**Airbnb Sydney Machine Learning Task**

*Huong Ly Hoang*

**TABLE OF CONTENTS**

# 1. INTRODUCTION

## 1.1. Executive summary

Machine learning (MI) is a type of Artificial Intelligence (AI) where it allows software applications to improve automatically through past experiences (Jordan & Michell, 2022). Nowadays, MI can be referred to as the 'first-class ticket' in data analytics and as a 'harmonious' blend between the field of computer science and statistics.

Meanwhile, Data Mining is referred to as "knowledge discovery from databases" (Chen, Hand & Yu, 1996), where its objective is to identify meaningful insights and valid information from the speculation of existing patterns within the databases (Jackson, 2002).

In this project, both Machine Learning and Data Mining techniques will be actively utilised on the given dataset of Airbnb Rentals in Sydney to extract useful findings and further provide market recommendation to existing and potential hosts, real estate investors, and other stakeholders. The platform used for task execution is Google Colaboratory and the outcomes will be reported under two-specified broad sections: Machine Learning and Data Mining.

## 1.2. Background context

Airbnb is a vacation rental company, established since 2008, where its primary product is the online platform that connects property owners who would like to rent out their spaces, and travellers who look for a place to stay. The type of accommodation provided by Airbnb can be a whole property such as a house, an apartment, etc. or a room like a private room, shared room, double room, etc. with different amenities and features, unique to each accommodation. From the 'guest' perspective, Airbnb provides the largest range of accommodation worldwide, along with a various price range to choose from and a community to interact with, making user experiences with the platform unique from traditional hospitals or accommodation providers. From the 'host' perspective, Airbnb provides an easy-to-use, established platform with streamlined processes and great rental potentials from Airbnb's large database of users. Therefore, its unique characteristics and its advantage as a pioneer within its own segment has made Airbnb one of the largest providers of overnight accommodation.

# 2. PROJECT FORMULATION & OBJECTIVES

## 2.1. Project formulation

From a client perspective, the main problem in understanding and predicting Airbnb rentals prices is due to the complex relationships between each accommodation prices and each of its unique features. For example, this can include varying relationships between prices & location, prices & property type,

etc. and since rental prices are affected by all of its unique attributes and not a particular single one, intuitive judgements on Airbnb rental prices is not possible and thus, the utilisation of computer science techniques like Machine Learning is required to derive such insights. In addition, if key drivers of Airbnb rental prices are identifiable, drivers of rental revenue and potential rental revenue will also be predictable. With this potential information, our clients such as hosts, property managers and real estate investors will be able to appropriately allocate more effort into profit maximisation factors and less into insignificant factors or even better, avoiding practices that guests do not favour.

### 2.2. Objectives

The primary objectives of this project include building a 'state-of-the-art' predictive machine learning model for Airbnb rental prices and uncovering at least three meaningful insights that can help hosts make better decisions in managing their Airbnb property, to further enhancing competitiveness of existing and potential listings and ultimately, increase profits.

In the fulfilment of these objectives, regressions models to be used will include *one* linear regression model, *three* tree-based models and *one* model stack. Particularly, these models are respectively Multiple Linear Regression, Extreme Gradient Boosting (XGBoost), Light Gradient Boosting (LightGBM), Cat Boost and a model stack of all mentioned tree-based models.

In addition, various Exploratory Data Analysis will also be used to accomplish the objective of Data Mining.

## 3. MACHINE LEARNING

### 3.1. Data pre-processing

#### 3.1.1. Understanding the original dataset

The original dataset has 74 attributes (Appendix 1), with each column corresponding to a different attribute. Alternatively, each row of the dataset corresponds to an Airbnb listing. Since the dataset is scraped directly from Airbnb's website, it contains various redundant columns with a lot of missing values and unnecessary information, which requires careful cleaning procedures before the fitting and predicting processes. Examples of redundant columns that can be removed include listing URL, licence, etc.

### 3.1.2. Data cleaning

The data cleaning process starts from manually removing intuitively redundant columns and creating a new listings dataset with more relevant columns. After the first pick, the number of attributes reduces from 74 columns to only 23 columns. The following action includes converting the attributes to its relevant format such as converting *price* from 'string' to 'float', *host_since, first_review, last_review* and *last_scraped* from 'string' to 'datetime' format.

Since as an Aibnb's culture, guests always leave their review after each stay, a new column called '*listing_duration'* was also added by the subtraction of *first_review*, indicating the first time the property was rented out and *last_review*, indicating the most recent time the property was rented out. Similarly, *hosting_duration* was also created by subtracting *host_since* from *last_scraped*. The reason behind creating this new information is due to the intuitive hypothesis that a more experienced host and a longer listing's existence might lead to better hosting practices and thus, higher prices or occupancy rates. After the calculations, all the variable columns used within the computation are removed since it will no longer be useful and the newly created columns are changed from 'datetime' to 'float' format.

Since information about bathrooms - one of the main contributors to price outcomes are usually provided within the description of the listing, number of bathrooms is under text format (i.e. *bathroom_text*) and must be converted to 'string' format with removal of other unnecessary test information such as "ensuite bathroom", "shared bathroom", etc. The column will also be re-named as *bathrooms*. For the missing values of *bathrooms*, it is replaced with its median value. Similarly, amenities play a major role in price contribution and must also be converted into 'float' format for prediction. Thus, *amenities* was also converted to 'float' format, removed unnecessary text information, filled in missing values using its median and finally, was re-named to *amenities_count*.

For other columns such as *hosting_duration*, *reviews_score_rating*, *listing_duration* and *reviews_per_month*, their missing values were also replaced with their median values. Missing values in categorical columns such as *host_is_superhost* is replaced with 'f' for false since it is assumed that the chance of host not being a superhost is higher.

After the data cleaning process above, the number of attributes is reduced to 21 columns with 0 missing values and is ready for further implementations.

### 3.1.3. Exploratory Data Analysis (EDA)

An initial summary statistics of continuous variables (Appendix 8) in the dataset reveals some notable information. For example, the average listing price for Airbnb in Sydney is $251, average minimum nights stays per listing is 63 days (approximatley 2 months) and the average maximum nights stay per listing is 906 days (nearly 2 years). In addition, the average hosting duration is 5.7 years and and average listing duration is 1.3 years. This generic summary vaguely portrays that the listing market of Airbnb in Sydney is concentrated with experienced hosts and sustainable listings rather than being seasonal.

In terms of the covariances between variables, a pairwise correlation matrix (Appendix 7) depicts relatively strong independence between the attributes. There only exists a few, highly correlated variables such as *bedrooms* and *accommodates* or *reviews_per_month* and *number_of_reviews*. These covariances within these pairs are reasonably explained based on their nature and thus, can been easily dealt with based on intuitions in the feature engineering sections. Notably, the target variable *price* is not significantly correlated with any other single variable, further underlying the low risks of multicollienarity issues and highlight the reliability of the dataset. Further referring to Appendix 9, it can be observed that even with the most correlated attributes with *price* (i.e. *bedrooms, accommodates* and *bathrooms*), the linear relationship is relatively weak, proposing the potential non-linear relationships between *price* and other attributes that needs to be explored.

In addition, variations within the variables itself can be evaluated via the coefficivent of variation (CV) in Appendix 10. By inspecting the most 5 important attributes (i.e. *bedrooms, bathrooms, accommodates, price, review_scores_rating*) based on intuitions, it was found that most variables have quite a stable degree of variation compared to its own mean since its CV are all smaller than one. However, *price* is relatively unstable in the fact that its CV is at 1.99. This figure is not only greater than one (the 'safe' boundary based on the rule-of-thumb) but is significantly high at nearly doubled the 'safe' level. Referring back to Appendix 9, *price* is extremely right skewed with a large right-tail. While it's self-explainable that price can have wide range of variation, this nature must be dealt with before the implementation of any statistical learning models. Otherwise, a relatively large degree of inaccuracy will be the tradeoffs.

## 3.2. Feature engineering

Feature engineering refers to the process of selecting the appropriate sets of predictors for the intended statistical learning processes. While one set of predictors can be chosen for both linear

regression model and tree-based models for simplicity, this process is decided to be separated. This practice is reasonable since there are generally five assumptions to be satisfied in choosing predictors for Odinary Least Squares model, while in tree-based models, there exists no assumptions. This is made possible since tree-based models are robust to outliers and do not require normality assumptions as well.

However, some shared feature engineering steps include log-transformation of *price* to remove skewness and improve accuracy, label encoding for *neighbourhood*, *room_type*, *property_type* and get dummy variables for binary categorical variables such as *instant_bookable*, *host_is_superhost*, etc. In addition, the response variable (y) for both types of model is decided to be *log_price* since while normality is not required in tree-based models, using a normalised response variable improves most of statistical learning models.

The training and testing dataset will also be splitted in the same ratio, such as 80:20. This means that 80% of the dataset will be used for the learning process and the remaining 20% will be used for validating/testing process.

#### 3.2.1. Mutiple Linear Regression (MLR)

Explanatory variables for MLR were chosen using both backward and forward selection functions. Backward selection function starts with a full model, containing all possible variables, then is removed one-by-one using the R-squared statistics of the model as the valuation metric, until all remaining variables are considered to have some significant contribution to the outcome (Chowdhury & Turin, 2020). In contrast, forward selection function starts with a null model and gradually adds in new variable if it improves the valuation metric (R-squared) of the model. While these two models are highly sufficient in selecting the 'best' subset since they can accommodate searching through a large set of potential predictors, the major drawbacks lie in the fact that once a new variable is added or removed, the action cannot be reversed. This is a disadvantage since a dropped variable in backward elimination process can become significant later in the model, and it's vice versa with the forward selection. Despite the disadvantages, these selection functions are still highly valuable and the same set of predictors derived from both functions include: ***accommodates, room_type, longitude, latitude, minimum_nights, maximum_nights, calculated_host_listings_count, number_of_reviews, amenities_count, property_type, instant_bookable_t, review_scores_rating, neighbourhood, hosting_duration, number_of_reviews_ltm, host_is_superhost_f*** and ***reviews_per_month*.**

### 3.2.2. Tree-based models

As mentioned above, tree-based models do not have any assumptions, therefore, using the set of predictors selected by backward or forward selection functions is not appropriated since these methods were designed for linear regression. Thus, the set of predictors for tree-based models will be selected based on intuitions and observation of pairwise correlation between the potential features and *log_price*. The final set of predictors is then arrived at: ***accommodates, bedrooms, bathrooms, longitude, latitude, amenities_count, room_type, property_type*** and ***review_scores_rating.***

The reason for choosing *accommodates, bedrooms* and *bathrooms* is quite straight-forward, since these variables are the most correlated with *log_price*, with correlation coefficient all greater than 0.4. From intuitions and inspections back in the EDA section, *amenities_count*, *review_score_rating, property_type* and *room_type* are also believed to be strongly associated with price-setting since for example, a whole apartment listing will be more expensive than a shared room listing. Similarly, *longitude* and *latitude* are also included since location is also believed as a prime determinant of price. This idea will be further uncovered in the data mining section later, where it will be founded that listings near the famous beaches areas on average, have higher prices per night.

## 3.3. Methodologies

### 3.3.1. Multiple Linear Regression (MLR)

MLR is one of the most basic models in statistical learning where it just follows the same basics as the simple linear regression, but expanded to allow for variables to be the explanators of the final response variable. Since MLR is a supervised learning, there are 5 assumptions for a MLR, which include:

1) Linearity: There exists a linear relationship between the response variable and each predictor.
2) Normal distribution: All predictors and the response variable follow a normal distribution.
3) Independent and identically distributed: All predictors must be identically distributed, but independently observed from each other.
4) No multicollinearity issues: Predictors must not be strongly correlated to each other.
5) Homocedasticity: Residuals must have constant variance across the linear model.

If one of these assumptions is violated, the outcomes from the MLR will be unreliable. In addition, since MLR is the simplest and the easiest to fit, it will be the benchmark model.

In the context of our dataset, assumption 1 is relatively weak since while most of the predictors do have linear relationships with the response variable, some of the predictors completely do not, such as *longitude* or *latitude*. These are the coordinates of the listings' location, therefore, it is clear that "as the longitude and latitude statistically increase, prices do not necessarily increase". Similarly, the chosen subset also contains some of the categorical predictors such as *host_is_superhost_f*. While this variable has been transformed using dummy variables, it does not truly relate to the response variable in a 'linear' manner. Thus, it can clearly be deduced that assumption 1 is relatively not satisfied.

Similarly, assumption 2 is also not reasonably satisfied. Most of the predictors do not follow normal distribution, including variables such as *review_scores_rating*, *accommodates, minimum_nights* or *maximum_nights*. Even by intuition, for example, it can be understandable that the number of accommodates will follow the common house sizes in Sydney while there only exists a small number of accommodations that can accommodate a large number of people. Therefore, the distribution of this variable will tend to be right-skewed and cannot satisfy the required assumption. Additionally, referring to Appendix 13, the box plot clearly depicts the left-skewed nature of the rating scores. While these variables can be log-transformed to meet the assumption required by the linear regression, transformation only complicates the model and diverts the predictions further away from the 'true' reflection.

In contrast, assumption 3 is strongly satisfied since the dataset was scraped directly from the Airbnb website, deducing that the observation of each data point is 'raw', automatically recorded by the computer algorithms, and is not dependently recorded or distributed based on any other observation.

Initially, assumption 4 is believed to relatively satisfied since by referring back to Appendix 7, the correlation heatmap did not record any significant correlation between predictors. However, variance inflation factor (VIF) calculated for each predictor in the subset reveals that there exists a few 'heavily' correlated factors such as *accommodates, room_type*, *property_type* and *reviews_per_month*. These attributes respectively have a VIF equal to 4.7, 12.4, 11.7 and 5.6. Besides, the average VIF of the model is also at 3.2. Based on a rule-of-thumb, any VIF that is greater than 3 is considered 'high'. While the inflated-nature of these variables are self-explainable since for example, the number of accommodates will be dependent on the property type and the room type, each of these variables still hold a distinctive value towards the valuation of listings price. Therefore, while removing one of these variables will satisfy the assumption 4 required, it will come at the costs of perhaps, unreliable predictions. In addition, while the average VIF of the model is not highly concerned yet, the overall

dissatisfaction of the assumption still suggests a direction shift towards other statistical learning methods.

In contrast to all previous assumption, assumption 5 seems to be reasonably satisfied. This is exemplified via the MLR's residual plot in Appendix 14, where it is clearly observable that there is no trend or patterns in the distribution of residuals across each fitted values. The residuals rather spreaded across the range of (-2, 2), clearly deducing the satisfaction of constant variance.

Therefore, after carefully inspecting each assumption required by a basic MLR, it can be concluded that assumption 1, 2 and 3 are relatively unsatisfied. While assumption 4 is also not fully satisfied, it still can be accepted since the overall performance of the assumption was not highly concerned. Out of all assumptions, only assumption 5 can be said as reasonably satisfied. Thus, based on the judgement of assumptions, it can be quite clearly deduced that MLR will not be a 'strong' model to explain and predict Airbnb's listing prices. This idea can be immediately confirmed by the R-squared of 64.9%, derived from training the dataset using MLR. This statistic reflects that only 64.9% of the response variables were able to be explained by the predictors. While this outcome is not statistically poor, it can be considered as not efficient since the number of predictors as input was quite large for such an 'average' outcome. The prediction result will be discussed in the next section in comparison with other model types. Ultimately, at the current outlook of the MLR, it is strongly suggested that other types of models can be attempted to seek further improvement in performance.

### 3.3.2. Extreme Gradient Boosting (XGBoost)

XGBoost is essentially a Gradient Boosting Decision Tree (GBDT), but it strives for maximum speed and efficiency. While XGBoost has some advantages including bringing high predictive accuracy, regularisation ability and ability to be applied to a large dataset, the main disadvantage lies in the computational costs. Particularly, XGBoost grows its trees horizontally (Appendix 18) and on a level-wise. Therefore, for each split, a new tree is constructed and a new function is required to fit the residuals of the previous prediction (Chen, & Guestrin, 2016) and thus, making this process slow and 'lengthy' in comparison to newer algorithms such as LightGBM.

In order to optimise this algorithm, its hyperparameters are required to be tuned, and the chosen optimiser was Random Search Cross-validation. Due to the limited time budget, only important parameters are tuned and these were including:

- **max_depth:** The maximum depth per tree which must be tuned since while increasing the number of depth might increase accuracy, the tradeoff lies at the model complexity.
- **learning_rate:** This determines the step-size of each iteration within the algorithm. While a low learning rate takes a longer time to achieve the same reduction in residual errors in comparison with a higher learning rate, a low learning has the ability to reach the optimum outcome.
- **n_estimators:** The total number of boosting rounds/trees for the training process.
- **subsample:** Fraction of training instances to be sampled prior to growing trees.
- **colsample_bytree:** Fraction of columns to be sampled prior to growing trees.
- **colsample_bylevel:** Fraction of levels to be sampled prior to growing trees.

The common objective of these hyperparameters is to prevent overfitting. The evaluation metric for the RandomizedSearchCV is the negative mean squared error. The idea behind using this metric rather than setting a normal "mean-squared error" is due to the expectation that the cross-validation will optimise this metric towards the zero direction and thus, in fact, minimizing positive mean-squared error.

After fitting the training dataset using cross-validation, the best set of parameters is recorded in Appendix 19. Additionally, the prediction outcomes from this trained model is recorded in the results section.

### 3.3.3. Light Gradient Boosting machine (LightGBM)

LightGBM is a gradient boosting framework that uses tree-based learning algorithm (Mandot, 2017). The main differentiation of this algorithm is that it grows trees vertically, rather than horizontally like in other boosting methods. Referring to Appendix 15, vertical growth can also be illustrated as leaf-wise growth, where the algorithm explicitly chooses the leaf with the maximum decrease in loss and moves on to fitting another leaf. Since this process avoids checking all losses in the previous leaves for each new leaf it fits, the computation costs significantly improve. Besides, instead of searching for the best split point on all sorted feature values, LightGBM uses histogram implementation and groups sorted attributes into bins and then, search for the split point via the grouped bins. The described streamlined process clearly do not only provide accuracy, but at a much faster speed and lower memory storage, making it highly suitable for large datasets. While the advantages are clear, its disadvantages lie at its vertically growth direction, making it proned to

growing deep trees and over-fitting issues (Mandot, 2017). Therefore, certain hyperparameters must be efficiently optimised to minimise these issues.

According to Scharth (2022), the most important hyperparameters include: number of tress, learning rate and maximum tree depth. However, since LightGBM offers more hyperparameters to increase the efficiencies of the algorithm, this project will attempt to tune more than the basics. In addition, the tuning process will be performed via Optuna - an automatic hyperparameter tuning software to ensure performance maximisation. In particular, the parameters chosen to be tuned in our context includes:

- **num_leaves:** The number of leaves in a full tree.
- **lambda_l1:** L1 regularization that will penalise the sum of absolute value of leaf scores, assist in reducing the tree depth via removing less-predictive features.
- **lambda_l2:** L2 regularization that will impose strong penalisation on the large leaf scores, while being less harsh on features with low predictability since it does not have the ability to remove features.
- **bagging_fraction:** The fraction of data to be used in each iteration. If the right proportion is chosen, speed can be increased while overfitting can be decreased.
- **bagging_freq:** This hyperparameter controls how often a new sample of training data is drawn. When this hyperparameter is combined with bagging_fraction, both accuracy and speed improves since continuously changing data samples will improve outcome reliability.
- **feature_fraction:** This hyperparameter decides the proportion of features to be used upon the start of a tree construction. This is reasonable to tune since using all the features at every new construction will only lead to increasing the number of splits required, the number of evaluation at each node and the time budget.
- **min_data_in_leaf:** The minimum number of records a leaf may have, to prevent overfitting.

Besides, basic hyperparmeters are intuitively decided to avoid computational costs in tuning processes. These include objective as ‘regression’ and boosting_type as ‘gbdt’ for utilisation of traditional gradient boosting.

In addition, the reason behind using Optuna is due to its flexibility in searching the most appropriate value across a specified range. For example, if num_leaves can only be an integer and is intuitively judged to be within 2 and 64, Optuna’s attribute called suggest_int can search the range to conclude a suitable value via the application of k-fold cross-validation. Furthermore, the number of folds decide

to be used is 5 and the number of early_stopping_rounds equals 50. This specification represents that if after 50 consecutive boosting rounds and the training's accuracy keep being worse than its previous one, the training process will stop. This statement is made to avoid wasting time into further unpromising boosting rounds. Moreover, 5 folds were decided based on the primary concern upon time budget.

In the context of our dataset, given a time budget of 10 minutes, the best set of parameters derived can be referred to Appendix 16 and the metric derived from the training can be referred to in Appendix 17. The evaluation of metric clearly shows that for this dataset, LightGBM tends to overfit a bit fast in comparison to the nature of the algorithm. However, since LightGBM is overall highly robust and efficient method, it is strongly believed to be the most accurate and most efficient models. This hypothesis will be confirmed in the results section.

#### 3.3.4. CatBoost

CatBoost is made up from Categorical and Boosting. From its name, it can be immediately understand that this algorithm focuses on efficiently and reasonably addressing category variables within the GBDT framework. In comparison with other boosting algorithms, CatBoost is innovated in the fact it automatically process category-based features into numerical features. Therefore, no pre-processed encodings are required, making the statistical learning experience much more streamlined and efficient. However, since our dataset has all been pre-enconded in the feature engineering step, CatBoost's advantages become redundant in our context. However, the algorithm is still given a fair chance, therefore, the hyperparameters chosen to be tuned in CatBoost includes:

- **loss_function:** This underlines which metric to be chosen for the evaluation of this algorithm.
- **learning_rate:** The step-size between each iteration as mentioned.
- **l2_leaf_reg:** L2 regularization to penalise large loss score within leaf to prevent overfitting.
- **depth:** Depth of tree allowed since while a big tree will increase train model accuracy, it comes at the costs of overfitting in the prediction model.
- **iterations:** Number of trees or boosting rounds for a training process.

Similar to LightGBM, Optuna integration was also used to maximise the tuning performance of hyperparameters (Appendix 20). The train model is then used to predict on the testing dataset with the results to be recorded in the next section.

Overall, CatBoost can match any advanced machine learning algorithm in terms of performance, reducing the need for large hyperparametric tuning and reducing the chance of overfitting, which also makes the model more versatile. It can also handle categorical and numeric features, and supports custom loss functions. However, the processing of category features requires a lot of memory and time, and in addition to that, encoding always affect prediction accuracy to a certain extent since generally, encoded features can only 'closely' present the original values, and could not 'replace' them.

### 3.3.5. Model stacking

Ensemble Learning is not a single machine learning algorithm, but a combination of many machine learning algorithms (Pavlyshenko, 2018) and Stacking is a type of ensemble learning. This process involves combining the predictions of multiple different projects and then use another algorithm to find the most appropriate model for the dataset. The process of stacking involves constructing a base model and a meta model, which can also be referred to as level-0 models and level-1 model, respectively. The ordinal structure of the models clearly reveals the idea that the outputs from the base model will become the inputs for the metal-model. Once this training dataset if prepared, the meta-model will be able to trained separately from the base models, while the base models still utilise the original training dataset.

In the context of our dataset, stacking is highly appropriate since all previously specified gradient boosting models worked well on the dataset, thus requiring a further algorithm to decide the most optimal model to choose from. In contrast to the complexity of the base models, meta-model tends to have a simple algorithm such as linear regression in regression problems or logistics regression in classification problems, since the main purpose of meta model is only to provide a smooth interpretation of the predictions made by the base models. Therefore, in the application of this theory upon our dataset, the chosen base models are XGBoost, LightGBM and CatBoost. The algorithm for the meta-model will be linear regression and the evaluation metrics will still be RMSE, R-squared and MAE. Overall, Stacking is simple and easy to implement since it does not require complicated hyperparameters tuning or complicated feature selection processes.

## 3.4. Results

### 3.4.1. Model selection

| Models | R-squared | RMSE | MAE |
|---|---|---|---|
| **Model Stacking** *(XGBoost, LightGBM, CatBoost)* | 0.7106 | 0.4727 | 0.3437 |
| **LightGBM** *(5-fold CV & Optuna integration as hyperparameters optimiser)* | 0.7100 | 0.4732 | 0.3444 |
| **CatBoost** *(Sckitlearn API)* | 0.7084 | 0.4745 | 0.3462 |
| **XGBoost** *(RandomizedSearchCV as hyperparameters optimiser)* | 0.7068 | 0.4758 | 0.3470 |
| **Multiple Linear Regression** *(Backward selected)* | 0.6383 | 0.5417 | 0.4029 |

### 3.4.2. Discussion

As all prediction models are evaluated using the valuation metrics R-squared, RMSE and MAE, it can clearly be compared that the best performing model is the Stacking model, following by LightGBM, CatBoost, XGBoost and Multiple Linear Regression. This outcome is considered closely consistent with the methodological theories discussed above. In particular, it is consistent with the expectation that Model Stacking will give the best performance, since it is a 'blend' of all the best-performing individual model.

However, it was unexpected that XGBoost performed worse than CatBoost. Based in the discussion in the previous section, XGBoost was expected to be more appropriate with the featured dataset rather than CatBoost, since CatBoost is more suitable for the dataset with raw categorical features. The main reason behind this outcome is believed to be due to the lack of tunned hyperparameters within XGBoost. In specific, since RandomizedSearchCV was used to tune hyperparameters for XGBoost without the support of Optuna integration, the number of hyperparameters specified to be tuned was

limited to only 6 most important hyperparameters. Even with this manually-created limitation, it took approximately 22 minutes to finish the search. Meanwhile, since CatBoost was tuned with the support of Optuna integration, the tuning process had a specified time budget and the hyperparameters were searched more thoroughly via Optuna trials. Therefore, more hyperparameters were able to be passed via the tuning process, and with the further maximisation of quality, it is explainable if CatBoost outperforms XGBoost in this study. Besides, it was also expected from the beginning that MLR underperforms compared to the remaining non-linear models due to its rigid assumptions and its non-suitability with the dataset, as discussed.

Moreover, the major drawbacks in this study refer to the response variable being log-transformed, rather than being in its original form. Therefore, all the model predictions generated are also still in the log form, and if these are 'exponentiated' back into its normal form of price, the prediction accuracy might drastically change. However, the log-transformation was necessary for the supervised learning since *price* was overally skewed and the outcomes are believed to have gone as accurate as they get.

Ultimately, based on the statistical outcome, Model Stacking is strongly recommended for all hosts, real estate investors and other stakeholders as a predictive model for Airbnb listings price. By using this model to study the attributes of potential or existing Airbnb properties, stakeholders will be able to accurately estimate their listing prices by up to 71.06%. Eventually, if these models are properly used, hosts can significantly improve their occupancy rate via appropriate pricings and real estate investors can make more accurate decisions on where to invest their next Airbnb business.

## 4. DATA MINING

**Insight 1: Best hosts focus on providing an entire accommodation as a property type.**

Referring to Appendix 4, the pie chart reveals that the largest number of property types being rented out belongs to the category of "Entire home/Apartment". In percentage terms, this is equivalent to 64.2% of Airbnb listings market share. This observation clearly identifies that in terms of property type, the major proportion of market demand lies in private accommodation, with more preferences in an 'entire' place rather than an 'entire' room in a shared place. Therefore, if stakeholders can further expanding on these preferences, they might be able to leverage further profits due to concentrated demand. However, another contrasted idea can also be derived from the pie chart, in particular, the lack of shared rooms being offered as a property type (1.61% of market share) might identify a potential market gap. Therefore, new hosts and potential investors who would like to target the

'budget' travellers can focus on 'filling' in this gap with new listings focusing on shared accommodations.

### Insight 2: Best hosts provide quality, branded products.

Referring to Appendix 12, hosts with perfect rating scores focus on providing quality in-house products such as: L'Occitane body soap and shampoo, Koi body soap, Alpha Keri body soap, Lab6 Professional shampoo, etc. In addition, high scores rating also appear to be associated with the quality entertainment systems that hosts provide. For example, top-rated accommodations have SONOS Speaker, Raurk & Cambridge Audio sound system and other bluetooth sound systems. This observation is understandable since Airbnb's guests are mostly travellers on vacation, therefore, their demand for entertainment products is high. Overall, the observations clearly deduce that high score ratings are associated with high quality products. Host who would like to "stand-out" requires attention to details and investment in quality products. While the initial spending is high, the long-term tradeoffs is outweighed.

### Insight 3: Most expensive listings on average provide extra utilities.

Referring to Appendix 11, accommodations that got listed with high prices provide a range of practical and additional utilities. At the top of the list, 'heat lamps' are associated with high listing price. This is a highly useful insight since investing in a heat lamp is relatively cheap, while the benefits translated via price increase is substantially large. Similarly, other affordable products but are associated with increase in profit margins include: exercise equipment, air-hockey table, natural gas BBQ, smart TV, Espresso machine, outdoor seatings, etc. Therefore, all hosts can easily improve their revenue via increasing listing prices by providing suggested utilities if they have not done so.

### Insight 4: North Sydney and the Eastern suburbs have the highest average listing price.

Referring to Appendix 6, the choropleth map clearly highlights North Sydney as having the highest average listing price, with up to 800 AUD per overnight accommodation. The bar plot further shows that the two most expensive suburbs in this area are Pittwater and Mosman, following by Manly. In addition, the second most expensive listings area is the Eastern suburbs, particularly Woollahra and Waverly. A common characteristics that these expensive suburbs share is their near-waterfronts locations such as the near the Manly Beach or Bondi Beach, further highlighting strong demand for water-activities. Therefore, in terms of geographical location, potential real estate investors and new host who prefer high profit margin can focus investing new Airbnb listings in these two areas since by considering the location alone, these suburbs already have higher-than-average listing prices

compared to the rest of Sydney. If hosts can further combine these great locations with good services, their listings will be highly attractive in the market.

### Insight 5: Sydney CBD has the highest occupancy rate.

Referring to Appendix 5, the bar plot clearly highlights that the most-wanted suburb is the Sydney city. This is intuitively reasonable since the most activities and tourist locations such as the Opera House, the QVB, Chinatown, etc. focus primarily in this area. Therefore, if potential investors would like to increase revenue via increasing occupancy rate rather than increasing profit margins per listing, they can certainly focus on investing in this particular area.

### Insight 6: Strong growth potentials post-pandemic

Referring to Appendix 3, the **number of new listings per year** can be spotted to increase annually only until 2015. After the peak, it gradually declines. Particularly, in the last 2 years, the number of new listings significantly decrease which can be reasonably explained by the COVID-19 pandemic, where closed borders lead to the shut down of many businesses, world-wide. From an investor and host perspective, this trend prompts uplifting future for the Airbnb rental markets in the upcoming years since travellers will urge to travel when the borders open. Furthermore, with the precautionary savings that they accumulated from the pandemic periods, guests will likely to spend more and thus, this will be highly profitable from the host perspectives since not only occupancy rates will increase, but also potential prices of rentals due to high demand. Therefore, hosts and real estate investors should treat the decreasing trend in new listings as only a temporary, cyclical downturn with strong outlook for rebounce, and should not make early conclusion regarding market saturation issues.

### Insight 7: Low market concentrations urge competitions in differentiations.

As a minor trend, by grouping listings by year (Appendix 2), it is observable that the **total number of Airbnb listings** increase drastically on an annual basis. From the host perspectives, this trend might deduce increase in competitions and perhaps, decrease in occupancy rates in the long-term since guests will have more choices over time. This trend further underlines that price competition will not be sustainable in the medium-long term and that competition differentiation must be prioritised. Hosts are required to be more serious and efficient with managing their properties.

## 5. CONCLUSION

Therefore, as both Machine Learning and Data Mining have been actively applied onto the Airbnb listings dataset, it has been clearly evidenced that the best predictive model for listing prices is the Model Stacking, with predictive accuracy of up to 71.06%. In addition, *seven* meaningful insights were derived from the study, which can be summaried as:

1) *Best hosts focus on providing an entire accommodation as a property type.*
2) *Best hosts provide quality, branded products.*
3) *Most expensive listings on average provide extra utilities.*
4) *North Sydney and the Eastern suburbs have the highest average listing price.*
5) *Sydney CBD has the highest occupancy rate.*
6) *Strong growth potentials post-pandemic*
7) *Low market concentrations urge competitions in differentiations.*

Ultimately, if hosts, real estate investors and other stakeholders can closely utilise and follow all the recommendations provided, the performance of their Airbnb listings can guaranteed to be improved which thus, will also translates to significant improvement in profit maximisation and occupancy rates.

## 6. REFERENCES


Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).

Chowdhury, & Turin. (2020). Variable selection strategies and its importance in clinical prediction modelling. *BMJ*. https://doi.org/10.1136/ fmch-2019-000262

Dorogush, A. V., Ershov, V., & Gulin, A. (2018). CatBoost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*.

Guttentag, D. (2019). Progress on Airbnb: a literature review. *Journal of Hospitality and TourismTechnology*.

Jackson. (2002). *Data mining: a conceptual overview*. Communications of the Association for Information Systems. https://www.academia.edu/5442707/DATA_MINING_A_CONCEPTUAL_OVERVIEW?from=cover_page

Jordan, & Mitchell. (2015). *Machine learning: Trends, perspectives, and prospects*. https://www.science.org/doi/pdf/10.1126/science.aaa8415

Mandot, P. (2017). What is LightGBM, how to implement it? How to fine tune the parameters? *Medium*. https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc

Ming-Syan Chen, Jiawei Han and P. S. Yu, "Data mining: an overview from a database perspective," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 8, no. 6, pp. 866-883, Dec. 1996, doi: 10.1109/69.553155.

Pavlyshenko, B. (2018, August). Using stacking approaches for machine learning models. In *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)* (pp. 255-258). IEEE.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: unbiased boosting with categorical features. *Advances in neural information processing systems*, *31*.

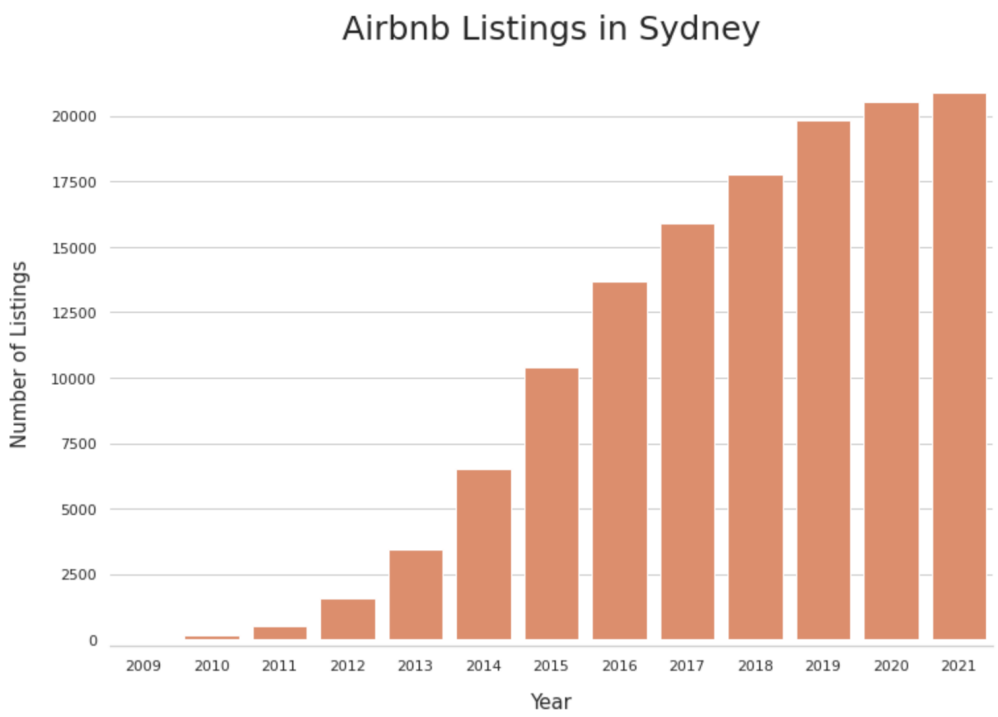
Scharth. (2022). *Lecture 9: Gradient Boosting* [PowerPoint Slides]. The University of Sydney. https://canvas.sydney.edu.au/courses/41494/pages/lecture-gradient-boosting?module_item_id=1506435

# 7. APPENDICES

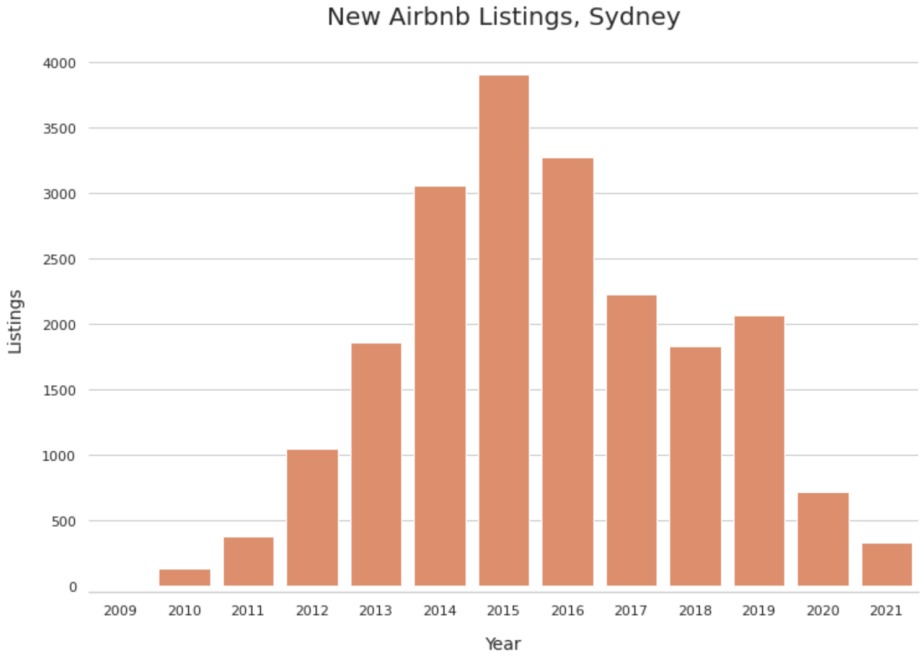
## Appendix 1: Dictionary of the original dataset

```
0   id                                            20880 non-null  int64
1   listing_url                                   20880 non-null  object
2   scrape_id                                     20880 non-null  int64
3   last_scraped                                  20880 non-null  object
4   name                                          20873 non-null  object
5   description                                   20126 non-null  object
6   neighborhood_overview                         12589 non-null  object
7   picture_url                                   20880 non-null  object
8   host_id                                       20880 non-null  int64
9   host_url                                      20880 non-null  object
10  host_name                                     20877 non-null  object
11  host_since                                    20877 non-null  object
12  host_location                                 20859 non-null  object
13  host_about                                    11224 non-null  object
14  host_response_time                            7884 non-null   object
15  host_response_rate                            7884 non-null   object
16  host_acceptance_rate                          8846 non-null   object
17  host_is_superhost                             20877 non-null  object
18  host_thumbnail_url                            20877 non-null  object
19  host_picture_url                              20877 non-null  object
20  host_neighbourhood                            11852 non-null  object
21  host_listings_count                           20877 non-null  float64
22  host_total_listings_count                     20877 non-null  float64
23  host_verifications                            20880 non-null  object
24  host_has_profile_pic                          20877 non-null  object
25  host_identity_verified                        20877 non-null  object
26  neighbourhood                                 12591 non-null  object
27  neighbourhood_cleansed                        20880 non-null  object
28  neighbourhood_group_cleansed                  0 non-null      float64
29  latitude                                      20880 non-null  float64
30  longitude                                     20880 non-null  float64
31  property_type                                 20880 non-null  object
32  room_type                                     20880 non-null  object
33  accommodates                                  20880 non-null  int64
34  bathrooms                                     0 non-null      float64
35  bathrooms_text                                20856 non-null  object
36  bedrooms                                      19444 non-null  float64
37  beds                                          19976 non-null  float64
38  amenities                                     20880 non-null  object
39  price                                         20880 non-null  object
40  minimum_nights                                20880 non-null  int64
41  maximum_nights                                20880 non-null  int64
42  minimum_minimum_nights                        20879 non-null  float64
43  maximum_minimum_nights                        20879 non-null  float64
44  minimum_maximum_nights                        20879 non-null  float64
45  maximum_maximum_nights                        20879 non-null  float64
46  minimum_nights_avg_ntm                        20879 non-null  float64
47  maximum_nights_avg_ntm                        20879 non-null  float64
48  calendar_updated                              0 non-null      float64
49  has_availability                              20880 non-null  object
50  availability_30                               20880 non-null  int64
51  availability_60                               20880 non-null  int64
52  availability_90                               20880 non-null  int64
53  availability_365                              20880 non-null  int64
54  calendar_last_scraped                         20880 non-null  object
55  number_of_reviews                             20880 non-null  int64
56  number_of_reviews_ltm                         20880 non-null  int64
57  number_of_reviews_l30d                        20880 non-null  int64
58  first_review                                  15071 non-null  object
59  last_review                                   15071 non-null  object
60  review_scores_rating                          15071 non-null  float64
61  review_scores_accuracy                        14459 non-null  float64
62  review_scores_cleanliness                     14469 non-null  float64
63  review_scores_checkin                         14452 non-null  float64
64  review_scores_communication                   14469 non-null  float64
65  review_scores_location                        14453 non-null  float64
66  review_scores_value                           14448 non-null  float64
67  license                                       7493 non-null   object
68  instant_bookable                              20880 non-null  object
69  calculated_host_listings_count                20880 non-null  int64
70  calculated_host_listings_count_entire_homes   20880 non-null  int64
71  calculated_host_listings_count_private_rooms  20880 non-null  int64
72  calculated_host_listings_count_shared_rooms   20880 non-null  int64
73  reviews_per_month                             15071 non-null  float64
dtypes: float64(23), int64(17), object(34)
memory usage: 11.8+ MB
```

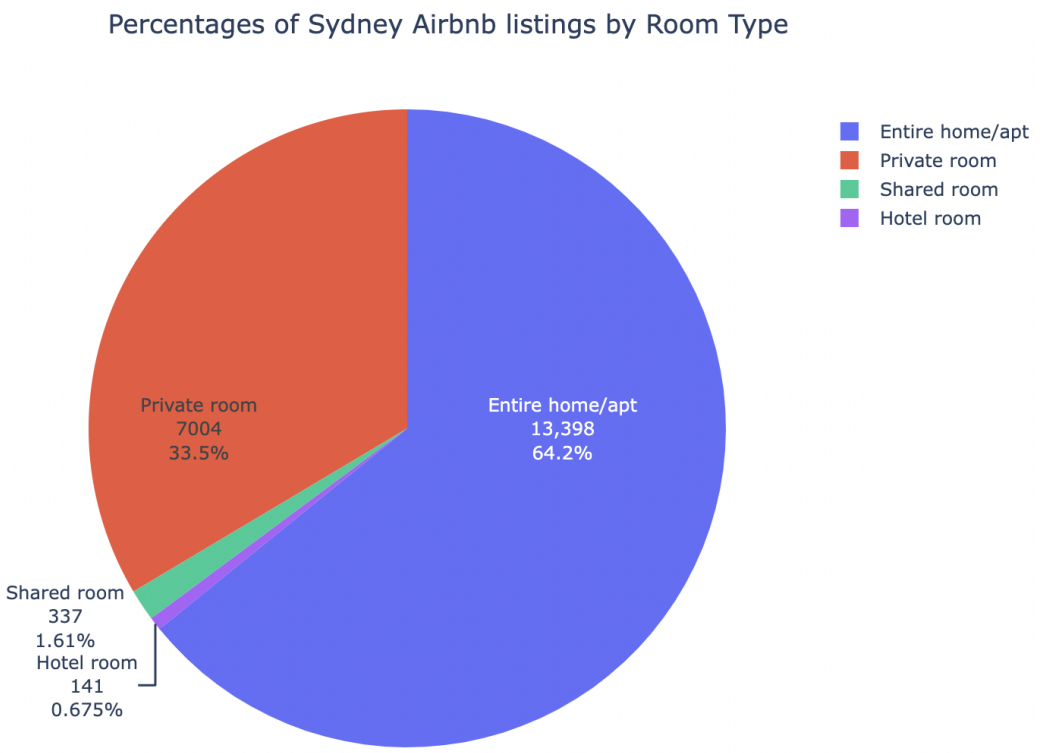
**Appendix 2: Total Airbnb Listings in Sydney**



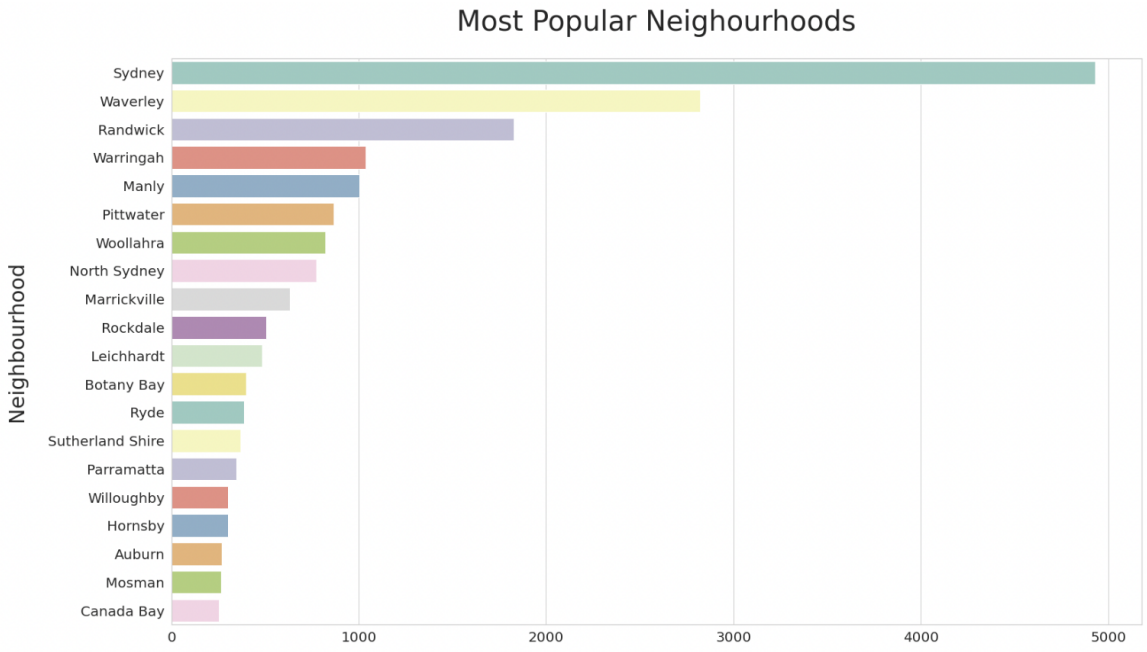

**Appendix 3: New Airbnb listings in Sydney per Year**



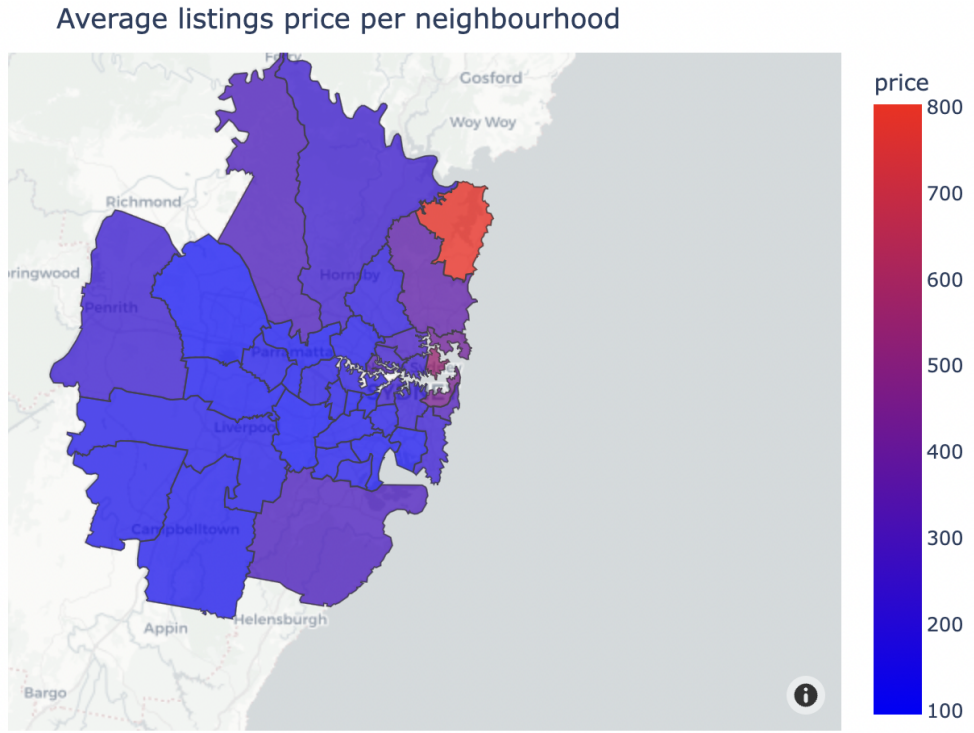
**Appendix 4: Types of Airbnb rental rooms in Sydney**



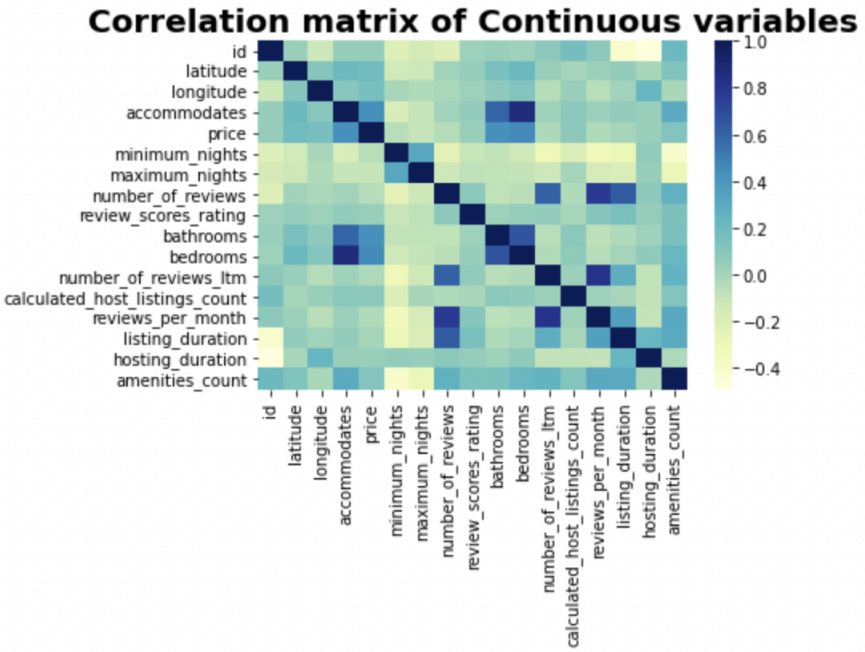

**Appendix 5: Most popular neighbourhood for Airbnb listings**



**Appendix 6: Average Airbnb listing prices per Neighbourhood in Sydney**

## Appendix 7: Pairwise correlation matrix of variables

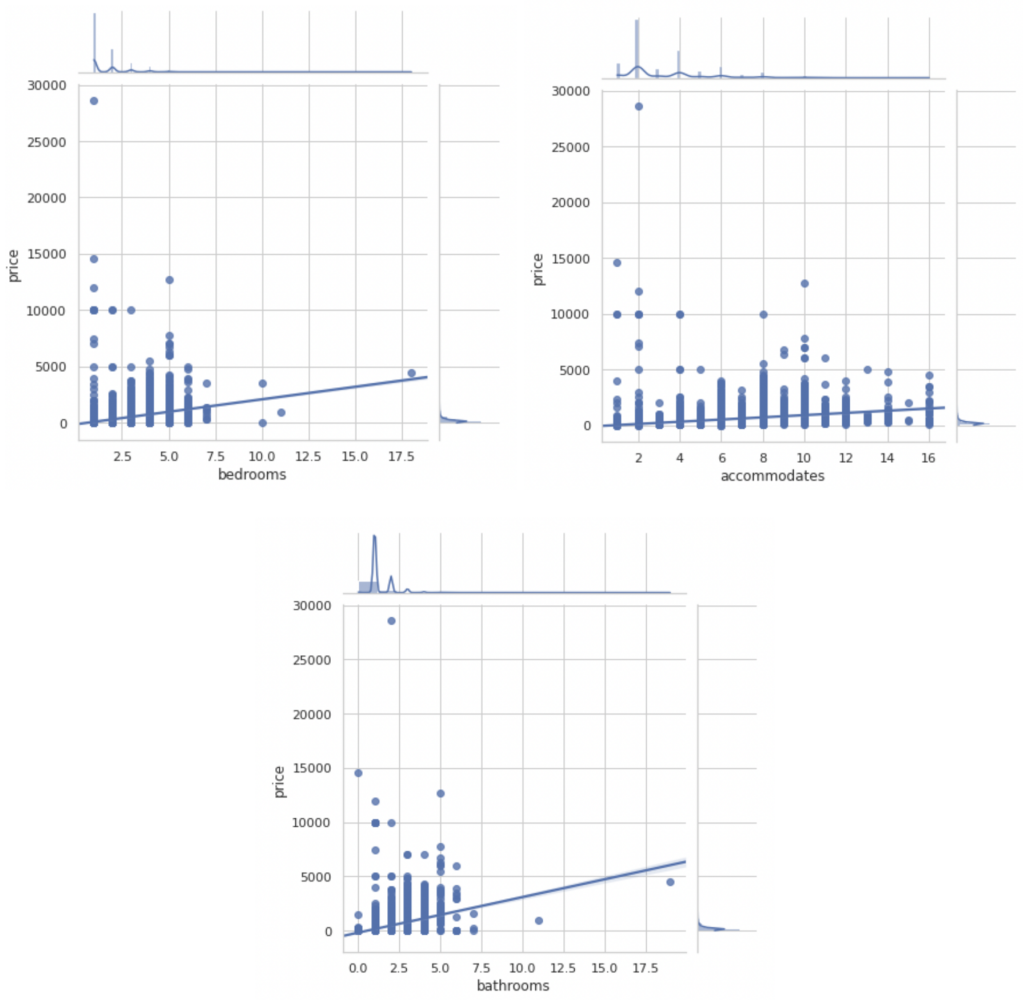



## Appendix 8: Summary statistics of the datatset

| | id | latitude | longitude | accommodates | price | minimum_nights | maximum_nights | number_of_reviews | review_scores_rating | bathrooms | bedrooms |
|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 20880.00 | 20880.00 | 20880.00 | 20880.00 | 20880.00 | 20880.00 | 20880.00 | 20880.00 | 20880.00 | 20880.00 | 20880.00 |
| mean | 27202395.07 | -33.86 | 151.20 | 3.38 | 251.17 | 62.19 | 905.05 | 17.90 | 4.57 | 1.32 | 1.67 |
| std | 14745864.03 | 0.08 | 0.09 | 2.19 | 498.60 | 52.74 | 407.08 | 42.99 | 0.90 | 0.66 | 1.02 |
| min | 11156.00 | -34.10 | 150.63 | 1.00 | 13.00 | 1.00 | 1.00 | 0.00 | 0.00 | 0.00 | 1.00 |
| 25% | 14905121.25 | -33.90 | 151.17 | 2.00 | 80.00 | 4.00 | 1125.00 | 0.00 | 4.67 | 1.00 | 1.00 |
| 50% | 26883106.50 | -33.88 | 151.21 | 2.00 | 140.00 | 90.00 | 1125.00 | 2.00 | 4.81 | 1.00 | 1.00 |
| 75% | 40238118.75 | -33.82 | 151.26 | 4.00 | 250.00 | 90.00 | 1125.00 | 13.00 | 4.94 | 1.00 | 2.00 |
| max | 53709998.00 | -33.39 | 151.34 | 16.00 | 28613.00 | 1125.00 | 1500.00 | 881.00 | 5.00 | 19.00 | 18.00 |

| number_of_reviews_ltm | calculated_host_listings_count | reviews_per_month | listing_duration | hosting_duration | amenities_count |
|---|---|---|---|---|---|
| 20880.00 | 20880.00 | 20880.00 | 20880.00 | 20880.00 | 20880.00 |
| 2.88 | 8.09 | 0.52 | 479.93 | 2082.44 | 22.99 |
| 9.41 | 25.19 | 0.95 | 584.94 | 842.41 | 12.41 |
| 0.00 | 1.00 | 0.01 | 0.00 | 9.00 | 1.00 |
| 0.00 | 1.00 | 0.09 | 70.00 | 1470.00 | 13.00 |
| 0.00 | 1.00 | 0.20 | 289.00 | 2166.00 | 20.00 |
| 1.00 | 3.00 | 0.47 | 616.00 | 2674.00 | 31.00 |
| 565.00 | 197.00 | 44.49 | 3989.00 | 4558.00 | 85.00 |

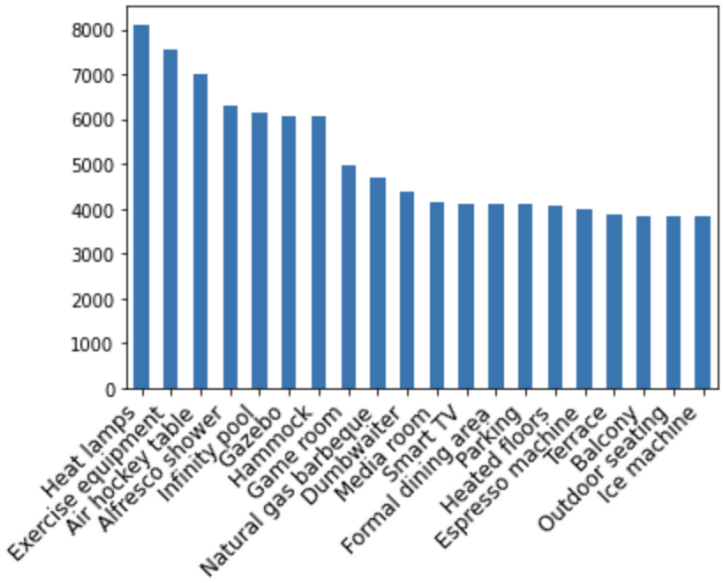
## Appendix 9: Linear plots of the 3 most correlated variables with *price*



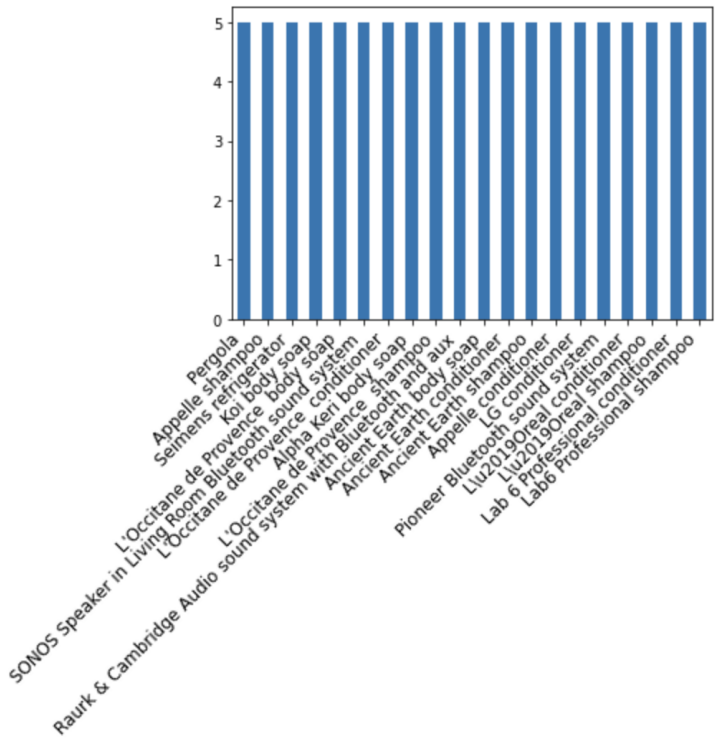

## Appendix 10: Coefficient of Variation

```
[51] # Coefficient of Variation (CV)

     print('price_CV:', np.std(listings['price'])/np.mean(listings['price']))
     print('bedrooms_CV:', np.std(listings['bedrooms'])/np.mean(listings['bedrooms']))
     print('bathrooms:', np.std(listings['bathrooms'])/np.mean(listings['bathrooms']))
     print('accommodates_cv:', np.std(listings['accommodates'])/np.mean(listings['accommodates']))
     print('review_scores_rating_CV:', np.std(listings['review_scores_rating'])/np.mean(listings['review_scores_rating']))

     price_CV: 1.9850879447942191
     bedrooms_CV: 0.6085173032692284
     bathrooms: 0.5007744542756484
     accommodates_cv: 0.6467191011803753
     review_scores_rating_CV: 0.1982125478254406
```

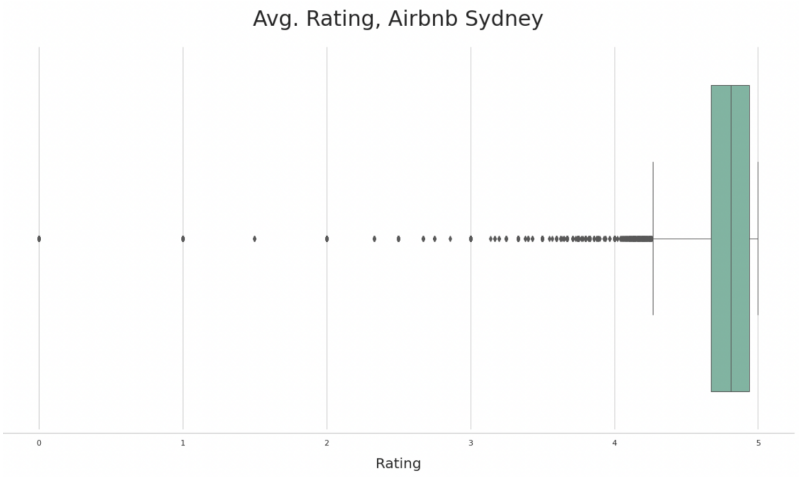
**Appnendix 11: Amenities vs. Listing prices (Top 20)**



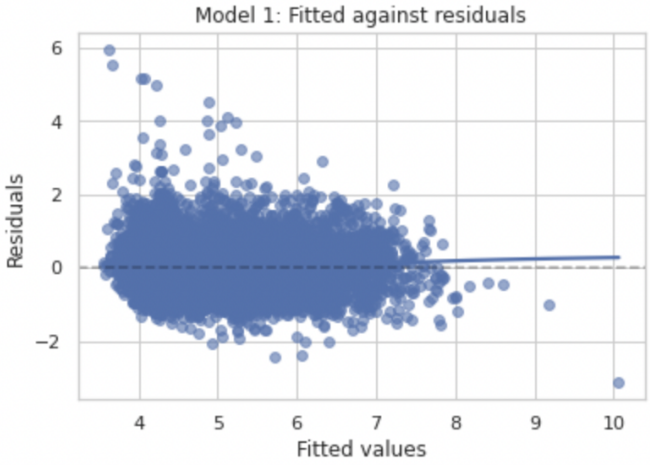

**Appendix 12: Amenities vs. Review Scores rating (Top 20)**



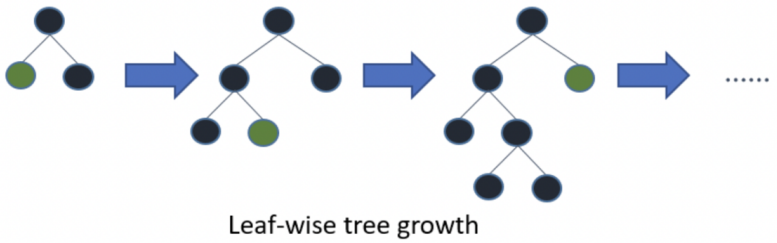
**Appendix 13: Average rating per Airbnb**




**Appendix 14: Residuals plot of MLR**



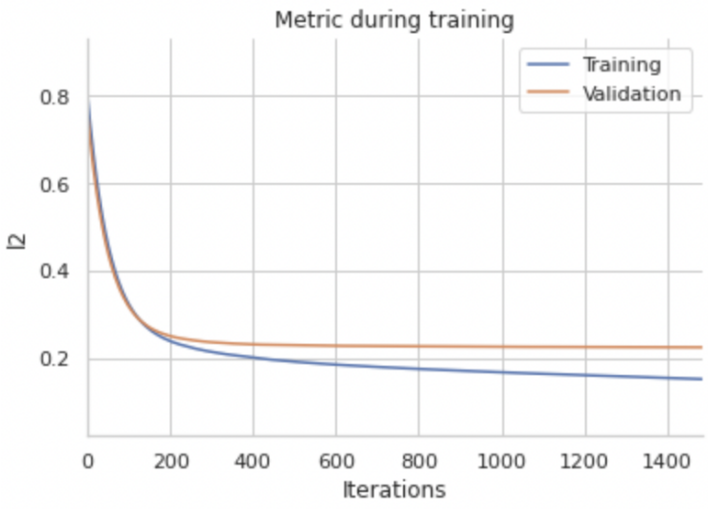

**Appendix 15: LightGBM algorithm illustration (Mandot, 2017)**



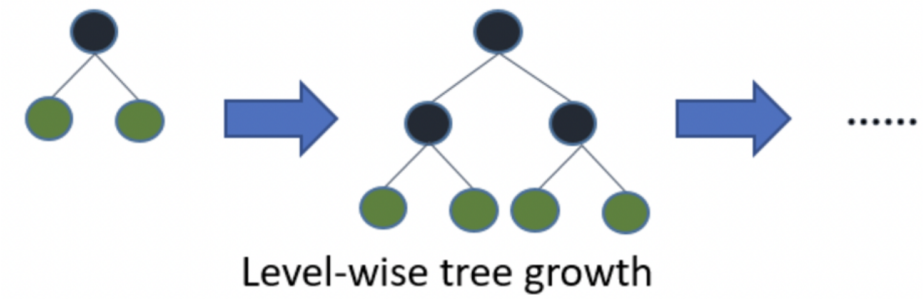
**Appendix 16: Best parameters after the Optuna study of LightGBM**

```
Number of boosting iterations: 1485

Best parameters:
{'bagging_fraction': 0.874297848963571,
 'bagging_freq': 2,
 'boosting_type': 'gbdt',
 'feature_fraction': 0.8175558194812812,
 'feature_pre_filter': False,
 'lambda_l1': 0.007147235613982666,
 'lambda_l2': 0.013899500025362737,
 'learning_rate': 0.01,
 'min_data_in_leaf': 2,
 'num_leaves': 39,
 'objective': 'regression',
 'verbosity': -1}
```

**Appendix 17: Metric during training the dataset using LightGBM**




**Appendix 18: XGBoost algorithm illustration (Mandot, 2017)**



**Appendix 19: Optimised hyperparameters for XGBoost**

```
Best parameters:
{'colsample_bylevel': 0.4,
 'colsample_bytree': 0.5,
 'learning_rate': 0.1,
 'max_depth': 5,
 'n_estimators': 500,
 'subsample': 0.8999999999999999}
```

**Appendix 20: Optimised hyperparameters for CatBoost**

```
Best Params:
    loss_function: RMSE
    learning_rate: 0.05164086188436666
    l2_leaf_reg: 0.026122840582070747
    depth: 10
    iterations: 671
```