2021
S1

QBUS3820 Machine Learning and Data Mining in Business

AIRBNB HOUSING PRICE ANALYSIS

GROUP 6: 480282594, 480473435, 500486184

THE UNIVERSITY OF SYDNEY

Group Assignment

Group Number: Group 6

Group Members: 480282594, 480473435, 500486184

Table of Contents

Introduction

With the increasing global popularity of Airbnb, hosts, real-estate investors and other stakeholders are presented with an easily accessible opportunity to supplement or diversify their income streams through short-term rentals of their properties.

This report seeks to analyse the most influential factors contributing to revenue generation, presenting patterns and insights from the available data to formulate a predictive model based on advanced machine learning techniques, generating insights for property owners and investors.

A range of machine learning methods, including linear regression, Lasso regression, Ridge regression, Random Forests, XGBoost, LightGBM and Model stacking, have been considered to achieve these goals. Feature engineering was conducted for the different machine learning methods based on their characteristics and assumptions to maximise the predictive power of the models while maintaining reasonable computational costs. Following the assessment of the predictive performance of the different models on the validation dataset, model stacking was selected as the final predictive model and used to predict the unknown Airbnb prices in the test dataset.

Through data mining, it was found that there were three features that most significantly influenced the price of a property: the number of bedrooms, if the listing was for a private room, and the cleaning fee as a percentage of the listing price. Of these features, the number of bedrooms had a positive relationship with the price, whereas the price generally decreased if the listing was for a private room or as the cleaning fee percentage increased.

The report will conclude with a recommendations section, providing potential suggestions and considerations for hosts and other stakeholders seeking to adjust the pricing of their Airbnb properties.

Data processing

The training dataset contains highly detailed information on the 10,635 existing Airbnb listings in Sydney, and the test dataset includes information on 24,818 listings. Both datasets shared the same features. The training dataset was used to find the best predictive model, while the test dataset was used for model evaluation. The data within these datasets were cleaned, and missing values were addressed prior to exploratory data analysis (EDA).

Data cleaning

For computational convenience, the ‘id’ column was used as the index for the training and the test datasets. These two datasets contained the same 82 features that could potentially affect the prices, including text, numerical and categorical features. Additionally, the training dataset contains the response variable ‘price’. As both datasets required data processing, they were first combined for data cleaning and imputation and separated in later sections.

The features ‘host_response_rate’ and ‘host_acceptance_rate’ were converted to type ‘float’ as they were misclassified as objects. The dabl package was then used to detect column types and remove useless variables automatically. Seven variables were considered useless by the dabl package: ‘experiences_offered’, ‘bed_type’, ‘requires_license’, ‘is_business_travel_ready’, ‘require_guest_profile_picture’, ‘require_guest_phone_verification’ and ‘calculated_host_listings_count_shared_rooms’. They were considered useless because a significant proportion of their values were concentrated within one category, failing to provide representative information. Thus, they were removed to enhance the performance of the model. Furthermore, the variables ‘host_id’, ‘host_since’, ‘first_review’, and ‘last_review’ were examined and deemed irrelevant to the response variable based on the correlation analysis and domain knowledge.

Dealing with missing values

The proportion of missing values in each feature were calculated and displayed in descending order. Table 1 shows 32 features with null values, of which ‘square_feet’ has the most null values (35,279). Three variables (‘square_feet’, ‘weekly_discount’, ‘monthly_discount’) have more than 90% of missing values and were removed from the dataset.

Table 1: Missing Percentages

Predictor	Missing Percentage (%)	Predictor	Missing Percentage (%)	Predictor	Missing Percentage (%)
square_feet	99.509	**transit**	33.289	**space**	26.542
monthly_discount	95.290	**host_acceptance_rate**	32.319	**reviews_per_month**	24.723
weekly_discount	92.147	**security_deposit_perc**	32.088	**cleaning_fee_perc**	23.532
notes	57.936	**host_neighbourhood**	29.656	**neighbourhood**	12.792
host_about	45.170	**review_scores_value**	27.710	**beds**	0.669
access	42.682	**review_scores_checkin**	27.699	**zipcode**	0.324
host_response_rate	42.016	**review_scores_location**	27.693	**host_location**	0.107
host_resposne_time	42.016	**review_scores_accuracy**	27.662	**city**	0.087
house_rules	41.418	**review_scores_communication**	27.628	**bedrooms**	0.056
interaction	38.544	**review_scores_cleanliness**	27.617	**bathrooms**	0.025
neighborhood_overview	34.347	**review_scores_rating**	27.563		

Several approaches were considered when processing the null values. Firstly, the potential relationship between ‘host_resposne_time’, ‘host_response_rate’ and ‘host_acceptance_rate’ was explored. It was considered unreasonable to assume that over 40% of hosts did not respond to inquiries, as Airbnb reduces the search priority of listings with unresponsive hosts (Airbnb, n.d.). Given this assessment, the missing entries in the ‘host_response_rate’ and ‘host_acceptance_rates’ were replaced with their respective means. The missing values within ‘host_response_time’ were replaced with a new category of 'None', which was later merged with the category with the most similar statistical characteristics.

Secondly, it was determined based on domain knowledge that the observations for the 'review_score_location' feature were likely to be similar for properties located in the same 'street'. Hence, the missing values in this feature were filled with the mean value of other properties on the same street. The remaining empty values within this feature were replaced with the mean value of all observations. The missing observations in the ‘bedrooms’ and ‘beds’ features were replaced with median values based on the number of guests that the property could accommodate. The median was chosen to keep all observations within the feature in integer form. The missing values in the ‘bathrooms’ feature were replaced with median values according to the number of ‘bedrooms’ within the property. The missing values in the ‘bathrooms’ feature were replaced with median values according to the number of ‘bedrooms’ included in the listing. For the remaining features, the median observation of the feature was used to replace the null values in each numerical feature, and missing observations in other text features were replaced with ‘None’.

Subsequently, the dataset was reassessed to ensure all missing values had been handled. Of the ten features related to the listing location, the feature 'neighbourhood_cleansed' was selected to represent the location of the property as it contained the least errors with no missing values. Other features were subsequently discarded from the dataset to prevent issues with multicollinearity.

Exploratory Data Analysis (EDA)

Response variable: Price

Prior to conducting EDA, the training and test datasets were split, as EDA is performed only on the training dataset by convention. Following this split, the training dataset contained 10,635 observations. The maximum value of price was $4300.000, while the median price was $144.000, and the mean price was $212.000. The distribution plots of price and the descriptive statistics suggest that the response variable is highly right-skewed, with a skewness of 5.703 and kurtosis of 52.601, significantly deviating from a normal distribution and potentially indicating the presence of outliers. This deviation suggests that log transformation may be needed for linear models. Following a log transformation, the price becomes approximately normally distributed with a slight right skew, with skewness 0.850 and kurtosis 0.826.

Figure 1: Distribution Plot of Price

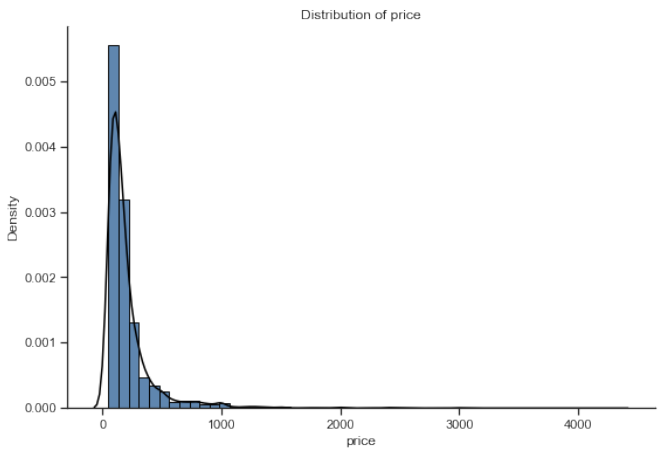


Table 2: Descriptive Statistics of Price

Price	
Count	10635.000
Mean	212.368
STD	255.625
Min	51.000
25%	89.000
50%	144.000
75%	231.000
Max	4300.000

Figure 2: Distribution Plot of Log Price

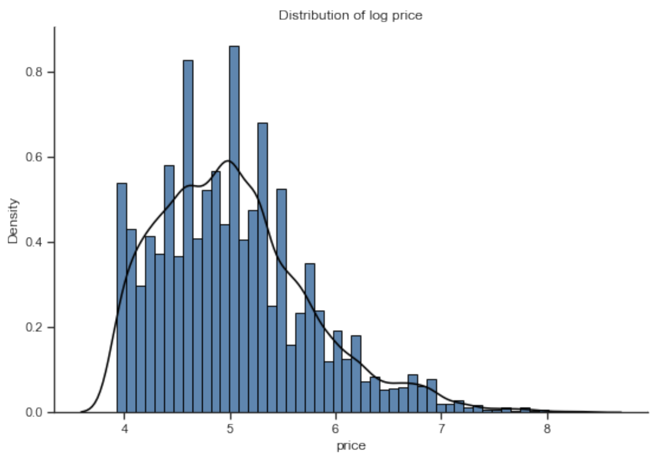


Table 3: Descriptive Statistics of Price

Price	
Count	10635.000
Mean	5.035
STD	0.723
Min	3.932
25%	4.489
50%	4.970
75%	5.442
Max	8.366

Numerical variables

A correlation heatmap was created to explore the bivariate relationships between numerical variables. The heatmap (Appendix A) indicates multiple highly correlated features that need to be deleted to reduce multicollinearity. To assess which features needed to be discarded, further assessment was conducted through the construction of a correlation and mutual information table to evaluate the relationship between the predictors, as well as the relationship between the predictors and the response variable.

1. As shown in Table 4, four features related to the total listings of each host showed a correlation above 0.9, raising significant concerns regarding multicollinearity between these features. Despite this, all four features demonstrated a correlation below 0.2 with the response variable, indicating a weak linear relationship with price. Thus, from the listed features, only ‘calculated_host_count_entire_homes’ will be kept in the data set as it has the most significant linear relationship with price, with a correlation of 0.361.

Table 4: Correlation Table of Host Listings

	host_listings_c	**host_total_listings**	**cal_host_listings**	**cal_host_count_**

	ount	_count	_count	entire _homes
host_listings_count	1.000			
host_total_listings_ count	1.000	1.000		
cal_host_listings_co unt	0.937	0.937	1.000	
cal_host_count_ent ire _homes	0.932	0.932	0.994	1.000
Price CORR	0.164	0.166	0.164	0.361
Price MI	0.182	0.182	0.168	0.181

2. Eight features were related to the minimum and maximum nights that a guest could stay. Among these eight features, 'min_average_nights_ntm' and 'max_average_nights_ntm' will be kept for further data analysis based on their low correlations with each other (Table 5), and their relatively high mutual information and correlation with price.

Table 5 Correlation Table of Max & Min Nights

	Min	**Max**	**Min_min**	**Max_min**	**Min_max**	**Max_max**	**Min_avg**	**Max_avg**
Min	1.000							
Max	0.003	1.000						
Min_min	0.863	-0.012	1.000					
Max_min	0.868	-0.002	0.957	1.000				
Min_max	-0.002	0.008	-0.002	-0.001	1.000	1.000		
Max_max	-0.002	0.008	-0.002	-0.001	1.000	1.000		
Min_avg	0.880	-0.003	0.978	0.992	-0.002	-0.002	1.000	
Max_avg	-0.002	0.008	-0.002	-0.001	1.000	1.000	-0.002	1.000
Price CORR	0.012	-0.015	0.015	0.024	0.010	0.010	0.017	0.010
Price MI	0.0889	0.108	0.0696	0.0971	0.0452	0.0289	0.114	0.0337

3. Among variables related to the accommodation of guests, the feature 'accommodates' will be removed as its correlation with both bedrooms and beds is greater than 0.8 (Table 5).

Table 6: Correlation Table of Accommodates, Bathrooms, Bedrooms & Beds

	accommodates	**bathrooms**	**bedrooms**	**beds**
accommodates	1.000			
bathrooms	0.549	1.000		
bedrooms	0.842	0.598	1.000	
beds	0.871	0.520	0.808	1.000
Price CORR	0.574	0.497	0.600	0.519
Price MI	0.360	0.227	0.365	0.263

4. Features regarding availability suggest the available days of a listing for the next 30/60/90

days and one year. Among these four integer variables, ‘availability_30’ is kept for further data analysis as it has the highest mutual information with price (Table 7).

Table 7: Correlation Table of Availability

Availability	_30	_60	_90	_365
_30	1.000			
_60	0.947	1.000		
_90	0.893	0.977	1.000	
_365	0.632	0.704	0.747	1.000
Price CORR	0.094	0.089	0.087	0.124
Price MI	0.044	0.043	0.040	0.034

5. There are seven features related to the review score: rating, accuracy, cleanliness, check-in, communication, location and value. Among these features, ‘review_score_rating’ will be deleted as it is an overall rating created by Airbnb with its algorithm based on other scores.

Table 8: Correlation Table of Review Scores

	Rating	Accuracy	Cleanliness	Checkin	Communication	Location	Value
Rating	1.000						
Accuracy	0.752	1.000					
Clean	0.722	0.660	1.000				
Checkin	0.581	0.555	0.457	1.000			
Commun	0.642	0.619	0.488	0.693	1.000		
Location	0.486	0.484	0.412	0.432	0.466	1.000	
Value	0.719	0.689	0.656	0.521	0.569	0.516	1.000
Price Corr	0.058	0.039	0.049	0.040	0.031	0.056	-0.007
Price MI	0.034	0.073	0.045	0.085	0.113	0.113	0.058

6. Among the features related to the number of reviews, the number of reviews for the property over the most recent twelve months, i.e., ‘number_of_reviews_ltm’, has been discarded as it had a significant correlation with ‘reviews_per_month’ (Table 9).

Table 9: Correlation Table of Reviews

	number_of_reviews	number_of_reviews_ltm	reviews_per_month
number_of_reviews	1.000		
number_of_reviews_ltm	0.784	1.000	
reviews_per_month	0.665	0.858	1.000
Price Corr	-0.072	-0.071	-0.066
Price MI	0.034	0.039	0.058

Categorical variables

There are 9 categorical variables in the training dataset. The boxplot (Appendix B), table 10 and table 11 show that ‘host_is_superhost’, ‘host_identity_verified’, ‘is_location_exact’ and ‘instant_bookable’, have similar median and mean listing prices among the different categories. Therefore, these four variables will be discarded as they are not representative.

Table 10: Median Price

Median price ($)	False	True
host_is_superhost	142.000	149.000
host_identity_verified	140.000	149.000
is_location_exact	140.000	144.000
instant_bookable	150.000	138.000

Table 11: Mean Price

Mean Price ($)	False	True
host_is_superhost	211.842	215.394
host_identity_verified	211.358	214.191
is_location_exact	204.992	214.699
instant_bookable	230.598	189.610

Detailed EDA on remaining variables

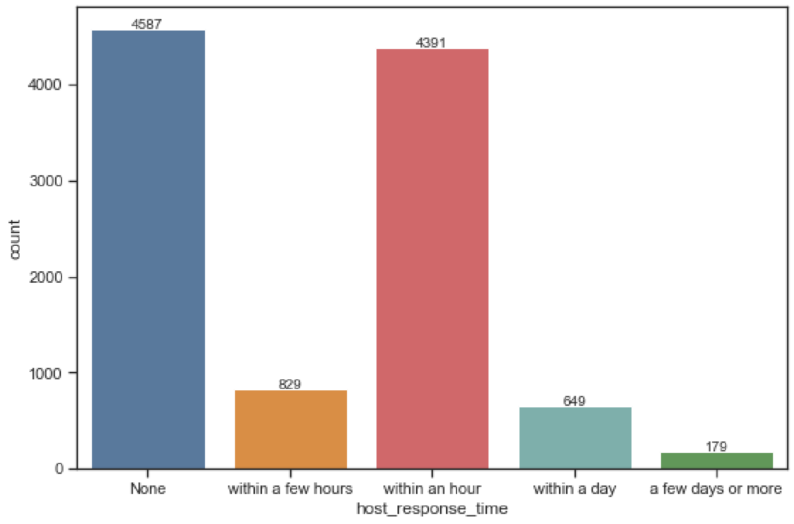
Host_response_rate & host_acceptance_rate: The missing values in these two features are imputed with their respective means to preserve information, as the vast majority of the observations within these two features are concentrated between 0.9 and 1. However, they appear to have a weak relationship with price (Appendix C). As the missing values within these two features were replaced with their respective mean values, both features demonstrate a significantly left-skewed distribution.

Host_response_time: This feature contains five different categories. Of these categories, Table 12 shows that 'within a few hours' and 'within an hour' had the highest median value of price while 'a few days or more' had the lowest median price. Figure 3 shows that most observations are concentrated in 'None' and 'within an hour'.

Table 12: Descriptive Statistics of host_response_time

	None	A few days or more	Within a day	Within a few hours	Within an hour
Count	4587	179	649	829	4391
Median	129	110	140	150	150

Figure 3: Count Plot of host_response_time



Neighborhood_cleansed: The count plot (Appendix E) shows that Sydney (2890 observations), Waverley (1469) and Randwick (890) have the most listings. This outcome is unsurprising, as Sydney is the most popular place for travellers in Australia and Waverley & Randwick have famous tourist attractions. Among all neighbourhoods, Pittwater has the highest median price at $350, followed by Manly ($195), Hunters Hill ($194.5), Mosman ($185) and Waverley ($165). These five neighbourhoods are located near beaches and famous tourist attractions and are thus likely to be in greater demand, which could be associated with higher prices, especially around peak tourist season.

Table 13: Median Prices of Different Neighbourhoods

Price ($)	Highest 5	Price ($)	Lowest 5
Pittwater	350.00	**Campbelltown**	89.00
Manly	195.00	**City of Kogarah**	88.50
Hunters Hill	194.50	**The Hills Shire**	85.00
Mosman	185.00	**Blacktown**	79.00
Waverley	165.00	**Holroyd**	74.00

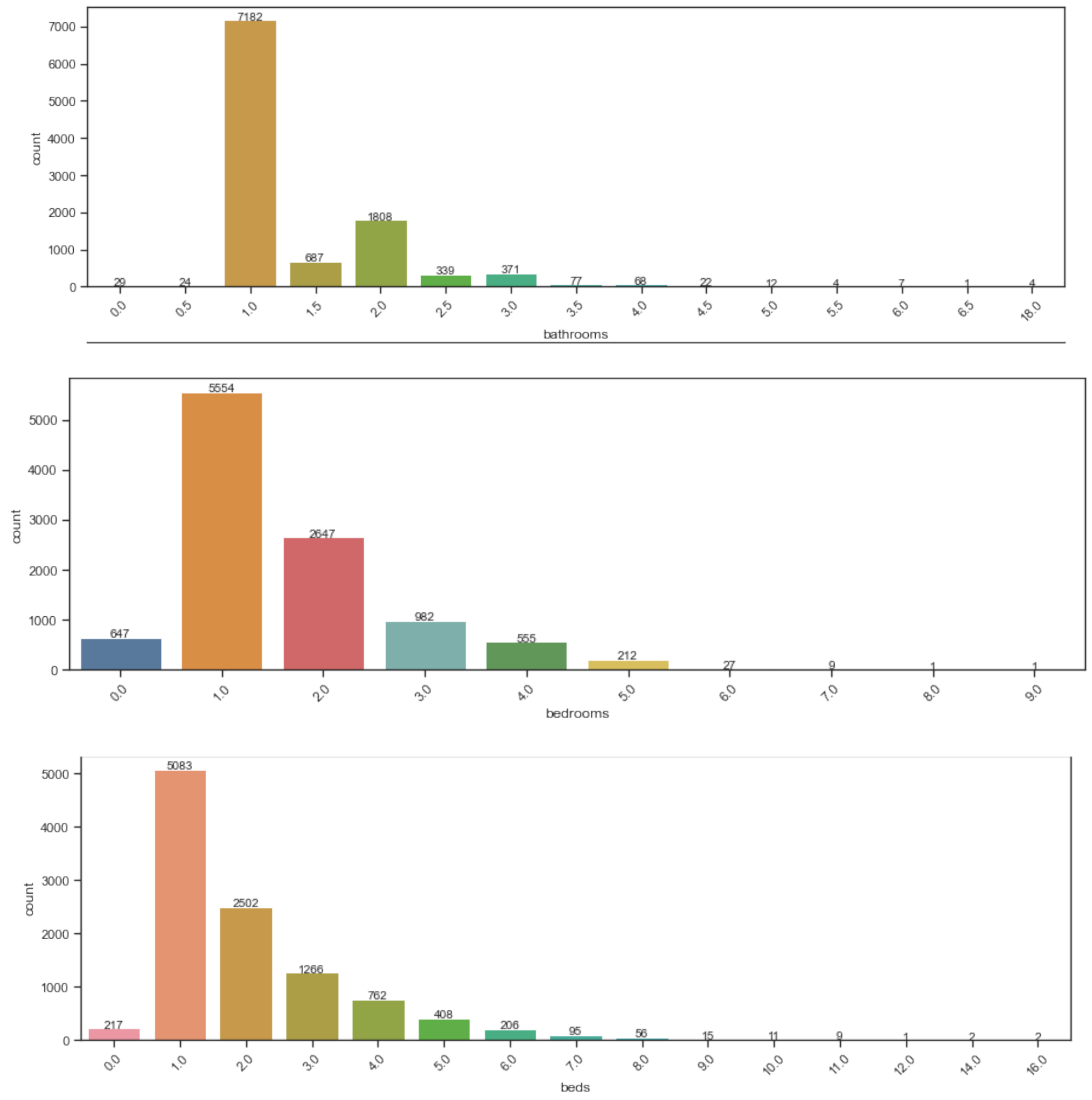
Property_type: Most observations within this feature were concentrated in 'Apartment' (6492 observations) and 'House' (2671 observations) (Appendix F). Among the different property types, boat and castle had the highest median price. However, they are sparse categories, with only 10 and 1 observations, respectively.

Room_type: (Appendix F) there are four different types of room; of these, hotel rooms had the highest median price ($191) and shared room had the lowest ($70). Most observations were concentrated in 'Entire home/apt' (7202 observations) and 'Private room' (3307 observations).

Cancellation_policy: There were seven categories within this feature, among which, two luxury cancellation policies, ‘luxury_moderate’ and ‘luxury_super_strict_125’, had the highest median price. The flexible cancellation policy had the lowest median price ($110) (Appendix F). These findings are intuitive as the cancellation policy refers to the lead time allowed for free cancellation. As the policy becomes stricter, the cancellation fee increases to protect the interests of the hosts. The observations are mostly concentrated in ‘flexible’ (3312 observations), ‘moderate’ (2586 observations), and ‘strict_14_with_grace_period’ (4651 observations).

<u>Bathrooms, Bedrooms & Beds:</u> Figure 3 shows that the observations for these features were concentrated at specific values. In the case of bathrooms, the observations are concentrated at 1.0 and 2.0. For bedrooms, the observations were concentrated between 1.0 to 3.0. For the number of beds, the observations were concentrated between 1.0 to 4.0. The boxplots suggest that these three features are ordinal, as the price of properties increased with the number of bathrooms, bedrooms and beds (Appendix G).

Figure 4: Count Plot of Bathrooms, Bedrooms & Beds



Security_deposit_perc, cleaning_fee_perc & extra_people_perc: The distributions of these features demonstrated a significant positive skew, indicating that most hosts keep the deposit, cleaning fee and the fee for additional guests within a certain range unless the room or house met other specific criteria (Appendix H). Transformations such as the log, box-cox and Yeo-Johnson transformation may be needed. Analysis of the features also revealed the presence of outliers. However, these outliers were not removed because there is no information suggesting that these observations were illegitimate.

Guest_included: The observations for this feature ranged from 1 to 15, concentrating on 1. The number of guests included was found to positively correlate with the response variable, indicating that this feature should be considered an ordinal variable (Appendix I).

Availability_30: This feature represents the availability of a property for the next 30 days as set by the host. The observations ranged from 0 to 30, concentrating on 0 (6035 observations) (Appendix I).

Reviews: Although 'number_of_reviews' and 'reviews_per_month' are highly correlated with a correlation of 0.665, they have both been retained as part of the training set, given that they convey different messages. Transformations are required since they are highly right-skewed (see Appendix J). Transformations are required since they are highly right-skewed (Appendix J).

Review scores: there are six different review scores, including accuracy, cleanliness, check-in, communication, location, and value. These features are nominal variables that act as a good indicator of customer satisfaction. The distribution plots (Appendix K) suggests that most customers' aggregate review scores were higher than 0.8. The regression plots suggest that a property's review score is positively correlated with its price, indicating that this feature should be considered an ordinal variable (Appendix K).

Calculated_host_listings_count_entire_homes: This feature represents a host's total listings for entire homes, while **calculated_host_listings_count_private_rooms** represents the number of a host's total listings for private rooms. These two features are highly right-skewed with a concentration from 0 to 2.   These two features are highly right-skewed, and observations are concentrated between 0 to 2 (Appendix L).

Feature engineering

1. For features with observations concentrated at specific categories, categories with fewer observations were combined into new categories to reduce sparsity and improve the representativeness of the sample.
    a. Host Response Time: The categories 'within a few hours', 'within a day', and 'a few days or more' were found to have significantly similar statistical characteristics and was subsequently combined into a new category: 'after an hour'.
    b. Property type: Except for 'apartment' and 'house, the different property types were combined based on their respective statistical characteristics into five categories: 'under 150', 'under 200', 'under 250' and 'under 350'. The variable types 'boat' and 'castle' were also not combined because they did not display similar statistical characteristics with any other property type.
    c. Room type: The categories 'private room' and 'shared room' were combined due to

their similar statistical characteristics.

d. Cancellation policy: The categories ‘luxury_moderate’ and ‘luxury_super_strict’ were combined as they had similar characteristics and were sparse.
e. Bathrooms: The existing categories for bathrooms were merged into ‘1-2’, ‘2-3’, ‘4-5’, and ‘>5’. The categories 0, 0.5, 3, and 3.5 were not merged due to them possessing significantly different statistical characteristics from other categories.
f. Beds: Since the values are concentrated from 0 to 5, observations greater than five were combined into ‘6-7’, ‘8’, ‘9-10’, and ‘>10’.
g. Guest_included: values greater than 4 were combined into ‘5-6’, and ‘>6’ to reduce sparsity.Guest_included: values greater than four were combined into ‘5-6’ and ‘>6’ to reduce sparsity.
h. Calculated_host_listings_count_entire_homes & calculated_host_listings_count_private_rooms: All values greater than 1 were combined into ‘>1’ to reduce sparsity.

Log transformation was applied to numerical variables with a skewness greater than 0.75 since machine learning methods usually prefer normally distributed predictors. The log transformation method was chosen because both the Box-Cox transformation and the Yeo-Johnson transformation have a transformation parameter λ, which can be sensitive to outliers, according to Atkinson's study in 2021. The utilisation of these two methods for practical applications can cause the following problems: First, it is difficult to find an ideal transformation parameter value that meets the fitting constraints of the assumed distribution of the converted data while also minimising model errors; Second, the transformations alter the nature of the link between model variables, potentially resulting in an imbalance between the efficiency of statistical inference and the ability to describe the magnitude of variable influence (Othman & Ali, 2021). Therefore, although these two methods have their advantages, the generally applicable log transformation method has been used for normalisation.

Based on the exploratory data analysis and the domain knowledge, the label encoder was applied for ‘host_response_time’, ‘bathrooms’, ‘bedrooms’, ‘beds’, ‘guests_included’, ‘calculated_host_listings_count_entire_homes’, ‘calculated_host_listings_count_private_rooms’, ‘review_scores_accuracy’, ‘review_scores_cleanliness’, ‘review_scores_checkin’, ‘review_scores_communication’, ‘review_scores_location’, ‘review_scores_value’ to encode these variables into ordinal variables.

Dummy variables were created for categorical variables. However, in the case of XGBoost and LightGBM, varying degrees of feature engineering were tested, and it was found that a lower level of feature engineering would generally lead to an increase in performance. This could be because these methods possess sophisticated and in-built methods to deal with missing values without making distribution assumptions. Hence, only the text variables were engineered for XGBoost and LightGBM. Although the random forest method could also deal with missing values with no assumptions on the dataset, feature engineering was applied to reduce the number of features, and hence reduce the computational cost of the hyperparameter optimisation process.

For text variables, two variables that were deemed to be the most relevant to listing prices based on domain knowledge were selected: ‘host_verifications’ and ‘amenities’. The process_text function was applied separately to each element of these two columns and the frequency of each word (feature) was counted for each cell in the dataset. The ten most common words in each feature were then recorded in the 'tokens' and 'tokens2' variables respectively. New columns were created in the dataset for each word featured in 'tokens' and 'tokens2', with the value of the column set to 1 if the corresponding word featured in the listing and 0 if otherwise.

Methodology

In this section, five models were built to predict the prices based on the available features. The three best models (Lasso, XGBoost and Model stacking), selected based on the scores given by Kaggle, will be thoroughly discussed, and the other two models (Random Forest and LightGBM) will be briefly explained. Other models, including Ridge regression and Catboost, were also explored but will not be discussed in this report.

Model 1: Lasso

The Lasso model was selected as the best linear model based on Kaggle scores. Lasso refers to the Least Absolute Shrinkage and Selection Operator, which performs ℓ_1 regularisation to enhance model performance with two main properties: variable selection and shrinkage. Through shrinking the coefficients of features, the lasso method is able to significantly reduce variance without substantially increasing bias. This particular application of shrinkage also allows it to perform variable selection, as the coefficients for features are reduced to 0 individually, compared to the ridge method, which simultaneously sets all feature coefficients to 0.

Therefore, an important advantage of Lasso over Ridge regression is its interpretability. Following data processing and feature engineering, the training dataset contained over 10,000 observations and 130 features. Therefore, following the goal of identifying important features to enhance the stakeholders' decision-making, the lasso method was deemed more appropriate than the ridge regression method due to its feature selection capabilities.

When compared with subset selection, which chooses the best subset of variables based on certain criteria, Lasso is much more computationally efficient. The subset selection method is much more time-consuming because of the combinatorial explosion of the parameters. It also leads to a nonconvex problem that is more difficult to fit.

Hyperparameter tuning

$$\widehat{\beta_{lasso}} = \underset{\beta}{argmin}\left\{\sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{i=1}^{p}|\beta_j|\right\}$$

The most critical part of the formula is the hyperparameter λ. The hyperparameter value is essential in enhancing prediction accuracy and interpretability as it controls the strength of variable selection and shrinkage.

When the hyperparameter λ equals 0, the Lasso will simply reduce to an OLS problem.

1. When $\lambda = \infty$, all of the coefficients in the model will be shrined to zero.
2. Increasing λ will increase the bias and decrease the variance, leading to an underfitting problem and vice versa.

Commonly, the value of λ is chosen by cross-validation. The lowest CV score is achieved when the hyperparameter for Lasso in this project is set to be 0.0001 (4 d.p.)

Since the lasso method constrains the size of the coefficients based on the magnitude of each variable, we first standardised the predictors in both the training and test dataset through the robust scaler to ensure that all predictors are on the same scale.

Model results & limitation

First, we use the k-fold cross-validation method to find how the Lasso model we built fits our training data. The method involves dividing the dataset into k equal groups, each of which was utilised once as a test group (to be predicted) and the rest of the time as a training group (to predict the others). Here we choose k=5, which divides the dataset into 5 equal groups. The cross-validation score of the lasso method was evaluated using the 'cross_val_score' function. When λ =0.0001, the lowest root mean squared error (RMSE) of this LASSO model on the training dataset is 0.3909. This model will be used to predict the daily Price of Airbnb rentals contained in the validation dataset, and an exponential transformation has been applied to the predictions, as the response variable used to train the model was log-transformed.

The main disadvantage of the lasso method is that it is not robust to highly correlated variables in the dataset. The steps taken in the data processing and feature engineering sections aimed to overcome this limitation by removing features exhibiting high correlation.

Model 2: XGBoost

XGBoost refers to Extreme Gradient Boosting. The XGBoost model was the best nonparametric model based on Kaggle scores. XGBoost was used in this project as there are more than 10,000 observations with more than 100 features after feature engineering, and the dataset contains both categorical and numeric variables. The XGBoost model is a widely used application of gradient boosting with significant advantages compared to other models. In particular, XGBoost tends to have greater execution speed and predictive performance (Jason, 2016). Furthermore, as the XGBoost model is well documented, it can be easily accessed and tuned to the requirements of the relevant stakeholders for future predictions, with a relatively lower level of required training.

In boosting, we use an additive model with decision trees as basis functions that are essentially weak learners to add up a strong learner. The model's capacity increases with the number of basis functions, eventually adapting to the size of the training data. However, XGBoost uses a regularised form of gradient boosting to penalise the complexity of each tree and thus prevent overfitting (Neetika, 2020). It solves the regularised risk minimisation problem with a penalty factor on complexity $\Omega(\theta_m)$.

We specified the penalty term by:

$$\Omega(\theta_m) = \gamma|T_m| + \frac{\lambda}{2}\sum_{j=1}^{|T_m|} w_{j_m}^2$$

The penalty term here can be considered as a ℓ_1 regularisation and the hyperparameters in $\Omega(\theta_m)$ is γ and λ.

Hyperparameter tuning

In python, we use the XGBRegressor package to perform the XGBoost model. In our implementation, we optimised the following hyperparameters through the random search method:

1. n_estimators: The number of gradient boosting trees (Python API Reference, n.d.). which is similar to the n_estimator in the RandomForestRegressor. Here we set the n_estimator to follow a discrete uniform distribution between 100 and 2500, and the best n_estimator found by the randomised search is 1623.

2. learning_rate: Boosting learning rate (Python API Reference, n.d.). This is the shrinkage factor in gradient boosting, adjusted to slow down the learning in the model and prevent overfitting; the model's performance and computational costs increase as the learning rate decreases. Here we set the learning_rate to follow a uniform distribution between 0.005 and 0.1, and the best learning_rate found by the randomised search is 0.033.

3. max_depth: Maximum tree depth for base learners (Python API Reference, n.d.). The higher the value of max_depth, the better the results. However, a higher maximum depth can also lead to overfitting as the model will capture too much information, picking up 'noise' from the training set. In this case, the optimal parameter is 4.

4. sub_sample: Subsample ratio of the training instance. A sub_sample of 0.5 means the XGBoost will randomly sample only half of the training data before growing trees to prevent overfitting (Python API Reference, n.d.). We set the sub_sample to follow a uniform distribution between 0.5 and 1, and the best sub_sample found by the randomised search is 0.695.

5. reg_alpha & reg_lamda : ℓ_1& ℓ_2 regularisation term (XGBoost Python Package, n.d.). The higher the value, the more conservative the model will be. The tuned parameter is 0.744.

Model results & limitation

The cross-validation RMSE of this XGBoost model is 0.326, which is significantly less than that of the Lasso model. This model will be used to predict the daily Price of Airbnb rentals contained in the validation dataset. Although this method makes no assumptions on the normality of the features and response variable, it was found that a better generalisation performance was achieved when a log-transformation was applied to the response variable. This is most likely due to the way in which the underlying decision trees split the data.

Although XGBoost is designed to handle large datasets, it can be memory and time-consuming, especially when compared to LightGBM. If Airbnb wants to use this model to fit a larger dataset, it will require much more computing power and resources.

Model 3: Random Forest

Random Forest is based on the idea of bagging. Bagging is a form of bootstrap aggregation where bootstrap samples are generated by randomly sampling from the training data with replacement, after which a decision tree is fitted to each sample, and the final prediction is taken as the average prediction across all trees (Marcel, 2021). The random forest method further improves model performance by only selecting a subset of the features for each decision node. Additionally, Random Forests can decorrelate trees to largely reduce the variance and improve prediction stability (James et al. 2013). The details of this algorithm are shown in Appendix M.

In this algorithm, multiple hyperparameters require tuning. The number of trees in the forest was set to 1000. The optimised value of max_features, the number of features to consider when looking for the best split, was 34. The minimum number of samples required to be at a leaf node (RandomForestRegressor, n.d.) was optimised at 1. This value will lead to better test set performance, but also raises concerns regarding overfitting.

The cross-validation RMSE of this Random Forest model is 0.355. The Random Forest algorithm has various limitations. Firstly, it requires a greater amount of computing power, time and resources than other models. Due to limited computational power, most hyperparameters were set to default values in this project, which may have affected the performance of the model. Additionally, the Random Forest model is more challenging to interpret due to the complexity of random sampling, which is not ideal for companies like Airbnb that want to assess the detailed influence of individual factors.

Model 4: LightGBM

Similar to XGBoost, LightGBM is another efficient implementation of Gradient Boosting Decision Trees (GBDT). Compared to XGBoost, LightGBM further optimises memory usage, efficiency and accuracy.

Several key features of LightGBM are described below (Ke et al., 2017):

1. LightGBM employs a histogram-based algorithm, which consumes less memory and simplifies data separation;

2. LightGBM abandons the level-wise decision tree growth strategy used by most GBDT tools and uses a leaf-wise algorithm with depth restrictions to obtain better accuracy for the same number of splits;

3. LightGBM has optimised support for categorical features, forgoing the need for categorical processing through dummy variables (forgoing the need for dummy variables).

To achieve the highest level of predictive accuracy, several hyperparameters must be optimised for the LightGBM model. The optimised hyperparameters and their values were found through a crossvalidated random search and shown in Table 14.

Table 14: Hyperparameters of LightGBM

Hyperparameter	Value
Learning rate	0.067
Number of estimators	2039
Number of leaves	5
Regression lambda	0.157
Subsample	0.519

The resulting LightGBM model had a validation set RMSE of 0.327, showing a significant performance improvement over the linear models.

A possible disadvantage of the LightGBM model is that the leaf-wise strategy may result in greater tree depth, potentially leading to over-fitting. Another limitation is that as a new method, LightGBM not only has less information from the literature but also lacks the community support that might facilitate the optimised utilisation of this model (Kasturi, 2019).

Model 5: Model Stacking (Final Model)

Of the models assessed, the model stack containing the LightGBM, XGBoost, random forest, and ordinary least squares models demonstrated the highest generalisation ability. This model stacking method is considered an ensemble modelling method, combining predictions from the selected base models to overcome weaknesses in the individual models and improve generalisation performance. The weights of the base models within the ensemble are typically set to be equal or optimised using meta regressors, with models demonstrating greater generalisation ability often possessing greater weights. Many meta regressors were tested when optimising the weights of the base models, including linear regression, regression tree, and boosting models. Of these, the meta regressor based on the ordinary least squares method had the best validation set performance and was thus selected to be the final model. The addition of a third layer was considered. However, this was not realistic given our resource constraints.

The ensemble was first generated based on all other base models and reduced to contain only those with positive weights. These were the random forest, XGBoost, LightGBM, and ordinary least squares models, with weights of 0.065, 0.517, 0.424, and ≈0.000 respectively. This outcome is consistent with our findings in the previous sections, as the XGBoost model showed the highest predictive accuracy on the validation set, followed by the LightGBM and random forest models.

To assess the performance of the model, it was analysed based on five-fold cross-validation. The model displayed an RMSE of 0.320. Compared with previous models, this showed a slight improvement In predictive accuracy, with RMSE being 0.002 higher than the XGBoost model. Indeed, the model stack also showed higher performance on the final validation set, returning an RMSE score of 0.325

However, due to the ensembled nature of the model stack, it can be challenging to interpret. Indeed, we could extract the most influential variables and their respective effect for the overall model. However, the importance of individual features could potentially be extrapolated as a function of the weight of the base model and its importance in the base model. One main limitation of ensemble methods in practice is that it can be computationally expensive and time-consuming to train, as it requires the assembly of many base models to be effectively used.

Model Validation

All predictions generated by the models in this report were based on the data provided from the Kaggle competition. The validation score of each model was taken from their respective public root mean squared logarithmic errors from Kaggle. These scores have been summarised in Table 15. The scores in table 14 have been rounded to 4 d.p. to better represent differences in predictive accuracy.

Table 15: Kaggle Scores of Five Models

Rank	Model	Validation set Kaggle results
1st	Model Stacking (Final Model)	0.3245
2nd	XGBoost (Bench Mark Model)	0.3259
3rd	LightGBM	0.3275
4th	Random Forest	0.3580
5th	Lasso	0.3939

The XGBoost model was chosen as the benchmark model due to its widespread use in the industry and its strong predictive capabilities. While the XGBoost model performed better than all other non-ensemble models, the LightGBM model achieved a similar level of predictive accuracy with a fraction of the computational cost. Overall, the XGBost model and the LightGBM model demonstrated significantly greater generalisation ability over other assessed non-ensemble methods, with the random forest method performing better than linear models. The differences in performance can likely be attributed to two characteristics: the ability of tree-based methods to model nonlinear relationships, and the generalisation ability of the XGBoost and LightGBM models.

The model stack demonstrated the best generalisation performance, showing an improvement over the benchmark XGBoost model of 0.0014. This result suggests that the ensemble was able to improve the likelihood of achieving the best performance by combining the learning algorithms. One caveat, however, is that a complex meta-model may lead to overfitting concerns, thus affecting its performance on the test dataset. Therefore, this should be taken into account if the company wishes to continue using this model in the future.

Best Hosts

To discuss what the best hosts are doing, the criteria by which they are judged must first be defined. In the absence of information concerning hosting costs, the success of a host can ultimately be determined by its overall level of revenue. The level of revenue can be expressed as a function of the property's price and occupation rate. However, while proxies for the occupancy rate of the properties, including their availability rates, were considered, none of the features in the dataset were determined to have a substantially similar definition to the occupancy rate of a property. Given these definitions and the lack of critical information, the analysis for best practices for hosts has been conducted primarily regarding property price.

Of the three best performing non-ensemble models (XGBoost, LightGBM, random forest), three variables were identified as having the most importance: the cleaning fee of the listing as a percentage of the price, if the listing was for a private room, and the number of bedrooms within the listing (Appendix N). Of these three variables, the number of bedrooms had a feature importance rating of almost 100 in both the random forest and the XGBoost and is hence likely the most significant factor for hosts aiming to increase their listing price.

Table 16 Coefficients of the Identified Features

	coef	**Std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	97.635	10.212	9.560	0.000	77.617	117.653
Bedrooms	307.934	5.646	54.551	0.000	296.907	319.040
Cleaning_fee_perc	-35.304	2.161	-16.336	0.000	-39.541	-31.068
Room_type_Private_room	-96.687	4.766	-20.287	0.000	-106.029	-87.345

Table 16 shows the coefficients of the identified features estimated through linear regression. This finding is highly intuitive as the number of bedrooms within a listing is typically associated with the number of available occupants and the size of the property. However, it should be noted that it may be difficult for hosts to increase the number of bedrooms within their listings without incurring significant renovation costs.

The second most important feature was room_type_private_room, i.e., if the listing was for a private room. The estimated beta coefficient for this feature from a simple linear regression was approximately -96.687, suggesting that the price for listings of private rooms was $96.687 lower than other listings on average, with all other factors held equal. This finding is unsurprising as private rooms often use a lower price to differentiate from traditional hotel rooms.

Finally, the third most important feature was the cleaning fee as a percentage of the listing price. This is also intuitive as the cleaning fee is not included in the listing price and effectively increases the listing price. Hence, when the cleaning fee was at a lower percentage of the listing price, customers were willing to pay for a higher price. Although this feature is simple for hosts to adjust, further research can be done into the psychological effects of higher listing prices and higher cleaning fee percentages on the perceived value proposition of the property.

However, it is essential to recognise the limitations of the above analysis. It should be noted that the importance of the features was calculated based only on the data from the training set, a limited sample of the broader population. Hence, as more data becomes available, the importance of these features may change as sampling bias is reduced. Therefore, the analysis in this section should be considered suggestions for hosts and other stakeholders to consider when determining the rental prices of their listings, rather than rules. Furthermore, due to the lack of availability of crucial data, the occupancy rates of the listings have gone unexplored. Intuitively, the demand for a property is likely to decrease when its listing price increases. Hence, the maximisation of listing price may not always be desirable toward the goal of maximising revenue. As data regarding occupancy rates become available, the models should be re-evaluated and feature importance re-calculated to generate a more accurate and comprehensive understanding of the relationship between specific features and the revenue potential of a property and the balance between price and occupancy.

Conclusion and Future work

This report aimed to develop a predictive model to accurately predict the daily prices of Airbnb rentals based on various advanced machine learning techniques, outlining the reasoning, methodology, advantages and limitations of each model. The data analysis conducted seeks to assist Airbnb in advising hosts and other potential stakeholders about factors and best practices to maximise potential revenue, which has been identified as a function of the rental price and the occupancy rate.

We chose model stacking as the final model as it has the lowest Kaggle score, which means it has the strongest generalisation ability. Although the model stacking method seems to be the most likely to achieve the best prediction results, we should pay attention to the following two points: First, using this method will decrease the interpretability of the result, which can make it difficult for our clients to identify the specific influence of individual variables based on the model results; Second, the model becomes prone to overfitting as the complexity of the meta-model increases.

Another problem is that the dataset contains too many missing values. In this report, many of the missing values were simply replaced with ‘None’, ‘0’, the median or the mode of the features. Different methods of imputation can have significant effects on the predictions generated by predictive models. Additionally, the feature engineering section conducted within this report is limited. Although the features within the dataset have been analysed as best as possible, there were simply too many other potential ways to handle the complex relationships between the over 80 features. Hence, this report cannot guarantee that the models produced and their predictions are perfectly optimised.

For future study, the following considerations could be implemented to improve the accuracy of the predictions: First, it is vital to find a better way to deal with missing values and explore the correlation between variables. This report deleted several features with missing values of more than 90%, including a feature ‘square_feet’ that we believe is relevant to the response variable ‘price’ based on domain knowledge. Therefore, although the dataset contains many variables, we would recommend Airbnb to improve their data recording and scraping practices and obtain necessary information such as the occupancy rate of each house to answer the substantive question better. Second, a neural network could be utilised to predict listing prices, as the neural networks generally outperform machine learning models in large data sets and may provide higher accuracy. In addition, due to the nature of Airbnb's business, the data of the house is very stable, and there is no need for real-time training on the newly generated data. Therefore, Airbnb can use historical data to train neural networks offline, significantly reducing the level of required resources. Lastly, the model can be improved by finely tuning more hyperparameters with the help of more computing resources.

References

Airbnb. (n.d.) *What factors determine how my listing shows in search results?* Retrieved May 22, 2021, from https://www.airbnb.com/help/article/39/what-factors-determine-how-my-listing-shows-in-search-results

Atkinson, A. C., Riani, M., & Corbellini, A. (2021). The Box–Cox Transformation: Review and Extensions. *Statistical Science*, *36*(2). https://doi.org/10.1214/20-sts778

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning : with Applications in R. Springer New York.

Jason, B. (2021, February 17). *A gentle introduction to XGBoost for Applied Machine Learning*. Machine Learning Mastery. https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

Kasturi, S. N. (2019, July 11). *XGBOOST vs LightGBM: Which algorithm wins the race !!!* Towards Data Science. https://towardsdatascience.com/lightgbm-vs-xgboost-which-algorithm-win-the-race-1ff7dd4917d

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. Proceedings of the 31st International Conference on Neural Information Processing Systems, 3149–3157.

Lasso (Statistics). (n.d.). Wikipedia. Retrieved May 22, 2021, from https://en.wikipedia.org/wiki/Lasso_(statistics)

Marcel, S. (2021). QBUS3820 Machine learning and data mining in business, lecture 9, week 9: Random forests and boosting [Lecture PowerPoint slides]. Retrieved from https://canvas.sydney.edu.au/courses/32278/modules

Neetika, K. (2020, July 7). *A brief introduction to XGBoost.* Towards Data Science. https://towardsdatascience.com/a-brief-introduction-to-xgboost-3eaee2e3e5d6

Othman, S. A., & Ali, H. T. M. (2021). Improvement of the Nonparametric Estimation of Functional Stationary Time Series Using Yeo-Johnson Transformation with Application to Temperature Curves. *Advances in Mathematical Physics*, *2021*, 1–6. https://doi.org/10.1155/2021/6676400

Python API Reference. (n.d.). XGBoost. Retrieved May 22, 2021, from https://xgboost.readthedocs.io/en/latest/python/python_api.html

Qian, W. (2020, November 11). *LightGBM of machine learning algorithm*. Biaodianfu. https://www.biaodianfu.com/lightgbm.html

sklearn.ensemble.RandomForestRegressor. (n.d.). Scikit-lean. Retrieved May 22, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

Appendices

Appendix A – Correlation Heatmap

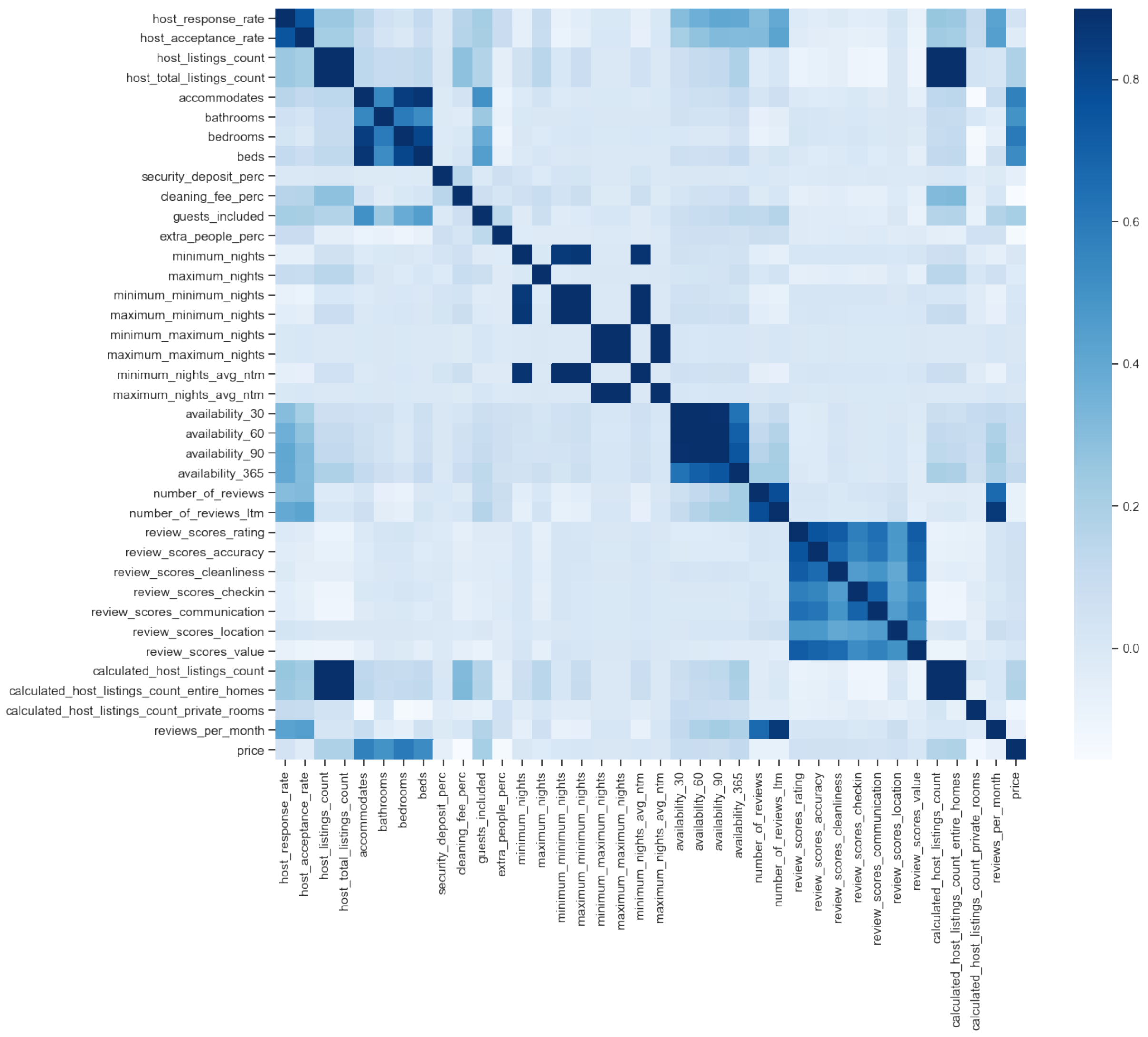


Figure A1 Correlation Heatmap

Appendix B – Categorical Variables

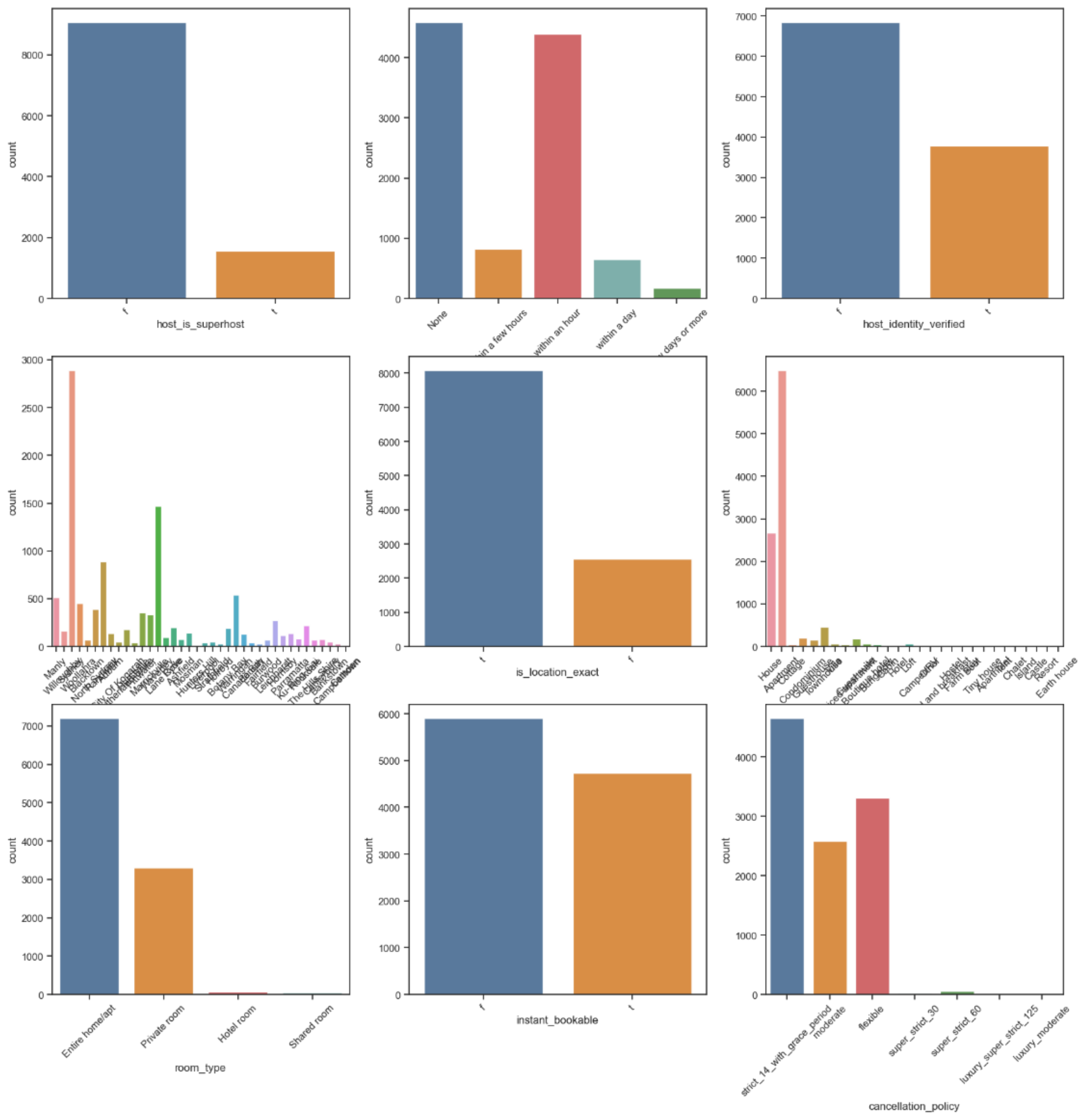


Figure B1 Count Plots of Categorical Variables

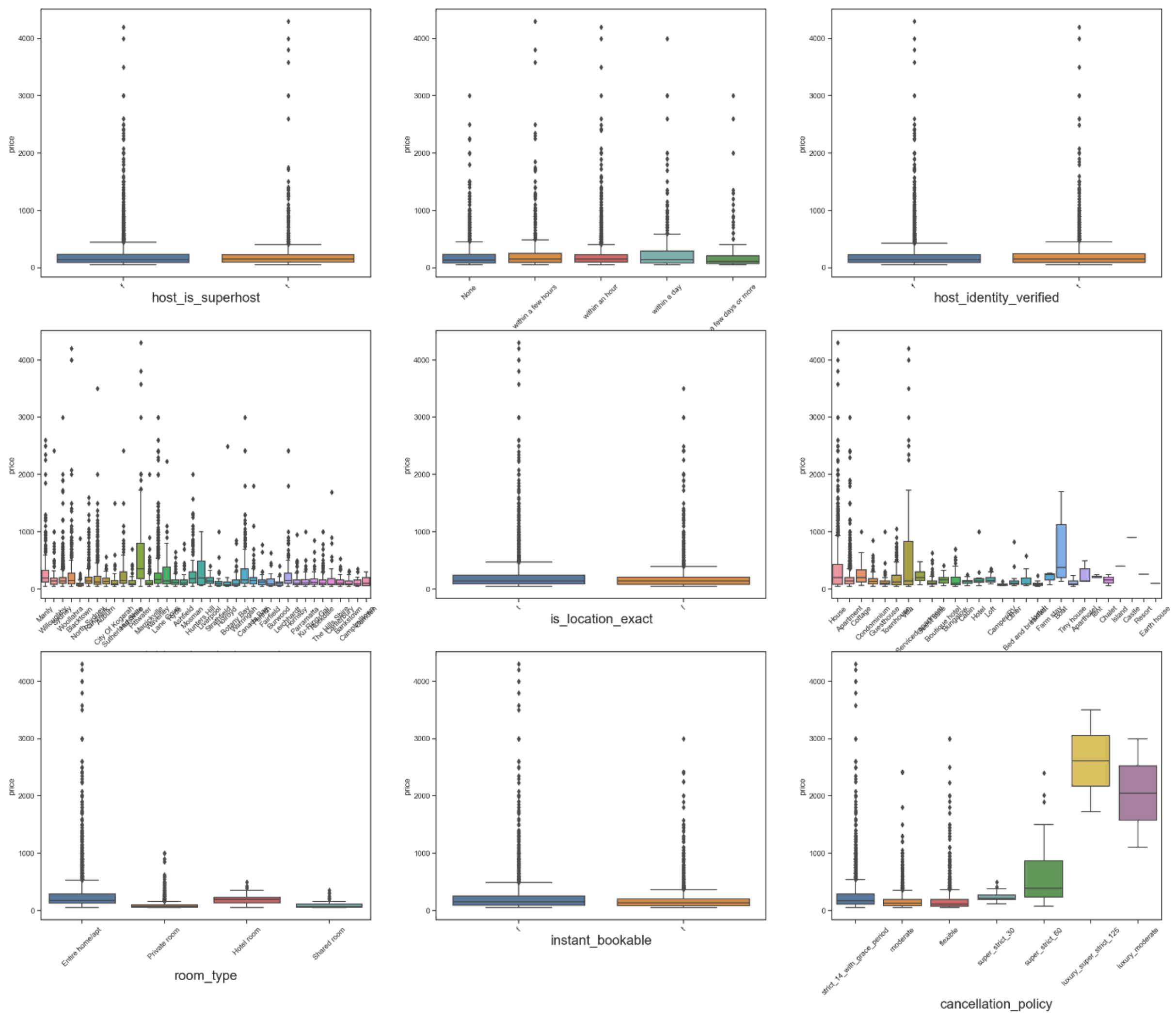


Figure B2 Boxplots of Categorical Variables

Appendix C – Host response rate & host acceptance rate

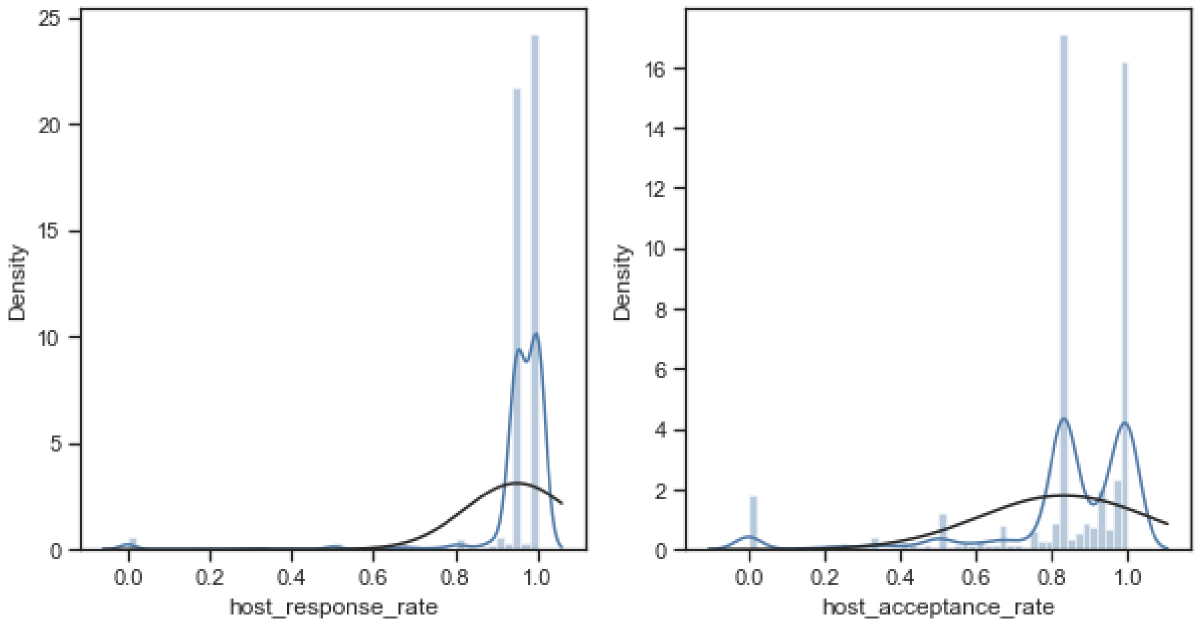


Figure C1 Distribution Plots of host_response_rate & host_acceptance_rate

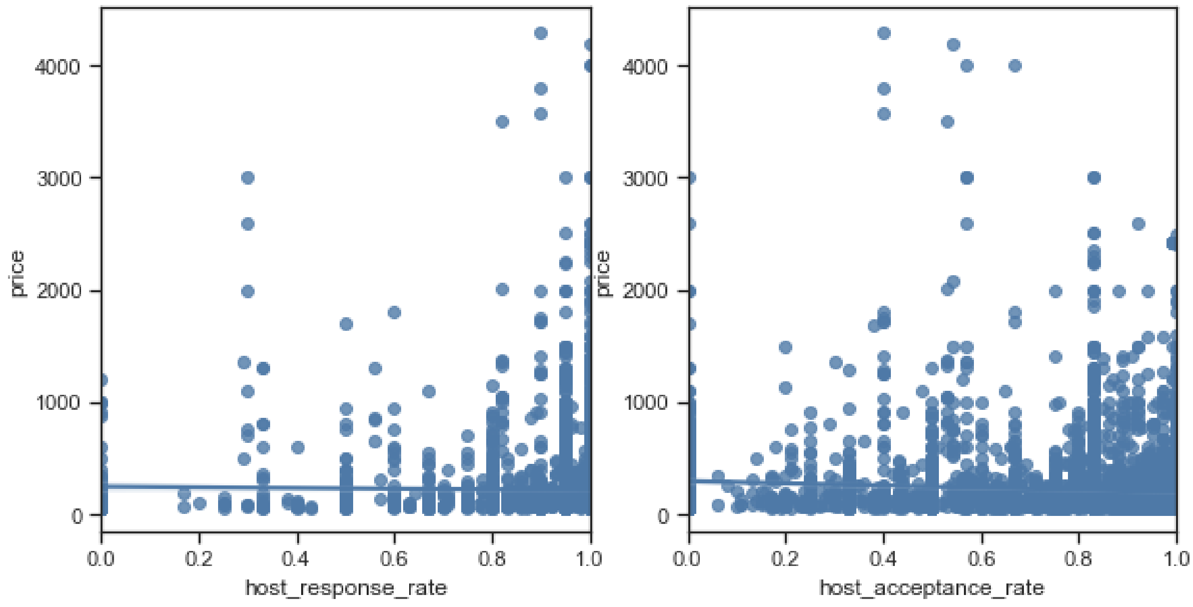


Figure C2 Regression Plots of host_response_rate & host_acceptance_rate

Appendix D – Host response time

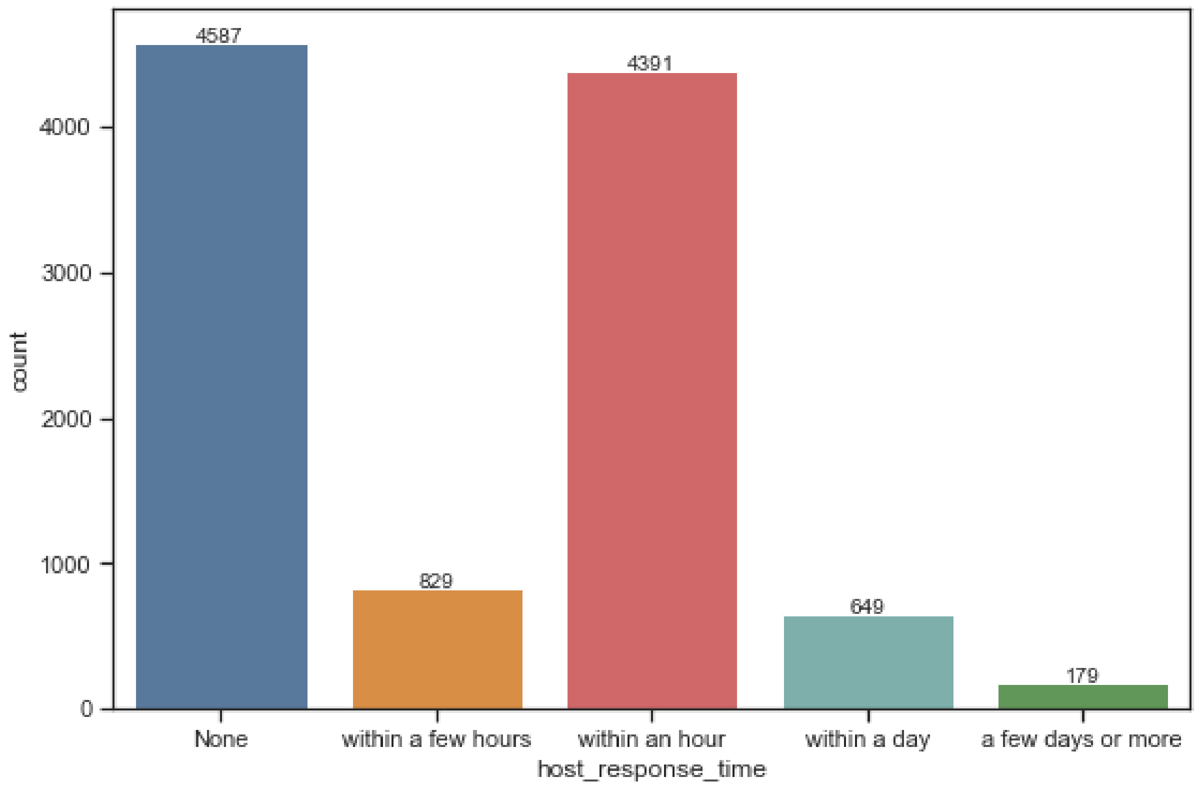


Figure D1 Count Plot of host_response_time

	None	A few days or more	Within a day	Within a few days	Within an hour
Count	4587.00	179.00	649.00	829.00	4391.00
Mean	198.9780	249.8380	266.9384	254.6888	208.7725
std	206.6150	400.3514	367.7830	370.6425	245.5139
Min	51.00	51.00	51.00	51.00	51.00
25%	85.00	72.50	85.00	89.00	100.00
50%	129.00	110.00	140.00	150.00	150.00
75%	231.00	205.00	290.00	250.00	220.00
Max	2999.00	2999.00	4000.00	4300.00	4200.00

Table D1 Descriptive Statistics of host_response_time

Appendix E – Neighborhood Cleansed

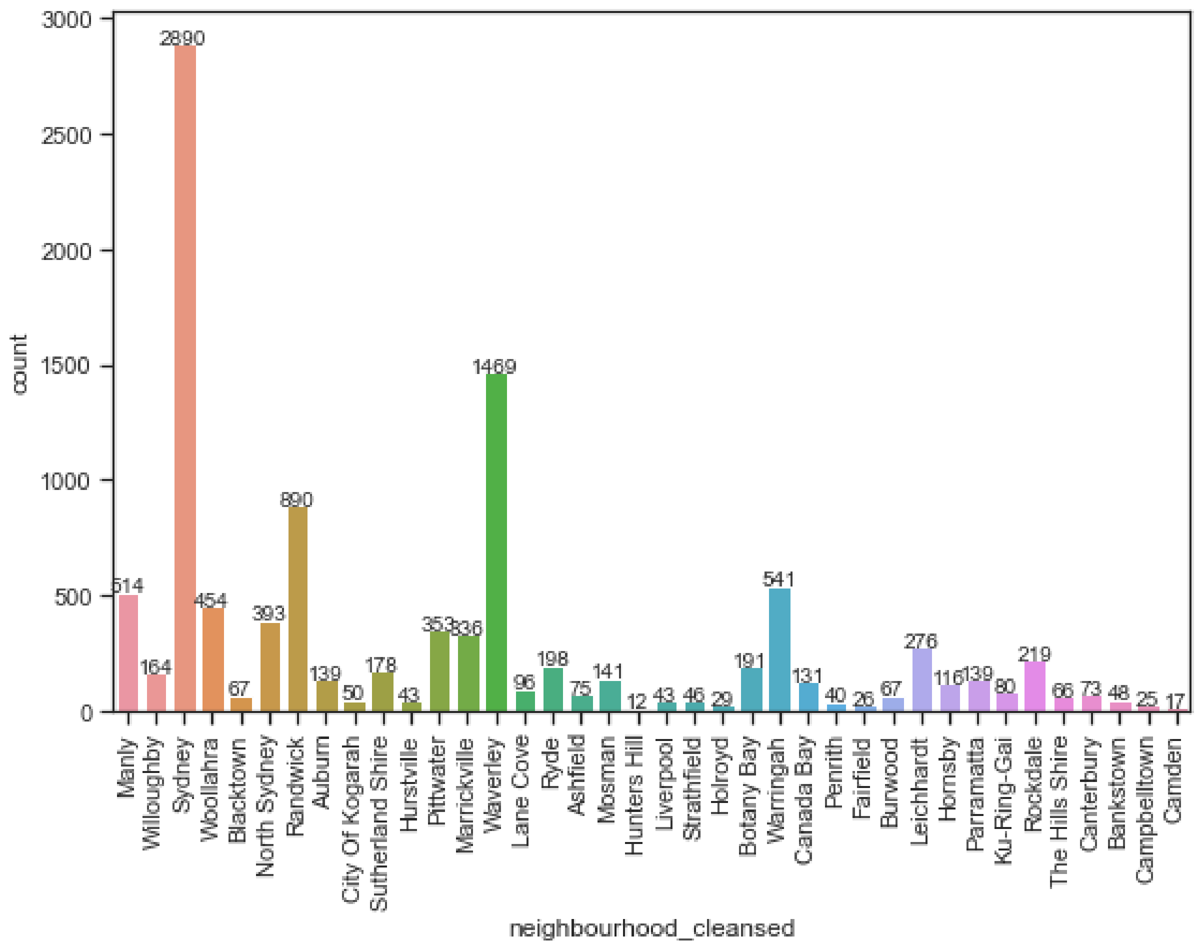


Figure E1 Count Plot of neighbourhood_cleansed

Appendix F – Property Type & Room Type & Cancellation Policy

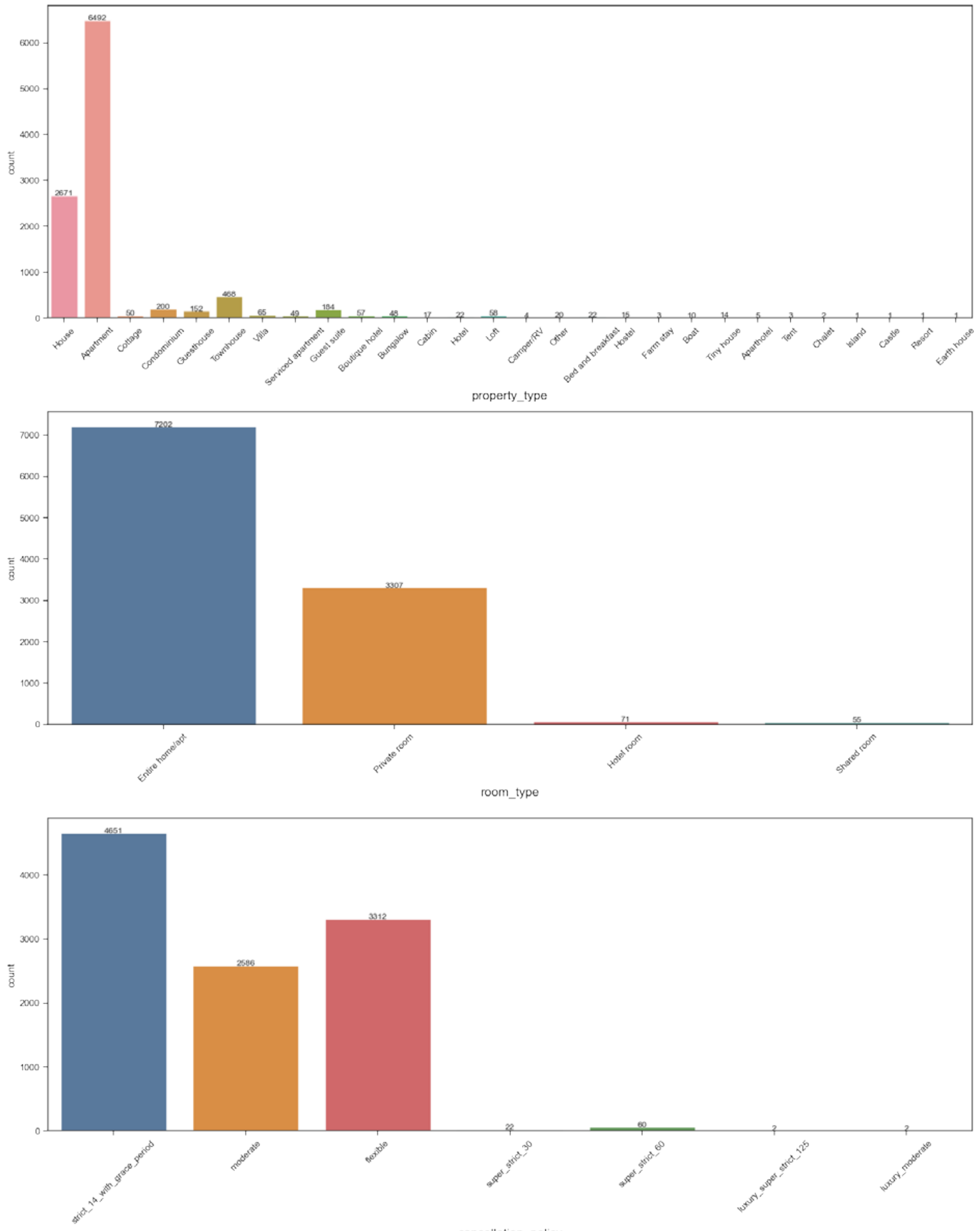


Figure F1 Count Plots of property_type, room_type & cancellation_policy

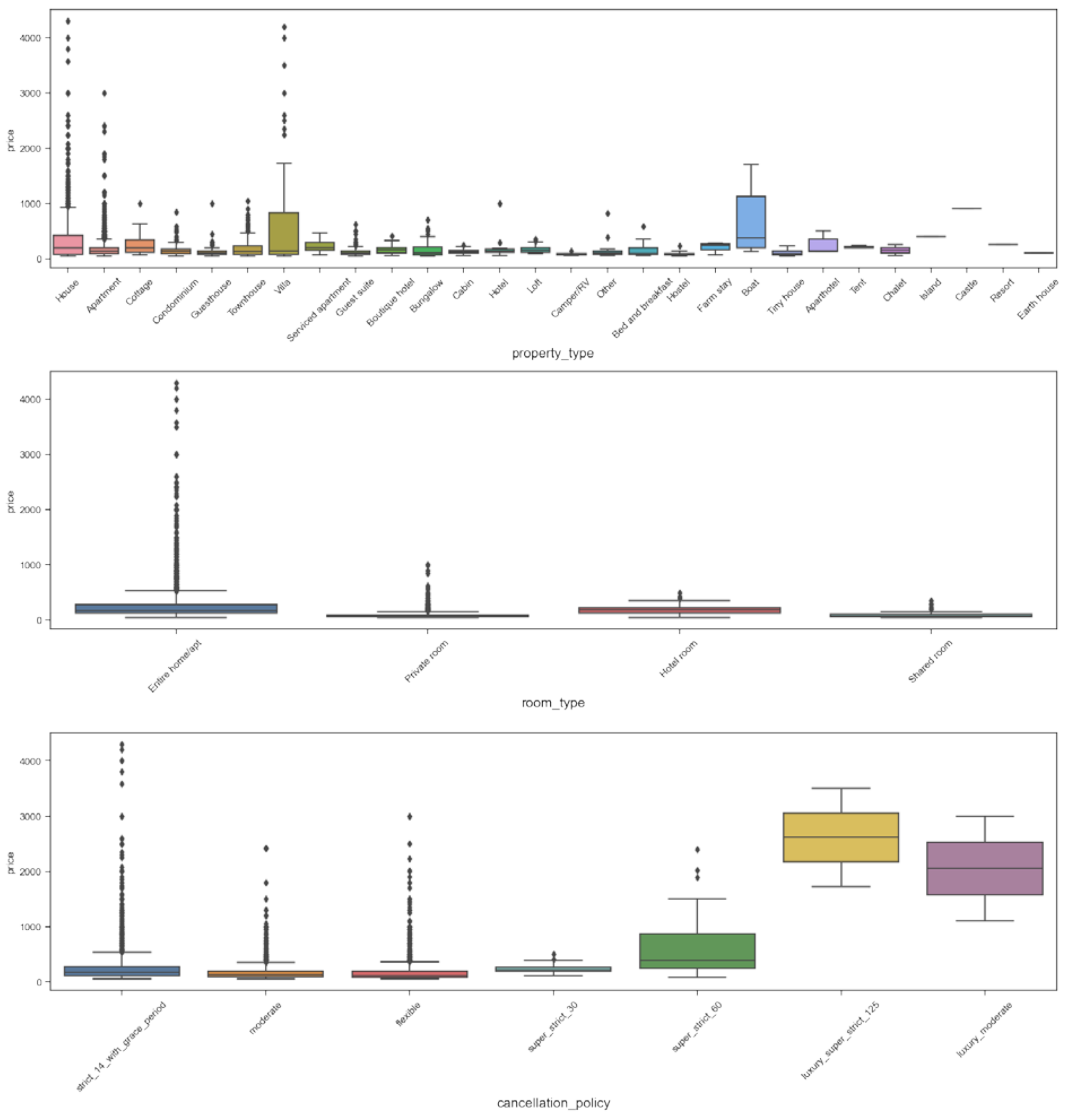


Figure F2 Boxplots of property_type, room_type & cancellation_policy

Appendix G – Bathrooms, Bedrooms & Beds

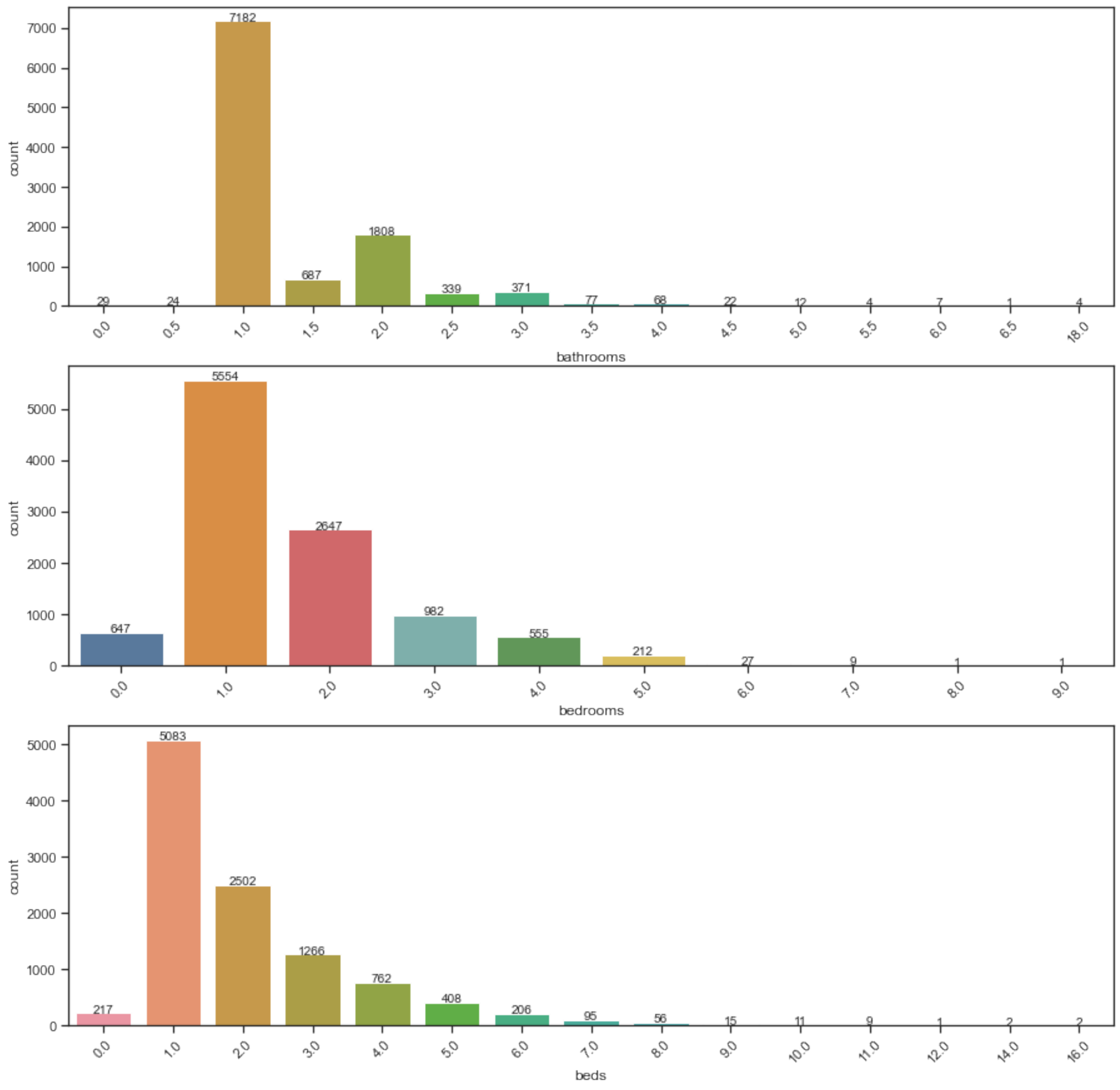


Figure G1 Count Plots of Bathrooms, Bedrooms & Beds

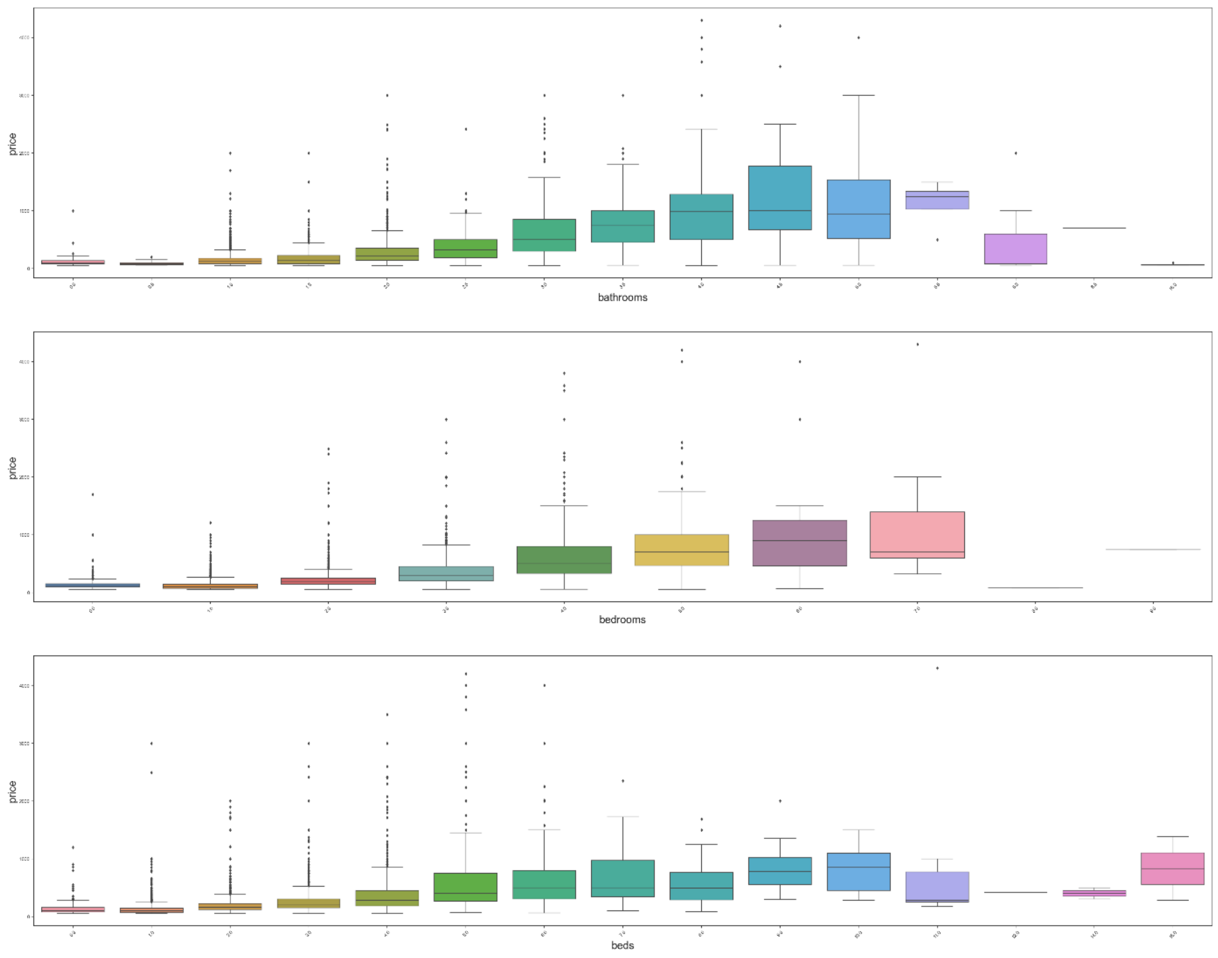


Figure G2 Boxplots of Bathrooms, Bedrooms & Beds

Appendix H - Security_deposit_perc, Cleaning_fee_perc & Extra_people_perc

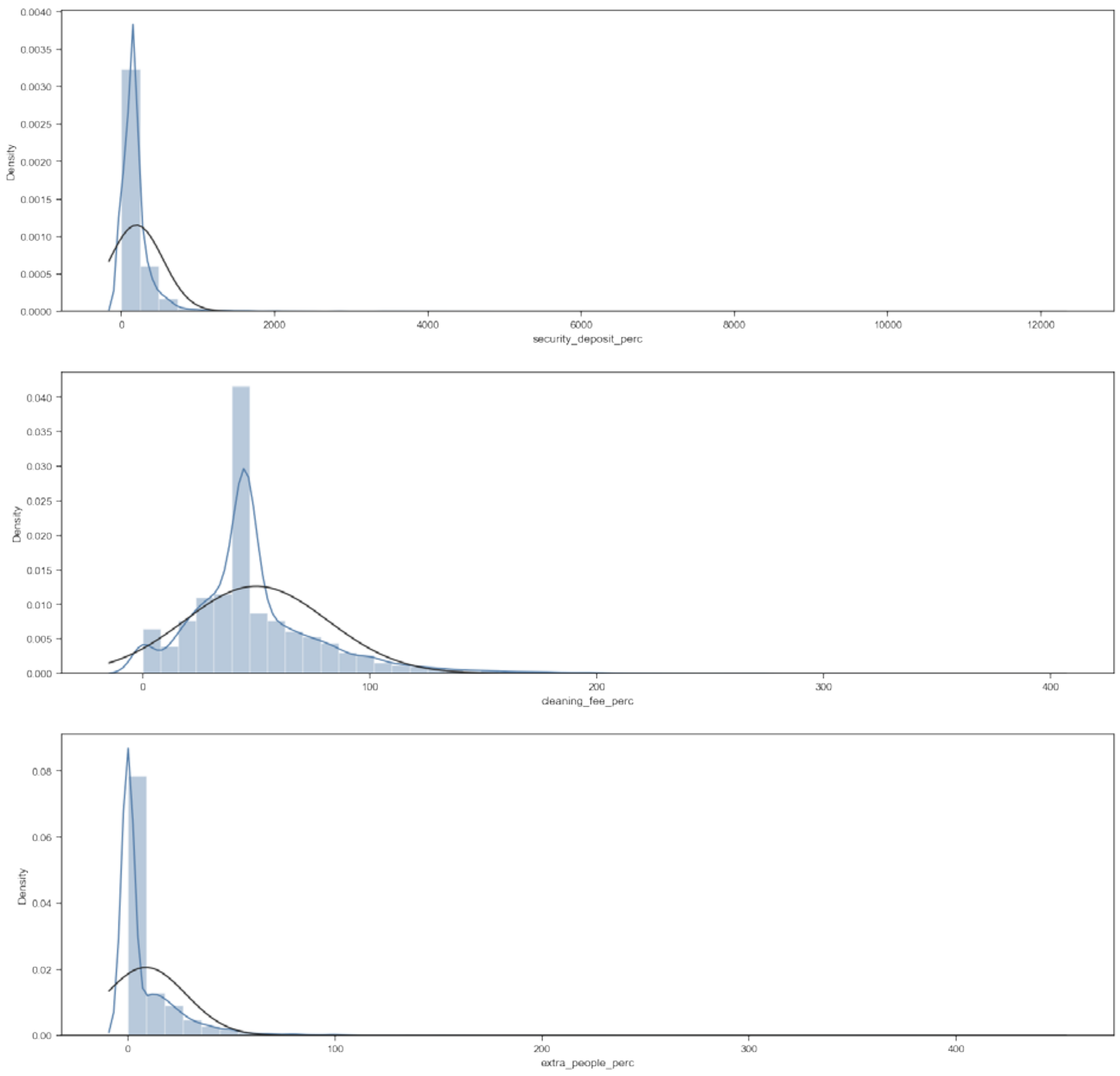


Figure H1 Distribution Plots of Security_deposit_perc, Cleaning_fee_perc & Extra_people_perc

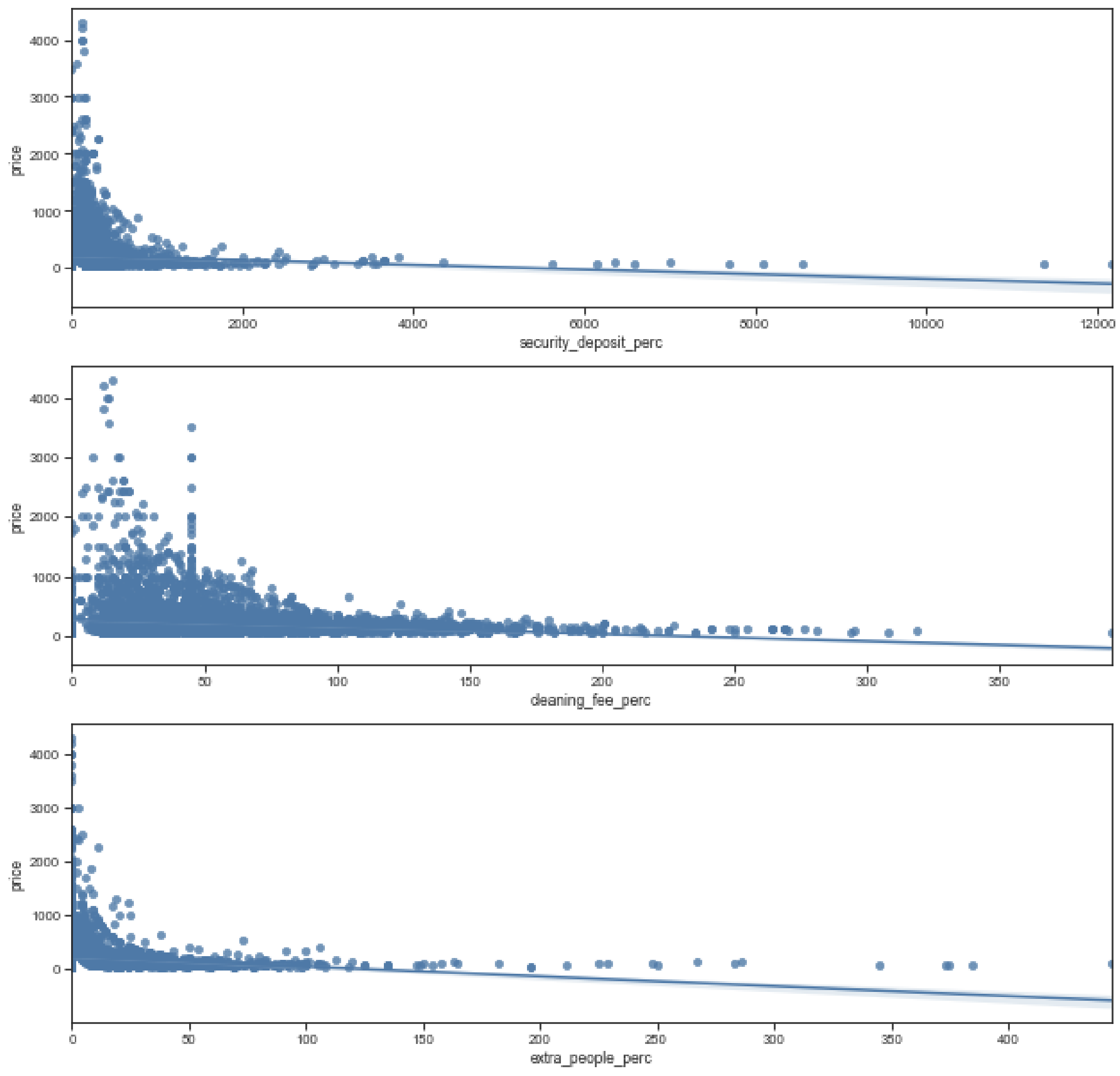


Figure H2 Regression Plots of Security_deposit_perc, Cleaning_fee_perc & Extra_people_perc

Appendix I – Guest included, minimum_nights_avg_ntm, maximum_nights_avg_ntm & availability_30

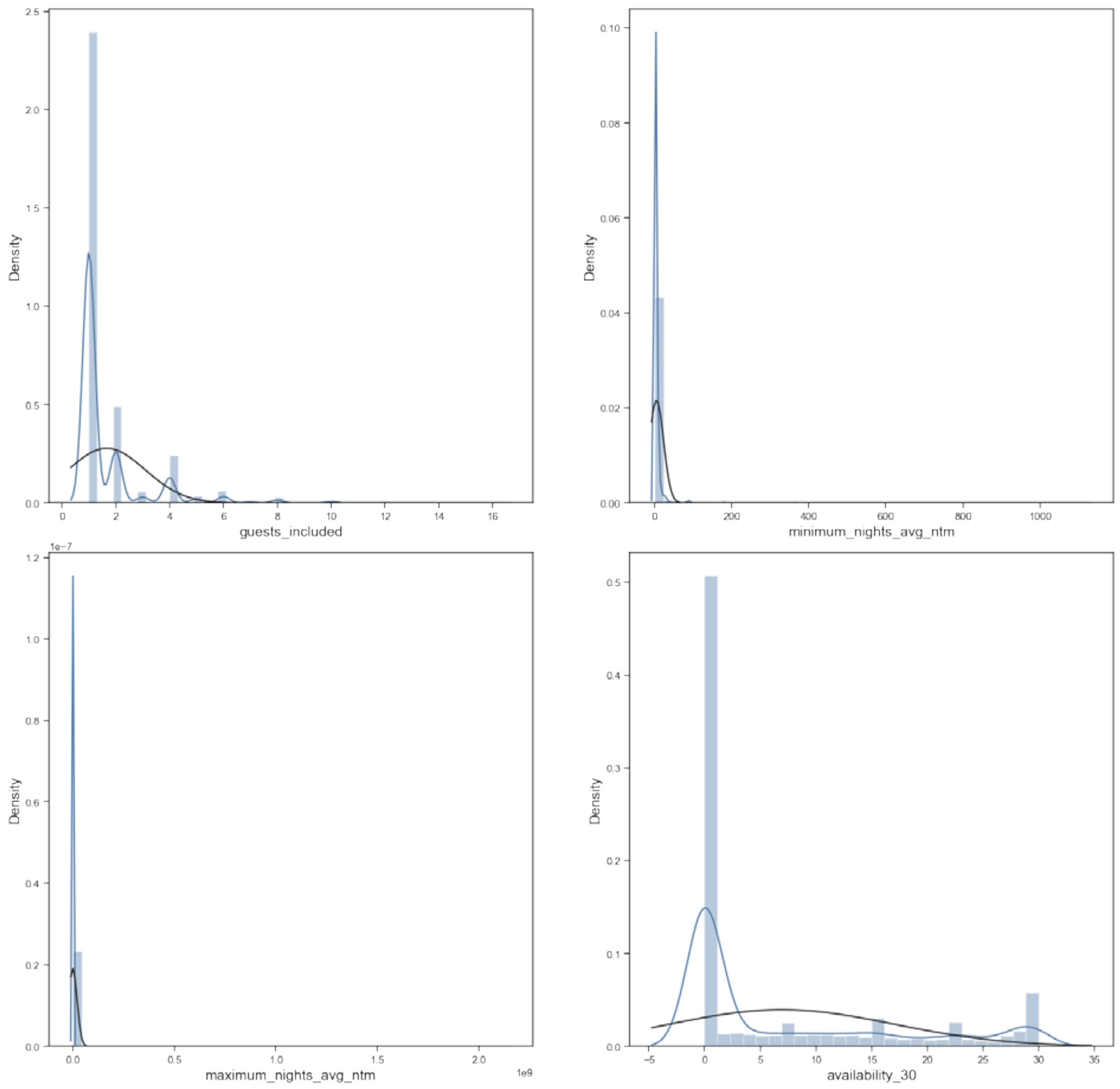


Figure I1 Distribution Plots of guests_included, minimum_nights_avg_ntm, maximum_nights_avg_ntm & availability_30

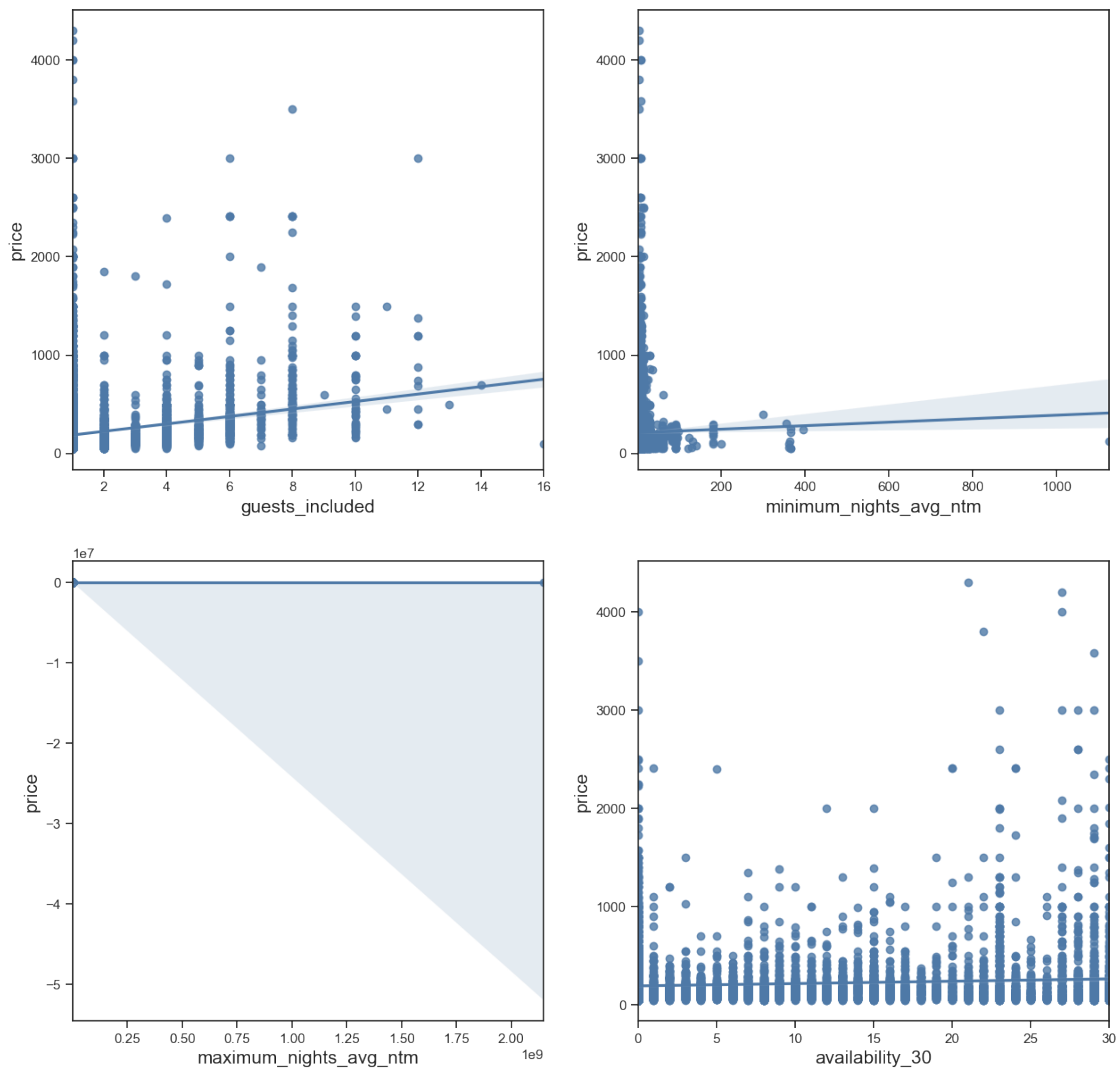


Figure I2 Regression Plots of guests_included, minimum_nights_avg_ntm, maximum_nights_avg_ntm & availability_30

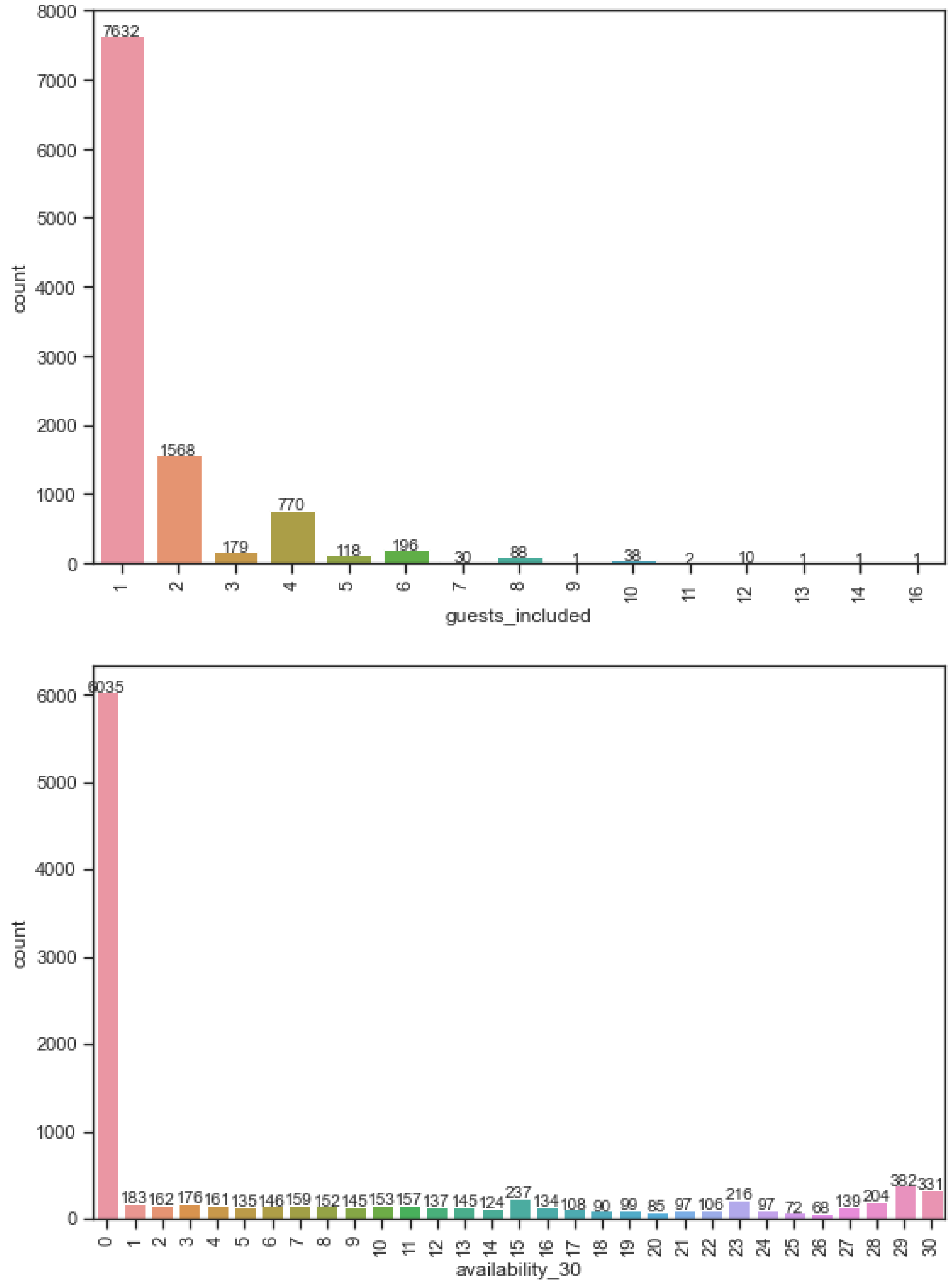


Figure I3 Count Plots of guests_included & availability_30

Appendix J – Number of reviews, reviews_per_month

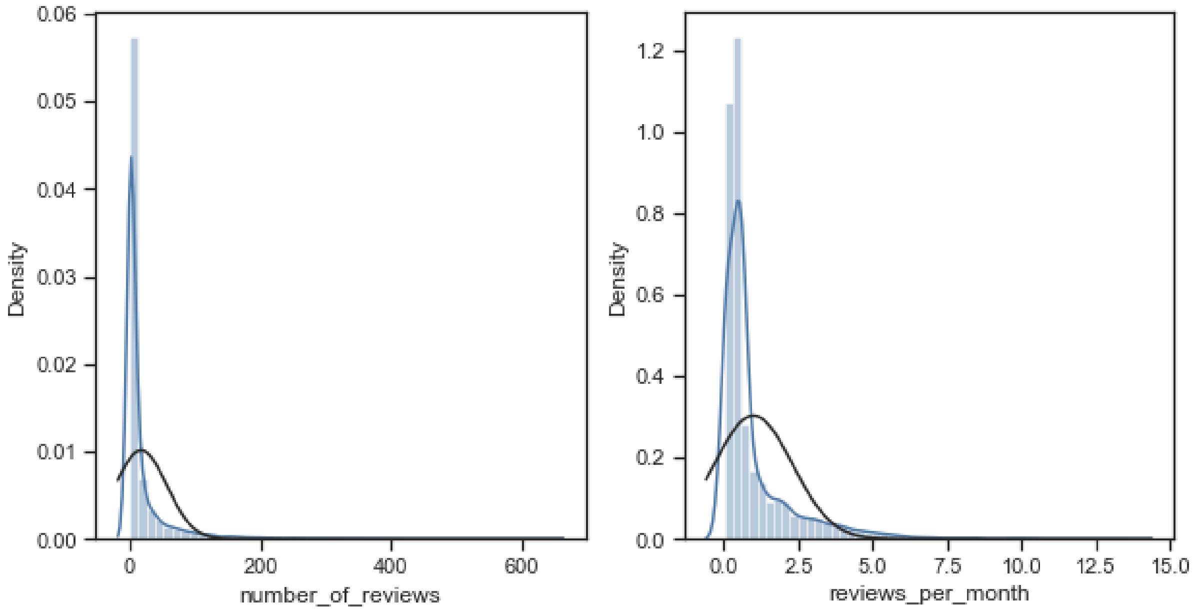


Figure J1 Distribution Plots of number_of_reviews & reviews_per_month

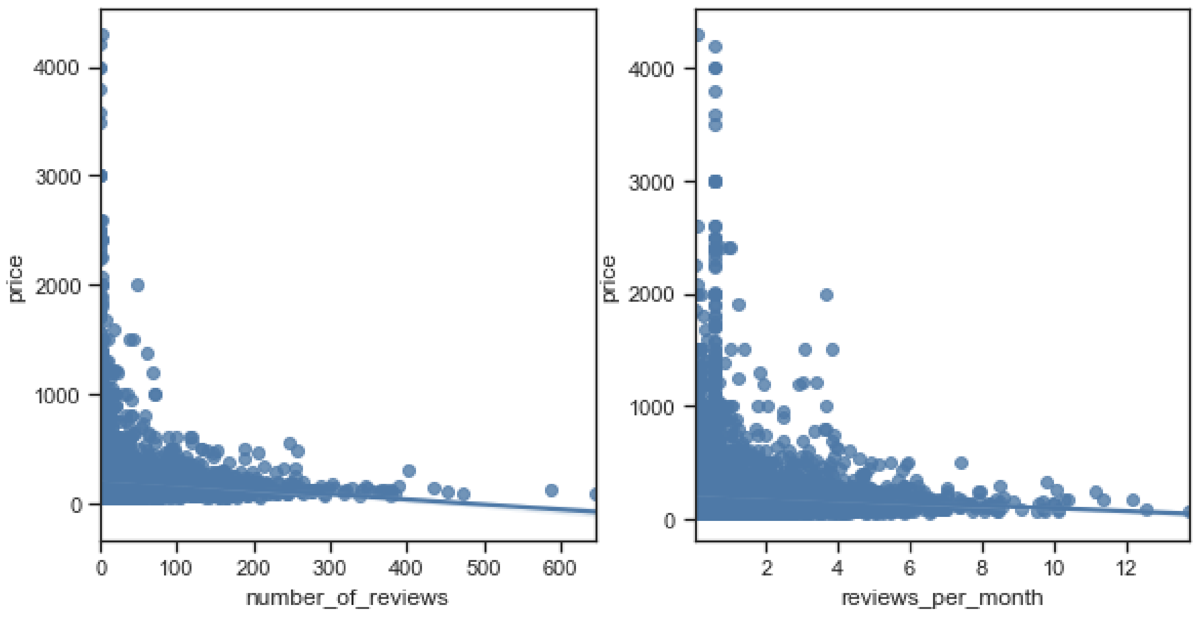


Figure J2 Regression Plots of number_of_reviews & reviews_per_month

Appendix K – Review Scores

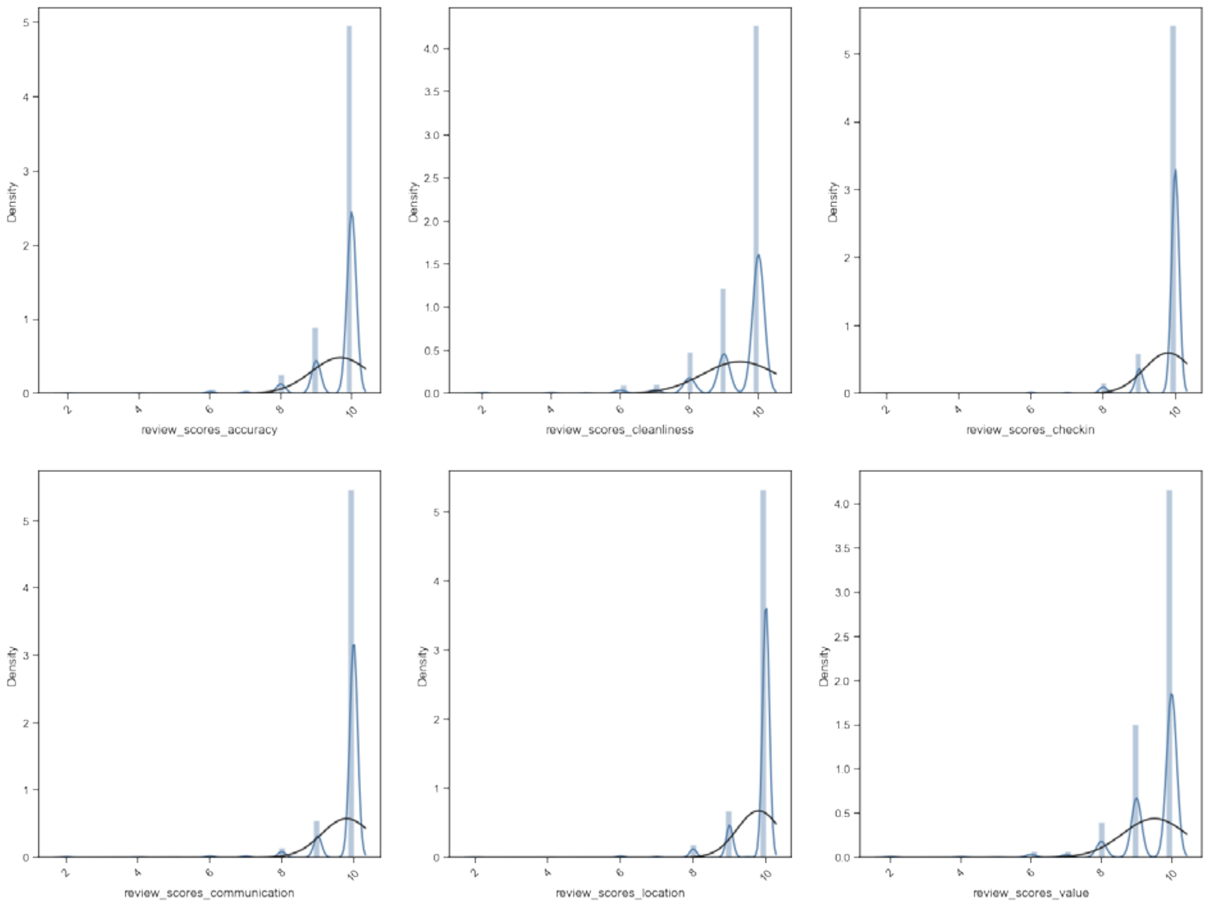


Figure K1 Distribution Plots of Review Scores

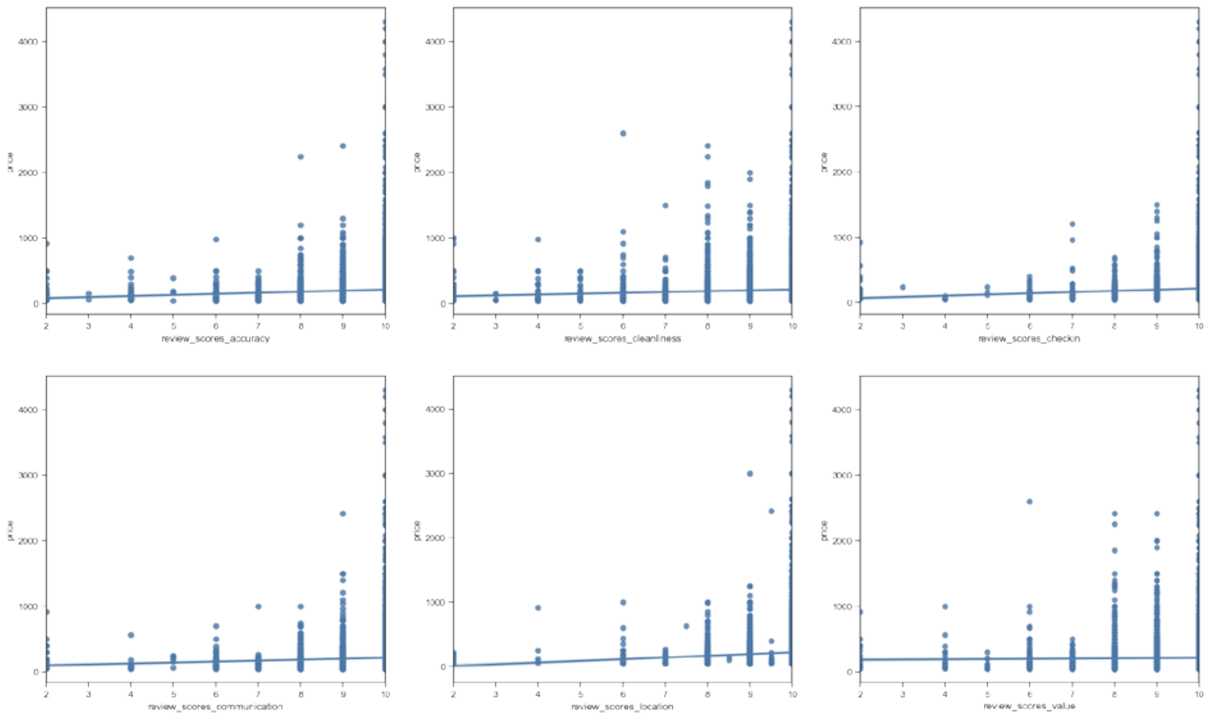


Figure K2 Regression Plots of Review Scores

Appendix L – calculated_host_listings_count_entire_homes & calculated_host_listings_count_private_roomes

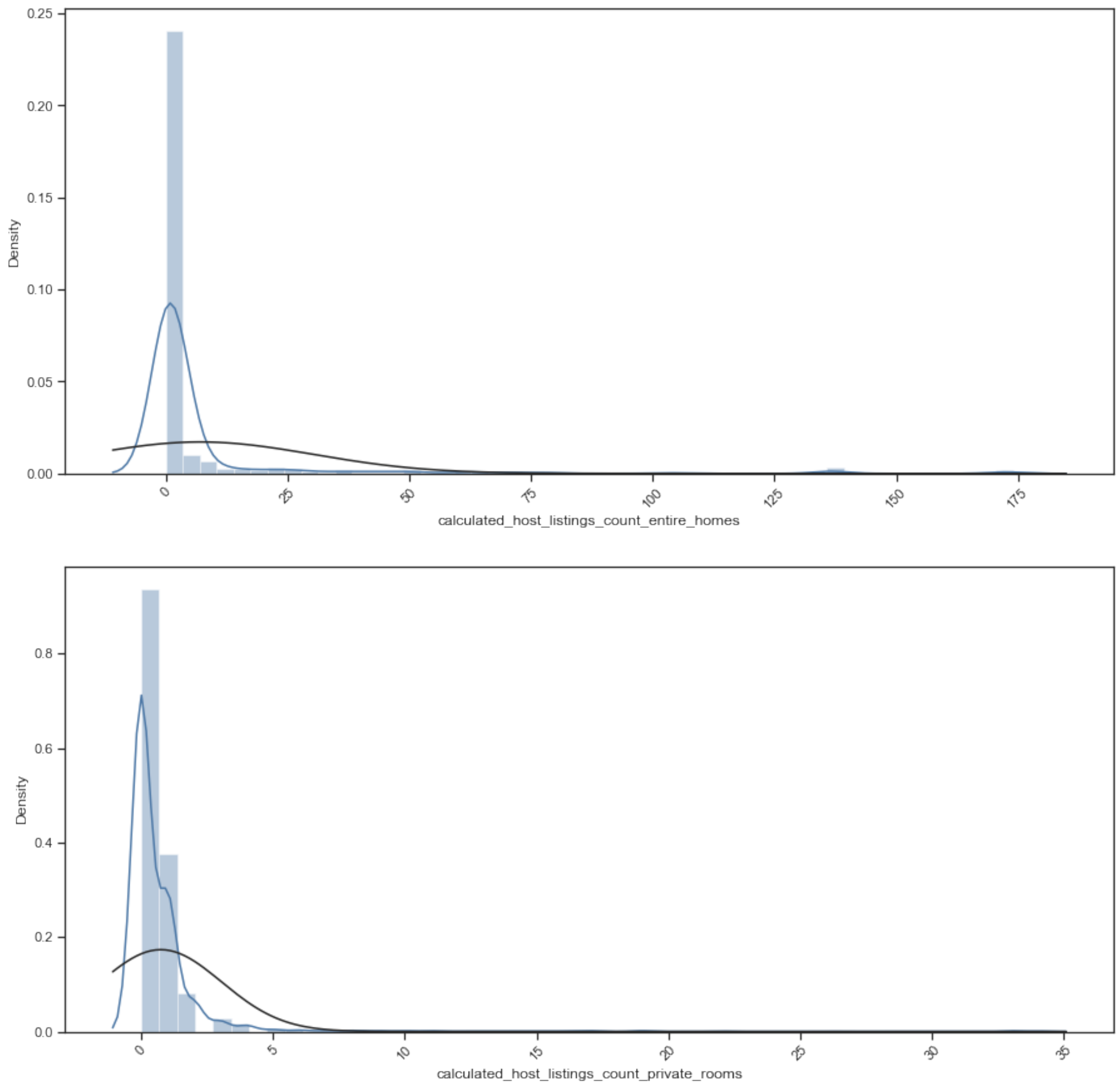


Figure L1 Distribution Plots of host_listings_count

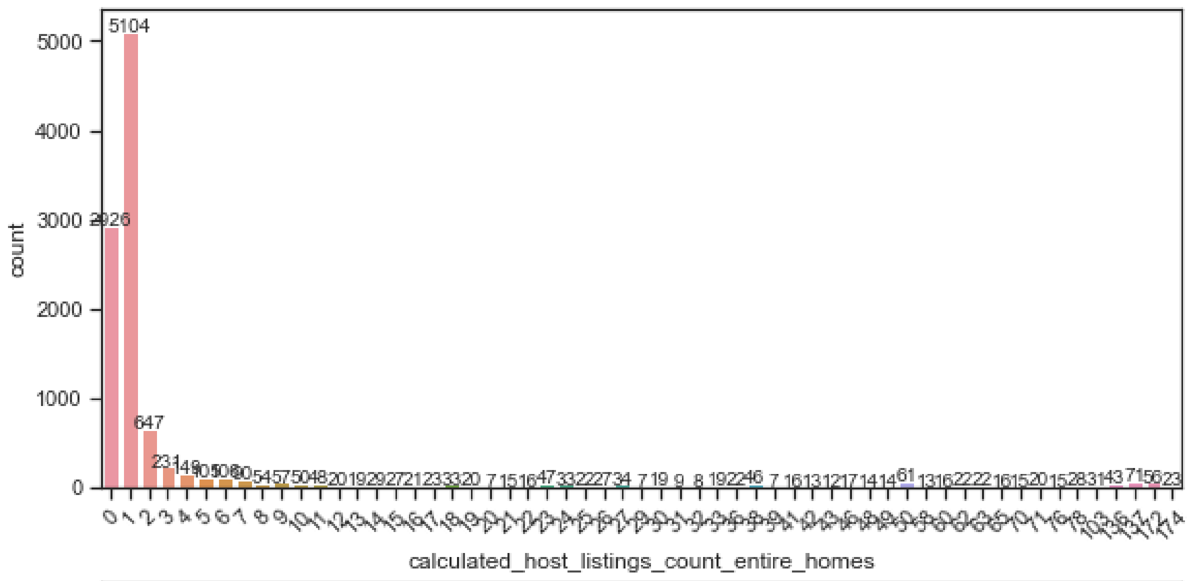
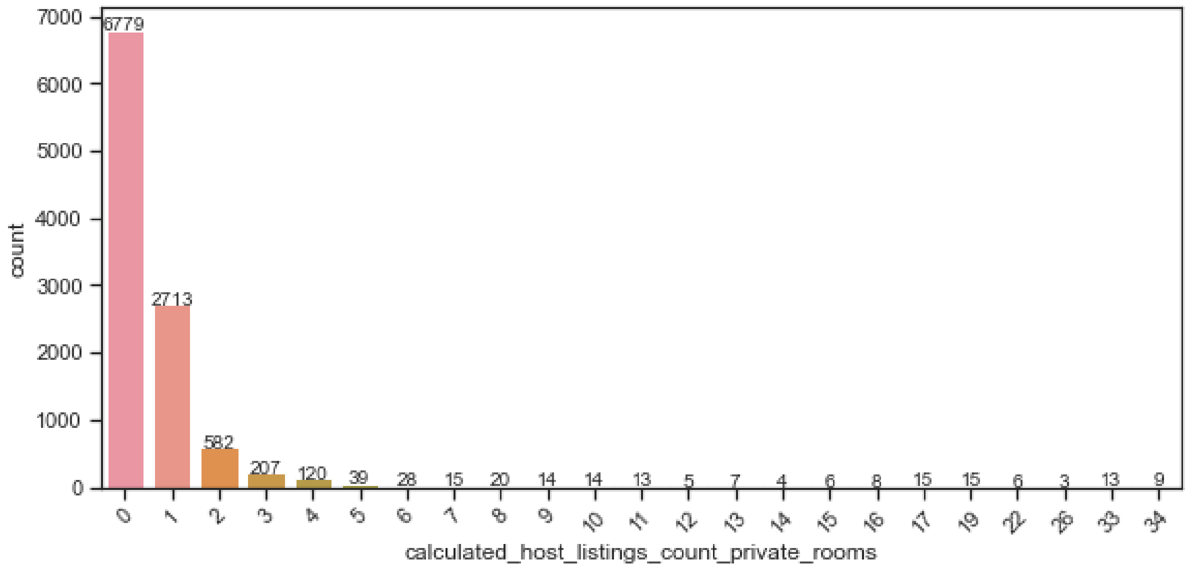
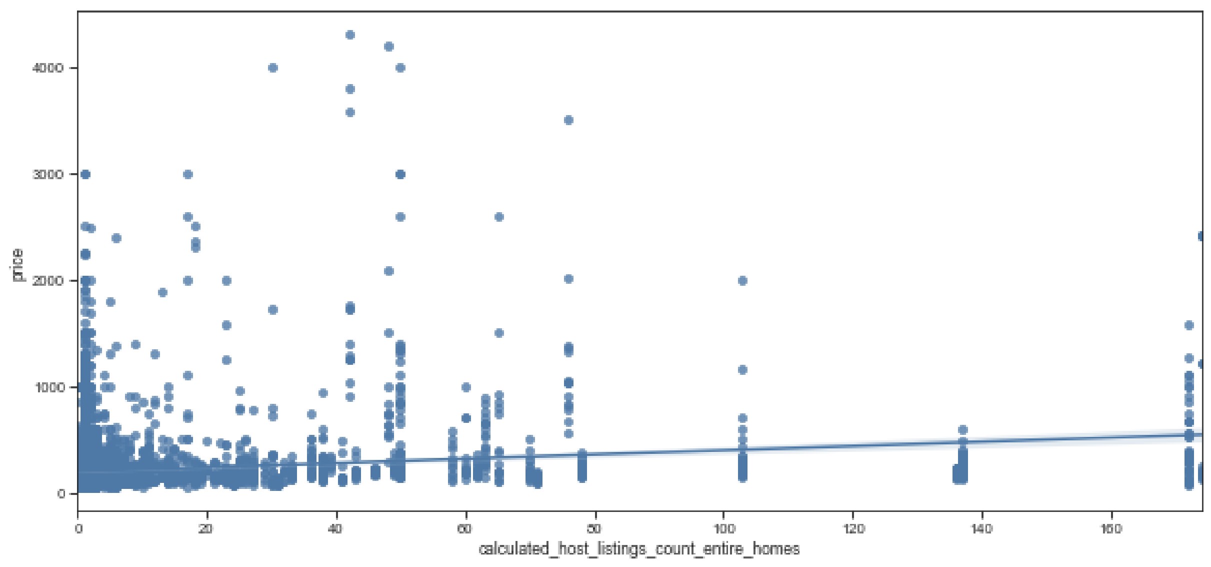



Figure L2 Count Plots of host_listings_count



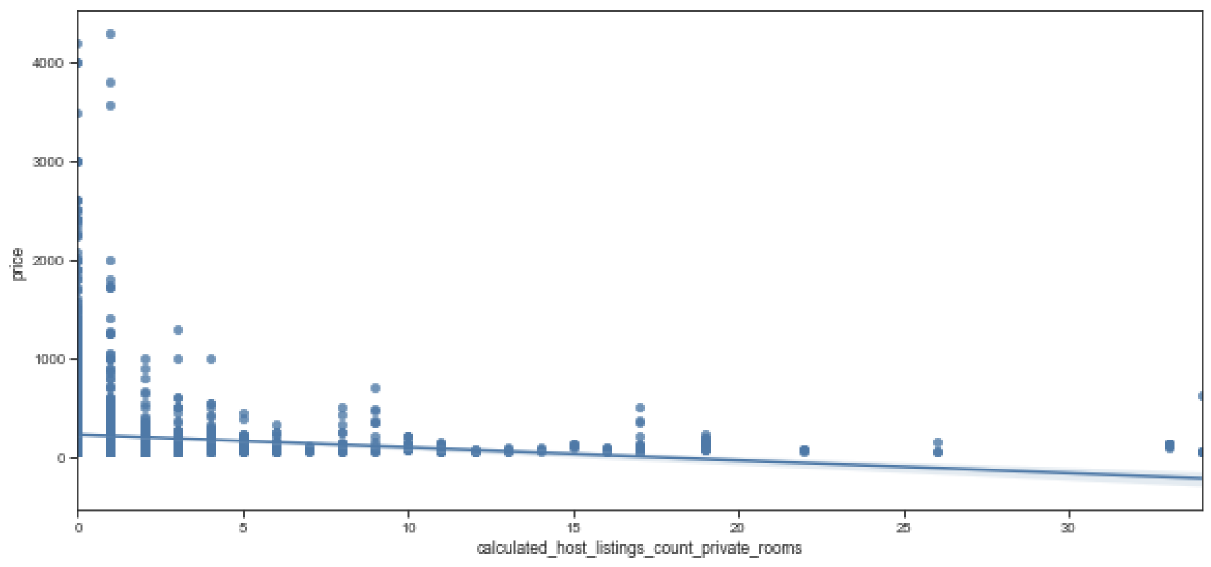
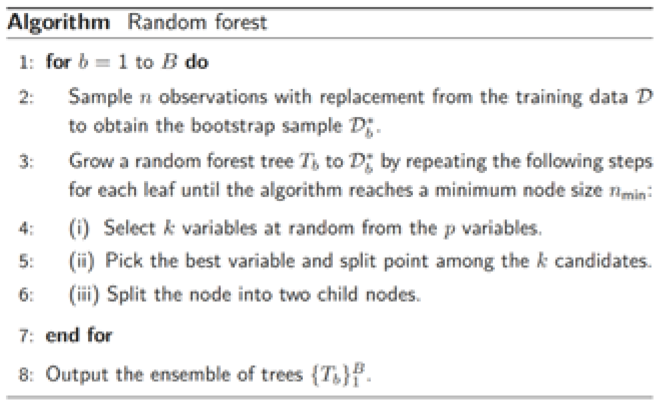


Figure L3 Regression Plots of host_listings_count

Appendix M – Algorithm of Random Forest



```
Algorithm  Random forest
1: for $b = 1$ to $B$ do
2:     Sample $n$ observations with replacement from the training data $\mathcal{D}$
       to obtain the bootstrap sample $\mathcal{D}_b^*$.
3:     Grow a random forest tree $T_b$ to $\mathcal{D}_b^*$ by repeating the following steps
       for each leaf until the algorithm reaches a minimum node size $n_{\text{min}}$:
4:     (i) Select $k$ variables at random from the $p$ variables.
5:     (ii) Pick the best variable and split point among the $k$ candidates.
6:     (iii) Split the node into two child nodes.
7: end for
8: Output the ensemble of trees $\{T_b\}_1^B$.
```

Figure M1 Algorithm of Random Forest

Appendix N – Variance Importance

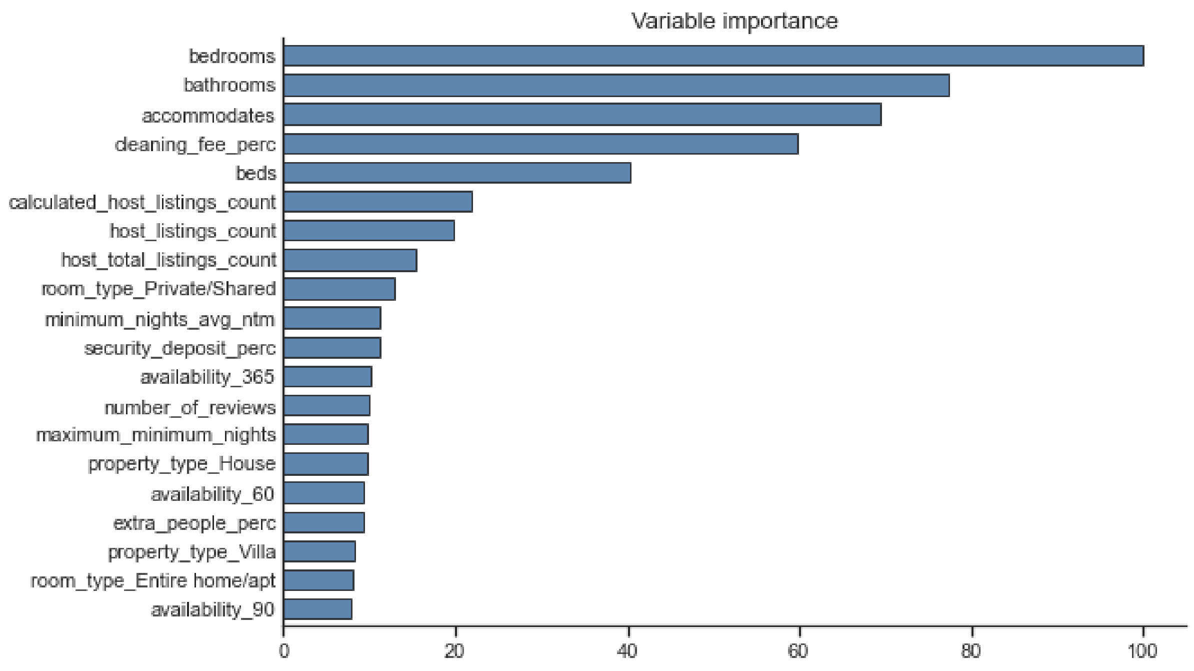


Figure L1 Variable Importance of Ridge

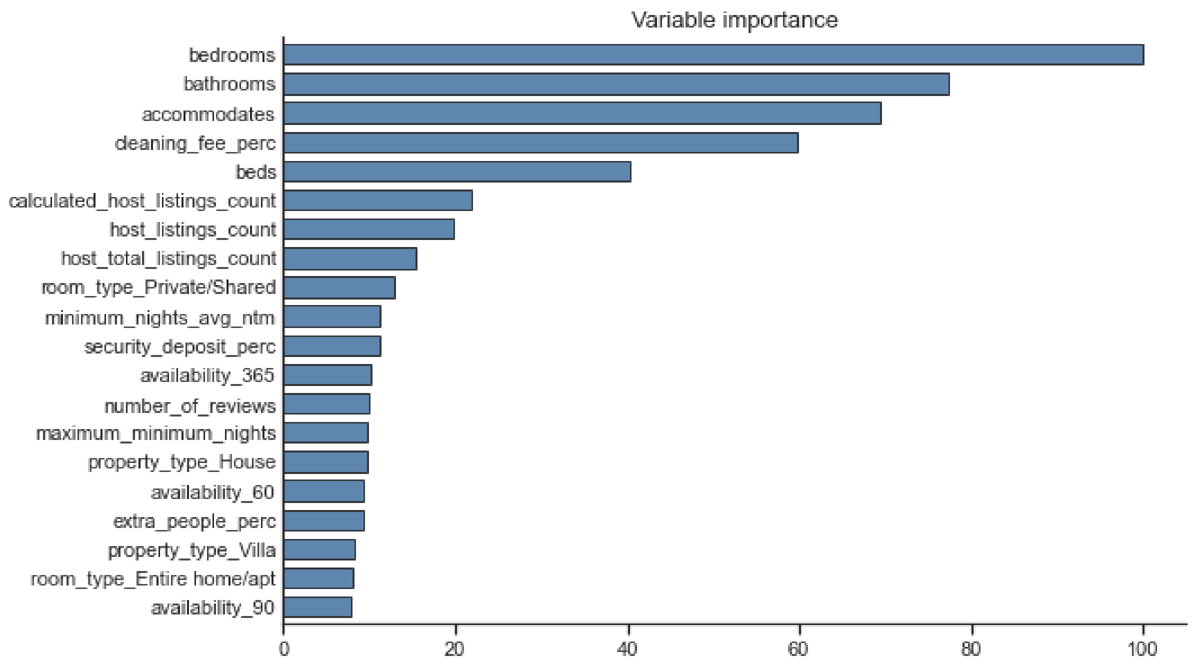


Figure L2 Variable Importance of Random Forest (without hyperparameter tuning)

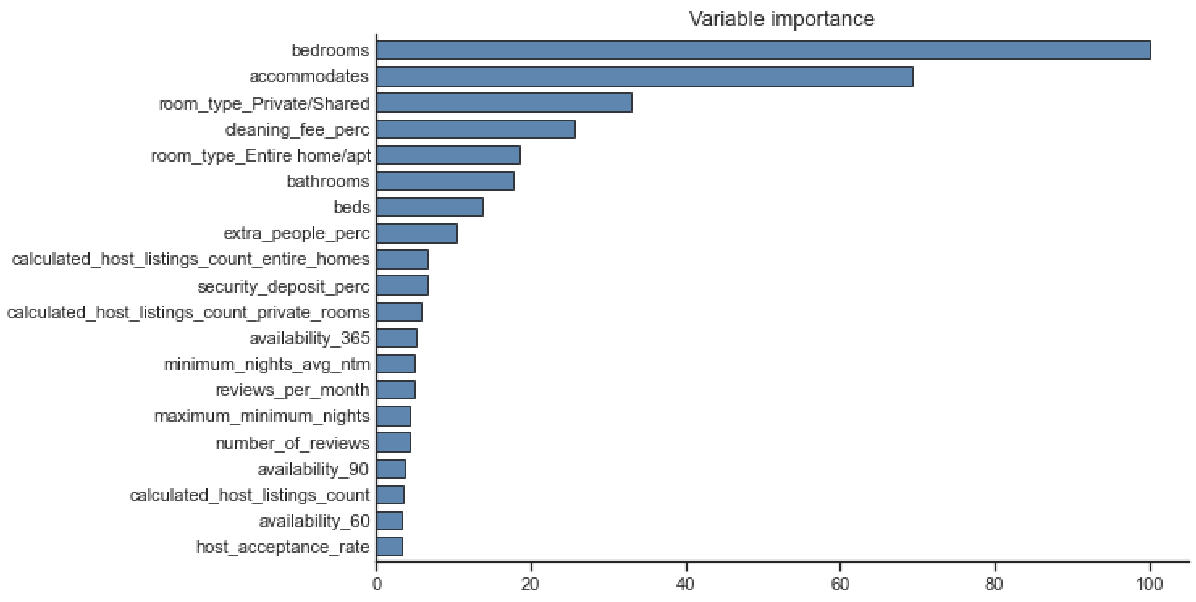


Figure L3 Variable Importance of Random Forest (after hyperparameter tuning)

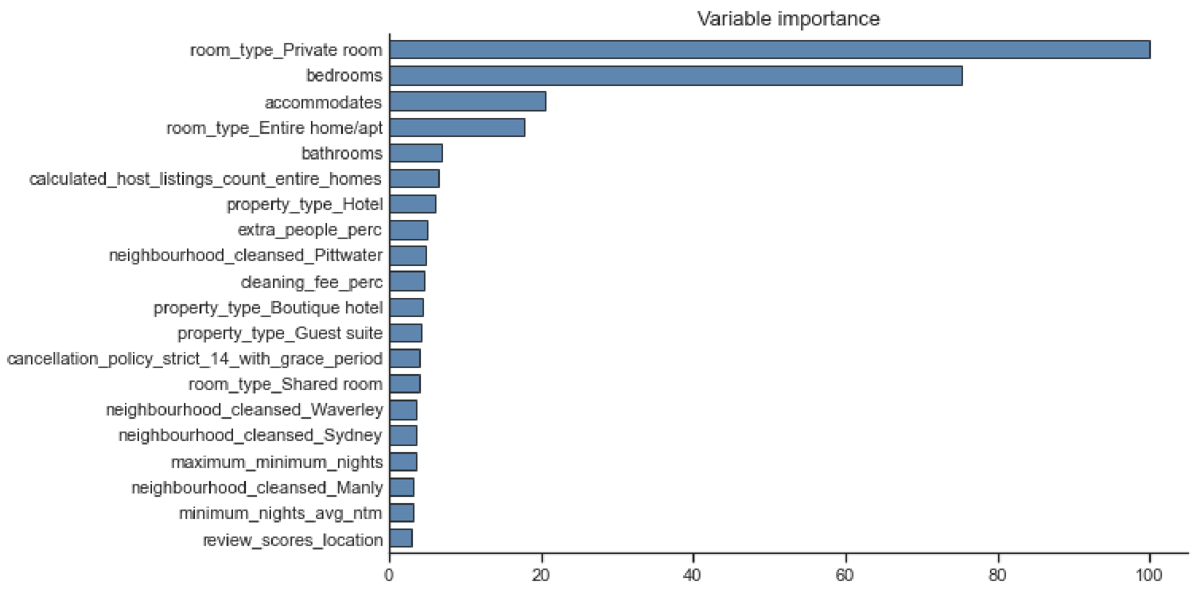


Figure L4 Variable Importance of XGBoost (without hyperparameter tuning)

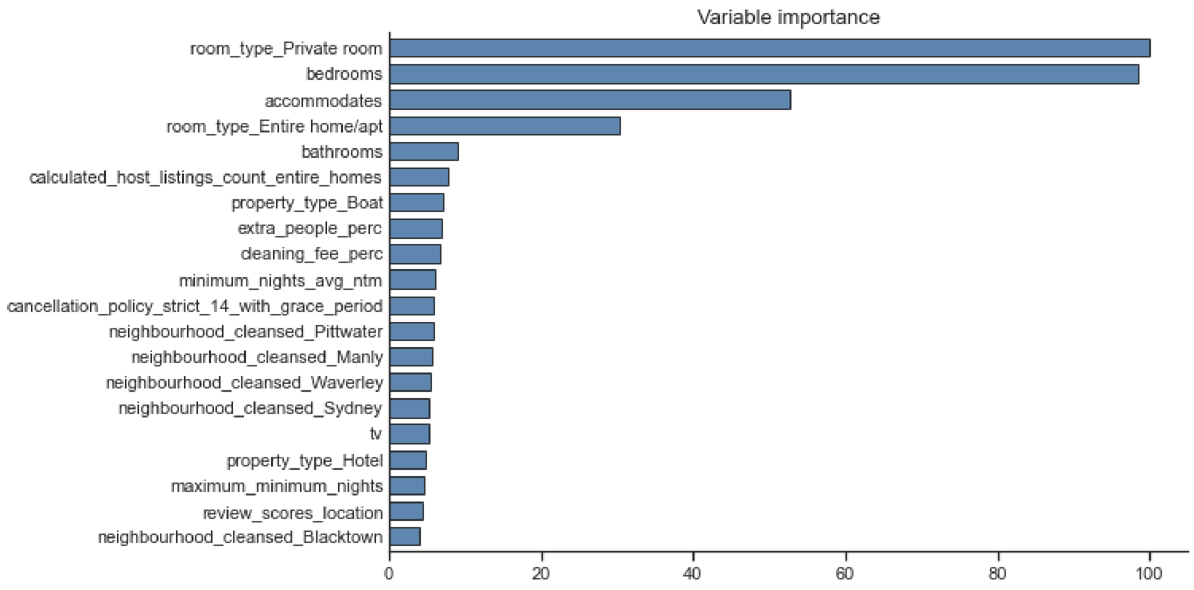


Figure L5 Variable Importance of XGBoost (after hyperparameter tuning)

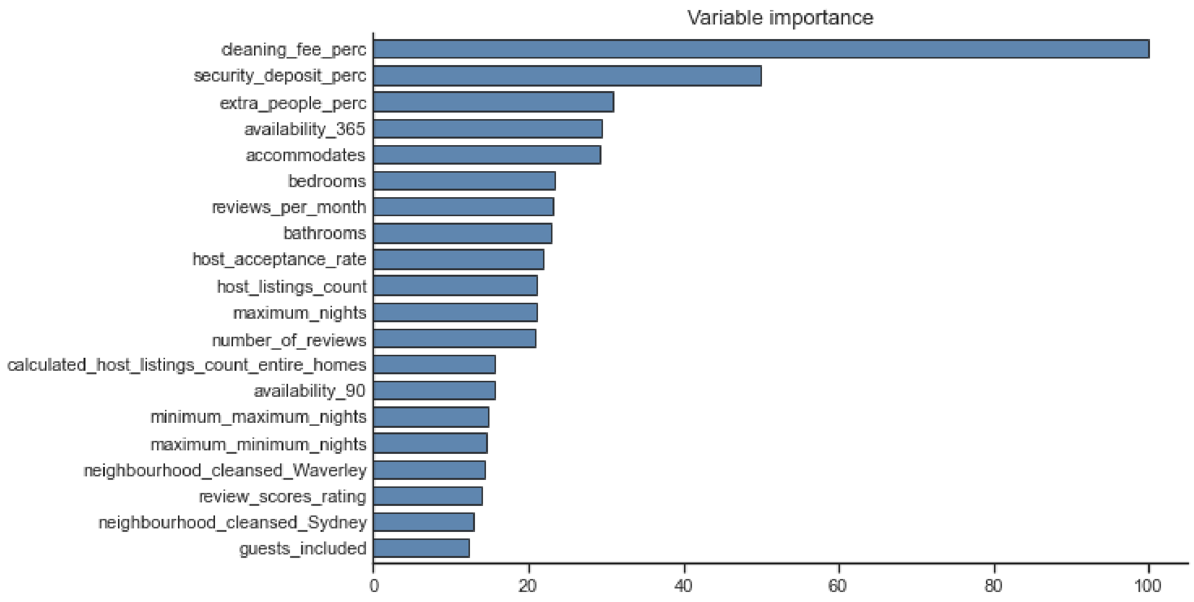


Figure L6 Variable Importance of LightGBM (without hyperparameter tuning)

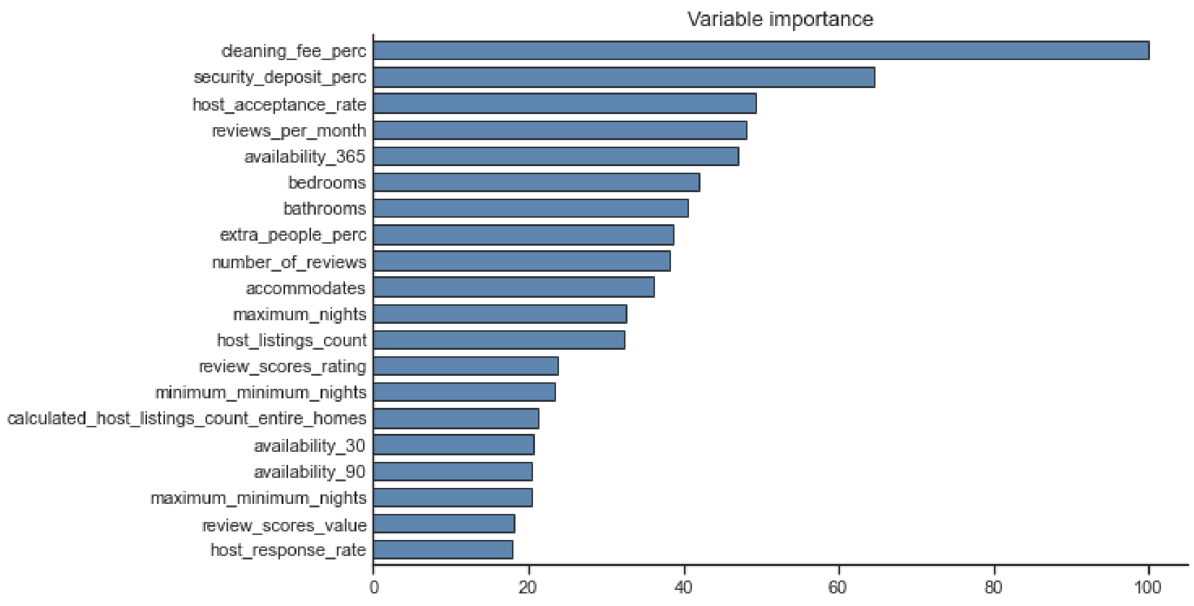


Figure L7 Variable Importance of LightGBM (after hyperparameter tuning)