Title: Predicting London Airbnb Prices using Regression, Clustering and Ensemble Models

Module: CO3093 - Big Data and Predictive Analytics

Student ID: 229022263

Written by David Lawal

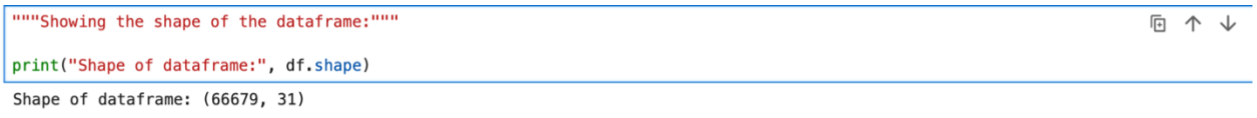
Signed by David Lawal

Introduction

This report presents a comprehensive analysis of London Airbnb listing data with the aim of developing predictive models for property prices. Suitable data cleansing, transformation and exploratory analysis were done on the dataset to enable me to derive various regression models, clusters and ensemble learners.

Exploring the data

The dataset which was used is titled “London_Listings.csv” it was a vast dataset which initially included 66,679 rows of data and 31 columns which were descriptive features of the Airbnb properties. For example: price, neighbourhood, room type etc.



```
"""Showing the shape of the dataframe:"""

print("Shape of dataframe:", df.shape)
```

```
Shape of dataframe: (66679, 31)
```

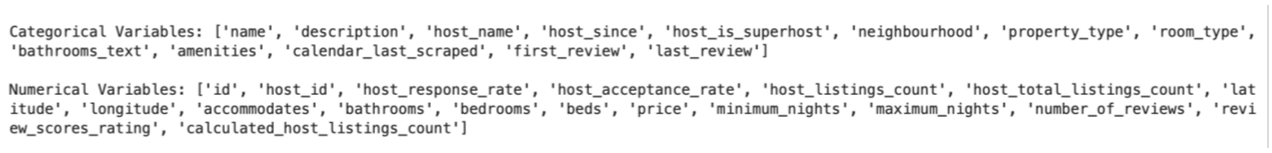
To ensure the accuracy of any models which I was to build, it was imperative that I cleaned and transformed the data first, as this is fundamental data science theory. However, as I explored the dataset more, I began to see the importance of data cleansing and transformation in a practical way.

Data cleansing and transformation

For this part of the assignment, the main aim was to identify missing values and treat them whilst also improving the overall quality of the dataset.

Creating a list of categorical variables and numerical variables

After seeing the initial set of the dataframe, then next thing I did was to categorise each column as either a categorical variable or numerical variable.

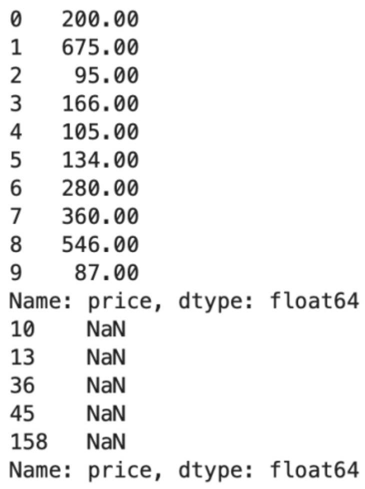


```
Categorical Variables: ['name', 'description', 'host_name', 'host_since', 'host_is_superhost', 'neighbourhood', 'property_type', 'room_type',
'bathrooms_text', 'amenities', 'calendar_last_scraped', 'first_review', 'last_review']

Numerical Variables: ['id', 'host_id', 'host_response_rate', 'host_acceptance_rate', 'host_listings_count', 'host_total_listings_count', 'lat
itude', 'longitude', 'accommodates', 'bathrooms', 'bedrooms', 'beds', 'price', 'minimum_nights', 'maximum_nights', 'number_of_reviews', 'revi
ew_scores_rating', 'calculated_host_listings_count']
```

This was a vital step as it enabled me to see which processing techniques, I needed to apply to what would become the predictors for my model.

Cleaning the price column

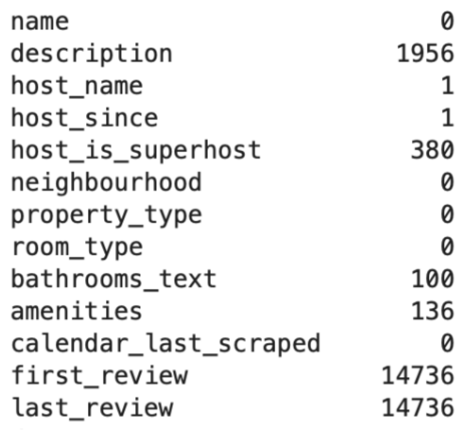
As this dataset was the about Airbnb listings, the price column had undesirable characters such as “$’ and “,” which needed to be removed. Even after cleansing, some price values were converted to NAN. This is due to them not being able to be converted to a number at all. This type of data is bad for predictive models.



```
0   200.00
1   675.00
2    95.00
3   166.00
4   105.00
5   134.00
6   280.00
7   360.00
8   546.00
9    87.00
Name: price, dtype: float64
10    NaN
13    NaN
36    NaN
45    NaN
158   NaN
Name: price, dtype: float64
```

Removing empty strings/lists from categorical variables

This was also another important step, as missing data values are useless for machine learning algorithms. Therefore, they need to be labelled as such “NAN” or else it could cause potential problems. Below shows a summary of how many NAN valued rows there were in each categorical column



```
name                        0
description              1956
host_name                   1
host_since                  1
host_is_superhost         380
neighbourhood               0
property_type               0
room_type                   0
bathrooms_text            100
amenities                 136
calendar_last_scraped       0
first_review            14736
last_review             14736
```

Summary of all missing values

A summary of missing values across the dataset, as well as summary statistics were generated. This revealed that some important predictors contained large gaps in data. For example, review scores rating, first review and last review each had over 22% missing values, while variables like host response rate and host acceptance rate had 10-15% missing data. Even the target variable price, had approximately 7.9% missing entries.

Dropping unnecessary columns

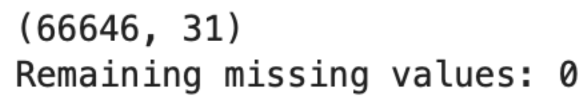
Some columns were dropped due to their lack of predictive value, high redundancy and incompatibility with machine learning models. I decided to drop columns id and host id as they served only as unique identifiers, they held no meaningful relationship with the target variable (price). The columns name, description, host name and amenities I deemed to be too unstructured due to their highly unique nature. Also, they would be difficult to quantify. Including these models would introduce excessive noise or require extra processing. Latitude and longitude were excluded I believe neighbourhood to be a better suited geographical predictor, for this assignment. Calendar last scraped, first review and last review are all unnecessary metadate, so I decided to drop those. Finally, bathrooms text was redundant due to their being a bathroom's column.

Dropping duplicates

Surprisingly I found there to be no duplicate columns.

Dropping rows with NAN values

This step was done to ensure the dataset used for model training is as complete and reliable as it can be. Although imputation techniques exists, the nature of the dataset and assignment was better suited for not attempting to guess missing values.

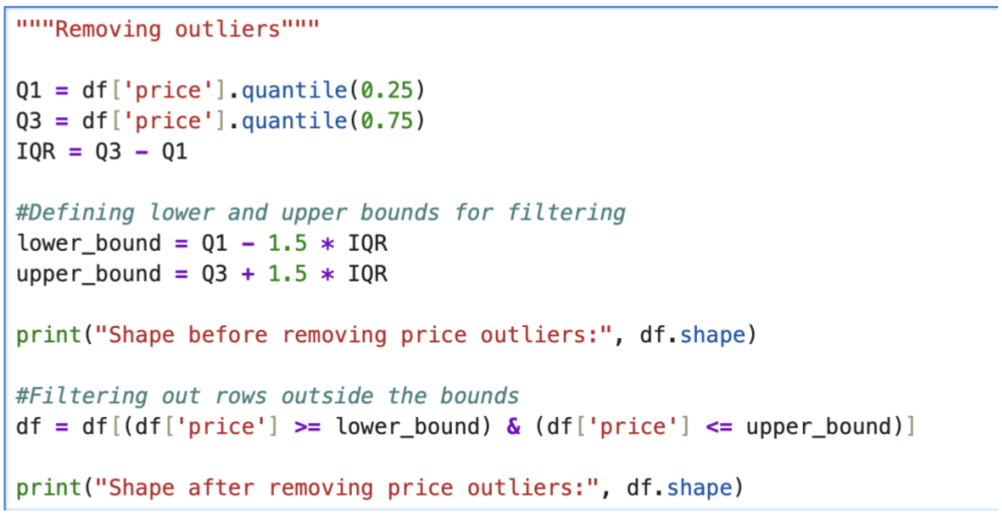


```
(66646, 31)
Remaining missing values: 0
```

If we compare the shape of the data frame know to what it was at the start we can see that only 33 columns have been dropped (66,679 – 66,646). Which is not very significant.

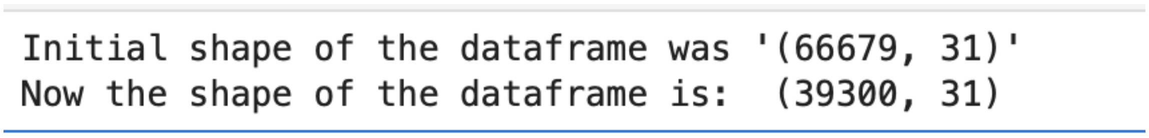
Identifying and removing outliers

In the context of Airbnb pricing, outliers may represent luxury listing or even incorrectly entered prices, both of which don't reflect the general market. To address this the interquartile range (IQR) method was applied to the price column. This was done as a measure to prevent extreme values from skewing regression coefficients.



```
"""Removing outliers"""

Q1 = df['price'].quantile(0.25)
Q3 = df['price'].quantile(0.75)
IQR = Q3 - Q1

#Defining lower and upper bounds for filtering
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

print("Shape before removing price outliers:", df.shape)

#Filtering out rows outside the bounds
df = df[(df['price'] >= lower_bound) & (df['price'] <= upper_bound)]

print("Shape after removing price outliers:", df.shape)
```

Showing the shape of the resulting dataframe



```
Initial shape of the dataframe was '(66679, 31)'
Now the shape of the dataframe is:  (39300, 31)
```

The result of the data cleansing and transformation is a reduced dataset by 41%. This shows the trade between quantity and quality of data.

Log transformation of price and predictor normalisation

To prepare the dataset for regression and clustering algorithms we log scaled the target variable (price) and normalised the numerical features.

Data exploration

Neighbourhood and price

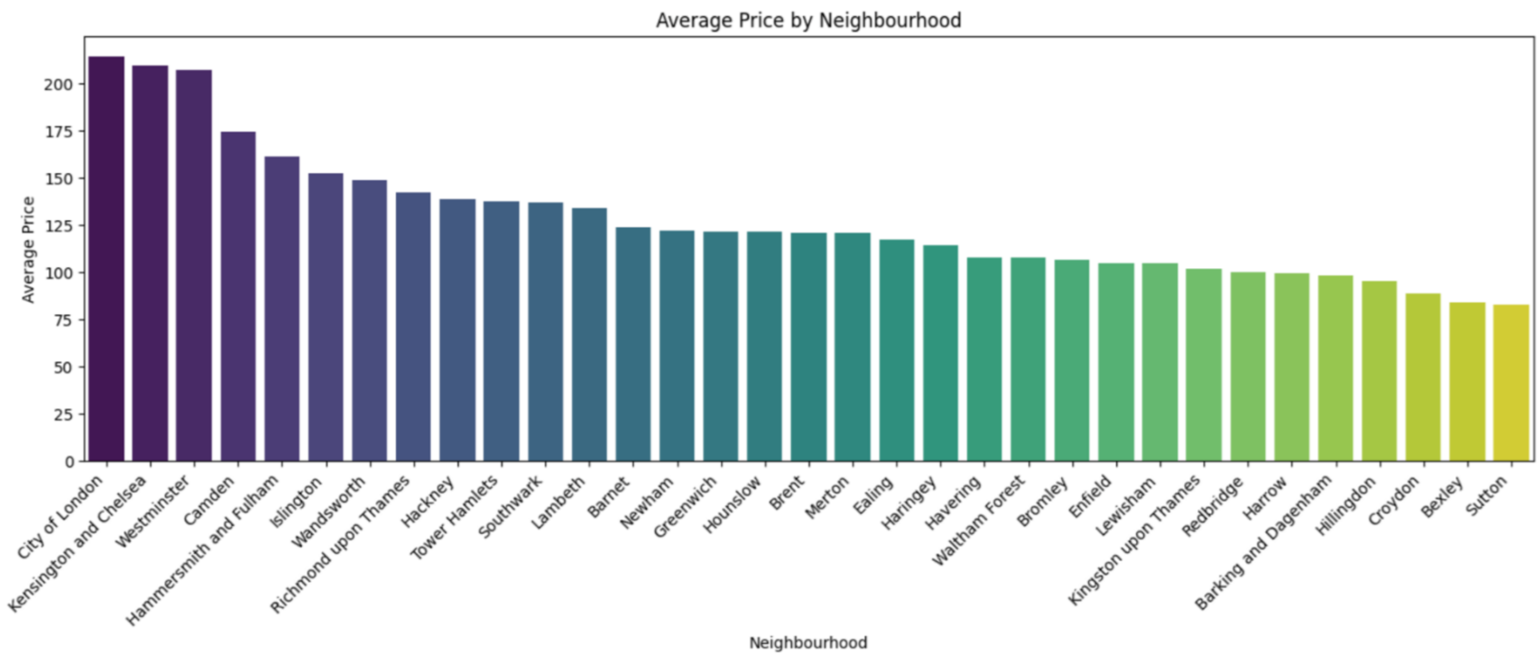


Figure 1 Bar plot of average price neighbourhood

Figure 1 shows a bar plot of the average prices of Airbnb's across different neighbourhoods from highest to lowest. There is a significant variation in pricing across the different boroughs of London, prices can differ by up to approximately 63%. At the top end are the likes of the City of London, Kensington and Chelsea and Westminster, each with averaged price Airbnb's more than £200 per night. These boroughs are central and a hot spot for tourists, which may explain the expensive prices.

On the other side of the spectrum, outer boroughs such as Sutton, Bexley and Croydon have the lowest average prices, typically below £100 per night. These are areas are further from central London, and have a lower tourist attraction, reasons for which the price may be lower.

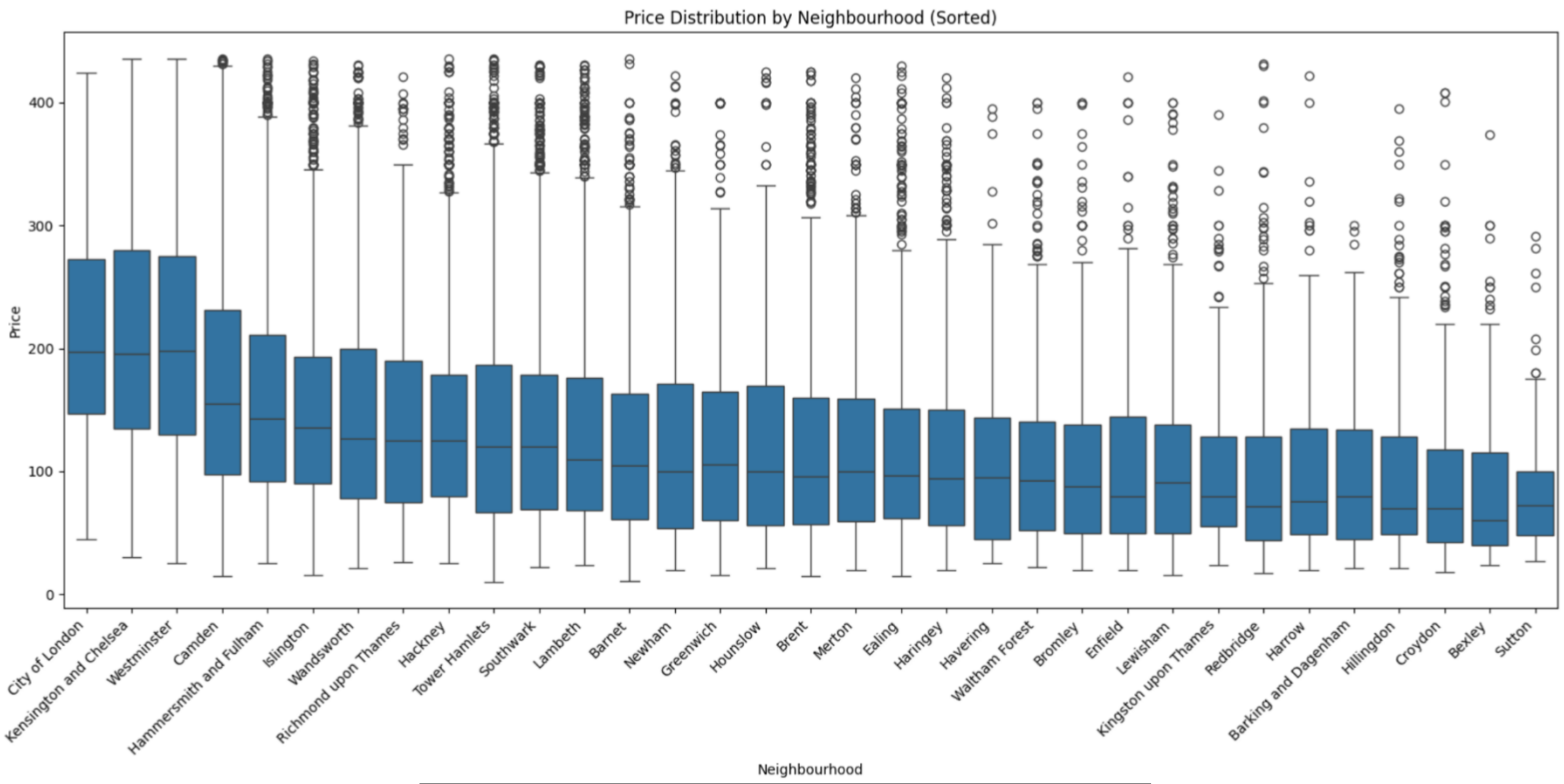


Figure 2 Box plot of price distribution by neighbourhood

Figure 2 tells us that not only do central boroughs like the City of London have the highest average prices, but only the widest interquartile ranges (IQR). This indicates a high price variability. Contrastingly, boroughs Like Croydon show lower median prices with narrower IQRs, this suggest more consistent and affordable airbnbs in those areas.

This plot also confirms the right skewed nature of price distributions. Numerous outliers (marked as dots) are shown above the whiskers. This further shows the need for a log transformation to be applied to the price variable during the data transformation and cleansing part of the assignment.

These two figures tell us that neighbourhood is a key predictor in the price of airbnbs, this helped form my decision of choosing it as one of my categorical predictors for the predictive model

Price across tenants

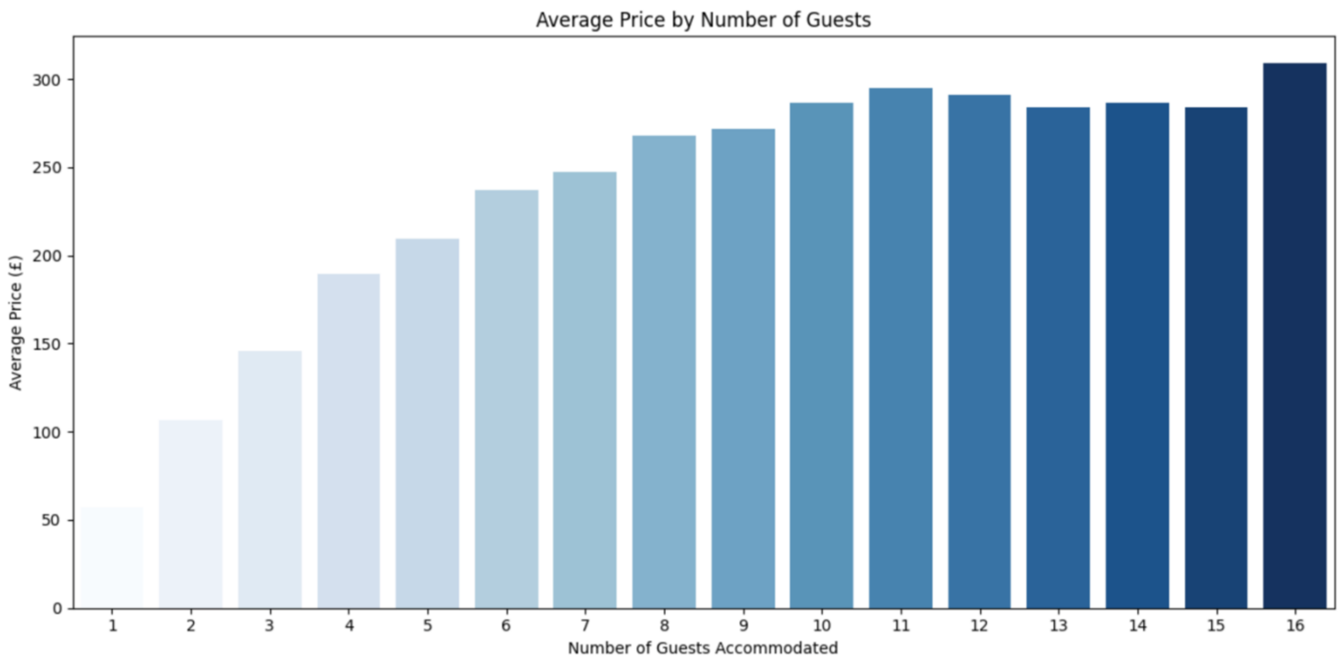


Figure 3 A bar plot of average price per number of guests

Figure 3 shows a clear trend; as the guest capacity increases so does the average price. This upwards trend shows a strong positive correlation between the number of tenants (guests) and the price. Hence accommodates is a key predictor in determining the price of an Airbnb.

Average Review Rating and Price

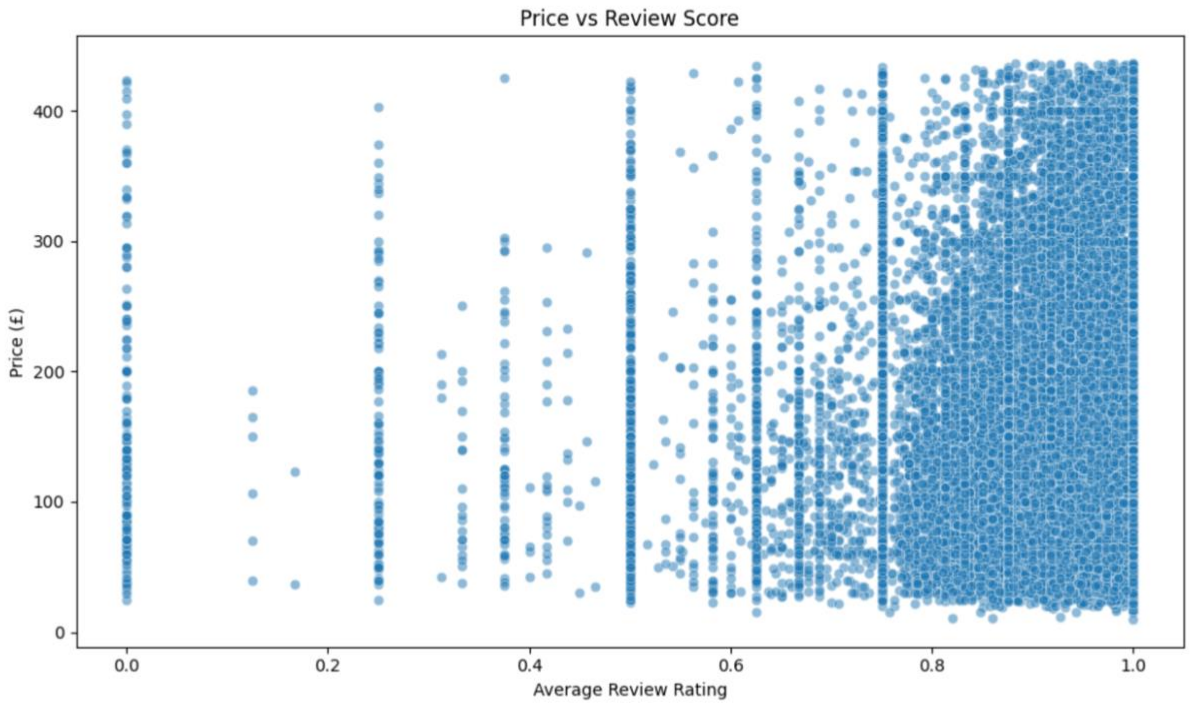


Figure 4 A scatter plot of

To investigate the relationship between the average review rating and the price, a scatter plot was plotted. The plot fails to show any sort of positive linear correlation between the two variables, which would be useful for a linear regression model. Listings with high review ratings span across a range of prices. Despite this, there seems to be something noticeable going on, there appears to be several straight vertical lines, and a high concentration of data points towards the right. The multiple straight vertical lines suggest that many of these listings share very similar or even identical review score ratings. The concentration of these points being towards the right that the average of these scores is quite high.

Despite this interesting trend, the review rating alone doesn't seem to be a good predictor of price, at least not for a linear regression.

Scatter matrix plot and correlation matrix

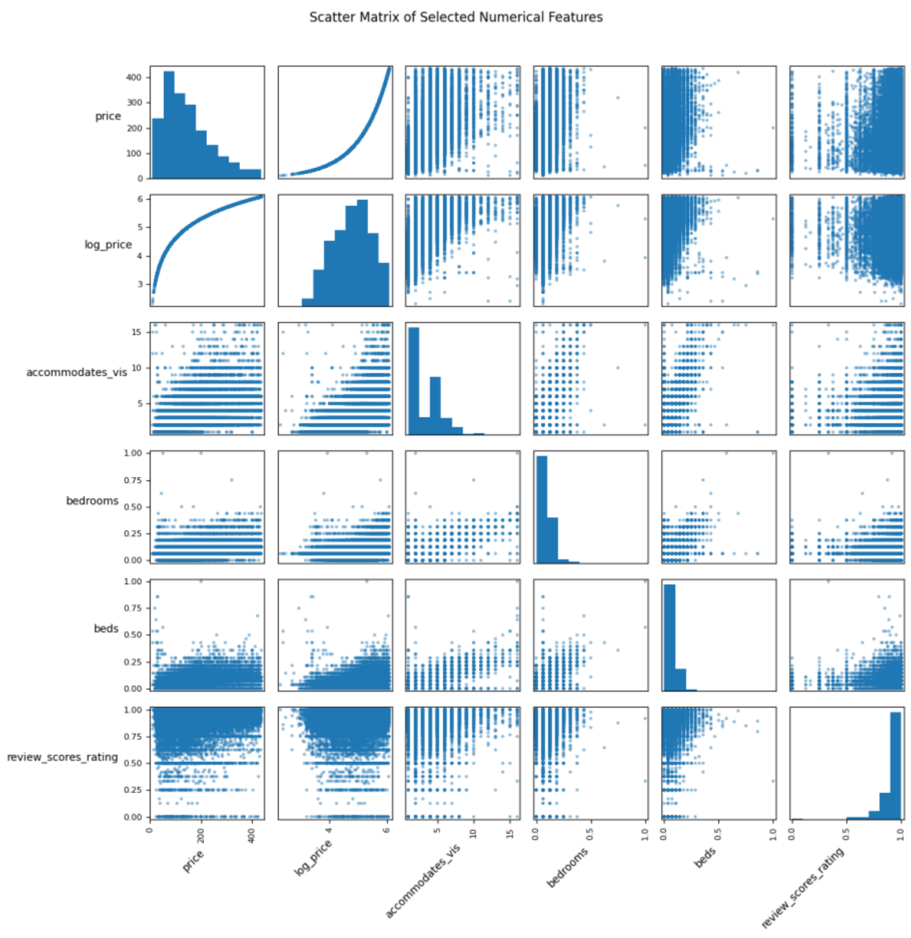
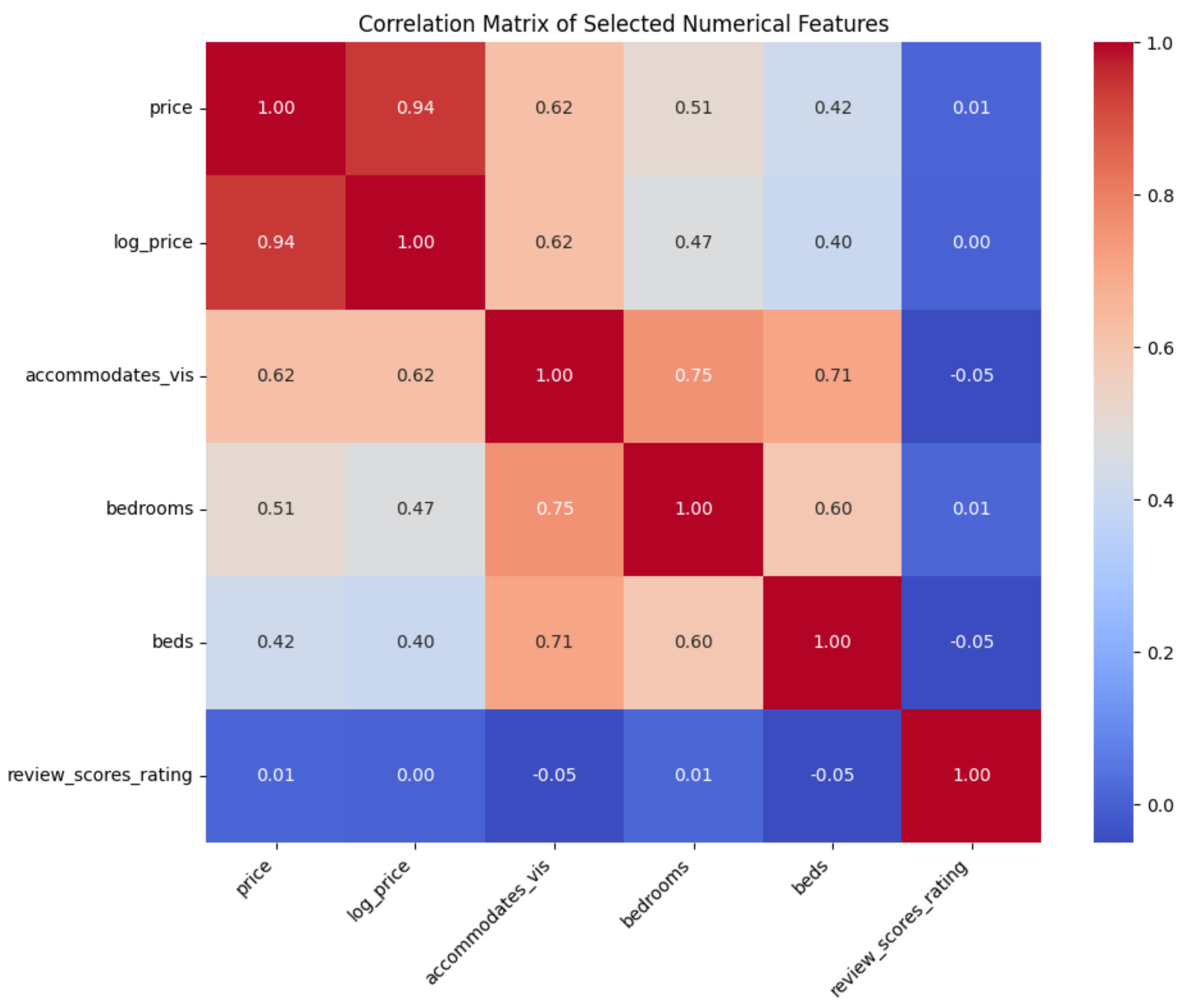


Figure 5 A scatter matrix between numerical predictors



	price	log_price	accommodates_vis	bedrooms	beds	review_scores_rating
price	1.00	0.94	0.62	0.51	0.42	0.01
log_price	0.94	1.00	0.62	0.47	0.40	0.00
accommodates_vis	0.62	0.62	1.00	0.75	0.71	-0.05
bedrooms	0.51	0.47	0.75	1.00	0.60	0.01
beds	0.42	0.40	0.71	0.60	1.00	-0.05
review_scores_rating	0.01	0.00	-0.05	0.01	-0.05	1.00

Figure 6 A correlation matrix showing selected numerical features

The scatter matrix (figure 5) visualises pairwise relationships between key numerical features: price, log price, accommodates, bedrooms, beds and review scores ratings. The plots show us that there are positive correlations between log price and accommodates, bedrooms and beds.

The correlation matrix shown in figure 6 is probably easier to interpret than the correlation matrix, due to the simplicity if the of reading numbers off the scale. A higher number in the shows a higher correlation between the two predictors. Similarly to the scatter matrix, the correlation matrix corroborates the strength of linear relationships between some key variables. For example, accommodates (shown as accommodates_vis on the plot, an unnormalized accommodates column kept for visualisations), has a moderate correlation between both price and log price of 0.62. Bedrooms and accommodates also have an obvious high correlation score of 0.75.

Despite this, figure 6 gave me more insight into how useful these variables are in predicting the price. Just from considering this correlation matrix only, out of all four predictors, only accommodates had a high enough correlation with price directly to be considered a very strong key predictor. The second highest was bedrooms which had a correlation score of 0.51 with price and 0.40 with log price.

Further plots

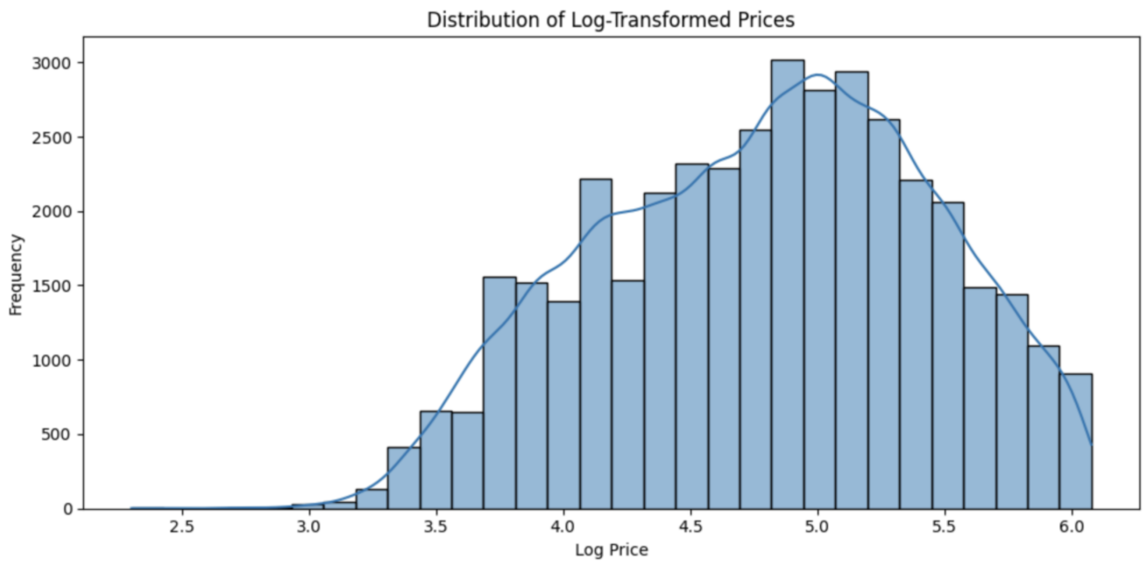


Figure 7 Histogram of log price

Figure 7 confirmed that applying a log transformation to a likely highly skewed price column, produced an approximate normal distribution, which is ideal for linear regression.

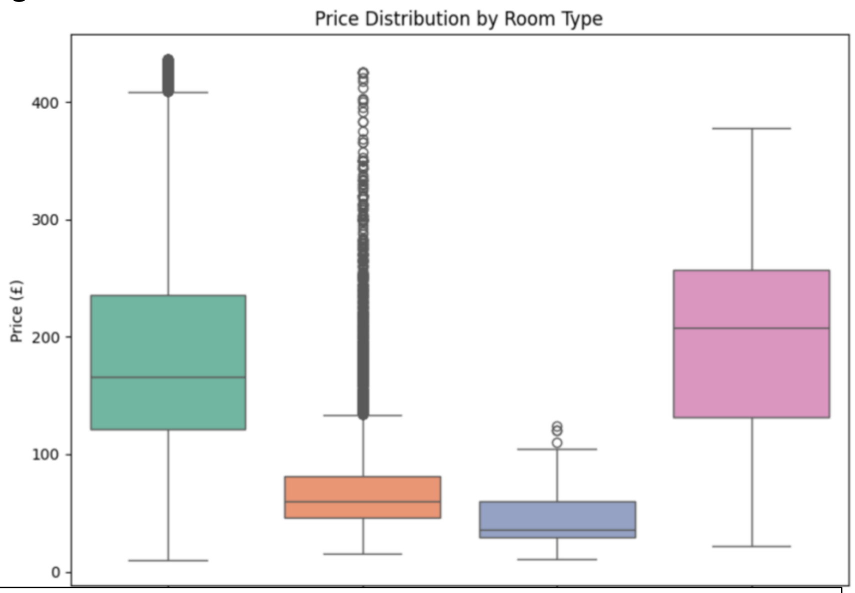


Figure 8 Box plot comparing price distribution by room

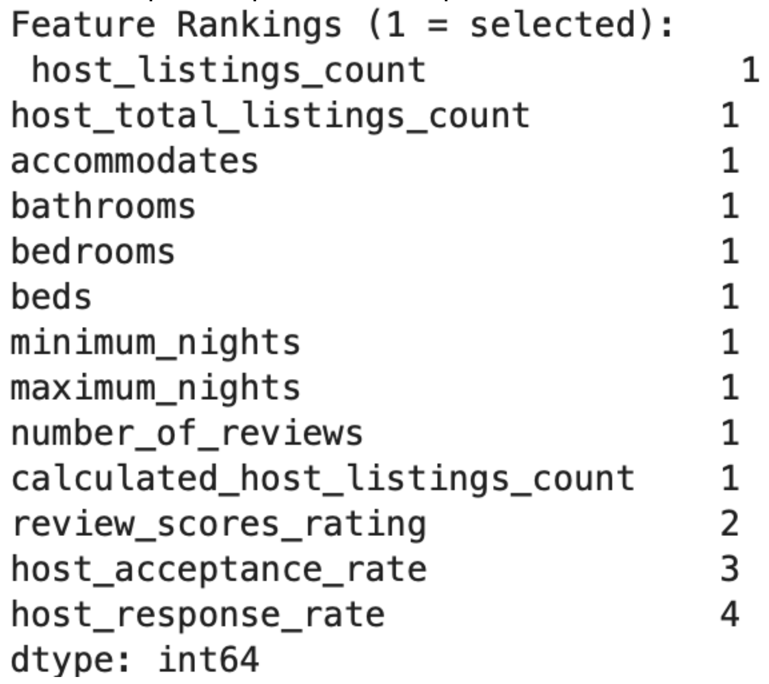
Figure 8 shows how comparing different room type revealed significant price variation. This insight was one of the reasons why I chose to include room type as a categorical feature in my predictive model.

Model Building

The aim of this section was to build and evaluate an initial regression model to predict the AirBnb prices. The key to this was selecting impactful and relevant predictors.

Predictor selection

The first thing to do to build the model was to select the predictors. There are too many columns to use for a simple regression model, also not all of them are entirely relevant as discussed in the data exploration and transformation section, therefore it's vital to choose the best ones. So that the model can be as accurate as possible. To do this, I put all the numerical predictors through a recursive feature elimination (RFE) with a linear regression estimator. I made sure to exclude the log price, as this was the outcome variable which we wanted to obtain. RFE is a backwards selection technique that ranks features by recursively removing the least significant ones. This was done to find the top 10 most impactful predictors for price.



```
Feature Rankings (1 = selected):
 host_listings_count               1
host_total_listings_count         1
accommodates                      1
bathrooms                         1
bedrooms                          1
beds                              1
minimum_nights                    1
maximum_nights                    1
number_of_reviews                 1
calculated_host_listings_count    1
review_scores_rating              2
host_acceptance_rate              3
host_response_rate                4
dtype: int64
```

Most of the predictors which were selected were the ones which had some sort of positive correlation with price and log price, as discussed and explored in the data cleansing and transformation section.

I made two linear regression models, the first one performed quite poorly with a low R squared value of 0.3873 and an RMSE of 0.5136. Which tells us that the model only explained approximately 39% of the variation in the target variable and that the model is wrong more than 50% of the time on average.

To improve on these poor metrics, I went back to my predictors and decided to some high impact categorical predictors also. These were neighbourhood (due to strong visualisations between neighbour and price which I discovered earlier on), property type and room type. As linear regression works with numerical values, these selected categorical variables had to be encoded using one-hot encoding. Meaning they had to be converted into binary columns, enabling the model to use them as numerical features. Once I did this, I concatenated them with the rest of the numerical predictors to form the full input matrix “X_Improved”.

Linear Regression models

I created two linear regression models, the first one was trained with only the top 10 numerical features and performed rather poorly. However, the second model which I created had significant improvement. The new set of predictors “X_improved”, had a significant impact. After training the model in the same was as I did the first, only choosing the predictors, the metrics improved drastically.

First Linear regression model	Second linear regression model
R squared value: 0.3873	R squared value: 0.7181
RMSE(Root Mean Squared Error): 0.5136	RMSE(Root Mean Squared Error): 0.3484
MAE(Mean Absolute Error): 0.2721	MAE(Mean Absolute Error): 0.2721

As you can see this single change, accounted for a 53.93 % increase in the R squared value and a 32.17% decrease in the RMSE. However, the MAE of both is the same. This suggest that the reason why the second regression model improved so much on those two metrics was due to the size of the errors. The two models are making a similar number of mistakes; however, the second one simply makes “smaller” or less critical mistakes than the first, which accounts for a higher R squared value and a lower RMSE.

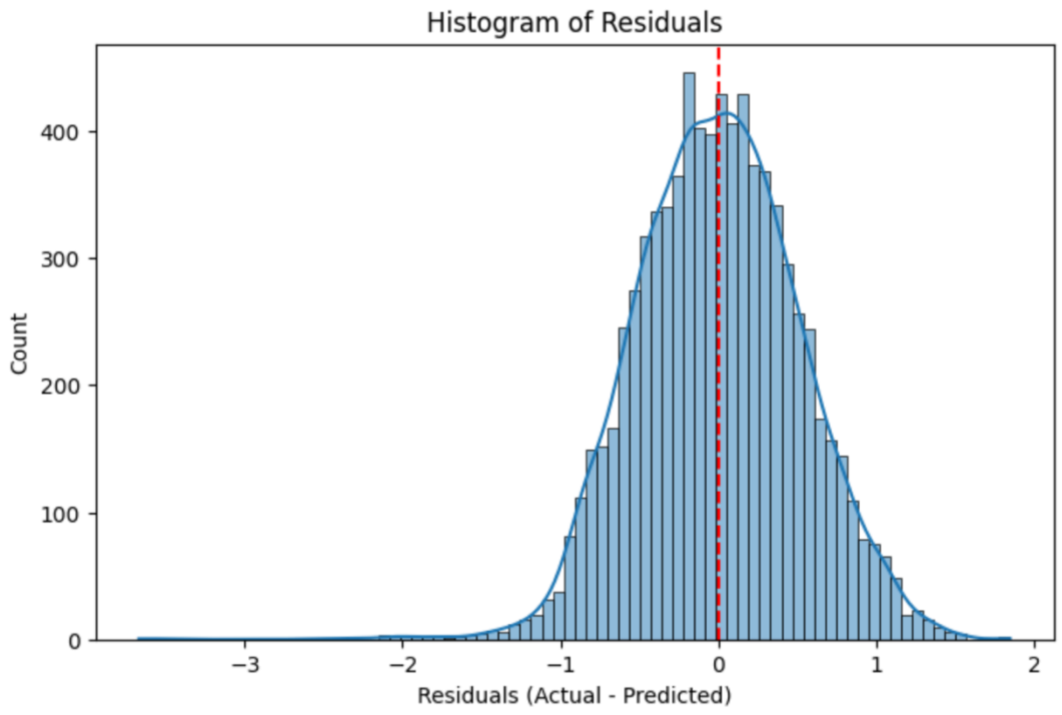
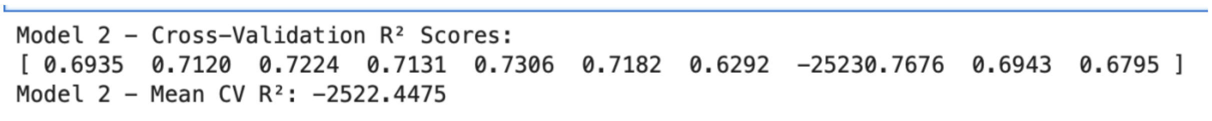


Figure 9. A histogram of residuals

Residuals = Actual value – predicted value. They tell you how far the model's prediction was from the real value. To evaluate the performance of the linear regression, model a histogram of residuals was plotted, as shown by Figure 9. The residuals were resembled a normal distribution. In terms of shape, and being centred around 0, with a similar variance also. Figure 9 is evidence that the linear regression model's predictions are generally unbiased, and it complies with the assumptions for a linear regression

Cross- validation

As part of a means to evaluation my model, cross validation was used to assess how well it generalises on unseen data which it hasn't trained with. I used10 fold cross validation, this, splits the data into 10 parts and trains the model on 9 parts and tests the model on the other 1. It repeats this 10 times. The k fold approach increases reliability of testing due to the repetition, which in turn helps reduce overfitting of the model.


```
Model 2 - Cross-Validation R² Scores:
[ 0.6935  0.7120  0.7224  0.7131  0.7306  0.7182  0.6292  -25230.7676  0.6943  0.6795 ]
Model 2 - Mean CV R²: -2522.4475
```

The results of the cross validation suggest that the model generalises well. It showed consistent R squared values, ranging between 0.6292 and 0.7306 on 9 out of the 10 folds. However, on one-fold the r squared value was extremely poor -2522.4475. This shows that for that fold something is wrong with the model. As even just guessing would produce a higher score. Something may have gone wrong during the training for the model to perform like this, especially after 9 consistent good folds. For example, maybe an inverse relationship was mistakenly formed during the testing. Considering the absurdity of that one-fold, it may be wise to consider that as an outlier and exclude it from the average r squared for the fold. As shown in the screenshot above the outlier make the average extremely negatively skewed. Removing that outlier, the average becomes 0.6992.

<u>Improved Model</u>

This purpose of this part of the assignment was build an improved predictive model, which beats the one built using linear regression. To achieve this K-means clustering was applied to segment listings based on key features.

K means is an unsupervised machine learning algorithm which is used to group similar data into clusters. It does this by finding k number of distinct groups in a dataset, based on the patterns in the predictor variables.

Determining the optimal K
The first task was to choose the number of clusters. I decided to use the elbow method to decide this.

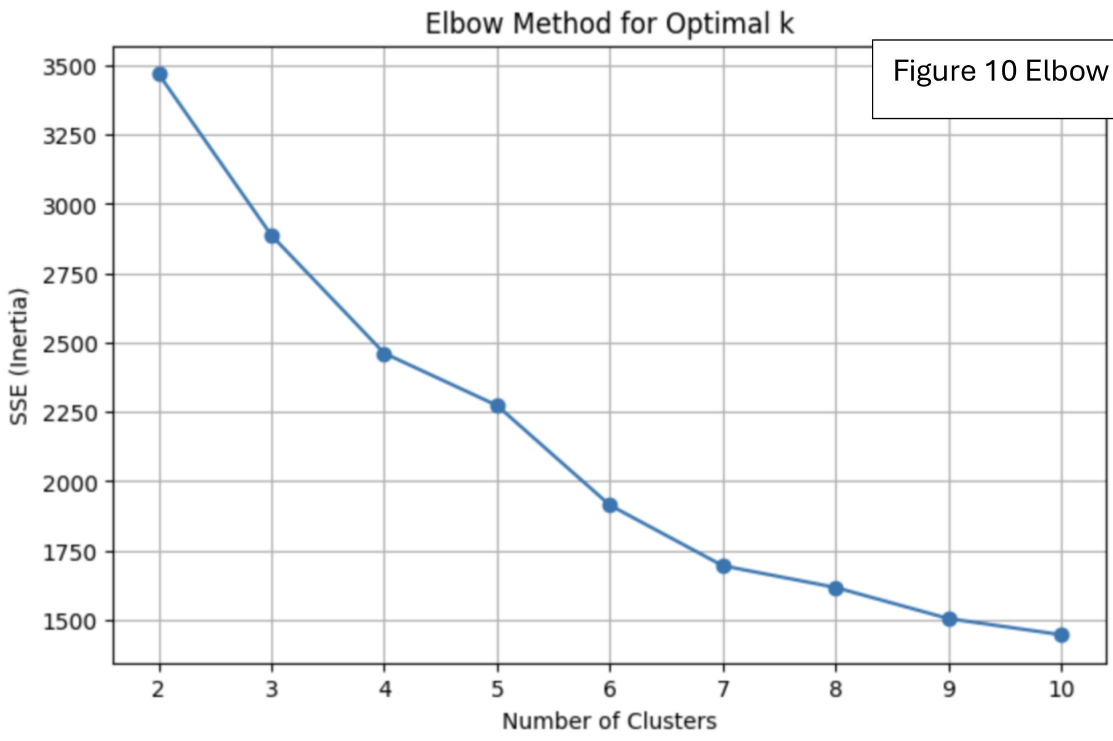
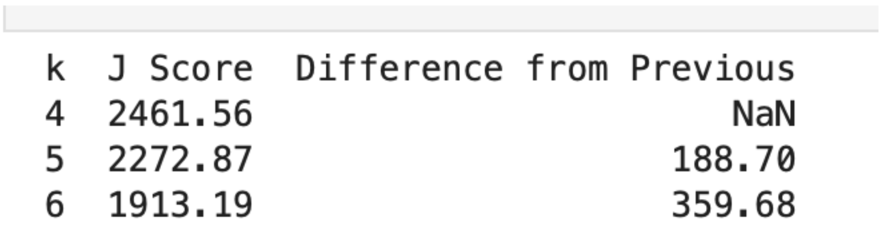


Figure 10 Elbow method

The elbow method allows you to visually find K, by looking at where the SSE stops decreasing significantly and the curve begins to flatten out. However, from the graph above it wasn't exactly clear whether the value was 4,6 or even 5. Therefore, I used the J test score test to decide.



```
k  J Score  Difference from Previous
4  2461.56                       NaN
5  2272.87                    188.70
6  1913.19                    359.68
```

As the screenshot above shows the most significant decrease was ,moving from K=5 to K=6, so I chose K as the most optimal amount of clusters.

Scaling features

Like we discussed in the previous sections, predictors for machine learning models are very important. We used the same technique which we used last section to choose the predictors, however this time we needed to scale. This was done using StanderScaler. Which is a data preprocessing tool, from the sklearn library. It subtracts the mean and divides by the variance. This is important, so the module doesn't assume that all features are on the same scale which can cause skewed results and unfair distances between clusters. In addition to this I use PCA to reduce the dimensionality of the data, which aimed at keeping the most vital information whilst removing as much noise and redundancy as possibly, which will ultimately reduce overfitting.

Cluster Visualisation

Figure 11 Cluster visualisation1

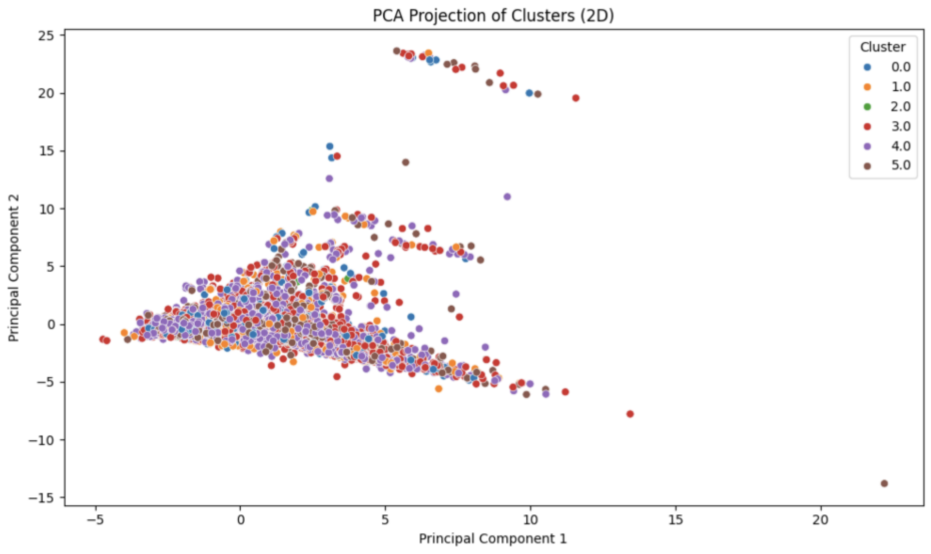


Figure 12 Cluster visualisation 2

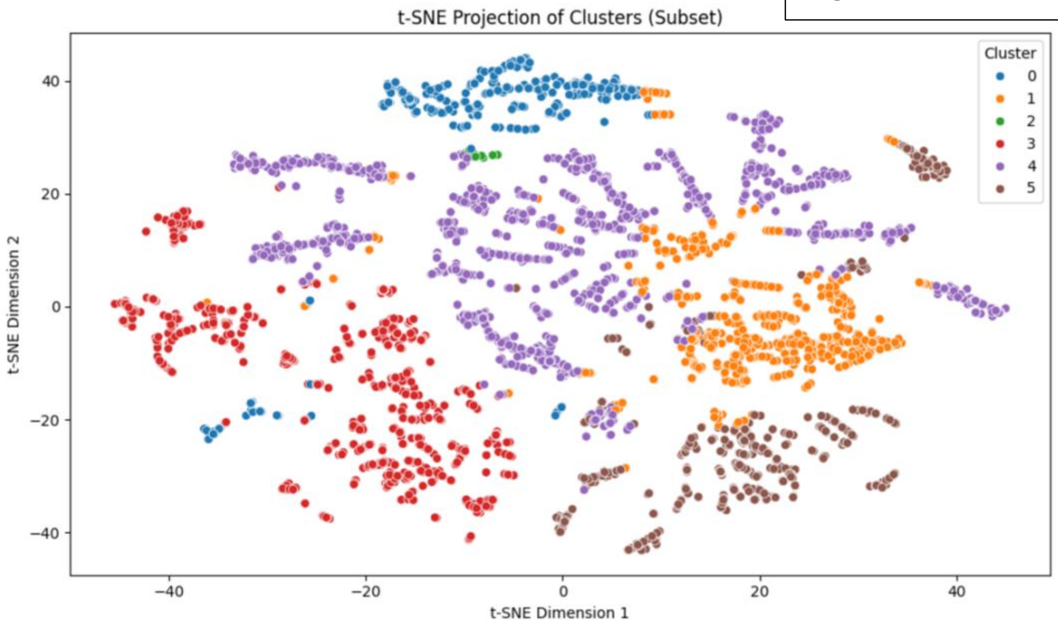


Figure 11 shows my 6 clusters. This visual was created with PCA 2D cluster projection. However, the visuals are not great as not clear distinct groups are visual, it just looks like one big cluster of data points. To improve upon this, I used a different method for visualising my cluster call t-SNE projection, the results are shown in figure 12. The results shown improvement. You can see the formation of different groups starting to come in, however there is still a lot of room for improve with the cluster, as a there are still many same-coloured points away far away from each other. The centroid needs reviewing.

The subpar clusters are reflected by a low silhouette score of 0.256, which in agreement with figures 11 and 12 show the clusters distinct different groupings are weak.

Getting strong clusters was a part which I struggled with the most during this assignment.

This can be seen and validated by figure 13

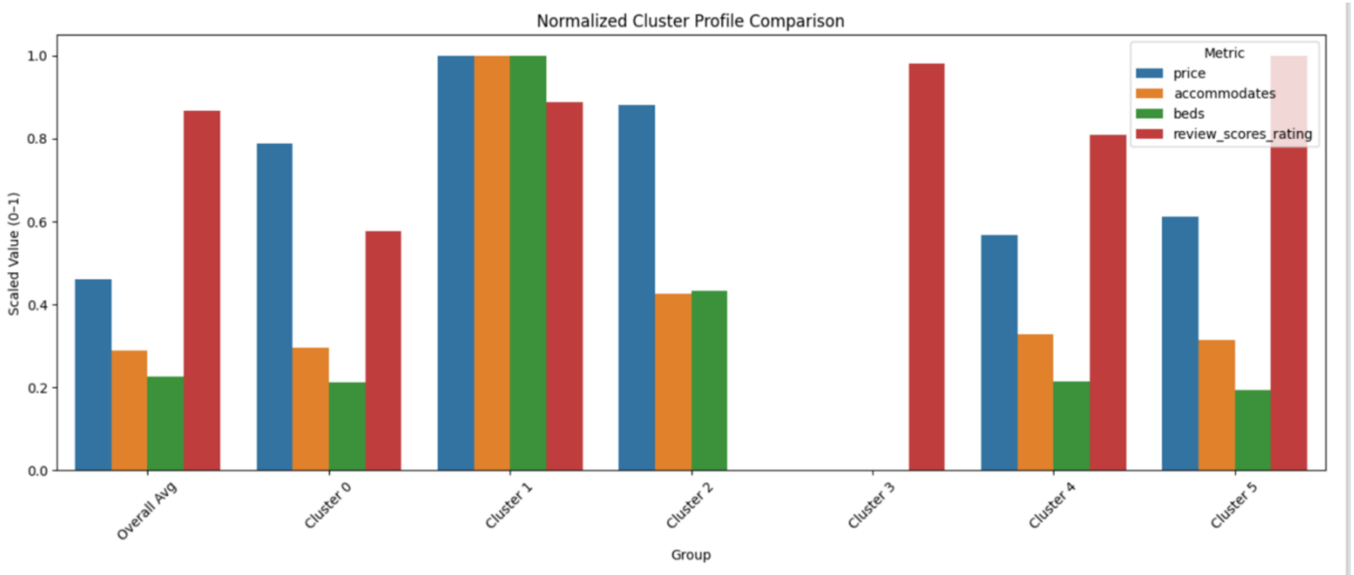


Figure 13 Cluster comparison

Figure 13 shows how the clusters performed in comparison to some key metrics. It shows rather mixed results and accuracy.

Clusters based model vs Regression model

Next, I looked at whether training separate models per cluster could outperform the regression model

This was what the separate clusters scored:

Cluster	R squared	RMSE	MAE
0	0.41	0.38	0.29
1	0.54	0.31	0.23
2	0.26	0.26	0.20
3	0.34	0.34	0.27
4	0.43	0.34	0.26
5	0.46	0.33	0.36

This is what the regression model score:

- R squared: 0.72
- RMSE: 0.35
- MAE: 0.27

The regression clearly outperformed the clusters, with a much higher r squared value and lower RMSE than all the clusters. However, the MAE was rather similar between all clusters and the regression model. This suggests although the clusters struggled to fully account for the variance in prices, they were still somewhat effective at producing reasonably accurate predictions in terms of absolute error.

Final Improved predictive model

Finally, to build a more powerful predictive model, I implemented a Random Forest regressor.

I used Featured engineering on top of the already cleaned dataset. Featured engineering allowed me to create new feature by combined two variables together. I did this for the accommodates column and bedrooms column

I then a baseline Random Forest model, using these new engineered features. Which performed quiet well: scoring 0.7844 in R squared, 0.3047 in RMSE and 0.2322 in MAE.

In attempt to optimise the model further, I used hyperparameter tuning as a means of trying to affect how the model learns and not just what it learns. This was my attempt at improving the modules accuracy and overfitting. However, finding the best combination for these hyperparameters proved rather challenging, especially with impractical run times! I used a tool from scikit learn called RandomizedSearchCV to help with this.

In the end I got my model to improve, but only slightly, the final metric scoring was 0.7857 in R squared, 0.3027 in RMSE and 0.2322 in MAE. With more time and increased computing power, there is a chance that these metrics could still improve.