BA305 London Airbnb Predictive Model

Group B4

Aditya Chopra, Dhruv Gandhi, Sahil Khatnani, David Ueda,

Benson Yu

1. Introduction

1.1 Selecting the Data Set

Our team has decided to evaluate a dataset that contains Airbnb listings in London. With this dataset, we used various characteristics of each listing in order to build models that predict the price of a listing. Selecting this dataset took much deliberation, as we parsed through multiple other datasets prior to choosing this one. Most notably, we were planning on using the New York Housing dataset to predict the price of homes across New York City. This specific dataset attracted our attention as many of us are interested in the real estate market. However, we found that there were only four useful variables to use as features after cleaning the data, which would mean our predictive models would be limited. Additionally, the dataset had many inconsistencies and missing values across critical features such as prices and property types, which could lead to unreliable predictions and skewed analyses. Upon looking for new data for the project, we eventually chose the London Airbnb dataset on Kaggle. Still being within our interest topic of real estate, and all of our group members having stayed in an Airbnb in London, we were interested to know how prices are determined based on their characteristics.

1.2 The Airbnb Dataset

Interestingly, we thought of this idea after one of our group members had actually been scammed in London by an Airbnb host, where he paid a low price within the City of London, and the Airbnb was completely different from how it was advertised. Because he did not know the fair prices of Airbnbs with these specific characteristics, he was unable to identify it as a scam. Our team wanted to create a use case for this dataset where individuals would be able to tell if they are paying a fair price for their Airbnb. One of the problems we identified is that Airbnbs do not provide a cost breakdown for how nightly prices are determined; therefore, through a predictive model, we can identify which key features impact room pricing to determine if a user is paying a fair price. On Kaggle, the data is split into two CSV files: one for listings on weekends and the other on weekdays. We wanted to have an all-inclusive dataset, so we concatenated the two files into one for our project. This new combined dataset has 9993 rows and 20 columns, with 14 features that are numerical and 6 that are categorical. Each row is an Airbnb listing in London, and the columns include comprehensive information about each listing, such as its price per

night, location, ratings, type of room(s), and other information. A data dictionary is provided in Appendix A for reference.

1.3 Project Objective

The primary aim of our study is to develop predictive models capable of estimating the price of Airbnb listings based on selected features. The "realSum" column in the dataset, which represents the price of each listing per night, serves as our target variable. By analyzing and modeling these data, we intend to uncover significant predictors of price and build a robust model that can aid stakeholders in pricing strategies and investment decisions in the Airbnb market. This analysis not only helps in price estimation but also contributes to a deeper understanding of market dynamics. Our final goal of the project is to create a predictive model that allows stakeholders to see if the price of their Airbnb is fair based on its characteristics (features).

2. Data Preprocessing

2.1 Removing Redundant Variables

After finalizing our dataset and becoming familiar with our topic, we began cleaning our data. The first variable we dropped was ‘Unnamed: 0’, as that was just an identifier for each variable. We then realized that ‘attr_index’ and ‘rest_index’ were unnormalized forms of the variables ‘attr_index_norm’ and ‘rest_index_norm’. Since our goal is to eventually create a predictive model for Airbnb prices, we needed a way to bound the variables for user input. This means that when users provide their preference for the attr_index and rest_index, they are able to provide a value between 0 and 100 since we are utilizing the normalized index. In the case that we did not use the normalized index, the scales would be different and produce meaningless results.

Other variables dropped were ‘room_shared’ and ‘room_private’. We felt like this was unnecessary to keep because we already had information about the number of bedrooms and number of people per room. Furthermore, the ‘room_type’ variable was able to differentiate between an entire apartment, private room and shared room. This means that including ‘room_shared’ and ‘room_private’ would increase the chance of multicollinearity, which may

affect our predictions. Finally, we dropped ‘lng’ and ‘lat’(longitude and latitude) because after normalizing those variables, they don’t have enough explaining power when putting them through our models.

2.2 Cleaning Data Values

Our next step was to clean the rows. We first had to ensure that there were no missing or NaN values, nor duplicates in our dataset. Surprisingly, we had no missing values nor duplicates after further examination. We then created dummy variables for the ‘room_type’ variable, and dropped the original variable and ‘room_type_Shared_room’. We did this through one hot encoding. We dropped the two variables because we did not need the original as we created dummies, and dropped the other because of the ‘n-1’ rule when creating dummies. We also created a dummy variable called ‘is_weekend’ which was required after merging the two csv files, to identify which pieces were weekend and weekday.

Next, in order to identify outliers throughout our dataset, we looked at our target variable price and plotted a box plot to visualize where outliers were present within our data. Seeing that there were some outliers present, we used the (Q +- IQR * 1.5) to remove all data that was above and below a certain threshold. This brought our total entries down from 9933 to 9462. *Figure 1* shows a boxplot before we removed the outliers and *Figure 2* shows a boxplot of after the outliers were removed.

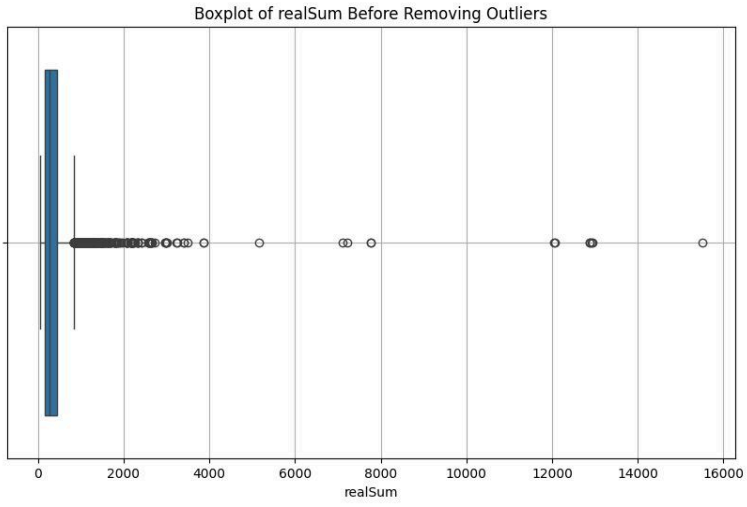

Figure 1

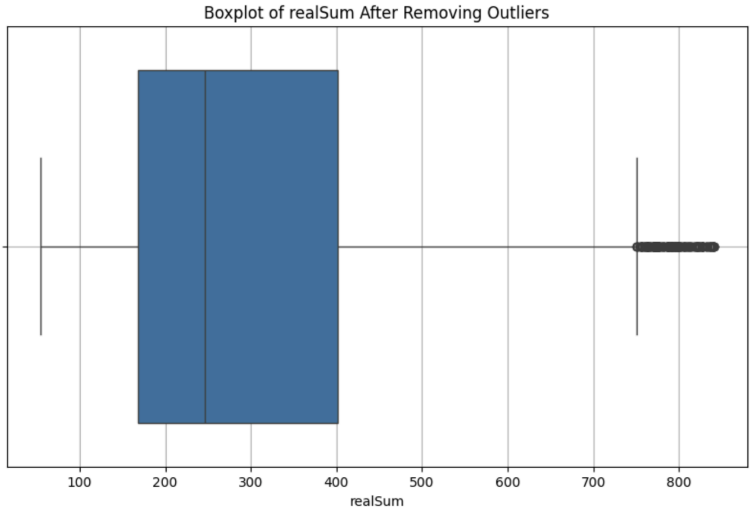

Figure 2

3. Principal Component Analysis (PCA)

3.1 Prepping the Data for PCA

Once we finished the data preprocessing, we created a correlation matrix to visualize the relationships between the 15 variables we had left.

As can be seen in Figure 3, we found that ‘dist’ had a very strong correlation with ‘metro_dist’, ‘attr_index’, and ‘rest_index’, having values of 0.7, -0.81, and -0.75. Dropping this allowed us to reduce the chance of having issues with variables that would be too highly correlated with each other (multicollinearity). After, we created another correlation matrix, as seen in Figure 4, to see the change this made to our other variables.

The next highest correlation was -0.99, which we did not see a problem with because it was a dummy variable. When looking at multicollinearity dummy variables are not continuous therefore does not prompt for concern.

We finished prepping our data for PCA by dropping ‘realSum’, as that was our target variable.

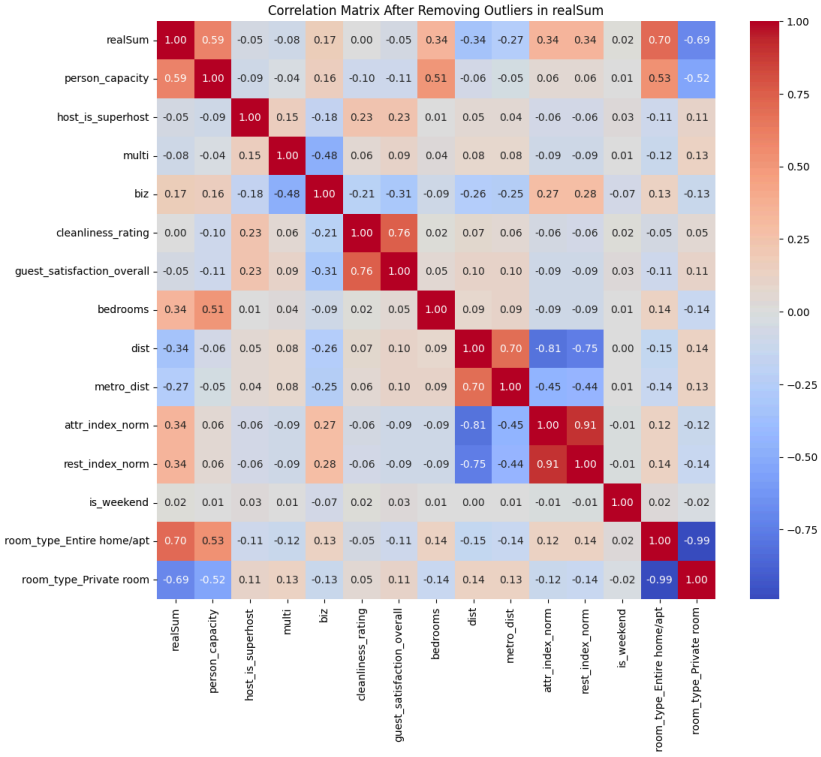

Figure 3

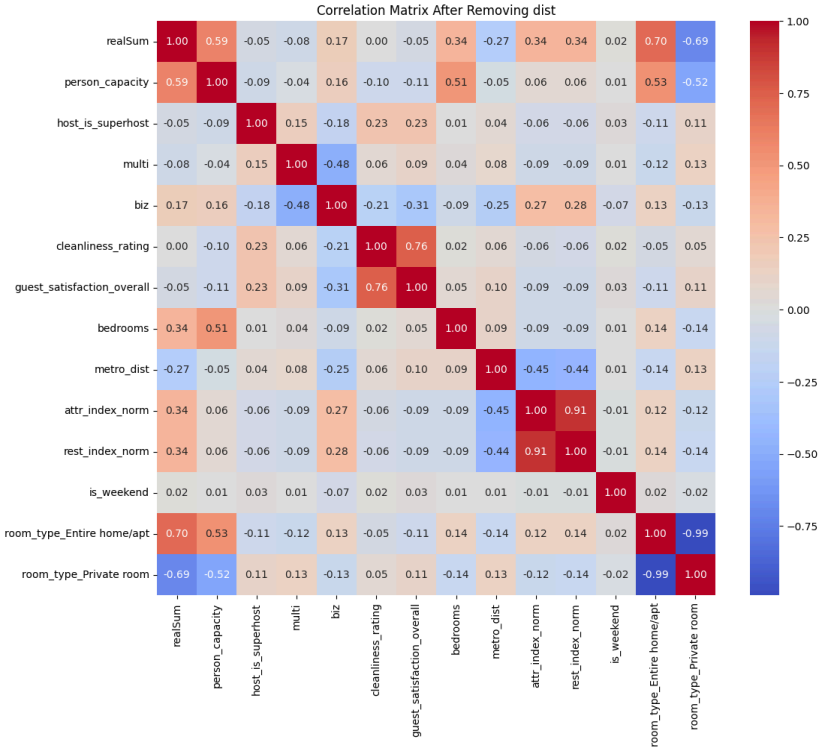

Figure 4

3.2 Running PCA

We ran PCA on the data and decided to use the latent root criterion, which tells us to only keep components with an eigenvalue greater than 1.

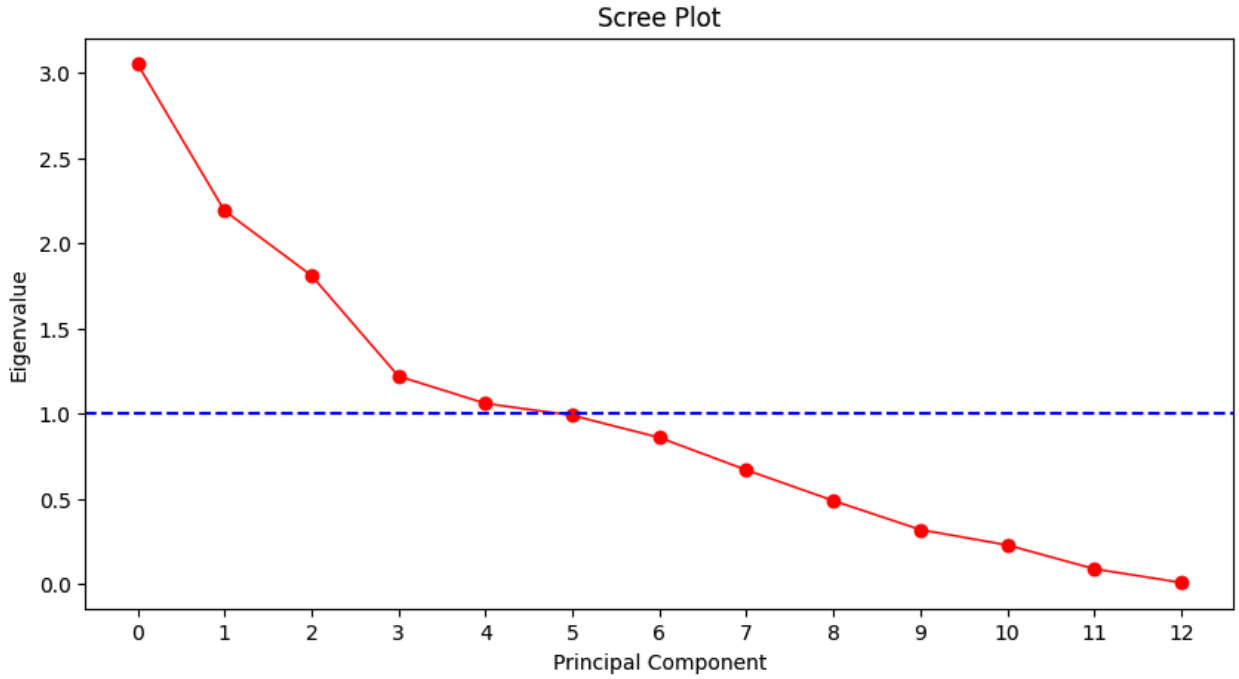

Figure 5

This left us with 5 components that explained 71.8% of the variance in our data. To further understand the components and their relationships with the original variables, we transposed them onto the original data and found that:

Component 0 - Seems to showcase homes that have space for multiple people and are used for business.

Component 1 - Showcases private rooms that have limited space but are close to good attractions and restaurants.

	0	1	2	3	4
person_capacity	0.30	-0.41	-0.02	0.12	0.34
host_is_superhost	-0.18	-0.03	-0.27	0.09	-0.01
multi	-0.20	-0.09	-0.12	0.69	-0.15
biz	0.32	0.20	0.16	-0.43	0.20
cleanliness_rating	-0.22	-0.11	-0.55	-0.33	0.04
guest_satisfaction_overall	-0.26	-0.12	-0.54	-0.26	0.08
bedrooms	0.06	-0.36	-0.03	0.21	0.67
metro_dist	-0.27	-0.26	0.21	-0.09	0.02
attr_index_norm	0.34	0.36	-0.33	0.19	0.10
rest_index_norm	0.34	0.36	-0.33	0.19	0.09
is_weekend	-0.01	-0.04	-0.05	0.10	-0.34
room_type_Entire home/apt	0.39	-0.39	-0.12	-0.06	-0.34
room_type_Private room	-0.39	0.39	0.12	0.06	0.34

Figure 6

Component 2 - Looks like this variable is focused on the relationship between a customer being satisfied with the cleanliness of their room.

Component 3 - This component looks at listings with multiple rooms that are not used for business.

Component 4 - Correlated with ‘bedrooms’, ‘person_capacity’, ‘is_weekend’, and both ‘room_types’. This component looks like it is concerned mainly with the size of the house, which is why we named it House Size.

3.3 Drawbacks of PCA with our Data

It is known that PCA works well as a data reduction technique that reduces the number of features while preserving the underlying patterns in the data. However, we decided against deploying it in our models for multiple reasons. First, the latent root test told us to use 5 components, but those 5 only explained 71.8% variance of the data. We thought that this was not a high enough representation to push us to use these components. Additionally, we wanted to investigate which features would affect the price individually, so the combination of features using PCA would inhibit us from interpreting each variable.

4. Models Trained and Tested

4.1 Naive Rule

As a baseline to compare our models to, we implemented the naive rule, where the predicted price of a listing is just the average price of all listings. We chose to use the average because it offered the lowest RMSE when compared to median and mode. The result of this was a train and test RMSE of approximately 169.7.

4.2 Linear Regression

The first model we trained was a linear regression model. Linear regression fits a linear model with certain coefficients to minimize the sum of square residuals between the observed and predicted data. We chose to try this model first as it is easy to understand and is very efficient.

Additionally, we wanted to determine whether variables had a linear relationship with the listing price or not.

The results of this model were a training RMSE of approximately 97.83 and a test RMSE of approximately 99.24. Based on these results, it is apparent that this model did not overfit as the train and test RMSE were very close together. However, these RMSE values still seemed high, so we wanted to try more models.

4.3 Random Forest

We also trained a random forest model. These models are ensembles of decision trees which means that they operate by building multiple decision trees, each trained on a random subset of the dataset with a random subset of features at each node. The predictions of each tree are then combined for an improved prediction. Additionally, random forests can capture non-linear relationships and can manage both numerical and categorical data, making it well-suited for this dataset.

Our random forest model had a training RMSE of 80.38 and a testing RMSE of 88.40. These results are better than those of the linear regression model as both RMSE values were lower for random forest. This indicates that the random forest model fits our data better than linear regression, and is therefore better at predicting the prices of listings.

4.4 KNN

K-nearest neighbors (KNN) was the third model that we trained and tested. This model finds other instances that are closest to a certain data point using euclidean distance. The desired number of neighbors (K) can then be specified by the model builder. The KNN regression model then takes the average of the values of these neighbors in order to calculate a prediction for the data point of interest. In order to determine the optimal number of neighbors to use in the model, we used the elbow method which plots how our desired accuracy metric (RMSE) changes as the number of neighbors in the model increases. As can be seen on the plot below, the point where the RMSE begins to plateau is when the number of neighbors is 5. Therefore, this is the number

of neighbors we chose to use in our model. The results of this was a train RMSE of 79.63 and a test RMSE of 95.63.

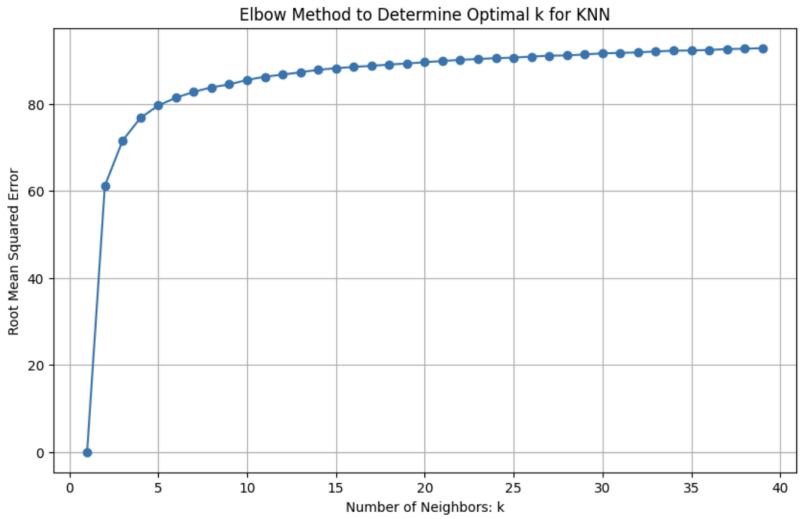

Figure 7

4.5 XGBoost

After trying all of the aforementioned models, we were determined to find another model that could have lower error values and thus provide better predictions. As most of our group members are data science majors, we have had experience with boosting methods such as those used by the XGBoost model. This model bootstraps multiple different samples and builds a single decision tree model on each sample. Then, the decision tree's output is used for the next model to try and correct mistakes from the previous one. This process is continuously repeated until there is one output. XGBoost models are incredibly popular in the industry as it uses regularization to reduce overfitting, but also uses gradient boosting to minimize bias and underfitting. Therefore, we wanted to train this model with the expectation that it would perform the best.

We were correct in our assumption as the XGBoost model had the lowest train and test RMSE with values of 71.96 and 84.28, respectively. Figure 8 depicts the 10 most important features used in the XGBoost model to make predictions of the price. As shown, the distance of the listing to the metro is most important, with the attractiveness and restaurant index being second and third most important. This will be useful information when analyzing the results of our price predictor.

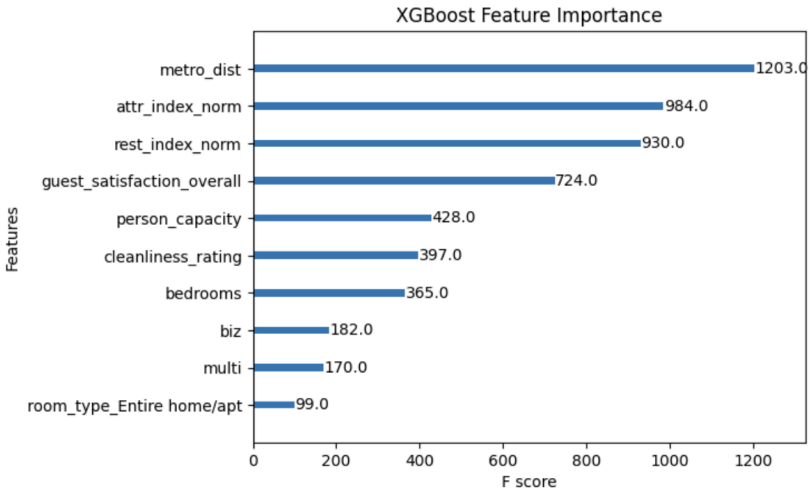

Figure 8

5. Our Predictive Model

5.1 How our Predictive Model works

As aforementioned, since the XGBoost model fits our data the best, we decided to use it for our price predictor. In our predictor, the user will be prompted to enter values for the features of the listing. We built a function in Python that will accept these inputs and enter them into our XGBoost model. The function then returns the prediction for the price of the listing from the XGBoost model.

A predictor like this one can have many use cases. First, Airbnb users can enter their desired value for each feature and obtain a prediction for what they should expect to pay per night. Additionally, Airbnb hosts can enter in the information about their listing and see what price they should list their home at. Finally, a third possible use of this could be for users to detect scams on Airbnb. If the price of a listing deviates from the model's prediction by an amount much larger than the model's RMSE of 84, then it is possible that the listing is not exactly as advertised.

6. Areas for Improvement

6.1 Areas for Improvement

Our XGBoost model allows us to predict the price of a listing according to the features provided from our dataset. Though we were quite confident in the predictive power of our model, we

found that there may be some key features that are missing from the dataset which could affect the price drastically. These missing features include the number of bathrooms in the listing or the dates and seasonality of the stay. In addition, the demand of a listing plays a role in the price it displays, so future versions of our model will improve when these variables are considered for the prediction. In addition, our model can only predict the price of a listing when every feature is known about the listing. Therefore, if a user who does not care about the cleanliness is trying to find a reasonable price for an Airbnb listing, our model will still require a value for that feature. This model would improve in usability if it is able to predict the price when the user only has to input features that they want or know.

In addition to these improvements, we hope to build upon this model for more impactful use cases, such as for scam prevention. If our dataset were larger and had more information such as those mentioned above, we could possibly detect outliers in Airbnb listing prices by comparing the listings' features to the price predictor model.

7. Appendix A

Column name	Description
realSum	The total price of the Airbnb listing. (Numeric)
room_type	The type of room being offered (e.g. private, shared, etc.). (Categorical)
room_shared	Whether the room is shared or not. (Boolean)
room_private	Whether the room is private or not. (Boolean)
person_capacity	The maximum number of people that can stay in the room. (Numeric)
host_is_superhost	Whether the host is a superhost or not. (Boolean)
multi	Whether the listing is for multiple rooms or not. (Boolean)
biz	Whether the listing is for business purposes or not. (Boolean)
cleanliness_rating	The cleanliness rating of the listing. (Numeric)
guest_satisfaction_overall	The overall guest satisfaction rating of the listing. (Numeric)
bedrooms	The number of bedrooms in the listing. (Numeric)
dist	The distance from the city centre. (Numeric)
metro_dist	The distance from the nearest metro station. (Numeric)
lng	The longitude of the listing. (Numeric)
lat	The latitude of the listing. (Numeric)

Figure 9: Data Dictionary for Airbnb Dataset

8. Works Cited

- https://www.kaggle.com/datasets/thedevastator/airbnb-prices-in-european-cities/data?select=london_weekends.csv
- https://www.airbnb.com/s/London--United-Kingdom/homes?tab_id=home_tab&refinement_paths%5B%5D=%2Fhomes&flexible_trip_lengths%5B%5D=one_week&monthly_start_date=2024-05-01&monthly_length=3&monthly_end_date=2024-08-01&price_filter_input_type=0&channel=EXPLORE&query=London%2C%20United%20Kingdom&place_id=ChIJdd4hrwug2EcRmSrV3Vo6llI&date_picker_type=calendar&adults=4&source=structured_search_input_header&search_type=user_map_move&price_filter_num_nights=5&ne_lat=51.502107110819544&ne_lng=-0.1590110114928791&sw_lat=51.491134177669174&sw_lng=-0.20434334169587487&zoom=15.343186249937508&zoom_level=15.343186249937508&search_by_map=true&location_search=MIN_MAP_BOUNDS&price_min=291&min_bedrooms=2&min_beds=4&min_bathrooms=2&l2_property_type_ids%5B%5D=3