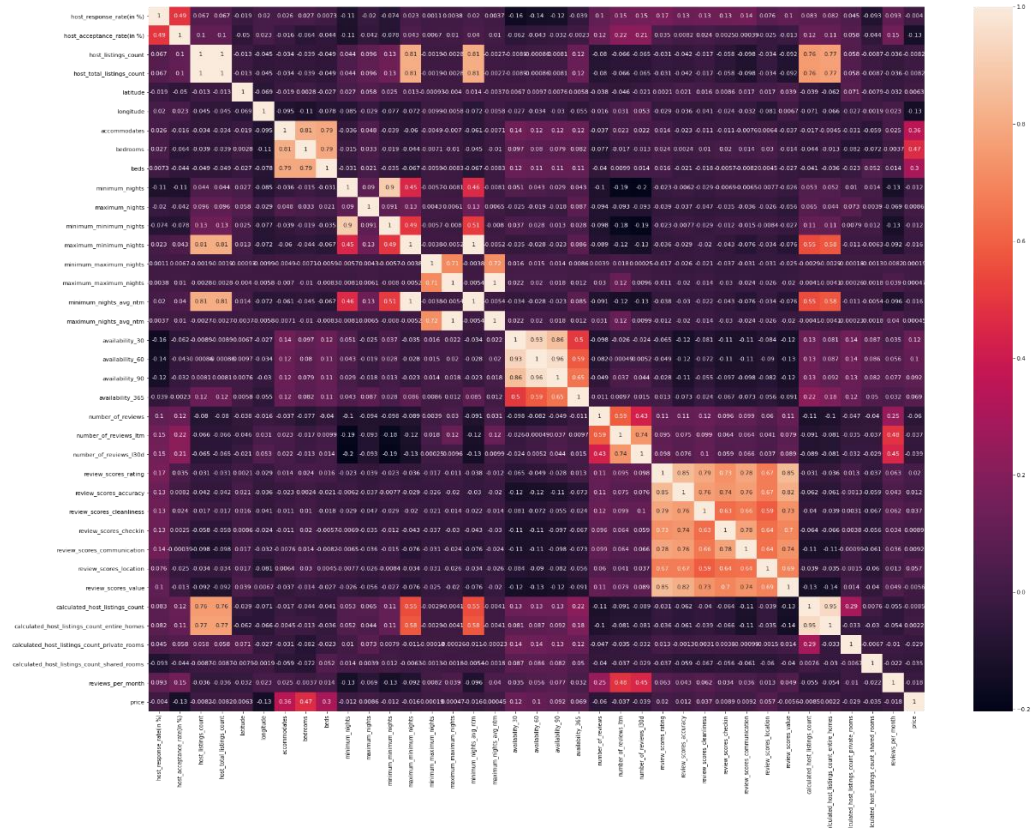
Inference: From the above graph, we could interpret that the high cleansed neighborhood group are from other cities followed by city of Los Angeles and finally the Unincorporated Areas.

3.c. Latitude vs Longitude vs Room_type:



Inference: From the above graph, we could interpret that most of the host are hosting majorly the entire home/apt followed private room and very negligible amount of hotel and shared room.

CORRELATION MATRIX



Inference:

1.Highly co-related variables

host_total_listings_count vs host_listings_count, Beds vs Bedroom, Beds vs Accommodates, Bedroom vs Accommodates, maximum_minimum nights vs host_listings_count, minimum_nights_avg_ntm vs host_listings_count, minimum_nights_avg_ntm vs host_total_listings_count, maximum_minimum nights vs host_total_listings_count, minimum minimum nights vs minimum nights, calculated_host_listings_count vs calculated_host_listings_count_entire_homes.

2. Moderate co-related Variables:

Host response rate vs Host acceptance rate, Maximum Minimum nights vs minimum nights, Maximum minimum nights vs Minimum Minimum Nights, calculated_host_listings_count vs Maximum minimum nights.

STATISTISTICAL TEST:

a. NUMERICAL VARIABLE VS NUMERICAL VARIABLE

1: Independent Variable = host_response_rate : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference: The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between host_response_rate and Price.

HA: There is a relationship between host_response_rate and Price.

Inference: Accept null hypothesis, the variable - host_response_rate is not a significant feature to predict the target column – Price

2: Independent Variable = host_acceptance_rate : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference: The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between host_acceptance_rate and Price.

HA: There is a relationship between host_acceptance_rate and Price.

Inference: Accept null hypothesis, the variable - host_acceptance_rate is not a significant feature to predict the target column – Price

3: Independent Variable = host_total_listings_count : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference : The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between host_total_listings_count and Price.

HA: There is a relationship between host_total_listings_count and Price.

Inference : Accepting to reject null hypothesis, the variable - host_total_listings_count is a significant feature to predict the target column – Price

4: Independent Variable = latitude : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference : The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between latitude and Price.

HA: There is a relationship between latitude and Price.

Inference: Accepting to reject null hypothesis, the variable - latitude is a significant feature to predict the target column – Price

5: Independent Variable = longitude : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference : The data is not normally distributed, hence using non parametric test - SpearMan C orrelation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between longitude and Price.

HA: There is a relationship between longitude and Price.

Inference : Accepting to reject null hypothesis, the variable - longitude is a significant feature to predict the target column - Price

6: Independent Variable = accommodates : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference : The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between accommodates and Price.

HA: There is a relationship between accommodates and Price.

Inference: Accepting to reject null hypothesis, the variable - accommodates is a significant feature to predict the target column - Price

7: Independent Variable = bathrooms_text : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference : The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between bathrooms_text and Price.

HA: There is a relationship between bathrooms_text and Price.

Inference: Accepting to reject null hypothesis, the variable - bathrooms_text is a significant feature to predict the target column - Price

8: Independent Variable = bedrooms : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference: The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between bedrooms and Price.

HA: There is a relationship between bedrooms and Price.

Inference: Accepting to reject null hypothesis, the variable - bedrooms is a significant feature to predict the target column - Price

9: Independent Variable = beds : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference: The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between beds and Price.

HA: There is a relationship between beds and Price.

Inference: Accepting to reject null hypothesis, the variable - beds is a signficant feature to predict the target column - Price

10: Independent Variable = amenities : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference : The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between amenities and Price.

HA: There is a relationship between amenities and Price

Inference: Accepting to reject null hypothesis, the variable - amenities is a signficant feature to predict the target column - Price

11 Independent Variable = minimum_nights : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference : The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between minimum nights and Price.

HA: There is a relationship between minimum_nights and Price.

Inference : Accepting to reject null hypothesis, the variable - minimum_nights is a signficant feature to predict the target column - Price

12: Independent Variable = maximum_nights : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference : The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between maximum_nights and Price.

HA: There is a relationship between maximum_nights and Price.

Inference : Accepting to reject null hypothesis, the variable - maximum_nights is a signficant feature to predict the target column - Price

13: Independent Variable = minimum_minimum_nights : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference : The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between minimum_minimum_nights and Price.

HA: There is a relationship between minimum_minimum_nights and Price.

Inference : Accepting to reject null hypothesis, the variable - minimum_minimum_nights is a signficant feature to predict the target column - Price

14: Independent Variable = maximum_minimum_nights : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference : The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between maximum_minimum_nights and Price.

HA: There is a relationship between maximum_minimum_nights and Price.

Inference : Accepting to reject null hypothesis, the variable - maximum_minimum_nights is a signficant feature to predict the target column - Price

15: Independent Variable = minimum_maximum_nights : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference: The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between minimum_maximum_nights and Price.

HA: There is a relationship between minimum_maximum_nights and Price.

Inference : Accepting to reject null hypothesis, the variable - minimum_maximum_nights is a signficant feature to predict the target column – Price

16: Independent Variable = maximum_maximum_nights : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference : The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between maximum_maximum_nights and Price.

HA: There is a relationship between maximum_maximum_nights and Price.

Inference: Accept null hypothesis, the variable - maximum_maximum_nights is not a significant feature to predict the target column - Price

17: Independent Variable = minimum_nights_avg_ntm : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference : The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between minimum_nights_avg_ntm and Price.

HA: There is a relationship between minimum_nights_avg_ntm and Price.

Inference : Accepting to reject null hypothesis, the variable - minimum_nights_avg_ntm is a signficant feature to predict the target column – Price

18: Independent Variable = minimum_nights_avg_ntm : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference : The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between minimum_nights_avg_ntm and Price.

HA: There is a relationship between minimum_nights_avg_ntm and Price.

Inference : Accepting to reject null hypothesis, the variable - minimum_nights_avg_ntm is a signficant feature to predict the target column - Price

19: Independent Variable = availability_30 : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference : The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between availability_30 and Price.

HA: There is a relationship between availability_30 and Price.

Inference: Accepting to reject null hypothesis, the variable - availability_30 is a signficant feature to predict the target column - Price

20: Independent Variable = availability_60 : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference : The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between availability_60 and Price.

HA: There is a relationship between availability_60 and Price.

Inference: Accepting to reject null hypothesis, the variable - availability_60 is a signficant feature to predict the target column - Price

21: Independent Variable = availability_90 : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference : The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between availability_90 and Price.

HA: There is a relationship between availability_90 and Price.

Inference: Accepting to reject null hypothesis, the variable - availability_90 is a signficant feature to predict the target column – Price

22: Independent Variable = availability_365 : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference: The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between availability_365 and Price.

HA: There is a relationship between availability_365 and Price.

Inference: Accepting to reject null hypothesis, the variable - availability_365 is a signficant feature to predict the target column – Price

23: Independent Variable = number_of_reviews : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference: The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between number_of_reviews and Price.

HA: There is a relationship between number_of_reviews and Price.

Inference: Accepting to reject null hypothesis, the variable - number_of_reviews is a significant feature to predict the target column - Price

24: Independent Variable = number_of_reviews_ltm : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference : The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between number_of_reviews_ltm and Price.

HA: There is a relationship between number_of_reviews_ltm and Price.

Inference: Accepting to reject null hypothesis, the variable - number_of_reviews_ltm is a significant feature to predict the target column - Price

25: Independent Variable = number_of_reviews_l30d : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference : The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between number_of_reviews_l30d and Price.

HA: There is a relationship between number_of_reviews_l30d and Price.

Inference: Accepting to reject null hypothesis, the variable - number_of_reviews_l30d is a significant feature to predict the target column - Price

26: Independent Variable = review_scores_rating : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference: The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between review_scores_rating and Price.

HA: There is a relationship between review_scores_rating and Price.

Inference: Accepting to reject null hypothesis, the variable - review_scores_rating is a significant feature to predict the target column - Price

27: Independent Variable = review_scores_accuracy : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference: The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between review_scores_accuracy and Price.

HA: There is a relationship between review_scores_accuracy and Price.

Inference: Accepting to reject null hypothesis, the variable - review_scores_accuracy is a significant feature to predict the target column – Price

28: Independent Variable = review_scores_cleanliness : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference: The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between review_scores_cleanliness and Price.

HA: There is a relationship between review_scores_cleanliness and Price.

Inference: Accepting to reject null hypothesis, the variable - review_scores_cleanliness is a significant feature to predict the target column - Price

29: Independent Variable = review_scores_checkin : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference: The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between review_scores_checkin and Price.

HA: There is a relationship between review_scores_checkin and Price.

Inference: Accepting to reject null hypothesis, the variable - review_scores_checkin is a significant feature to predict the target column – Price

30: Independent Variable = review_scores_communication : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference : The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between review_scores_communication and Price.

HA: There is a relationship between review_scores_communication and Price.

Inference: Accepting to reject null hypothesis, the variable - review_scores_communication is a significant feature to predict the target column - Price

31: Independent Variable = review_scores_location : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference : The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between review_scores_location and Price.

HA: There is a relationship between review_scores_location and Price.

Inference : Accepting to reject null hypothesis, the variable - review_scores_location is a significant feature to predict the target column - Price

32: Independent Variable = review_scores_value : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference: The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between review_scores_value and Price.

HA: There is a relationship between review_scores_value and Price.

Inference : Accepting to reject null hypothesis, the variable - review_scores_value is a significant feature to predict the target column - Price

33: Independent Variable = calculated_host_listings_count : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference: The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between calculated_host_listings_count and Price.

HA: There is a relationship between calculated_host_listings_count and Price.

Inference: Accept null hypothesis, the variable - calculated_host_listings_count is not a significant feature to predict the target column – Price

34: Independent Variable = calculated_host_listings_count_entire_homes : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference: The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between calculated_host_listings_count_entire_homes a nd Price.

HA: There is a relationship between calculated_host_listings_count_entire_homes a nd Price.

Inference : Accepting to reject null hypothesis, the variable - calculated_host_listings_count_ entire_homes is a signficant feature to predict the target column - Price

35: Independent Variable = calculated_host_listings_count_private_rooms : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference: The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between calculated_host_listings_count_private_rooms and Price.

HA: There is a relationship between calculated_host_listings_count_private_rooms and Price.

Inference : Accepting to reject null hypothesis, the variable - calculated_host_listings_count_ private_rooms is a signficant feature to predict the target column - Price

36: Independent Variable = calculated_host_listings_count_shared_rooms : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference : The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between calculated_host_listings_count_shared_rooms and Price.

HA: There is a relationship between calculated_host_listings_count_shared_rooms and Price.

Inference : Accepting to reject null hypothesis, the variable - calculated_host_listings_count_shared_rooms is a signficant feature to predict the target column - Price

37: Independent Variable = reviews_per_month : Target Variable = Price

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference : The data is not normally distributed, hence using non parametric test - SpearMan Correlation.

Hypothesis for SpearMan Correlation test

H0: There is no relationship between reviews_per_month and Price.

HA: There is a relationship between reviews_per_month and Price.

Inference: Accepting to reject null hypothesis, the variable - reviews_per_month is a significant feature to predict the target column – Price

Summary of Correlation test:

	Variable_a	Variable_b	Test_stats	P_value	Conclusion
0	host_response_rate	Price	-0.00	0.75	Variable are independent
1	host_acceptance_rate	Price	0.01	0.35	Variable are independent
2	host_total_listings_count	Price	-0.02	0.00	Variable are dependent
3	latitude	Price	-0.01	0.01	Variable are dependent
4	longitude	Price	-0.25	0.00	Variable are dependent
5	accommodates	Price	0.67	0.00	Variable are dependent
6	bathrooms_text	Price	0.48	0.00	Variable are dependent
7	bedrooms	Price	0.60	0.00	Variable are dependent
8	beds	Price	0.55	0.00	Variable are dependent
9	amenities	Price	0.24	0.00	Variable are dependent
10	minimum_nights	Price	-0.05	0.00	Variable are dependent
11	maximum_nights	Price	0.01	0.03	Variable are dependent
12	minimum_minimum_nights	Price	-0.05	0.00	Variable are dependent
13	maximum_minimum_nights	Price	-0.03	0.00	Variable are dependent
14	minimum_maximum_nights	Price	-0.02	0.00	Variable are dependent
15	maximum_maximum_nights	Price	-0.01	0.33	Variable are independent
16	minimum_nights_avg_ntm	Price	-0.04	0.00	Variable are dependent
17	maximum_nights_avg_ntm	Price	-0.01	0.02	Variable are dependent
18	availability_30	Price	0.18	0.00	Variable are dependent
19	availability_60	Price	0.17	0.00	Variable are dependent
20	availability_90	Price	0.16	0.00	Variable are dependent
21	availability_365	Price	0.16	0.00	Variable are dependent
22	number_of_reviews	Price	-0.06	0.00	Variable are dependent
23	number_of_reviews_ltm	Price	0.07	0.00	Variable are dependent
24	number_of_reviews_l30d	Price	0.07	0.00	Variable are dependent
25	review_scores_rating	Price	0.13	0.00	Variable are dependent
26	review_scores_accuracy	Price	0.06	0.00	Variable are dependent
27	review_scores_cleanliness	Price	0.09	0.00	Variable are dependent
28	review_scores_checkin	Price	0.04	0.00	Variable are dependent
29	review_scores_communication	Price	0.06	0.00	Variable are dependent
30	review_scores_location	Price	0.16	0.00	Variable are dependent
31	review_scores_value	Price	-0.07	0.00	Variable are dependent
32	calculated_host_listings_count	Price	0.01	0.34	Variable are independent
33	calculated_host_listings_count_entire_homes	Price	0.43	0.00	Variable are dependent
34	calculated_host_listings_count_private_rooms	Price	-0.50	0.00	Variable are dependent
35	calculated_host_listings_count_shared_rooms	Price	-0.22	0.00	Variable are dependent
36	reviews_per_month	Price	0.15	0.00	Variable are dependent

b. CATEGORICAL VS NUMERICAL:

Hypothesis for Normality

H0: The data is normally distributed

HA: The data is not normally distributed

Inference : The price data is not normally distributed, hence using non-parametric test like Mann-Whitney and Kruskal-Wallis test for checking dependency.

1: Independent Variable = host_responce_time : Target Variable = Price

Hypothesis for Kruskal-Wallis test

H0: The host_response_time and price are independent.

HA: The host_response_time and price are dependent.

Inference: Accepting to reject null hypothesis, the variable - host_response_time is a significant feature to predict the target column – Price

2: Independent Variable = host_is_superhost : Target Variable = Price

Hypothesis for Mann-Whitney test

H0: The host_ is_superhost and price are independent.

HA: The host_ is_superhost and price are dependent.

Inference: Accepting to reject null hypothesis, the variable - host_ is_superhost is a significant feature to predict the target column – Price

3: Independent Variable = neighbourhood_group_cleansed : Target Variable = Price

Hypothesis for Kruskal-Wallis test

H0: The neighbourhood_group_cleansed and price are independent.

HA: The neighbourhood_group_cleansed and price are dependent.

Inference : Accepting to reject null hypothesis, the variable - neighbourhood_group_cleansed is a significant feature to predict the target column – Price

4: Independent Variable = room_type : Target Variable = Price

Hypothesis for Kruskal-Wallis test

H0: The room_type and price are independent.

HA: The room_type and price are dependent.

Inference: Accepting to reject null hypothesis, the variable - room_type is a signficant feature to predict the target column – Price

5: Independent Variable = has_availability : Target Variable = Price

Hypothesis for Mann-Whitney test

H0: The has_availability and price are independent.

HA: The has_availability and price are dependent.

Inference : Accepting to reject null hypothesis, the variable - has_availability is a significant feature to predict the target column – Price

6: Independent Variable = instant_bookable : Target Variable = Price

Hypothesis for Mann-Whitney test

H0: The instant_bookable and price are independent.

HA: The instant_bookable and price are dependent.

Inference : Accepting to reject null hypothesis, the variable - instant_bookable is a signficant feature to predict the target column – Price

Summary for Categorical Statistical analysis:

	Variable_a	Variable_b	Test_stats	P_value	Conclusion
0	host_response_time	Price	8.675000e+01	0.0	Variable are dependent
1	host_is_superhost	Price	1.111199e+08	0.0	Variable are dependent
2	neighbourhood_group_cleansed	Price	5.220000e+01	0.0	Variable are dependent
3	room_type	Price	1.143977e+04	0.0	Variable are dependent
4	has_availability	Price	1.036526e+07	0.0	Variable are dependent
5	instant_bookable	Price	1.236434e+08	0.0	Variable are dependent

Encoding:

We have 38 numeric features and 6 categorical features in the table. As these 6 categorical variables do not have any ordinal data type and it is nominal in nature. We are concluding to use the one hot encoding imputation with criteria of dropping of first column from each variable. Post the imputation, we have 49 features among these 11 features are from categorical imputation.

	host_response_time_within a day	host_response_time_within a few hours	host_response_time_within an hour	host_is_superhost_t	neighbourhood_group_cleansed_Other Cities	neighbourhoo
0	0	0	1	0	1	
1	0	1	0	1	0	
2	0	0	1	1	1	
3	0	0	1	0	1	
4	0	0	1	0	0	
...	...	...	...	...	...	
33324	0	0	1	0	0	
33325	0	0	1	0	0	
33326	0	0	1	0	0	
33327	0	0	1	0	0	
33328	0	0	1	0	0	

33329 rows × 49 columns

BASE MODEL

Linear Regression:

We have selected Linear Regression as our base model. The purpose of selecting this model is very useful to understand about each features contribution from the data to predict the target variable and also that it is suitable to check whether the data is satisfying all the assumptions of linear regression.

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

```
X=df_final.drop('price',axis=1)
y=df_final.price


Xtrain,Xtest,ytrain,ytest=train_test_split(X,y,test_size=0.2,random_state=10)
```

```
linreg=LinearRegression()
linreg.fit(Xtrain,ytrain)

# Generating the predicted values
y_pred=linreg.predict(Xtest)
```

```
# Evaluating the base model

print('The train R2 score: ',linreg.score(Xtrain,ytrain),'\n')
print('The test R2 score: ',linreg.score(Xtest,ytest),'\n')
print('The RMSE score: ',np.round((np.sqrt(mean_squared_error(ytest,y_pred))),2))
```

```
The train R2 score:  0.21516973708895237

The test R2 score:  0.1549887727812731

The RMSE score:  706.58
```

Inference: By the only activities of imputing null values, encoding the categorical variables. With the base model linear regression, we were able to achieve only the train R_square of 0.21 and test R_square of 0.15. Thus, states that created base model is underfit model which has high bias error.

ASSUMPTIONS OF LINEAR REGRESSION:

1. Mutlicollinearity

Multicollinearity is the occurrence of high intercorrelations among two or more independent variables in multiple regression model. When two or more independent variables are representing the same information, thus leads to inappropriate information in prediction of the target variable. From the OLS model, we can check whether that the presence of multicollinearity from the parameter called 'Condition Number'.

In real time scenario, we cannot expect all the time that model supposed to have 100% multicollinearity free features with the condition number less than 100. For an ideal situation, we can have the condition number between 100 to 1000 and still no treatment is required. If the condition no is greater than 1000, then we can conclude that there is strong relationship between the independent features and it is to be treated.

Variance inflation factor (VIF) is a measure of the amount of multicollinearity in a set of multiple regression variables.

1. When the VIF is <= 1, we can conclude that no multicollinearity between the independent features.
2. VIF between >1 to <=5, we can conclude that moderate multicollinearity between the independent features.
3. VIF >5, we can conclude that strong multicollinearity presents between the independent features.

Multi collinearity model check:

From the base model, we have got the condition no of 4.72e+10 and it is greater than condition no of 1000. Thus, states clearly that the independent features from the model have very strong relationship among them.

Omnibus:	56238.741	**Durbin-Watson:**	2.000
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	343034629.703
Skew:	18.268	**Prob(JB):**	0.00
Kurtosis:	557.472	**Cond. No.**	4.72e+10

We have dropped the individual independent features one by one and reiterated the VIF values for them. We have only 23 features that that has VIF values less than or equal to 5 from 49 features.

Performed another regression model with these 23 features where VIF values are less than 5. However, we could see that the accuracy score of the base model has been improved from 0.21% to 0.25% but the condition No has not reduced to less than 1000 and it still ranges around 1.44e+04.

Dep. Variable:	price	**R-squared:**	0.251
Model:	OLS	**Adj. R-squared:**	0.250
Method:	Least Squares	**F-statistic:**	387.5
Date:	Mon, 21 Mar 2022	**Prob (F-statistic):**	0.00
Time:	13:07:53	**Log-Likelihood:**	-2.0912e+05
No. Observations:	26663	**AIC:**	4.183e+05
Df Residuals:	26639	**BIC:**	4.185e+05
Df Model:	23		
Covariance Type:	nonrobust		

Omnibus:	55578.942	**Durbin-Watson:**	2.007
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	316818618.068
Skew:	17.782	**Prob(JB):**	0.00
Kurtosis:	535.833	**Cond. No.**	1.44e+04

2.Linearity:

The linearity check is one of the main assumptions that talks about independent variables supposed to have linear relationship with target variable in order to predict the target variable for the better accurate prediction.

We can check the linearity assumption by using the rainbow test.

Rainbow test assumption of hypothesis:

- H0: It is good linear model
- HA: It is not good linear model

Test Statistics and P_value:

```
1 sm.stats.linear_rainbow(MLR_Mc)
```

```
(1.0463214878684113, 0.004491001525071741)
```

Conclusion:

From the test statistics, we have got the pvalue as 0.004 and it is lesser than alpha value of 0.05. Hence, we are agreeing to reject the null hypothesis and concluding that the independent variables are not having linear relationship with the target variable.

3.Autocorrelation

Independence of observation should exist (i.e.,) when the residuals are correlated within themselves then the assumption is violated.

The following assumption can be evaluated by using 'Durbin-Watson-Test'.

The value from the test can be denoted as 'd'.

1. $0 < d < 2$ - Positive auto-correlation,
2. $d = 2$ - No auto correlation,
3. $2 < d < 4$ - Negative correlation.

Auto-correlation check

Omnibus:	55578.942	**Durbin-Watson:**	2.007
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	316818618.068
Skew:	17.782	**Prob(JB):**	0.00
Kurtosis:	535.833	**Cond. No.**	1.44e+04

Conclusion:

From the OLS model, we have derived the score of Durbin-Waston as 2.007. Hence, we can conclude that there is no auto-correlation between the residuals.

4. Homoscedastic

Homoscedastic refers to a condition in which the variance of the residual, or error term, in a regression model is constant. If the variance of the error term is homoscedastic, the model was well-defined. If there is too much variance, the model may not be defined well and then it is called as heteroscedastic.

We can check the homoscedastic assumption by using the goldfeld test.

Homoscedastic check:

Hypothesis for goldfeld test:

- H0: Heteroscedasticity is not present
- HA: Heteroscedasticity is present.

Test Statistics and P value:

```
1 sm.stats.het_goldfeldquandt(y = MLR_Mc.resid, x = X_train)
```

```
(0.7986980658518356, 0.9999999999999999, 'increasing')
```

Conclusion:

From the test statistics, we have got the pvalue as 0.99 and it is greater than alpha value of 0.05. Hence, we are agreeing to accept the null hypothesis and concluding that the residuals have same variance.

5. Normality of the residue:

Normality is the assumption that the underlying residuals or error terms are normally distributed, or approximately.

Normality of the residue check:

We can check the normality of the residue assumption by using the shapiro test.

Hypothesis for Normality

H0: Residue is normal
HA: Residue is not normal

Test Statistics and P_value:

```
1 ss.shapiro(MLR_Mc.resid)
```

```
ShapiroResult(statistic=0.28699231147766113, pvalue=0.0)
```

Conclusion:

From the test statistics, we have got the pvalue as 0.00 and it is lesser than alpha value of 0.05. Hence, we are agreeing to reject the null hypothesis and concluding that the residuals of the model do not form normal distribution.

ASSUMPTIONS INFERENCE:

From the above statistical approach, we could observe that non-violated and violated assumptions are enlisted below.

Non-Violated Assumption:

1. Homoscedasticity,
2. Autocorrelation.

Violated Assumption:

1. Multicollinearity,
2. Linearity,
3. Normality check of the residuals.

As, three assumption checks are violating and fitting the above data in the linear model is strongly not recommendable. Hence, we can go-ahead and start using the non-linear models for the prediction.

Non-Linear Model:

1. Decision Tree:

```
dt=DecisionTreeRegressor(random_state=10)
dt.fit(Xtrain,ytrain)
dt_pred=dt.predict(Xtest)
```

```
print("Decision tree train r2",dt.score(Xtrain,ytrain),'\n')
print("Decision tree test r2",dt.score(Xtest,ytest),'\n')
```

```
Decision tree train r2 0.9999997766577781

Decision tree test r2 0.380324441658629
```

While building a Decision Tree Model using sklearn we have come across the above R2 value of the Decision Tree Model.

Inferences for Decision Tree Regressor Model:

• The train R2 and test R2 shows that the model is overfitting

• The Root_mean_square_error of the model came around 614.32.

By doing the R2 score analysis for train and test of the decision tree model. The R2 of train is coming around to be 99.99 percent and the R2 of test is around 38.03 percent. So it states that the model is over fitting. We can overcome the problem of over fitting by doing decision tree pruning or hyper parameter tuning with a given set of parameters using Grid Search CV that will help us getting the best parameters for our model building.

2. Random Forest:

```
rfc=RandomForestRegressor(random_state=10)
rfc.fit(Xtrain,ytrain)
rfc_pred=rfc.predict(Xtest)
```

```
print("RandomForest train r2",rfc.score(Xtrain,ytrain),'\n')
print("RandomForest test r2",rfc.score(Xtest,ytest),'\n')
```

```
RandomForest train r2 0.9420140513276459

RandomForest test r2 0.6203929779536403
```

While building a Random forest using sklearn we have come across the above R2 value of the Random forest Model.

Inferences for Random forest Regressor Model:

• The train R2 and test R2 shows that the model is overfitting

• The Root_mean_square_error of the model came around 480.82.

By doing the R2 score analysis for train and test of the decision tree model. The R2 of train is coming around to be 94.20 percent and the R2 of test is around 62.03 percent. So it states that the model is over fitting. We can overcome the problem of over fitting by doing decision tree pruning or hyper parameter tuning with a given set of parameters using Grid Search CV that will help us getting the best parameters for our model building.

3. Ada-Boosting Regressor:

```
ada=AdaBoostRegressor(random_state=10)
ada.fit(Xtrain,ytrain)
ada_pred=ada.predict(Xtest)
```

```
print("Ada-Boost train r2",ada.score(Xtrain,ytrain),'\n')
print("Ada-Boost test r2",ada.score(Xtest,ytest),'\n')
```

```
Ada-Boost train r2 -13.028590800069948

Ada-Boost test r2 -10.909824283849344
```

While building an Ada-Boosting using sklearn we have come across the above R2 value of the Ada-Boosting Model.

Inferences for Ada-Boosting Regressor Model:

• The train R2 and test R2 shows that the model is underfitting

• The Root_mean_square_error of the model came around 2693.18.

By doing the R2 score analysis for train and test of the decision tree model. The R2 of train is coming around to be -13.02 percent and the R2 of test is around -10.90 percent. So it states that the model is under fitting. We can overcome the problem of over fitting by doing decision tree pruning or hyper parameter tuning with a given set of parameters using Grid Search CV that will help us getting the best parameters for our model building.

4. Gradient Boosting Regressor:

```
gb=GradientBoostingRegressor(random_state=10)
gb.fit(Xtrain,ytrain)
gb_pred=gb.predict(Xtest)
```

```
print("Gradient boost train r2",gb.score(Xtrain,ytrain),'\n')
print("Gradient boost test r2",gb.score(Xtest,ytest),'\n')
```

```
Gradient boost train r2 0.5927189097968777

Gradient boost test r2 0.4293242211498578
```

While building a Gradient Boosting using sklearn we have come across the above R2 value of the Gradient Boosting Model.

Inferences for Gradient Boosting Regressor Model:

• The train R2 and test R2 shows that the model is underfitting

• The Root_mean_square_error of the model came around 589.53.

By doing the R2 score analysis for train and test of the decision tree model. The R2 of train is coming around to be 59.27 percent and the R2 of test is around 42.93 percent. So it states that the model is under fitting. We can overcome the problem of over fitting by doing decision tree pruning or hyper parameter tuning with a given set of parameters using Grid Search CV that will help us getting the best parameters for our model building.

5. XG Boosting Regressor:

```
xg=XGBRegressor(random_state=10)
xg.fit(Xtrain,ytrain)
xg_pred=xg.predict(Xtest)
```

```
print("XG boost train r2",xg.score(Xtrain,ytrain),'\n')
print("XG boost test r2",xg.score(Xtest,ytest),'\n')
```

```
XG boost train r2 0.9556800882027592

XG boost test r2 0.6285701354087327
```

While building a XG Boosting using xgboost library we have come across the above R2 value of the XG Boosting Model.

Inferences for XG Boosting Regressor Model:

• The train R2 and test R2 shows that the model is overfitting

• The Root_mean_square_error of the model came around 475.61.

By doing the R2 score analysis for train and test of the decision tree model. The R2 of train is coming around to be 95.56 percent and the R2 of test is around 62.85 percent. So it states that the model is under fitting. We can overcome the problem of over fitting by doing decision tree pruning or hyper parameter tuning with a given set of parameters using Grid Search CV that will help us getting the best parameters for our model building.

Inference from the non-linear model:

	Model	Train_R2	Test_R2	RMSE
0	Decision_tree	1.00	0.38	614.32
1	Random_forest	0.94	0.62	480.82
2	Ada Boosting	-13.03	-10.91	2693.18
3	Gradient_Boosting	0.59	0.43	589.53
4	XG_Boosting	0.96	0.63	475.61

From the five non-linear model performed we can come to the conclusion that all the model are under performing and is influenced by various features of the data. We can improve the perform of these non-linear model by doing the following:

- Treating the multi-collinearity in the data.
- Treating the outliers in the data.

Treating the multi- collinearity:

```
# We can try to reduce the multicolinearity present in the data by keeping variable with VIF<11.

X = df_final.drop(columns = ['price','maximum_nights_avg_ntm','longitude','calculated_host_listings_count',
                             'review_scores_accuracy','review_scores_checkin','minimum_nights_avg_ntm','review_scores_communic
                            'review_scores_location','review_scores_value','host_response_rate','latitude','review_scores_clea
                            'availability_60','review_scores_rating','has_availability_t','host_acceptance_rate','availability
                            ])
y = df_final.price

#VIF
vif = [VIF(X.values, i) for i in range(X.shape[1])]
vif_df = pd.DataFrame(vif, index = X.columns, columns = ['VIF_index']).sort_values(by = ['VIF_index'], ascending=False)
vif_df
```

	VIF_index
accommodates	10.63
beds	8.07
minimum_maximum_nights	8.03
maximum_maximum_nights	8.00
host_response_time_within an hour	7.27
amenities	6.68
minimum_nights	6.41
maximum_minimum_nights	5.87
bathrooms_text	5.79
host_total_listings_count	5.78
minimum_minimum_nights	5.24
number_of_reviews_ltm	3.94
availability_365	3.79
calculated_host_listings_count_entire_homes	3.03
number_of_reviews_l30d	2.92
availability_30	2.75
maximum_nights	2.73
host_response_time_within a few hours	2.71
number_of_reviews	2.10
room_type_Shared room	1.87
room_type_Private room	1.87
host_is_superhost_t	1.84
neighbourhood_group_cleansed_Other Cities	1.83
calculated_host_listings_count_shared_rooms	1.81
host_response_time_within a day	1.79
instant_bookable_t	1.75
reviews_per_month	1.52
neighbourhood_group_cleansed_Unincorporated Areas	1.22
calculated_host_listings_count_private_rooms	1.21
room_type_Hotel room	1.03

We can reduce the multi-collinearity using the VIF, removing the variable with VIF threshold less than 11.

We are left with 30 attributes as shown by the above table.

Treating the outliers in the model:

- **Dropping the outlier:**

```
# We have outlier in 28 out of the 30 columns remaining from VIF.
```

```
# Dropping the data with outliers.

df_copy=df_final.copy(deep=True) # Creating a copy

Q1 = df_copy.quantile(0.25)
Q3 = df_copy.quantile(0.75)
IQR = Q3 - Q1
df_copy = df_copy[~((df_copy < (Q1 - 1.5 * IQR)) | (df_copy > (Q3 + 1.5 * IQR))).any(axis=1)]
```

```
df_copy.shape
```

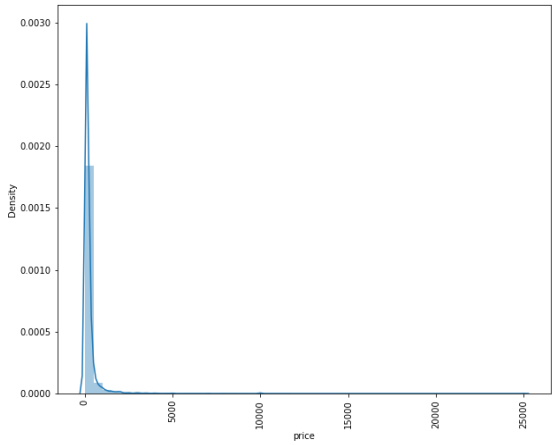
```
(3541, 49)
```

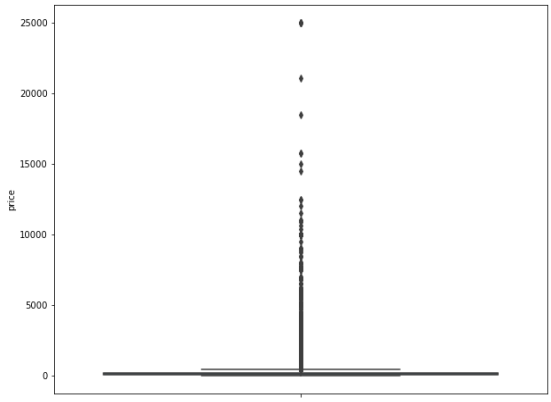
Inference: As we can see by dropping the rows with outliers we are left with only 3541 rows out of the 33000 rows, which is approx. 90% of the data. Hence we have to find a better way to treat the outliers.

- **Transforming the target:**

Also we can have outliers is 28 out of the 30 variables left hence transforming all the variable is not a good practice. Therefore we can try and transform only the target variable and run the non-linear model again.

Price distribution before transformation:

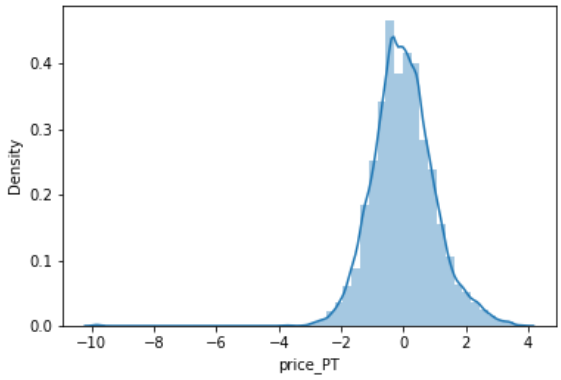


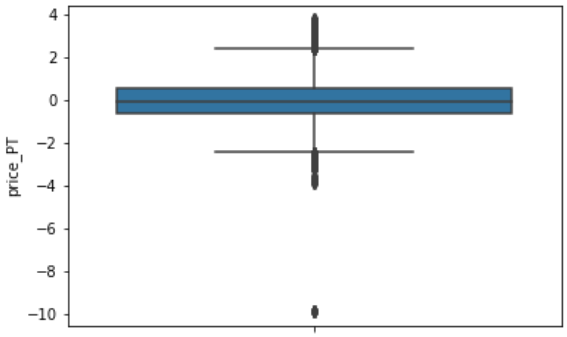



```
pt=PowerTransformer() # Instantiation

pt.fit(df_final[['price']])
df_final['price_PT']= pt.transform(df_final[['price']])
```

```
# Declaring X and y

X = df_final.drop(columns = ['price','price_PT','maximum_nights_avg_ntm','longitude','calculated_host_listings_count',
                             'review_scores_accuracy','review_scores_checkin','minimum_nights_avg_ntm','review_scores_communic
                            'review_scores_location','review_scores_value','host_response_rate','latitude','review_scores_clea
                            'availability_60','review_scores_rating','has_availability_t','host_acceptance_rate','availability
                            ])
y=df_final['price_PT']
```

```
# Train test split

Xtrain,Xtest,ytrain_tran,ytest_tran=train_test_split(X,y,test_size=0.2,random_state=10)
```





Non-Linear Model after treating multi-collinearity and outliers:

1. Decision Tree:

```
dt=DecisionTreeRegressor(random_state=10)
dt.fit(Xtrain,ytrain_tran)
dt_pred=dt.predict(Xtest)
```

```
print("train r2",dt.score(Xtrain,ytrain_tran),'\n')
print("test r2",dt.score(Xtest,ytest_tran))
```

```
train r2 0.9980812332343478

test r2 0.45186114687268997
```

While building a Decision Tree Model using sklearn we have come across the above R2 value of the Decision Tree Model.

Inferences for Decision Tree Regressor Model:

• The train R2 and test R2 shows that the model is overfitting

• The Root_mean_square_error of the model came around 0.73.

By doing the R2 score analysis for train and test of the decision tree model. The R2 of train is coming around to be 99.80 percent and the R2 of test is around 45.18 percent. So it states that the model is over fitting. We can overcome the problem of over fitting by doing decision tree pruning or hyper parameter tuning with a given set of parameters using Grid Search CV that will help us getting the best parameters for our model building.

2. Random Forest:

```
rfc=RandomForestRegressor(random_state=10)
rfc.fit(Xtrain,ytrain_tran)
rfc_pred=rfc.predict(Xtest)
```

```
print("RandomForest train r2",rfc.score(Xtrain,ytrain_tran),'\n')
print("RandomForest test r2",rfc.score(Xtest,ytest_tran))
```

```
RandomForest train r2 0.9608796469779705

RandomForest test r2 0.7297571926233111
```

While building a Random forest using sklearn we have come across the above R2 value of the Random forest Model.

Inferences for Random forest Regressor Model:

• The train R2 and test R2 shows that the model is overfitting

• The Root_mean_square_error of the model came around 0.51.

By doing the R2 score analysis for train and test of the Random forest model. The R2 of train is coming around to be 96.08 percent and the R2 of test is around 72.97 percent. So it states that the model is over fitting. We can overcome the problem of over fitting by doing random forest tree pruning or hyper parameter tuning with a given set of parameters using Grid Search CV that will help us getting the best parameters for our model building.

3. Ada-Boosting Regressor:

```
ada=AdaBoostRegressor(random_state=10)
ada.fit(Xtrain,ytrain_tran)
ada_pred=ada.predict(Xtest)
```

```
print("Ada Boost train r2",ada.score(Xtrain,ytrain_tran),'\n')
print("Ada Boost test r2",ada.score(Xtest,ytest_tran))
```

```
Ada Boost train r2 0.5233835981994082

Ada Boost test r2 0.5298408832134248
```

While building an Ada-Boosting using sklearn we have come across the above R2 value of the Ada-Boosting Model.

Inferences for Ada-Boosting Regressor Model:

• The train R2 and test R2 shows that the model is underfitting

• The Root_mean_square_error of the model came around 0.68.

By doing the R2 score analysis for train and test of the Ada-Boosting model. The R2 of train is coming around to be 52.33 percent and the R2 of test is around 52.98 percent. So it states that the model is under fitting. We can overcome the problem of under fitting by hyper parameter tuning with a given set of parameters using Grid Search CV that will help us getting the best parameters for our model building.

4. Gradient Boosting Regressor:

```
gb=GradientBoostingRegressor(random_state=10)
gb.fit(Xtrain,ytrain_tran)
gb_pred=gb.predict(Xtest)
```

```
print("Gradient Boost train r2",gb.score(Xtrain,ytrain_tran),'\n')
print("Gradient boost test r2",gb.score(Xtest,ytest_tran))
```

```
Gradient Boost train r2 0.7080542922866698

Gradient boost test r2 0.6879204128099166
```

While building a Gradient Boosting using sklearn we have come across the above R2 value of the Gradient Boosting Model.

Inferences for Gradient Boosting Regressor Model:

• The train R2 and test R2 shows that the model is slightly underfitting.

• The Root_mean_square_error of the model came around 0.55.

By doing the R2 score analysis for train and test of the Gradient Boosting model. The R2 of train is coming around to be 70.80 percent and the R2 of test is around 68.79 percent. So it states that the model is slightly under fitting. We can overcome the problem of under fitting by hyper parameter tuning with a given set of parameters using Grid Search CV that will help us getting the best parameters for our model building.

5. XG Boosting Regressor:

```
xg=XGBRegressor(random_state=10)
xg.fit(Xtrain,ytrain_tran)
xg_pred=xg.predict(Xtest)
```

```
print("XG Boost train r2",xg.score(Xtrain,ytrain_tran),'\n')
print("XG Boost test r2",xg.score(Xtest,ytest_tran))
```

```
XG Boost train r2 0.835816524785392

XG Boost test r2 0.7389928147399825
```

While building a XG Boosting using xgboost library we have come across the above R2 value of the XG Boosting Model.

Inferences for XG Boosting Regressor Model:

• The train R2 and test R2 shows that the model is performing well.

• The Root_mean_square_error of the model came around 0.51.

By doing the R2 score analysis for train and test of the XG Boosting model. The R2 of train is coming around to be 83.58 percent and the R2 of test is around 73.89 percent. So we can see that the model performance is good and we can improve the model by performing hyper parameter tuning with a given set of parameters using Grid Search CV that will help us getting the best parameters for our model building.

6. Light Gradient Boosting Regressor:

```
lit=LGBMRegressor(random_state=10)
lit.fit(Xtrain,ytrain_tran)
lit_pred=lit.predict(Xtest)
```

```
print("LGBM Boost train r2",lit.score(Xtrain,ytrain_tran),'\n')
print("LGBM Boost test r2",lit.score(Xtest,ytest_tran))
```

```
LGBM Boost train r2 0.7679900985446397

LGBM Boost test r2 0.7310667115509359
```

While building a Light Gradient Boosting using lgmboost we have come across the above R2 value of the Light Gradient Boosting Model.

Inferences for Light Gradient Boosting Regressor Model:

• The train R2 and test R2 shows that the model is performing well.

• The Root_mean_square_error of the model came around 0.51.

By doing the R2 score analysis for train and test of the Light Gradient Boosting model. The R2 of train is coming around to be 76.79 percent and the R2 of test is around 73.10 percent. So we can see that the model performance is good and we can improve the model by performing hyper parameter tuning with a given set of parameters using Grid Search CV that will help us getting the best parameters for our model building.

7. CAT Boosting Regressor:

```
cat=CatBoostRegressor(random_state=10)
cat.fit(Xtrain,ytrain_tran)
cat_pred=cat.predict(Xtest)
```

```
220:    learn: 0.5090524        total: 2.79s    remaining: 9.83s
221:    learn: 0.5088599        total: 2.8s     remaining: 9.81s
222:    learn: 0.5087103        total: 2.81s    remaining: 9.79s
223:    learn: 0.5084843        total: 2.82s    remaining: 9.77s
224:    learn: 0.5082927        total: 2.83s    remaining: 9.75s
225:    learn: 0.5080968        total: 2.84s    remaining: 9.73s
226:    learn: 0.5079642        total: 2.85s    remaining: 9.7s
227:    learn: 0.5077815        total: 2.86s    remaining: 9.68s
228:    learn: 0.5075632        total: 2.87s    remaining: 9.67s
229:    learn: 0.5075326        total: 2.88s    remaining: 9.64s
230:    learn: 0.5071685        total: 2.89s    remaining: 9.62s
231:    learn: 0.5069965        total: 2.9s     remaining: 9.6s
232:    learn: 0.5069305        total: 2.91s    remaining: 9.58s
233:    learn: 0.5066741        total: 2.92s    remaining: 9.56s
234:    learn: 0.5063229        total: 2.93s    remaining: 9.54s
235:    learn: 0.5061967        total: 2.94s    remaining: 9.52s
236:    learn: 0.5059516        total: 2.95s    remaining: 9.5s
237:    learn: 0.5057394        total: 2.96s    remaining: 9.48s
238:    learn: 0.5056100        total: 2.97s    remaining: 9.47s
```

```
print("Cat Boost train r2",cat.score(Xtrain,ytrain_tran),'\n')
print("Cat Boost test r2",cat.score(Xtest,ytest_tran))
```

```
Cat Boost train r2 0.8075481650999742

Cat Boost test r2 0.7508177297753744
```

While building a CAT Boosting using catboost library we have come across the above R2 value of the CAT Boosting Model.

Inferences for CAT Boosting Regressor Model:

• The train R2 and test R2 shows that the model is performing well.

• The Root_mean_square_error of the model came around 0.49.

By doing the R2 score analysis for train and test of the XG Boosting model. The R2 of train is coming around to be 80.75 percent and the R2 of test is around 75.08 percent. So we can see that the model performance is good and we can improve the model by performing hyper parameter tuning with a given set of parameters using Grid Search CV that will help us getting the best parameters for our model building.

Inference from the non-linear model after treating outlier and multi-colinearity:

	Model	Train_R2	Test_R2	RMSE
0	Decision_tree	1.00	0.45	0.73
1	Random_forest	0.96	0.73	0.51
2	Ada Boosting	0.52	0.53	0.68
3	Gradient_Boosting	0.71	0.69	0.55
4	XG_Boosting	0.84	0.74	0.51
5	LGBM_Boosting	0.77	0.73	0.51
6	CAT_Boosting	0.81	0.75	0.49

From the above evaluation table we can see that the performance for all the non-linear model after treating the multi-colinearity and the transforming the target has significantly improved. Among the six non-linear model we can see the Random forest, XG Boost and the CAT Boost model perform the best.
These base model performance can be further improved by fine tuning the hyper parameters of these algorithms using the GridSearchCV.

Hyperparameter Tunning:

Random forest, XG Boost and the CAT Boost model have been chosen and a set of parameters has been considered for fine tuning.
Using these set of parameters we have used Grid Search Cross Validation technique for Hyper parameter tuning the models where the Cross Validation method considered is 5-Fold Cross Validation.

1. **Random Forest regressor:**

```
rfs=RandomForestRegressor(random_state=10)
param={'n_estimators':np.arange(100,400,50),'max_depth':np.arange(2,11)}
rfc_tune=GridSearchCV(estimator=rfs,param_grid=param,cv=5,scoring='neg_mean_squared_error',n_jobs=-1)
```

```
rfc_tune.fit(Xtrain,ytrain_tran)
rfc_tune.best_params_
```

```
{'max_depth': 10, 'n_estimators': 350}
```

Random Forest Regressor : {'n_estimator': 350, 'max_depth': 10}

Refitting the Tunned Based models to compare the performances

```
rfc_best=RandomForestRegressor(random_state=10,n_estimators=350,max_depth=10)
rfc_best.fit(Xtrain,ytrain_tran)
rfc_pred=rfc_best.predict(Xtest)
print("Random Forest tuned train r2",rfc_best.score(Xtrain,ytrain_tran))
print("Random Forest tuned test r2",rfc_best.score(Xtest,ytest_tran))
```

```
Random Forest tuned train r2 0.752163473628543
Random Forest tuned test r2 0.7050772089035544
```

After refitting the tunned base model of Random forest regressor we carried out the R2 score analysis for the tunned Random Forest model.

Inference:

- The R2 for the train data came around 75.21 percent
- The R2 for the train data came around 70.5 percent
- The Root_mean_square for the model : 0.54
- The Mean_absolute_error for the model: 0.39
- The Mean_absolute_percentage_error for the model: 143.94

From the above R2 score analysis we can see that the overfitting present the based model was mitigated. The overall performs of the model looks good.

2. **XG Boost Regressor:**

```
xg=XGBRegressor(random_state=10)
param={'n_estimators':np.arange(80,200,10),'learning_rate':[0.15,0.16,0.17,0.18,0.19],'gamma':(0.5,0.6,0.7,0.8,0.9),
       'max_depth':np.arange(2,10)}
xg_tune=GridSearchCV(estimator=xg,param_grid=param,cv=5,scoring='neg_mean_squared_error',n_jobs=-1)
```

```
xg_tune.fit(Xtrain,ytrain_tran)
xg_tune.best_params_
```

```
{'gamma': 0.5, 'learning_rate': 0.15, 'max_depth': 7, 'n_estimators': 160}
```

XG Boost Regressor : {'n_estimator': 160, 'max_depth': 7, ‘learning_rate’: 0.15, ‘gamma’:0.5}

```
xg_best=XGBRegressor(random_state=10,n_estimators=160,learning_rate=0.15,gamma=0.5,max_depth=7)
xg_best.fit(Xtrain,ytrain_tran)
xg_pred=xg_best.predict(Xtest)
print("XG Boost tuned train r2",xg_best.score(Xtrain,ytrain_tran))
print("XG Boost tuned test r2",xg_best.score(Xtest,ytest_tran))
```

```
XG Boost tuned train r2 0.836356824995569
XG Boost tuned test r2 0.7357165016686051
```

After refitting the tunned base model of XG Boost regressor we carried out the R2 score analysis for the tunned Random Forest model.

Inference:

- The R2 for the train data came around 83.63 percent
- The R2 for the train data came around 73.57 percent
- The Root_mean_square for the model : 0.51
- The Mean_absolute_error for the model: 0.36
- The Mean_absolute_percentage_error for the model: 138.22

From the above R2 score analysis we can see that the overall performance wrt base model has not improved any further. The overall performs of the tuned XG Boost regeressor model looks good.

3. CAT Boost Regeressor:

```
cat=CatBoostRegressor(random_state=10)
param={'iterations':np.arange(300,1000,50),'learning_rate':[0.1,0.2,0.3,0.4,0.5],'depth':np.arange(4,11),'l2_leaf_reg':[0,1,
cat_tune=GridSearchCV(estimator=cat,param_grid=param,cv=5,scoring='neg_mean_squared_error',n_jobs=-1)
```

```
cat_tune.fit(Xtrain,ytrain_tran)
cat_tune.best_params_
```

```
{'depth': 8, 'iterations': 900, 'l2_leaf_reg': 2, 'learning_rate': 0.1}
```

CAT Boost Regressor : {'iterations': 900, 'depth': 8, 'learning_rate': 0.1, 'l2_leaf_reg': 2}

```
cat_best=CatBoostRegressor(depth= 8, iterations= 900, learning_rate= 0.1,l2_leaf_reg=2,random_state=10)
cat_best.fit(Xtrain,ytrain_tran)
cat_pred=cat_best.predict(Xtest)
```

```
print("CAT Boost tuned train r2: ",cat_best.score(Xtrain,ytrain_tran))
print("CAT Boost tuned test r2: ",cat_best.score(Xtest,ytest_tran))
```

```
CAT Boost tuned train r2:  0.8734667315367685
CAT Boost tuned test r2:  0.7566642881613608
```

After refitting the tunned base model of CAT Boost regressor we carried out the R2 score analysis for the tunned Random Forest model.

Inference:

- The R2 for the train data came around 87.34 percent
- The R2 for the train data came around 75.66 percent
- The Root_mean_square for the model : 0.49
- The Mean_absolute_error for the model: 0.35
- The Mean_absolute_percentage_error for the model: 135.80

From the above R2 score analysis we can see that the overall performance wrt base model has improved considerably. The overall performs of the tuned CAT Boost regeressor model looks great.

Evaluation of the tuned model:

	Model	Train_R2	Test_R2	RMSE	MAE	MAPE
0	Random_forest	0.75	0.71	0.54	0.39	143.94
1	XG_Boosting	0.84	0.74	0.51	0.36	138.22
2	CAT_Boosting	0.87	0.76	0.49	0.35	135.80

From the above evaluation matrix, we can see that the tuned CAT Boost regressor is the best performing model on all the evaluation parameter.

Feature Selection:

Feature Selection is a process of dimensionality reduction by which an initial set of raw data is reduced to more manageable groups for processing. A characteristic of these large data sets is a large number of variables that require a lot of computing resources to process. Feature Selection is the name for methods that select and /or combine variables into features, effectively reducing the amount of data that must be processed, while still accurately and completely describing the original data set. The process of feature selection is useful when you need to reduce the number of resources needed for processing without losing important or relevant information. Feature Selection can also reduce the amount of redundant data for a given analysis. Also, the reduction of the data and the machine's efforts in building variable combinations (features) facilitate the speed of learning and generalization steps in the machine learning process.

Using Forward Feature Selection:

Since the analysis shows that the CAT boost regressor has the best overall performance from all the previously deployed base and tuned models, it will be considered for deriving the final set of features affecting the ML algorithm using the forward selection model.

The Forward selection process was carried out and the best 26 feature that have significant impact on the target was identified.

['host_response_time_within a day','host_response_time_within an hour', 'host_is_superhost_t', 'neighbourhood_group_cleansed_Other Cities', 'neighbourhood_group_cleansed_Unincorporated Areas', 'room_type_Hotel room', 'room_type_Private room', 'room_type_Shared room', 'instant_bookable_t', 'host_total_listings_count', 'accommodates', 'bathrooms_text', 'beds', 'minimum_nights', 'maximum_nights', 'maximum_minimum_nights', 'minimum_maximum_nights', 'maximum_maximum_nights', 'availability_30', 'availability_365', 'number_of_reviews_ltm', 'number_of_reviews_l30d', 'calculated_host_listings_count_entire_homes', 'calculated_host_listings_count_private_rooms', 'calculated_host_listings_count_shared_rooms', 'reviews_per_month']

Refitting the tuned CAT Boost regressor model with the 26 significant variable:

```
print("Forward train r2",cat_for_best.score(Xtrain,ytrain_tran),'\n')
print("Forward test r2",cat_for_best.score(Xtest,ytest_tran))
```

```
Forward train r2 0.8569985633753834

Forward test r2 0.7520275956931872
```

After refitting the model we checked the R2 for the model and the train R2 came around: 85.69 percent and the test R2 came around 75.20 percent.

Using Backward Feature Selection:

Since the analysis shows that the CAT boost regressor has the best overall performance from all the previously deployed base and tuned models, it will be considered for deriving the final set of features affecting the ML algorithm using the backward selection model.

The Backward selection process was carried out and the best 23 feature that have significant impact on the target was identified.

['host_response_time_within a day', 'host_response_time_within an hour', 'neighbourhood_group_cleansed_Other Cities', 'neighbourhood_group_cleansed_Unincorporated Areas', 'room_type_Hotel room', 'room_type_Private room', 'room_type_Shared room', 'instant_bookable_t', 'accommodates', 'bathrooms_text', 'beds', 'amenities', 'maximum_nights', 'maximum_minimum_nights', 'maximum_maximum_nights', 'availability_30', 'availability_365', 'number_of_reviews_ltm', 'number_of_reviews_l30d', 'calculated_host_listings_count_entire_homes', 'calculated_host_listings_count_private_rooms', 'calculated_host_listings_count_shared_rooms', 'reviews_per_month']

Refitting the tuned CAT Boost regressor model with the 23 significant variable:

```
print("Backward train r2",cat_back_best.score(Xtrain,ytrain_tran),'\n')
print("Backward test r2",cat_back_best.score(Xtest,ytest_tran))
```

```
Backward train r2 0.8619621511660819

Backward test r2 0.746035003803753
```

After refitting the model, we checked the R2 for the model and the train R2 came around: 86.19 percent and the test R2 came around 74.60 percent.

The R2 score is better for the model fitted with features obtained from backward selection, also with lesser number variables. Hence, we can conclude that the backward feature selection method produced the better model.

Final Prediction Model (CostPrediction_CATBoostModel):

Source Code:

```
def CostPrediction_CATBoostModel(fin_model):
    X = cost_df [['host_response_time_within a day', 'host_response_time_within an
        hour', 'neighbourhood_group_cleansed_Other Cities',
        'neighbourhood_group_cleansed_Unincorporated Areas', 'room_type_Hotel
        room', 'room_type_Private room', 'room_type_Shared room',
        'instant_bookable_t', 'accommodates', 'bathrooms_text', 'beds', 'amenities',
        'maximum_nights', 'maximum_minimum_nights',
        'maximum_maximum_nights', 'availability_30', 'availability_365',
        'number_of_reviews_ltm', 'number_of_reviews_l30d',
        'calculated_host_listings_count_entire_homes',
```

```
        'calculated_host_listings_count_private_rooms',
        'calculated_host_listings_count_shared_rooms', 'reviews_per_month']]
Y=cost_df[‘price_PT’]
X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size=0.2,
        random_state=10)
Res_model = fin_model.fit(X_train,Y_train)
Y_pred = Res_model.predict(X_test)
print(“Overall R2 for the CATBoost Model for train data : ”,
        np.round(Res_model.score(X_train,Y_train),2))
print(“Overall R2 for the CATBoost Model for test data : ”,
        np.round(Res_model.score(X_test,Y_test),2))
print(“Root mean squared error for the CATBoost Model: ”,
        np.round(mean_squared_error(Y_test,Y_pred,squared=False),2))
print(“Mean absolute error for the CATBoost Model: ”,
        np.round(mean_absolute_error(Y_test,Y_pred),2))
print(“Mean absolute percentage error for the CATBoost Model: ”,
        np.round(mape(Y_test,Y_pred),2))
print('--------------------------------------------------------------------------------','\n')
df_feature = pd.DataFrame(
        {'Features':Xtrain.columns,'Coefficient':fin_model.feature_importances_})

df_feature.sort_values('Coefficient',ascending=False,inplace=True)
print(“The Significant variable and the coefficient are: ”,df_feature)
plt.figure(figsize=(15,8))
sns.barplot(data=df_feature,x='Features',y='Coefficient')
plt.title('Coefficient of Variable')
plt.xticks(rotation=90)
plt.show()
k=KFold(n_splits=5)
score=cross_val_score(estimator=fin_model,X=X,y=Y,cv=k,scoring='r2',n_jobs=-1)
print("Mean Score : ",np.mean(scores))
```

```
 print("Bias error : ",(1-np.mean(scores))*100)
 print("Variance error : ",(np.std(scores)/np.mean(scores))*100,'\n')
 print('--------------------------------------------------------------------------------','\n')
```

```
fin_model= CatBoostRegressor(depth= 8, iterations= 900, learning_rate= 0.1,
        l2_leaf_reg=2, random_state=10)
CostPrediction_CATBoostModel(fin_model)
```

Output:

Overall R2 for the CATBoost Model for train data : 0.86

Overall R2 for the CATBoost Model for test data : 0.75

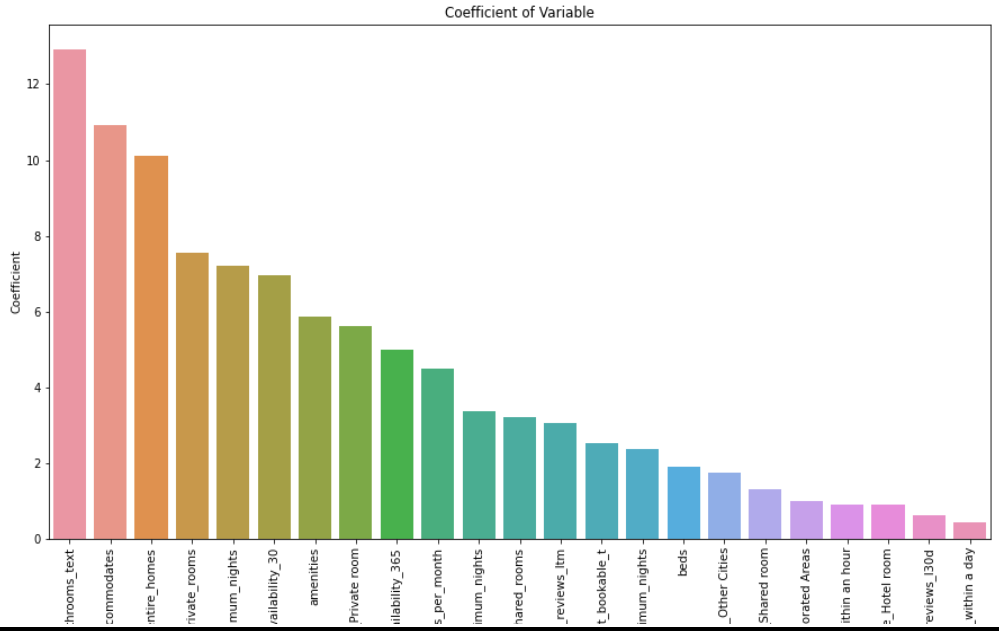
Root mean squared error for the CATBoost Model: 0.50

Mean absolute error for the CATBoost Model: 0.36

Mean absolute percentage error for the CATBoost Model: 139.78

The Significant variable and the coefficient are:

	Features	Coefficient
9	bathrooms_text	12.93
8	accommodates	10.92
19	calculated_host_listings_count_entire_homes	10.11
20	calculated_host_listings_count_private_rooms	7.56
13	maximum_minimum_nights	7.21
15	availability_30	6.95
11	amenities	5.87
5	room_type_Private room	5.60
16	availability_365	4.98
22	reviews_per_month	4.48
12	maximum_nights	3.36
21	calculated_host_listings_count_shared_rooms	3.23
17	number_of_reviews_ltm	3.04
7	instant_bookable_t	2.54
14	maximum_maximum_nights	2.38
10	beds	1.90
2	neighbourhood_group_cleansed_Other Cities	1.76
6	room_type_Shared room	1.32
3	neighbourhood_group_cleansed_Unincorporated Areas	0.98
1	host_response_time_within an hour	0.91
4	room_type_Hotel room	0.91
18	number_of_reviews_l30d	0.62
0	host_response_time_within a day	0.44



Mean Score: 0.686833741595677
Bias error: 31.316625840432298
Variance error: 7.103604294714113

INFERENCES AND RECOMMENDATIONS:

Inference – Some of the important features are bathrooms_text (no_of_bathrooms), accomdates,calculated_host_listings_count_entire_homes,calculated_host_listings_count_private_homes, maximum_minimum_nights, availability_30, amenities,room_type_private, reviews_per_month, beds and etc.

1. The no of accomdates helps to predict and also increase the cost of the property for a night.
2. The no of bathrooms tends to predict and increase the cost of the property for a night.
3. The amenities tends to predict and the increase of the property for a night.
4. Types of rooms – entire home has high price for the property for a night.
5. Types of rooms – private rooms has lesser price than compare to entire homes of the property for a night.
6. The maximum and minimum night's feature helps to predict the price of the property for a night.
7. Previous number of reviews helps to predict the price of the property for a night.
8. Instant_Bookable feature helps to predict the price of the property for a night.
9. The more cleaner neighbourhood tends to increase the price of the property for a night.

BUSINESS INFERENCE:

- Based on the type of rooms – entire home/type, its amenities, no of beds and no of bathrooms and no of accomdates are the top features that helps to predict the price of the property for a night.
- Despite of the property size and its amenities, if the property is located at the cleaner neighbourhood that tends to increase the price of the property for a night.
- Also the features like instant bookable and no of reviews helps to predict the price of the property.

LIMITATIONS, CHALLENGES AND SCOPE

Limitations of Data

The dataset belongs to properties from the location of Los Angeles, USA. The model will be more robust if the data would have belonged from different regions of the USA.

Challenges

1. High cardinality results in huge training effort in model tuning due to increase in model complexity (i.e. more number of features)
2. We also faced challenges on robust model tuning on all the models. Due to computational limitations, we are limited to very limited hyper parameters tuning technique using Grid Search.

Scope

1. We can approach the same concepts, using PCA as more than 60% of features having multicollinearity. Explanation of the each feature contribution's is not feasible.
2. Can perform much more hyper parameter tuning for the CatBoost model. Due to lower processing power of our laptops, we couldn't approach that.
3. Exploring Google collab as an option for model training and tuning with faster lead time.
4. Exploring some robust data sampling technique as part of choosing smaller sample (a true representation of population data) from the population data.