## Feature engineering

1. For features with observations concentrated at specific categories, categories with fewer observations were combined into new categories to reduce sparsity and improve the representativeness of the sample.
    a. Host Response Time: The categories 'within a few hours', 'within a day', and 'a few days or more' were found to have significantly similar statistical characteristics and was subsequently combined into a new category: 'after an hour'.
    b. Property type: Except for 'apartment' and 'house, the different property types were combined based on their respective statistical characteristics into five categories: ‘under 150’, ‘under 200’, ‘under 250’ and ‘under 350’. The variable types ‘boat’ and ‘castle’ were also not combined because they did not display similar statistical characteristics with any other property type.
    c. Room type: The categories ‘private room’ and ‘shared room’ were combined due to their similar statistical characteristics.
    d. Cancellation policy: The categories ‘luxury_moderate’ and ‘luxury_super_strict’ were combined as they had similar characteristics and were sparse.
    e. Bathrooms: The existing categories for bathrooms were merged into ‘1-2’, ‘2-3’, ‘4-5’, and ‘>5’. The categories 0, 0.5, 3, and 3.5 were not merged due to them possessing significantly different statistical characteristics from other categories.
    f. Beds: Since the values are concentrated from 0 to 5, observations greater than five were combined into ‘6-7’, ‘8’, ‘9-10’, and ‘>10’.
    g. Guest_included: values greater than 4 were combined into ‘5-6’, and ‘>6’ to reduce sparsity.Guest_included: values greater than four were combined into ‘5-6’ and ‘>6’ to reduce sparsity.
    h. Calculated_host_listings_count_entire_homes & calculated_host_listings_count_private_rooms: All values greater than 1 were combined into ‘>1’ to reduce sparsity.

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

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

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

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