1 Introduction

The goal of this project is to analyze Sydney Airbnb rental data. Providing effective advice to landlords, real estate investors and other stakeholders through building an accurate price prediction model. We implemented comprehensive data exploration and feature engineering to extract important variables related to price, and chose multiple models for predictive analysis.

Data exploration and analysis is the most important process, including data cleaning, visualization, and correlation analysis between variables. Combined with the actual business background to screen out features closely related to price. In order to transform the raw data into features that better represent the essence of the problem and improve the performance and prediction accuracy of the model, we performed feature engineering on multiple variables. Afterwards, we conducted a train test split with a train size of 80% of the overall data.

A variety of machine learning algorithms were used during the model training phase, including regression models, tree models, boosting. We also optimized the hyperparameters of these models. We further tried model stacking technology based on these models to reduce the bias of a single model by combining the advantages of multiple models, it can improve the accuracy and stability of the prediction. Finally, we determined the best prediction model based on RMSE and applied it to help different stakeholders make more informed rental pricing decisions.

1.1 Problem Formulation

In the highly competitive short-term rental market, pricing strategy plays a pivotal role in maximizing occupancy and revenue for hosts. Airbnb hosts must determine an optimal nightly rate considering the unique characteristics of each property such as property attributes, seasonal trends, local demand and its surrounding market conditions. Pricing strategy models can help landlords achieve this goal.Hosts can understand when they can charge higher prices to meet demand, and when they can lower prices to stay competitive and maximize bookings. If landlords plan ahead, they can avoid leaving too many vacancies. They also won't lose profits by charging low prices when demand for properties in your area is huge (Hopkins, 2024).

Airbnb pricing is a prediction problem, as it involves forecasting future rental demand and revenue based on historical data and the property's attributes. Hosts face an uncertainty where they must make decisions to maximize revenue and occupancy while managing risks associated with incorrect pricing. Machine learning offers more accurate price predictions based on data. It helps hosts make better decisions under uncertainty.

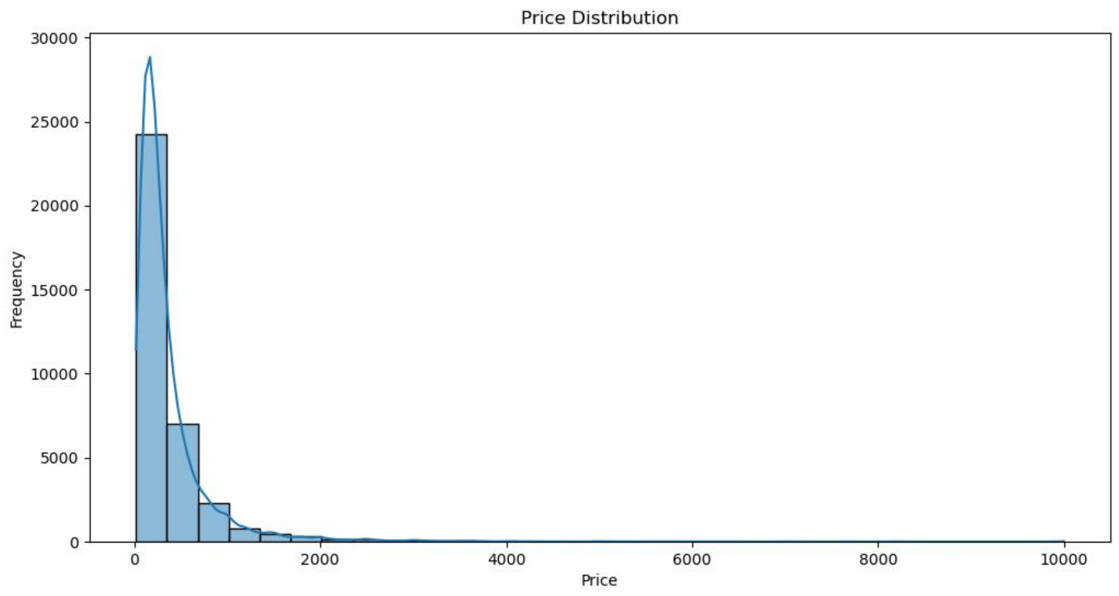
Machine learning models can support Airbnb hosts by generating predictive pricing strategies. These models enable accurate predictions of rental prices using property characteristics such as size, location, and so on. Additionally, machine learning can provide insights into which factor drives rental prices most. So, it can highlight the most influential features. This allows hosts to refine their houses by focusing on aspects that enhance value. Furthermore, it can also facilitate dynamic pricing strategies similar to hotel industries. It allows hosts to adjust rates in response to changing markets, make sure that prices remain competitive. In addition to helping hosts maximize revenue, machine learning can also aid in cost reduction. By forecasting low demand periods, hosts can offer discounts to maintain occupancy. It reduces the costs associated with vacancies. Predictive analytics enable hosts to simulate different scenarios and assess how pricing changes impact revenue. It provides a clearer picture of the financial outcomes of various strategies.

2 EDA

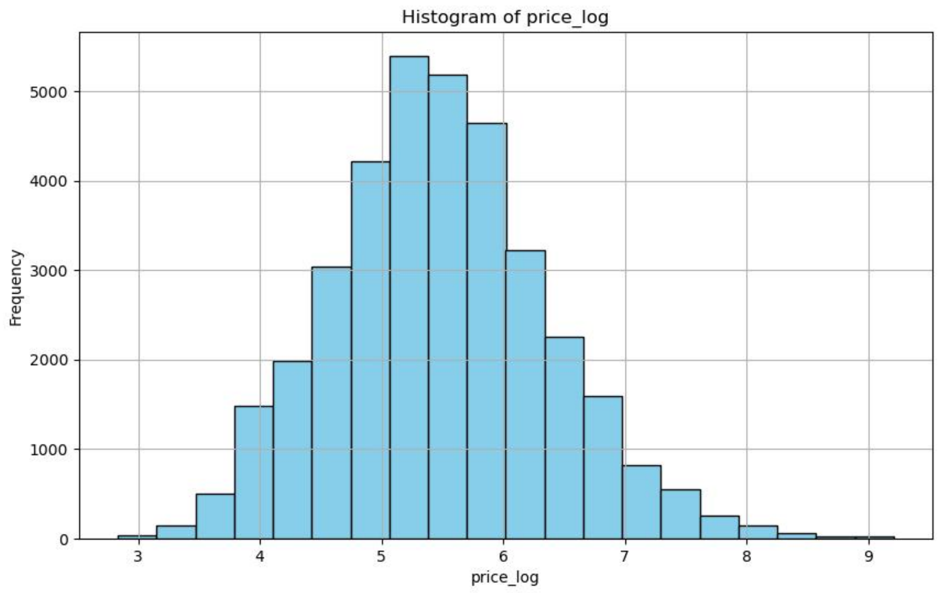
2.1 Target variable processing

In exploratory data analysis (EDA), it is crucial to combine business understanding to conduct preliminary data processing and correlation analysis for subsequent modeling.

The first step is to check for missing values in the price column. The target variable (i.e. price) is the core of subsequent modeling, so it is necessary to ensure the data integrity of this column. After verification, it was found that there were 2006 missing values in the target variable. We chose to delete the missing values instead of filling them in as the average or median of the price. The reason for deleting them is that the price variable has a large variance and uneven data distribution. Even though there were 2006 missing values, the overall data is close to 40000. Deleting them will not result in a shortage of data quantity. In this case, deleting missing values can better ensure the accuracy of data analysis and modeling. Due to the fact that price columns typically contain dollar symbols and other characters (such as commas), we use regular expressions to clear these characters and convert them to floating-point numbers. The purpose of this step is to ensure that the price column can be used as a numerical variable for subsequent statistical analysis and modeling. Next, we explored the overall distribution of the price data column and found that the price data is extremely skewed to the right. Combining common sense and business understanding, the data in the range of rent prices from \$10000 to \$100000 per night is unreasonable. After a detailed analysis of each row of these price data, it is easy to determine that most of them are junk data. Therefore, we deleted the data with prices greater than \$10000. The adjusted price data distribution chart is shown below.



From the graph, it can be seen that the right skewed phenomenon of price data is still very obvious. We perform logarithmic transformation on the data, which can reduce the skewness of the data and make it closer to a normal distribution. In addition, logarithmic transformation can reduce the influence of extreme values, making the data more suitable for training linear or tree models. After completing the logarithmic transformation, we removed the original price column and only retained the logarithmic prices to ensure that the processed values were used for subsequent analysis. The distribution of the converted data is shown in the following figure.



2.2 Numerical

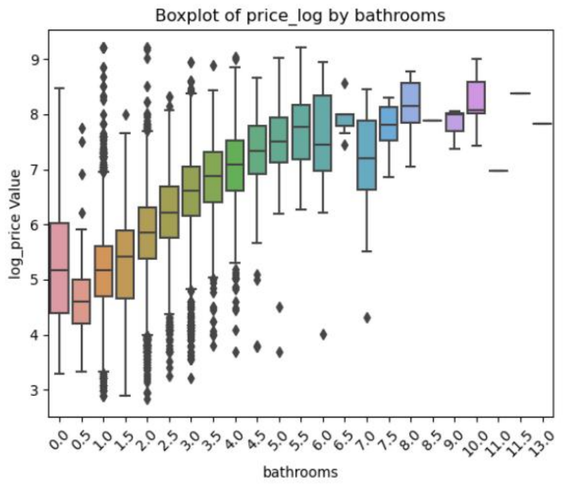
Regarding numerical variables, we first processed the rating columns of some properties, such as overall rating, cleanliness rating, occupancy rating, etc. In order to simplify these ratings and enhance the interpretability of the data, we have reclassified the rating columns. By defining the rescore function, the code divides these ratings into four levels:

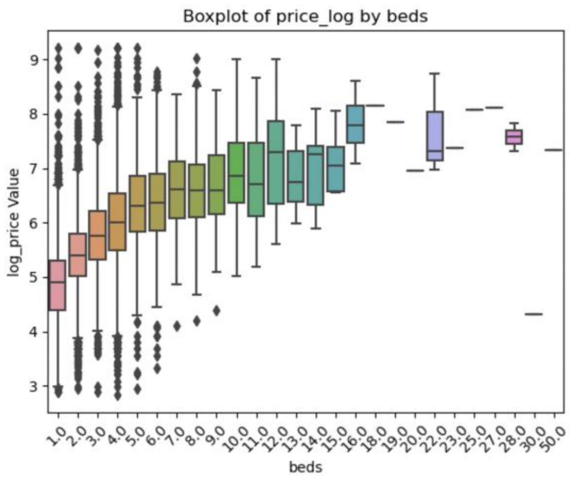
Rating Score	Reclassified rating
Missing values	0
Scores below 4	1
Scores between 4 and 4.5	2
Scores above 4.5	3

The advantage of this approach is that it simplifies the rating into categorical variables, making it easier for the model to process. Meanwhile, these rating columns will affect the pricing of the property, so categorizing them will aid in subsequent regression analysis.

In Airbnb data, bathroom information is usually provided in string form, such as "1.5 bathrooms" or "half bath". In order to convert this information into numerical format, we used a custom function extract.bathroom_numberto extract the number of bathrooms. This function extracts the numerical part based on different input strings and returns the corresponding numerical value. At the same time, if 'half batch' is detected, it will be converted to 0.5. This step ensures that the model can correctly process and understand such features by converting the textual description of the bathroom into numerical data. The following is a box plot of the logarithm of the number of bathrooms and housing prices, in order to more intuitively verify their correlation.

The number of bathrooms shows a positive correlation with log price.It shows that properties with more bathrooms generally have a higher rental price. And as the number of bathrooms increases, so does the median of log price. It indicates that properties with more bathrooms are perceived as more valuable and thus can be priced higher. Similar to bathrooms, the number of beds shows a trend where more beds are associated with a higher log price. This correlation suggests that properties with a higher capacity to accommodate guests can command higher rental prices.





Regarding the four variables related to housing availability: availability 30, availability_60, availability_90, And availability 365, these variables represent the number of available days for the property in the next 30 days, 60 days, 90 days, and 365 days, respectively. In order to simplify the information of these columns during processing, we convert the available days into binary classification: 0 represents unavailable, and 1 represents available.

When dealing with the number of reviews for a property, we have reclassified the number of reviews into the following categories:

Number of Comments	Category
0 comments	category 0
1 to 5 comments	category 1
6 to 26 comments	category 2
Comments exceeding 26	category 3

The number of comments reflects the popularity of the property and market feedback, and reclassification can simplify the feature processing of the model.Finally, we processed the host_desponse_rate column by converting the response rate from a string form (such as "80%") to a floating point number, and then padding the missing values to 0.

2.3 Catagorical

Through observing the 'property_type' attribute, we can find that there are many repeated words in this attribute, so we extract only the last word of each entry. This can help reduce noise in data, so that property types are standardized for further analysis. The final result of the 'property_type' attribute is shown in the below picture, each entry is translated to simple and standardized words.

Property Type	Count	Property Type	Count
unit	15826	chalet	23
home	12462	Barn	22
suite	1149	park	18
townhouse	1078	room	17
guesthouse	980	Campsite	16
condo	738	...	...
villa	473	building	1
cottage	463	hut	1
hotel	431	riad	1
apartment	419		

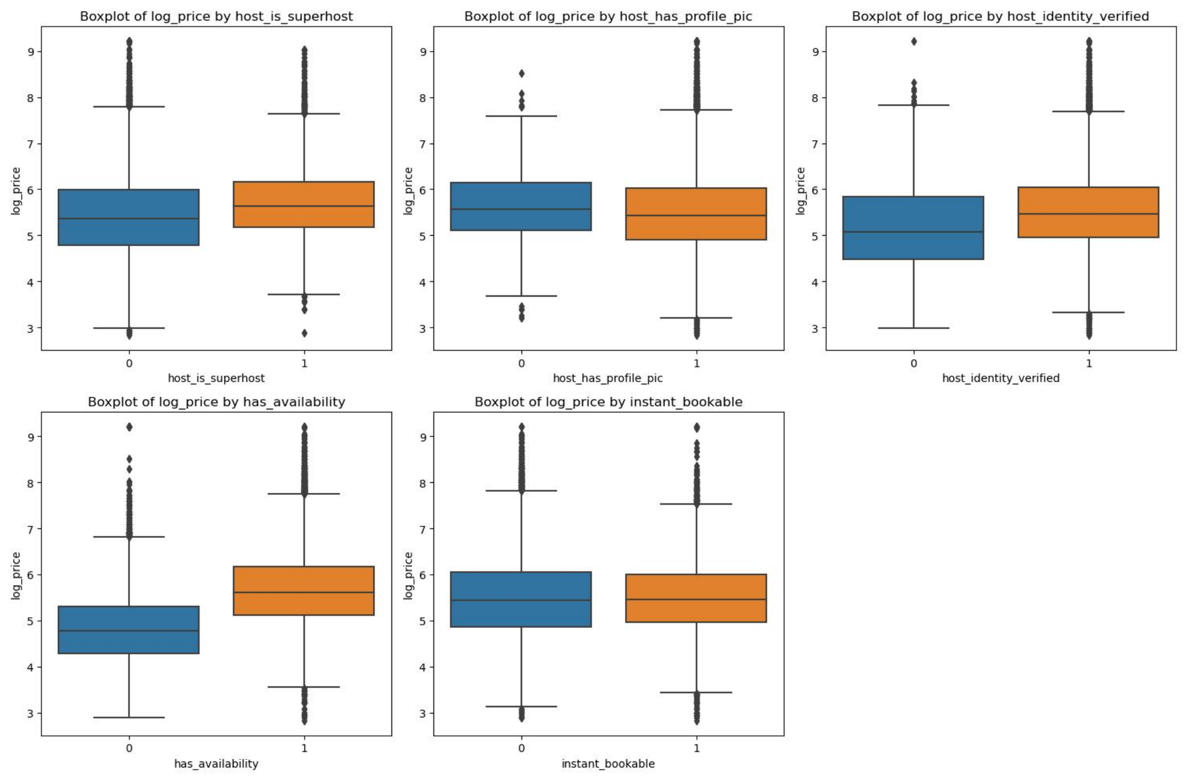
Besides, we can find that the ‘property_type’ attribute has a rare category. The occurrences of "unit" and "home" are over 10,000 times, while "Campsite" and "room" are only a dozen times. The number of different categories has a huge difference with each other. So by iterating through the categories, replacing those with fewer than 100 occurrences with "Other", to avoid data sparsity issues.

2.4 Boolean variable

Next, we will discuss the preprocessing of Boolean variables in the Airbnb dataset and visualize the relationship between different Boolean variables and log price through box plots, conducting correlation analysis between independent variables and target variables.

Firstly, the Boolean variables in the dataset include 'host_is_superhost', 'host_has_profile_pic', 'host_identity_verified', 'has_availability', 'instant_bookable'. We separately checked the value counts of each Boolean variable, which is the number of data classified as' t' or 'f' for each variable. The purpose of this step is to understand the frequency of variable distribution, prevent meaningless variables that are almost entirely 't' or' f ', and to check for missing values. After observation, all 5 variables have missing values. Based on business understanding, if these 5 variables are 'yes', they are likely to have a favorable impact on rent pricing. For example, non-super hosts may not be willing to fill in this option, while super hosts naturally will not miss it. As a result, we choose to fill in the missing values with 'f'.

Because many commonly used machine learning methods such as linear regression and decision trees require data to be numerical, which can significantly improve computational efficiency. At the same time, Boolean variables essentially represent two states and converting them to 0 and 1 not only preserves the original information but also simplifies data processing. Therefore, the next step is to encode Boolean variables, converting 't ' and ' f ' to integers 0 and 1.



Next, we will compare each Boolean variable with log_price using a box plot to examine the impact of different values of Boolean variables on prices. Box plots are used to display the distribution and extreme values of data and are an effective means of classifying variable data. In particular, the horizontal lines at the median can visually display the median differences of variables in different classifications, which facilitates correlation analysis. Below are five box plots of the relationship between Boolean variables and prices.

Combining image analysis, the median, first quartile and third quartile of price data classified as 1 for the variables of super landlord, availability, and landlord identity verification are significantly higher than those classified as 0. This indicates that these three variables are positively correlated with the target variable and are suitable as feature variables for modeling. On the contrary, the variables of landlord resume photos and immediate booking have little impact on prices, as the distribution of price data under different categories is relatively similar, which may be due to various reasons. For example, although houses that can be booked immediately without the landlord's permission are convenient and fast, and are more favored by some people, some landlords hope to communicate with tenants before booking, answer questions, and provide necessary information to enhance the tenant's stay experience, which also makes some guests more preferred. However, these variables will have a certain impact on prices, and from the perspective of improving the accuracy of model predictions, they can all be used as characteristic variables of the model.

2.5 Text Data

'Neighbourhood_overview' column describes in detail the landlord's subjective evaluation of the neighbourhood in which the house is located, including information on the neighbourhood environment, accessibility, amenities, etc. This is an important reference for tenants to decide on a house. This is an important reference for tenants deciding on a house. In our analysis, focusing on the 'neighbourhood_overview' column can effectively capture these neighbourhood characteristics, thus providing strong support for the subsequent model construction.

First, we systematically cleaned and preprocessed the 'neighbourhood_overview' column. In order to eliminate the interference of irrelevant information, we normalise the url, HTML tags and emoticons in the text, for example, replacing all urls with 'xxurl', and common emoticons with 'xxsmile ' and 'xxsadface' to ensure the purity and consistency of text data. Also, all numbers are replaced with 'xxnumber' to ensure that the model is not affected by specific values. We also use the space library to align the text and filter stop words to retain the most valuable information for analysis.

For text feature extraction, we use TfidfVectorizer, which focuses on extracting the TF-IDF values of the binary. By calculating the weights of these phrases, we can determine which phrases are highly discriminatory in the description of landlords. The analysis shows that phrases such as 'walking distance' and 'public transport' appear frequently, indicating that landlords usually emphasise the convenience of location when describing their properties. This information is helpful to find out how the neighborhood environment and surrounding transportation affect tenants' decisions.

Although the Airbnb dataset contains multiple text fields, such as "name" and "description", they are not selected. The reason for choosing to focus on the "neighbourhood_overview" column is that "neighbourhood_overview" provides the landlord's description of various aspects of the community, and its content is more detailed and specific. In contrast, the "description" column focuses more on the internal facilities of the room and lacks the diversity of the community background. In addition, the "name" column is shorter and more difficult to analyze in depth.

In summary, through in-depth analysis of "neighbourhood_overview", we can better discover how landlords attract tenants' interests through community descriptions. From our text analysis, we can see that factors such as convenient transportation and quiet neighborhoods are the focus of tenants. Landlords need to meet these requirements in order to attract more tenants. Subsequent analyses will continue to explore the impact of these descriptions on factors such as listing pricing and ratings, and modelling will further reveal the relationship between these textual features and tenant choice behaviour.

Overall, by analysing the ‘neighbourhood_overview’ field, we were able to reveal the community characteristics of Airbnb listings and hosts' promotional strategies, while avoiding the redundancy of information that could be introduced by other text fields. This provides a clearer direction for our analysis and lays the foundation for subsequent modelling and prediction.

3 Feature Engineering

3.1 Encoder

When analyze the Airbnb Data Dictionary, it can be find that the attributes 'neighbourhood_cleansed', 'property_type' and 'room_type' categorize data based on location, property type, and room type. So it is necessary to apply encoding methodology on these three attributes. Target encoding maps categorical variables to the mean of the corresponding target variables. Specifically,the ‘neighbourhood_cleansed’ and ‘property_type’ can use Target Encoding and ‘room_type’ attribute can be performed One-Hot Encoding.

At the beginning, using the Target Encoding immediately, but when building the tree model and calculating the R^2 and MSE, the $R^2=1$,MSE = 0, this shows that the previous steps have some problem, which causes the data leakage. After careful code inspection, it was found that simple Target Encoding cannot tackle this dataset, so it is necessary to use the K-Fold Cross Validation to avoid the data leakage. So 5-fold was chosen, this a commonly used approach for model validation. For each fold, the data will be split into training dataset and validation dataset, and train the TargetEncoder on training dataset.

In this dataset, ‘room_type’ attribute has four classifications. In order to make the model to understand this attribute easily, One-Hot Encoding is used for the attribute. One-Hot Encoding maps each category to an independent binary vector, which ensures there is no order or size relationship between categories. One-Hot Encoding creates a new feature for every category, which will increase the dimensionality of data, but in our dataset, ‘room_type’ attribute has four classifications, this increase will not affect the performance. Besides, One-Hot Encoding can avoid data leakage. In this process, we used specific columns of the training set to fit the encoder, and then transformed the training and test set separately, using them to generate binary feature matrices.

3.2 TF-IDF

In the feature engineering phase, we performed TF-IDF feature extraction on the neighbourhood_overview column in the Airbnb dataset. This column mainly contains a description of the neighbourhood in which the listing is located, which has a significant impact on the listing price. First, we cleaned the column, filled all missing values with empty strings, and ensured that the column's data type was string for subsequent processing.

Next, we extract features from the text data using the TF-IDF method.TF-IDF can highlight words with important information, especially those specific phrases in the neighbourhood descriptions that may affect the listing price, by measuring the frequency of words in the document and the scarcity of words in the whole corpus. We fit the TF-IDF model on the training set and apply it to the test set to generate the corresponding TF-IDF feature matrices. We then convert these matrices into DataFrame form for use with other numerical features.

In addition to the textual features, we selected a number of numerical features including key variables such as host response rate, listing capacity, ratings and availability. These features are highly correlated with listing prices and can provide the model with important information about host behaviour, listing quality and user experience. After combining the text features with these numerical features, we normalized these numerical features to ensure that the model can correctly handle different types of data.

Finally, after combining text and numerical features, we prepared a complete dataset and added the target variable log_price to it, which allowed us to more clearly discover the relationship between price and text. The TF-IDF method can provide a more complete understanding of how landlords affect house prices through text descriptions, and combined with numerical features, it adds quantitative information to the model.

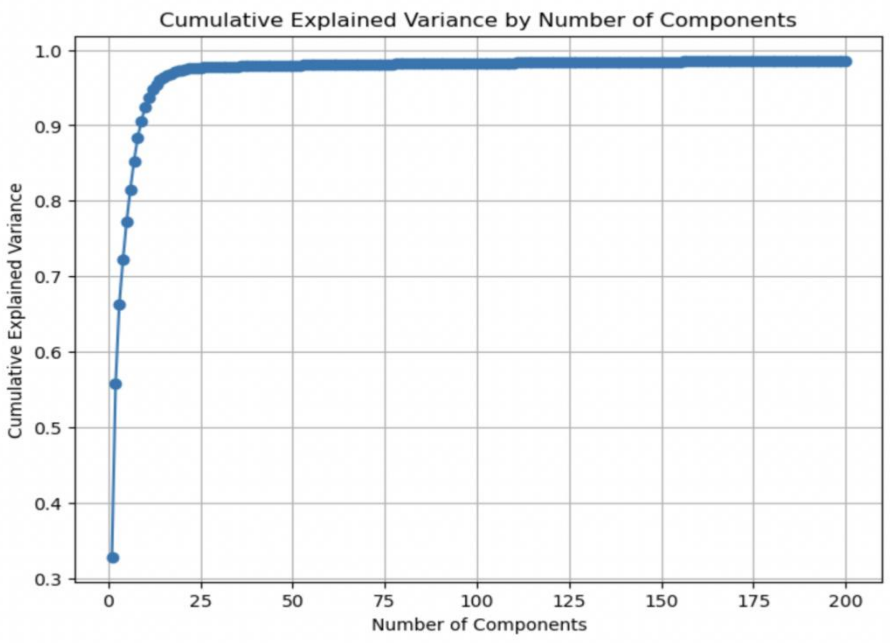
4 Model

4.1 PCA

In this project, Principal Component Analysis (PCA) was employed as a key technique for dimensionality reduction. It is particularly important when working with high-dimensional datasets. In this project, the Airbnb dataset contained a total of 22,789 samples and 14,817 features. Therefore, the dataset is highly dimensional and complex for model training. To ensure the efficiency of models and avoid overfitting, PCA was introduced in modeling. It reduces the number of features while retaining the essential information that drives variance in the dataset.

The motivation of PCA was twofold: Firstly, it aims to improve computational efficiency by reducing the feature space. And second, to mitigate the risk of multicollinearity which can negatively impact the performance of regression models. PCA helps transform the original features into a set of uncorrelated principal components. So that the most important variance is captured while discarding noise and redundant information.

To determine the appropriate number of principal components, we analyzed the cumulative explained variance plot. The plot illustrates how much of the dataset's total variance is captured by increasing numbers of principal components. From the graph, it is evident that the first 100 components account for nearly 98% of the total variance in the dataset. This indicated that retaining 100 components would be sufficient to capture most of the information while drastically reducing the dimensionality. Beyond 100 components, the additional variance explained becomes minimal, indicating diminishing returns.



After applying PCA, the transformed data with reduced dimensionality was used as input for several machine learning models, including Ridge, Lasso, ElasticNet, Decision Trees, and ensemble methods like Bagging and Random Forest. PCA not only helped improve the speed and efficiency of model training but also allowed for more robust and interpretable models by removing irrelevant or less important features.

4.2 Linear Regression

4.2.1 Ridge Regression

Ridge regression is a linear model particularly useful for dealing with multicollinearity by applying L2 regularization. In this analysis, Ridge regression was combined with PCA to reduce the feature space while maintaining interpretability and preventing overfitting. RidgeCV was used to perform cross-validation for optimal alpha selection. The best alpha selected was 1.0, balancing model complexity and fit. The model achieved a Root Mean Squared Error (RMSE) of 0.5715, and an R^2 score of 0.5820 on the validation set, indicating a moderate fit. The Ridge model performed well with PCA, effectively handling high-dimensional data and providing reasonably accurate predictions.

4.2.2 Lasso Regression

Lasso regression, which applies L1 regularization, was combined with PCA in this analysis to improve the interpretability and efficiency of the model. Lasso not only penalizes large coefficients but also performs feature selection by shrinking some coefficients to zero, thus promoting sparsity in the feature space. Using LassoCV for cross-validation after applying PCA, we identified the optimal alpha as 0.01. This relatively small alpha provides sufficient regularization to control for overfitting without overly penalizing the coefficients. The Lasso model achieved a Root Mean Squared Error (RMSE) of 0.6027, and an R^2 score of 0.5352 on the validation set. This performance indicates that Lasso with PCA effectively reduced dimensionality and controlled model complexity, although it still lagged slightly behind the Ridge model in predictive accuracy.

4.2.3 ElasticNet Regression

ElasticNet combines both L1 and L2 regularization, leveraging the strengths of both Ridge and Lasso.

$$\min_{\beta}\left(\frac{1}{2n}\sum_{i=1}^{n}(y_i - X_i\beta)^2 + \alpha\left((1-\lambda)\sum_{j=1}^{p}|\beta_j| + \lambda\sum_{j=1}^{p}\beta_j^2\right)\right)$$

This is the formulation of ElasticNet regression model:

where:

- yi represents the target variable (the log of Airbnb rental prices),
- Xi represents the input features for each observation (such as location, amenities),
- β represents the model coefficients,
- α is a hyperparameter controlling the overall strength of the regularization,
- λ is a hyperparameter (l1_ratio in ElasticNetCV) that balances between L1 and L2 penalties, where $\lambda=0$ is equivalent to Ridge regression and $\lambda=1$ is equivalent to Lasso.

In this analysis, ElasticNetCV used cross-validation to select the best combination of alpha and the L1_ratio, balancing the mix between Lasso and Ridge penalties. The optimal alpha was determined to be 0.01, and the best L1_ratio was 0.1, indicating a stronger emphasis on Ridge regularization (L2) over Lasso (L1). The model produced an RMSE of 0.5884, and an R^2 score of 0.5571 on the validation set. These results show that the ElasticNet model provides a balanced approach to regularization, with robust model performance and predictive accuracy that sits between those of the Ridge and Lasso models. The combination of both penalties enhances model stability and captures a broader range of relationships within the data.

4.3 Tree Model

4.3.1 Analysis of a single decision tree

The reason for choosing a single decision tree model is that it is simple to understand and has intuitive interpretability. Therefore, selecting the decision tree model as the initial attempt to explore the relationship between the distribution of data and features. Decision tree is a data mining method that recursively divides the data set into different subsets to make predictions (Song & Lu, 2015). The model selects an optimal feature for each split so that the data points in the subset have as similar labels as possible. The decision tree finally outputs a tree structure based on feature partitioning, and makes predictions through the leaf nodes of the tree (Suthaharan, 2016).

Building a decision tree regression model and optimizing the hyperparameters by random search. The key hyperparameters of the decision tree model include the maximum depth of the tree, the minimum number of split samples, and the minimum number of leaf samples. Using RandomSearchCV to randomly select 100 parameter combinations within the given parameter range and evaluating these combinations by using cross validation. Finally, got the best model and the RMSE was calculated to be 0.5777 on the validation set.

However, a single decision tree tends to overfit the training data, resulting in large prediction errors on the test set (García Leiva, Fernández Anta, Mancuso, & Casari, 2019). In particular, when the depth of the tree increases, the model becomes more complex.

4.3.2 Bagging model analysis

We chose the bagging method to avoid the overfitting problem of the single decision tree. Bagging trains multiple models through multiple random samplings and then averages their prediction results. This method can effectively reduce the variance of the model and improve the stability of predictions (Bühlmann & Yu, 2002).

The RMSE of the bagging model on the validation set is 0.5603, which significantly reduces the prediction error compared to the single decision tree. Although bagging improves the stability and generalization ability of the model, training multiple models results in greater computational costs and longer training time.

4.3.3 Analysis of Random Forest Model

Random forest increases the diversity of trees and further improves the performance of the model base on the bagging model (Rodriguez-Galiano, Sanchez-Castillo, Chica-Olmo, & Chica-Rivas, 2015). Random forest avoids the over-reliance of a single tree on a few important features by randomly selecting some features for splitting each time a decision tree is built (Paul et al., 2018).

The random forest shows the best prediction performance, it has an RMSE of 0.5448 on the validation set. Random forest not only performs well in capturing complex patterns in the data, but also effectively reduces the risk of overfitting the model.

Fig.1 shows the relationship between the rental price predicted by the model (y-axis) and the actual rental price (x-axis). As can be seen from the figure, most of the data points are closely distributed around the reference line, indicating that the random forest model is relatively accurate in predicting the prices of most properties. However, there are still some data points that deviate from the reference line, especially in the higher housing price range, where some errors may occur.

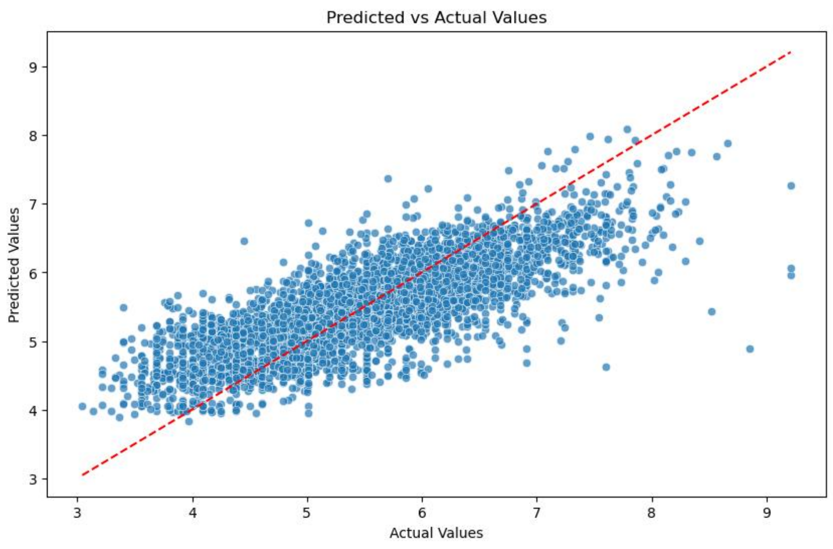


Fig.1 Predicted vs Actual values

Fig.2 shows the 10 most important features in the model and their contribution weights to the model prediction results. As can be seen in the figure, PC3 and PC2 are the most important features, accounting for 16.2% and 11% respectively. They play a decisive role in predicting house prices in the model. This may indicate that these two features contain the most relevant information in the data about the target variable.

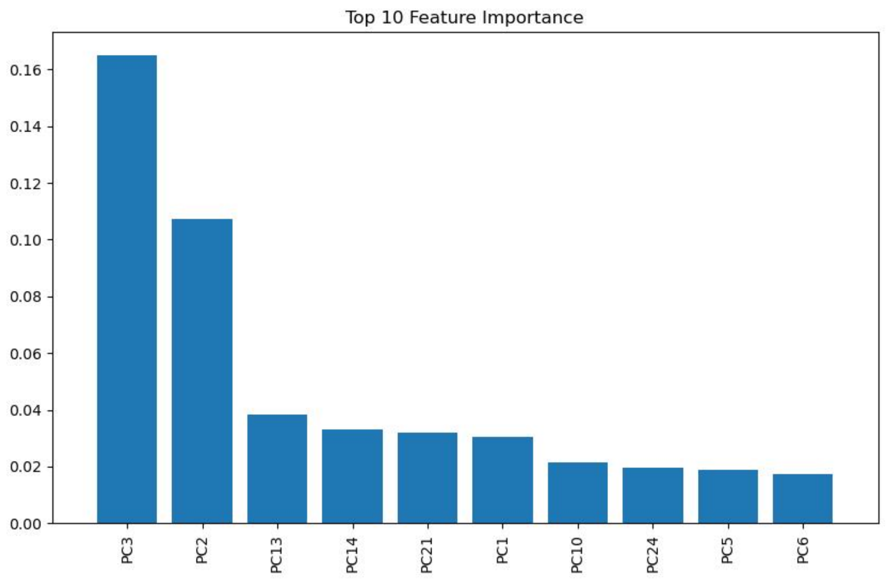


Fig.2 Top 10 Feature Importance

4.4 Boosting

4.4.2 XGBoost

We chose XGBoost as our first model. We set the learning rate to 0.02 to maintain progressive learning and the maximum depth to 6 to balance bias and variance. In this model, we chose 1822 estimators to ensure that the model can learn more detailed patterns in the data. Regularization terms (including reg_alpha = 0.0 and reg_lambda = 0.0) were set to zero because the dataset did not exhibit significant multicollinearity. Next, a subsampling of 0.60 is used to ensure that each tree is trained on a different subset of the data, thus reducing the possibility of overfitting.

XGBoost performs comparable to Gradient Boost, but with faster training time and slightly improved interactive properties of the process. Most importantly, we calculated that XGBoost has the lowest root mean square error (RMSE) of 0.4911, which is the best performance among all models.

4.4.3 LightBGM

In our approach, LightBGM is the final and best performing model. The reason why LightBGM was chosen is that it is an efficient gradient boosting model that can be used to handle large amounts of data while maintaining high accuracy by using the decision tree method. (Ke et al., 2017)) We set several hyperparameters to optimize the performance of this model. The learning rate is set to 0.01 and num_leaves is set to 25, which ensures that the model can continuously learn from the data and reduce the possibility of overfitting, and the complexity of the tree structure is controlled by limiting the number of leaves in each tree. In addition, L1 regularization (lambda_l1 = 3.60) and L2 regularization (lambda_l2 = 0.0387) are set and large lambda coefficients are penalized to prevent overfitting. Then, the bagging fraction is set to 0.799 to randomize a portion of the training data in each iteration. The result of Feature_fraction is 0.577 to ensure that the model does not rely too much on a specific feature. Finally, the lowest root mean square error (RMSE) of LightBGM is calculated to be 0.6194.

4.3.4 CatBoosting

CatBoosting is used for regression tasks and combines it with the Optuna for hyperparameter optimization. Several steps were set to finish this process, from data preprocessing, basic parameter settings, hyperparameter optimization, model training, and finally evaluated the performance of the model. Optuna is a tool which can do hyperparameter optimization automatically.The error changes of the model on the training dataset and validation dataset were plotted to assess the model's performance.

Before model training, it must prepare training dataset and validation dataset. So CatBoost's Pool data structure was used to process data and labels. Pool is a specific data structure for CatBoostused to store data and labels. Then, to define CatBoost model parameters,the parameters of the CatBoost model were set to optimize the performance of the model. These parameters were set: ‘iterations’: Maximum number of iterations (2000 times), ‘learning_rate’: Learning rate, which controls the step size of each model update, ‘loss_function’ and ‘eval_metric’: Use RMSE as the loss function and evaluation metric for the regression task, ‘random_seed’: Ensures the experiment is reproducible, ‘verbose’: Controls the frequency of output during the training process.

Then, using Optuna to do hyperparameter optimization. Optuna will calls the objective function on each trial, during the parameter search, trial generates a set of random parameter combinations through the suggest method, such as ‘depth’、’l2_leaf_reg’ and ‘bagging_temperature’ parameter. Then train the model using the generated parameter combination and evaluate the model on the validation set with RMSE metric. Following this, the optimal parameters and number of iterations were identified, with the output indicating that the best iteration count is 1998. The best parameters can then be used to retrain the model for improved performance. Train the model again and set the early stopping (early_stopping_rounds=100) to avoid overfitting.

Finally, extracting the RMSE value (Validation RMSE is 0.503) and plot the RMSE curves of the training set and the validation set to observe the error changes of the model under different iteration numbers, the curve is shown in Figure 4.1. In this picture, it can be found that within the first 250 iterations, the RMSE for both the training and validation sets decreases rapidly, the blue line represents that the training error gradually decreases, showing the model fits the training data better over time. And the Orange Line shows that the validation error stabilizes after around 500 iterations.

4.5 Model stacking

One important advantage of stacked models is to utilize the diversity of different models to improve overall predictive performance. Therefore, choosing different types of models can help improve prediction accuracy, but the quantity should not be too large, which can easily lead to overfitting and greatly increase computational costs. In the previous coding work, we obtained three linear regression models, three tree models, and three boost models, which can be said to have laid a good foundation for model stacking. Firstly, regarding the selection of the basic model, we believe that one model can be chosen from both the boost model and the tree model, namely Random Forest and LightGBM. This not only ensures model diversity based on existing models, but also reduces computational costs and the possibility of overfitting.

The meta model is used for secondary learning of the output of the base model. The main task of the meta model is to integrate the prediction results of different base models and utilize the complementarity between the base models to improve the final prediction performance. It should be a simple and well generalized model to avoid overfitting again. Elastic Net regression is a linear regression model that combines L1 regularization (Lasso) and L2 regularization (Ridge), with wide applicability. Therefore, we chose the Elastic Net linear regression model as the original model.

5 Results

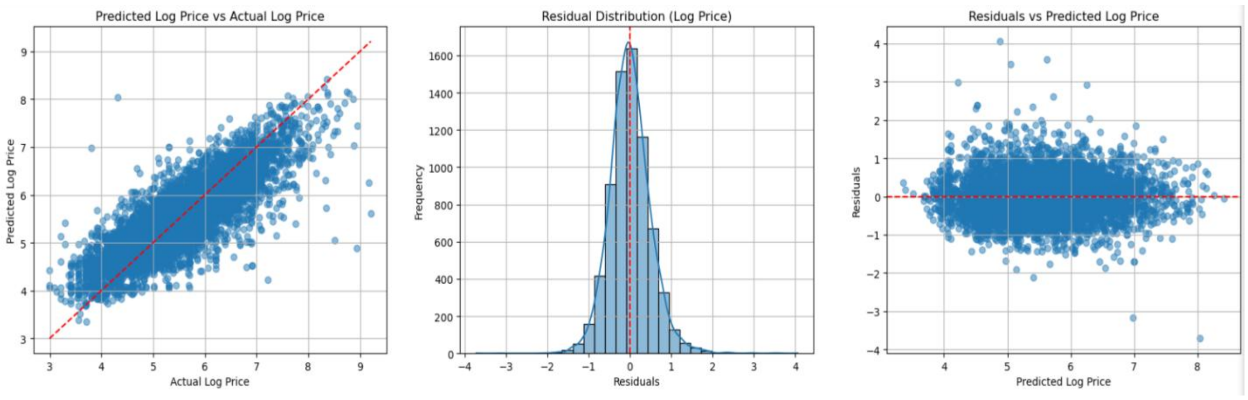
This article first conducted a detailed exploratory data analysis of Airbnb rental data in Sydney, including data cleaning, target variable preprocessing, inter variable correlation analysis, and data visualization. Before modeling, we conducted feature engineering using target encoding, TF-IDF, and One Hot encoding to transform and extract important features. In addition, we also used principal component analysis (PCA) for dimensionality reduction to improve the computational efficiency and stability of the model.

In the modeling process, we tried various models, including linear regression models (ridge regression, Lasso regression, ElasticNet), tree models (decision tree, bagging, random forest), boosting models (XGBoost, LightGBM, and CatBoost), and stacked models. We position the ElasticNet linear regression model as a benchmark and use the RMSE metric as the model evaluation criterion. The predictive performance of 10 models under the validation set is summarized in the following graph.

Model Name	RMSE
Ridge Regression	0.5715
Lasso Regression	0.6027
ElasticNet	0.5884
Decision Tree	0.5777
Bagging	0.5603
Random Forest	0.5448
Catboosting	0.5082
XGBoost	0.4910
LightGBM	0.6194
Stacking	0.4990

Regarding model selection, we selected XGBoost with the smallest RMSE as the final model and combined its training set and validation set for re fitting. Finally, the prediction results on the test set showed that the RMSE of the final model was 0.5024, and the square of R was 0.6881.

We also created additional graphs to further evaluate the final model, including the relationship between predicted logarithmic prices and actual logarithmic prices, residual distribution graphs, and the relationship between residuals and predicted logarithmic prices.



The red dashed line in the left image is the reference line (45 degree line), indicating the position where the predicted value is equal to the actual value. From the graph, it can be seen that most of the data points are near the reference line, indicating that the model's overall prediction is accurate. However, in the high price area, the data points deviate from the main line, indicating that the model has some errors in high price housing. In an ideal situation, the residuals should be close to a normal distribution and the mean should be close to zero. The middle picture perfectly matches this point. The horizontal axis in the right figure represents the predicted logarithmic price, and the vertical axis represents the residual. The red dashed line is the zero reference line. From the graph, it can be seen that the residuals are evenly distributed throughout the entire predicted price range, without any obvious trends or patterns. This reflects that the model does not exhibit systematic errors with price changes and has good generalization ability.

As a result, the final model has good predictive performance overall, but there is still room for further optimization. For example, we can conduct more detailed EDA to select more suitable feature combinations, obtain more data, especially high price housing data, to improve the predictive ability of the model in high price areas, adjust the parameters when constructing the model, or learn to use a more suitable model for this dataset to further improve model performance.

6 Business recommendations

6.1 Quantitative analysis of Boolean variables to provide recommendations

Statistical summary of Boolean variables and log price relationship

Metric	host_is_ superhost=0	host_is_ superhost=1	host_identity_ verified=0	host_identity_ verified=1	has_ availability=0	has_ availability=1
Mean	5.40538	5.700586	5.160821	5.506294	4.873809	5.653535
Median	5.35422	5.634790	5.075174	5.463832	4.787492	5.598422
Variance	0.82408	0.615655	0.888039	0.770182	0.656908	0.690992
Min	2.83321	3.401197	3.091042	2.833213	2.944439	2.833213
Max	9.21034	9.01918	9.21034	9.21034	9.21034	9.21034
Count	18994	5932	2212	22714	5687	19239

According to the above statistical table, it can be concluded that the average and median rental prices of super landlords are higher than those of non super landlords. The visualization of data in the previous EDA clearly shows that the quartiles of super landlord rental prices are higher than those of non super landlords. At the same time, the variance of super landlords is relatively small, indicating that landlord price fluctuations are relatively small. This shows that landlords have more stable pricing after becoming super landlords, which is conducive to improving income and increasing income stability. Similarly, the analysis of super landlords is fully applicable to landlords who pass identity verification.

As a result, landlords should strive to become super landlords by providing high-quality services, actively and quickly responding to messages, such as responding to over 90% of messages within 24 hours, avoiding canceling reservations proactively, requesting guests to evaluate their accommodation experience after check-out, and striving to obtain an average rating of over 4.8 out of 5, which can also increase the exposure of the property. In addition, landlords should take the initiative to verify their identity, which can not only increase users' trust in the property, but also help landlords increase the price of the property.

There may be contradictions and difficulties regarding the bookable status of the house. Although quantitative analysis of the table shows that being in a bookable state is beneficial for price increases, it also indicates that being in a vacant state is not conducive to maximizing income. Therefore, the key is to optimize the bookable status and pricing strategy of the property to increase occupancy rates and revenue. Landlords should flexibly manage the status of their properties and set reasonable pricing strategies, such as implementing dynamic pricing, appropriately lowering prices during off-season and holidays, increasing prices during peak season and canceling weekly and monthly discounts, etc. In addition, landlords should set up flexible cancellation mechanisms, which can not only increase customer appeal but also quickly return the property to a bookable state.

6.2 Detailed property description

Detailed neighbourhood descriptions and location advantages are one of the effective strategies for top landlords to attract tenants. By analysing the neighborhood_overview text data, we found that landlords' descriptions of the property's surroundings have a direct and significant impact on tenant choice. The best performing landlords usually provide detailed surrounding amenities in the property description, such as restaurants, supermarkets, and public transportation near the house. This information can help tenants quickly understand the advantages of the property location and play a positive role in promoting tenants to book a house.It is worth mentioning that if the landlord can also update the transportation network near the property, such as bus stops, subway lines or walking time to these transportation locations, it will also increase the attractiveness of the property. Because this information is very important for first-time tourists to the city. Clearly indicating whether your property is close to popular dining areas, shopping centres or cultural attractions can help renters better plan their trip and raise their expectations of the property.

For this reason, we recommend that landlords describe their properties in as much detail as possible to emphasise the community's accessibility, accessibility to amenities, and the safety of the neighbourhood. Especially for first-time guests, providing this detailed information can increase their trust in the property. In addition, landlords can further enrich the tenant's stay by sharing information such as local travel advice or recommendations of popular restaurants and cafes. Such detailed descriptions not only help attract more tenants to book, but also increase rebooking rates, making tenants willing to choose the property again in the future. By providing tenants with clear information about the neighbourhood and a guide to local life, landlords can effectively increase the attractiveness of their listings, enhance the tenant's experience and gain an edge over the competition.

6.3 Improving Cleanliness and Hospitality

Through analyzing the ‘review_scores_cleanliness’ attributes and ‘review_scores_rating’ attributes, it can be found that host ratings are highly related with the room cleanliness. High review_scores_rating always has good cleanliness. Regular cleaning is very important, any customer likes a clean and tidy room, so landlords are recommended to consider cleaning more frequently, especially in case of high frequency booking. Good environment can provide a chill feeling to customers. In addition, it is recommended that landlords can regularly check the facilities of the property. This will ensure that all equipment (such as air conditioners, water heaters, kitchen appliances, etc.) are functioning properly, to avoid negative reviews.

While ensuring that it provides the basic facilities, the landlord can also consider adding some additional facilities. Landlords can give customers a warm and special welcome when they move in, such as preparing a welcome card, local snacks or small gifts, which can make them feel the warmth of home and the landlord's care. This is also a good method to improve the review scores.

Reference

Bühlmann, P., & Yu, B. (2002). Analyzing bagging. The Annals of Statistics, 30(4). https://doi.org/10.1214/aos/1031689014

García Leiva, R., Fernández Anta, A., Mancuso, V., & Casari, P. (2019). A Novel Hyperparameter-Free Approach to Decision Tree Construction That Avoids Overfitting by Design. IEEE Access, 7, 99978–99987. https://doi.org/10.1109/ACCESS.2019.2930235

Hopkins, J. (2024, July 28). *Vacation rental pricing strategy: How to find the optimal rates to maximize occupancy and revenue*. Hostfully. https://www.hostfully.com/blog/vacation-rental-pricing-strategy/

Paul, A., Mukherjee, D. P., Das, P., Gangopadhyay, A., Chintha, A. R., & Kundu, S. (2018). Improved Random Forest for Classification. IEEE Transactions on Image Processing, 27(8), 4012–4024. https://doi.org/10.1109/TIP.2018.2834830

Rodriguez-Galiano, V., Sanchez-Castillo, M., Chica-Olmo, M., & Chica-Rivas, M. (2015). Machine learning predictive models for mineral prospectivity: An evaluation of neural networks, random forest, regression trees and support vector machines. Ore Geology Reviews, 71(71), 804–818. https://doi.org/10.1016/j.oregeorev.2015.01.001

Song, Y.-Y., & Lu, Y. (2015). Decision tree methods: applications for classification and prediction. Shanghai Archives of Psychiatry, 27(2), 130–135. https://doi.org/10.11919/j.issn.1002-0829.215044

Suthaharan, S. (2016). Decision Tree Learning. Machine Learning Models and Algorithms for Big Data Classification, 36, 237–269. https://doi.org/10.1007/978-1-4899-7641-3_10

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T. (2017). LightGBM: A highly efficient gradient boosting decision tree. *31st Conference on Neural Information Processing Systems (NIPS 2017)*. Long Beach, CA, USA. https://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree