Airbnb - Pricing Model - Technical Report

Phakphum Jatupitpornchan

This technical report aims to provide a more detailed description of the implemented prediction process and obtained results. The main goal of the prediction models is to predict the price of the Airbnb apartments in Bangkok in September 2023 which can accommodate 2-6 people.

Code for the analysis can be found in the following link:https://github.com/PhakphumJ/DA3-phdma/tree/main/Assignment%202

Task 1 - Building, Selecting, and Evaluating the Models

Sample Design

I build the prediction models using data on Airbnb listings in Bangkok in March 2023. The whole sample includes both apartments and condominiums. I include condominiums to increase the sample size. Condominiums are more similar to apartments than other types of properties such as houses or villas. The sample only include listings that can accommodate 2-6 people.

I split the sample into a work set and a hold-out set. To further ensure that the evaluation results on the hold-out set is as close to the live data as possible, I include only apartments in the hold-out set by randomly select 20% of the apartments from the whole sample to be in the hold-out set. The remaining observations are in the work set.

I chose this approach as opposed to randomly splitting the whole sample into a work set and a hold-out set, and filtering the hold-out set to include only apartments later. This is because the latter approach would waste a lot of observations.

This results in 5,058 observations in the work set and 146 observations in the hold-out set (before cleaning for missing values).

Label Engineering

The target variable is daily price of the listing in Thai Baht (THB). I use the level of the price as the target variable in model building without any transformation.

Although the price may have a non-linear relationship with the important predictors, I choose to capture this by using quadratic terms of the predictors instead.

Feature Engineering

Feature Selection

I use almost all the variables in the dataset as features. The variables that excluded are those without hints of being relevant to the price. For example, *host_name*, *latitude*.

The list of variables selected are:

- Information about host (*host_response_time*, *host_response_rate*, *host_acceptance_rate*, *host_is_superhost*, *host_total_listings_count*, *host_identity_verified*)
- Information about the accomodation (*neighbourhood_cleansed*, *property_type*, *room_type*, *accommodates*, *bathrooms_text*, *bedrooms*, *beds*, *amenities*, *minimum_nights*, *maximum_nights*, *has_availability*, *availability_30*, *availability_60*, *availability_90*, *availability_365*, *number_of_reviews*, *number_of_reviews_ltm*, *last_review*, *last_scraped*, *review_scores_rating*, *review_scores_accuracy*, *review_scores_cleanliness*, *review_scores_checkin*, *review_scores_communication*, *review_scores_location*, *review_scores_value*, *instant_bookable*, *description*, *neighborhood_overview*, *amenities*)

The descriptions of these variables are in the appendix.

Feature Transformation

Apart from transforming categorical variables and amenities into dummy variables[1], I also create a few new variables which may be useful for prediction.

The new variables are: *num_baths*, *num_shared_baths*, *num_private_baths*, *is_half_bath*[2], *time_since_last_review*, *near_MRT*, *near_BTS*, *near_ARL* [3]

The descriptions of these variables are in the appendix.

[1] Code provided by ChatGPT is used to transform categorical variables: https://chat.openai.com/share/7389fcaf-0deb-47d0-86b7-edf607782d53

[2] Extracting number of baths is helped by ChatGPT: https://chat.openai.com/share/2d4158e7-5a01-494d-ad85-583a3992ec02

[3] Extracting information about the rail transportation is helped by ChatGPT: https://chat.openai.com/share/5d79847d-efc7-4a9d-93bf-a3b1fbb25484

For amenities, since the names of some amenities are not standardized, I do not use hot encoding. Instead, I create a list of name of amenities that are more relevant to the price.

The list is: ("Hair dryer", "Shampoo", "Shower gel", "Air conditioning", "Essentials", "Wifi", "Washer", "Iron", "Smoking allowed", "Free parking on premises", "Luggage dropoff allowed", "Kitchen", "Refrigerator", "Dining table", "Dedicated workspace", "Elevator", "Microwave", "Dishes and silverware", "TV", "Pool", "Gym", "Pets allowed")

This list is used to create dummy variables indicating whether the listing has the corresponding amenity or not[4].

Dealing with NAs

There are a few variables with NAs. The reasons for the missing values maybe from the hosts not providing the information or the listings/hosts have not gotten any requests/reviews yet.

The variables with missing values are shown in the Table 1.

Table 1: Missing Values

Variable	NA Count
review_scores_location	1465
review_scores_value	1465
review_scores_checkin	1464
review_scores_communication	1463
review_scores_accuracy	1462
review_scores_cleanliness	1462
review_scores_rating	1411
time_since_last_review	1411
host_acceptance_rate	1006
host_response_rate	872
host_response_time_within a day	872
host_response_time_within a few hours	872
host_response_time_within an hour	872
bedrooms	362
near_MRT	199
near_BTS	199
near_ARL	199
beds	132
num_baths	18
num_shared_baths	12

[4]Code provided by ChatGPT is used to transform amenities into dummy variables: https://chat.openai.com/share/19bca20a-eef6-4ef9-bb6a-5fce8f2fd0d9

- For scores, replace it with the mean and add binary flag variable to capture this.
- For *time_since_last_review*, replace it with max(time_since_last_review). These are accommodations with no review yet. Prospect customers may see these place similar to those which did not get any review for a long time.
- Those with *host_acceptance_rate* and *host_response_rate* = NA are likely to be newcomers. We will drop these.
- For *near_MRT*, *near_BTS*, and *near_ARL*, replace NA with 0.
- For *host_is_superhost*, replace NA with 0.
- For *bedrooms* and *beds*, replace NA with the mean and add binary flag variable to capture this.
- Those with missing values for the number of bathrooms can be dropped since there are only a few of them.

Model Building

I consider three main types of models: linear regression, random forest, and bagging. Linear regression is simple and offers high interpretability, random forest offers high-quality prediction. Bagging is chosen to demonstrate the gain from decorrelating the trees.

The details of each model will be discussed in the following sections.

OLS

8 linear regression models are considered. 5-fold cross-validation is used to select the best model. The one that with the lowest RMSE is chosen.

accommodates, *bedrooms*, *beds* have the highest correlation with *price*. Therefore, I start with a model with only these variables.

The eight models are:

1. Model 1: number of people that can be accommodated, number of beds, number of bedrooms (enter linearly)
2. Model 2: Model 1 + other characteristics of the living space except amenities
3. Model 3: Model 2 + information about rail transportation
4. Model 4: Model 3 + variables related to reviews (only those with correlation less than 0.7 with each other)
5. Model 5: Model 4 + squared terms of variables in model 1
6. Model 6: Model 5 + information about the neighborhood

7. Model 7: Model 6 + information about the host
8. Model 8: All variables

The details of the models are:

Model 1: $price = f(accommodates, bedrooms, beds, bedrooms\_imputed)$

Model 2: $price$ = model 1 + $property\_type$, $room\_type$, num_baths, num_shared_baths, is_half_bath, $num\_private\_baths$

Model 3: $price$ = model 2 + $near\_MRT$, $near\_BTS$, $near\_ARL$

Model 4: $price$ = model 4 + $review\_scores\_rating$, $review\_scores\_cleanliness$, $review\_scores\_communication$, $review\_scores\_location$, $review\_scores\_rating\_imputed$, $time\_since\_last\_review$, $number\_of\_reviews$

Model 5: $price$ = model 4 + $accommodates^2$, $bedrooms^2$, $beds^2$

Model 6: $price$ = model 6 + $neighborhood$

Model 7: $price$ = model 7 + $host\_is\_superhost$, $host\_response\_time$, $host\_response\_rate$, $host\_acceptance\_rate$, $host\_total\_listings\_count$, $host\_identity\_verified$

Model 8: $price = f$(all variables)

The RMSE of the models on the test set are present in Table 2. The best model is Model 2.

Table 2: RMSE of the OLS Models[5]

Model	RMSE
OLS_1	9822.85
OLS_2	9818.92
OLS_3	9824.60
OLS_4	9825.56
OLS_5	9833.93
OLS_6	9880.11
OLS_7	9896.80
OLS_8	9937.50

[5]ChatGPT is used to transform the table into markdown format: https://chat.openai.com/share/7f201f41-efc2-4996-9c7d-d630b7954011

Random Forest

Next, I build a random forest model. 5-fold cross-validation is used to tune the parameters (*mtry*: Number of variables randomly sampled as candidates at each split). To save computation time, I only use 200 trees.

$mtry = 20$ produces the lowest RMSE.

Bagging

I build a bagging model with 200 trees. 5-fold cross-validation is used to tune the parameters (*minsplit*: Number of minimum observations in the terminal nodes).

The optimal *minsplit* is 6.

Evaluation and Diagnostics

Before evaluating the model. Each model is re-estimated using the whole work set. The hold-out set is cleaned and transformed using the same procedure as the work set. There is no additional missing value issue that need to be dealth with.

Then, the models are evaluated on the hold-out set. The RMSE and RMSE/Mean(Price) of the models on the hold-out set are presented in Table 3 and Table 4. While the OLS performs reasonably well, is significantly outperformed by the bagging model. However, the random forest comes out on top. It performs better than the bagging model because the benefit from decorrelating the trees. Nonetheless, the difference is small.

Table 3: RMSE of The Models on The Hold-out Set

	OLS	RF	Bagging	Mean price
Small apt.	1926.32	1428.94	1474.40	1846.50
Large apt.	2265.10	1713.20	1784.41	3480.42
All apt.	2027.62	1514.53	1568.06	2307.35

Table 4: RMSE/Mean(Price) of The Models on The Hold-out Set

	OLS	Random Forest	Bagging	Mean price
Small apt.	1.04	0.77	0.80	1846.50
Large apt.	0.65	0.49	0.51	3480.42
All apt.	0.88	0.66	0.68	2307.35

Looking at the performance across the size of the apartments, the random forest also model performs the best for all sizes. One interesting observation is that all of three models perform significantly worse for the small apartments. The performance of the OLS for small apartments is 60% worse than the performance for large apartments (RMSE/Mean(Price) increases by 60%). The random forest model and bagging model both performs 57% worse for small apartments than for large apartments.

I extract the the table showing the results of random forest model in the case study and show it in Table 5.

Table 5: Performance of Random Forest Model in the London Case Study

	RMSE	Mean price	RMSE/price
Small apt	28.53	62.3	0.46
Large apt	62.11	144.6	0.43
All	42.36	88.8	0.48

Comparing with the results in Table 4, models built in this exercise are not as successful as the models in the case study, especially for the small apartments. This might be because of the sample design, sample size, or number of trees. However, I believe that a major factor is the difference in the features. In Bangkok's dataset, there is no information about cancellation policy, which is shown to be an important feature in the case study. Another potentially important difference in feature is the time since the first review. For this exercise I opted for the time since the last review instead.

Next, the actual price and predicted price are plotted in three figures below. The dashed line is the 45-degree line. The closer the points are to the line, the better the prediction is.

Two observations emerge from the figures. First, the random forest model and bagging are very good at predicting the price of lower-priced apartments. Second, all models are not good at predicting the price of the very high-priced apartments.

Figure 1: Actual and Predicted Price from OLS (Hold-out set)

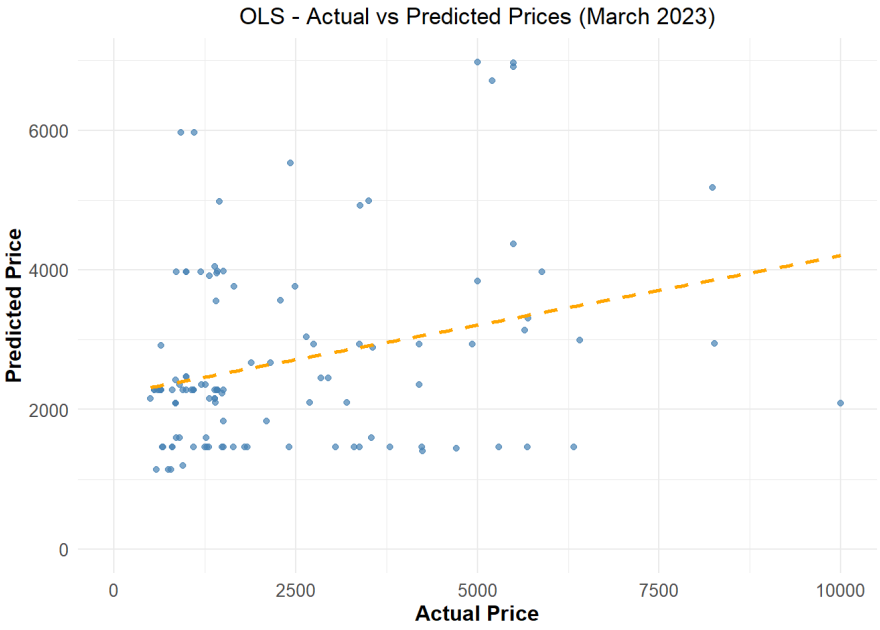


Figure 2: Actual and Predicted Price from Random Forest Model (Hold-out set)

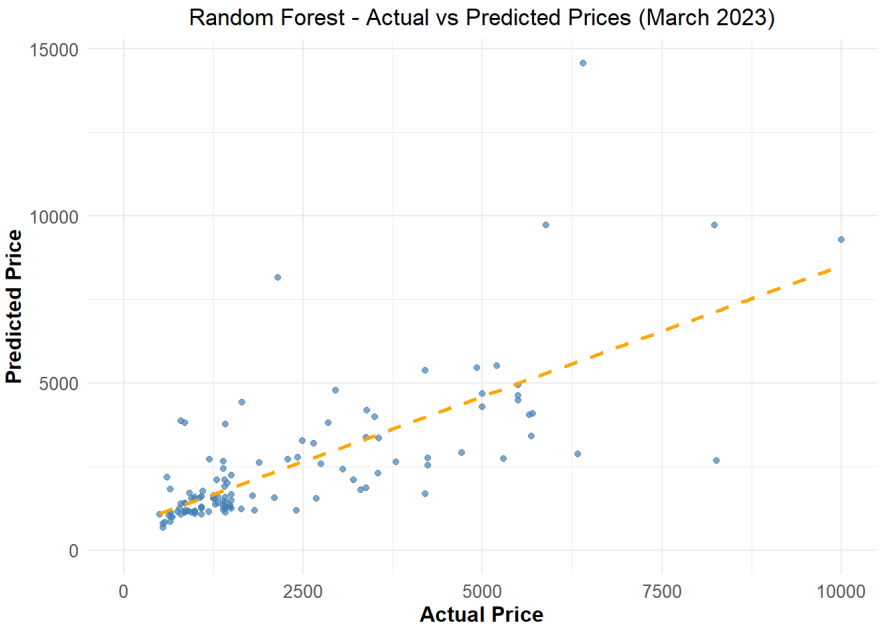
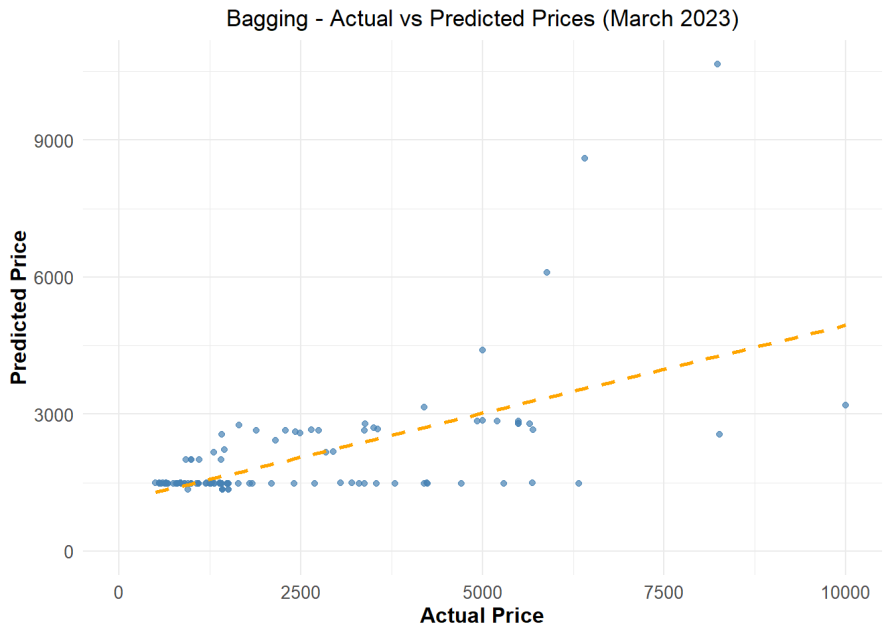


Figure 3: Actual and Predicted Price from Bagging Model (Hold-out set)



Task 2 - Predicting on the Live Data

I take the three models and predict the price of the apartments in the Bangkok in September 2023 which can accommodate 2-6 people. The performance of the models are presented in Table 6.

Table 6: RMSE/Mean(Price) of The Models on Live Data

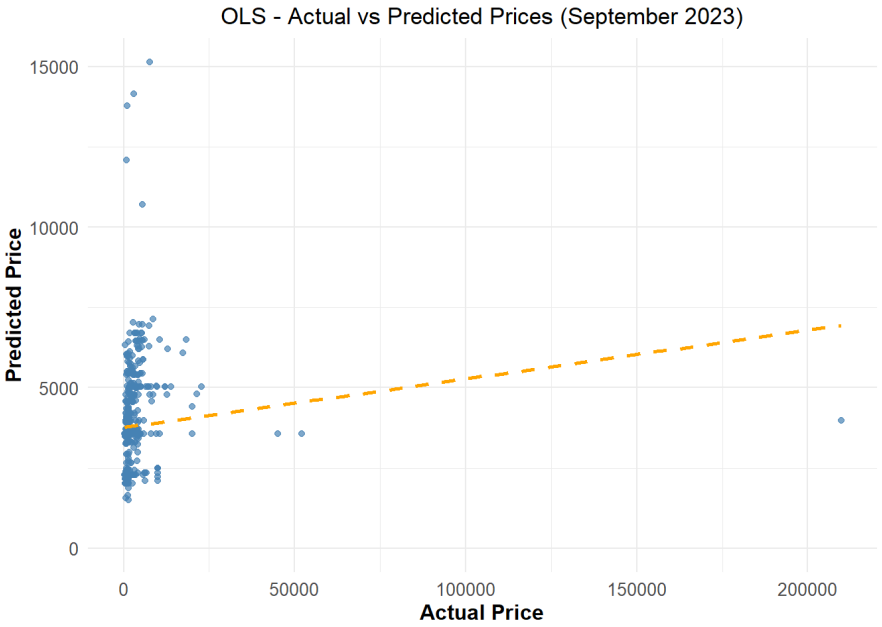
	OLS	Random Forest	Bagging	Mean price
Small Apt.	4.13	4.06	4.18	2552.30
Large Apt.	0.96	0.66	1.00	3852.46
All Apt.	3.15	3.06	3.19	2912.05

The order of the performance is the same as in the hold-out set. The random forest model performs the best, followed by the bagging model, and the OLS model. However, all models perform significantly worse than in the hold-out set. This is true across all sizes of the apartments, especially for small apartments. This might be because the relationship between the price and the features changes from March to September. March is the low season in Thailand while September is the high season.

The actual price are predicted price (from OLS) are shown in Figure 4. The dashed line is the 45-degree line. The plot tells that there is something very unusual in the data. It can be seen

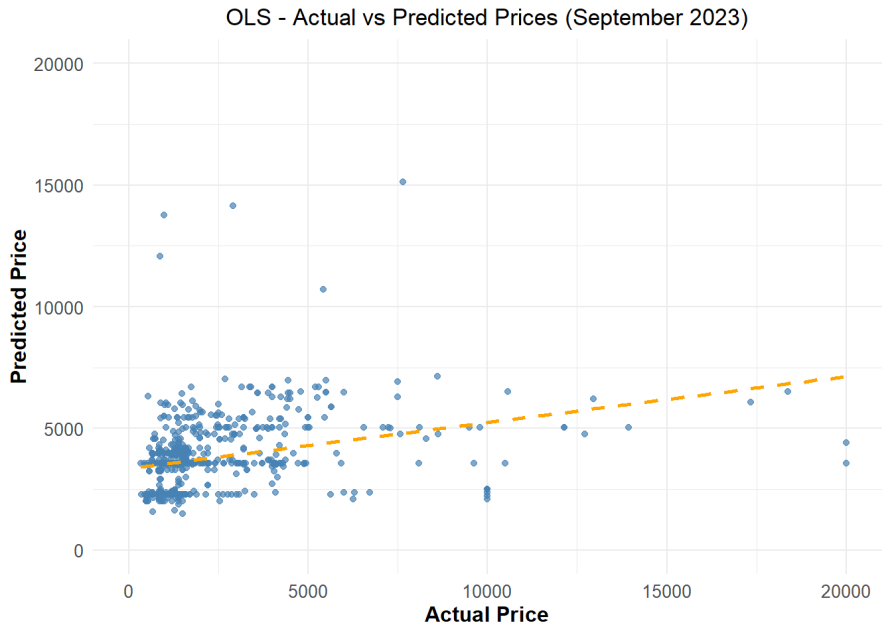
that there are extreme outliers in the price of the apartments in September. These may partly explain the significance increase in the RMSE of the models. Recall that the maximum price of apartments in March is only around 10,000 THB.

Figure 4: Actual and Predicted Price from OLS (Live Data)



To get more insights on the performance, I exclude the outliers from the plot ($price > 20,000$). The new plots are presented below.

Figure 5: Actual and Predicted Price from OLS (Live Data, outliers excluded)



Similar stories with the hold-out set emerge here. One additional observation is that the random forest model is exceptionally good at predicting the price of the lowest-priced apartments.

Figure 6: Actual and Predicted Price from Random Forest Model (Live Data, outliers excluded)

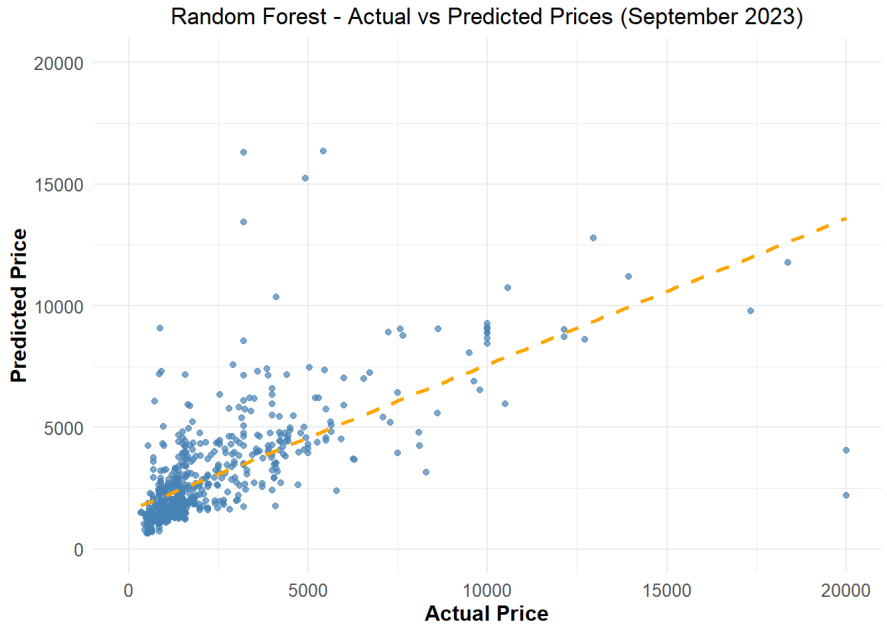
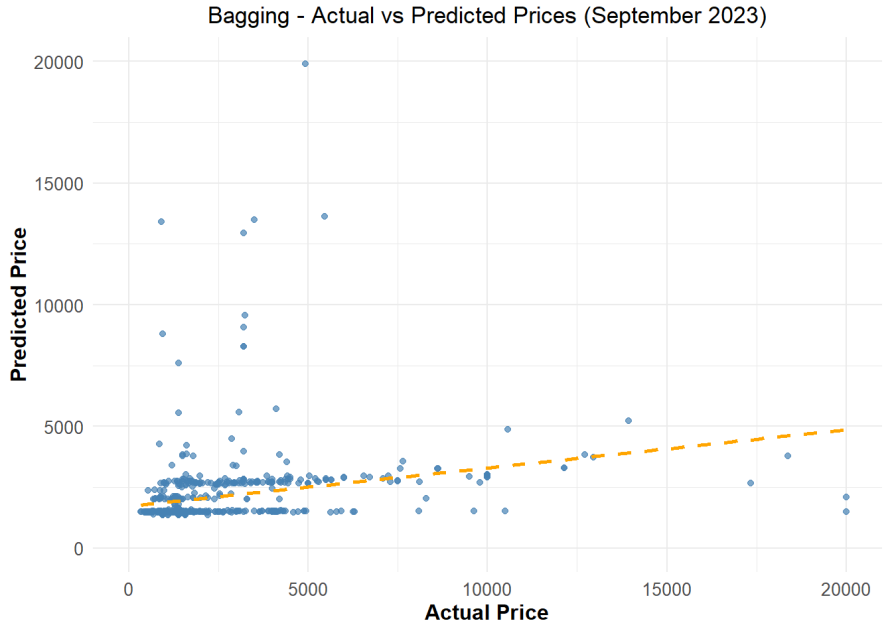


Figure 7: Actual and Predicted Price from Bagging Model (Live Data, outliers excluded)



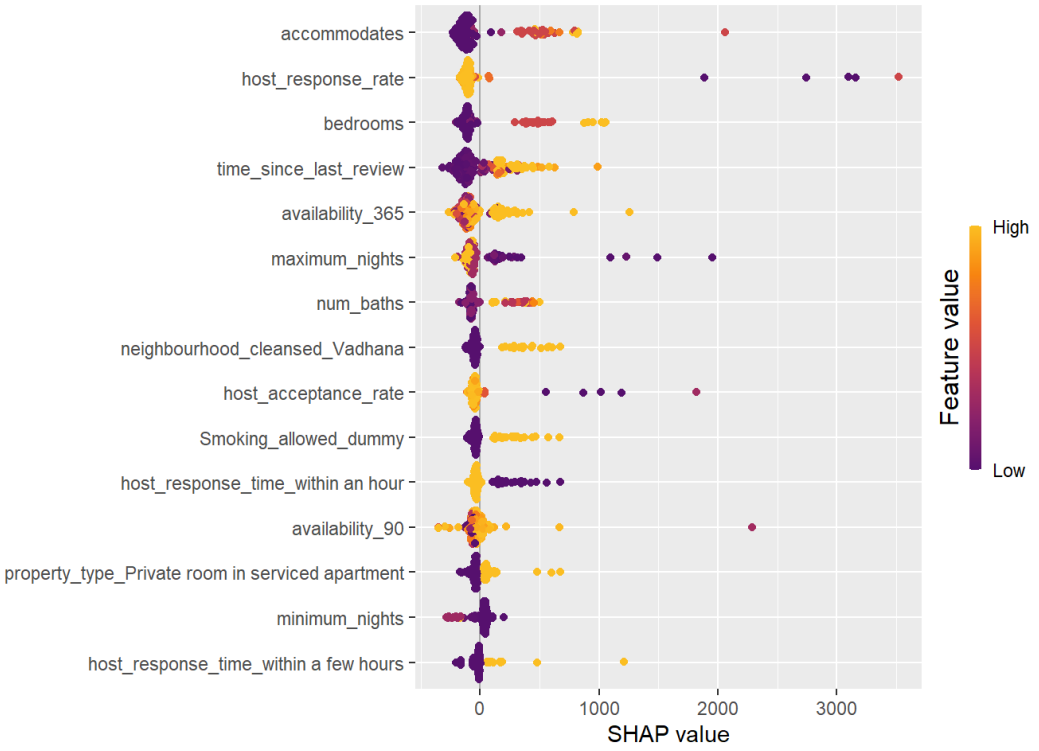
Task 3 - Explaining the Predictions from ML Model.

The distribution of SHAP value of the top features in the random forest model is presented by using a beeswarm plot (using observations in the hold-out set). It shows how the values of the features affect the predicted price of each observation.

accommodates is shown to be the most important feature. The effect of *accommodates* is also consistent with the usual expectation. The more people that can be accommodated, the higher the predicted price.

For most features, the effect is consistent with the usual expectation. For example, having number of beds lower than the average decreases the predicted price. Or, when smoking is allowed, the predicted price is higher. Being situated in the Vadhana district is also important and increases the predicted price. This is consistent with the fact that Vadhana is a central commercial district in Bangkok. It is also one of where the BTS intersects with the MRT.

Figure 8: Beeswarm Plot

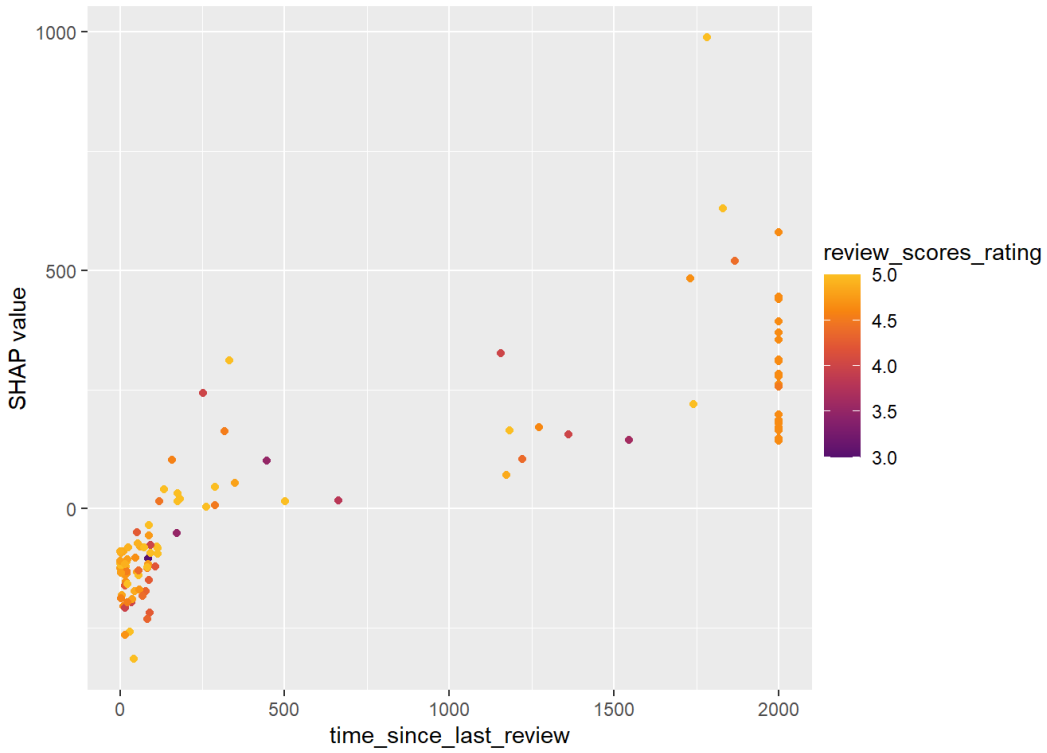


However, there are some features that have opposite effect from what we would expect. These are *host_response_rate*, *time_since_last_review*, and *host_response_time_within an hour*. I expected that when the time since the last review is longer, the prospect guests would be less likely to trust the reviews and the predicted price would be lower. However, the plot

shows that the predicted price is higher when the time since the last review is longer. For *host_response_time_within an hour*, it turns out that when the host does not respond in an hour, the predicted price is higher. The contribution of *host_response_rate* is also puzzling, since it indicates that when the host response rate is higher than average, then the predicted price is lower. One would expect that prospect guests would like more responsive hosts.

To investigate the issue on *time_since_last_review*, I create a dependence plot illustrating the relationship between *time_since_last_review*, *review_scores_rating*, and the SHAP value. The plot is presented below.

Figure 9: Dependence Plot



It is now become clearer why higher *time_since_last_review* increases the predicted price. It can be seen that all of those with the highest *time_since_last_review*, the *review_scores_rating* is also high. This is because the missing value of *time_since_last_review* is imputed to be the maximum of *time_since_last_review* and the missing value of *review_scores_rating* is imputed to be the mean of *review_scores_rating* which is considerably high. On the other hand, those with lower *time_since_last_review* consists of both high and low *review_scores_rating*.

Appendix

The description of variables selected from the original dataset are as follows:

- Information about host
  - Host response time (*host_response_time*)
  - Host response rate (*host_response_rate*)
  - Host acceptance rate (*host_acceptance_rate*)
  - Is superhost (*host_is_superhost*)
  - Total listings (*host_total_listings_count*)
  - Has identity identified (*host_identity_verified*)
- Information about the accomodation
  - Neighbourhood (*neighbourhood_cleansed*)
  - Types of property (*property_type*)
  - Types of room (*room_type*)
  - Number of people that can be accommodated (*accommodates*)
  - Types and number of bathrooms (*bathrooms_text*)
  - Number of bedrooms (*bedrooms*)
  - Number of beds (*beds*)
  - Amenities (*amenities*)
  - Minimum number of night stay for the listing (*minimum_nights*)
  - Maximum number of night stay for the listing (*maximum_nights*)
  - Is availability (*has_availability*)
  - The availability of the listing 30 days in the future as determined by the calendar (*availability_30*)
  - The availability of the listing 60 days in the future as determined by the calendar (*availability_60*)
  - The availability of the listing 90 days in the future as determined by the calendar (*availability_90*)
  - The availability of the listing 365 days in the future as determined by the calendar (*availability_365*)

  - The number of reviews the listing has (*number_of_reviews*)
  - The number of reviews the listing has (in the last 12 months) (*number_of_reviews_ltm*)
  - The date and time this listing was "scraped". (*last_scraped*)
  - The date of the last/newest review (*last_review*)
  - Overall review rating (review_scores_rating)
  - Average accuracy review rating (*review_scores_accuracy*)
  - Average cleanliness review rating (*review_scores_cleanliness*)
  - Average check-in review rating (*review_scores_checkin*)
  - Average communication review rating (*review_scores_communication*)
  - Average location review rating (*review_scores_location*)
  - Average value review rating (*review_scores_value*)
  - Whether the guest can automatically book the listing without the host requiring to accept their booking request. (*instant_bookable*)
  - Description of the accommodation (*description*)
  - Description of the neighborhood (*neighborhood_overview*)

The description of variables created are as follows:

- Number of bathrooms (*num_baths*)
- Number of shared bathrooms (*num_shared_baths*)
- Number of private bathrooms (*num_private_baths*)
- The bathroom is half-bath (*is_half_bath*)
- Time since the last review (*time_since_last_review*) (created from *last_review* and *last_scraped*)
- Is the accommodation near The Metropolitan Rapid Transit (Metro) (*near_MRT*) (created from *overview* and *description* - using the relevant keywords)
- Is the accomodation near The Bangkok Mass Transit System (Skytrain) (*near_BTS*) (created from *overview* and *description* - using the relevant keywords)
- Is the accomodation near Bangkok Airport Rail Link (*near_ARL*) (created from *overview* and *description* - using the relevant keywords)

The rest of variables created are dummy variables from the original categorical variables.