CO3093 Big Data and Predictive Analysis Coursework Assignment

Student No. 229062975

Objective: Given the dataset London_Listings.csv, I was tasked with building a predictive model that can predict the prices per night of Airbnb properties in London. This ties directly with previous lectures capturing supervised and unsupervised modelling within machine learning, what it means to manipulate, transform and explore data.

```
(66679, 31)
```

```
#   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
0   id                              66679 non-null  float64
1   name                            66679 non-null  object
2   description                     64723 non-null  object
3   host_id                         66679 non-null  int64
4   host_name                       66678 non-null  object
5   host_since                      66678 non-null  object
6   host_response_rate              56504 non-null  object
7   host_acceptance_rate            59533 non-null  object
8   host_is_superhost               66299 non-null  object
9   host_listings_count             66678 non-null  float64
10  host_total_listings_count       66678 non-null  float64
11  neighbourhood                   66679 non-null  object
12  latitude                        66679 non-null  float64
13  longitude                       66679 non-null  float64
14  property_type                   66679 non-null  object
15  room_type                       66679 non-null  object
16  accommodates                    66679 non-null  int64
17  bathrooms                       61350 non-null  float64
18  bathrooms_text                  66579 non-null  object
19  bedrooms                        66578 non-null  float64
20  beds                            61279 non-null  float64
21  amenities                       66679 non-null  object
22  price                           61418 non-null  object
27  first_review                    51943 non-null  object
28  last_review                     51943 non-null  object
29  review_scores_rating            51943 non-null  float64
30  calculated_host_listings_count  66679 non-null  int64
```

Part 1: Building Up a Basic Predictive Model

Exploratory Data Analysis (first look)

Using df.info(), df.describe and df.shape(); I was able to attain a general structure overview of the dataset. I discovered that this large dataset contained 66679 entries (rows) for 31 columns, meaning 30 potential predictors/features. Examining missing values using df.info() most variables were revealed to have a high count of non-null values indicating minimal missing data. This allowed me to be optimistic as fewer missing values reduces the risk of introducing bias into the model/analysis. This is because it reduces the chance of inadvertently skewing results when trying to impute (or just generally compensate) for missing data points. Specific columns I was able to identify with a high number of missing values (compared to other features) were: first_review, last_review, review_scores_rating, host_acceptance_rate and host_response_rate. I was able to use this insight later whilst considering the most impactful features to include in my overall model.

Other interesting insights from EDA:

- Mean number of listings per host being 28.56 suggesting that many hosts manage multiple properties
- Mean number of bedrooms = 1.54, suggesting many listings for small accommodations
- Maximum nights max=1124(nights) indicating listing available for long-term occupancy
- 'Accommodates' ranges from 1 to 16, showing the variety in property sizes.

Cleaning the data (highlights additional unique implementations)

1. I began by splitting the dataset into numerical and categorical variables using select_dtypes. This is important as different predictive models handle these types differently. For example, linear regression requires categorical variables to be converted into dummy variables.
2. Next, I cleaned the price column by removing non-numerical symbols.
3. Whilst cleaning the price column, I realised that other numerical columns might also contain non-numerical characters, possibly even due to human error. I therefore re-examined the categorical features to identify any that might have been misclassified as non-numerical due to contain non-numerical symbol (like $ for price) or others.
4. The review revealed that rating features included percentages, I treated this using similar methods applied to the price column.
5. I then created an updated list of numerical and categorical variables using select_dtypes again. This step helped me verify that the cleaning process was effective.
6. I replaced empty strings or lists in categorical features with NaN. I verified this was successful by comparing the number of NaN values before and after the process. Initially, there were 31,910 NaN values, which increased to 32,046 after replacing empty strings with NaN.

```
Number of NaN values: 31910
```

```
Number of NaN values: 32046
```

7. I combined the cleaned categorical variables with the rest of the dataset to ensure all changes were integrated.
8. Next, I examined the dataset holistically to assess missing values.
9. Upon evaluation, I discovered bathrooms and bathrooms_text convey the same information about the number of bathrooms, but bathrooms_text provides additional details. To avoid multicollinearity and reduce missing values, I created a function extract_bathrooms_num to extract numerical values

```python
def extract_bathrooms_num(text):
    if isinstance(text, str):
        match = re.search(r'(\d+\.?\d*)', text)  # finding numbers including decimals
        if match:
            return float(match.group(1))
    return np.nan  # return NaN if no number is found
```

from bathrooms_text and replace bathrooms with this information. I chose to keep bathrooms_text because it contained drastically less missing values.

Initially, bathrooms had 5,329 missing values, which decreased to 659 after this process.

10. I removed latitude and longitude because they are redundant with the neighbourhood feature, which already provides geographic information relevant to pricing and is more easily interpretable.

11. I evaluated calendar_last_scraped decipher whether the dates from last scraped data were reasonably close together and recent/reliable. I found it contained dates mostly from June 2024, further highlighting the London's listings dataset as recent and reliable data. However, this column was deemed unrelated to pricing and was removed.

```
Earliest date: 2024-06-14 00:00:00
Latest date: 2024-06-20 00:00:00
Number of unique dates: 4
[Timestamp('2024-06-14 00:00:00'), Timestamp('2024-06-15 00:00:00'), Timestamp('2024-06-16 00:00:00'), Timestamp(
```

12. Next, I was able to clarify the difference between host_listings_count, hosts_total_listings_count, and calculated_host_listings_count. I understood host_listings_count to give current host listings count, hosts_total_listings_count to show overall total listings of a particular host and calculate_llistings_count to give total listings per host within the particular data set.

13. I verified my understanding of this by finding the number of data points where hosts_total_listings_count< host_listings_count. This returned 0 as expected, verifying my understanding of the difference between the variables.

14. I removed all duplicate rows and rows with missing values to ensure data quality. The dataset's shape was regularly checked using df.shape() to monitor data loss during cleaning. `print(df.shape)`
15. I transformed the price column using logarithms to stabilise the variance and make the distribution more normal.

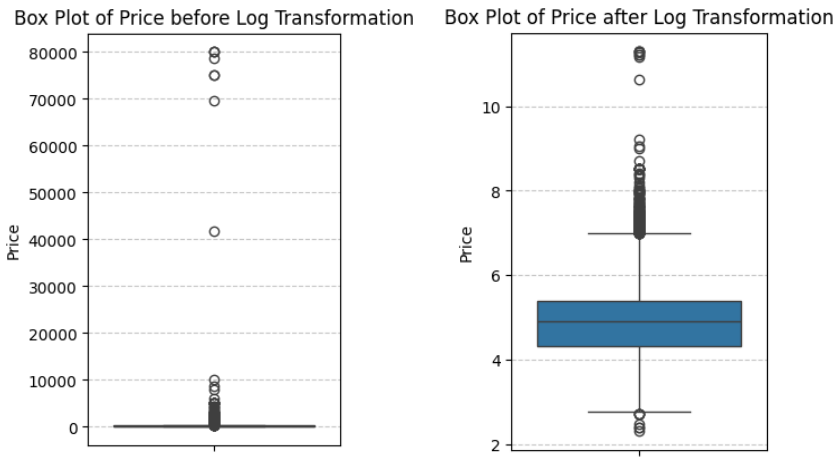
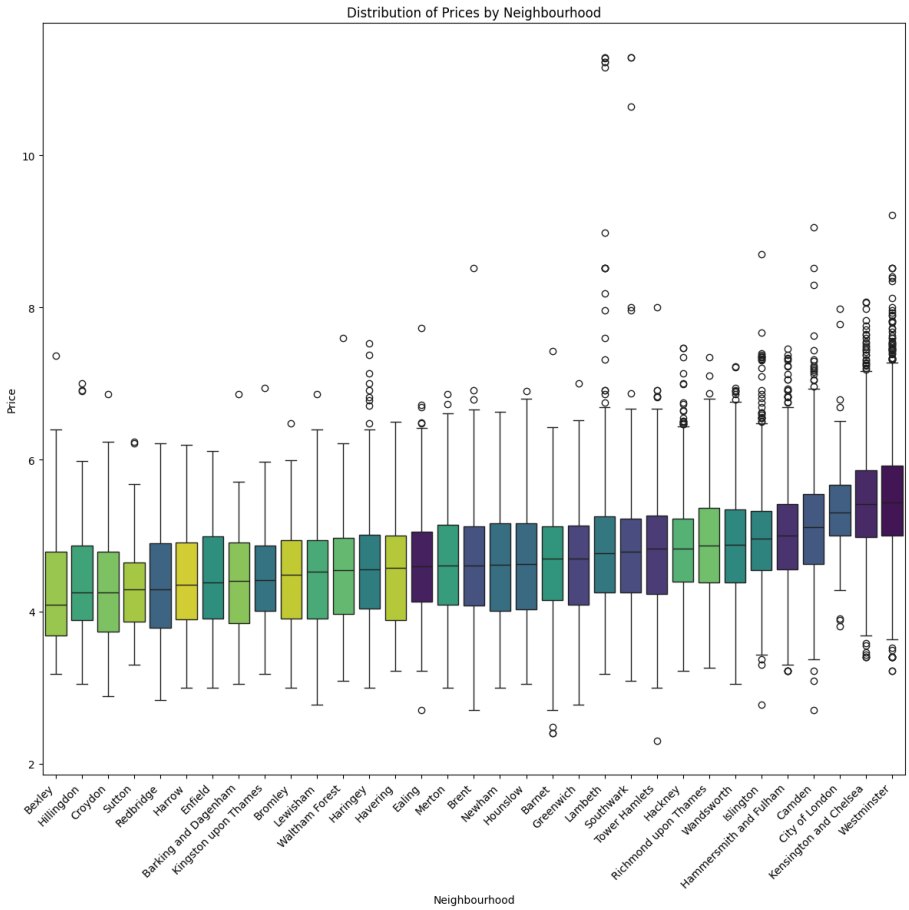
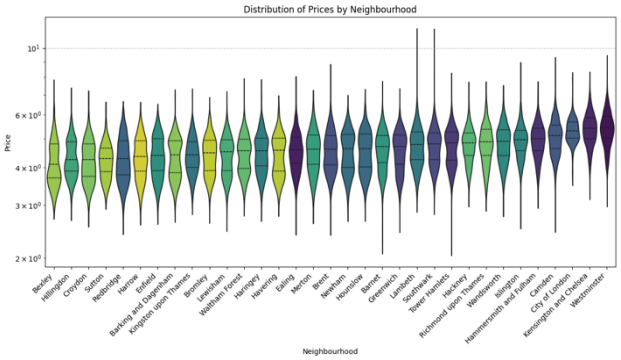


Figure 1. Log Transformation Effect 1

16. I normalized all numerical variables (excluding prices) to ensure they were on the same scale, which is crucial for many machine learning algorithms. Normalisation helps prevent features with large ranges from dominating the model.
17. Figure 1 allows you to visualise the impact of these transformations and ensure that outliers were effectively managed.
18. Some extras: unincluded id and host from outlier removal using z score calculations and removed decimal bathrooms e.g. 1.5 bathrooms etc.





Distribution of Prices by Neighbourhood

o gain insights into how prices vary across different neighbourhoods in London, I employed two visualisation techniques: box plots and violin plots.

The box plots can highlight variability in prices across neighbourhoods whilst capturing key statistical measures such as median, interquartile range (IQR), and outliers.

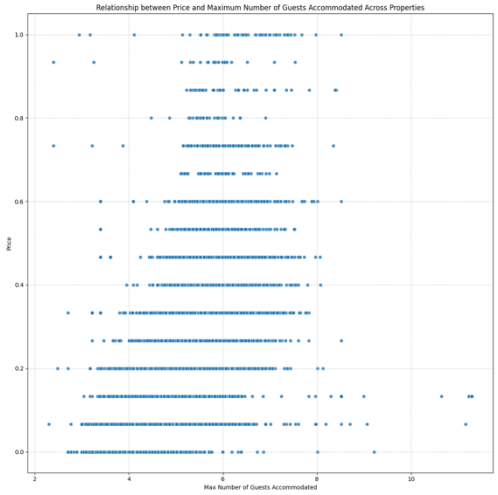
Observations:

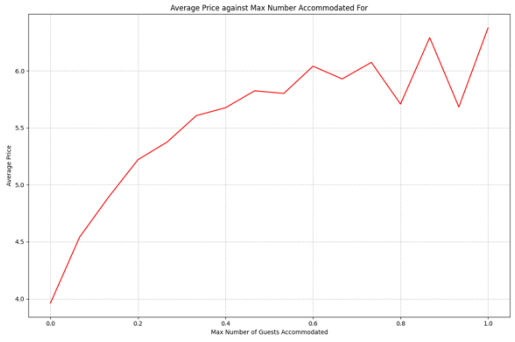
- High-priced neighbourhoods like Kensington and Chelsea, Westminster, and Camden exhibit higher median prices and greater variability.
- Lower-priced neighbourhoods, such as Bexley and Hillingdon, have narrower IQRs, indicating more consistent pricing.
- Outliers are prevalent in high-end areas, suggesting the presence of premium or luxury listings.
- Median prices vary significantly across neighbourhoods, with Kensington and Chelsea showing the highest median price at approximately at 5 on the scale (converted due to log transformation).
- The spread in prices suggests that neighbourhood is a strong predictor of Airbnb pricing.

Violin plots provide additional insights into the density of price distributions within each neighbourhood.

- High-priced neighbourhoods display a density skewed toward higher values, confirming that these areas attract premium listings.
- Lower-priced neighbourhoods exhibit more uniform distributions, indicating stable pricing strategies.

Price Across Number of Possible Tenants





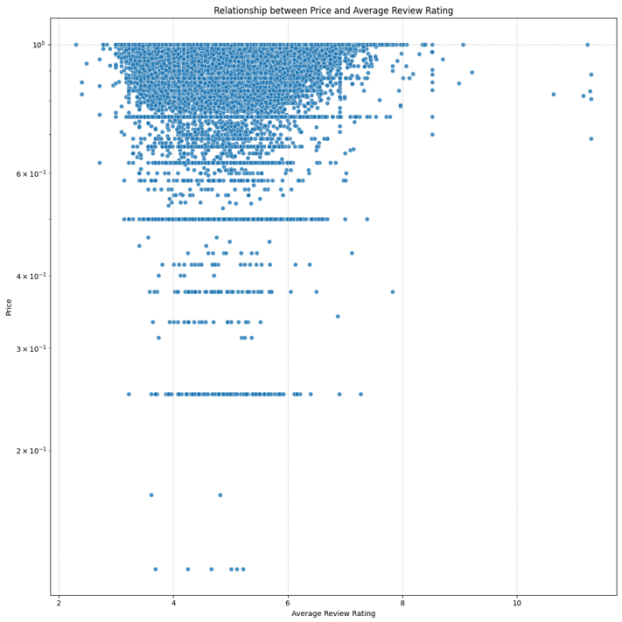
To analyse the relationship between the maximum number of guests accommodated and price, I used two visualization techniques: scatter plots and line plots.

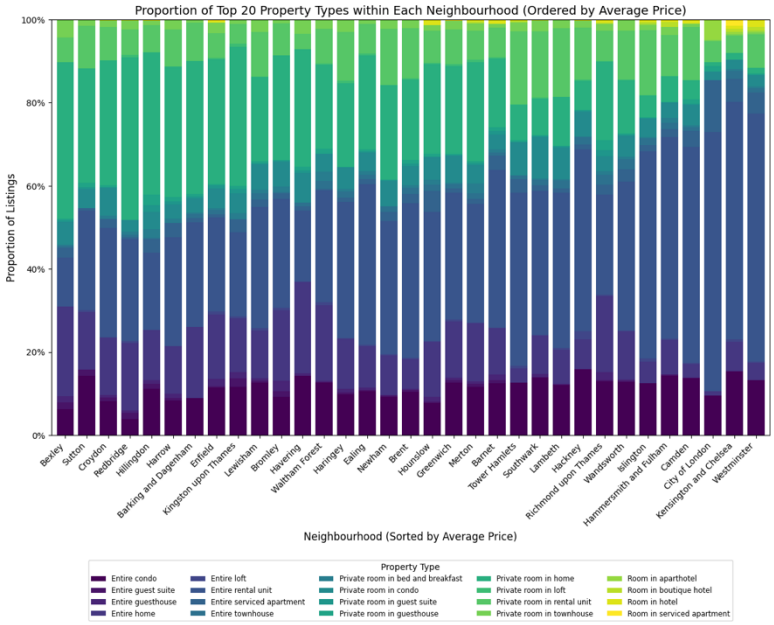
Observations

- There appears to be a weak positive correlation between the number of guests and price, as prices tend to increase slightly with guest capacity. The weak positive correlation suggests that while guest capacity influences pricing, it is not the sole determinant.
- The higher the price the more scattered the plots (in scatterplot) become, this could suggest possible unpredictability with high prices.
- Most listings accommodate fewer guests (1–6), with higher guest capacities (7–10) being less frequent.

- Outliers are visible for high-capacity properties, indicating luxury or unique accommodations limits for properties priced significantly higher than average.
- The line plot provides a clearer view of the trend by showing the average price for each guest capacity.
- Within the line plot, a consistent upward trend is evident, with average prices increasing as guest capacity rises. However, I noticed as accommodation limit increases, price also does, but at a decreasing rate.
- Trends suggest that larger accommodations are priced at a premium. However, fluctuation at high prices highlights potentially less predictable pricing at higher accommodation limits. Clear trend definitely still supports this feature as relevant for modelling pricing predictions.

Average Review Rating, Properties and Neighbourhoods





To explore whether average review ratings influence Airbnb pricing, I created a scatter plot to visualise the relationship between these two features.

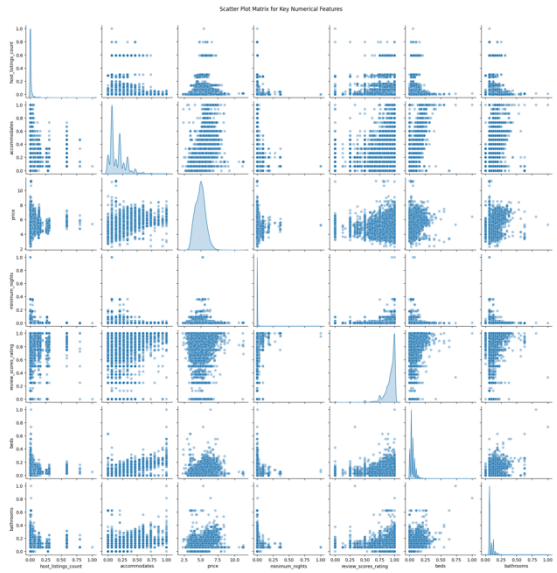
Observations

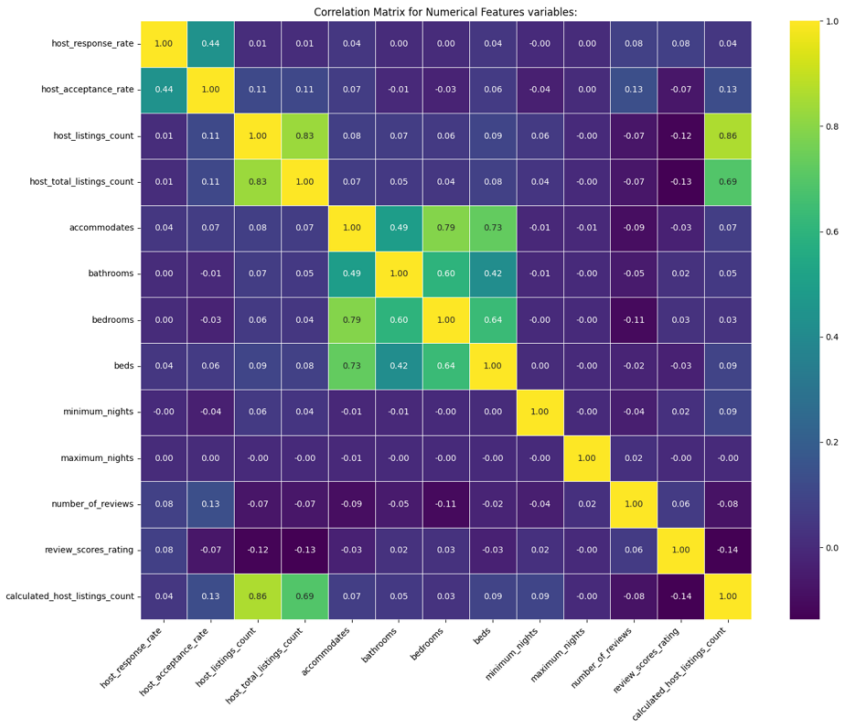
- The scatter plot shows that properties with higher review ratings (above 8 on scale) tend to have slightly higher prices, but the correlation appears weak overall.
- Most listings are clustered around review scores between 4 and 10(on scale), with prices varying widely within this range.
- The lack of a strong trend suggests that while review ratings may contribute to pricing decisions, they are not a primary determinant. This is expected in the hospitality industry, factors like location and property type often outweigh customer reviews in determining price.

To analyse the distribution of property types within neighbourhoods and their relationship to pricing, I created a stacked bar chart showing the proportion of property types across neighbourhoods ordered by average price.

- High-priced neighbourhoods like Westminster and Kensington & Chelsea are dominated by entire homes, serviced apartments, and boutique hotels - property types associated with luxury accommodations.
- Lower-priced neighbourhoods such as Bexley and Sutton have a higher proportion of private rooms and guest suites, which cater to budget-conscious travellers.
- The diversity in property types decreases in high-priced neighbourhood's, suggesting that these areas focus on premium accommodations.

While these observations strongly suggest that both 'neighbourhood' and 'property_type' are valuable in explaining price variations, their combined use in a predictive model could introduce potential multicollinearity...





Scatter and Correlation Matrices

Scatter matrices and correlation matrices are not the same, but they are related and complementary. A scatter matrix gives you a more detailed visual overview of the relationships between variables, including type, strength, direction and outliers. Alternatively, a correlation matrix provides a concise numerical summary of the strength and direction of the linear relationships (between the two variables). However, they both explore the different relationships between features in the dataset.

Observations

- The scatter matrix highlights positive correlations between price and features such as accommodates, bathrooms, and beds. Although these correlations can be considered weak by the graphs.
- The scatter matrix highlights features such as minimum_nights and maximum_nights to show no clear relationship with price, suggesting limited predictive utility.
- The diagonal histograms reveal that most features, including price, have skewed distributions, which were addressed earlier through log transformations and normalisation.

Correlation Matrix Observations

To quantify relationships between numerical features, I computed a correlation matrix and visualised it using a heatmap. This approach was useful for identifying multicollinearity and guiding feature selection a bit later on.

The correlation matrix reveals strong positive correlations between; accommodates and bathrooms (0.79), accommodates and beds (0.73), bathrooms and bedrooms (0.60), host_listings_count and calculated_host_listings_count (0.86).

Moderate correlations can be seen between these features and 'price' (target variable);

accommodates (0.49), bathrooms (0.49), beds (0.42).

Conclusion: Features with high correlations, such as accommodates, bathrooms, and beds, pose a risk of multicollinearity if included together in predictive models.
What is Multicollinearity? Multicollinearity is a phenomenon where two or more predictor variables in a regression model are highly correlated, making it difficult to isolate their individual effects on the dependent variable. Therefore, this occurs when there is a strong linear relationship among the predictor variables in a regression analysis. E.g. since both host_listings_count and calculated_host_listings_count represent similar information, one of these features may need to be excluded during feature selection to avoid redundancy. This is highlighted in the correlation matrix where they have high correlation of 0.86.

Feature Selection

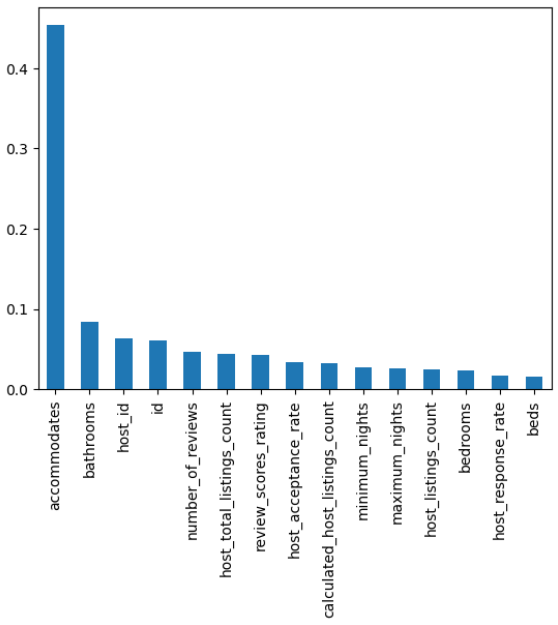
To select features for predicting Airbnb prices for my first linear model, I followed a series of steps/considerations. I began by considering all available features and narrowing them down based on domain knowledge and statistical analysis.
Features Initially Considered Include:

1. **host_listings_count:** Indicates the number of properties managed by a host, which could reflect host experience and therefore pricing strategy
2. **accommodates**
3. **bathrooms:** A key indicator of property quality and capacity (along with beds and bedrooms)
4. **bedrooms**
5. **beds**
6. **maximum_nights:** Reflects rental policies that may influence pricing
7. **neighbourhood:** Captures geographic location, usually a critical factor in pricing
8. **property_type:** Differentiates between accommodation types (e.g., apartments vs. houses).
9. **room_type**
10. **host_is_a_superhost:**Reflects host reliability and quality of service.

Random Forest Regressor

To refine my feature selection further, I used a random forest regressor to evaluate feature importance quantitatively.
To select the most relevant numerical predictors, a Random Forest Regressor, an ensemble learning method, was employed. This technique operates by generating multiple decision trees. First, bootstrap sampling is performed, where random rows of the data are selected with replacement to create different training datasets for each tree. Subsequently, feature sampling is applied, where only a random subset of features is used to build each individual tree, thus promoting diversity among the models. Each trained tree then makes a prediction, and for regression tasks, the final prediction is determined by aggregation, specifically by averaging the predictions of all the individual trees. The resulting feature importance scores from this Random Forest Regressor were then used to rank and select the most influential numerical features. I was able to consider these resulting feature importance scores whilst selecting features for my linear model. I produced a bar chart to visualise the rankings of my numerical features against each other.

Observations



- accommodates and bathrooms emerged as the most influential predictor, confirming its importance in determining price.
- Bathrooms also showed significant importance, aligning with expectations about property quality.
- Features like review scores rating performed relatively poorly despite common assumptions of its relevance. This was highlighted earlier by the weak correlation observed in the scatter plots.

After attained enough insight about the features in the dataset, I was finally able to cross reference my domain knowledge and features selected in random forest regressor, to effectively eliminate between relevant pairs posing the risk of multicollinearity. These pairs of features that are too highly correlated, were identified earlier in the heat mapped correlation matrix.

- host_listings_count were in my original list of features considered with domain knowledge. However, as expected, it

correlates strongly with host_total_listings_count and calculated_ host_total_listings_count. Therefore, using the random forest visualisaiton I selected the feature deemed most influential and went with hosts_total_listings_count

- accommodates and bathrooms are both features that scored highly on with the random forest regressor. But they also have a high correlation to each other according to the correlation matrix. Therefore, I went with selecting accommodates >bathrooms as it scored the highest with random forest.

Based on domain knowledge, random forest results, and correlation analysis, I finalised the following features for my predictive model: **accommodates, bathrooms, host_total_listings_count, host_acceptance_rate, neighbourhood, property_type, host_is_superhost.**

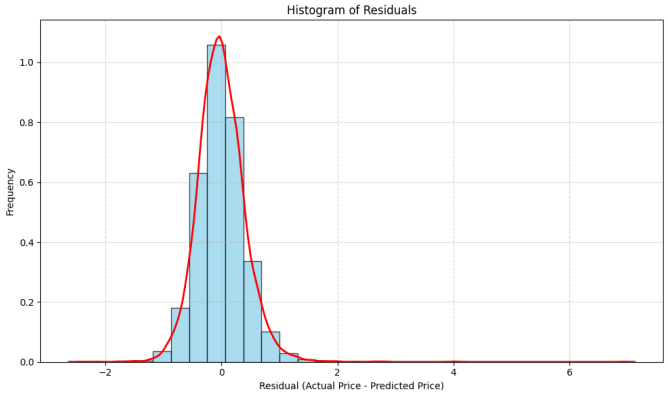
Further justification= Neighbourhood and Property type performed well in previous data visualisations, and for a host to be considered a 'superhost' logically indicates experience and quality in service, which usually influences cost (of any service).

Initial Model

Linear regression models work by establishing a linear relationship between numerical input variables and a numerical output variable. Preparation towards building my initial linear regression model included converting the categorical features from my chosen list of features into dummy variables. I achieved this by using pd.get_dummies()with drop_first=True to avoid multicollinearity. Each variable is converted in as many 0/1 variables as there are different values.

I split the dataset into training (80%) and test (20%) sets using train_test_split() with a random seed (random_state=42) for reproducibility. This partitioning resulted in a training set size of 33,283 samples and a test set size of 8,321 samples. This split is important to evaluate the model's ability to generalise to unseen data, preventing overfitting, where the model learns the training data too well and performs poorly on new data. My training set size was 33,283 samples and test set size was 8,321 samples.

I soon instantiated and trained the linear regression model using LinearRegression() from scikit-learn. The model was trained on the normalised numerical data and dummy-encoded categorical features. To evaluate the performance of my model on the test set I calculated a few key metrics.



- **R-squared (R^2):** The R^2 score was 0.6994, indicating that approximately 70% of the variance in Airbnb prices is explained by the predictors in my model. This demonstrates a reasonably strong relationship between the selected features and prices.
- **Mean Squared Error (MSE):** The MSE was 0.1718, representing the average squared difference between predicted and actual prices.
- A lower MSE indicates better predictive accuracy.
- **Root Mean Squared Error (RMSE):** The RMSE was 0.4145, which provides an interpretable measure of error in price units after log transformation. RMSE is a relative measure. Comparing it to the RMSE of other models or to a baseline model is a good way to assess its performance, therefore I have begun taking the score here.

I analysed the residuals (differences between actual and predicted prices) to assess model fit. The histogram of residuals followed an approximately normal distribution centred around zero, suggesting that the model captures most of the variability in prices effectively.

Cross-Validation: To ensure model stability and robustness, I performed 5-fold cross-validation. K-fold cross-validation assesses model performance by iteratively training and testing on different subsets (folds) of the data. The dataset is divided into *k* folds, and each fold is used once as a test set while the rest are used for training. This method provides a more reliable measure of model performance.

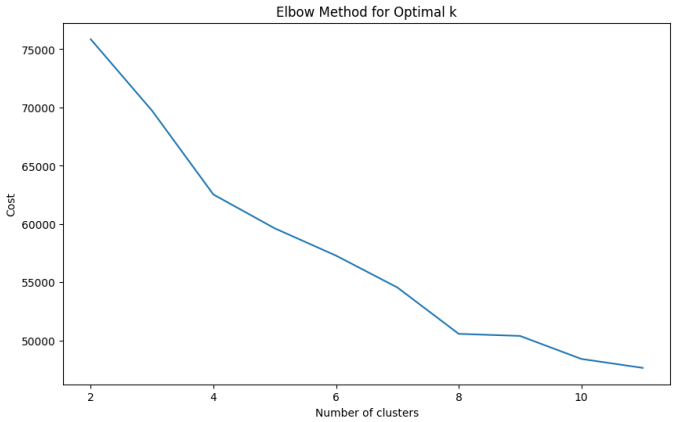
Cross-Validation R^2 Scores: Scores ranged from 0.6665 to 0.7133, with an average R^2 of 0.6834 ± 0.0166. This consistency across folds indicates that the model generalises well to unseen data.

Cross-Validation RMSE Scores: RMSE values ranged from 0.4099 to 0.4523, with an average RMSE of 0.4242 ± 0.0169. These scores confirm that the model maintains reasonable predictive accuracy across different subsets of data.
Some residual outliers suggest areas where predictions deviate significantly from actual prices. This was able to alert me that certain features may require further refinement or transformation to improve predictive power.

Part 1: Improved Model

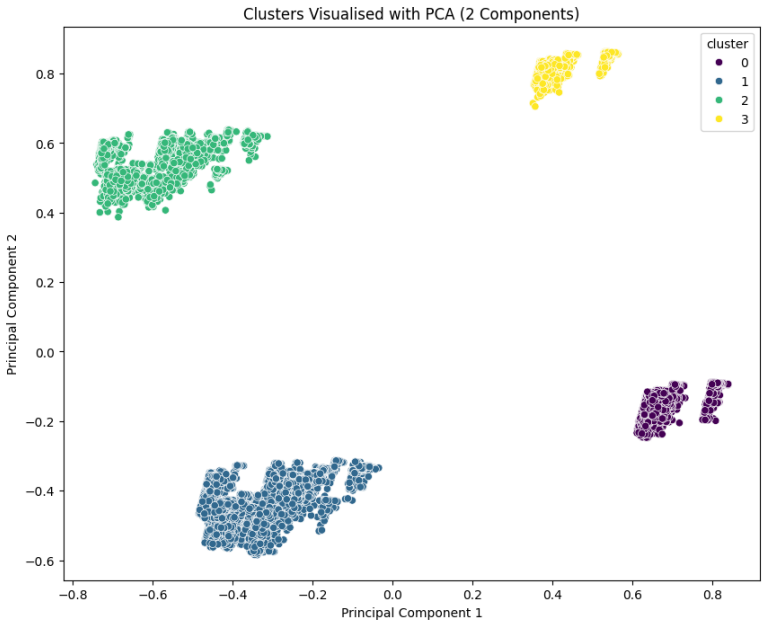
To explore potential local variations in the data, I employed the K-Means algorithm, an unsupervised machine learning technique. K-Means clusters data points into *k* distinct groups based on their feature similarity. Unlike supervised methods that learn from labelled data, K-Means identifies inherent patterns in the data distribution without prior knowledge of class labels. This aligns with the goal of discovering underlying structures within the Airbnb dataset that might influence pricing.

Before applying K-Means, I addressed missing values in the dataset by imputing missing values using the median (for each column). This strategy is robust to outliers, which could skew the mean, thus providing a more representative measure of central tendency.



The elbow method was used to determine the optimal number of clusters (*k*). This method involves plotting the within-cluster sum of squares (WCSS) for a range of *k* values and selecting the "elbow point" where the rate of decrease in WCSS sharply changes. The result of this analysis was the suggestion that *k*=4 was the most appropriate choice for this dataset. The plotted graph shows the elbow forming at k=4. The "elbow" in a K-Means graph indicates the point where adding more clusters provides diminishing returns in reducing the within-cluster sum of squares (WCSS), suggesting an optimal balance between cluster compactness and the number of clusters.

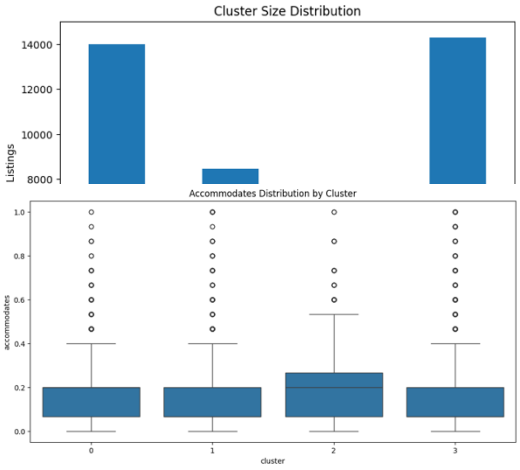
To visualize the resulting clusters, I used Principal Component Analysis (PCA) to reduce the dimensionality of the data to two principal components. PCA is a dimensionality reduction technique that transforms the original features into a new set of uncorrelated variables (principal components) that capture the most variance in the data. This allowed for a 2D scatter plot representation of the clusters, where each point represents an Airbnb listing, and the colour indicates its assigned cluster.



My clusters were generated using the same features as the global regressor (linear model) from Part 1. This was a deliberate choice to ensure a more meaningful comparison between the global model's performance and the performance of local regressors built on these clusters.

To understand the characteristics of each cluster in relation to the overall data distribution, I generated descriptive statistics for each cluster and visualised the cluster sizes.

These visualisations provide insights into the distinct characteristics of each cluster. For example, some clusters might represent more luxury properties with higher average prices and larger accommodation capacities, while others might capture budget-friendly listings.



- The clusters exhibit a notably uneven distribution of data points (listings).
- Clusters 0 and 3 are the largest, containing significantly more listings than clusters 1 and 2.
- Clusters 1 and 2 represent smaller segments of the data, suggesting these clusters might capture less frequent or more specialised listing types.

Local Regressor vs Global Regressor

Following the clustering, I built local regression models. A global regressor, such as the linear regression model used in Part 1, attempts to model the relationship between features and the target variable across the entire dataset with a single equation. In contrast, local regressors fit separate models to subsets of the data, in this case, the clusters identified by K-Means. The hypothesis is that local regressors can capture more nuanced relationships that a global model might miss, potentially leading to improved prediction accuracy within each cluster.

I trained a separate linear regression model for each of the four clusters. The performance of these local regressors were then evaluated using Root Mean Squared Error (RMSE) and R-squared (R2) score, the same metrics used to evaluate the global regressor in Part 1.

```
Global Model (from Part 1): RMSE = 0.41, R2 = 0.70
 Local Model Performance:
 Cluster 2: RMSE = 0.50, R2: 0.49
 Cluster 1: RMSE = 0.39, R2: 0.78
 Cluster 0: RMSE = 0.38, R2: 0.46
 Cluster 3: RMSE = 0.44, R2: 0.67
```

The results revealed a mix. Some local regressors (e.g. Cluster 1) outperform the global model in terms of R2, suggesting a better fit within that specific cluster, whilst others did not. Cluster 0 also exhibits a lower RMSE than the global model. However, other clusters (e.g. Cluster 2) show worse performance.

Possible reasons for these variations include:

- Clusters may have high internal variability might be more challenging to model accurately. If the cluster has high variability (e.g. properties are very different with a lot of different elements), it's harder to predict their prices accurately with a single model.
- The number of data points within each cluster varies. Smaller clusters might suffer from overfitting, leading to poorer generalisation performance.

Overall, while the cluster-based approach shows promise, it does not outperform the global regressor. This highlights the importance of careful cluster analysis, model selection, and validation when employing local regression techniques as opposed to global regression models.

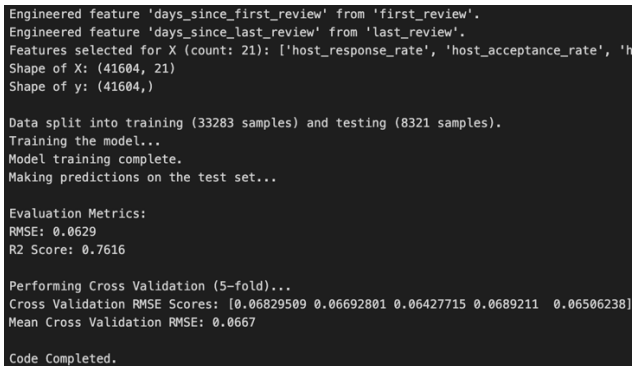
Final Improved Predictive Model

Considering all I'd learnt up to here, this is how I decided to implement my improved predictive model.
Feature Engineering: I engineered new features from the date variables 'first_review' and 'last_review'. I achived this by creating code that calculated the number of days since each review, creating 'days_since_first_review' and 'days_since_last_review'. This transformation captures the longevity of listing activity, which is likely to influence price.

Updated Feature Selection: To identify the most relevant predictors and reduce overfitting, I employed the SelectKBest method. This technique selects the *k* best features based on their F-statistic, which measures the linear relationship between each feature and the target variable. I set *k*=10, effectively choosing the top 10 features. This feature section was performed on a wider range of features than the range of features in part 1.

Why linear regression again? I focused on linear regression as it has proven to be an effective and interpretable model for this type of data, as seen in Part 1. While other models were considered, linear regression consistently yielded strong results. For example, I experimented with Ridge regression, a regularised form of linear regression that can help to prevent overfitting. However, the standard linear regression model performed comparably or better, suggesting that overfitting was not a major concern in this particular case.

The model was trained on the training set, and its performance was evaluated on the test set. I used 5-fold cross-validation to obtain the more robust estimate of the model's performance.



Metrics for the final improved model are included.

Compared to the global model from Part 1 (RMSE = 0.41, R2 = 0.70), the final improved model demonstrates a significant improvement in RMSE and a modest improvement in R2. The reduction in RMSE shows that the improved model's predictions are, on average, much closer to the actual prices. The cross-validation results further validate the model's robustness and generalisation ability.

The enhanced performance can be attributed to the combined effect of the alternate implementations such as the refined feature engineering. These methods capture the underlying patterns in the data more effectively, leading to more accurate price predictions.

Potential further improvements to my model:

- Experimenting more with models beyond linear regression (e.g. XGBoost) that might even still adhere to linear relationships but capture more complexities.
- Implementing nested cross-validation for reliable model evaluation and to prevent overfitting.
- Implementing RFE and comparing it to SelectK performance wise.