Airbnb

Pricing Analytics

480140618 480150525 490518478

500225505 510170981

Group 42

Contents

I. Introduction and problem formulation

Airbnb is a platform that runs an online marketplace to provide short term travel rentals. This marketplace facilitates renters, and helps hosts, property managers, or real estate investors make profits from it.

This project is to address two main problems for our clients.

The first problem is to predict the daily price of Airbnb rentals. We will use the data obtained from Airbnb to train different regression models and provide our clients with the most predictive model by comparing the RMSLE of the validation set. This best model will help customers predict their revenue through rentals.

The second problem is to analyze the "best host" model. Some hosts can make more revenues through Airbnb. Without considering the cost, we will classify the real estates based on their unit price. Our goal is to use the XGBoost model to analyze the characteristics of the two classifications and provide our business insights on the 'best hosts' to help our clients maximize their revenues.

II. Data understanding

i. Deal with missing value

As preparation, firstly, we analyze the patterns of missing values. The bar chart below shows the features with missing values and how many values are missing in different features. Higher bar indicates a lower proportion of missing value.

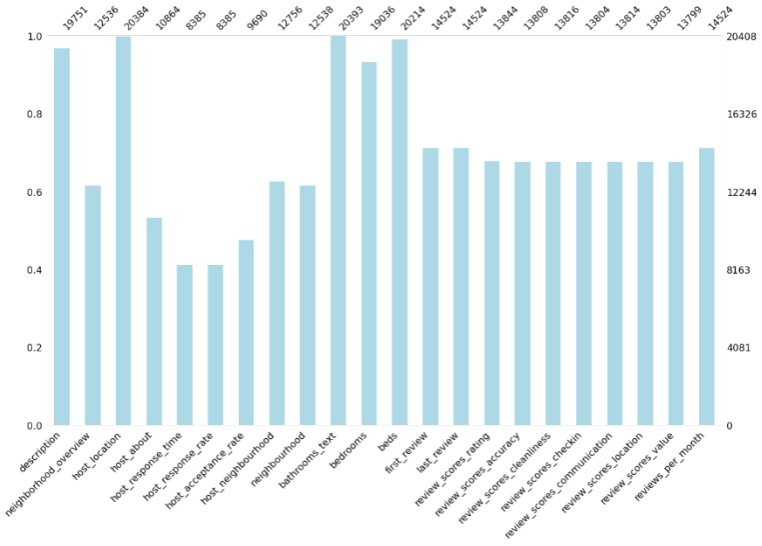

Figure 1.1 Patterns of missing values

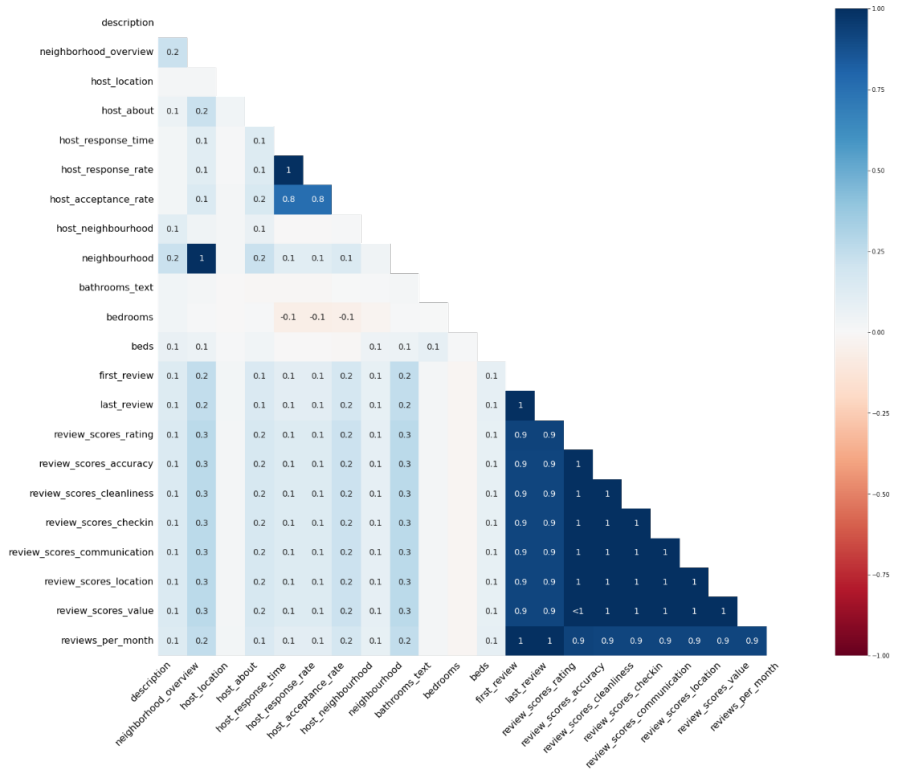

Figure 1.2 Heatmap for missing values

To optimize our model, we set the missing value as zero. And we add some new columns with suffix *'_isna'* to describe the information of missing value. Based on *Figure 1.2*, if some features' missing values only exist in the same observations, we will create one *'_isna'* feature as a representative.

ii. Exploratory Data Analysis (EDA)

We categorize all the features into 8 groups according to their practical meanings. The analysis results are listed:

a) Target variable: price

At the beginning we would like to analyze the pattern of target variable '*price*'.

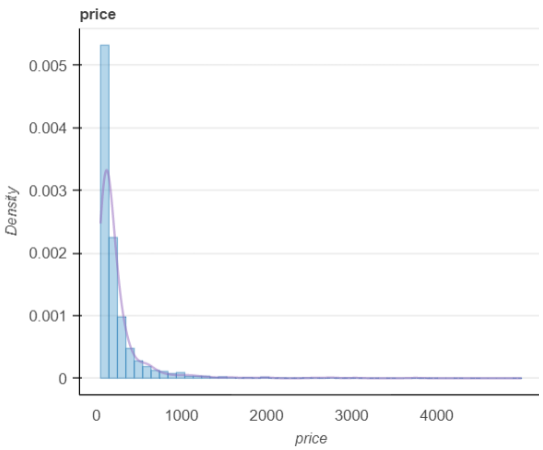

Figure 2.1 price distribution

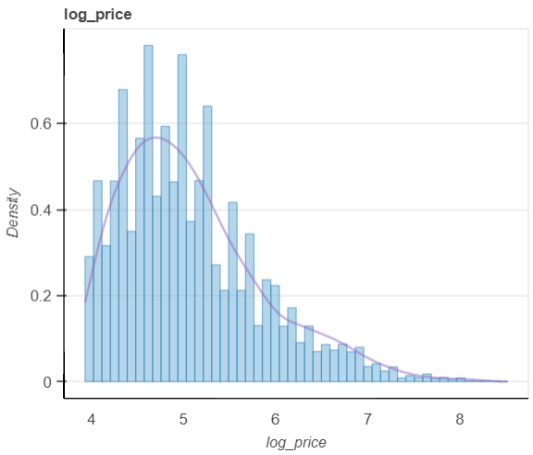

Figure 2.2 log-price distribution

The plots above show the distribution of price is right-skewed. We implement a logarithm transform to price.

b) Features about basic information

We include *'name'* and *'description'* in this category. These two are both subjective features posted by the host to attract audience.

After checking the text content, we find that the information in *'name'* and *'description'* overlap with other features, so we do not analyze the content. But the hypothesis test shows that listings with descriptions have higher average price than the ones without description. We create a dummy variable of *'description_isna'* to describe this feature.

c) Features about host

We include *'host_name'*, *'host_response_time'*, *'host_response_rate'*, *'host_is_superhost'*, *'host_identity_verified'*, etc. in this category. These features describe the characteristics of host.

The first finding is that some features could reflect the degree of activeness of hosts, like *'host_response_rate'* and *'host_acceptance_rate'*. As shown in *Figure 3.1*, hosts with high response rate also have high acceptance rate, indicating high activeness. Based on *Figure 3.2*, we could conclude that there are slight differences in the distribution of price between different response time.

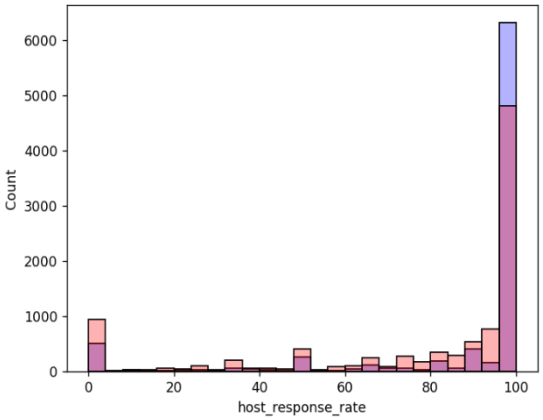

Figure 3.1 features about host activeness

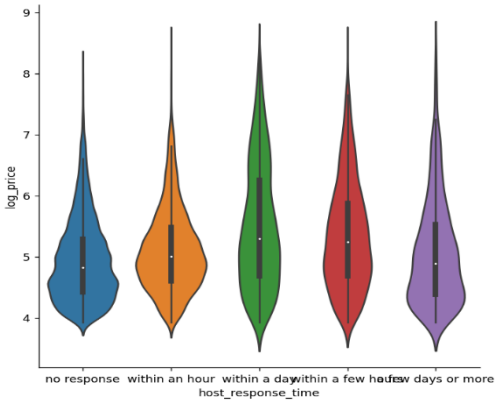

Figure 3.2 Price of different response time

The second finding is about *'host_total_listings_count'*. The interesting fact is that statistically tested, hosts with more than 1 listings are always related with higher price. The possible explanation is that hosts with more listings are richer than others, so their accommodations are luxury enough to set relatively higher rental price.

The last finding in this group is that identity verified host could receive slightly higher price. But with the number of verifications increasing, there is no obvious increment in price.

d) Features about location

We include *'neighbourhood_overview'*, *'neighbourhood_cleaned'*, *'latitude'*, *'longitude'*, etc. in this category. These features describe the exact and relative location of the listings.

The most useful finding in this group is shown in *Figure 4*. It demonstrates that geometric distribution impacts the price.

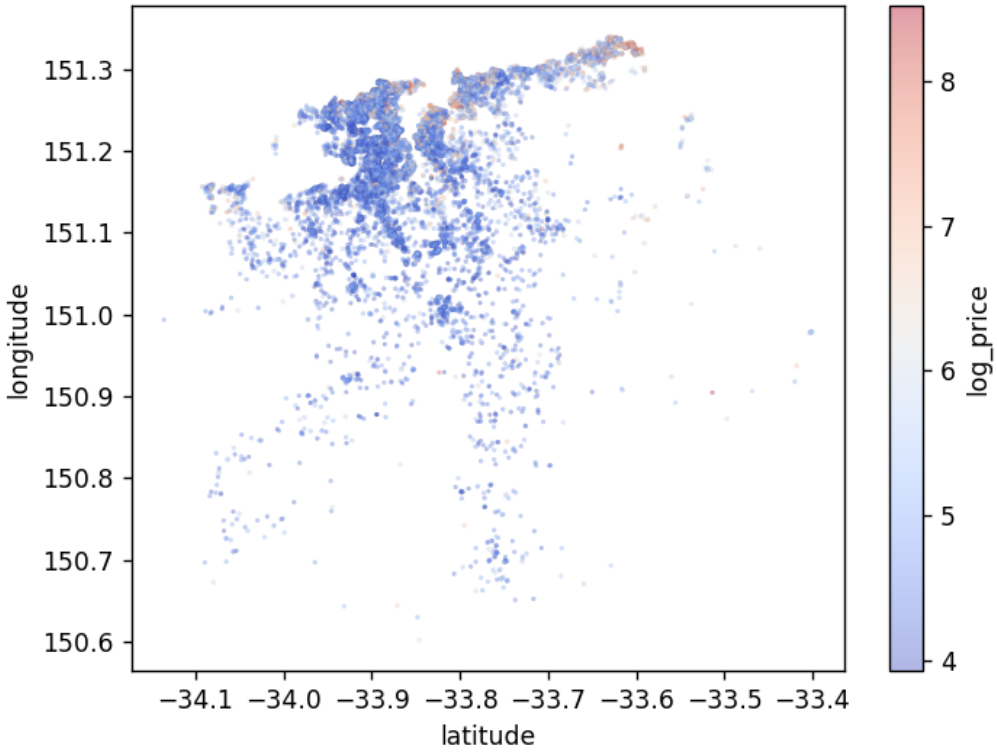

Figure 4 Price geometric distribution

e) Features about accommodations

We include *'property_type'*, *'room_type'*, *'accommodates'*, *'beds'*, *'amenities'*, etc. in this category. These features show the types and facilities of the accommodations, which are useful information on pricing.

Based on observation, we find that *'property_type'*, *'bathrooms_text'* and 'amenities' contains many useful words, we will process these two features using natural language processing model afterwards.

As shown in *Figure 5.1*, *'room_type'* gives us some enlightening conclusions that private room and shared room have lower average price compare with entire hotel/apt and hotel room.

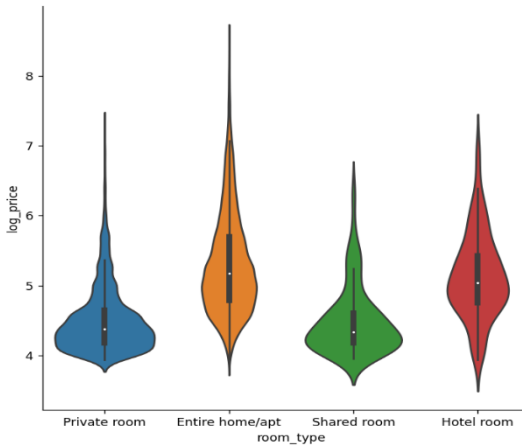

Figure 5.1 Price of different room types

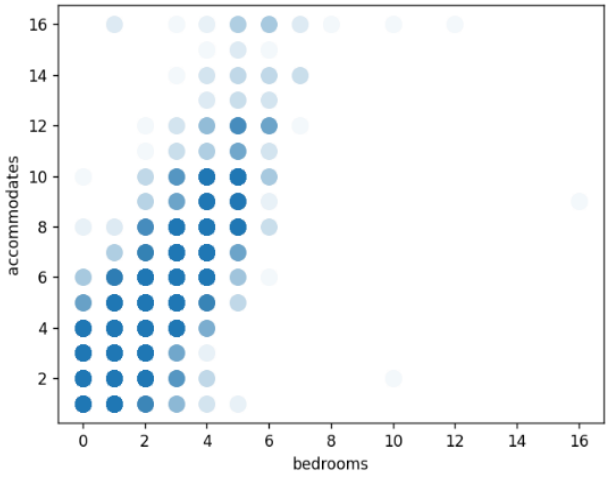

Figure 5.2 accommodates and bedrooms

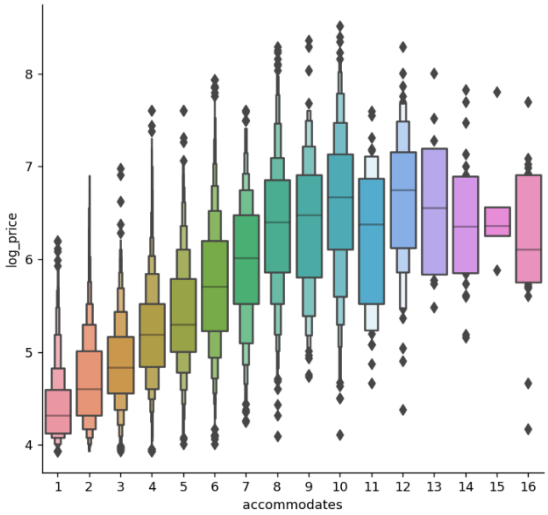

Figure 5.3 log-price and accommodates

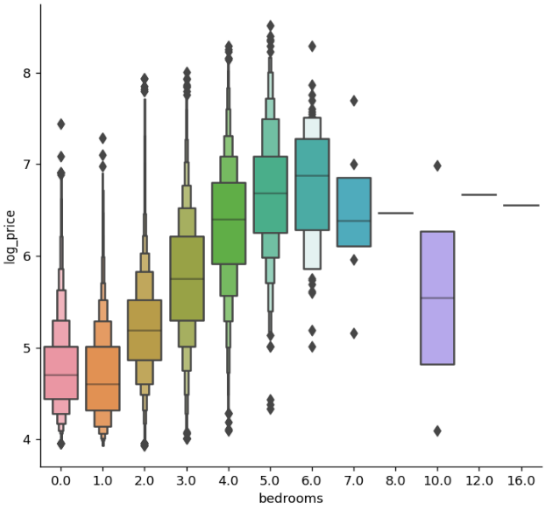

Figure 5.4 log-price and bedrooms

Another important finding exists in features *'accommodates'* and *'bedrooms'*. *Figure 5.2* demonstrates that *'accommodates'* and *'bedrooms'* have positive relationship. Then, we use these two features to draw boxplot against logarithm of price separately. We could conclude that when accommodates are less than 10, there is a positive linear relationship between price and accommodates. When the number of bedrooms is less than 6, there is a positive linear relationship either.

f) Features about availability

We include *'minimum_nights'*, *'maximum_nights'*, *'availability_30'*, *'availability_60'*, *'instant_bookable'*, etc. in this category. These features reflect the length of days customers could stay and the availability of the listings.

We use plots to illustrate the relationship between features and price, finding that the limitation of maximum or minimum days barely has impact on the price.

In this category, the most useful feature is '*has_availability*'. As shown in *Figure 6*, if the listings are available, they could rent for a higher price on average compared with those not available.

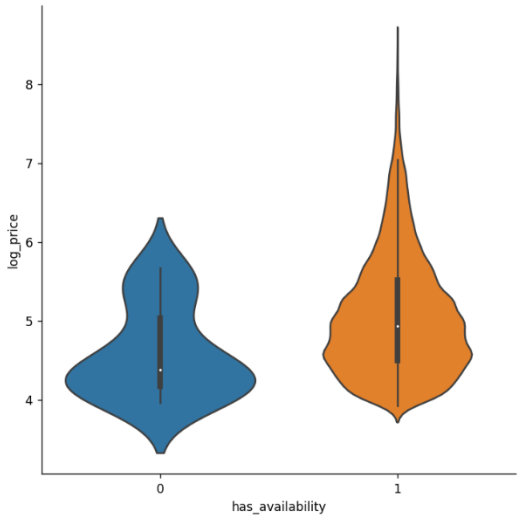

Figure 6 price of different availability

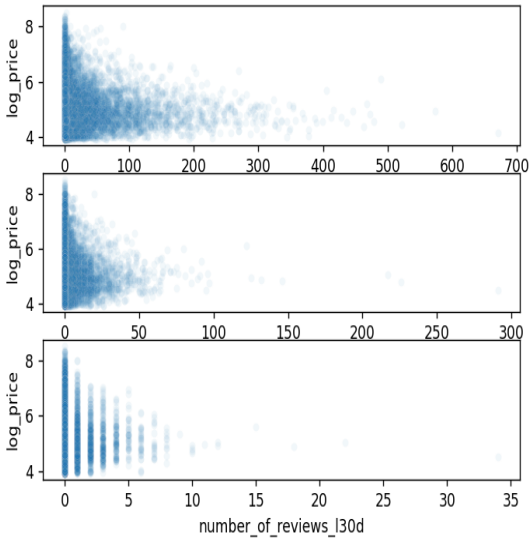

Figure 7 log-price and review numbers

g) Features about reviews

We include *'number_of_reviews'*, *'review_scores_rating'*, *'review_score_accuracy'*, *review_score_cleanliness'*, etc. in this category. These features describe the reviews given by the customers.

Among *'number_of_reviews'*, *'number_of_reviews_1tm'* and *'number_of_reviews_130d'*, there are many missing values. As shown in figure 7, these three features don't have a strong relationship with price.

Then we plot the price against many features about reviews in figure 8. The results show that accommodation with low rating has lower price, but high rating accommodation can still be cheap. But low rating and high price accommodations are extremely rare.

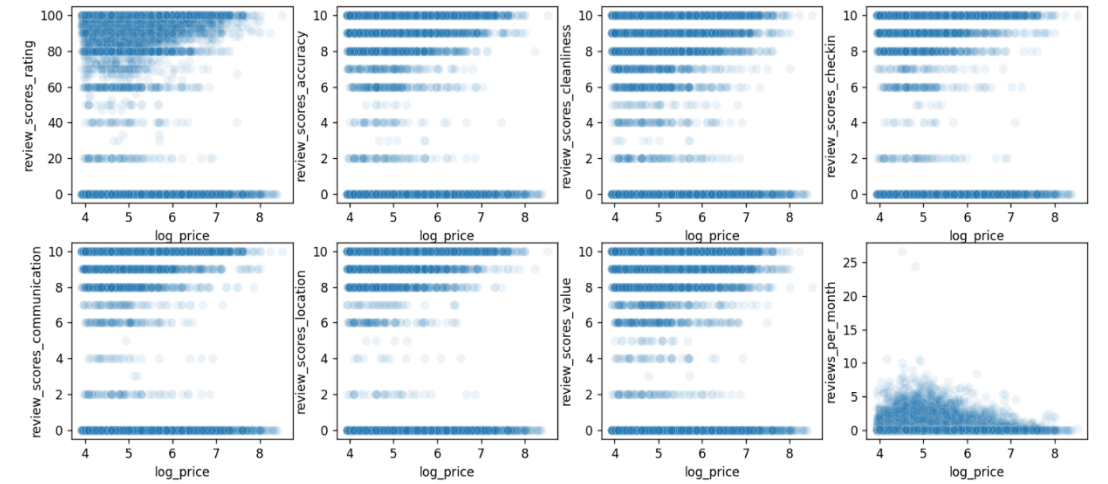

Figure 8 Relationship between log-price and reviews relating features

h) Features about listings

We include *'calculated_host_listings_count', 'calculated_host_listings_count_entire_homes', 'calculated_host_listings_count_private_rooms'* and *'calculated_host_listings_count_shared_rooms'* in this category. Listings are created to record accommodations that follow certain themes.

There is no special relationship between these four features and price.

In conclusion, we drop 16 features that are useless for prediction or could be replaced by others and add 5 features to describe missing values. The remaining features are useful, and we will deal with these features afterwards.

Types of features	**Dealing methods**
Features irrelevant for prediction	'name', 'host_name', 'host_location', 'host_about', 'first_review', 'last_review', 'calculated_host_listings_count', 'calculated_host_listings_count_entire_homes', 'calculated_host_listings_count_private_rooms', 'calculated_host_listings_count_shared_rooms',
Features can be replaced by others	'neighborhood_overview', 'host_neighbourhood', 'neighbourhood', 'neighbourhood_group_cleansed', 'host_total_listings_count', 'description'
Features added for describe missing values	'description_isna', 'host_about_isna', 'host_response_rate_isna', 'host_acceptance_rate_isna', 'review_isna'

Features transformed to logarithm form	'price'

Table 1 Dealing methods of features

III. Feature engineering

In process of exploratory data analysis, we also do some feature engineering, like implementing a logarithm transform to price, making *'description'* a dummy variable.

Then, we will continue to deal with some remaining features.

i. Text mining

Firstly, we deal with three text features, *'property_type', 'bathrooms_text'* and *'amenities'*.

(i) *'property_type'*

Based on observation, we find that the text contents in this feature are almost made up with adjectives and nouns. The adjectives contain 'private', 'shared' and 'entire', describing whether tenants need to share rooms with others. The nouns describe the type of the accommodations, like 'house', 'room', 'apartment', etc. There are also some prepositions like 'in the'. We drop those useless prepositions and remain 45 meaningful words.

We create 45 new dummy variables combine a prefix of 'property_' with the useful words we find above, like *'property_private', 'property_apartment'*, etc.

(ii) *'bathrooms_text'*

We find that *'bathrooms_text'* describes the number of bathrooms and the types of bathrooms. There are two types of bathrooms, 'private' and 'shared', so we created two variables *'bath_private'* and *'bath_shared'* to cover these messages. Besides, we transform the text 'half' to 0.5 and create a variable *'number_of_baths'* to represent the quantity information in *'bathrooms_text'*.

(iii) *'amenities'*

'Amenities' lists the useful facilities of the accommodations. In common sense, amenities could have positive effect on price. But the types of amenities vary from each other and there are 598 types of amenities listed in this feature. It is considered that rarely appeared amenities is not informative in predicting the price. The amenities that appear more than 30 times would be included in a new list. Every amenity in this list would be a new variable. In all, we set 120 new variables to describe the useful amenities.

ii. Encode categorial features

Secondly, we encode those categorical data. Based on our analysis above, we select three most representative features, *'host_response_time', 'neighbourhood_cleansed'* and *'room_type'*.

IV. Task 1: Build predictive models

In this section, five models are selected for the prediction: simple linear regression model, random forest, XGBoost, LightGBM, and model stacking. As mentioned in previous paragraphs, the distribution of 'price' is right-skewed, and our team used the log-transformation to make the distribution of this data closer to normal (Htoon, 2020). For all models, the price after log-transformation (*log_price*) is the dependent variable, and other selected features after feature engineering are independent variables.

i. Train-test-split

Given that the train dataset is large enough, we split the dataset into training set and validation set, with 70% data in the training set and 30% data in the validation set. The train-test-split technique is applied to evaluate the performance of selected models (Brownlee, 2020).

ii. Methodology

a) Linear model - Simple linear regression (Benchmark)

Linear regression model refers to the process of fitting the dependent variable and independent variables into a linear equation (Brownlee, 2020). The function 'LinearRegression' from scikit learn is used to fit the training set data into the model. This model will be used as the benchmark for evaluation as it is easy to use and interpret.

b) Random Forest

For the tree-based model, we first choose the random forest model rather than decision tree model since it has the higher prediction accuracy. As an ensemble learning method, random forest constructs multiple decision trees based on subsets of training data. The algorithm for random forests refers to the training method of Bagging and Boostrap.

To run in the random forest, the dataset is randomly grouped into training subsets and each training set is given to a decision tree which would predict a result. Then, the random forest regressor makes the final prediction based on the majority voting among the trees. According to Ho (1998), it returns the average prediction of a single tree for regression tasks, thus, improve the accuracy of prediction. Additionally, the increasing number of trees addresses the issue of overfitting and increase stability of prediction.

Before fitting the model, we use 'optuna' for hyperparameter optimization. 'optuna' contains three core terms: 'objective', 'trial' and 'study' in its optimization framework. 'objective' defines the function to be optimized and specifies the range of hyperparameter numbers and the term 'trial' is invoked to suggest methods to seek the best hyperparameters. Lastly, 'study' determines the optimization method and objective function.

Meanings and optimized values for parameters are displayed in the following table:

Parameters	Meaning	Value
n_estimators	The number of trees in the forest	150
min_samples_leaf	The minimum number of samples at a leaf node	2
max_features	The number of features to consider when looking for the best split	Auto
max_depth	The maximum depth of the tree	90

Table 2 Optimized parameters of Random Forest

After running the random forest model with optimized parameters, our team plot the variable importance of this model:

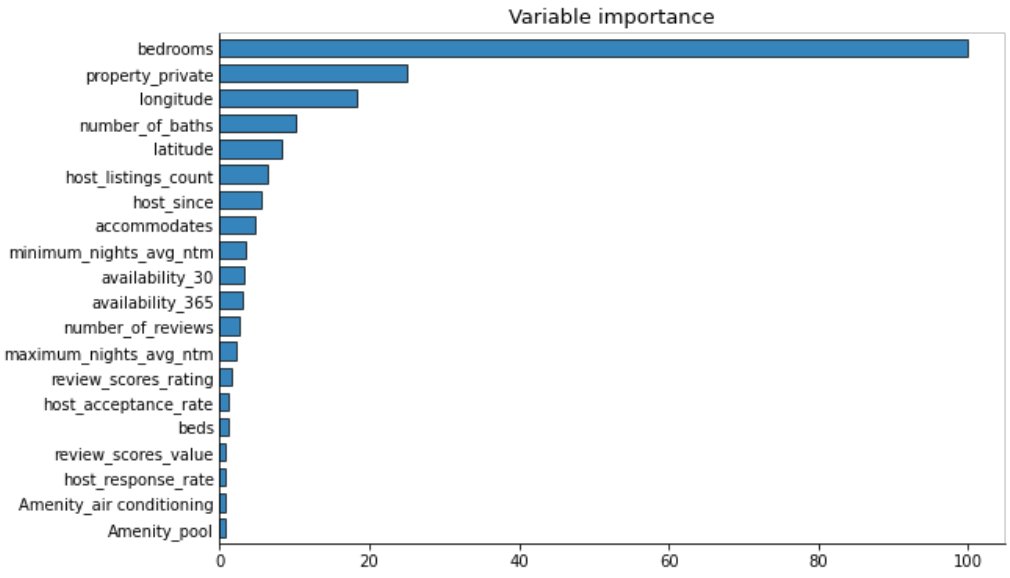

Figure 9 Variable importance of Random Forest

As shown in the figure, 'bedroom' has the highest feature importance, which means the prediction of *'log_price'* will be significantly affected by the value of *'bedrooms'*. Other variables such as properties, location of listings, and the number of bathrooms will also influence the *'log_price'*.

However, since random forest is not very suitable for regression tasks, we would consider other tree-based models such as XGBoost and LightGBM.

c) XGBoost

Bagging idea adopted by the random forest are also applications of the Bootstrap idea. The difference is that Bagging uses uniform sampling with replacement while Boosting samples according to the error rate. In addition, the selection of Bagging's training set is random, which means the training sets are independent of each other, while the selection of Boosting's training

set is related to the results of the previous round of learning. Boosting algorithm is an iterative process, and each new training could improve the previous results.

XGBoost is currently one of the widely used and well-performing models. When using CART as a base classifier, XGBoost explicitly adds a regular term to control the complexity of the model, which helps prevent overfitting and improve the generalization ability of the model. In addition, the traditional GBDT uses all the data in each iteration, but XGBoost uses a strategy that supports sampling of the data.

When determining the hyperparameters, we used a grid search method and cross-validation to compare several major hyperparameters that have a greater impact on the model. Finally, following customization are made to parameters of the XGBoost:

Parameters	**Meaning**	**Value**
learning_rate	Shrinks the contribution of each tree by learning rate	0.04
max_depth	Maximum depth of the individual regression estimators	8
n_estimitors	The number of boosting stages to perform	500

Table 3 Optimized parameters of XGBoost

The feature importance of XGBoost is displayed as below:

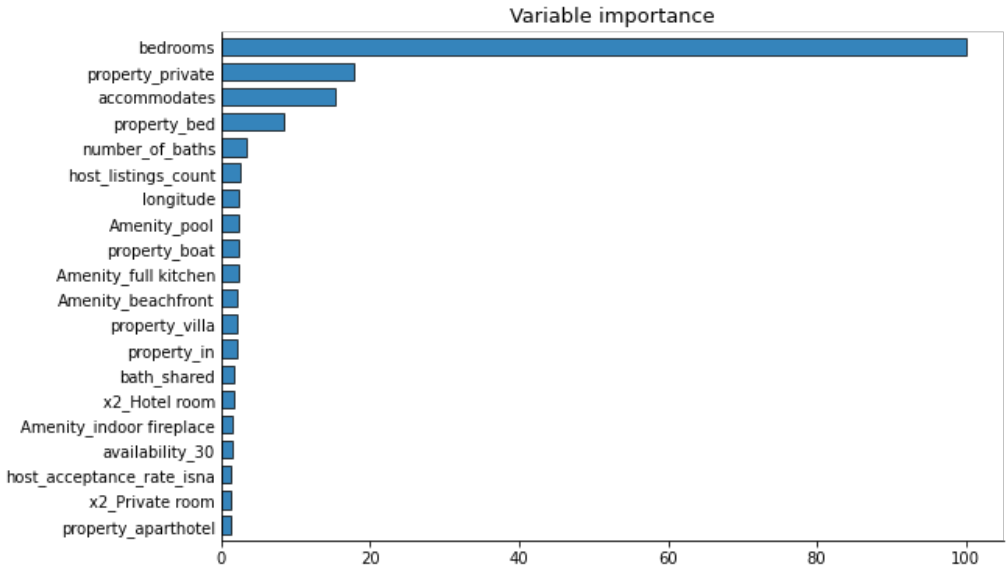

Figure 10 Variable importance of XGBoost

Same as the random forest model, *'bedrooms'* is the most significant features for the independent variable *'log_price'*. Additionally, *'log_price'* may be affected by other variables including *'property_private'*, *'accommodates'*, and *'property_bed'*.

d) LightGBM

LightGBM is another successful gradient boosting method that can generate great outcomes under many settings. Moreover, research and implementations suggest that this model has several advantages such as faster calculation speed, higher accuracy, and lower memory occupation when compared to XGBoost and CatBoost (Al Daoud, 2017).

Before the hyperparameter optimization, symbols in variable names that have been added in the feature engineering process needs to be removed to avoid errors. Then, hyperparameters in the LightGBM are selected using the module called 'optuna', detailed description of 'optuna' has already been covered when discussing the random forest. Timeout is set at 40 minutes as the model is more likely to be well-tuned with a longer time, this is the optimized result:

Parameters	Meaning	Value
learning_rate	Shrinks the contribution of each tree by learning rate	0.03
num_leaves	Maximum tree leaves for base learners	57
max_depth	limit the max depth for tree model	7
lambda_l1	L1 regularization term on weights.	5.41×10^{-5}
lambda_l2	L2 regularization term on weights.	8.52137
bagging fraction	Used to speed up training and deal with over-fitting	0.88360
bagging_freq	Frequency for bagging	6
feature_fraction	Select % of features before training each tree	0.35673
min_data_in_leaf	Minimal amount of data in one leaf	2

Table 4 Hyperparameter optimization of LightGBM

After fitting the model using results of the hyperparameter optimization, our team plot the variable importance of LightGBM:

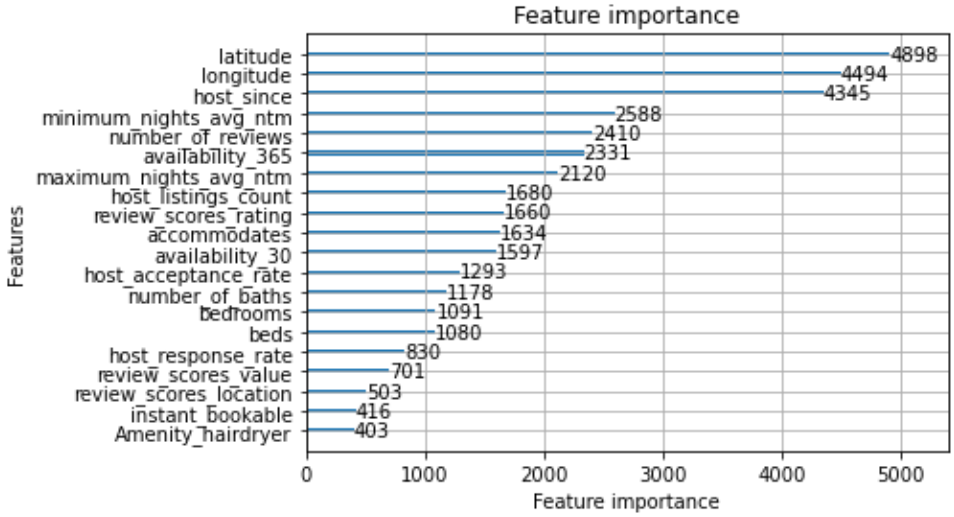

Figure 11 Variable importance of LightGBM

The graph indicates that *'latitude'* and *'longitude'*, which refers to the location of the listing, have the most significant influence. How long since the person became a host is also critical to the value of *'log_price'*. Moreover, the minimum and maximum nights required for staying, number of reviews, availability in a year, and many other features will have certain level of influence on *'log_price'* of listings.

e) Model stacking

Model stacking is a method to improve prediction performance by using outputs of multiple models and running them through a meta-model, which can be any learning algorithm (Pedersen, 2021). Model stacking has the function to minimize the weakness and maximize the strength of individual models, which has the potential to achieve the best generalization performance. However, the risk of overfitting validation set may also increase as the complexity of the meta-model becomes higher.

The function 'StackingCVRegressor' is imported to make the stacked model. Before model stacking, our team notice that 'StackCVRegressor' only supports the API generated from scikit learn. However, the LightGBM is trained using a separate module 'lightgbm' and those parameters cannot be used directly in 'StackCVRegressor'. We solve this problem by extracting hyperparameters optimized using 'lightgbm' and then redefining a model called 'LGBMSTACKmodel'.

Simple linear regression, random forest, XGBoost, and LightGBM are included in the list of regressors for the stacked model. We use LightGBM with the same hyperparameter as mentioned above to be the meta-regressor due to its good performance in the original dataset. 5-fold CV is set to determine the cross-validation strategy.

'use_features_in_secondary' is set to be True as to include the original dataset together with the stacking models, in order to enhance the performance. Other parameters such as *'store_train_meta_features'* is set as True and the value of *'n_jobs'* is -1.

iii. RMSLE with the validation set

After parameters and settings for all five models are determined, models can be built. Then, validation set can be used to measure the performance of models using root mean squared logarithmic error (RMSLE). RMSLE is an extension on mean squared error (MSE) and it is mainly used when predictions have large deviations (Lepelaars, 2019).

Model	RMSLE of validation set
Simple Linear Regression	0.42400
Random Forest	0.39779
XGBoost	0.37850
LightGBM	0.37426
Model Stacking	0.37224

Table 5 RMSLE of validation set for each model

From the table, other four models outperform the benchmark model and model stacking has the lowest RMSLE, which means it has the highest predictive accuracy. Hence, the stacked model will be selected as the final prediction model and used for Kaggle competition.

iv. Back transformation bias and Kaggle competition results

Before using models for the test dataset, our team retrains all models with the train dataset, which is the cleansed dataset before splitting into training set and validation set.

We also observe that in previous modelling phase, *'log_price'* is selected as the dependent variable. However, the purpose of the project is to predict the rental price of listings, so prediction of *'log_price'* needs to converted back to *'price'*. In that case, the re-transformation bias must be considered. The formula of back transformation is:

$$\hat{f}(x) = \exp(\hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_j)[(\frac{1}{n}) \sum_{i=1}^{n} \exp(e_i)]$$

The correction of re-transformation bias is same for each of five models and involves four steps. Firstly, we calculate the predicted *'log_price'* value by running all predictors in the model. The second step is using the actual *'log_price'* minus the predicted *'log_price'* to get the residual error. Then, the adjustment equals to the average expected value of residuals. As stated in the back transformation formula, expected value of *'log_price'* times the adjustment will give use the final predicted value of *'price'*.

After finishing all modelling processes, predictions are submitted to Kaggle. The stack model receives the best score among five models with the RMSLE of 0.37508 on Kaggle.

Model	RMSLE
Simple Linear Regression	0.44810
Random Forest	0.40088
XGBoost	0.38182
LightGBM	0.37714
Model Stacking	**0.37508**

Table 6 Kaggle score of each model

V. Task 2: Data mining for 'best hosts'

i. Criteria of 'best hosts' and model for data mining

After getting the results of data analysis, data mining would be processed to determine the criteria of 'best hosts' and extract the business insights describing what the best hosts are doing.

Firstly, we decide the best hosts from the perspective of unit price of a property which is calculated by Price/Accommodates since the purpose of hosts and real estate investors is to

maximize the property income. According to our EDA for the original dataset, we find that *'accommodates'* and the target variable *'log price'* have the positive linear relationship when accommodates are less than 10. It is assumed that the per capita living area of different property types is similar, if hosts could earn more revenue from each customer, they would be more profitable, holding the cost constant.

The third quartile of per capita price which is $75 is set to be the hurdle of best hosts. When the unit price of a property is greater than $75, it is regarded as 1; otherwise, it is regarded as 0. We could build a XGBoost classification model to select the best hosts whose property have reached the per capita price of $75. Additionally, the classification model could indicate the most important features and enable us to obtain insights about characteristics of the best hosts. The variable importance of XGBoost classification model is indicated as below figure:

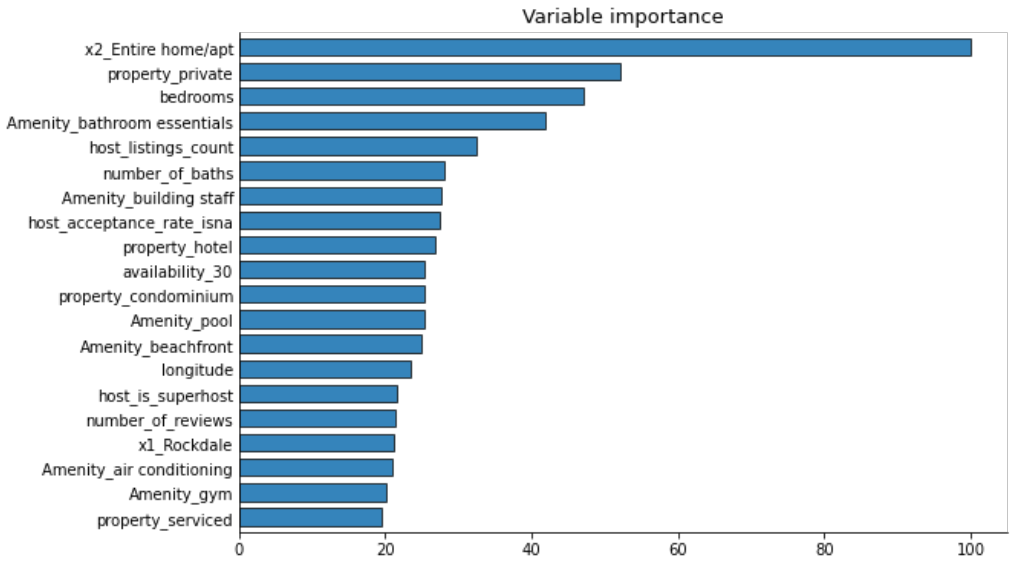

Figure 12 Variable importance of XGBoost classification model

ii. Business insights

As can be seen from the results in the above figure, the type of house has a very large impact on the unit price. Entire house leasing represents greater autonomy to tenants, which means tenants could choose to live with their families or other tenants and they are able to enjoy the whole space and furnishings in the house.

The number of bathrooms also plays an important role. The best host always enable tenants to have their own separate bathroom. It could ensure that there is no interference between tenants to a certain extent.

Moreover, best hosts would consider the housing amenities, such as building staff, air-conditioning, pool, or gymnasiums. A well-equipped house will obviously have a higher unit price. They could increase the value of their rental housing by providing supporting facilities to tenants.

Whether the host is a super host also affects the unit price of the house. Super hosts need to reach a high response rate and acceptance rate. From this result, it can be known that in addition to the house itself, the host's service attitude will also have influence on pricing. Therefore, it is necessary for the best hosts to have a positive attitude towards tenants.

VI. Conclusion

According to results in the modelling stage, the stack model has the highest accuracy to predict the rental price and is the best of the five models with RMSLE of 0.37508. Then, we use selected features to process data mining about "best hosts". The business insights are based on important features that influence unit price of the house and best hosts. However, the criteria need to be improved since it is affected by outliers and is not normally distributed. We should consider more factors which affect profits such as locations and the marginal cost of rental and synthesize a variety of factors to get a better criterion of "best hosts" in the future work.

Reference list

Brownlee, J. (2020). *Linear regression for machine learning.* Retrieved from https://machinelearningmastery.com/linear-regression-for-machine-learning/

Brownlee, J. (2020). *Train-test split for evaluating machine learning algorithms.* Retrieved from https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/

Ho, T.K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(8),* 832-844. https://ieeexplore.ieee.org/document/709601

Htoon, K.S. (2020). *Log transformation: Purpose and interpretation.* Retrieved from https://medium.com/@kyawsawhtoon/log-transformation-purpose-and-interpretation-9444b4b049c9

Lepelaars, C. (2019). *Understanding the metric: RMSLE.* Retrieved from https://www.kaggle.com/carlolepelaars/understanding-the-metric-rmsle

Pedersen, T. (2020). *Improve machine learning predictions.* Retrieved from https://medium.com/geekculture/how-to-use-model-stacking-to-improve-machine-learning-predictions-d113278612d4

Appendix – Project management

Our group management method is to split the group work as reasonably as possible, understand the different situations that each member is in, and give play to each person's abilities.

We divided this project into two parts, one is the preliminary preparation, and the other is the modeling and analysis results. In these two parts, we ensure the participation of each member, so that everyone can make full use of classroom knowledge to practice. After completing the tasks, we also cross-check the members' part according to our own time allocation and form the conclusion through several meetings.

We think the most useful management idea is to record our discussion process and speak out our thoughts candidly.

Recording the discussion process allows us to find a basis when completing the task and avoid repeated discussions. This method is conducive to us in advancing the task.

On the other hand, we were a little worried about expressing our thoughts at first because we were afraid of disputes. As the project progressed, we understood each other better and realized that we should not reach consensus too easily because of the fear of disputes. Thus, we agreed to express our opinions frankly and discuss the problem thoroughly to achieve better results.