
The University of Chicago Booth School of Business

# ANALYZING KEY FACTORS INFLUENCING AIRBNB PROPERTY PRICES USING ADVANCED DATA TECHNIQUES

GROUP 11: YU-TING WENG, MENGDI HAO, ELENA LI, MINJI PARK, SARAH LEE

*We pledge our honor that we have not violated the Honor Code during the preparation of this assignment.*

## Contents

## 1 Executive Summary

In this paper, we have performed an extensive analysis of Airbnb data using various statistical and machine-learning techniques to understand the key factors influencing property prices.

Before modeling, we conducted missing value imputation and exploratory data analysis (EDA) to ensure data integrity and gain insights into the dataset. We employed median imputation for numerical variables and specific replacements for categorical variables. Through EDA, we examined the distribution of log-transformed prices and their relationships with numerical variables such as accommodations, bathrooms, and bedrooms. Additionally, we analyzed the impact of categorical variables like city, neighborhood, property type, and room type on prices. This analysis revealed significant correlations, indicating that properties with more accommodations, bathrooms, and bedrooms generally have higher prices. Moreover, the geographical location and type of property also play crucial roles in determining the price.

To further enhance our understanding, we used principal component analysis (PCA), K-means clustering, Latent Dirichlet Allocation (LDA) for topic modeling, cross-validated LASSO and Ridge regressions, CART, random forest, gradient boosting models, and examined treatment effects. Our specific goals were to predict and understand the drivers of property prices and provide actionable insights for Airbnb hosts. The key findings of our analysis are summarized below:

- **Principal Component Analysis (PCA)**
  - Employing methods like Akaike Information Criterion corrected (AICc), Generalized Linear Models (GLM), and Lasso regression aided in determining the optimal number of principal components.
  - In the numeric columns scenario, all 12 numeric variables were deemed significant by both Lasso and GLM methods.
  - With all columns considered (39 variables), Lasso identified 10 significant principal components, while GLM recognized 34, showcasing the varying degrees of relevance attributed to different principal components.

- **K-Means Clustering**
  - Three key methods—the Elbow Method, the Silhouette Method, and Information Criteria (AIC/BIC)—were employed to determine the optimal number of clusters.
  - The Silhouette Method recommended K=2. For K=2, significant differences were observed in various accommodation features, such as parking availability, family-friendliness, and review scores.
  - AIC/BIC identified K=10 as optimal. The 10-cluster solution exhibited considerable overlap, indicating less distinctiveness among clusters.

- **Topic Modeling**
  - Topic modeling and sentiment analysis on Airbnb listing descriptions and titles reveal that certain textual elements can help predict listing prices, with titles providing clearer and more actionable insights compared to reviews.
  - Clustering analysis using AIC and BIC identified the optimal number of clusters for both listing titles and descriptions, with titles showing more coherent themes related to amenities and location, while reviews were more dispersed and less effective in topic differentiation.
  - Sentiment analysis indicated that listings with more positive sentiment scores tend to have higher prices, but the overlap in sentiment distributions between high and low-price listings suggests that other factors also play significant roles in determining Airbnb pricing.
- **Regression Analysis: CV.LASSO vs Ridge**
  - The general analysis reveals that both LASSO and Ridge regression models demonstrate comparable performance, as evidenced by their similar in-sample and out-of-sample R-squared values.
  - Positive coefficients for accommodation-related variables, such as the number of `accommodates`, `bathrooms`, and `bedrooms` suggest that larger properties tend to command higher prices, while negative coefficients for variables, such as the number of reviews and days since the last review, imply that older or less-reviewed listings, may be priced lower.
  - The Naive LASSO Model demonstrates robust predictive performance, with approximately 69.7% of the variance between predictors and log price from the training dataset.
- **Ensemble Learning**
  - A single decision tree only explains 55% of variance but with minimal over-fitting.
  - As an ensemble method, a random forest model significantly improves predictive accuracy (OOS $R^2$ of 71%) but suffers from over-fitting.
  - A more advanced XGBoost model balances high predictive accuracy (OOS $R^2$ of 72%) with reduced over-fitting.
  - Across all tree models, key features influencing Airbnb prices are `room_type`, `accommodates`, `bedrooms`, `longitude`, `latitude`, and `bathrooms`.

By integrating these techniques, we have gained a comprehensive understanding of the factors influencing Airbnb property prices. These insights can help hosts optimize their listings and pricing strategies to attract more guests and maximize revenue.

## 2 Introduction

Airbnb, established in 2008, has evolved into one of the largest and most influential online accommodation booking platforms in the world. With operations spanning over 220 countries, Airbnb has transformed the way people travel and experience new destinations. As a global enterprise, its user base continues to expand, a trend that Morgan Stanley predicts will persist in the coming years. One of the pivotal factors contributing to Airbnb's remarkable success is its competitive pricing strategy, which plays a critical role in balancing the intricate dynamics of supply and demand within the marketplace.

Unlike traditional hotels that adhere to standardized pricing models, Airbnb hosts set their prices based on personal experience and market conditions. This flexibility, while advantageous, presents a significant challenge for new hosts who may struggle to determine a reasonable and competitive price for their listings. From a consumer perspective, although potential guests can compare prices across various listings, having a reference or benchmark price can be invaluable in assessing the fairness and attractiveness of a given rate.

This report is structured to provide a comprehensive analysis of the factors influencing Airbnb pricing. The subsequent section offers a detailed overview of the dataset utilized in this research, highlighting its scope and key variables. Following this, we delve into an exploration of various advanced modeling techniques, including Principal Component Analysis (PCA), K-means clustering, topic modeling, LASSO regressions, and decision tree models. Each of these methodologies is employed to meticulously examine the relationship between feature variables and the target variable, namely the listing price, offering insights into the determinants of Airbnb property valuations.

## 3 Dataset Summary

This extensive dataset offers a detailed glimpse into the dynamic world of Airbnb accommodations spanning diverse locations worldwide. From cozy apartments in bustling urban centers to serene retreats nestled in picturesque countryside, this collection provides valuable insights into the various types of lodging options available on the Airbnb platform. With comprehensive information on property features, pricing, reviews, and host characteristics, researchers and enthusiasts alike can delve into the intricate ecosystem of shared lodging, uncovering trends, patterns, and preferences that shape the evolving landscape of modern hospitality. Whether analyzing market trends, assessing the impact of tourism on local economies, or simply satisfying a curiosity about global travel trends, this dataset serves as a rich resource for exploring the multifaceted realm of Airbnb accommodations.

This dataset, downloaded from Kaggle, contains 74,111 rows and 29 columns, providing a comprehensive overview of various Airbnb listings.

The dataset includes the following variables, categorized by type:

**Numeric Variables**

- **log_price**: Log-transformed price of the listing and our target variable
- **accommodates**: Number of guests the property can accommodate
- **bathrooms**: Number of bathrooms in the property
- **host_response_rate**: Response rate of the host to inquiries
- **latitude**: Latitude coordinate of the property
- **longitude**: Longitude coordinate of the property
- **number_of_reviews**: Total number of reviews received by the listing
- **review_scores_rating**: Average rating score from reviews
- **bedrooms**: Number of bedrooms in the property
- **beds**: Number of beds in the property

**Categorical Variables**

- **id**: Unique identifier for each listing
- **property_type**: Type of property (e.g., Apartment, House)
- **room_type**: Type of room being offered (e.g., Entire home/apt, Private room)
- **bed_type**: Type of bed provided (e.g., Real Bed)
- **cancellation_policy**: Cancellation policy of the listing (e.g., strict, flexible)
- **cleaning_fee**: Indicator of whether a cleaning fee is charged
- **city**: City where the property is located
- **host_has_profile_pic**: Indicator of whether the host has a profile picture
- **host_identity_verified**: Indicator of whether the host's identity is verified
- **instant_bookable**: Indicator of whether the listing is available for instant booking
- **neighbourhood**: Neighbourhood where the property is located
- **thumbnail_url**: URL of the thumbnail image for the listing
- **zipcode**: Zip code of the property's location

**Text Variables**

- **amenities**: List of amenities available in the property
- **description**: Description of the property
- **name**: Name of the listing

**Date Variables**

- **first_review**: Date of the first review received by the listing
- **host_since**: Date since the host has been active on Airbnb
- **last_review**: Date of the most recent review received by the listing

The dataset summary is illustrated in Table 1, which provides detailed statistics for the numerical attributes.

TABLE 1. Summary statistics for numerical attributes

| Statistic | id | log price | accommodates | bathrooms | latitude |
|---|---|---|---|---|---|
| Min. | 344 | 0.000 | 1.000 | 0.000 | 33.34 |
| 1st Qu. | 6261964 | 4.317 | 2.000 | 1.000 | 34.13 |
| Median | 12254147 | 4.710 | 2.000 | 1.000 | 40.66 |
| Mean | 11266617 | 4.782 | 3.155 | 1.235 | 38.45 |
| 3rd Qu. | 16402260 | 5.220 | 4.000 | 1.000 | 40.75 |
| Max. | 21230903 | 7.600 | 16.000 | 8.000 | 42.39 |
| NA's | 0 | 0 | 0 | 0 | 0 |

| Statistic | longitude | number of reviews | review scores rating | bedrooms | beds |
|---|---|---|---|---|---|
| Min. | -122.51 | 0.0 | 20.00 | 0.000 | 0.000 |
| 1st Qu. | -118.34 | 1.0 | 92.00 | 1.000 | 1.000 |
| Median | -77.00 | 6.0 | 96.00 | 1.000 | 1.000 |
| Mean | 92.40 | 20.9 | 94.07 | 1.266 | 1.711 |
| 3rd Qu. | -73.95 | 23.0 | 100.00 | 1.000 | 2.000 |
| Max. | -70.99 | 605.0 | 100.00 | 10.00 | 18.00 |
| NA's | 0 | 16722 | 91 | 131 | 0 |

### 3.1 Missing Value Imputation for Visualization

TABLE 2. Variable Missing Count

| # | Variable | Missing Count |
|---|---|---|
| 1 | host_response_rate | 18299 |
| 2 | review_scores_rating | 16722 |
| 3 | first_review | 15864 |
| 4 | last_review | 15827 |
| 5 | thumbnail_url | 8216 |
| 6 | neighbourhood | 6872 |
| 7 | zipcode | 966 |
| 8 | bathrooms | 200 |
| 9 | host_has_profile_pic | 188 |
| 10 | host_identity_verified | 188 |
| 11 | host_since | 188 |
| 12 | beds | 131 |
| 13 | bedrooms | 91 |
| 14 | id | 0 |
| 15 | log_price | 0 |
| 16 | property_type | 0 |
| 17 | room_type | 0 |
| 18 | amenities | 0 |
| 19 | accommodates | 0 |
| 20 | bed_type | 0 |
| 21 | cancellation_policy | 0 |
| 22 | cleaning_fee | 0 |
| 23 | city | 0 |
| 24 | description | 0 |
| 25 | instant_bookable | 0 |
| 26 | latitude | 0 |
| 27 | longitude | 0 |
| 28 | name | 0 |
| 29 | number_of_reviews | 0 |

Table 2 shows the count of missing values for each variable in the dataset. Missing value imputation was performed separately for numerical and categorical variables.

#### 3.1.1 Missing Value Imputation Summary

This section summarizes the approach to handling missing values in numeric and categorical columns, ensuring data integrity for subsequent analysis.

(1) **Convert `host_response_rate` to Numeric**:
   - The percentage sign (%) was removed from the `host_response_rate` column, and the values were converted to numeric.

(2) **Replace Missing Values in Numeric Columns with the Median**:
   - For the columns `bathrooms`, `bedrooms`, `beds`, `review_scores_rating`, and `host_response_rate`, missing values were replaced with the median of their respective columns.

(3) **Replace Missing Values in Categorical Columns with Specific Values**:
- For the column `host_has_profile_pic`, missing values and empty strings were replaced with `'f'`.
- For the column `host_identity_verified`, missing values and empty strings were replaced with `'f'`.
- For the columns `first_review`, `host_since`, `last_review`, and `neighbourhood`, missing values and empty strings were replaced with `'Unknown'`.
- For the column `thumbnail_url`, missing values and empty strings were replaced with `'No URL'`.
- For the column `zipcode`, missing values and empty strings were replaced with `'Unknown'`.

### 3.1.2 Updated Summary Statistics

After updating the missing values, the updated summary statistics are presented in Table 3.

TABLE 3. Updated Summary statistics for numerical attributes

| Statistic | id | log price | accommodates | bathrooms | latitude |
|---|---|---|---|---|---|
| Min. | 344 | 0.000 | 1.000 | 0.000 | 33.34 |
| 1st Qu. | 6261964 | 4.317 | 2.000 | 1.000 | 34.13 |
| Median | 12254147 | 4.710 | 2.000 | 1.000 | 40.66 |
| Mean | 11266617 | 4.782 | 3.155 | 1.235 | 38.45 |
| 3rd Qu. | 16402260 | 5.220 | 4.000 | 1.000 | 40.75 |
| Max. | 21230903 | 7.600 | 16.000 | 8.000 | 42.39 |
| Statistic | longitude | number of reviews | review scores rating | bedrooms | beds |
| Min. | -122.51 | 0.0 | 20.00 | 0.000 | 0.000 |
| 1st Qu. | -118.34 | 1.0 | 93.00 | 1.000 | 1.000 |
| Median | -77.00 | 6.0 | 96.00 | 1.000 | 1.000 |
| Mean | 92.40 | 20.9 | 94.50 | 1.265 | 1.711 |
| 3rd Qu. | -73.95 | 23.0 | 99.00 | 1.000 | 2.000 |
| Max. | -70.99 | 605.0 | 100.00 | 10.00 | 18.00 |

## 3.2 Variable Transformation for Modeling

In this section, we describe the steps taken to transform variables for our modeling efforts. Building on the missing value imputation performed in Section 3.1, we made several key transformations to prepare the data for analysis:

- **Dummy Variable Creation for Amenities**: The amenities variable, which contains a list of amenities available in each property, was transformed into dummy variables. Each unique amenity was converted into a separate binary variable indicating its presence or absence in the property.
- **Date Transformations**: The date variables `first_review`, `last_review`, and `host_since` were converted into numerical variables representing the number of days since the respective dates. This transformation was achieved by calculating the difference between the current date and the specific review or host date.

- **Binary Indicator for Thumbnail URL**: The `thumbnail_url` variable was transformed into a binary categorical variable. Instead of using the actual URL, we created a binary indicator to denote whether a thumbnail image was available or not.

# 4 Exploratory Data Analysis (EDA) of Y Variable

In this section, we perform an exploratory data analysis (EDA) on the target variable, log price, to understand its distribution and characteristics.

## 4.1 Log Price Distribution

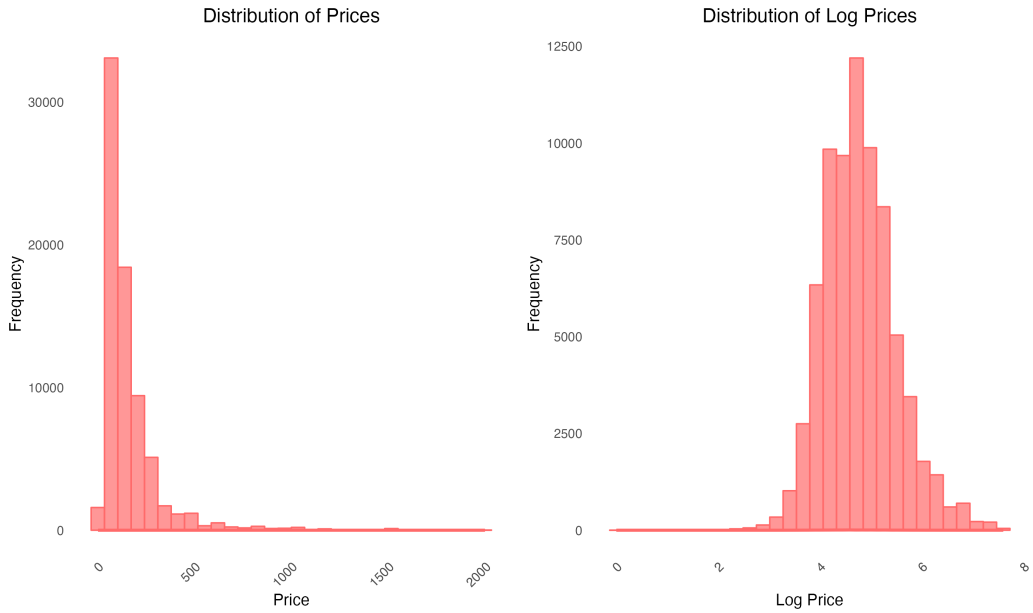



FIGURE 1. Distribution of the Price and Log Price

Figure 1 shows the distribution of prices and log-transformed prices. The original price distribution is highly skewed with a long tail, while the log-transformed prices follow a more symmetric, Gaussian-like distribution.

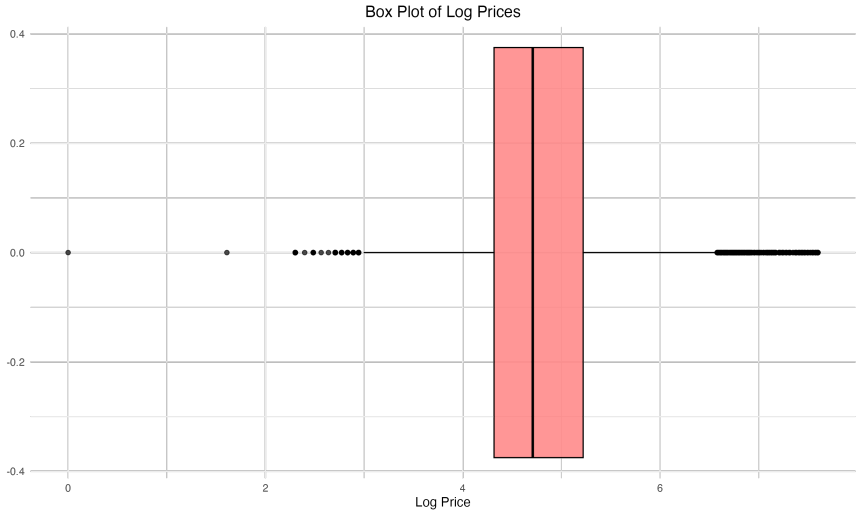



FIGURE 2. Box Plot of Log Prices

Figure 2 shows the box plot of log-transformed prices. The median log price is approximately 4.71, with an interquartile range (IQR) from about 3.9 to 5.1. The plot highlights the presence of several outliers on both ends, indicating variability in the log prices.

## 4.2 Relationship between Log Price and Numerical Variables

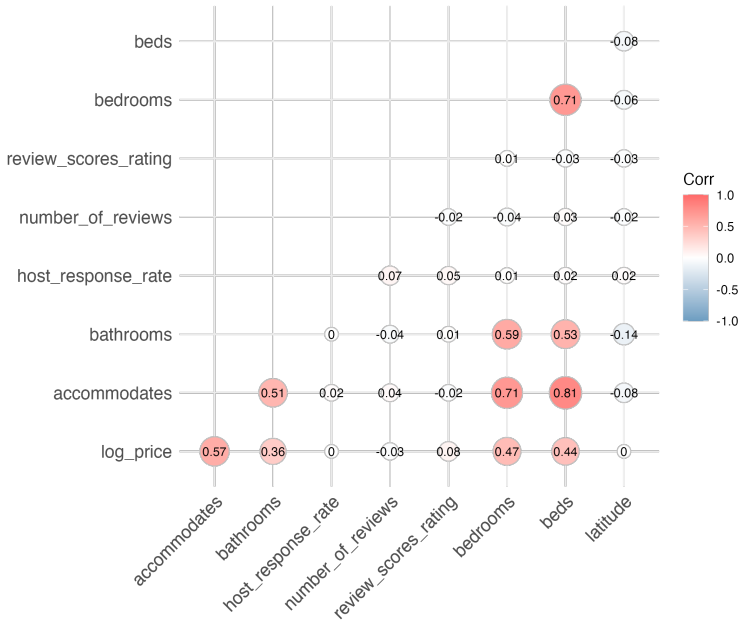



FIGURE 3. Correlation Matrix of Numerical Property Features with Log Price

The correlation matrix in Figure 3 highlights the relationships between log price and various numerical features. Log price has the highest positive correlations with accommodates (0.57), bathrooms (0.47), and bedrooms (0.44). These correlations indicate that the log-transformed price also tends to increase as the number of accommodations, bathrooms, and bedrooms increases.

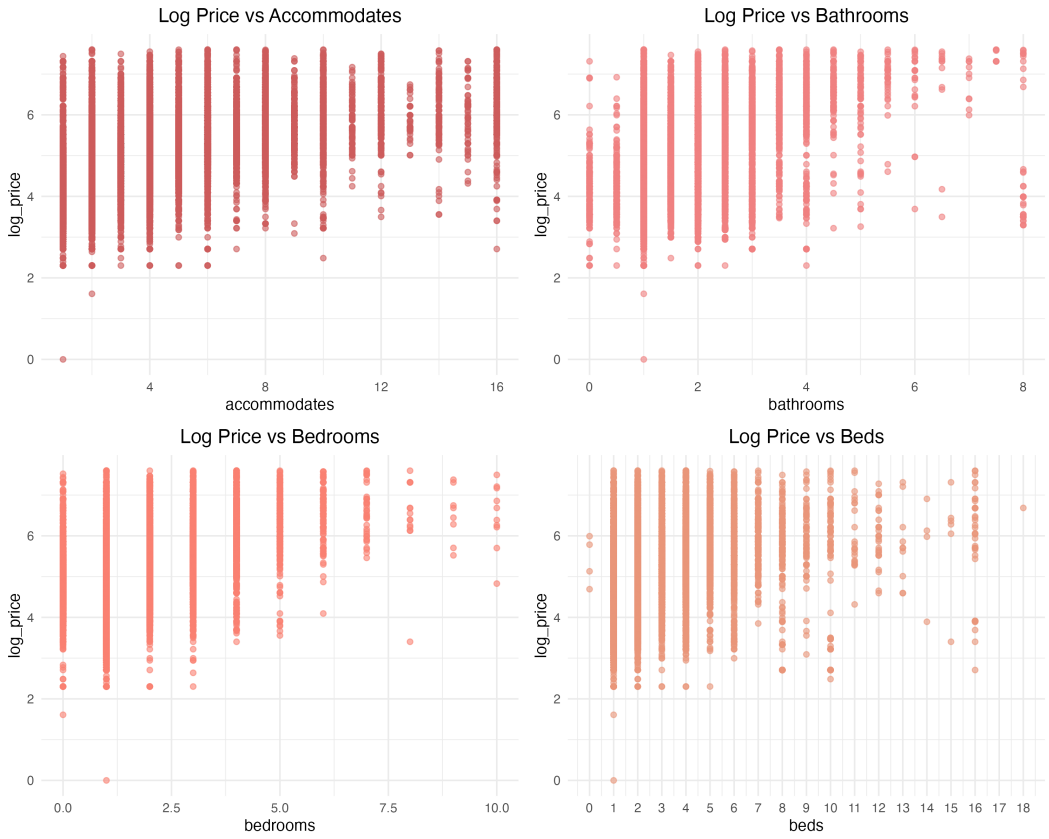



FIGURE 4. Scatter Plots of Log Price vs Property Features

Figure 4 shows the scatter plots of log price versus various property features.

- **Log Price vs. Accommodates:** As the number of accommodates increases, the log price generally increases, indicating a positive relationship between the number of guests a property can accommodate and its price.
- **Log Price vs. Bathrooms:** There is a positive correlation between the number of bathrooms and log price, suggesting that properties with more bathrooms tend to have higher prices.
- **Log Price vs. Bedrooms:** Similar to accommodates and bathrooms, there is a positive relationship between the number of bedrooms and log price, with more bedrooms corresponding to higher prices.
- **Log Price vs. Beds:** The plot indicates that as the number of beds increases, the log price also tends to increase, although the relationship appears to be less strong compared to the other features.

### 4.3 Relationship between Log Price and Categorical Variables

- **Log Price vs. City**

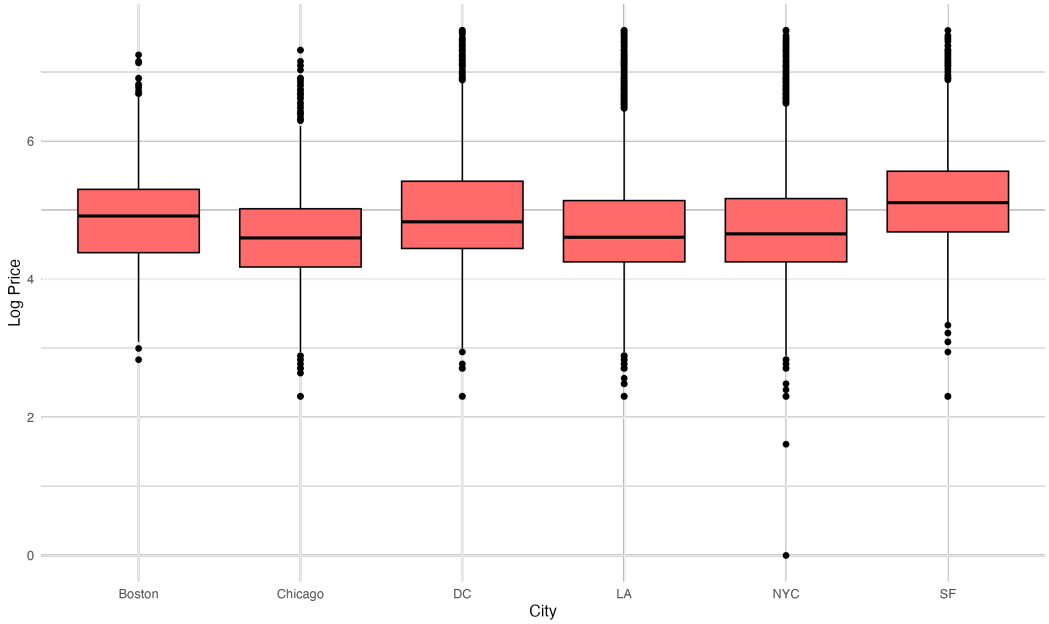



FIGURE 5. Log Price Distribution Across Different Cities

As shown in Figure 5, the distribution of log prices varies across different cities. San Francisco (SF) has the highest median log price, while Chicago has the lowest. The variability of log prices is relatively consistent across cities, with all showing a similar range of values.

- **Log Price vs. Neighborhood in NYC**

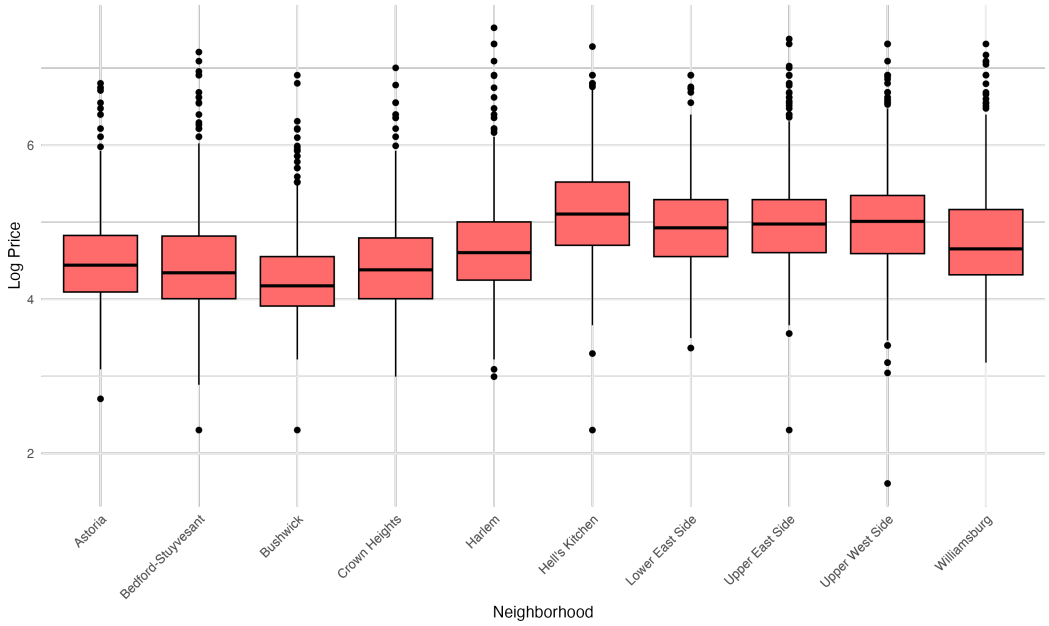



FIGURE 6. Log Price Distribution Across Top 10 Neighborhoods in NYC

Since New York City (NYC) has the highest proportion of listings among all cities, we further examined the distribution of log prices across its top 10 neighborhoods. As shown in Figure 6, the neighborhood of Hell's Kitchen has the highest median log price, while Bushwick has the lowest. The variability of log prices is relatively consistent across these neighborhoods.

- **Log Price vs. Property Type**

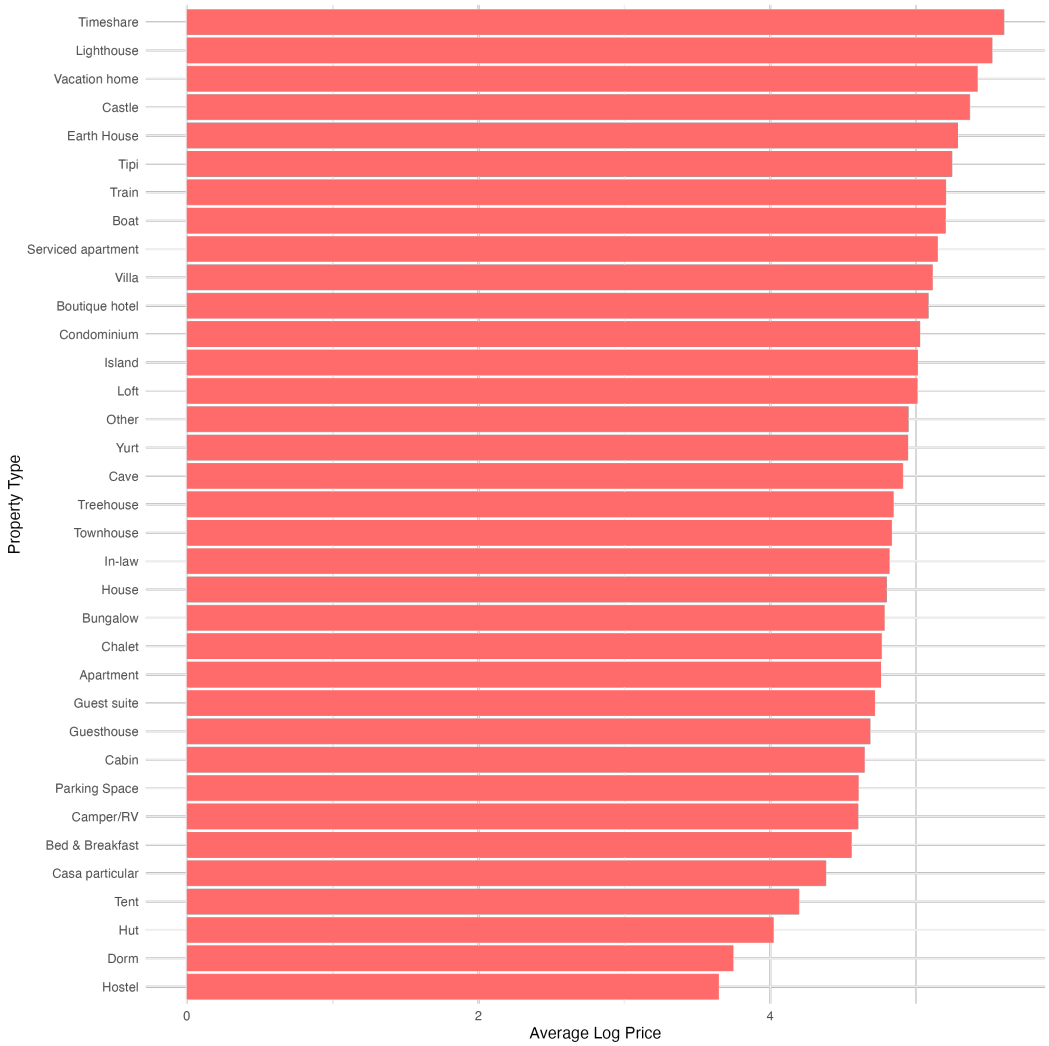



FIGURE 7. Average Log Prices for Different Property Types

Figure 7 shows that the property types with the highest average log prices include Timeshare, Lighthouse, and Vacation home. Timeshares are often located in prime vacation spots and offer shared ownership, which can drive up prices. Lighthouses provide unique and historic accommodations with scenic views, making them more expensive. Vacation homes typically offer spacious, fully-equipped properties in desirable locations, contributing to their higher prices.

- **Log Price vs. Room Type**

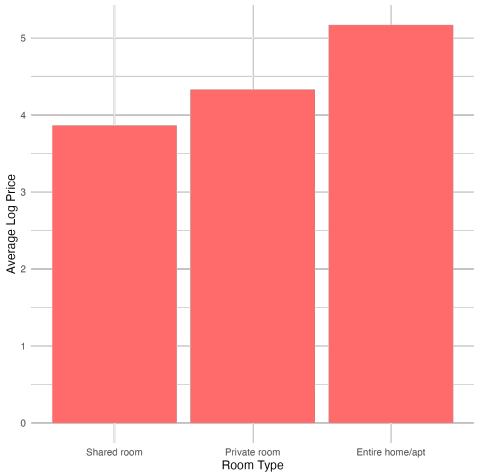



FIGURE 8. Average Log Prices for Different Room Types

Figure 8 shows that the room type with the highest average log price is the Entire home/apt, followed by Private room and Shared room. Entire homes/apartments provide more space and privacy, which generally leads to higher prices. Private rooms balance privacy and cost, making them less expensive than entire homes but more costly than shared rooms. Shared rooms, offering the least privacy, are the most economical option.

- **Log Price vs. Host Characteristics**

Figure 9 shows the correlation matrix of host characteristics. The relationships between log price and the host characteristics are not strong. However, the correlation between `host_identity_verified` and `host_duration_years` is the most significant, indicating that longer active hosts are more likely to have their identity verified.

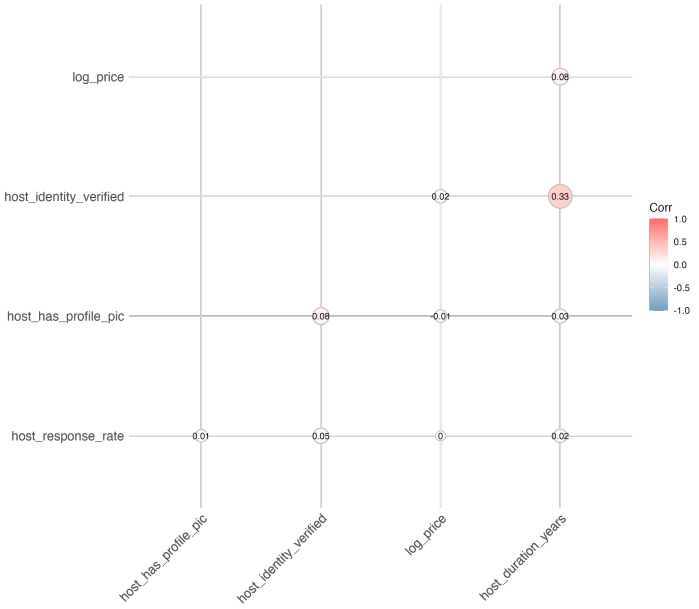



FIGURE 9. Correlation Matrix of Host Characteristics

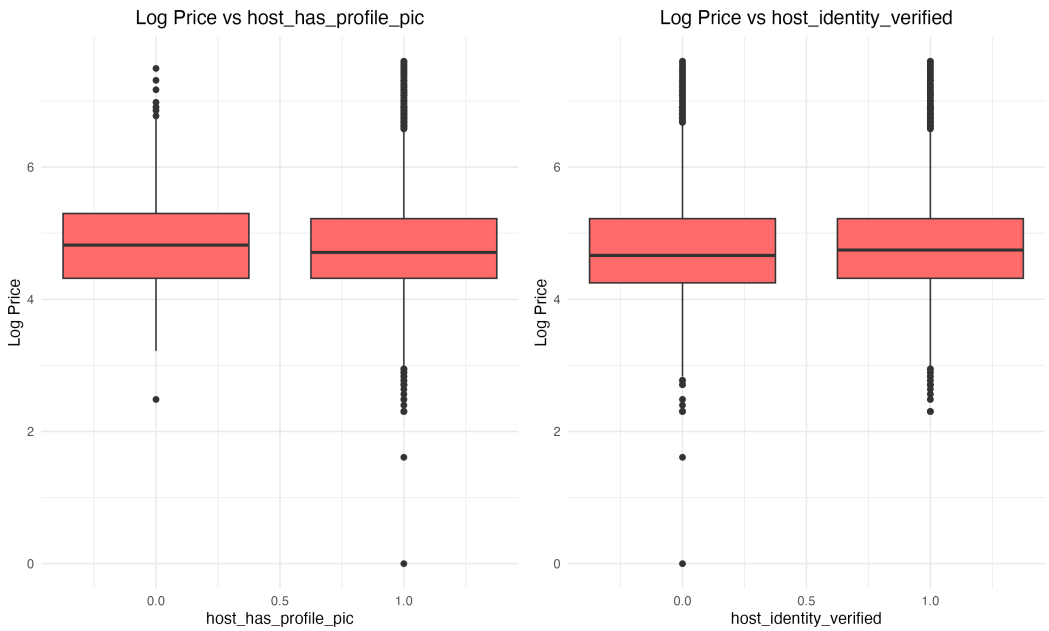



FIGURE 10. Log Price vs Host Characteristics

In Figure 10, we compare the log prices of listings based on two host characteristics: whether the host has a profile picture and whether the host's identity is verified. The box plots show that hosts with a profile picture have slightly lower log prices than those without a profile picture. Additionally, hosts with a verified identity tend to have slightly higher log prices than those without identity

verification. However, the differences are not substantial, indicating that these host characteristics do not have a significant impact on log prices.

- **Log Price vs. Amenities**

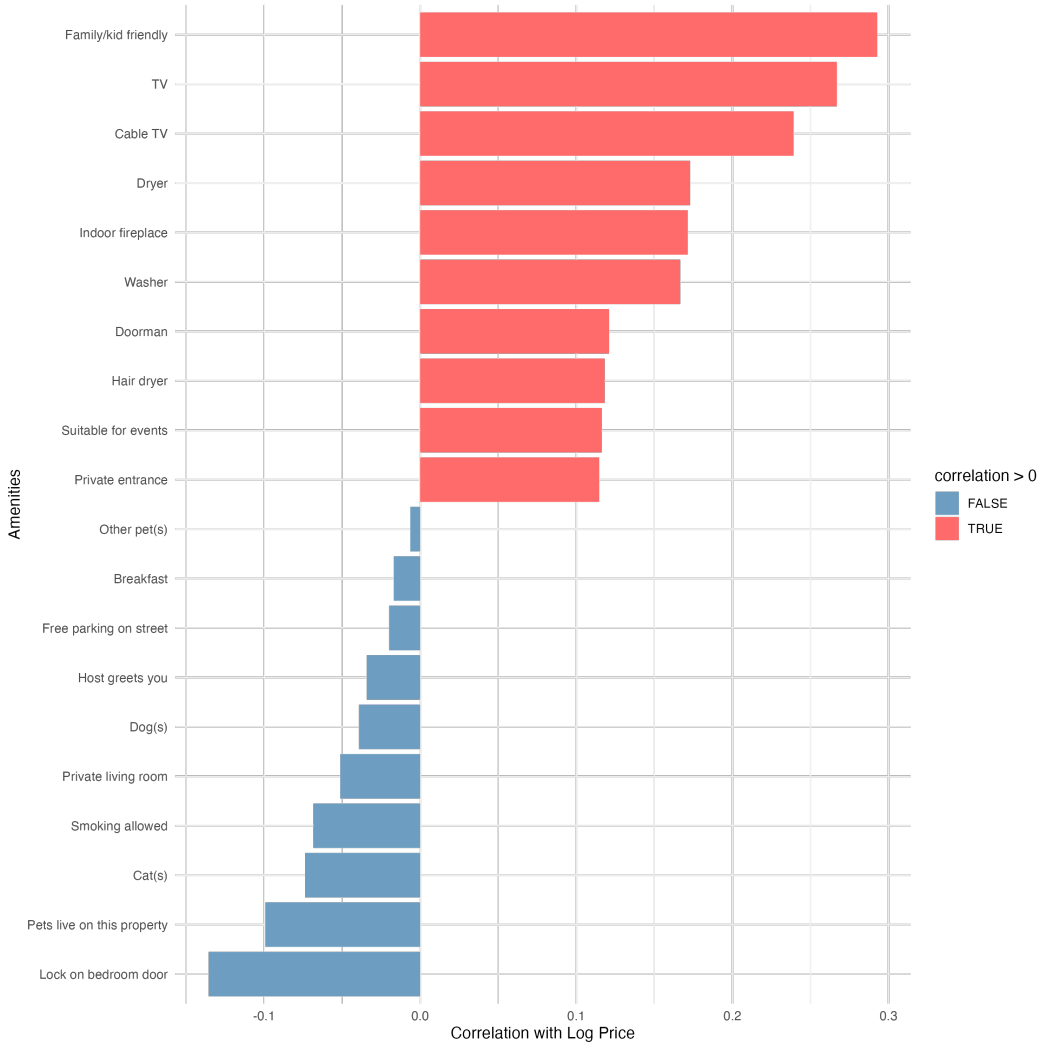



FIGURE 11. Top 10 Positive and Negative Impacts of Amenities on Log Price

In Figure 11, the top amenities that positively impact log price include family/kid friendly, TV, and cable TV, indicating that properties offering these amenities tend to command higher prices. Conversely, amenities such as "lock on bedroom door," "pets live on this property," and "cats" are associated with lower log prices, suggesting these features might be less desirable or indicative of less luxurious properties. The presence of family-oriented and entertainment amenities generally correlates with higher prices, likely because they add value to the guest

experience. On the other hand, pet amenities and additional security measures seem to correlate with lower prices, potentially due to their association with more basic or shared accommodations.

# 5 Exploratory Data Analysis (EDA) of X Variables

In this section, we conduct an exploratory data analysis (EDA) on the X variables to uncover patterns and relationships within the dataset. This includes examining categorical variables such as property type, room type, and city-wise distributions, as well as reviewing scores. Through visualizations and detailed analysis, we aim to understand the distribution and impact of these variables on the overall Airbnb listings.

## 5.1 Categorical Variables

- **Property Type**

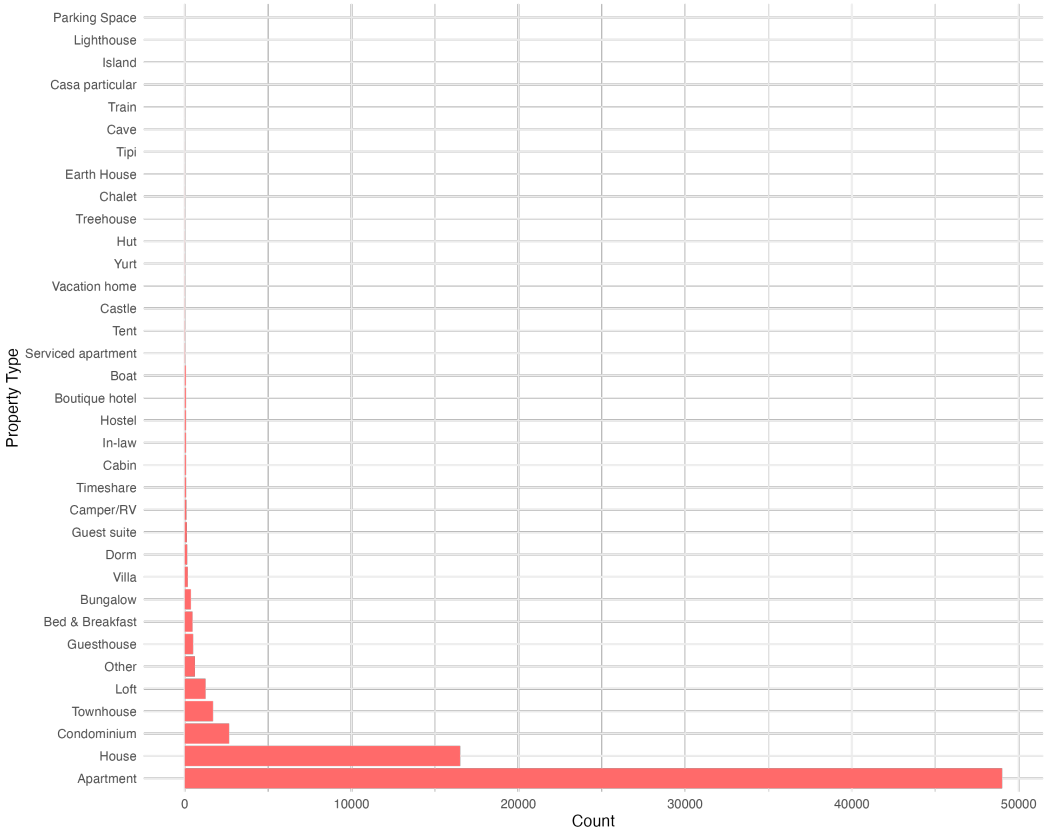



FIGURE 12. Distribution of Property Types

In Figure 12, the distribution of property types shows that apartments and houses are the most common types, with apartments having the highest count by a significant margin. This indicates that most listings in the dataset are for apartments, followed by houses. Less common property types, such as parking spaces, lighthouses, and islands, have very few listings. The high prevalence of apartments and houses suggests that these types of properties are more readily available and likely cater to a broader range of guests.

- **Room Type**

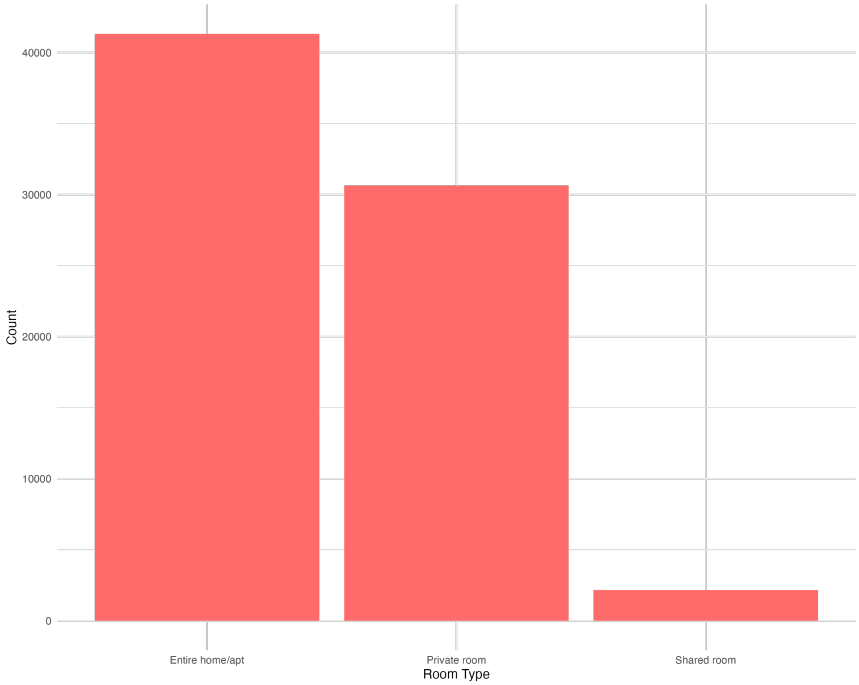



FIGURE 13. Distribution of Room Types

In Figure 13, the distribution of room types reveals that entire home/apartment listings are the most prevalent, followed by private rooms. Shared rooms are the least common. This suggests that many hosts prefer to rent out entire properties or private rooms rather than shared spaces, possibly to attract guests who value privacy and exclusivity.

## 5.2 City-wise Analysis

- **Room Type**

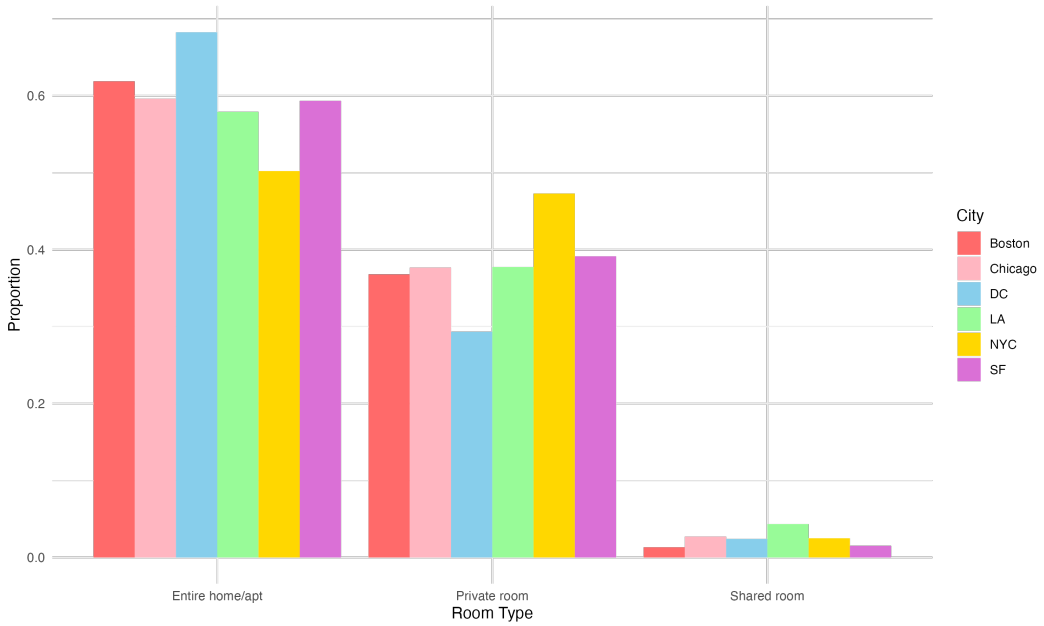



FIGURE 14. Popularity of Room Types by City

In Figure 14, the distribution of room types across different cities shows notable variations. Entire homes/apartments are most popular in all cities, with DC having the highest proportion. Private rooms are significantly popular in NYC and SF, while shared rooms have the lowest proportion across all cities, particularly in Boston and SF. The comparison is based on the proportion of room types since the total number of listings varies across cities.

- **Number of properties in each city**

In Figure 15, the distribution of listings shows that NYC and LA have the highest number of properties, significantly more than other cities. This high concentration of listings in NYC and LA indicates that these cities will have a substantial impact on overall property analysis and trends. Due to this distribution, NYC properties are likely to influence the overall results considerably.

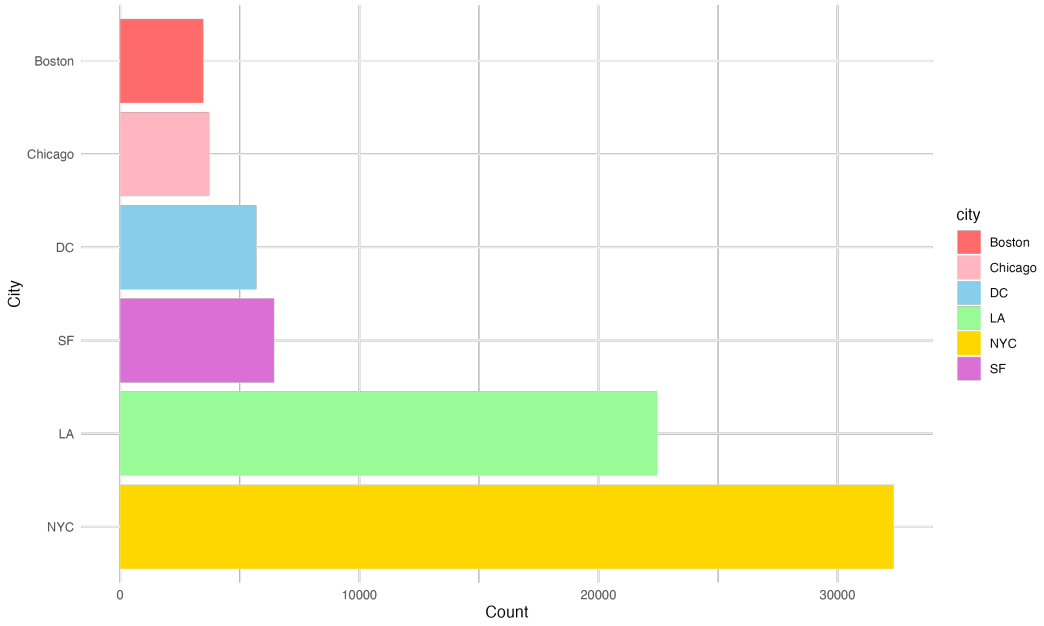



FIGURE 15. Distribution of Listings Across Different Cities

- **Neighborhoods**

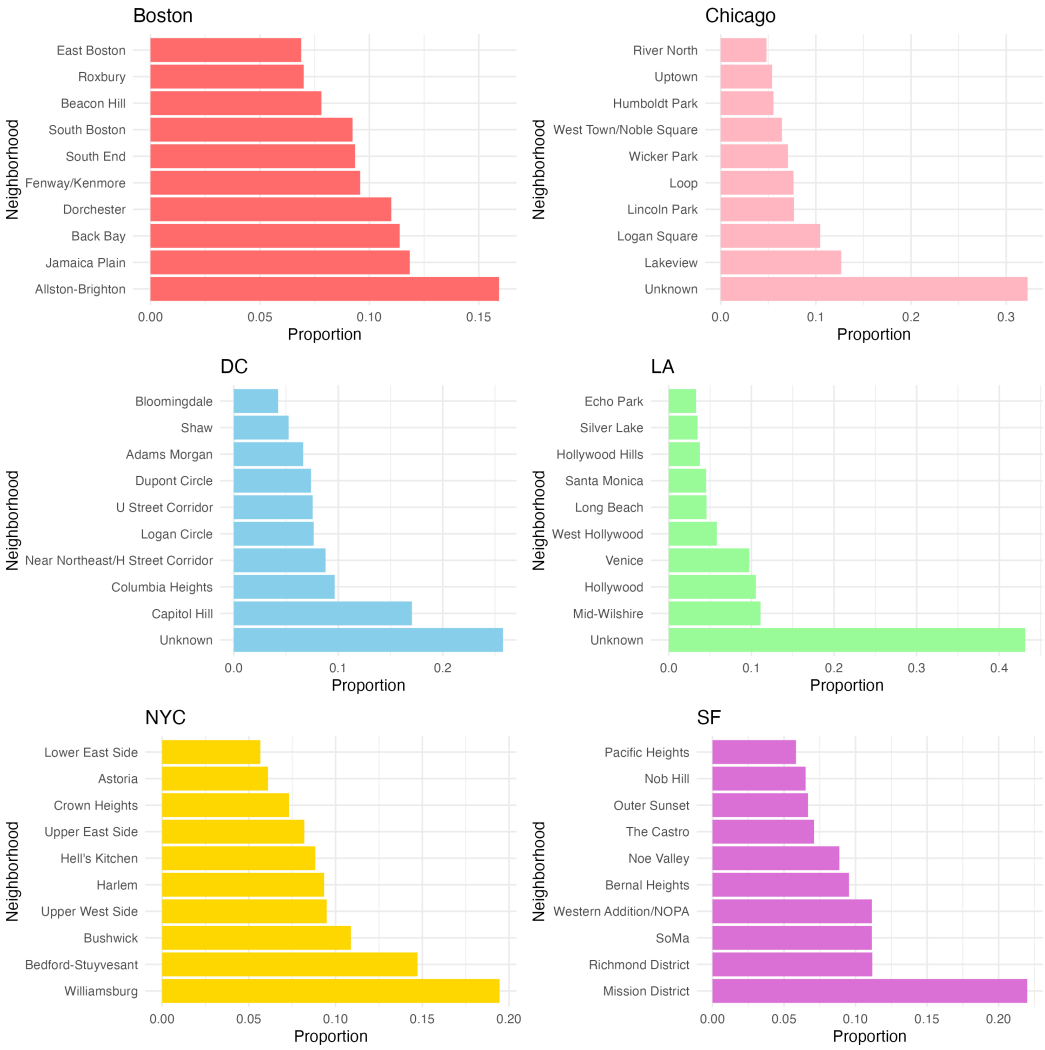



FIGURE 16. Distribution of Listings in Neighborhoods for the six Cities

As seen in Figure 16, the distribution of listings varies significantly across different neighborhoods within the six cities. For instance, in Boston, Allston-Brighton has the highest proportion of listings, while in Chicago, Lakeview is the most popular neighborhood, except for the places where the neighborhoods are unknown. DC shows a significant concentration of listings in the 'Unknown' category, suggesting many listings may not specify their neighborhood. In LA, 'Unknown' also dominates, followed by Mid-Wilshire. Williamsburg has the highest proportion in NYC, and Mission District leads in SF. The variation in neighborhood popularity indicates localized preferences within each city.

### 5.3 Review Scores Analysis

Given that many ratings are at 100, review scores may not be the best variable for prediction or detailed analysis. However, like log price, they are still a key variable that provides significant insights into property quality and guest satisfaction. Therefore, we dedicate a separate section to explaining and analyzing review scores in depth.


FIGURE 17. Distribution of Review Scores

The distribution of review scores shown in Figure 17 is heavily skewed towards the higher end, with most properties receiving ratings close to 100. This indicates that guests generally rate their stays very positively. The box plot on the right further emphasizes this, showing a concentration of scores around 100 with minimal variance, indicating a high level of reviewer satisfaction.

- **Room Type**

  Figure 18 shows the distribution of review scores by room type. The review scores for entire homes/apartments, private rooms, and shared rooms are all closely clustered around 100. However, entire homes/apartments and private rooms tend to have slightly higher review scores than shared rooms. The overall high ratings across all room types indicate a generally positive guest experience, with only minor variations between different room types.

- **Property Type**

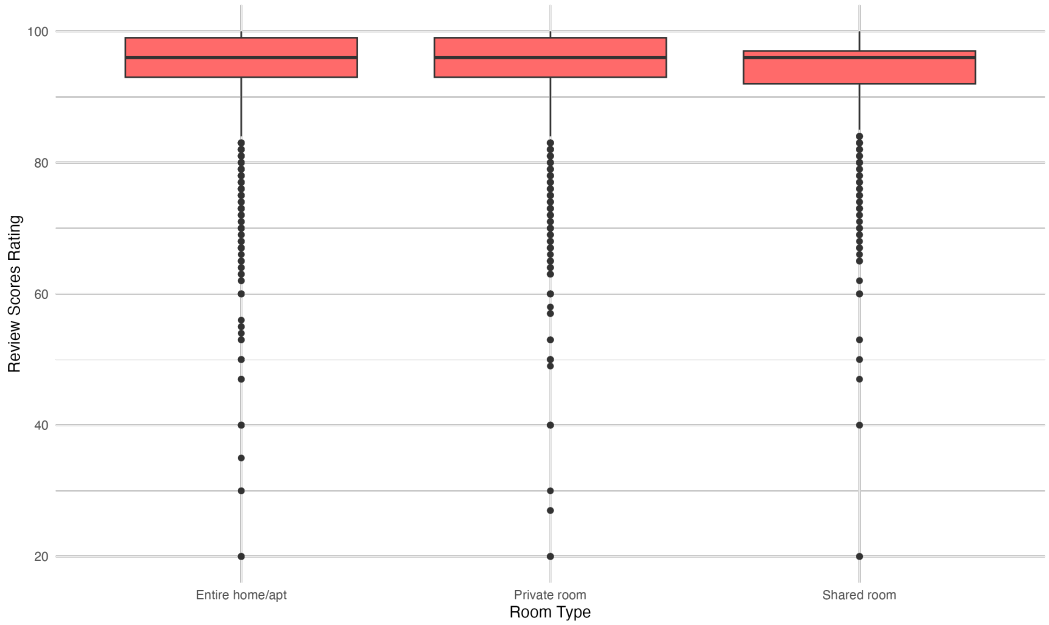



FIGURE 18. Review Scores by Room Type

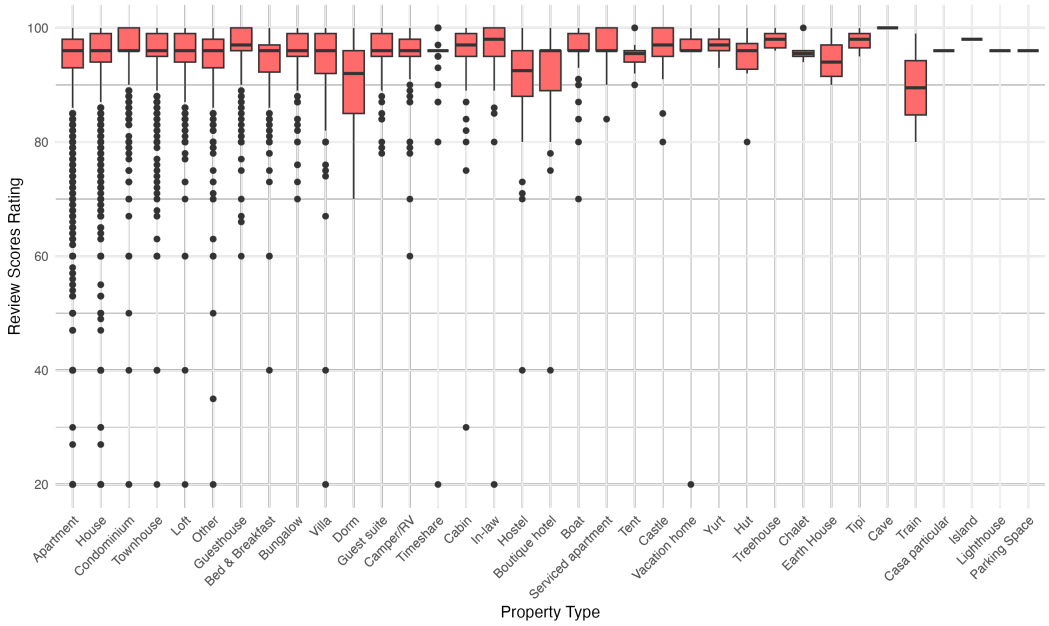



FIGURE 19. Review Scores by Property Type

Figure 19 illustrates the review scores by property type. Most property types have high review scores, generally clustering around 100. However, there is some variability, with certain property types like "Lighthouse," "Parking Space," and "Train" showing more dispersed and lower review scores. This variability suggests

that while most property types are consistently well-rated, a few specific types might offer a more varied guest experience.

- **Host Characteristics**

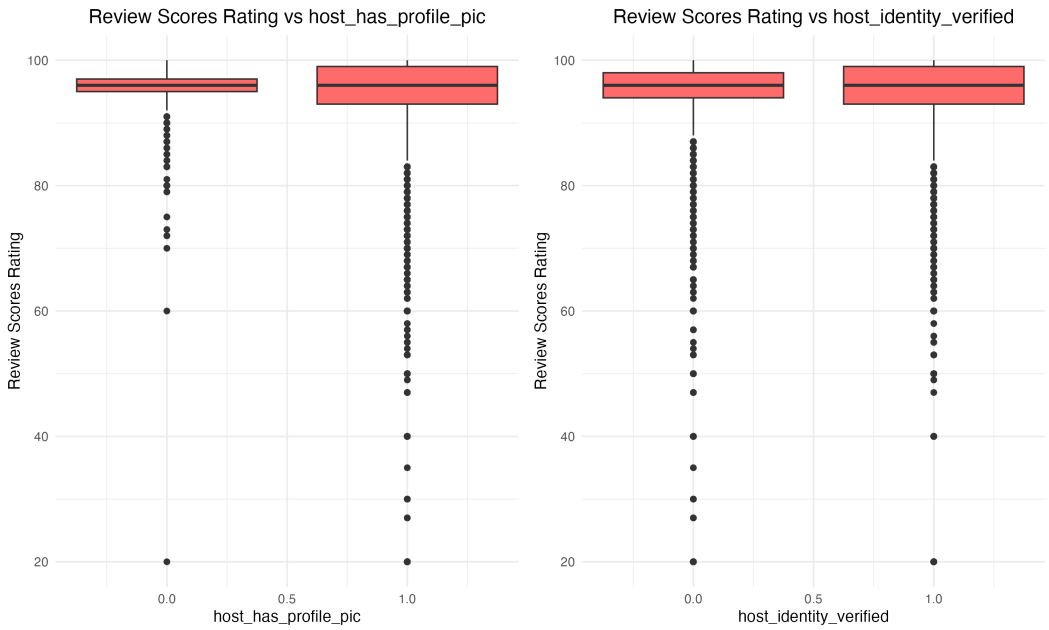



FIGURE 20. Review Scores Rating vs Host Characteristics

In Figure 20, we compare the review scores ratings based on two host characteristics: whether the host has a profile picture and the host's identity is verified. The box plots show hosts with a profile picture have slightly higher review scores than those without a profile picture. Similarly, hosts with a verified identity tend to have slightly higher review scores than those without identity verification. However, the differences are not substantial, indicating that these host characteristics do not significantly impact review scores as in the log price.

- **Amenities**

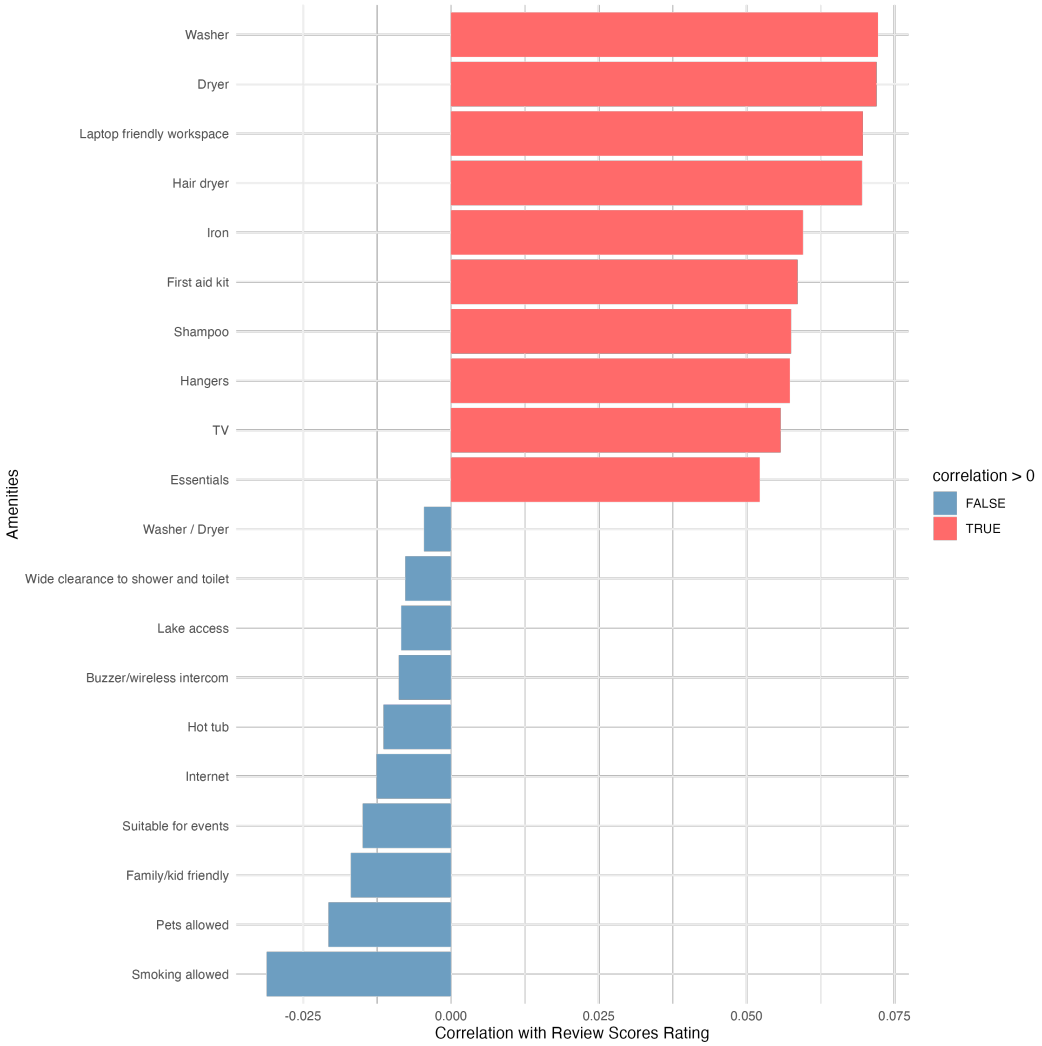



FIGURE 21. Top 10 Positive and Negative Impacts of Amenities on Review Scores

The top positive impacts of amenities on review scores include basic household items such as a washer, dryer, and a laptop-friendly workspace, indicating that essential conveniences are highly valued by guests. Conversely, the presence of smoking allowed, pets allowed, and family/kid-friendly amenities show a negative impact, possibly due to the preferences for a quieter, non-smoking environment. When comparing these results to the impact on log price (Figure 11), there are some similarities, such as the positive correlation of washers and dryers, but also differences; for example, the 'Family/kid friendly' amenity has a positive impact on price but a negative impact on review scores, suggesting varying guest priorities between cost and satisfaction.

# 6 Principal Component Analysis

## 6.1 Description

Principal Component Analysis (PCA) is a dimensionality reduction technique used to reduce the dimensionality of a dataset while preserving its variance as much as possible. It achieves this by transforming the original variables into a new set of orthogonal variables called *principal components*. It is useful for visualizing high--dimensional data, identifying patterns, and reducing noise.

## 6.2 PCA using only Numeric Variables

- Since PCA is mainly suitable for numeric and continuous variables as it operates on the covariance matrix. Only numeric columns are included in the first example analysis:
- The numeric variables contain: `log_price`, `accommodates`, `bathrooms`, `host_response_rate`, `latitude`, `longitude`, `number_of_reviews`, `review_scores_rating`, `bedrooms`, `beds`, `days_since_first_review`, `days_since_last_review`, and `host_duration`.

### 6.2.1 The Summary Table of Principal Components

We can see from Table 4 that PC1 explains 24.98% of the total variance, and PC2 explains 16.41%. Together, PC1 and PC2 (the first two components) explain a cumulative proportion of 41.39% of the total variance.

TABLE 4. Importance of Principal Components

| | PC1 | PC2 | PC3 | PC4 | PC5 | PC6 |
|---|---|---|---|---|---|---|
| Standard deviation | 1.7313 | 1.4031 | 1.3226 | 1.0981 | 1.0106 | 0.9779 |
| Proportion of Variance | 0.2498 | 0.1641 | 0.1458 | 0.1005 | 0.0851 | 0.0797 |
| Cumulative Proportion | 0.2498 | 0.4139 | 0.5596 | 0.6601 | 0.7452 | 0.8249 |
| | PC7 | PC8 | PC9 | PC10 | PC11 | PC12 |
| Standard deviation | 0.8067 | 0.7452 | 0.5632 | 0.5359 | 0.4346 | 0.3192 |
| Proportion of Variance | 0.0542 | 0.0463 | 0.0264 | 0.0239 | 0.0157 | 0.0085 |
| Cumulative Proportion | 0.8791 | 0.9254 | 0.9518 | 0.9758 | 0.9915 | 1.0000 |

### 6.2.2 The Scree Plot of Principal Components

The scree plot, as indicated in Figure 22, visualizes the variance explained by each principal component, indicating a decreasing trend in variance explained by each subsequent component.

### 6.2.3 The Scatter Plot of Principal Component

The data points, as indicated in Figure 23, show a concentration on the left, suggesting that the variability in your data is primarily captured by the first principal component (PC1). Moreover, the lesser spread along the second principal component (PC2) axis implies that while PC2 contributes to capturing variability, its

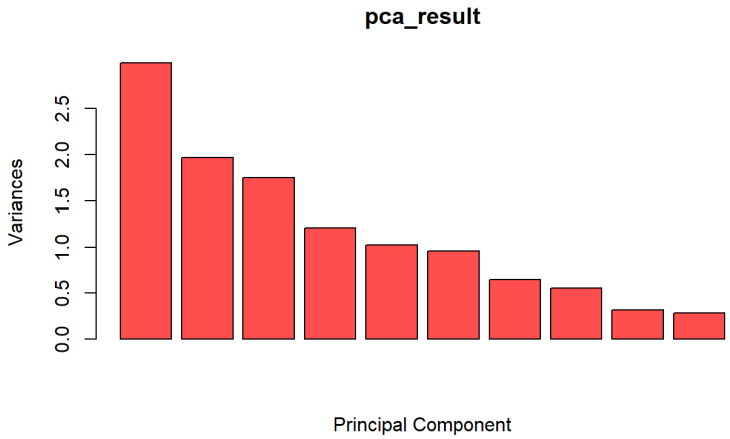



FIGURE 22. Scree Plot of Principal Component

influence is relatively smaller compared to PC1. This pattern underscores the importance of PC1 in the dimensionality reduction process and justifies its selection for further analysis and modeling.

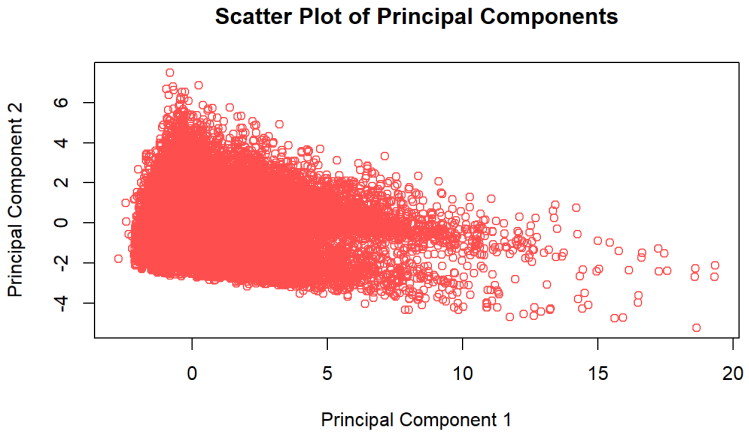



FIGURE 23. Scatter Plot of PCA Component

### 6.2.4 AICc Plot for Model Selection

- The Akaike Information Criterion corrected (AICc) was used to select the optimal number of principal components for modeling.
- The AICc plot, as indicated in Figure 24, shows a minimum value at 12 principal components, suggesting that using all 12 components provides the best model fit according to the AICc criterion.

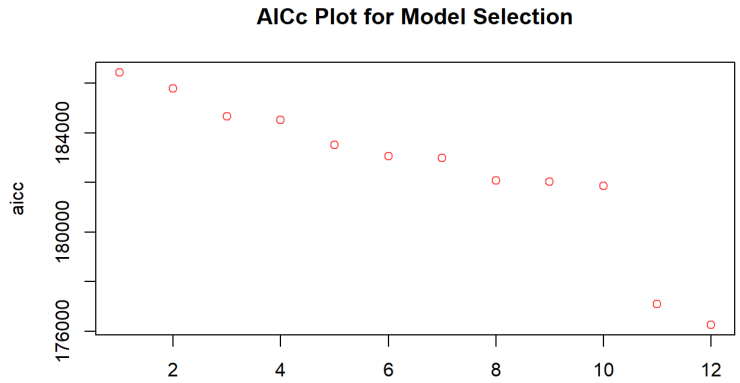



FIGURE 24. AICc Plot for Model Selection

### 6.2.5 Generalized Linear Model (GLM)

A GLM was fitted using the selected 12 principal components as predictors and the `log_price` as the response variable.

```
Call:
glm(formula = log_price ~ ., family = gaussian, data = pca_df)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-6.0912  -0.5084  -0.0237   0.4777   4.6717

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.372e-16  2.919e-03   0.000        1
PC1          3.032e-01  1.686e-03 179.808  < 2e-16 ***
PC2         -5.701e-02  2.080e-03 -27.402  < 2e-16 ***
PC3          7.833e-02  2.207e-03  35.488  < 2e-16 ***
PC4         -3.376e-02  2.658e-03 -12.699  < 2e-16 ***
PC5          9.700e-02  2.889e-03  33.582  < 2e-16 ***
PC6          6.728e-02  2.985e-03  22.539  < 2e-16 ***
PC7         -3.236e-02  3.619e-03  -8.944  < 2e-16 ***
PC8         -1.226e-01  3.917e-03 -31.291  < 2e-16 ***
PC9         -3.983e-02  5.183e-03  -7.684 1.56e-14 ***
PC10         7.199e-02  5.447e-03  13.216  < 2e-16 ***
PC11         4.744e-01  6.718e-03  70.620  < 2e-16 ***
PC12         2.646e-01  9.145e-03  28.937  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.6315095)

    Null deviance: 74110  on 74110  degrees of freedom
Residual deviance: 46794  on 74098  degrees of freedom
AIC: 176268

Number of Fisher Scoring iterations: 2
```

FIGURE 25. Summary Table of GLM Model

The results, as indicated in Figure 25, show that all principal components have significant effects on the `log_price`, as evidenced by their low p-values in the GLM analysis.

### 6.2.6 Lasso Regression

A Lasso regression was applied using cross-validation to select the optimal regularization parameter.

```
13 x 1 sparse Matrix of class "dgCMatrix"
                      s1
(Intercept)  3.943241e-16
PC1          2.925212e-01
PC2         -4.387320e-02
PC3          6.439348e-02
PC4         -1.697507e-02
PC5          7.876617e-02
PC6          4.843329e-02
PC7         -9.517705e-03
PC8         -9.784452e-02
PC9         -7.104454e-03
PC10         3.760060e-02
PC11         4.319853e-01
PC12         2.068982e-01
```

FIGURE 26. Summary Table of Lasso Regression

The coefficients obtained from the Lasso model, as indicated in Figure 26, show the magnitude of the effect of each principal component on the `log_price` after regularization.

### 6.2.7 Interpretation

Both GLM and Lasso models suggest that the principal components have significant associations with the `log_price` variable. The results indicate that the 12 principal components all capture relevant information for predicting the `log_price` based on the provided numerical variables.

## 6.3 PCA using All Columns

Categorical variables are transformed into dummy variables. Additionally, 131 amenities variables are further categorized into the following groups: Essentials, Facilities, Parking, Privacy, Family, Safety, Pets, Kitchen, Internet, Self-Checkin, Accessibility, Events, Others, Bathroom, Outdoor, Smoking, and Miscellaneous.

### 6.3.1 The Summary Table of Principal Components

Table 5. Importance of Principal Components

| | PC1 | PC2 | PC3 | PC4 | PC5 | PC6 |
|---|---|---|---|---|---|---|
| Standard deviation | 2.05200 | 1.74716 | 1.73209 | 1.48004 | 1.34121 | 1.22530 |
| Proportion of Variance | 0.10800 | 0.07827 | 0.07693 | 0.05617 | 0.04612 | 0.03850 |
| Cumulative Proportion | 0.10800 | 0.18623 | 0.26316 | 0.31932 | 0.36545 | 0.40390 |
| | PC7 | PC8 | PC9 | PC10 | PC11 | PC12 |
| Standard deviation | 1.15809 | 1.14928 | 1.10778 | 1.05839 | 1.04532 | 1.01270 |
| Proportion of Variance | 0.03439 | 0.03387 | 0.03147 | 0.02872 | 0.02802 | 0.02630 |
| Cumulative Proportion | 0.43833 | 0.47220 | 0.50367 | 0.53239 | 0.56041 | 0.58670 |
| | PC13 | PC14 | PC15 | PC16 | PC17 | PC18 |
| Standard deviation | 1.01024 | 0.99920 | 0.98600 | 0.97595 | 0.95640 | 0.94372 |
| Proportion of Variance | 0.02617 | 0.02560 | 0.02493 | 0.02442 | 0.02345 | 0.02284 |
| Cumulative Proportion | 0.61287 | 0.63850 | 0.66341 | 0.68783 | 0.71128 | 0.73412 |
| | PC19 | PC20 | PC21 | PC22 | PC23 | PC24 |
| Standard deviation | 0.91652 | 0.90930 | 0.90125 | 0.88787 | 0.87166 | 0.85620 |
| Proportion of Variance | 0.02154 | 0.02120 | 0.02083 | 0.02021 | 0.01948 | 0.01880 |
| Cumulative Proportion | 0.75566 | 0.77690 | 0.79768 | 0.81790 | 0.83738 | 0.85620 |
| | PC25 | PC26 | PC27 | PC28 | PC29 | PC30 |
| Standard deviation | 0.84738 | 0.83068 | 0.79892 | 0.77074 | 0.74492 | 0.68833 |
| Proportion of Variance | 0.01841 | 0.01769 | 0.01637 | 0.01523 | 0.01423 | 0.01215 |
| Cumulative Proportion | 0.87459 | 0.89228 | 0.90864 | 0.92388 | 0.93810 | 0.95025 |
| | PC31 | PC32 | PC33 | PC34 | PC35 | PC36 |
| Standard deviation | 0.67688 | 0.62783 | 0.57121 | 0.53498 | 0.50012 | 0.40996 |
| Proportion of Variance | 0.01175 | 0.01011 | 0.00837 | 0.00734 | 0.00641 | 0.00431 |
| Cumulative Proportion | 0.96200 | 0.97211 | 0.98047 | 0.98781 | 0.99423 | 0.99854 |
| | PC37 | PC38 | PC39 | | | |
| Standard deviation | 0.23897 | 1.418e-14 | 1.671e-16 | | | |
| Proportion of Variance | 0.00146 | 0.000e+00 | 0.000e+00 | | | |
| Cumulative Proportion | 1.00000 | 1.000e+00 | 1.000e+00 | | | |

The summary of PCA, as indicated in Table 5, shows the standard deviation, proportion of variance, and cumulative proportion explained by each principal component. We can see PC1 explains 10.8% of the total variance, PC2 explains 7.83%, and so on. Higher proportions imply that the corresponding principal component captures a larger amount of the total variance in the data.

### 6.3.2 The Scree Plot of Principal Components

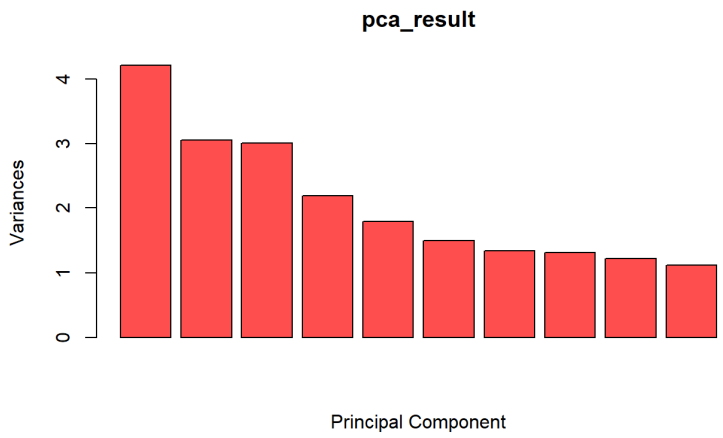



FIGURE 27. Scree Plot of Principal Component with All Variables

Figure 27 shows the scree plot of the principal components, illustrating the variances explained by each component. The plot reveals that the first few principal components capture the majority of the variance in the dataset.

### 6.3.3 The Scatter Plot of Principal Component

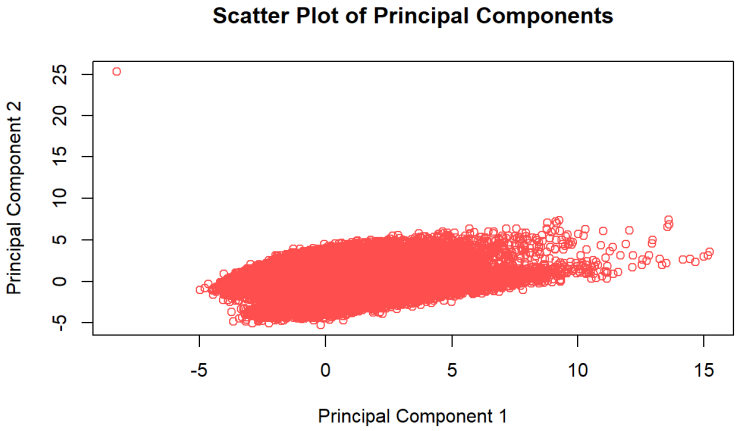



FIGURE 28. Scatter Plot of Principal Component will All Variables

Figure 28 shows a scatter plot of the first two principal components, highlighting the distribution of the data points. The plot indicates that the majority of the variability in the dataset is captured by these components, with a noticeable concentration along the principal component 1 axis.

### 6.3.4 AICc Plot for Model Selection

Figure 29 indicates that the minimum AICc value, suggesting the best model, is achieved when using 37 principal components. This means that a model with 37 principal components strikes the optimal balance between explaining the variability in the data and avoiding overfitting.

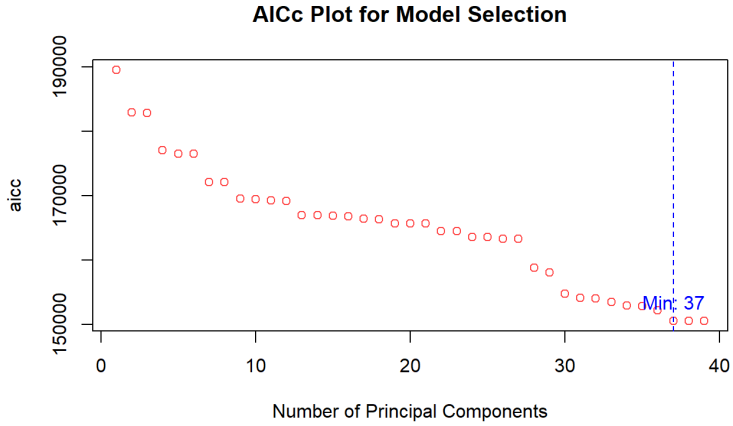



FIGURE 29. AICc Plot for Model Selection with All Variables

### 6.3.5 Generalized Linear Model (GLM)

A GLM was fitted using the first 37 principal components, which were selected based on the AICc criterion.

```
Call:
glm(formula = log_price ~ ., family = gaussian, data = pca_df[1:37])

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-5.4809  -0.4211  -0.0425   0.3560   5.1674

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.400e-16  2.455e-03   0.000  1.00000
PC1          2.409e-01  1.196e-03 201.358  < 2e-16 ***
PC2          1.455e-01  1.405e-03 103.573  < 2e-16 ***
PC3         -1.886e-02  1.417e-03 -13.311  < 2e-16 ***
PC4          1.539e-01  1.658e-03  92.779  < 2e-16 ***
PC5         -5.065e-02  1.830e-03 -27.677  < 2e-16 ***
PC6         -1.743e-04  2.003e-03  -0.087  0.93067
PC7         -1.642e-01  2.119e-03 -77.474  < 2e-16 ***
PC8          1.756e-02  2.136e-03   8.221  < 2e-16 ***
PC9         -1.289e-01  2.216e-03 -58.181  < 2e-16 ***
PC10        -1.516e-02  2.319e-03  -6.535 6.40e-11 ***
PC11        -4.311e-02  2.348e-03 -18.358  < 2e-16 ***
PC12         1.990e-02  2.424e-03   8.209 2.27e-16 ***
PC13         1.293e-01  2.430e-03  53.230  < 2e-16 ***
PC14        -6.757e-03  2.456e-03  -2.751  0.00594 **
PC15        -1.504e-02  2.489e-03  -6.043 1.52e-09 ***
PC16        -2.955e-02  2.515e-03 -11.749  < 2e-16 ***
PC17        -5.478e-02  2.566e-03 -21.343  < 2e-16 ***
PC18        -3.070e-02  2.601e-03 -11.805  < 2e-16 ***
PC19         7.708e-02  2.678e-03  28.782  < 2e-16 ***
PC20        -2.016e-03  2.699e-03  -0.747  0.45528
PC21         1.719e-03  2.723e-03   0.631  0.52802
PC22        -1.033e-01  2.764e-03 -37.353  < 2e-16 ***
PC23         6.461e-03  2.816e-03   2.295  0.02176 *
PC24        -9.447e-02  2.867e-03 -32.952  < 2e-16 ***
PC25         1.237e-02  2.897e-03   4.269 1.96e-05 ***
PC26        -5.025e-02  2.955e-03 -17.007  < 2e-16 ***
PC27        -3.097e-02  3.072e-03 -10.079  < 2e-16 ***
PC28        -2.277e-01  3.185e-03 -71.508  < 2e-16 ***
PC29         9.812e-02  3.295e-03  29.779  < 2e-16 ***
PC30        -2.105e-01  3.566e-03 -59.026  < 2e-16 ***
PC31         9.853e-02  3.626e-03  27.172  < 2e-16 ***
PC32        -2.743e-02  3.910e-03  -7.017 2.28e-12 ***
PC33        -1.050e-01  4.297e-03 -24.438  < 2e-16 ***
PC34        -1.092e-01  4.588e-03 -23.790  < 2e-16 ***
PC35        -5.718e-02  4.908e-03 -11.650  < 2e-16 ***
PC36        -1.559e-01  5.987e-03 -26.043  < 2e-16 ***
PC37        -4.110e-01  1.027e-02 -40.014  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.446489)

    Null deviance: 74110  on 74110  degrees of freedom
Residual deviance: 33073  on 74073  degrees of freedom
AIC: 150599

Number of Fisher Scoring iterations: 2
```

FIGURE 30. Summary Table of GLM Model

Figure 30 shows that out of the 37 principal components, 34 were statistically significant with p-values less than 0.05.

### 6.3.6 Lasso Regression

```
38 x 1 sparse Matrix of class "dgCMatrix"
                      s1
(Intercept)  4.967721e-16
PC1          1.957321e-01
PC2          9.250012e-02
PC3          .
PC4          9.129320e-02
PC5          .
PC6          .
PC7         -8.423479e-02
PC8          .
PC9         -4.531141e-02
PC10         .
PC11         .
PC12         .
PC13         3.765852e-02
PC14         .
PC15         .
PC16         .
PC17         .
PC18         .
PC19         .
PC20         .
PC21         .
PC22         .
PC23         .
PC24         .
PC25         .
PC26         .
PC27         .
PC28        -1.075665e-01
PC29         .
PC30        -7.593608e-02
PC31         .
PC32         .
PC33         .
PC34         .
PC35         .
PC36         .
PC37        -2.344882e-02
```

FIGURE 31. Summary Table of Lasso Regression

Lasso regression with cross-validation was applied to the first 37 principal components. The significant coefficients (non-zero) from the Lasso model indicated the most relevant principal components for predicting `log_price`. Figure 31 shows that Lasso identified 10 significant principal components: PC1, PC2, PC4, PC7, PC9, PC13, PC28, PC30, and PC37.

## 6.4 Conclusion

Overall, the PCA analysis demonstrates that PCA combined with regression techniques can effectively handle high-dimensional data, providing insights into the target variable's underlying structure and key predictors.

# 7 K-Means Analysis

## 7.1 Elbow Method (Within-Cluster Sum of Squares):

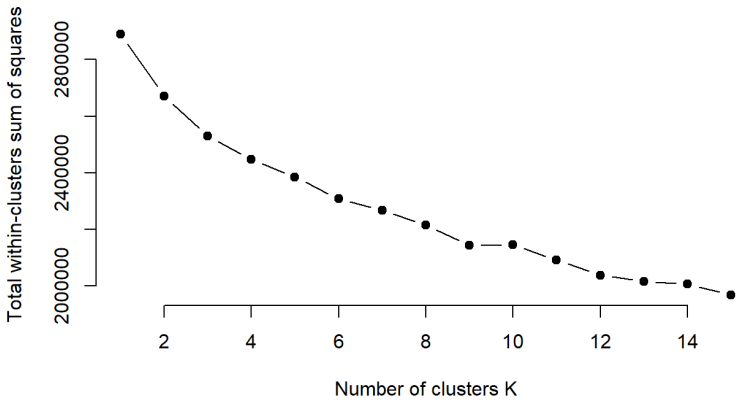



FIGURE 32. WSS Plot

Figure 32 illustrates the within-cluster sum of squares (WSS) plotted against the number of clusters (K). The plot shows a clear "elbow point" at K = 4, where the decrease in WSS starts to level off. This indicates that the optimal number of clusters for this dataset is 4, as adding more clusters beyond this point results in only marginal improvements in the WSS, suggesting diminishing returns in terms of clustering performance.

## 7.2 Silhouette Method:

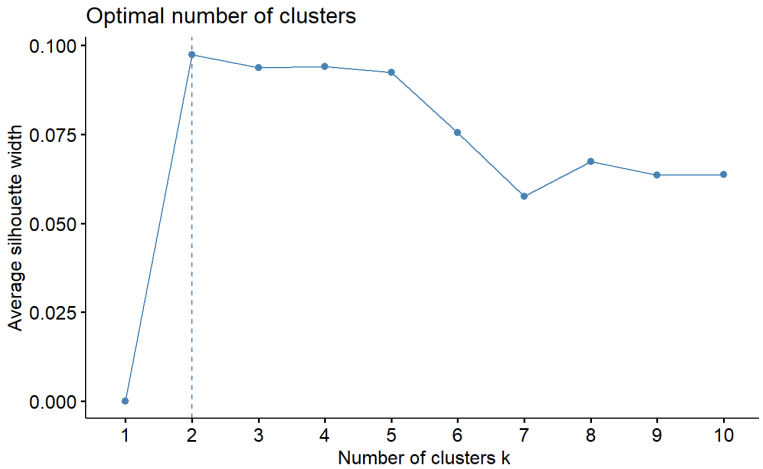



FIGURE 33. Silhouette Plot

Figure 33 uses silhouette scores to determine the optimal number of clusters. The silhouette score measures how similar an object is to its own cluster compared to other clusters. The plot indicates that the optimal number of clusters is 2, as this

is where the average silhouette score is maximized. This suggests that two clusters provide the best separation and cohesion for the data, leading to more distinct and well-defined clusters. Beyond two clusters, the silhouette score decreases, indicating less distinct clustering and potential overfitting.

### 7.2.1 Comparison of K-Means Clustering with Different Numbers of Clusters

Each point in Figure 34 represents an accommodation, and the colors indicate the different clusters. This visualization helps to distinguish the groups and understand the distribution of accommodations within each cluster.

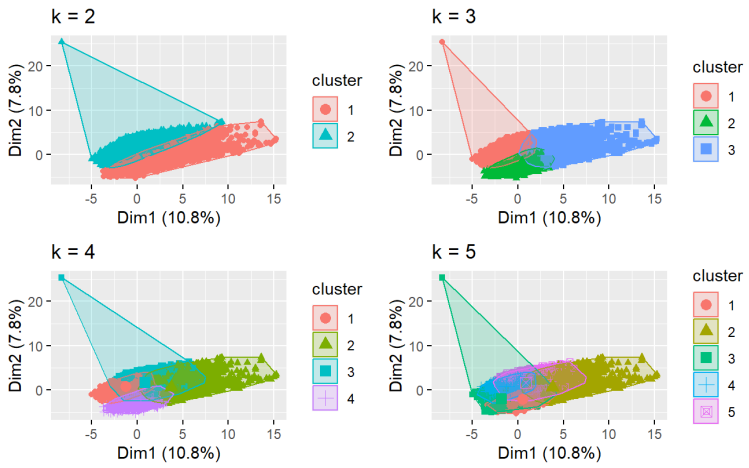



FIGURE 34. Comparison of 2 - 5 K-Means Clusters

### 7.2.2 Optimal Clustering with K = 2

Based on the silhouette method, the optimal number of clusters, K=2, is selected. The silhouette plot (Figure 33) indicates that K=2 maximizes the average silhouette score, suggesting that this number of clusters provides the best separation and cohesion for the data.

Although the WSS plot (Figure 32) shows a noticeable "elbow point" at K=4, indicating a reduction in the within-cluster sum of squares, K=2 was chosen because it allows for easier interpretation and analysis of the clusters. With two clusters, the characteristics of each cluster are more distinct and manageable, facilitating a clearer understanding of the data and the key factors that differentiate the clusters.

(1) **Cluster Centers:** The output of `k2$centers` shows the cluster centers for each variable. These centers represent the average values of the variables for each cluster, indicating the characteristics of the data points in each cluster.
   - There are significant differences between Cluster 1 and Cluster 2 in the variables of *longitude, latitude, parking, and property type*, indicating clear distinctions in these characteristics between the two clusters.

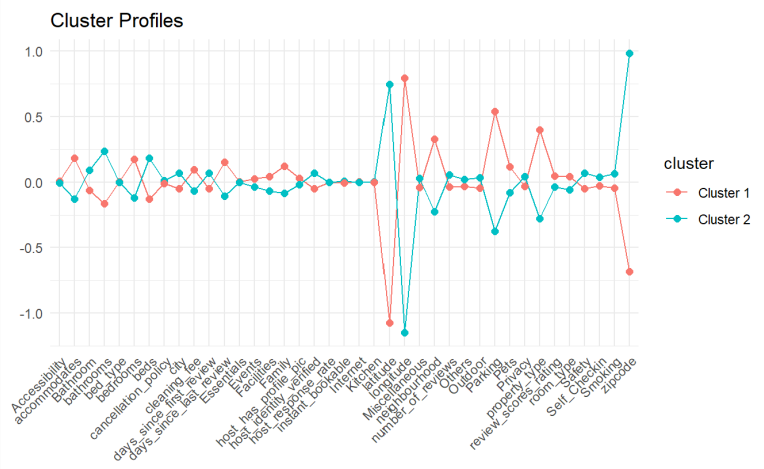



FIGURE 35. Cluster Profile of K=2

- For the variable of *Parking*: Since Cluster 1 has higher values for the "Parking" variable, accommodations in Cluster 1 are more likely to offer better parking services compared to Cluster 2.
- For the variable of *Family*: Cluster 1 has higher values for the "Family" variable, suggesting that accommodations in this cluster are more family-friendly. This includes amenities such as kid-friendly features, children's books and toys, children's dinnerware, cribs, high chairs, stair gates, etc.
- For the variable of *Bathrooms*: Cluster 1 also has higher values for the number of bathrooms, indicating that accommodations in this cluster are more likely to offer more bathrooms.
- For the variable of *Review Scores Rating*: Cluster 1 has higher review scores than Cluster 2, suggesting that accommodations in Cluster 1 are rated more favorably by guests.

Overall, Cluster 1 tends to have higher values for these variables, which may indicate that the accommodations in Cluster 1 are generally better or offer more amenities than those in Cluster 2.

(2) **Top 10 Variables that Most Distinguish the Two Clusters:**

- **Cluster 1** is primarily characterized by location-specific variables (e.g., zipcode, neighborhood), amenities related to accommodation capacity (e.g., bathrooms, accommodates, beds, bedrooms), and features appealing to families.
- **Cluster 2** is differentiated by geographical coordinates (longitude and latitude), room and facility types, outdoor amenities, and other specific features.

This differentiation provides insights into the distinct characteristics and features that separate the two clusters, which can help tailor services or marketing strategies for each group.

(3) **Summary Statistics of Log Prices:**

| Cluster | Property Type | Room Type | Accommodates |
|---|---|---|---|
| 1 | 0.3985161 | -0.05970107 | 0.1848483 |
| 2 | -0.2767880 | 0.04146518 | -0.1283857 |
| **Cluster** | **Bathrooms** | **Bed Type** | **Cancellation Policy** |
| 1 | 0.2377917 | 0.003152604 | 0.012465888 |
| 2 | -0.1651574 | -0.002189631 | -0.008658142 |
| **Cluster** | **Cleaning Fee** | **City** | **Host Has Profile Pic** |
| 1 | 0.09483309 | 0.06919552 | 0.02945537 |
| 2 | -0.06586601 | -0.04805952 | -0.02045813 |
| **Cluster** | **Host Identity Verified** | **Host Response Rate** | **Instant Bookable** |
| 1 | 0.07139106 | -0.0006199509 | 0.007077341 |
| 2 | -0.04958443 | 0.0004305849 | -0.004915544 |
| **Cluster** | **Latitude** | **Longitude** | **Neighbourhood** |
| 1 | -1.0744381 | -1.1473738 | 0.3272967 |
| 2 | 0.7462474 | 0.7969047 | -0.2273229 |
| **Cluster** | **Number of Reviews** | **Review Scores Rating** | **Days Since First Review** |
| 1 | 0.05484926 | 0.04912955 | 0.06969415 |
| 2 | -0.03809537 | -0.03412277 | -0.04840584 |
| **Cluster** | **Days Since Last Review** | **Zipcode** | **Bedrooms** |
| 1 | 0.1533787 | 0.9844294 | 0.1742993 |
| 2 | -0.1065286 | -0.6837322 | -0.1210590 |
| **Cluster** | **Beds** | **Essentials** | **Facilities** |
| 1 | 0.1819626 | 0.003673319 | -0.06474507 |
| 2 | -0.1263815 | -0.002551292 | 0.04496848 |
| **Cluster** | **Parking** | **Privacy** | **Family** |
| 1 | 0.5401212 | 0.04271227 | 0.12421423 |
| 2 | -0.3751394 | -0.02966567 | -0.08627258 |
| **Cluster** | **Safety** | **Pets** | **Kitchen** |
| 1 | 0.07157302 | 0.11708982 | 0.003673319 |
| 2 | -0.04971081 | -0.08132435 | -0.002551292 |
| **Cluster** | **Internet** | **Self Checkin** | **Accessibility** |
| 1 | 0.003673319 | 0.03805538 | 0.009204160 |
| 2 | -0.002551292 | -0.02643124 | -0.006392719 |
| **Cluster** | **Events** | **Others** | **Bathroom** |
| 1 | -0.03591121 | -0.02995578 | 0.09058853 |
| 2 | 0.02494201 | 0.02080568 | -0.06291797 |
| **Cluster** | **Outdoor** | **Smoking** | **Miscellaneous** |
| 1 | -0.04680085 | 0.06394040 | -0.04078856 |
| 2 | 0.03250538 | -0.04440959 | 0.02832956 |

TABLE 6. Cluster Analysis Results

- Accommodations in Cluster 1 tend to have higher log prices (and hence higher prices) than those in Cluster 2.
- There is greater variability in the prices of accommodations in Cluster 1, as evidenced by the higher standard deviation.

| Cluster 1 | Cluster 2 |
|---|---|
| "zipcode" | "longitude" |
| "Parking" | "latitude" |
| "property_type" | "Facilities" |
| "neighbourhood" | "room_type" |
| "bathrooms" | "Outdoor" |
| "accommodates" | "Miscellaneous" |
| "beds" | "Events" |
| "bedrooms" | "Others" |
| "days_since_last_review" | "host_response_rate" |
| "Family" | "bed_type" |

TABLE 7. Top 10 Variables that Most Distinguish the Two Clusters

| Cluster | Mean Log Price | Median Log Price | Standard Deviation Log Price |
|---|---|---|---|
| 1 | 0.1004 | 0.0076 | 1.0702 |
| 2 | -0.0698 | -0.1265 | 0.9420 |

TABLE 8. Summary Statistics of Log Price by Cluster


FIGURE 36. Log Price Distribution Across Clusters

### 7.2.3 Information Criteria (AIC/BIC):

To determine the optimal number of clusters for the k-means clustering algorithm, statistical criteria such as the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC) are used. These criteria evaluate the goodness-of-fit of the model while penalizing for the number of parameters (clusters), balancing model fit and complexity.

Steps Involved:

- Multiple k-means models are fit to the data, each with a different number of clusters ranging from 1 to 10.
- For each k-means model, the AIC and BIC values are calculated.

- The number of clusters corresponding to the minimum AIC and BIC values is selected as the optimal number of clusters.
- The process identified **10 clusters** as the optimal number of clusters based on the BIC value.

(1) Top 5 Variables that Most Distinguish the 10 Clusters
To illustrate the distinguishing characteristics of the clusters, we will examine three example clusters:
- Cluster 1: Accommodations in this cluster are characterized by higher values in miscellaneous amenities, self-check-in options, outdoor facilities, privacy features, and a higher number of reviews.
- Cluster 2: Accommodations in this cluster are more likely to allow smoking, be pet-friendly, have specific room types, host events, and offer parking facilities.
- Cluster 3: This cluster is defined by particular room types, various other amenities, specific zip codes, neighborhood characteristics, and the recency of the last review.

| **1** | **2** | **3** | **4** |
|---|---|---|---|
| Miscellaneous | Smoking | room_type | Accessibility |
| Self_Checkin | Pets | Others | Events |
| Outdoor | room_type | zipcode | Family |
| Privacy | Events | neighbourhood | Bathroom |
| number_of_reviews | Parking | days_since_last_review | days_since_first_review |
| **5** | **6** | **7** | **8** |
| cancellation_policy | days_since_last_review | zipcode | zipcode |
| room_type | Others | Family | latitude |
| longitude | room_type | Parking | Others |
| latitude | neighbourhood | neighbourhood | room_type |
| Facilities | property_type | property_type | Facilities |
| **9** | **10** | | |
| room_type | longitude | | |
| latitude | latitude | | |
| Others | Others | | |
| room_type | room_type | | |
| Facilities | Facilities | | |

TABLE 9. Top Variables that Most Distinguish the 10 Clusters

As shown in Table 9, this highlights that different clusters are characterized by distinct sets of features, suggesting that accommodations within each cluster share common attributes. Understanding these key characteristics helps identify each cluster's defining traits, which can be used for targeted marketing, improved customer experience, and better accommodation recommendations.

(2) **Summary Statistics of Log Prices**
Table 10 presents the summary statistics of log prices for each cluster, including the mean, median, and standard deviation of log prices.

| Cluster | mean_log_price | median_log_price | sd_log_price |
|---|---|---|---|
| 1 | 0.1016320 | 0.10836907 | 0.8658702 |
| 2 | -0.3561208 | -0.47312628 | 0.9263098 |
| 3 | -0.3578067 | -0.47312628 | 1.1062021 |
| 4 | 0.1945448 | 0.15615602 | 0.8902140 |
| 5 | 1.4500326 | 1.45927736 | 1.0627136 |
| 6 | 0.2440016 | 0.22243476 | 0.8588549 |
| 7 | -0.3947571 | -0.47312628 | 0.9506722 |
| 8 | -0.3877028 | -0.40902604 | 0.8624963 |
| 9 | 0.1112507 | 0.06446199 | 0.8390096 |
| 10 | -0.3996511 | -0.47312628 | 0.8312889 |

TABLE 10. Cluster Statistics

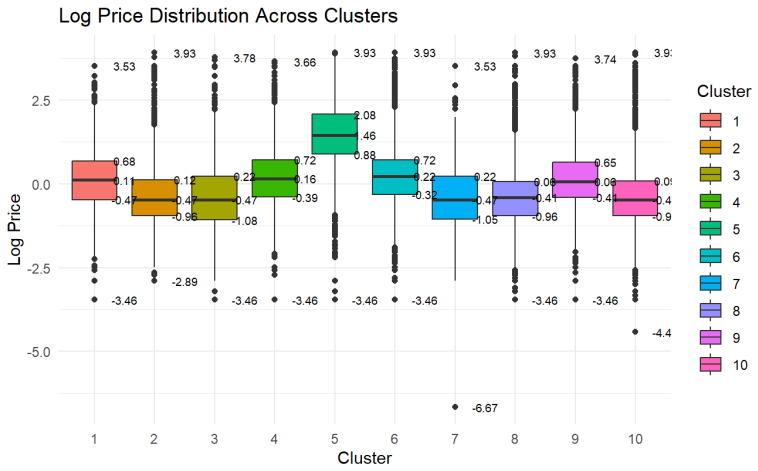



FIGURE 37. Log Price Distribution Across Clusters

Figure 37 illustrates the log price distribution across different clusters.

- Cluster 5 has the highest median log price, indicating that properties in this cluster are generally more expensive.
- Clusters 2, 3, 7, 8, and 10 have lower median log prices, suggesting that properties in these clusters are less expensive.
- The variability within clusters is also evident, with some clusters showing a wider spread in log prices than others.

(3) **K-Means Clustering of Accommodations with 10 Clusters**

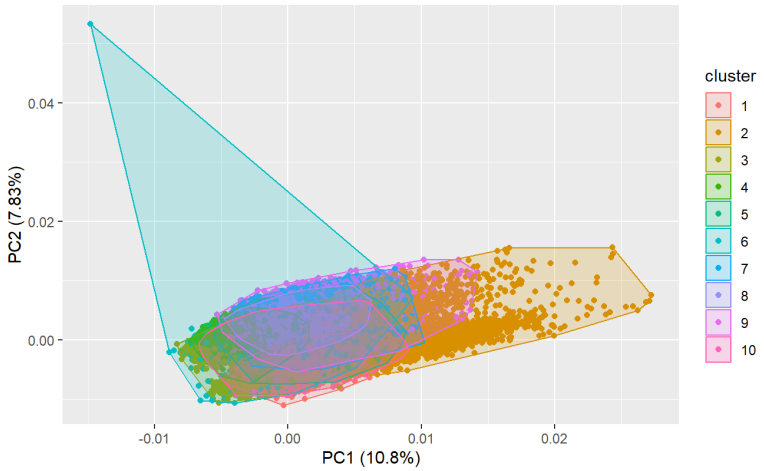



FIGURE 38. Summary Statistics of Log Prices

Figure 38 illustrates that the 10 clusters do not exhibit clear separation and there is significant overlap among them. This suggests that the features used for clustering might not distinctly differentiate the accommodations into separate groups.

## 7.3 Conclusion

In our analysis, we explored the optimal number of clusters for K-means clustering using different methods. The silhouette method indicated that K=2 is the optimal number of clusters, which appears more reasonable and interpretable compared to the AIC and BIC suggested K=10 clusters. The K=2 clusters show clear differences in key characteristics, providing more meaningful insights into the grouping of accommodations.

As shown in Table 9, the distinct sets of features that characterize each cluster highlight the significance of using a lower number of clusters for better interpretation. The overlap observed in the K=10 clusters indicates that increasing the number of clusters does not necessarily result in better-defined or more interpretable groupings. This emphasizes the importance of evaluating clustering results not just by statistical criteria but also by practical interpretability and clarity.

Overall, the silhouette method's recommendation of K=2 offers a more straightforward and insightful classification of accommodations, facilitating targeted marketing, improved customer experience, and better accommodation recommendations. This approach underscores the need to balance statistical validity with practical usability in clustering analyses.

# 8 Topics Model

Topic modeling is a powerful statistical technique used to uncover abstract themes within large collections of unstructured text. It is particularly useful in natural language processing (NLP) for analyzing vast amounts of textual data, such as articles, reviews, or social media posts. The core idea behind topic modeling is to represent each document as a mixture of topics, each being a word distribution. This approach allows for the automatic extraction of coherent and meaningful topics from the text, providing a high-level understanding of the underlying structure of the data. One of the most commonly used algorithms for topic modeling is Latent Dirichlet Allocation (LDA), which assumes that documents are generated from a fixed set of topics in specific proportions. The Document-Term Matrix (DTM) serves as the input for these algorithms, capturing the frequency of terms within documents and facilitating the identification of key topics.

The applications of topic modeling are diverse and span various fields. In text analysis, it helps identify prevalent themes and trends in news articles, research papers, and social media posts. Content recommendation systems leverage topic modeling to better understand user interests and improve recommendations. In customer feedback analysis, topic modeling reveals common issues and areas for improvement from reviews and feedback. In healthcare, it aids in extracting themes from medical records and patient surveys to enhance services and policies. Specifically, in predicting Airbnb prices, topic modeling analyzes guest reviews and listing descriptions to uncover influential factors such as location, amenities, and overall quality. This information enhances predictive models, improving their accuracy and providing valuable insights into pricing determinants.

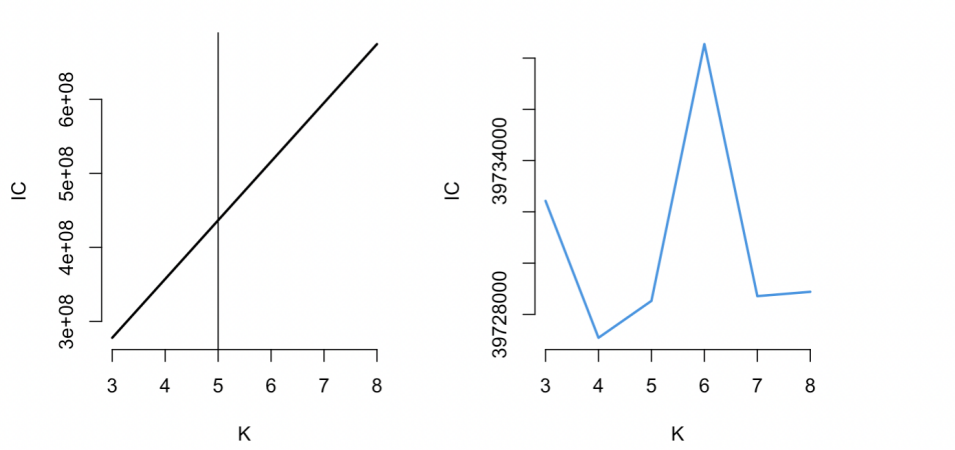
## 8.1 Text Clustering

### 8.1.1 AIC and BIC for Model Selection

In this part, we performed topic modeling on two different text features of Airbnb listings: the listing titles ('name') and the reviews ('description'). The goal was to identify coherent topics within these text datasets that could potentially explain variations in listing prices. We applied K-means clustering to the normalized term frequencies, testing for different numbers of clusters (K) ranging from 3 to 8. To evaluate the quality of the clusters, we used information criteria (AIC and BIC) and plotted the AIC and BIC values for different cluster numbers to identify the optimal number of clusters. The model with the minimum BIC value was selected for interpretation. Finally, we identified the top words associated with each cluster center to interpret the topics and evaluated the cluster sizes to understand the distribution of topics.

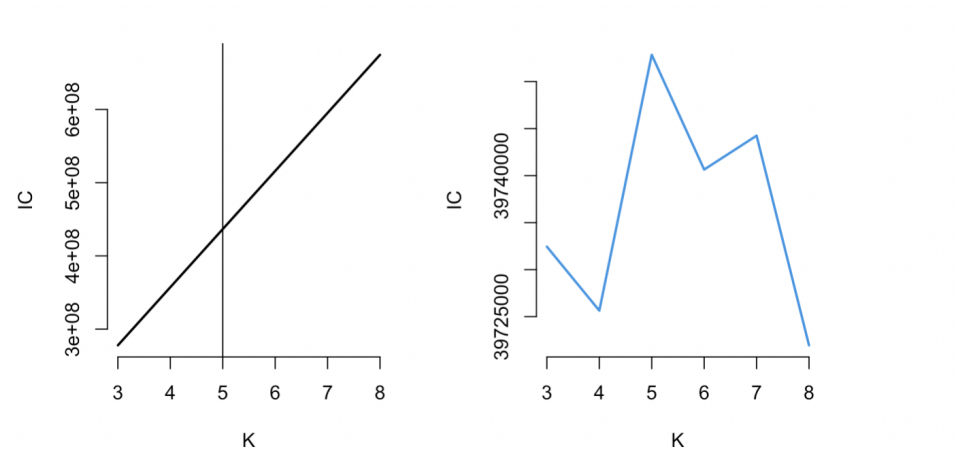
The analysis of the 'name' field involved clustering the listing titles to uncover key topics. The AIC and BIC plots indicated the optimal number of clusters, and the best model was selected with $K = 4$, as indicated by the vertical line in Figure 39. The minimum BIC value was around 2465000000. The top words for each cluster provided insights into the main themes of the listing titles. For example, Cluster 1 included words like "hightnopa," "homeparking," and "larger." Cluster 2 had words like "aeroview," "lush," and "screen," while Cluster 4 included "cozy,"

FIGURE 39. AIC and BIC for Different Number of Clusters on Names of AirBnB




"apt," and "spacious." The distribution of listings across clusters was 1, 7, 16, and 9976 respectively, indicating a significant skew towards one large cluster.

FIGURE 40. AIC and BIC for Different Number of Clusters on Descriptions of AirBnB




The analysis of the 'description' field followed the same steps but showed that the topics extracted from the reviews were less coherent and more dispersed compared to the listing titles. The best model was selected based on the minimum BIC value, with $K = 6$ as indicated by the vertical line in the plot. The minimum BIC value was around 397250000. The top words for each review cluster included a mix of general terms. For example, Cluster 1 included "kitchen," "room," and "restaurants." Cluster 2 had words like "bains," "accueillis," and "anime," while Cluster 3 included "accs," "ainsi," and "avez." The cluster sizes were 1, 7, 9950, 1, 1, 1,

6, and 33 respectively, indicating a highly uneven distribution with one dominant cluster and several very small ones. This distribution suggests that the clustering was less effective in differentiating between topics in the reviews.

Thus, from the AIC and BIC analysis, we are able to conclude the optimal number of clusters for the name and description of Airbnb. The analysis also revealed that the listing titles ('name') are a better feature to study for topic modeling compared to the reviews ('description'). The topics derived from listing titles were more coherent and focused, providing clear insights into the features and selling points of the listings, such as "cozy," "spacious," and "modern." The listing titles are often crafted to highlight key aspects of the property that are directly related to its pricing, making the extracted topics more relevant for predictive modeling. Furthermore, titles are generally more concise and to the point, whereas reviews can be long-winded and contain a lot of extraneous information that dilutes the main themes.

Therefore, focusing on the listing titles for topic modeling in the context of predicting Airbnb prices provides clearer and more actionable insights compared to analyzing the reviews. The more coherent and focused topics derived from the titles are directly relevant to pricing, while the extraneous information in the reviews makes it harder to extract clear themes. This makes the listing titles a more valuable feature for predictive modeling and understanding the factors influencing Airbnb prices.

### 8.1.2 Cluster Interpretation

First, we will analyze each cluster's theme to determine its meaning. Then, we will use the optimal number of clusters, $K = 4$, to examine and interpret the clusters for AirBnB names and descriptions.
First, we look at the analysis of AirBnb names as shown in Table 12.

- Topic 1 The first topic revolves around basic amenities and features of the listings, with frequent mentions of "room," "place," "kitchen," "bathroom," and "bed." This cluster likely reflects listing titles that highlight essential elements of the property, emphasizing the availability of fundamental facilities. Words like "walk," "private," "parking," and "restaurants" suggest that listings often emphasize the convenience and accessibility of nearby services and amenities. Additionally, terms such as "apartment," "living," and "neighborhood" indicate that the titles may also describe the type of property and its immediate surroundings.
- Topic 2 The second topic focuses on the overall living experience and comfort within the property, highlighted by words such as "room," "bedroom," "private," and "apartment." Listing titles in this cluster seem to emphasize the comfort and quality of the living spaces, as indicated by terms like "kitchen," "full," "home," and "located." Additionally, words like "minutes," "available," "area," and "neighborhood" suggest that listings frequently mention the convenience and accessibility of the property, as well as its proximity to key locations and services.
- Topic 3 The third topic centers on the surrounding area and neighborhood of the property, with terms such as "apartment," "bedroom," "one,"

and "room" being prominent. Words like "home," "house," "walk," and "restaurants" suggest that listing titles in this cluster often highlight the property's location and its proximity to parks and other outdoor spaces. Additionally, terms like "new," "kitchen," "floor," and "area" indicate a focus on the property's amenities and its suitability for both short-term and long-term stays, providing guests with a comfortable living environment.

- Topic 4 The fourth topic emphasizes the quality and convenience of the property, with frequent mentions of "room," "away," "bedroom," and "kitchen." Listing titles in this cluster often discuss the overall satisfaction with the property, as indicated by terms like "apartment," "one," "great," and "large." Words such as "street," "park," "blocks," and "living" highlight the property's strategic location and the convenience of its amenities. Additionally, terms like "home," "two," "guests," and "new" suggest that listings often appeal to families or groups looking for well-located, fully-equipped properties with easy access to various services.

TABLE 11. Top 24 Words in Clusters on Names of AirBnB for $K = 4$

| | **Topic 1** | **Topic 2** | **Topic 3** | **Topic 4** |
|---|---|---|---|---|
| 1 | room | room | apartment | room |
| 2 | place | bedroom | bedroom | away |
| 3 | kitchen | private | one | bedroom |
| 4 | bathroom | apartment | room | kitchen |
| 5 | bed | bed | located | apartment |
| 6 | walk | kitchen | home | one |
| 7 | private | full | house | street |
| 8 | parking | home | minutes | walk |
| 9 | restaurants | located | new | great |
| 10 | apartment | available | bed | large |
| 11 | one | restaurants | kitchen | home |
| 12 | living | area | floor | blocks |
| 13 | bedroom | just | area | living |
| 14 | access | away | bathroom | two |
| 15 | neighborhood | neighborhood | neighborhood | just |
| 16 | walking | house | living | east |
| 17 | park | one | park | space |
| 18 | house | access | city | guests |
| 19 | available | new | private | also |
| 20 | street | clean | street | new |
| 21 | minute | place | heart | bed |
| 22 | quiet | queen | two | can |
| 23 | distance | living | train | bathroom |
| 24 | space | large | space | village |

Next, we will look at the theme of each cluster in the AirBnb description, i.e., the reviews people left.

- Topic 1 The first topic predominantly revolves around the amenities and spatial aspects of the property, with frequent mentions of "restaurants,"

TABLE 12. Top 25 Words in Clusters on Descriptions of AirBnB for $K = 4$

| | **Topic 1** | **Topic 2** | **Topic 3** | **Topic 4** |
|---|---|---|---|---|
| 1 | restaurants | room | apartment | apartment |
| 2 | room | bedroom | private | bedroom |
| 3 | place | apartment | house | room |
| 4 | kitchen | kitchen | park | full |
| 5 | space | bed | walk | home |
| 6 | located | one | bedroom | access |
| 7 | bed | place | room | away |
| 8 | living | available | one | great |
| 9 | bathroom | great | close | new |
| 10 | access | comfortable | neighborhood | bed |
| 11 | close | floor | street | location |
| 12 | private | street | block | can |
| 13 | two | bathroom | city | street |
| 14 | walk | walking | bars | one |
| 15 | distance | block | area | kitchen |
| 16 | queen | beautiful | manhattan | two |
| 17 | neighborhood | large | subway | minutes |
| 18 | away | blocks | stay | large |
| 19 | walking | space | located | building |
| 20 | can | just | heart | park |
| 21 | area | minutes | beautiful | quiet |
| 22 | available | perfect | home | best |
| 23 | floor | closet | minutes | hollywood |
| 24 | quiet | two | guests | living |
| 25 | building | city | clean | spacious |

"room," "place," and "kitchen." This cluster likely reflects reviews discussing the convenience and accessibility of dining options near the property. Additionally, words like "space," "located," "living," and "bathroom" indicate that guests often comment on the interior space and layout of the property. Terms such as "access," "close," and "private" suggest a focus on the ease of access and the privacy offered by the property.

- Topic 2 The second topic focuses on the overall living experience within the property, highlighted by words such as "room," "bedroom," "apartment," and "kitchen." Reviews in this cluster seem to emphasize the comfort and quality of the living spaces, as indicated by terms like "great," "comfortable," "floor," and "street." Additionally, words like "available," "large," and "walking" suggest that guests frequently mention the spaciousness and accessibility of the property, as well as the proximity to key locations.
- Topic 3 The third topic centers on the neighborhood and surrounding area of the property, with terms such as "apartment," "private," "house," and "park" being prominent. Words like "walk," "bedroom," "close," and "neighborhood" suggest that reviews in this cluster often highlight the property's location and its proximity to parks and other outdoor spaces. Additionally, terms like "street," "blocks," "city," and "area" indicate a

focus on the property's integration within the urban environment and its accessibility to various amenities.

- Topic 4 The fourth topic emphasizes the quality and convenience of the property, with frequent mentions of "apartment," "room," "full," and "home." Reviews in this cluster often discuss the overall satisfaction with the property, as indicated by terms like "away," "great," "new," and "bed." Words such as "location," "kitchen," "minutes," and "two" highlight the property's strategic location and the convenience of its amenities. Additionally, terms like "large," "building," "park," and "quiet" suggest that guests appreciate the spaciousness, tranquility, and nearby recreational areas.

Each of these topics provides insights into different aspects of Airbnb listing names and descriptions, from the basic amenities and spatial features of the property to the overall living experience and the quality of the neighborhood. This comprehensive analysis helps understand what hosts commonly highlight in their listing titles, offering valuable information for improving property listings and attracting potential guests.

## 8.2 Prediction Through Text Mining

### 8.2.1 High and Low Price Categorization Through Text Analysis

In this part, we investigate the impact of textual descriptions on the pricing of Airbnb listings, focusing on whether specific language patterns can predict if a listing's price is above or below the median. The foundation of this study lies in Natural Language Processing (NLP), where text data is transformed into numerical features for use in statistical models. The primary tools employed are text mining techniques and Linear Discriminant Analysis (LDA), aiming to classify listings based on their price category.

Linear Discriminant Analysis is a statistical method used for classifying objects into categories by finding a linear combination of features that best separate the classes. In this study, LDA is used to classify Airbnb listings into "high price" and "low price" categories based on their descriptions. The high price label is determined by whether the log-transformed price of a listing is above the median log price. By training the LDA model on the DTM features, we aim to identify words or terms indicative of higher or lower prices.

In this part, we treat price as categorical data, using an indicator of 0 and 1 for whether the price of the AirBnB is considered high. To create a binary label for high prices, each listing's log price is compared to the median log price, $113. Listings above the median are labeled as '1' (high price), while those below are labeled as '0' (low price). This binary label serves as the target variable for classification.

The DTM data frame is split into training and testing sets. The LDA model is trained on the training set using the terms as predictors and the high price labels as the grouping factor. After training, the model is used to predict high-price labels for the test set. The predictions are evaluated using a confusion matrix, which compares predicted labels to actual labels, and the model's accuracy is calculated. In this case, the confusion matrix shown in Table 13 indicates that the model

correctly classified 56 out of 90 listings, resulting in an accuracy of approximately 62.2%.

TABLE 13. Confusion Matrix and Model Accuracy

| **Actual \Predicted** | **0** | **1** |
|---|---|---|
| **0** | 23 | 19 |
| **1** | 15 | 33 |

This study demonstrates the application of text mining and statistical modeling in predicting Airbnb listing prices based on descriptions. By converting text data into a Document-Term Matrix and applying Linear Discriminant Analysis, we gain insights into how specific words and phrases are associated with pricing. The accuracy of 62.2% suggests that while text descriptions provide some predictive power, there may be other factors at play. Additionally, sentiment analysis offers a deeper understanding of how the emotional tone of descriptions might influence pricing. This approach highlights the potential of NLP techniques in market analysis and pricing strategies, suggesting further exploration and refinement could enhance predictive accuracy.

### 8.2.2 Sentiment Analysis within Category

Next, we performed sentiment analysis on the listing descriptions to explore if the sentiment expressed in the text correlates with pricing. Sentiment scores for each description are calculated using the AFINN sentiment lexicon, which assigns sentiment values to words. These scores are then added to the dataset for further analysis.

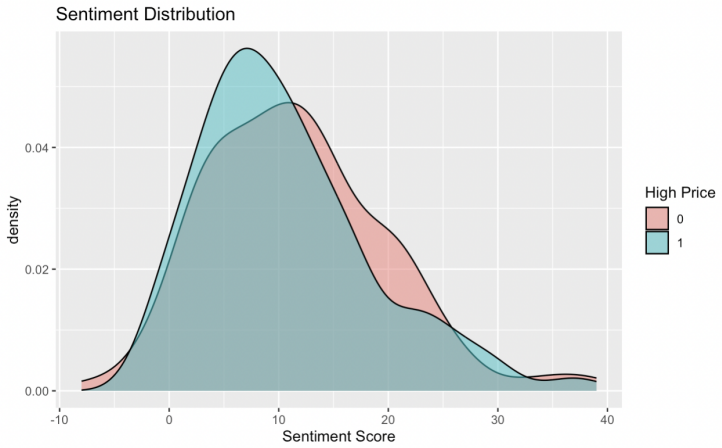



FIGURE 41. Sentiment Distribution for High and Low Price AirBnB

Figure 41 illustrates the distribution of sentiment scores for Airbnb listings, with the data separated into two categories based on the pricing: high price (represented in blue) and low price (represented in red). The sentiment scores, which range from negative to positive values, indicate the emotional tone of the listing descriptions, with higher scores generally reflecting more positive sentiments.

The density plot shows that both high-price and low-price listings tend to have sentiment scores clustered around a similar range, primarily between 0 and 20. This suggests that most listing descriptions, regardless of their price category, have neutral to moderately positive sentiment scores. However, there are notable differences between the two distributions:

- **Peak and Mode**: The high-price listings exhibit a higher peak in the density plot, indicating a larger concentration of listings with sentiment scores around 15-20. This suggests that higher-priced listings are more likely to have descriptions with more positive sentiments compared to lower-priced listings.
- **Distribution Spread**: The distribution of sentiment scores for high-price listings is slightly broader, extending towards higher sentiment scores up to 40. This indicates that some high-priced listings have exceptionally positive descriptions. In contrast, the low-price listings have a more compact distribution, with fewer instances of very high sentiment scores.
- **Overlap**: There is a significant overlap between the sentiment scores of high and low-price listings in the range of 0 to 20. This overlap implies that while there is a general trend for high-priced listings to have more positive descriptions, sentiment alone is not a definitive predictor of listing price.

The sentiment distribution suggests that the emotional tone of listing descriptions does play a role in differentiating high and low-price listings. Listings with more positive descriptions are more likely to be priced higher, possibly because positive sentiment in descriptions can enhance the perceived value of the listing. Descriptions that highlight appealing features, amenities, and overall positive experiences are likely to attract higher bids.

However, the overlap in sentiment scores also indicates that other factors beyond sentiment influence pricing. Features such as location, size, amenities, and market demand likely play significant roles in determining the price. Thus, while sentiment analysis provides valuable insights, it should be considered alongside other variables for a comprehensive understanding of pricing dynamics in Airbnb listings.

Overall, this figure supports the hypothesis that sentiment in descriptions is correlated with price, but it also highlights the complexity of pricing mechanisms in the rental market. Further studies incorporating additional factors could provide a more nuanced understanding of the relationship between sentiment and price.

# 9 CV.LASSO VS RIDGE

## 9.1 Description

LASSO (Least Absolute Shrinkage and Selection Operator) is a linear regression technique that modifies the standard linear regression cost function by adding a penalty term. This penalty is the sum of the absolute values of the coefficients, each multiplied by a tuning parameter, lambda. The LASSO penalty induces sparsity in the model, forcing some coefficients to be exactly zero, thereby performing variable selection.

In contrast, Ridge regression incorporates a different type of penalty, which is the sum of the squared coefficients, also multiplied by a tuning parameter, lambda. Unlike LASSO, the Ridge penalty does not result in sparse coefficients; instead, it shrinks all coefficients towards zero without necessarily making any of them exactly zero. This shrinkage can reduce the variance of the model, thereby enhancing its generalization performance.

The primary distinction between LASSO and Ridge regression lies in the nature of their respective penalties. LASSO employs an L1 penalty (sum of absolute values), while Ridge uses an L2 penalty (sum of squared values). These differences lead to varied effects on coefficient shrinkage and variable selection: LASSO tends to yield sparse solutions by setting some coefficients to zero, making it useful for high-dimensional data with many irrelevant features. Ridge, however, shrinks coefficients towards zero without eliminating any, which is advantageous in situations involving multicollinearity among predictors or when all predictors are potentially relevant.

## 9.2 General Analysis

### 9.2.1 CV LASSO vs Ridge for General analysis

Table 14. Model Comparison

| Model | RMSE | MSE | MAE | R-squared |
|---|---|---|---|---|
| LASSO | 0.5605820 | 0.3142522 | 0.4046589 | 0.6895676 |
| Ridge | 0.5611957 | 0.3149406 | 0.4045983 | 0.6895261 |

The comparison Table 14 presents the performance metrics for the LASSO and Ridge regression models. Both models exhibit very similar RMSE, MSE, and MAE values, indicating comparable predictive accuracy. The R-squared values are also nearly identical, suggesting that both models explain approximately 69% of the variance in the target variable. This analysis shows that there is no significant difference in performance between the LASSO and Ridge regression models based on these metrics. However, focusing on MAE highlights that the average prediction errors are almost the same, emphasizing their similar overall performance in practical terms.

### 9.2.2 Non-zero Coefficients with Different Lambda Values

The analysis in Table 15 highlights distinct patterns in the sparsity of coefficients between LASSO and Ridge regression models. For LASSO, the optimal lambda

TABLE 15. Comparison of Non-zero Coefficients in LASSO and Ridge Regression Models

| **Model** | **Non-zero Coefficients** |
|---|---|
| LASSO (Best Lambda) | 1169 |
| LASSO (1 SE Rule) | 824 |
| Ridge (Best Lambda) | 1587 |
| Ridge (1 SE Rule) | 1587 |

value, which minimizes the cross-validated error, results in a model with 1169 non-zero coefficients. The lambda value chosen by the 1 standard error (1 SE) rule leads to an even sparser model with 824 non-zero coefficients. This demonstrates LASSO's ability to induce sparsity by setting certain coefficients to exactly zero, effectively performing variable selection as part of the optimization process.

In contrast, Ridge regression in the table shows a different behavior. Both the best lambda value and the lambda value selected by the 1 SE rule result in models with 1587 non-zero coefficients. Unlike LASSO, Ridge regression does not enforce exact sparsity; instead, it shrinks the coefficients towards zero, allowing all variables to remain in the model with reduced magnitudes. Consequently, Ridge models maintain a higher degree of complexity compared to LASSO models, as indicated by the consistently larger number of non-zero coefficients for both lambda values. This underscores Ridge's tendency to include all predictors in the model, albeit with smaller weights, rather than performing variable selection.

### 9.2.3 Interpretation of Coefficients

The intercept term shown in Table 16 indicates the baseline value of log price when all predictor variables are zero. Key continuous variables include `accommodates` and `bathrooms`, both having positive coefficients, suggesting that properties accommodating more people or having more bathrooms tend to have higher log prices. Conversely, variables such as `number of reviews` and `beds` have negative coefficients, indicating that more reviews or additional beds are associated with lower log prices. These relationships help in understanding the direct impact of quantitative features on pricing.

For categorical variables, the coefficients represent the effect of different property types compared to a reference category. For instance, properties listed as *Bed & Breakfast* or *Boat* have higher log prices compared to the reference category, as indicated by their positive coefficients. In contrast, *Camper/RV* properties have a lower log price relative to the reference. The zero coefficients for latitude and longitude suggest that these predictors do not significantly contribute to the model and are thus excluded, highlighting their lack of impact on log price.

The coefficients from the lambda.1se Lasso regression model in Table 17 provide insights into the relationship between predictor variables and log_price. The intercept term of approximately 0.244 represents the baseline value of log_price when all predictor variables are zero. For continuous variables, we observe that accommodates (0.224), bathrooms (0.084), and bedrooms (0.169) have positive coefficients,

TABLE 16. Coefficients for the Best Lambda Value

| **Feature** | **Coefficient** |
|---|---|
| Intercept | 0.2434 |
| accommodates | 0.2311 |
| bathrooms | 0.0822 |
| host_response_rate | -0.0114 |
| latitude | . |
| longitude | . |
| number_of_reviews | -0.0614 |
| review_scores_rating | 0.0526 |
| bedrooms | 0.1707 |
| beds | -0.0459 |
| days_since_first_review | 0.0388 |
| days_since_last_review | -0.0189 |
| host_duration | 0.0089 |
| property_typeBed & Breakfast | 0.1798 |
| property_typeBoat | 0.2454 |
| property_typeBoutique_hotel | 0.1667 |
| property_typeBungalow | 0.0655 |
| property_typeCabin | 0.0211 |
| property_typeCamper/RV | -0.1255 |
| property_typeCasa_particular | 0.3949 |

TABLE 17. Coefficients for the Lambda 1se Value

| **Feature** | **Coefficient** |
|---|---|
| Intercept | 0.2444 |
| accommodates | 0.2237 |
| bathrooms | 0.0845 |
| host_response_rate | -0.0096 |
| latitude | . |
| longitude | . |
| number_of_reviews | -0.0543 |
| review_scores_rating | 0.0531 |
| bedrooms | 0.1687 |
| beds | -0.0364 |
| days_since_first_review | 0.0322 |
| days_since_last_review | -0.0127 |
| host_duration | 0.0118 |
| property_typeBed & Breakfast | 0.1313 |
| property_typeBoat | 0.1911 |
| property_typeBoutique_hotel | 0.1044 |
| property_typeBungalow | 0.0095 |
| property_typeCabin | . |
| property_typeCamper/RV | -0.0918 |
| property_typeCasa_particular | . |

indicating that an increase in these features leads to an increase in log_price. In contrast, *host_response_rate* (0.010), *number_of_reviews* (0.054), and *beds* (0.036) have

negative coefficients, suggesting that higher values in these variables correspond to a decrease in log_price. These relationships highlight how property characteristics and host behavior metrics can impact pricing.

For categorical variables, the model shows that certain property types significantly affect log_price. For instance, being a *Bed & Breakfast* (0.131), *Boat* (0.191), or *Boutique hotel* (0.104) has a positive impact on log_price compared to the baseline property type. This implies that these property types command higher prices. Conversely, being a *Camper/RV* (0.092) has a negative impact, indicating lower prices for this type of property. Notably, some coefficients are zero, such as *latitude*, *longitude*, *property_typeCabin*, and *property_typeCasa_particular*, indicating these features do not contribute to the model at this level of regularization.

The lambda.1se model's tendency to set more coefficients to zero helps simplify the model, reducing the risk of over-fitting and enhancing its generalizability by focusing on the most impactful predictors. This characteristic of the Lasso model, particularly with the 1 SE rule, ensures that only the most relevant variables are included, leading to a more interpretable and robust model.

#### 9.2.4 10 Significant coefficients for general analysis

Table 18. Top 10 Features with Coefficients with LASSO

| **Feature** | **Coefficient** |
|---|---|
| neighbourhoodWilmington | 4.507890 |
| zipcode20064 | 2.737377 |
| zipcode90067 | 1.628671 |
| zipcode93552 | -1.522108 |
| zipcode10119.0 | 1.492983 |
| zipcode10704 | 1.386475 |
| neighbourhoodBellevue | 1.366074 |
| zipcode94103.0 | 1.285344 |
| room$_t$*ypeSharedroom* | -1.236834 |
| neighbourhoodSea Cliff | 1.223330 |

Table 18 presents the most influential features identified by the LASSO regression model, along with their corresponding coefficients. These coefficients reflect the strength and direction of the relationship between each feature and the target variable, `log_price`. Positive coefficients indicate a positive association with `log_price`, meaning that an increase in these features is associated with higher prices. Conversely, negative coefficients indicate a negative association, suggesting that an increase in these features is linked to lower prices.

For example, the feature *neighbourhoodWilmington* has the highest positive coefficient (4.507890), indicating that properties in the Wilmington neighborhood are associated with significantly higher log_prices compared to the baseline neighborhood. Similarly, features like *zipcode20064* (2.737377) and *neighbourhoodBellevue* (1.366074) also have strong positive associations with `log_price`. In contrast, *zipcode93552* (-1.522108) and *room_typeSharedroom* (-1.236834) have negative coefficients, indicating that these features are associated with lower `log_price`. This

analysis helps in understanding which features have the most substantial impact on the pricing of properties.

TABLE 19. Top 10 Features with Coefficients with Ridge

| **Feature** | **Coefficient** |
|---|---|
| neighbourhoodWilmington | 4.4395 |
| zipcode20064 | 2.8615 |
| zipcode10119.0 | 1.8012 |
| zipcode93552 | -1.7320 |
| zipcode94103.0 | 1.6665 |
| zipcode10704 | 1.6328 |
| zipcode90067 | 1.5721 |
| zipcode210 | -1.4660 |
| neighbourhoodBellevue | 1.4535 |
| neighbourhoodSea Cliff | 1.4395 |

The Table 19 displays the most influential features identified by the Ridge regression model, along with their corresponding coefficients. These coefficients represent the magnitude and direction of the effect that each predictor variable has on the outcome variable, log_price. Positive coefficients indicate that an increase in the predictor variable is associated with a higher predicted log_price, while negative coefficients indicate the opposite.

For example, *neighbourhoodWilmington* has the highest positive coefficient (4.4395), indicating that properties in the Wilmington neighborhood are associated with significantly higher log prices compared to the baseline neighborhood. Similarly, features like *zipcode20064* (2.8615) and *neighbourhoodBellevue* (1.4535) also show strong positive associations with log prices. Conversely, *zipcode93552* (-1.7320) has a negative coefficient, suggesting that properties in this area are associated with lower log prices.

These coefficients provide valuable insights into the relative importance and direction of influence of each predictor within the model, helping to understand how different factors contribute to property pricing. Positive coefficients, such as those for *neighbourhoodWilmington* and *zipcode20064*, indicate that these areas command higher prices, while negative coefficients, like that for *zipcode93552*, suggest lower prices in those regions.

Due to their different penalty terms, Lasso and Ridge regression produce distinct coefficient estimates. Lasso uses an L1 penalty, which tends to produce sparse models by setting many coefficients to zero, effectively performing variable selection. This makes Lasso particularly useful for identifying the most important predictors in a model. In contrast, Ridge regression uses an L2 penalty, which shrinks all coefficients towards zero but does not eliminate any predictors. This approach is beneficial for addressing multicollinearity by keeping all predictors in the model, albeit with reduced influence.

Therefore, Lasso coefficients are often more interpretable as they highlight key predictors, while Ridge coefficients reflect the overall importance of each predictor

without excluding any. The choice between Lasso and Ridge regression depends on the specific characteristics of the dataset and the goals of the analysis: Lasso is preferred for variable selection and simplifying models, whereas Ridge is preferred for managing multicollinearity and retaining all predictors.

#### 9.2.5 In-sample R2 vs Out-of-sample R2

In-sample R-squared measures how effectively the model fits the data it was trained on, providing insight into its performance within the training dataset. However, it can be overly optimistic because the model was directly optimized to fit this specific data. Out-of-sample R-squared (also known as test or validation R-squared) evaluates the model's performance on new, unseen data, offering a more reliable estimate of its ability to generalize to unseen observations.

TABLE 20. Model Performance Comparison

| Model | In_sample_R2 | Out_sample_R2 |
|---|---|---|
| LASSO | 0.6919482 | 0.6890894 |
| Ridge | 0.6925175 | 0.6884083 |

The close resemblance in performance between LASSO and Ridge, as indicated by their similar in-sample and out-of-sample R-squared values in Table 20, suggests that both models possess comparable predictive capabilities. However, it's important to acknowledge the limitations of in-sample R-squared, notably its inclination towards over-optimism due to over-fitting. This underscores the significance of relying on out-of-sample metrics for a more realistic evaluation of a model's generalization ability. Considering factors beyond mere predictive accuracy, such as model interpretability, computational efficiency, and theoretical preferences, is essential for making a well-informed choice between LASSO and Ridge regression models.

### 9.3 CV.LASSO for Log Price Prediction on Treatment Variables

In our previous analysis, we utilized cross-validation to train and evaluate both Ridge and LASSO models. While Ridge regression offers a stable solution, LASSO presents a simpler and more interpretable model by selecting only the most relevant treatment variables. Given the objectives of our analysis and the characteristics of the treatment variables, it's valuable to investigate the potential advantages of LASSO regression in identifying the most influential treatment variables while mitigating multicollinearity concerns.

#### 9.3.1 Coefficients of LASSO Model with Treatment Variables

The coefficients from the LASSO regression model in Table 21 provide insights into how different treatment variables influence the logtransformed price (`log_price`) of Airbnb listings. Positive coefficients indicate that an increase in the treatment variable is associated with a higher log price. For instance, the number of people a property can accommodate (0.237) and the number of bedrooms (0.173) have positive coefficients, suggesting that larger properties can command higher prices. Similarly, higher review scores (0.051) and longer host durations (0.009) positively

TABLE 21. Coefficients for the best lambda value

| Variable | Coefficient |
|---|---|
| (Intercept) | 0.371457543 |
| accommodates | 0.237067925 |
| bathrooms | 0.082435347 |
| host_response_rate | -0.012273986 |
| latitude | . |
| longitude | . |
| number_of_reviews | -0.067304989 |
| review_scores_rating | 0.050521200 |
| bedrooms | 0.173026800 |
| beds | -0.049523499 |
| days_since_first_review | 0.040893165 |
| days_since_last_review | -0.022518811 |
| host_duration | 0.008761634 |
| property_typeApartment | -0.130830031 |
| property_typeBed & Breakfast | 0.074784033 |
| property_typeBoat | 0.211902709 |
| property_typeBoutique hotel | 0.070544476 |
| property_typeBungalow | -0.032616889 |
| property_typeCabin | -0.025109947 |
| property_typeCamper/RV | -0.298589045 |

impact prices, indicating that well-rated and experienced hosts can charge more. Conversely, negative coefficients indicate that an increase in the treatment variable corresponds to a lower log price. For example, the number of reviews (-0.067) and days since the last review (-0.023) negatively affect the log price, suggesting that older or less recently reviewed listings might be priced lower.

Regarding property types, the model reveals varied impacts on log price. Being a boat (0.212) or a bed & breakfast (0.075) increases the `log_price` compared to the reference property type while being an apartment (-0.131) or a camper/RV (0.299) decreases it. These findings underscore how different types of properties influence pricing strategies on Airbnb. Notably, variables like latitude and longitude have zero coefficients, indicating they do not significantly impact the price and were not selected by the LASSO model. This feature selection characteristic of LASSO helps identify the most influential treatment variables while excluding the less relevant ones, providing a focused understanding of the key factors affecting Airbnb listing prices.

### 9.3.2 Coefficients for Lambda Selected by 1 Standard Error Rule

The coefficients obtained from the LASSO regression model using the lambda value selected by the 1 standard error rule in Table 22 shed light on the impact of each predictor on the log-transformed price. Positive coefficients signify an increase in log price with an increase in the predictor, while negative coefficients imply a decrease.

TABLE 22. Coefficients for the lambda value selected by 1 standard error rule

| **Variable** | **Coefficient** |
|---|---|
| Intercept | 3.619302e-01 |
| accommodates | 2.337900e-01 |
| bathrooms | 8.325909e-02 |
| host_response_rate | -1.124972e-02 |
| latitude | . |
| longitude | . |
| number_of_reviews | -6.365468e-02 |
| review_scores_rating | 5.074904e-02 |
| bedrooms | 1.723221e-01 |
| beds | -4.549106e-02 |
| days_since_first_review | 3.774622e-02 |
| days_since_last_review | -1.947345e-02 |
| host_duration | 1.031600e-02 |
| property_typeApartment | -1.130953e-01 |
| property_typeBed & Breakfast | 7.113918e-02 |
| property_typeBoat | 1.946299e-01 |
| property_typeBoutique hotel | 6.321633e-02 |
| property_typeBungalow | -7.532084e-03 |
| property_typeCabin | -8.339538e-05 |
| property_typeCamper/RV | -2.673303e-01 |

The coefficients for accommodation-related variables indicate that the number of *accommodates* (0.2337900), *bathrooms* (0.08325909), and *bedrooms* (0.1723221) positively influence the log price, suggesting that larger accommodations tend to command higher prices. The *host_response_rate* coefficient is slightly negative (-0.01124972), indicating a minor decrease in log price with higher response rates. Furthermore, negative coefficients for the *number_of_reviews* (-0.06365468) and *days_since_last_review* (-0.01947345) suggest that more recent reviews and fewer reviews might lead to lower prices. Regarding property types, certain categories like *Boats* (0.1946299) and *Bed & Breakfasts* (0.07113918) increase the log price, while others like *Apartments* (-0.1130953) and *CampersRVs* (-0.2673303) decrease it. This analysis provides insights into how different treatment variables influence pricing strategies for Airbnb listings.

The distinction between coefficients obtained with *coefficients_best_lambda_treatment* and *coefficients_1se_treatment* lies in the choice of lambda values within the LASSO regularization framework. *coefficients_best_lambda_treatment* corresponds to coefficients derived using the lambda value that minimizes cross-validated error (lambda.min). This selection prioritizes optimal predictive performance on the training data and often results in more non-zero coefficients. On the other hand, *coefficients_1se_treatment* is obtained from the lambda value within one standard error of the minimum cross-validated error (lambda.1se). This choice leads to a more conservative model with fewer non-zero coefficients, aimed at striking a balance between simplicity and prediction accuracy. While sacrificing a small amount of accuracy, this approach

enhances generalizability and interpretability, facilitating a clearer understanding of the model's behavior.

### 9.3.3 NAIVE LASSO Model

TABLE 23. Performance of the Naive LASSO Model

| Model | In-sample R-squared | Out-of-sample R-squared |
|---|---|---|
| Naive LASSO Model | 0.6969931 | 0.6670235 |

The Naive LASSO Model in Table 23 achieves an in-sample R-squared value of 0.6969931 and an out-of-sample R-squared value of 0.6670235.

The in-sample R-squared (*in_sample_r2_treatment*) quantifies how well the model fits the training data, indicating the proportion of variance in the response variable explained by the predictors. Here, the in-sample R-squared suggests that the Naive LASSO model explains approximately 69.7% of the variance in the response variable within the training dataset.

The out-of-sample R-squared (*Out_sample_R2*) assesses the model's performance on new, unseen data, providing insight into its generalization capability. With an out-of-sample R-squared value of about 66.7%, the Naive LASSO model demonstrates robust predictive performance when applied to unseen data, indicating its ability to capture significant relationships between predictors and the response variable.

## 9.4 Summary

In summary, we explored various aspects of LASSO regression, including its regularization techniques, coefficient interpretation, and model performance evaluation. We discussed how LASSO's penalty term induces sparsity in coefficient estimates, facilitating variable selection and model simplification. Through comparisons with Ridge regression, we highlighted LASSO's ability to set some coefficients to zero, thus providing a more interpretable model with fewer predictors.

Additionally, we examined the impact of different lambda values on coefficient estimates, emphasizing the trade-off between model complexity and predictive accuracy. The selection of lambda values, such as the best lambda and the lambda chosen by the 1 standard error rule, affects the number of nonzero coefficients and the overall model performance.

Furthermore, we evaluated the performance of the Naive LASSO Model with treatment variables, considering both insample and out-of-sample R-squared values. The model demonstrated strong performance in capturing the underlying relationships between predictors and the response variable, both within the training data and when applied to new, unseen data.

Overall, LASSO regression offers a powerful tool for feature selection, model simplification, and predictive modeling, making it valuable for various applications in data analysis.

# 10 CART, Random Forest, and Gradient Boosting

## 10.1 CART Model

### 10.1.1 Decision Tree

Figure 42 illustrates the decision tree structure using all the features in the dataset. The original decision tree has an RMSE of 0.4855 and a $R^2$ of 0.5469, as seen in Table 24. The close values of in-sample (IS) $R^2$ and out-of-sample (OOS) $R^2$ suggest that the decision tree model does not suffer from over-fitting. Both IS and OOS $R^2$ values are slightly over 50%. Since $R^2$ indicates the proportion of the variance in the dependent variable that is predictable from the independent variables, a $R^2$ of 0.55 means that approximately 55% of the variance in the dependent variable can be explained by the model. While this indicates that the model captures a significant portion of the variance, it also suggests that nearly half of the variance is unexplained, indicating room for improvement.

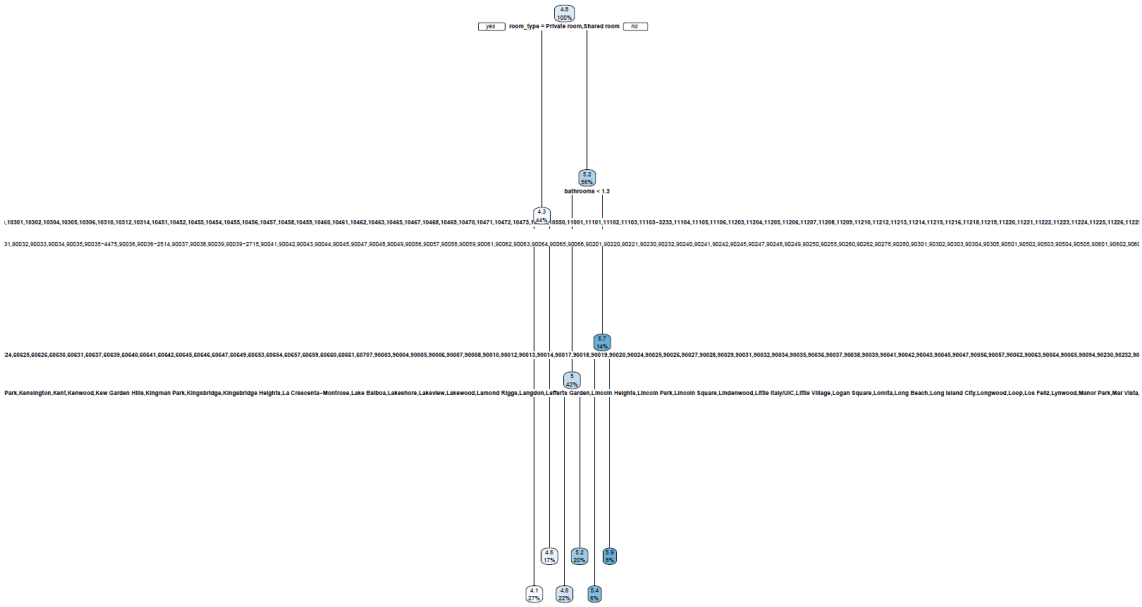



FIGURE 42. Decision Tree Structure

| | Initial Decision Tree | Pruned |
|---|---|---|
| In-sample $R^2$ | 0.5568 | 0.5568 |
| Out-of-sample $R^2$ | 0.5469 | 0.5469 |
| RMSE | 0.4855 | 0.4855 |

TABLE 24. Decision Tree Results

### 10.1.2 Feature Importance

From the feature importance plot in Figure 43, it is evident that `zipcode`, `bedrooms`, `accommodates`, `neighbourhood`, `room_type`, `bathrooms`, `beds`, `city`, `longitude`, and `latitude` are the most significant splitting criteria. An importance score of over 0.8 for `zipcode` suggests that geographical location plays a crucial role in predicting `log_price`. The other variables are also relevant but to a lesser extent compared to `zipcode`. This indicates that Airbnb house prices are primarily determined by location and house conditions, with `zipcode` being the most influential factor.

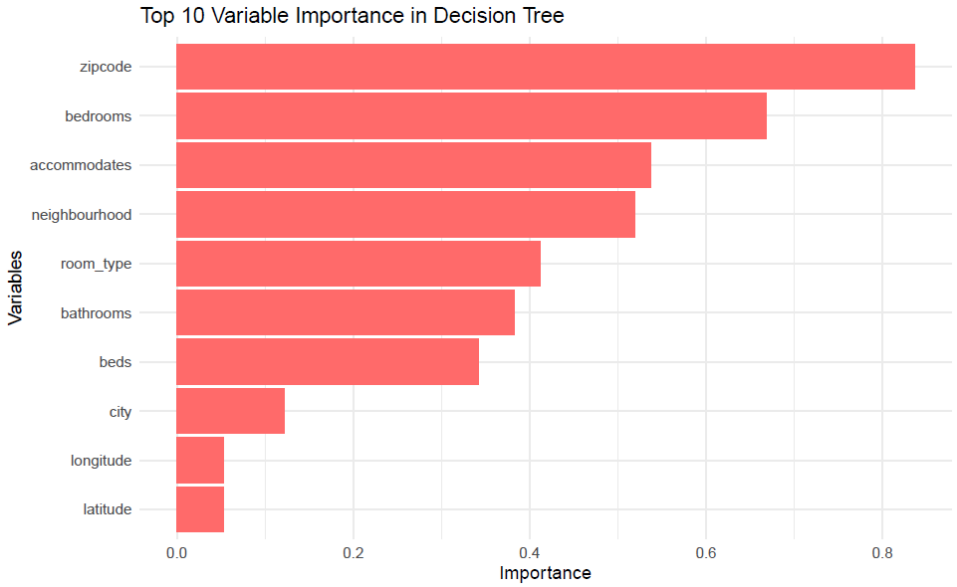



FIGURE 43. Feature Importance of Decision Tree

### 10.1.3 Complexity Parameter (CP) Plot

The CP plot in Figure 44 displays the cross-validation error versus the complexity parameter (cp). The horizontal line represents the minimum cross-validation error plus one standard error (1-SE rule). The plot indicates that the error stabilizes and does not significantly decrease after a certain point of complexity.

After pruning the decision tree using cross-validation, the RMSE and $R^2$ values remain unchanged, indicating that the decision tree does not undergo any modifications. The identical $R^2$ and RMSE values before and after pruning suggest that the tree was already optimally pruned at the chosen CP value. This implies that further pruning would not simplify the model without losing predictive power. The decision tree captures the essential patterns in the data with the selected variables, and additional complexity does not contribute to model improvement.

To further enhance the model's performance, the next step constructs a random forest model comprising 250 single decision trees. This ensemble method aims to

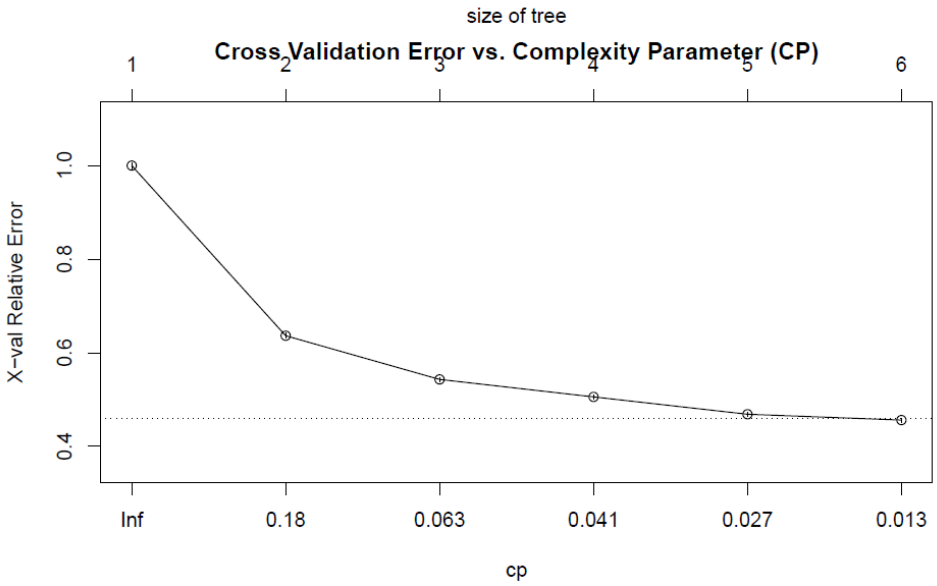



FIGURE 44. CP Plot of Decision Tree

improve predictive accuracy and robustness by averaging the results of multiple decision trees. Random forests reduce overfitting and improve generalization by combining the predictions of many individual trees, each trained on a random subset of the data and features.

## 10.2 Random Forest

### 10.2.1 Random Forest vs. Decision Tree

Unlike a single decision tree, the results of a random forest cannot be easily visualized in a tree structure, as it usually comprises hundreds of trees. Instead, we can plot a trend line indicating the model error as the number of trees in the model increases, as in Figure 45. From the plot, we can observe that the model error decreases steadily as the number of trees increases. However, after reaching approximately 50 trees, the reduction in error becomes less significant, indicating diminishing returns from adding more trees.

The in-sample (IS) $R^2$ of the random forest model is 0.867, and the out-of-sample (OOS) $R^2$ is 0.713, as seen in Table 25. Both IS and OOS $R^2$ values are higher than those of the single decision tree. However, the larger difference between IS $R^2$ and OOS $R^2$ in the random forest model, with IS $R^2$ being much higher, suggests that the random forest model might suffer from overfitting. Despite this potential overfitting, a random forest model is still preferred over a single decision tree due to its better out-of-sample performance.

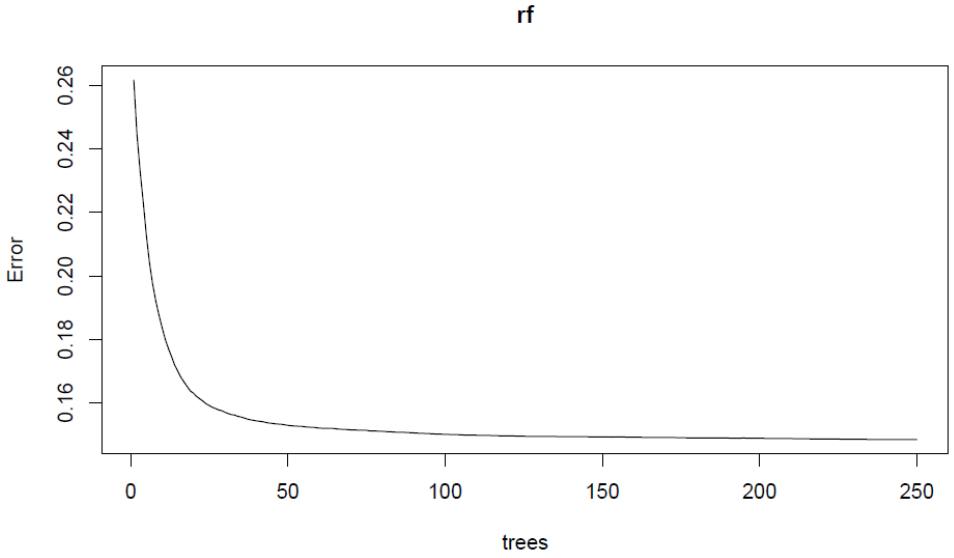



FIGURE 45. Model Error with Number of Trees Included

| | Random Forest |
|---|---|
| In-sample $R^2$ | 0.8617 |
| Out-of-sample $R^2$ | 0.7116 |

TABLE 25. Random Forest Results

### 10.2.2 Feature Importance

From the feature importance plot in Figure 46, we can see that more features are deemed important in the random forest than in the decision tree. The top 10 important features are `room_type`, `longitude`, `number_of_reviews`, `latitude`, `bathrooms`, `host_duration`, `accommodates`, `property_type`, `city`, and `review_scores_rating`.

Generally, features that are important in a single decision tree are also important in the random forest. `zipcode` and `neighborhood`, which are considered very important in the decision tree, do not fit in a random forest model because they cannot support categorical variables that have more than 53 categories. This issue can be fixed using more advanced techniques in the XGBoost model.

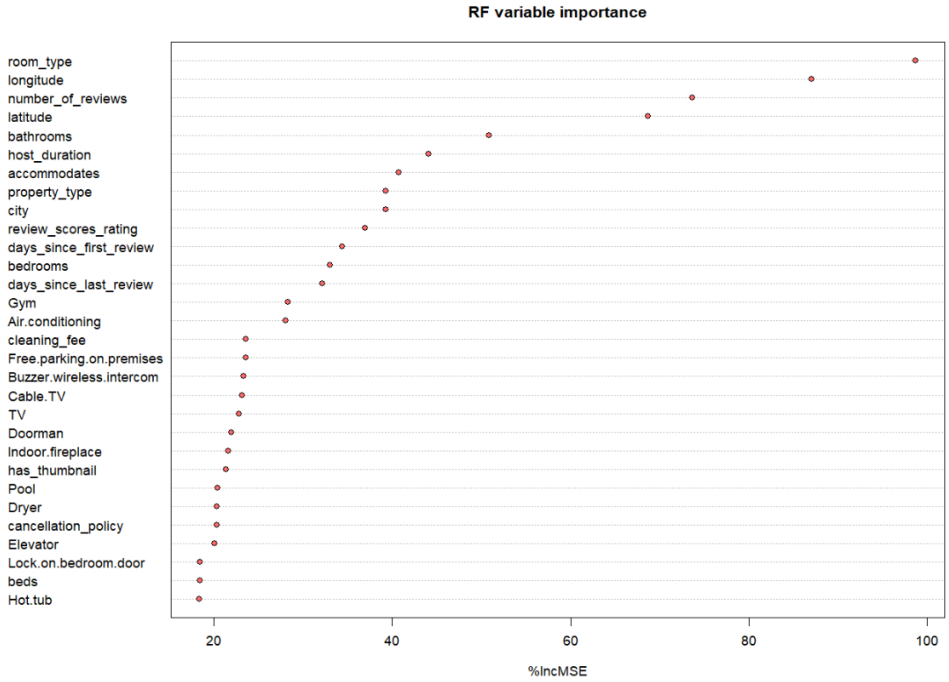



FIGURE 46. Feature Importance of Random Forest

### 10.2.3 Actual vs. Predicted Values

From the scatter plot of actual vs. predicted values of the random forest in Figure 47, we can see that the points are generally distributed uniformly around the reference line. The residual plot shows that the model tends to make higher errors in predicting prices in the middle range. This could be due to the fact that house prices are easier to predict when they are extremely low or high.

Despite the random forest outperforming a single decision tree, a drawback is that it requires significantly more computational resources in terms of both space and time to construct. To address these issues, the next step involves using a more advanced algorithm that has gained popularity due to its speed and performance.

The next step implements Gradient Boosting, an advanced ensemble learning algorithm. Gradient Boosting builds models sequentially, each new model correcting errors made by previous models. This approach can significantly improve model accuracy and is well-suited for handling complex datasets. By using Gradient Boosting, we aim to achieve better predictive accuracy while maintaining reasonable computational efficiency. This advanced technique will help address the limitations observed in the random forest model and further enhance the performance of our predictive model.

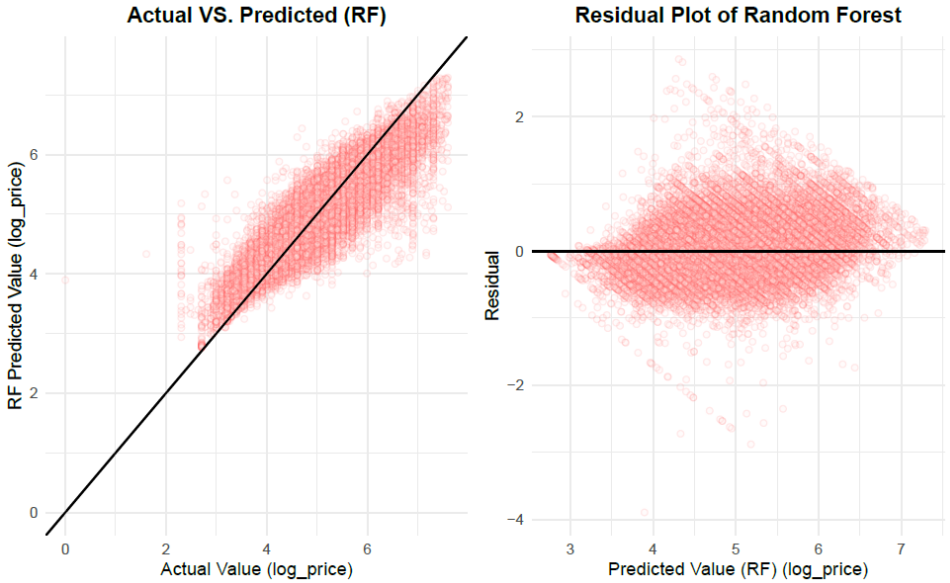



FIGURE 47. Scatter Plots of Random Forest

## 10.3 Gradient Boosting

### 10.3.1 Gradient Boosting vs. Random Forest

Unlike random forest, which restricts the number of categories for a categorical variable, XGBoost does not have such a restriction. Therefore, we are able to include `zipcode` and `neighborhood` into the model. The in-sample (IS) $R^2$ using XGBoost is 0.774, and the out-of-sample (OOS) $R^2$ is 0.722, as seen in Table 26. XGBoost shows a reduced over-fitting problem compared to the random forest model, as indicated by the closer IS and OOS $R^2$ values. Additionally, XGBoost achieves better overall performance, which can be attributed to its advanced boosting mechanism that builds trees sequentially, each one correcting the errors of the previous ones. This allows XGBoost to handle complex patterns in the data more effectively than the random forest.

### 10.3.2 Feature Importance

The most important features in splitting are generally consistent between XGBoost and the random forest. `room_type` is identified as the most important feature in

| | XGBoost |
|---|---|
| In-sample $R^2$ | 0.774 |
| Out-of-sample $R^2$ | 0.7217 |

TABLE 26. XGBoost Results

both models, as seen in Figure 48. Other significant features include `accommodates`, `bedrooms`, `longitude`, `room_typeShared room`, `latitude`, `bathrooms`, `number_of_reviews`, `review_scores_rating`, and `city`.

In XGBoost, the importance score is derived from the feature's contribution to the model's predictive accuracy. It measures the gain, which is the improvement in accuracy brought by a feature to the branches it is involved in. Higher gain indicates more important features. For instance, `room_type` has the highest gain, meaning it significantly improves the model's predictions when used for splitting.

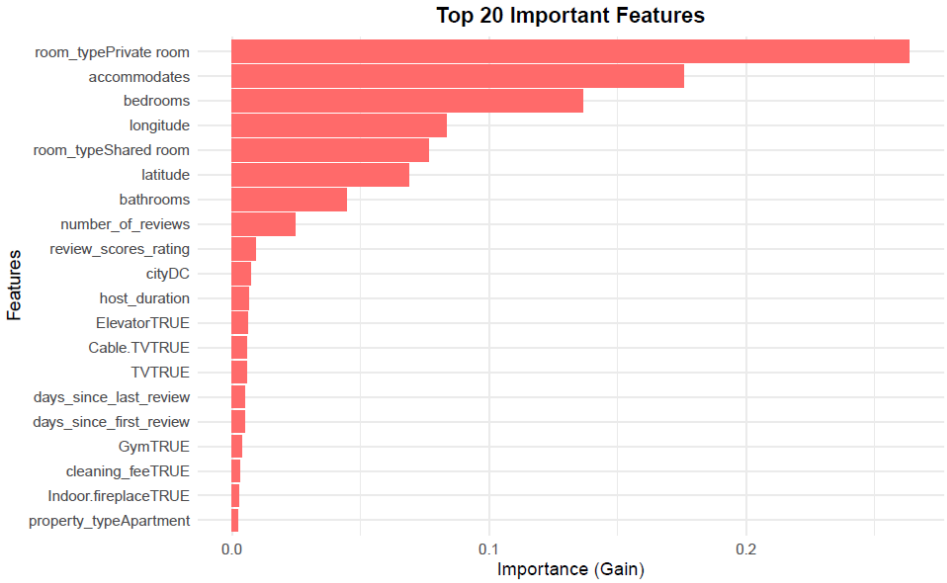



FIGURE 48. Feature Importance of Gradient Boosting

### 10.3.3 Actual vs. Predicted Values

From the actual vs. predicted values scatter plot and the residual plot in Figure 49, it is evident that the issues observed in the random forest model, such as higher errors in the middle range prices, are mitigated to some extent with XGBoost. The scatter plot shows a more uniform distribution around the reference line, indicating better alignment between actual and predicted values.

The residual plot for XGBoost also demonstrates a more uniform distribution of errors, with residuals centered more closely around zero. This indicates that the model's predictions are more accurate across different price ranges. Additionally,

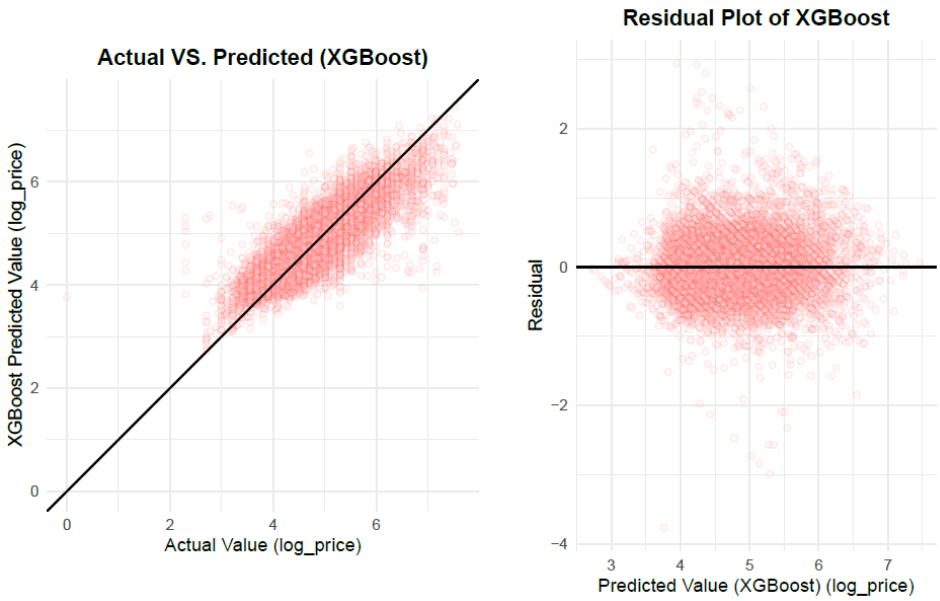



FIGURE 49. Scatter Plots of Gradient Boosting

the residual range has shrunk compared to the random forest model, suggesting that XGBoost produces fewer extreme prediction errors.

Overall, XGBoost provides a significant improvement in predictive performance and error distribution, making it a more reliable model for predicting Airbnb house prices.

## 11 Next Steps

Based on the extensive analysis conducted in this paper, there are several avenues for future research and exploration:

- **User Behavior Analysis:** Utilizing network graphs and Graph Neural Networks (GNNs) in the Airbnb case can reveal patterns in user behavior by representing users, listings, and reviews as interconnected nodes. This approach helps identify groups of similar users and influential listings. GNNs can learn embeddings from these graphs to predict booking likelihood and review sentiments, providing insights for optimizing listings, pricing, and marketing strategies. This method leverages the relational nature of Airbnb data to enhance predictive accuracy and inform better business decisions.

- **Using Review Texts for Rating Prediction:** Many deep learning models can predict ratings from review texts. In addition to leveraging LLMs like BERT and GPT2, investigating alternative approaches such as LSTM and CNNs could further enhance predictive capabilities. Through the evaluation and integration of these diverse methodologies, a more comprehensive understanding of text-based rating prediction can be achieved.

- **Multimodal Modeling by Including Room Images:** Multimodal modeling incorporating images along with other variables can enhance predictive accuracy by capturing both visual and contextual information. Techniques such as deep learning architectures can be employed to process images alongside traditional machine learning algorithms that handle categorical data. This combined approach enables the model to learn complex relationships between visual features and property attributes, improving predictions for outcomes such as pricing.

- **Incoporating Dimensionality Reduction Methods in the Model**: In addition to the results of PCA, t-distributed Stochastic Neighbor Embedding (t-SNE) can also be employed to reduce the dimensionality of the data and enhance model performance.

- **External Out-of-Sample Data Testing:** It involves evaluating the model's performance on data that it has not been exposed to during training or validation. This process helps assess the generalization ability of the model and its effectiveness in making predictions on unseen data from the real world.

- **Cross Validation:** Due to memory space reasons, this report did not conduct cross-validation for random forest and XGBoost. It would be better if cross-validation could be applied to ensure the robustness of the model seen above.

# 12 Appendix

## 1. EDA

```
knitr::opts_chunk$set(echo = TRUE)
library(dplyr)
library(ggplot2)
library(caret)
library(cluster)
library(factoextra)
library(corrplot)
library(GGally)
library(mice)
library(tidyverse)
library(corrplot)
library(reshape2)  # To reshape data
library(ggcorrplot)
library(gridExtra)
library(stringr)
library(mltools)
library(data.table)
library(patchwork)
data <- read.csv("Airbnb_Data.csv")
cat("\nSummary statistics for numerical attributes:\n")
data %>%
  select_if(is.numeric) %>%
  summary()
data$room_type <- as.factor(data$room_type)
data$property_type <- as.factor(data$property_type)
data$bed_type <- as.factor(data$bed_type)
data$cancellation_policy <- as.factor(data$cancellation_policy)
data$city <- as.factor(data$city)
#data$amenities <- as.factor(data$amenities)
#data$description <- as.factor(data$description)
#data$first_review <- as.factor(data$first_review)
#data$host_has_profile_pic <- as.factor(data$host_has_profile_pic)
#data$first_review <- as.factor(data$first_review)
#data$host_identity_verified <- as.factor(data$host_identity_verified)
#data$host_response_rate <- as.factor(data$host_response_rate)
#data$host_since <- as.factor(data$host_since)
data$instant_bookable <- as.factor(data$instant_bookable)
#data$last_review <- as.factor(data$last_review)
#data$neighbourhood <- as.factor(data$neighbourhood)
#data$thumbnail_url <- as.factor(data$thumbnail_url)
#data$zipcode <- as.factor(data$zipcode)
cat("\nSummary statistics for categorical attributes:\n")
data %>%
  select_if(is.factor) %>%
  summary()
count_missing <- function(x) {
  sum(is.na(x) | x == "")
}
# Create a summary of missing values (NA and empty strings) for each column
missing_values_summary <- data %>%
```

```r
  summarise_all(count_missing) %>%
  gather(key = "variable", value = "missing_count") %>%
  arrange(desc(missing_count))

print(missing_values_summary)
# Convert host_response_rate to numeric by removing the '%' sign
data$host_response_rate <- as.numeric(gsub("%", "", data$host_response_rate))

# Replace missing values with the median for numeric columns
data$bathrooms[is.na(data$bathrooms)] <- median(data$bathrooms, na.rm = TRUE)
data$bedrooms[is.na(data$bedrooms)] <- median(data$bedrooms, na.rm = TRUE)
data$beds[is.na(data$beds)] <- median(data$beds, na.rm = TRUE)
data$review_scores_rating[is.na(data$review_scores_rating)] <- median(data$review_scores_rating,
                                                                      na.rm = TRUE)
data$host_response_rate[is.na(data$host_response_rate)] <- median(data$host_response_rate,
                                                                  na.rm = TRUE)

# Replace missing values with specific values for categorical columns
data$host_has_profile_pic[is.na(data$host_has_profile_pic) | data$host_has_profile_pic == ''] <- 'f'
data$host_identity_verified[is.na(data$host_identity_verified)
                            | data$host_identity_verified == ''] <- 'f'
data$first_review[is.na(data$first_review) | data$first_review == ''] <- 'Unknown'
data$host_since[is.na(data$host_since) | data$host_since == ''] <- 'Unknown'
data$last_review[is.na(data$last_review) | data$last_review == ''] <- 'Unknown'
data$neighbourhood[is.na(data$neighbourhood) | data$neighbourhood == ''] <- 'Unknown'
data$thumbnail_url[is.na(data$thumbnail_url) | data$thumbnail_url == ''] <- 'No URL'
data$zipcode[is.na(data$zipcode) | data$zipcode == ''] <- 'Unknown'


# Convert categorical columns to factors
# data$thumbnail_url <- factor(data$thumbnail_url)
data$zipcode <- factor(data$zipcode)
data$host_has_profile_pic <- factor(data$host_has_profile_pic)
data$first_review <- factor(data$first_review)
data$last_review <- factor(data$last_review)
data$host_since <- factor(data$host_since)
data$host_identity_verified <- factor(data$host_identity_verified)
data$neighbourhood <- factor(data$neighbourhood)


# Summarize key statistics for numerical attributes after handling missing values
cat("Updated summary statistics for numerical attributes:\n")
data %>%
  select_if(is.numeric) %>%
  summary()

# Summarize key statistics for categorical attributes after handling missing values
cat("\nUpdated summary statistics for categorical attributes:\n")
data %>%
  select_if(is.factor) %>%
  summary()
# Calculate price from log_price assuming log_price is the natural log of price
df <- data.frame(price = exp(data$log_price), log_price = data$log_price)
```

```r
# Plot for 'price' using indianred1 color
price_plot <- ggplot(df, aes(x = price)) +
  geom_histogram(bins = 30, fill = "indianred1", color = "indianred1", alpha = 0.7) +
  geom_density(aes(y = ..density.. * 10), color = "indianred1", linewidth = 1) +
  theme_minimal() +
  labs(title = "Distribution of Prices",
       x = "Price",
       y = "Frequency") +
  theme(plot.title = element_text(hjust = 0.5),
        axis.text.x = element_text(angle = 45, hjust = 1),
        panel.grid.major = element_blank(),  # Removes major grid lines
        panel.grid.minor = element_blank())  # Removes minor grid lines

# Plot for 'log_price' using indianred1 color
log_price_plot <- ggplot(df, aes(x = log_price)) +
  geom_histogram(bins = 30, fill = "indianred1", color = "indianred1", alpha = 0.7) +
  geom_density(aes(y = ..density.. * 10), color = "indianred1", linewidth = 1) +
  theme_minimal() +
  labs(title = "Distribution of Log Prices",
       x = "Log Price",
       y = "Frequency") +
  theme(plot.title = element_text(hjust = 0.5),
        axis.text.x = element_text(angle = 45, hjust = 1),
        panel.grid.major = element_blank(),  # Removes major grid lines
        panel.grid.minor = element_blank())  # Removes minor grid lines

# Combine the plots using patchwork
combined_plot <- price_plot + log_price_plot +
  plot_layout(ncol = 2)

# Print the combined plot
print(combined_plot)
ggsave("price_plot.png", plot = combined_plot, width = 10, height = 6, dpi = 300)
box_plot <- ggplot(df, aes(x = log_price)) +
  geom_boxplot(fill = "indianred1", color = "black", alpha = 0.7) +
  theme_minimal() +
  labs(title = "Box Plot of Log Prices",
       x = "Log Price") +
  theme(plot.title = element_text(hjust = 0.5))
print(box_plot)
ggsave("box_plot.png", plot = box_plot, width = 10, height = 6, dpi = 300)
numerical_features <- data[, c('log_price', 'accommodates', 'bathrooms', 'host_response_rate',
                               'number_of_reviews', 'review_scores_rating', 'bedrooms', 'beds',
                               'latitude')]

correlation_matrix <- cor(numerical_features, use = "complete.obs")

corr_plot <- ggcorrplot(correlation_matrix,
           method = "circle",
           type = "lower",
           lab = TRUE,
           ggtheme = ggplot2::theme_minimal(),
           colors = c("#6D9EC1", "white", "indianred1"),
```

```
           lab_size = 3,
           outline.color = "gray")
print(corr_plot)
ggsave("corr_plot.png", plot = corr_plot, width = 10, height = 6, dpi = 300)
scatter_plots <- list(
  ggplot(data, aes(x = accommodates, y = log_price)) +
    geom_point(color = "#CD5C5C", alpha = 0.6) +  # IndianRed
    theme_minimal() +
    ggtitle('Log Price vs Accommodates') +
    theme(plot.title = element_text(hjust = 0.5)),

  ggplot(data, aes(x = bathrooms, y = log_price)) +
    geom_point(color = "#F08080", alpha = 0.6) +  # LightCoral
    theme_minimal() +
    ggtitle('Log Price vs Bathrooms') +
    theme(plot.title = element_text(hjust = 0.5)),

  ggplot(data, aes(x = bedrooms, y = log_price)) +
    geom_point(color = "#FA8072", alpha = 0.6) +  # Salmon
    theme_minimal() +
    ggtitle('Log Price vs Bedrooms') +
    theme(plot.title = element_text(hjust = 0.5)),

  ggplot(data, aes(x = beds, y = log_price)) +
    geom_point(color = "#E9967A", alpha = 0.6) +  # DarkSalmon
    theme_minimal() +
    scale_x_continuous(breaks = seq(min(data$beds, na.rm = TRUE), max(data$beds, na.rm = TRUE),
                                    by = 1)) +
    ggtitle('Log Price vs Beds') +
    theme(plot.title = element_text(hjust = 0.5))
)

# Arrange all plots in a 2x2 grid layout
two_two_plot <- grid.arrange(grobs = scatter_plots, ncol = 2)
print(two_two_plot)
ggsave("scatter_plot.png", plot = two_two_plot, width = 10, height = 8, dpi = 300)
city <- ggplot(data, aes(x = city, y = log_price)) +
  geom_boxplot(fill = "indianred1", color = "black") +
  theme_minimal() +
  labs(
       x = "City",
       y = "Log Price")
print(city)
ggsave("city_plot.png", plot = city, width = 10, height = 6, dpi = 300)
nyc_data <- subset(data, city == "NYC")

# Identify the top 10 neighborhoods based on frequency
top_neighbourhoods <- names(sort(table(nyc_data$neighbourhood), decreasing = TRUE)[1:10])

# Filter NYC data to only include the top 10 neighborhoods
nyc_top_neighbourhoods_data <- subset(nyc_data, neighbourhood %in% top_neighbourhoods)

# Log Price Distribution Across Top 10 Neighborhoods in NYC
```

```r
nyc <- ggplot(nyc_top_neighbourhoods_data, aes(x = neighbourhood, y = log_price)) +
  geom_boxplot(fill = "indianred1", color = "black") +
  theme_minimal() +
  labs(
       x = "Neighborhood",
       y = "Log Price") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
print(nyc)
ggsave("nyc_plot.png", plot = nyc, width = 10, height = 6, dpi = 300)
property <- ggplot(data, aes(y = reorder(property_type, -table(property_type)[property_type]))) +
  geom_bar(fill = "indianred1") +
  theme_minimal() +
  labs(
       x = "Count",
       y = "Property Type")
print(property)
ggsave("property_plot.png", plot = property, width = 10, height = 8, dpi = 300)
room <- ggplot(data, aes(x = reorder(room_type, -table(room_type)[room_type]))) +
  geom_bar(fill = "indianred1") +
  theme_minimal() +
  labs(
       x = "Room Type",
       y = "Count")
print(room)
ggsave("room_plot.png", plot = room, width = 10, height = 8, dpi = 300)
property_type_proportion <- data %>%
  group_by(city, property_type) %>%
  summarize(count = n(), .groups = 'drop') %>%
  group_by(city) %>%
  mutate(proportion = count / sum(count)) %>%
  filter(count > 0)  # Exclude property types with a count of zero

# Plot the proportions
ggplot(property_type_proportion, aes(y = reorder(property_type, -count), x = proportion
                                     , fill = city)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  labs(title = "Popularity of Property Types by City",
       x = "Proportion",
       y = "Property Type") +
  guides(fill = guide_legend(title = "City"))
# Calculate proportions of room types by city
room_type_proportion <- data %>%
  group_by(city, room_type) %>%
  summarize(count = n()) %>%
  group_by(city) %>%
  mutate(proportion = count / sum(count))

city_colors <- c("indianred1", "#FFB6C1", "#87CEEB", "#98FB98", "#FFD700", "#DA70D6")

# Plot the proportions
city_portion <- ggplot(room_type_proportion, aes(x = reorder(room_type, -count), y = proportion,
                                                 fill = city)) +
```

```r
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  labs(x = "Room Type",
       y = "Proportion") +
  scale_fill_manual(values = city_colors) +
  guides(fill = guide_legend(title = "City"))
print(city_portion)
ggsave("city_room_portion_plot.png", plot = city_portion, width = 10, height = 6, dpi = 300)
property_avg <- data %>%
  group_by(property_type) %>%
  summarize(mean_log_price = mean(log_price, na.rm = TRUE)) %>%
  arrange(desc(mean_log_price))

property <- ggplot(property_avg, aes(y = reorder(property_type, mean_log_price),
                                     x = mean_log_price)) +
  geom_bar(stat = "identity", fill = "indianred1") +
  theme_minimal() +
  labs(x = "Average Log Price",
       y = "Property Type")
print(property)
ggsave("average_price_plot.png", plot = property, width = 10, height = 10, dpi = 300)
room_avg <- data %>%
  group_by(room_type) %>%
  summarize(mean_log_price = mean(log_price, na.rm = TRUE)) %>%
  arrange(desc(mean_log_price))

room <- ggplot(room_avg, aes(x = reorder(room_type, mean_log_price), y = mean_log_price)) +
  geom_bar(stat = "identity", fill = "indianred1") +
  theme_minimal() +
  labs(x = "Room Type",
       y = "Average Log Price")
print(room)
ggsave("average_price_room_plot.png", plot = room, width = 6, height = 6, dpi = 300)
city_colors <- c("indianred1", "#FFB6C1", "#87CEEB", "#98FB98", "#FFD700", "#DA70D6")

# Create the bar plot with city-specific colors
city_count <- ggplot(data, aes(y = reorder(city, -table(city)[city]), fill = city)) +
  geom_bar() +
  theme_minimal() +
  labs(x = "Count",
       y = "City") +
  scale_fill_manual(values = city_colors)

print(city_count)
ggsave("city_count_plot.png", plot = city_count, width = 10, height = 6, dpi = 300)
city_colors <- c("Boston" = "indianred1",
                 "Chicago" = "#FFB6C1",
                 "DC" = "#87CEEB",
                 "LA" = "#98FB98",
                 "NYC" = "#FFD700",
                 "SF" = "#DA70D6")

# Analyze the distribution of listings in neighborhoods for the top cities
```

```r
top_cities <- data %>%
  count(city, sort = TRUE) %>%
  top_n(6, n) %>%
  pull(city)

# Create a list to hold the plots
plots <- list()

# Loop through each city and create a plot
for (i in 1:length(top_cities)) {
  current_city <- top_cities[i]
  city_data <- data %>%
    filter(city == current_city)

  # Select the top 10 neighborhoods for each city
  top_neighbourhoods <- city_data %>%
    count(neighbourhood, sort = TRUE) %>%
    top_n(10, n) %>%
    pull(neighbourhood)

  city_data_top_neighbourhoods <- city_data %>%
    filter(neighbourhood %in% top_neighbourhoods) %>%
    group_by(neighbourhood) %>%
    summarize(count = n(), .groups = 'drop') %>%
    mutate(proportion = count / sum(count))

  p <- ggplot(city_data_top_neighbourhoods, aes(y = reorder(neighbourhood, -proportion),
                                                x = proportion)) +
    geom_bar(stat = "identity", fill = city_colors[[current_city]]) +
    theme_minimal() +
    labs(title = current_city,
         x = "Proportion",
         y = "Neighborhood")

  plots[[current_city]] <- p
}

# Arrange the plots in a multi-column layout
neigh <- do.call(grid.arrange, c(plots, ncol = 2))
print(neigh)
ggsave("neigh_plot.png", plot = neigh, width = 10, height = 10, dpi = 300)
if (is.character(data$amenities[1])) {
  data <- data %>%
    mutate(amenities = str_remove_all(amenities, '[{}"]')) %>%
    mutate(amenities = str_split(amenities, ','))
}

# Unnest the amenities list column into multiple binary columns
data_unnested <- data %>%
  unnest(amenities) %>%
  mutate(amenities = str_trim(amenities)) %>%  # Trim white spaces
  mutate(amenities = ifelse(amenities == "", NA, amenities)) %>%  # Handle empty strings
  filter(!is.na(amenities)) %>%  # Remove NA amenities
```

```r
  distinct() %>%
  mutate(dummy = 1) %>%
  pivot_wider(names_from = amenities, values_from = dummy, values_fill = list(dummy = 0))

# Select only the amenities columns for correlation analysis
amenities_cols <- data_unnested %>%
  select(-id, -log_price, -review_scores_rating, -accommodates, -bathrooms, -number_of_reviews,
         -latitude, -longitude, -host_response_rate, -bedrooms, -beds, -host_has_profile_pic) %>%
  select(where(is.numeric))

# Analyze the impact of amenities on log_price
amenities_price_impact <- amenities_cols %>%
  summarise(across(everything(), ~ cor(.x, data_unnested$log_price, use = "complete.obs"))) %>%
  pivot_longer(everything(), names_to = "amenity", values_to = "correlation") %>%
  filter(!str_detect(amenity, "translation missing")) %>%  # Exclude translation missing amenities
  arrange(desc(correlation))

# Select top 10 positive and top 10 negative correlations for log_price
top_positive_price <- amenities_price_impact %>% top_n(10, correlation)
top_negative_price <- amenities_price_impact %>% top_n(-10, correlation)
top_price_impact <- bind_rows(top_positive_price, top_negative_price)

# Plot the top 10 positive and top 10 negative correlations for log_price
amenities_price <- ggplot(top_price_impact, aes(x = correlation, y = reorder(amenity, correlation),
                                                fill = correlation > 0)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = c("TRUE" = "indianred1", "FALSE" = "#6D9EC1")) +
  theme_minimal() +
  labs(x = "Correlation with Log Price",
       y = "Amenities")

# Analyze the impact of amenities on review_scores_rating
amenities_review_impact <- amenities_cols %>%
  summarise(across(everything(), ~ cor(.x, data_unnested$review_scores_rating, use = "complete.obs"))) %
  pivot_longer(everything(), names_to = "amenity", values_to = "correlation") %>%
  filter(!str_detect(amenity, "translation missing")) %>%  # Exclude translation missing amenities
  arrange(desc(correlation))

# Select top 10 positive and top 10 negative correlations for review_scores_rating
top_positive_review <- amenities_review_impact %>% top_n(10, correlation)
top_negative_review <- amenities_review_impact %>% top_n(-10, correlation)
top_review_impact <- bind_rows(top_positive_review, top_negative_review)

# Plot the top 10 positive and top 10 negative correlations for review_scores_rating
amenities_review <- ggplot(top_review_impact, aes(x = correlation,
                                                  y = reorder(amenity, correlation),
                                                  fill = correlation > 0)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = c("TRUE" = "indianred1", "FALSE" = "#6D9EC1")) +
  labs(x = "Correlation with Review Scores Rating",
       y = "Amenities")
print(amenities_price)
print(amenities_review)
```

```r
ggsave("amenities_price_plot.png", plot = amenities_price, width = 10, height = 10, dpi = 300)
ggsave("amenities_review_plot.png", plot = amenities_review, width = 10, height = 10, dpi = 300)
# Convert 'host_since' to Date and create a new feature 'host_duration_years' to see
# how long they have been a host
data$host_since <- as.Date(data$host_since)
data$host_duration_years <- as.numeric((Sys.Date() - data$host_since) / 365.25)

# Convert 'host_has_profile_pic' and 'host_identity_verified' to numeric for correlation analysis
data$host_has_profile_pic <- ifelse(data$host_has_profile_pic == "t", 1, 0)
data$host_identity_verified <- ifelse(data$host_identity_verified == "t", 1, 0)
# Select relevant columns for analysis
host_data <- data %>%
  select(host_response_rate, host_has_profile_pic, host_identity_verified,
         log_price, host_duration_years)

# Calculate the correlation matrix for host_data
correlation_matrix <- cor(host_data, use = "complete.obs")

# Plot the correlation matrix using ggcorrplot
corr_plot <- ggcorrplot(correlation_matrix,
           method = "circle",
           type = "lower",
           lab = TRUE, # Show correlation coefficients
           ggtheme = ggplot2::theme_minimal(),
           colors = c("#6D9EC1", "white", "indianred1"),
           lab_size = 3,
           outline.color = "gray")
print(corr_plot)
ggsave("corr_host_plot.png", plot = corr_plot, width = 10, height = 8, dpi = 300)
# Define host features and output variables
host_features <- c("host_has_profile_pic", "host_identity_verified")

# Create lists to store plots separately for log_price and review_scores_rating
plot_list_log_price <- vector("list", length = length(host_features))
plot_list_review_scores <- vector("list", length = length(host_features))

# Create plots and store them in the respective lists
for (i in seq_along(host_features)) {
  host_feature <- host_features[i]

  plot_list_log_price[[i]] <- ggplot(data, aes(x = .data[[host_feature]], y = log_price)) +
    geom_boxplot(aes(group = .data[[host_feature]]), fill = "indianred1") +
    theme_minimal() +
    labs(title = paste("Log Price vs", host_feature), x = host_feature, y = "Log Price") +
    theme(plot.title = element_text(hjust = 0.5))

  plot_list_review_scores[[i]] <- ggplot(data, aes(x = .data[[host_feature]],
                                                   y = review_scores_rating)) +
    geom_boxplot(aes(group = .data[[host_feature]]), fill = "indianred1") +
    theme_minimal() +
    labs(title = paste("Review Scores Rating vs", host_feature), x = host_feature,
         y = "Review Scores Rating") +
    theme(plot.title = element_text(hjust = 0.5))
```

```
}

# Display log_price plots
log_price_host <- grid.arrange(grobs = plot_list_log_price, ncol = 2)
print(log_price_host)

# Display review_scores_rating plots
review_scores_host <- grid.arrange(grobs = plot_list_review_scores, ncol = 2)
print(review_scores_host)
ggsave("log_price_host_plot.png", plot = log_price_host, width = 10, height = 6, dpi = 300)
ggsave("review_host_plot.png", plot = review_scores_host, width = 10, height = 6, dpi = 300)
histogram_plot <- ggplot(data, aes(x = review_scores_rating)) +
  geom_histogram(bins = 20, fill = "indianred1", color = "indianred1", alpha = 0.7) +
  geom_density(color = "indianred1", size = 1) +
  labs(title = "Distribution of Review Scores",
       x = "Review Scores Rating",
       y = "Frequency") +
  theme_minimal()

boxplot_plot <- ggplot(data, aes(y = review_scores_rating)) +
  geom_boxplot(fill = "indianred1", color = "indianred1", alpha = 0.7) +
  coord_flip() +  #
  labs(title = "Box Plot of Review Scores",
       x = "",
       y = "Review Scores Rating") +
  theme_minimal()

combined_plot <- histogram_plot + boxplot_plot + plot_layout(ncol = 2)
print(combined_plot)
ggsave("review_scores_plot.png", plot = combined_plot, width = 10, height = 6, dpi = 300)
review_room <- ggplot(data, aes(x = room_type, y = review_scores_rating)) +
  geom_boxplot(fill = "indianred1") +
  labs(x = "Room Type",
       y = "Review Scores Rating") +
  theme_minimal()
print(review_room)
ggsave("review_room_plot.png", plot = review_room, width = 10, height = 6, dpi = 300)
review_property <- ggplot(data, aes(x = reorder(property_type, -table(property_type)[property_type]),
                                    y = review_scores_rating)) +
  geom_boxplot(fill = "indianred1") +
  labs(
       x = "Property Type",
       y = "Review Scores Rating") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
print(review_property)
ggsave("review_property_plot.png", plot = review_property, width = 10, height = 6, dpi = 300)
```

## 2. PCA & K-Means

```
#### Neccessary Packages ####

library(tidyverse)
library(gamlr)
library(glmnet)
library(factoextra)
library(ggplot2)
library(ggfortify)
```

```
#### Import Data ####

setwd("C:/Users/user/Desktop/Big Data/HW/final")
data <- read.csv("Airbnb_Data.csv")

# delete id and text variables
data <- select(data, -c(id, description, name))
```

```
#### Data Preprocessing (by Mengdi) ####

data$room_type <- as.factor(data$room_type)
data$property_type <- as.factor(data$property_type)
data$bed_type <- as.factor(data$bed_type)
data$cancellation_policy <- as.factor(data$cancellation_policy)
data$city <- as.factor(data$city)
data$host_has_profile_pic <- as.factor(data$host_has_profile_pic)
data$host_identity_verified <- as.factor(data$host_identity_verified)
data$host_response_rate <- as.factor(data$host_response_rate)
data$instant_bookable <- as.factor(data$instant_bookable)

# Convert host_response_rate to numeric by removing the '%' sign
data$host_response_rate <- as.numeric(gsub("%", "", data$host_response_rate))
```

```
# handle amenities variable: transform into multiple columns
data$amenities <- str_replace_all(data$amenities, '[{}"]', '')
amenities_list <- str_split(data$amenities, ",")

all_amenities <- unique(unlist(amenities_list))

for (amenity in all_amenities) {
  data[[amenity]] <- sapply(amenities_list, function(x) amenity %in% x)
}

data <- select(data, -amenities)
```

```r
# handle date variables
data$first_review <- as.Date(data$first_review, format="%Y-%m-%d")
data$last_review <- as.Date(data$last_review, format="%Y-%m-%d")
data$host_since <- as.Date(data$host_since, format="%Y-%m-%d")

# transform date variables into more meaningful variables
data$days_since_first_review <- as.numeric(Sys.Date() - data$first_review)
data$days_since_last_review <- as.numeric(Sys.Date() - data$last_review)
data$host_duration <- as.numeric(Sys.Date() - data$host_since)

# delete first_review, last_review, host_since
data <- select(data, -c(first_review, last_review, host_since))
```

```r
# turn thumbnail_url into a binary categorical variable
data$has_thumbnail <- ifelse(is.na(data$thumbnail_url) | data$thumbnail_url == "", FALSE, TRUE)

# delete thumbnail_url
data <- select(data, -thumbnail_url)
```

```r
count_missing <- function(x) {
  sum(is.na(x) | x == "")
}

# Create a summary of missing values (NA and empty strings) for each column
missing_values_summary <- data %>%
  summarise_all(count_missing) %>%
  gather(key = "variable", value = "missing_count") %>%
  arrange(desc(missing_count))

print(missing_values_summary)
```

```r
# Impute the above numerical variables that have missing values with median values
data$host_response_rate[is.na(data$host_response_rate)] <-
  median(data$host_response_rate, na.rm = TRUE)
data$review_scores_rating[is.na(data$review_scores_rating)] <-
  median(data$review_scores_rating, na.rm = TRUE)
data$days_since_first_review[is.na(data$days_since_first_review)] <-
  median(data$days_since_first_review, na.rm = TRUE)
data$days_since_last_review[is.na(data$days_since_last_review)] <-
  median(data$days_since_last_review, na.rm = TRUE)
data$bathrooms[is.na(data$bathrooms)] <-
  median(data$bathrooms, na.rm = TRUE)
data$host_duration[is.na(data$host_duration)] <-
  median(data$host_duration, na.rm = TRUE)
data$beds[is.na(data$beds)] <- median(data$beds, na.rm = TRUE)
data$bedrooms[is.na(data$bedrooms)] <- median(data$bedrooms, na.rm = TRUE)

# Replace missing values with specific values for categorical columns
data$host_has_profile_pic[is.na(data$host_has_profile_pic) |
                            data$host_has_profile_pic == ''] <- 'f'
data$host_identity_verified[is.na(data$host_identity_verified) |
                              data$host_identity_verified == ''] <- 'f'
data$neighbourhood[is.na(data$neighbourhood) |
```

```
                    data$neighbourhood == ''] <- 'Unknown'
data$zipcode[is.na(data$zipcode) | data$zipcode == ''] <- 'Unknown'

# Convert to factors
data$neighbourhood <- as.factor(data$neighbourhood)
data$zipcode <- as.factor(data$zipcode)
```

```
# extract numerical variable names
numeric_vars <- c("log_price", "accommodates", "bathrooms", "host_response_rate",
                  "latitude", "longitude", "number_of_reviews", "review_scores_rating",
                  "bedrooms", "beds", "days_since_first_review", "days_since_last_review",
                  "host_duration")

# extract categorical variable names
categorical_vars <- setdiff(names(data), numeric_vars)

# standardize numerical variables
data_numeric <- scale(data[numeric_vars])
data_numeric <- as.data.frame(data_numeric)

# combine standardized numerical variables with categorical variables
data_scale <- cbind(data_numeric, data[categorical_vars])
```

```
names(data) <- make.names(names(data), unique = TRUE)

print(names(data))
```

```
#### Further Data Cleaning ####

factor_cols <- names(data_scale)[sapply(data_scale, is.factor)]

# Convert factor columns to numeric values
for (col in factor_cols) {
  data[[col]] <- as.numeric(factor(data[[col]])) - 1
}
```

```
# Identify columns with only TRUE and FALSE values
logical_cols <- sapply(data, function(col) is.logical(col) && all(col %in% c(TRUE, FALSE)))

# Convert logical columns to 0s and 1s
logical_col_names <- names(logical_cols)[logical_cols]
data[logical_col_names] <- lapply(data[logical_col_names], as.integer)
```

```
# Convert "True" and "FALSE" strings to logical
data$cleaning_fee <- tolower(data$cleaning_fee) == "true"

# Convert logical values to numeric 0s and 1s
data$cleaning_fee <- as.integer(data$cleaning_fee)
```

```
# Define categories
Essentials = c("Essentials", "Hangers", "Hair.dryer", "Iron",
               "First.aid.kit",  "Safety.card", "Lock.on.bedroom.door",
```

```
               "TV", "Cable.TV", "Bed.linens", "Extra.pillows.and.blankets",
               "Changing.table")
Facilities = c("Air.conditioning", "Heating", "Breakfast", "Pool",
               "Gym", "Hot.tub", "Elevator", "Elevator.in.building",
               "Washer", "Dryer", "Laptop.friendly.workspace")
Parking = c("Free.parking.on.street","Free.parking.on.premises",
            "Paid.parking.off.premises")
Privacy = c("Private.bathroom", "Private.living.room", "Private.entrance")
Family = c("Family.kid.friendly", "Children.s.books.and.toys",
           "Children.s.dinnerware", "Crib", "High.chair", "Stair.gates",
           "Window.guards", "Table.corner.guards", "Baby.monitor",
           "Baby.bath", "Fireplace.guards", "Game.console",
           "Babysitter.recommendations", "Pack..n.Play.travel.crib")
Safety = c("Fire.extinguisher","Smoke.detector","Indoor.fireplace",
           "Carbon.monoxide.detector")
Pets = c("Pets.allowed", "Dog.s.", "Cat.s.", "Other.pet.s.",
         "Pets.live.on.this.property")
Kitchen = c("Kitchen", "Microwave", "Coffee.maker", "Refrigerator",
            "Dishes.and.silverware", "Dishwasher", "Oven", "Stove",
            "Cooking.basics", "Hot.water.kettle")
Internet = c("Internet","Wireless.Internet", "Ethernet.connection",
             "Buzzer.wireless.intercom")
Self_Checkin = c("Self.Check.In", "Lockbox")
Accessibility = c("Wheelchair.accessible", "Wide.clearance.to.bed",
                  "Accessible.height.bed", "Wide.doorway",
                  "Accessible.height.toilet", "Wide.entryway", "Step.free.access",
                  "Ground.floor.access", "Wide.clearance.to.shower...toilet",
                  "Wide.clearance.to.shower.and.toilet", "Wide.hallway.clearance",
                  "Flat.smooth.pathway.to.front.door",
                  "X.smooth.pathway.to.front.door", "Well.lit.path.to.entrance")
Events = c("Suitable.for.events", "Doorman", "Doorman.Entry")
Others = c("Other", "translation.missing..en.hosting_amenity_49", "V106",
           "Single.level.home","Flat")
Bathroom = c("Bathtub", "Hot.water", "Shampoo", "Bathtub.with.shower.chair",
             "Handheld.shower.head", "Grab.rails.for.shower.and.toilet",
             "Body.soap", "Hand.soap", "Bath.towel", "Hand.or.paper.towel",
             "Toilet.paper")
Outdoor = c("Garden.or.backyard", "Patio.or.balcony", "BBQ.grill",
            "Lake.access", "Beachfront", "Beach.essentials", "Ski.in.Ski.out",
            "Path.to.entrance.lit.at.night", "Waterfront")
Smoking = c("Smoking.allowed")
Miscellaneous = c("Luggage.dropoff.allowed","Outlet.covers",
                  "Long.term.stays.allowed", "Firm.mattress", "Pocket.wifi",
                  "Cleaning.before.checkout", "EV.charger", "Keypad",
                  "Smart.lock")


data_new <- data %>%
  mutate(
    Essentials = ifelse(rowSums(select(., all_of(Essentials))) > 0, 1, 0),
    Facilities = ifelse(rowSums(select(., all_of(Facilities))) > 0, 1, 0),
    Parking = ifelse(rowSums(select(., all_of(Parking))) > 0, 1, 0),
    Privacy = ifelse(rowSums(select(., all_of(Privacy))) > 0, 1, 0),
```

```
    Family = ifelse(rowSums(select(., all_of(Family))) > 0, 1, 0),
    Safety = ifelse(rowSums(select(., all_of(Safety))) > 0, 1, 0),
    Pets = ifelse(rowSums(select(., all_of(Pets))) > 0, 1, 0),
    Kitchen = ifelse(rowSums(select(., all_of(Kitchen))) > 0, 1, 0),
    Internet = ifelse(rowSums(select(., all_of(Internet))) > 0, 1, 0),
    Self_Checkin = ifelse(rowSums(select(., all_of(Self_Checkin))) > 0, 1, 0),
    Accessibility = ifelse(rowSums(select(., all_of(Accessibility))) > 0, 1, 0),
    Events = ifelse(rowSums(select(., all_of(Events))) > 0, 1, 0),
    Others = ifelse(rowSums(select(., all_of(Others))) > 0, 1, 0),
    Bathroom = ifelse(rowSums(select(., all_of(Bathroom))) > 0, 1, 0),
    Outdoor = ifelse(rowSums(select(., all_of(Outdoor))) > 0, 1, 0),
    Smoking = ifelse(rowSums(select(., all_of(Smoking))) > 0, 1, 0),
    Miscellaneous = ifelse(rowSums(select(., all_of(Miscellaneous))) > 0, 1, 0)
  )
```

```
# Define columns to keep
columns_to_keep <- c("log_price", "property_type", "room_type", "accommodates",
                     "bathrooms", "bed_type", "cancellation_policy",
                     "cleaning_fee", "city", "host_has_profile_pic",
                     "host_identity_verified", "host_response_rate",
                     "instant_bookable", "latitude", "longitude", "neighbourhood",
                     "number_of_reviews", "review_scores_rating",
                     "days_since_first_review", "days_since_last_review",
                     "zipcode", "bedrooms", "beds", "Essentials", "Facilities",
                     "Parking", "Privacy", "Family", "Safety", "Pets", "Kitchen",
                     "Internet", "Self_Checkin",  "Accessibility", "Events",
                     "Others", "Bathroom", "Outdoor", "Smoking", "Miscellaneous")

# Create new data frame with only columns to keep
data_new <- data_new %>%
  select(all_of(columns_to_keep))
```

```
#### Principal Component Analysis (PCA)####
#### PCA with only numerical values ####

# Remove the 'log_price' column from data_numeric
data_without_y <- data[numeric_vars][, setdiff(names(data[numeric_vars]), c("log_price"))]

# Perform PCA
pca_result <- prcomp(data_without_y, scale. = TRUE)

# Summary of PCA
summary(pca_result)
```

The summary of PCA shows the standard deviation, proportion of variance, and cumulative proportion explained by each principal component.

```
# The scree plot
plot(pca_result, xlab = "Principal Component", col = "#ff4e4e")
```

```
# Extract the scores of the observations along the principal components
scores <- pca_result$x
```

```r
# Plot the scores of the observations on the first two principal components
plot(scores[,1], scores[,2],
     xlab = "Principal Component 1", ylab = "Principal Component 2",
     main = "Scatter Plot of Principal Components", col="#ff4e4e")
```

```r
##### Selection of the Number of Principal Components:####

# Extract the first K principal components
pca_data <-predict(pca_result)
pca_df <- as.data.frame(pca_data)

log_price <- data_numeric$log_price

kfits <- lapply(1:12, function(K) glm(log_price~., data = pca_df[, 1:K, drop=FALSE]))

aicc <- sapply(kfits, AICc)
plot(aicc, col = "#ff4e4e", main = "AICc Plot for Model Selection")
```

```r
which.min(aicc)
```

```r
##### Generalized Linear Model (GLM):####

# GLM on First K Technique
log_price <- data_numeric$log_price
glm <- glm(log_price ~ ., data = pca_df, family = gaussian)
summary(glm)
```

```r
##### Lasso Technique:####

# Lasso Technique
lasso_model <- cv.glmnet(x=pca_data, y=log_price, nfold=20)

coef(lasso_model)
```

```r
#### PCA with all columns (transforming catgorical columns into dummy)####

# Scale the Data
data_scale2 <- scale(data_new)
data_scale2 <- as.data.frame(data_scale2)
```

```r
# Remove the 'log_price' column
data_without_y <- data_scale2[, !names(data_scale2) %in% "log_price"]

# Perform PCA
pca_result <- prcomp(data_without_y, scale. = FALSE)

# Summary of PCA
summary(pca_result)
```

```r
# The scree plot
plot(pca_result, xlab = "Principal Component", col="#ff4e4e")
```

```r
# Extract the scores of the observations along the principal components
scores <- pca_result$x

# Plot the scores of the observations on the first two principal components
plot(scores[,1], scores[,2],
     xlab = "Principal Component 1", ylab = "Principal Component 2",
     main = "Scatter Plot of Principal Components", col="#ff4e4e")
```

```r
# Extract the first K principal components
pca_data <-predict(pca_result)
pca_df <- as.data.frame(pca_data)

log_price <- data_numeric$log_price

kfits <- lapply(1:39, function(K) glm(log_price~., data = pca_df[, 1:K, drop=FALSE]))

aicc <- sapply(kfits, AICc)
# Find the index of the minimum AICc value
min_index <- which.min(aicc)

# Plot the AICc values and add a vertical line at the minimum
plot(aicc, col = "#ff4e4e", main = "AICc Plot for Model Selection",
     xlab = "Number of Principal Components")
abline(v = min_index, col = "blue", lty = 2)

# Annotate the plot with the number of the principal component
text(min_index, aicc[min_index], labels = paste("Min:", min_index), pos = 3, col = "blue")
```

```r
which.min(aicc)
```

```r
##### Using GLM #####

# GLM on First K Technique
log_price <- data_scale2$log_price
glm <- glm(log_price ~ ., data = pca_df[1:37], family = gaussian)
summary(glm)
```

```r
# Extract coefficients and their p-values
coefficients <- coef(glm)
p_values <- summary(glm)$coefficients[, "Pr(>|t|)"]

# Filter statistically significant coefficients
significant_coefficients <- coefficients[p_values < 0.05]

# Extract names of significant coefficients
significant_coefficient_names <- names(significant_coefficients)

# Count the number of significant coefficients (p-value < 0.05)
num_significant <- sum(p_values < 0.05)
```

```r
# Print the number of significant coefficients
print(significant_coefficient_names)
print(num_significant)
```

```r
##### Using Lasso Regression #####

# Lasso Technique
lasso_model <- cv.glmnet(x=pca_data[, 1:37], y=log_price, nfold=20)

coef(lasso_model)
```

```r
# Extract coefficients
coefficients <- coef(lasso_model)

# Extract variable names
variable_names <- rownames(coefficients)[-1]  # Exclude the intercept term

# Find significant coefficients
significant_indices <- which(coefficients[-1, ] != 0)

# Print names of significant coefficients
significant_variable_names <- variable_names[significant_indices]
print(significant_variable_names)


# Number of significant coefficients
num_significant_coefficients <- sum(coefficients != 0)

# Print number of significant coefficients
print(num_significant_coefficients)
```

```r
# Predict using the Lasso model
lasso_predictions <- predict(lasso_model, newx = pca_data[, 1:37])

# Calculate mean squared error (MSE)
mse <- mean((lasso_predictions - log_price)^2)

# Calculate mean absolute error (MAE)
mae <- mean(abs(lasso_predictions - log_price))

# Calculate R-squared (R2)
actual_mean <- mean(log_price)
ss_total <- sum((log_price - actual_mean)^2)
ss_residual <- sum((log_price - lasso_predictions)^2)
r_squared <- 1 - (ss_residual / ss_total)

# Print the evaluation metrics
print(paste("Mean Squared Error (MSE):", mse))
print(paste("Mean Absolute Error (MAE):", mae))
print(paste("R-squared (R2):", r_squared))
```

```r
#### Kmeans Cluster ####

data_without_price <- data_scale2[, setdiff(names(data_scale2), c("log_price"))]
```

```r
#### 1. Elbow Method (Within-Cluster Sum of Squares): ####

set.seed(123)

# function to compute total within-cluster sum of square
wss <- function(k) {
  kmeans(data_without_price, k, nstart = 10 )$tot.withinss
}

# Compute and plot wss for k = 1 to k = 15
k.values <- 1:15

# extract wss for 2-15 clusters
wss_values <- map_dbl(k.values, wss)

plot(k.values, wss_values,
       type="b", pch = 19, frame = FALSE,
       xlab="Number of clusters K",
       ylab="Total within-clusters sum of squares")
```

```r
#### 2. Silhouette Method:####

# Assuming your data is stored in a data frame called "your_data"
sampled_data <- data_without_price[sample(nrow(data_without_price), 25000), ]

# Finding the Optimal Number of Clusters with the Silhouette Method
fviz_nbclust(sampled_data, kmeans, method = "silhouette")
```

```r
# Plotting K2 to K5
k2 <- kmeans(data_without_price, centers = 2, nstart = 25)
k3 <- kmeans(data_without_price, centers = 3, nstart = 25)
k4 <- kmeans(data_without_price, centers = 4, nstart = 25)
k5 <- kmeans(data_without_price, centers = 5, nstart = 25)

# plots to compare
p1 <- fviz_cluster(k2, geom = "point", data = data_without_price) + ggtitle("k = 2")
p2 <- fviz_cluster(k3, geom = "point",  data = data_without_price) + ggtitle("k = 3")
p3 <- fviz_cluster(k4, geom = "point",  data = data_without_price) + ggtitle("k = 4")
p4 <- fviz_cluster(k5, geom = "point",  data = data_without_price) + ggtitle("k = 5")

library(gridExtra)
grid.arrange(p1, p2, p3, p4, nrow = 2)
```

```r
#### K=2 ####

k2$centers
```

```r
# Size of each Cluster
k2$size
```

```r
# Data frame containing cluster centers
cluster_centers <- data.frame(
  cluster = c("Cluster 1", "Cluster 2"),  # Cluster labels
  variable = colnames(k2$centers),     # Variable names
  value = c(k2$centers[1, ], k2$centers[2, ])  # Average values for each variable in each cluster
)

# Bar plot
bar_plot <- ggplot(cluster_centers, aes(x = variable, y = value, fill = cluster)) +
  geom_bar(stat = "identity", position = "dodge", width = 0.5) +
  labs(x = "Variable", y = "Average Value", title = "Cluster Profiles") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Radar chart
radar_plot <- ggplot(cluster_centers, aes(x = variable, y = value,
                                          color = cluster, group = cluster)) +
  geom_line() +
  geom_point(size = 2) +
  labs(x = NULL, y = NULL, title = "Cluster Profiles") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Display plots
print(bar_plot)
print(radar_plot)
```

```r
# Top 10 Variables that Most Distinguish the Two Clusters
print(apply(k2$centers,1,function(c) colnames(data_without_price)[order(-c)[1:10]]))
```

```r
# Assign clusters to the original data
clusters_k2 <- data_scale2
clusters_k2$cluster <- k2$cluster

# Calculate summary statistics of log_price for each cluster
cluster_summary_k2 <- clusters_k2 %>%
  group_by(cluster) %>%
  summarize(
    mean_log_price = mean(log_price),
    median_log_price = median(log_price),
    sd_log_price = sd(log_price)
  )
print(cluster_summary_k2)
```

```r
# Visualize the log_price distribution for each cluster with different colors
ggplot(clusters_k2, aes(x = factor(cluster), y = log_price, fill = factor(cluster))) +
  geom_boxplot() +
  stat_summary(geom = "text", fun = quantile, aes(label = sprintf("%1.2f", ..y..)),
               position = position_nudge(x = 0.1), size = 3) +
  labs(title = "Log Price Distribution Across Clusters",
```

```
       x = "Cluster",
       y = "Log Price") +
  scale_fill_discrete(name = "Cluster") +
  theme_minimal()
```

```
# Visualize the Review Scores Rating distribution for each cluster with different colors
ggplot(clusters_k2, aes(x = factor(cluster), y = review_scores_rating, fill = factor(cluster))) +
  geom_boxplot() +
  stat_summary(geom = "text", fun = quantile, aes(label = sprintf("%1.2f", ..y..)),
               position = position_nudge(x = 0.1), size = 3) +
  labs(title = "Review Scores Rating Distribution Across Clusters",
       x = "Cluster",
       y = "Review Scores Rating") +
  scale_fill_discrete(name = "Cluster") +
  theme_minimal()
```

```
# Visualize the Property Type distribution for each cluster with different colors
ggplot(clusters_k2, aes(x = factor(cluster), y = property_type, fill = factor(cluster))) +
  geom_boxplot() +
  stat_summary(geom = "text", fun = quantile, aes(label = sprintf("%1.2f", ..y..)),
               position = position_nudge(x = 0.1), size = 3) +
  labs(title = "Property Type Distribution Across Clusters",
       x = "Cluster",
       y = "Property Type") +
  scale_fill_discrete(name = "Cluster") +
  theme_minimal()
```

```
#### 3. Information Criteria (AIC/BIC): ####

kfit <- lapply(1*(1:10), function(k) kmeans(data_without_price,k))
```

```
source("kIC.R")

kaicc <- sapply(kfit,kIC)
kbic <- sapply(kfit,kIC,"B")
```

```
k_values <- c(1,2,3,4,5,6,7,8,9,10)
optimal_k_aicc <- k_values[which.min(kaicc)]
optimal_k_bic <- k_values[which.min(kbic)]
optimal_k_bic
```

```
kmfs <- kfit[[which(k_values==optimal_k_bic)]]
print(apply(kmfs$centers,1,function(c) colnames(data_without_price)[order(-c)[1:5]]))
```

```
## Size of Each Cluster
kmfs$size
```

```
# Calculate summary statistics of log_price for each cluster
cluster_summary_k10 <-data_with_clusters %>%
  group_by(cluster) %>%
  summarize(
```

```
    mean_log_price = mean(log_price),
    median_log_price = median(log_price),
    sd_log_price = sd(log_price)
  )
print(cluster_summary_k10)
```

```
# Visualize the log_price distribution for each cluster with different colors
ggplot(data_with_clusters, aes(x = factor(cluster), y = log_price, fill = factor(cluster))) +
  geom_boxplot() +
  stat_summary(geom = "text", fun = quantile, aes(label = sprintf("%1.2f", ..y..)),
               position = position_nudge(x = 0.5), size = 2.5) +
  labs(title = "Log Price Distribution Across Clusters",
       x = "Cluster",
       y = "Log Price") +
  scale_fill_discrete(name = "Cluster") +
  theme_minimal()
```

```
k10 <- kmeans(data_without_price, centers = 10, nstart = 25)
autoplot(k10, data_without_price, frame = TRUE)
```

## 3. Topics Model

```
#KIC function from kIC.R
## get AICc and BIC for the output of kmeans
kIC <- function(fit, rule=c("A","B")){
    df <- length(fit$centers) # K*dim
    n <- sum(fit$size)
    D <- fit$tot.withinss # deviance
    rule=match.arg(rule)
    if(rule=="A")
        return(D + 2*df*n/max(n-df-1,1))
    else
        return(D + log(n)*df)
}
#load and handle data
library(tidyverse)
library(dplyr)
library(tidyr)
library(tm)
library(topicmodels)
library(caret)
library(tidytext)
library(textdata)
library(quanteda)
library(stringr)
library(textmineR)
library(cld2)
library(textir)
library(maptpx)
library(tm)
library(slam)
library(wordcloud)
library(gamlr)
library(dplyr)
library(tidyr)
library(caret)
library(textdata)
data <- read.csv("Airbnb_Data.csv")
data <- select(data, c(id, log_price, description, name))

clean_text <- function(text) {
  text <- tolower(text)
  text <- removePunctuation(text)
  text <- removeNumbers(text)
  text <- stripWhitespace(text)
  text <- removeWords(text, stopwords("en"))
  text <- str_replace_all(text, "[^\x20-\x7E]", "")
  return(text)
}

data$clean_description <- sapply(data$description, clean_text)
data$clean_name <- sapply(data$name, clean_text)
```

```r
data <- data %>%
  mutate_all(~ na_if(trimws(.), ""))

# Alternatively, if you only want to replace strings with only spaces (not empty strings):
data <- data %>%
  mutate_all(~ na_if(., "   "))

# Remove rows with any NA values
clean_data <- data[complete.cases(data), ]
# Topics modeling for Descriptions
# Create a Corpus
corpus_description <- Corpus(VectorSource(clean_data$clean_description[1:10000]))

# Create a Document-Term Matrix
dtm_description <- DocumentTermMatrix(corpus_description)

# Normalize term frequencies
fs <- scale(as.matrix(dtm_description))

# Fit k-means for k in 5, 10, 15, 20, 25
kfit <- lapply(5*(1:5), function(k) kmeans(fs, k))

# Calculate Information Criteria
kaicc <- sapply(kfit, kIC)
kbic <- sapply(kfit, kIC, "B")

# Plot Information Criteria
par(mfrow=c(1,2))
plot(5*(1:5), kaicc, xlab="K", ylab="IC", bty="n", type="l", lwd=2)
abline(v=which.min(kaicc)*5)
plot(5*(1:5), kbic, xlab="K", ylab="IC", bty="n", type="l", lwd=2, col=4)
abline(v=which.min(kbic)*5, col=4)

# Choose the best model
best_k <- which.min(kbic)*5
kmfs <- kfit[[best_k / 5]]

# Interpretation: Words with highest cluster centers
print(apply(kmfs$centers, 1, function(c) colnames(fs)[order(-c)[1:10]]))

# Cluster sizes
kmfs$size
# Topics modeling for Descriptions
# Create a Corpus
corpus_name <- Corpus(VectorSource(clean_data$clean_name[1:10000]))

# Create a Document-Term Matrix
dtm_name <- DocumentTermMatrix(corpus_name)

# Normalize term frequencies
fs <- scale(as.matrix(dtm_name))

# Fit k-means for k in 5, 10, 15, 20, 25
```

```r
kfit <- lapply((3:8), function(k) kmeans(fs, k))

# Calculate Information Criteria
kaicc <- sapply(kfit, kIC)
kbic <- sapply(kfit, kIC, "B")

# Plot Information Criteria
par(mfrow=c(1,2))
plot((3:8), kaicc, xlab="K", ylab="IC", bty="n", type="l", lwd=2)
abline(v=which.min(kaicc)*5)
plot((3:8), kbic, xlab="K", ylab="IC", bty="n", type="l", lwd=2, col=4)
abline(v=which.min(kbic)*5, col=4)

# Choose the best model
best_k <- which.min(kbic)
kmfs <- kfit[[best_k]]

# Interpretation: Words with highest cluster centers
print(apply(kmfs$centers, 1, function(c) colnames(fs)[order(-c)[1:10]]))

# Cluster sizes
kmfs$size
# Topics modeling for Descriptions
# Create a Corpus
corpus_name <- Corpus(VectorSource(clean_data$clean_name[1:10000]))

# Create a Document-Term Matrix
dtm_name <- DocumentTermMatrix(corpus_name)

# Normalize term frequencies
fs <- scale(as.matrix(dtm_name))

# Fit k-means for k in 5, 10, 15, 20, 25
kfit <- lapply((3:8), function(k) kmeans(fs, k))

# Calculate Information Criteria
kaicc <- sapply(kfit, kIC)
kbic <- sapply(kfit, kIC, "B")
# Plot Information Criteria
par(mfrow=c(1,2))
plot((3:8), kaicc, xlab="K", ylab="IC", bty="n", type="l", lwd=2)
abline(v=which.min(kaicc)*5)
plot((3:8), kbic, xlab="K", ylab="IC", bty="n", type="l", lwd=2, col=4)
abline(v=which.min(kbic)*5, col=4)

# Choose the best model
best_k <- which.min(kbic)
kmfs <- kfit[[best_k]]

# Interpretation: Words with highest cluster centers
print(apply(kmfs$centers, 1, function(c) colnames(fs)[order(-c)[1:10]]))

# Cluster sizes
```

```r
kmfs$size
# Topics modeling for Descriptions
# Create a Corpus
corpus_description <- Corpus(VectorSource(clean_data$clean_description[1:1000]))

# Create a Document-Term Matrix
dtm_description <- DocumentTermMatrix(corpus_description)

# Apply Latent Dirichlet Allocation (LDA)
lda_model <- LDA(dtm_description, k = 4, control = list(seed = 123))

# Extract topics
topics <- terms(lda_model, 25)
topics
#Topics modeling for names
# Create a Corpus
corpus_name <- Corpus(VectorSource(clean_data$clean_name[1:300]))

# Create a Document-Term Matrix
dtm_name <- DocumentTermMatrix(corpus_name)

# Apply Latent Dirichlet Allocation (LDA)
lda_model <- LDA(dtm, k = 4, control = list(seed = 123))

# Extract topics
topics <- terms(lda_model, 25)
topics
med <- median(clean_data$log_price)
exp(as.numeric(med))
# Load required libraries
library(dplyr)
library(tidyr)
library(tm)
library(MASS)
library(tidytext)
library(textdata)

# Assuming your dataframe is named 'clean_data'
# and it contains a column 'clean_description' and 'log_price'
# Sample clean_data for illustration purposes


# Create a binary label for high price (log_price > 5 is high price)
clean_data$high_price <- ifelse(clean_data$log_price > median(clean_data$log_price), 1, 0)

# Create a Corpus
corpus <- Corpus(VectorSource(clean_data$clean_description[1:300]))

# Create a Document-Term Matrix
dtm <- DocumentTermMatrix(corpus)

# Convert DTM to a data frame
dtm_df <- as.data.frame(as.matrix(dtm))
```

```r
# Remove columns with zero variance
dtm_df <- dtm_df[, apply(dtm_df, 2, var) > 0]

# Remove columns that are excessively sparse
sparse_threshold <- 0.95  # Adjust this threshold as needed
dtm_df <- dtm_df[, colMeans(dtm_df != 0) > (1 - sparse_threshold)]

# Add the high_price label to the DTM data frame
dtm_df$high_price <- as.factor(clean_data$high_price[1:300])  # Ensure it's a factor

# Split the data into training and testing sets
set.seed(123)
train_indices <- sample(1:nrow(dtm_df), size = 0.7 * nrow(dtm_df))
train_data <- dtm_df[train_indices, ]
test_data <- dtm_df[-train_indices, ]

# Prepare the predictor matrix and grouping factor
x_train <- train_data[, -ncol(train_data)]  # All columns except the last one
grouping_train <- train_data$high_price

# Fit the LDA model
lda_model <- lda(x_train, grouping_train)

# Prepare the test predictor matrix
x_test <- test_data[, -ncol(test_data)]
grouping_test <- test_data$high_price

# Predict on test data
predictions <- predict(lda_model, newdata = x_test)

# Evaluate the model
confusion_matrix <- table(grouping_test, predictions$class)
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
print(confusion_matrix)
print(paste("Accuracy:", accuracy))

# Optionally, add sentiment analysis
# Perform sentiment analysis
clean_data <- clean_data %>%
  rowwise() %>%
  mutate(sentiment = {
    tokens <- unlist(strsplit(clean_description, " "))
    sentiment_values <- inner_join(tibble(word = tokens), get_sentiments("afinn"), by = "word")
    sum(sentiment_values$value, na.rm = TRUE)
  }) %>%
  ungroup()

# Replace NA sentiment values with 0
clean_data$sentiment[is.na(clean_data$sentiment)] <- 0

# Display the resulting data
print(clean_data)
#Confusion Matrix Graph
```

```r
confusion_matrix <- table(grouping_test, predictions$class)
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
print(confusion_matrix)
print(paste("Accuracy:", accuracy))
#Sentiment Analysis Graph
# Prepare data for modeling
model_data <- data %>%
  select(log_price, topic_1, topic_2, topic_3, sentiment) %>%
  na.omit()

# Split data into training and testing sets
set.seed(1234)
trainIndex <- createDataPartition(model_data$log_price, p = .8,
                                  list = FALSE,
                                  times = 1)
train_data <- model_data[ trainIndex,]
test_data <- model_data[-trainIndex,]

# Train a regression model
model <- train(log_price ~ ., data = train_data, method = "lm")

# Predict on test data
predictions <- predict(model, newdata = test_data)

# Evaluate model performance
RMSE(predictions, test_data$log_price)
```

# 4. CV.LASSO, RIDGE, and NAIVE LASSO

General Analysis

```
data_scale$cleaning_fee <- as.logical(data_scale$cleaning_fee)

x <- model.matrix(log_price ~ ., data_scale)[, -1]
y <- data_scale$log_price

# Set up cross-validation for LASSO and Ridge Regression
set.seed(123) # For reproducibility
cv_lasso <- cv.glmnet(x, y, alpha = 1, family = 'gaussian', standardize = TRUE)
cv_ridge <- cv.glmnet(x, y, alpha = 0, family = 'gaussian', standardize = TRUE)

# Best lambda for each model
best_lambda_lasso <- cv_lasso$lambda.min
best_lambda_ridge <- cv_ridge$lambda.min

# Train the final models using the best lambda
lasso_model <- glmnet(x, y, alpha = 1, lambda = best_lambda_lasso, family = 'gaussian',
                      standardize = TRUE)
ridge_model <- glmnet(x, y, alpha = 0, lambda = best_lambda_ridge, family = 'gaussian',
                      standardize = TRUE)

set.seed(123)
trainIndex <- createDataPartition(data_scale$log_price, p = 0.8, list = FALSE, times = 1)
loft_train <- data_scale[trainIndex, ]
loft_test <- data_scale[-trainIndex, ]

# Prepare training and test sets for prediction
x_train <- model.matrix(log_price ~ ., loft_train)[, -1]
y_train <- loft_train$log_price
x_test <- model.matrix(log_price ~ ., loft_test)[, -1]
y_test <- loft_test$log_price

lasso_pred <- predict(lasso_model, s = best_lambda_lasso, family = 'gaussian', newx = x_test)
ridge_pred <- predict(ridge_model, s = best_lambda_ridge, faimily = 'gaussian', newx = x_test)

lasso_mse <- mean((y_test - lasso_pred)^2)
lasso_rmse <- sqrt(lasso_mse)
lasso_mae <- mean(abs(y_test - lasso_pred))
lasso_r2 <- cor(y_test, lasso_pred)^2

ridge_mse <- mean((y_test - ridge_pred)^2)
ridge_rmse <- sqrt(ridge_mse)
ridge_mae <- mean(abs(y_test - ridge_pred))
```

```r
ridge_r2 <- cor(y_test, ridge_pred)^2

comparison <- data.frame(
  Model = c("LASSO", "Ridge"),
  RMSE = c(lasso_rmse, ridge_rmse),
  MSE = c(lasso_mse, ridge_mse),
  MAE = c(lasso_mae, ridge_mae),
  R_squared = c(lasso_r2, ridge_r2)
)
comparison
```

```r
set.seed(123)
train_index <- createDataPartition(data_scale$log_price, p = 0.8, list = FALSE)
train_data <- data_scale[train_index, ]
test_data <- data_scale[-train_index, ]

X <- as.matrix(train_data[, -which(names(train_data) == "log_price")])
Y <- train_data$log_price

cv.fit <- cv.glmnet(x_train, y_train, family = "gaussian", alpha = 1)

# Coefficients for the best lambda value
coefficients_best_lambda <- coef(cv.fit, s = "lambda.min")
head(coefficients_best_lambda, 20)
nonzero_coef_best_lambda_count <- sum(coefficients_best_lambda[-1] != 0)
print(nonzero_coef_best_lambda_count)

# Coefficients for the lambda value selected by 1 standard error rule
coefficients_1se <- coef(cv.fit, s = "lambda.1se")
head(coefficients_1se, 20)
nonzero_coef_1se_count <- sum(coefficients_1se[-1] != 0)
print(nonzero_coef_1se_count)
```

```r
#Top 10 coefficients for general analysis with LASSO
lasso_coefs <- coef(lasso_model, s = best_lambda_lasso)
lasso_coefs <- as.data.frame(as.matrix(lasso_coefs))
colnames(lasso_coefs) <- c("Coefficient")
lasso_coefs <- lasso_coefs %>%
  rownames_to_column(var = "Feature") %>%
  arrange(desc(abs(Coefficient)))
head(lasso_coefs, 10)
```

```r
# Extract coefficients for general analysis with Ridge
ridge_coefs <- coef(ridge_model, s = best_lambda_ridge)
ridge_coefs <- as.data.frame(as.matrix(ridge_coefs))
colnames(ridge_coefs) <- c("Coefficient")
ridge_coefs <- ridge_coefs %>%
  rownames_to_column(var = "Feature") %>%
  arrange(desc(abs(Coefficient)))
head(ridge_coefs, 10)
```

```r
#In_sample_R2 vs OOS_R2 comparison
lasso_pred_train <- predict(lasso_model, s = best_lambda_lasso, family = 'gaussian', newx = x_train)
ridge_pred_train <- predict(ridge_model, s = best_lambda_ridge, family = 'gaussian', newx = x_train)

lasso_pred_test <- predict(lasso_model, s = best_lambda_lasso, newx = x_test)
ridge_pred_test <- predict(ridge_model, s = best_lambda_ridge, newx = x_test)

in_sample_r2 <- function(y_true, y_pred) {
  cor(y_true, y_pred)^2
}
out_of_sample_r2 <- function(y_true, y_pred) {
  1 - sum((y_true - y_pred)^2) / sum((y_true - mean(y_true))^2)
}

lasso_in_sample_r2 <- in_sample_r2(y_train, lasso_pred_train)
ridge_in_sample_r2 <- in_sample_r2(y_train, ridge_pred_train)

lasso_out_sample_r2 <- out_of_sample_r2(y_test, lasso_pred_test)
ridge_out_sample_r2 <- out_of_sample_r2(y_test, ridge_pred_test)

comparison <- data.frame(
  Model = c("LASSO", "Ridge"),
  In_sample_R2 = c(lasso_in_sample_r2, ridge_in_sample_r2),
  Out_sample_R2 = c(lasso_out_sample_r2, ridge_out_sample_r2)
)
comparison
```

Model with Treatment Variables

```r
X_treatment <- model.matrix(log_price~.-1,train_data)
Y_treatment <- train_data$log_price

X_test <- model.matrix(log_price~.-1,test_data)
Y_test <- test_data$log_price

cv_fit_treatment <- cv.glmnet(X_treatment, Y_treatment, family = "gaussian", alpha = 1)
cv_fit_test <- cv.glmnet(X_test,Y_test, family = "gaussian", alpha = 1)

# Coefficients for the best lambda value
coefficients_best_lambda_treatment <- coef(cv_fit_treatment, s = "lambda.min")
head(coefficients_best_lambda_treatment, 20)

# Coefficients for the lambda value selected by 1 standard error rule
coefficients_1se_treatment <- coef(cv_fit_treatment, s = "lambda.1se")
head(coefficients_1se_treatment,20)

treatment_pred_train <- predict(cv_fit_treatment, s = "lambda.min", newx = X_treatment)
treatment_pred_test <- predict(cv_fit_treatment, s = "lambda.min", newx = X_test)

in_sample_r2_treatment <- in_sample_r2(Y_treatment, treatment_pred_train)
in_sample_r2_treatment
out_of_sample_r2_treatment <- out_of_sample_r2(Y_test, treatment_pred_test)
out_of_sample_r2_treatment
```

```r
comparison_treatment <- data.frame(
  Model = "Treatment Model",
  In_sample_R2 = in_sample_r2_treatment,
  Out_sample_R2 = out_of_sample_r2_treatment
)
comparison_treatment
```

NAIVE LASSO

```r
set.seed(123)

X_treatment <- model.matrix(log_price ~ . - 1, train_data)
Y_treatment <- train_data$log_price

X_test <- model.matrix(log_price ~ . - 1, test_data)
Y_test <- test_data$log_price

naive_lasso_fit <- glmnet(X_treatment, Y_treatment, family = "gaussian", alpha = 1)
lambda_values <- naive_lasso_fit$lambda
coefficients_best_lambda_treatment <- coef(naive_lasso_fit, s = min(lambda_values))
head(coefficients_best_lambda_treatment, 20)

# Coefficients for the lambda value selected by 1 standard error rule
lambda_value <- cv_fit_treatment$lambda.min
treatment_pred_lambda <- predict(cv_fit_treatment, s = lambda_value, newx = X_treatment)
coefficients_1se_treatment <- coef(naive_lasso_fit, s = lambda_value)
head(coefficients_1se_treatment, 20)

treatment_pred_train <- predict(naive_lasso_fit, s = min(lambda_values), newx = X_treatment)
treatment_pred_test <- predict(naive_lasso_fit, s = min(lambda_values), newx = X_test)

in_sample_r2_treatment <- in_sample_r2(Y_treatment, treatment_pred_train)
out_of_sample_r2_treatment <- out_of_sample_r2(Y_test, treatment_pred_test)

comparison_treatment <- data.frame(
  Model = "Naive LASSO Model",
  In_sample_R2 = in_sample_r2_treatment,
  Out_sample_R2 = out_of_sample_r2_treatment
)
comparison_treatment
```

# 5. CART Model, Random Forest, and Gradient Boosting

## Data Processing

```
#################### DATA Processing ####################

# read Airbnb data
data <- read.csv("Airbnb_Data.csv")

# delete id and text variables
data <- select(data, -c(id, description, name))

# convert categorical variables to factors
data$room_type <- as.factor(data$room_type)
data$property_type <- as.factor(data$property_type)
data$bed_type <- as.factor(data$bed_type)
data$cancellation_policy <- as.factor(data$cancellation_policy)
data$city <- as.factor(data$city)
data$host_has_profile_pic <- as.factor(data$host_has_profile_pic)
data$host_identity_verified <- as.factor(data$host_identity_verified)
data$host_response_rate <- as.factor(data$host_response_rate)
data$instant_bookable <- as.factor(data$instant_bookable)

# convert host_response_rate to numeric by removing the '%' sign
data$host_response_rate <- as.numeric(gsub("%", "", data$host_response_rate))

# handle amenities variable: transform into multiple binary columns
data$amenities <- str_replace_all(data$amenities, '[{}"]', '')
amenities_list <- str_split(data$amenities, ",")
all_amenities <- unique(unlist(amenities_list))
for (amenity in all_amenities) {
  data[[amenity]] <- sapply(amenities_list, function(x) amenity %in% x)
}
data <- select(data, -amenities)

# handle date variables
data$first_review <- as.Date(data$first_review, format="%m/%d/%Y")
data$last_review <- as.Date(data$last_review, format="%m/%d/%Y")
data$host_since <- as.Date(data$host_since, format="%m/%d/%Y")

# transform date variables into more meaningful variables
data$days_since_first_review <- as.numeric(Sys.Date() - data$first_review)
data$days_since_last_review <- as.numeric(Sys.Date() - data$last_review)
data$host_duration <- as.numeric(Sys.Date() - data$host_since)
# delete first_review, last_review, host_since
data <- select(data, -c(first_review, last_review, host_since))

# turn thumbnail_url into a binary categorical variable
data$has_thumbnail <- ifelse(is.na(data$thumbnail_url) | data$thumbnail_url == "", FALSE, TRUE)
data <- select(data, -thumbnail_url) # delete thumbnail_url

# handle missing values
count_missing <- function(x) {
  sum(is.na(x) | x == "")}
```

```r
# create a summary of missing values (NA and empty strings) for each column
missing_values_summary <- data %>%
  summarise_all(count_missing) %>%
  gather(key = "variable", value = "missing_count") %>%
  arrange(desc(missing_count))
print(missing_values_summary)

# impute the above numerical variables that have missing values with median values
data$host_response_rate[is.na(data$host_response_rate)] <-
  median(data$host_response_rate, na.rm = TRUE)
data$review_scores_rating[is.na(data$review_scores_rating)] <-
  median(data$review_scores_rating, na.rm = TRUE)
data$days_since_first_review[is.na(data$days_since_first_review)] <-
  median(data$days_since_first_review, na.rm = TRUE)
data$days_since_last_review[is.na(data$days_since_last_review)] <-
  median(data$days_since_last_review, na.rm = TRUE)
data$bathrooms[is.na(data$bathrooms)] <-
  median(data$bathrooms, na.rm = TRUE)
data$host_duration[is.na(data$host_duration)] <-
  median(data$host_duration, na.rm = TRUE)
data$beds[is.na(data$beds)] <- median(data$beds, na.rm = TRUE)
data$bedrooms[is.na(data$bedrooms)] <-
  median(data$bedrooms, na.rm = TRUE)

# replace missing values with specific values for categorical columns
data$host_has_profile_pic[is.na(data$host_has_profile_pic)
                          | data$host_has_profile_pic == ''] <- 'f'
data$host_identity_verified[is.na(data$host_identity_verified)
                            | data$host_identity_verified == ''] <- 'f'
data$neighbourhood[is.na(data$neighbourhood)
                   | data$neighbourhood == ''] <- 'Unknown'
data$zipcode[is.na(data$zipcode)
             | data$zipcode == ''] <- 'Unknown'
# convert to factors
data$neighbourhood <- as.factor(data$neighbourhood)
data$zipcode <- as.factor(data$zipcode)

# make columns names unique and legal
names(data) <- make.names(names(data), unique = TRUE)

# extract numerical variable names
numeric_vars <- c("log_price", "accommodates", "bathrooms", "host_response_rate",
                  "latitude", "longitude", "number_of_reviews", "review_scores_rating",
                  "bedrooms", "beds", "days_since_first_review", "days_since_last_review",
                  "host_duration")
# extract categorical variable names
categorical_vars <- setdiff(names(data), numeric_vars)

# standardize numerical variables
data_numeric <- scale(data[numeric_vars])
data_numeric <- as.data.frame(data_numeric)
# combine standardized numerical variables with categorical variables
data_scale <- cbind(data_numeric, data[categorical_vars])
```

## CART Model

### Decision Tree

```
# Split the training data and testing data
set.seed(123)
trainIndex_dt <- createDataPartition(data$log_price, p = .8, list = FALSE, times = 1)
trainData_dt <- data[trainIndex_dt, ]
testData_dt <- data[-trainIndex_dt, ]

# Construct the decision tree model
tree_model <- rpart(log_price ~ ., data = trainData_dt, method = "anova")
```

```
# Visualize the decision tree
rpart.plot(tree_model)

# In-sample prediction and R^2
train_predictions <- predict(tree_model, newdata = trainData_dt)
train_r2 <- 1 - sum((trainData_dt$log_price - train_predictions)^2)
            / sum((trainData_dt$log_price - mean(trainData_dt$log_price))^2)
print(paste("Decision Tree In-sample R^2:", train_r2))

# Out-of-sample prediction and R^2
tree_predictions <- predict(tree_model, newdata = testData_dt)
tree_r2 <- 1 - sum((testData_dt$log_price - tree_predictions)^2)
            / sum((testData_dt$log_price - mean(testData_dt$log_price))^2)
print(paste("Decision Tree Out-of-sample R^2:", tree_r2))

# Calculate RMSE for out-of-sample
tree_rmse <- sqrt(mean((testData_dt$log_price - tree_predictions)^2))
print(paste("Decision Tree RMSE:", tree_rmse))
```

### Feature Importance of Decision Tree

```
# Calculate importance of features
var_importance <- varImp(tree_model)

# Order the importance and choose the non-zeros
importance_data <-
  data.frame(Variables = rownames(var_importance), Importance = var_importance$Overall)
importance_data <- importance_data[order(-importance_data$Importance), ]  # order
top10_importance <- head(importance_data, 10)  # choose top 10 variables

# Plot the importance graph
ggplot(top10_importance, aes(x = reorder(Variables, Importance), y = Importance)) +
  geom_bar(stat = "identity", color = "indianred1", fill = "indianred1") +
  coord_flip() +
  xlab("Variables") +
  ylab("Importance") +
  ggtitle("Top 10 Variable Importance in Decision Tree") +
  theme_minimal()
```

**Complexity Parameter (CP) Plot of Decision Tree**

```
# Visualize "cp" table/plot CV result
plotcp(tree_model, main = "Cross Validation Error vs. Complexity Parameter (CP)")


# Adjust "cp" to prune the tree
optimal_cp <- tree_model$cptable[which.min(tree_model$cptable[,"xerror"]),"CP"]
pruned_tree_model <- prune(tree_model, cp = optimal_cp)

# Use the pruned tree to do in-sample prediction
pruned_train_predictions <- predict(pruned_tree_model, newdata = trainData_dt)

pruned_train_r2 <-
  1 - sum((trainData_dt$log_price - pruned_train_predictions)^2)
  /sum((trainData_dt$log_price - mean(trainData_dt$log_price))^2)

print(paste("Pruned Decision Tree In-sample R^2:", pruned_train_r2))

# Use the pruned tree to do out-of-sample prediction
pruned_tree_predictions <- predict(pruned_tree_model, newdata = testData_dt)
pruned_tree_r2 <-
  1 - sum((testData_dt$log_price - pruned_tree_predictions)^2)
  / sum((testData_dt$log_price - mean(testData_dt$log_price))^2)

print(paste("Pruned Decision Tree Out-of-sample R^2:", pruned_tree_r2))

# Calculate RMSE for pruned tree
pruned_tree_rmse <- sqrt(mean((testData_dt$log_price - pruned_tree_predictions)^2))
print(paste("Pruned Decision Tree RMSE:", pruned_tree_rmse))
```

## Random Forest

```
#################### Random Forest ####################

# delete zipcode and neighbourhood, because they have too much
# categories that rf model does not support
sample_data <- select(data, -c(zipcode, neighbourhood))

# split training and testing set
set.seed(123)
trainIndex_rf <- createDataPartition(sample_data$log_price, p = .8, list = FALSE, times = 1)
trainData_rf <- sample_data[trainIndex_rf, ]
testData_rf <- sample_data[-trainIndex_rf, ]

# construct a ramdom forest model
rf <- randomForest(log_price ~ ., data = trainData_rf, ntree = 250, nodesize = 25, importance = TRUE)
```

```
# plot the rf: this is an error graph instead of the tree visualization
plot(rf)

# calculate in-sample R2
in_sample_predictions <- predict(rf, trainData_rf)
in_sample_r2 <- 1 - sum((trainData_rf$log_price - in_sample_predictions)^2)
```

```
/ sum((trainData_rf$log_price - mean(trainData_rf$log_price))^2)

print(paste("In-sample R2:", in_sample_r2))

# calculate out-of-sample R2
out_sample_predictions <- predict(rf, testData_rf)
out_sample_r2 <- 1 - sum((testData_rf$log_price - out_sample_predictions)^2)
/ sum((testData_rf$log_price - mean(testData_rf$log_price))^2)

print(paste("Out-of-sample R2:", out_sample_r2))
```

**Feature Importance of Random Forest**

```
# plot the importance of features
varImpPlot(rf, type=1, pch=21, bg="indianred1", main='RF variable importance')
```

**Actual vs. Predicted Values of Random Forest**

```
# predict using rf
predicted <- predict(rf, newdata = sample_data)

# calculate residuals
residuals <- sample_data$log_price - predicted

# prepare data for ggplot
rf_graph_data <- data.frame(
  Actual = sample_data$log_price,
  Predicted = predicted,
  Residuals = residuals
)

range_vals_rf <- range(c(rf_graph_data$Actual, rf_graph_data$Predicted))

# plot predicted values vs. actual values & residual plot
p1 <- ggplot(rf_graph_data, aes(x = Actual, y = Predicted)) +
  geom_point(shape = 21, color = "indianred1", fill = "mistyrose1", size = 1.5, alpha = 0.15) +
  geom_abline(intercept = 0, slope = 1, color = "black", linewidth = 0.6) +
  labs(title = "Actual VS. Predicted (RF)",
       x = "Actual Value (log_price)", y = "RF Predicted Value (log_price)") +
  xlim(range_vals) + ylim(range_vals) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))

p2 <- ggplot(rf_graph_data, aes(x = Predicted, y = Residuals)) +
  geom_point(shape = 21, color = "indianred1", fill = "mistyrose1", size = 1.5, alpha = 0.15) +
  geom_hline(yintercept = 0, color = "black", linewidth = 0.6) +
  labs(title = "Residual Plot of Random Forest",
       x = "Predicted Value (RF) (log_price)", y = "Residual") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))

grid.arrange(p1, p2, ncol = 2)
```

## XGBOOST

```
#################### XGBOOST ########################

# split training and testing data
set.seed(123)
trainIndex_xg <- createDataPartition(data$log_price, p = .8, list = FALSE, times = 1)
trainData_xg <- data[trainIndex_xg, ]
testData_xg <- data[-trainIndex_xg, ]

# turn factor variables into one-hot-code
train_matrix <- model.matrix(log_price ~ . - 1, data = trainData_xg)
test_matrix <- model.matrix(log_price ~ . - 1, data = testData_xg)

# prepare data matrix
train_label <- trainData_xg$log_price
test_label <- testData_xg$log_price
dtrain <- xgb.DMatrix(data = train_matrix, label = train_label)
dtest <- xgb.DMatrix(data = test_matrix, label = test_label)

# set XGBoost model parameters
params <- list(
  objective = "reg:squarederror",
  eta = 0.1,
  max_depth = 6,
  eval_metric = "rmse"
)
```

```
# train the model
xgb_model <- xgb.train(
  params = params,
  data = dtrain,
  nrounds = 250,
  watchlist = list(train = dtrain, eval = dtest),
  early_stopping_rounds = 10,
  print_every_n = 10
)
```

```
# calculate in-sample R2
xgb_train_predictions <- predict(xgb_model, dtrain)
xgb_train_r2 <- 1 - sum((train_label - xgb_train_predictions)^2)
                / sum((train_label - mean(train_label))^2)
print(paste("XGBoost In-sample R2:", xgb_train_r2))

# calculate out-of-sample R2
xgb_test_predictions <- predict(xgb_model, dtest)
xgb_test_r2 <- 1 - sum((test_label - xgb_test_predictions)^2)
                / sum((test_label - mean(test_label))^2)
print(paste("XGBoost Out-of-sample R2:", xgb_test_r2))
```

### Feature Importance of XGBoost

```
# obtain the feature importance and plot it
importance_matrix <- as.data.frame(head(xgb.importance(
```

```
  feature_names = colnames(train_matrix), model = xgb_model), 20))
ggplot(importance_matrix, aes(x = reorder(Feature, Gain), y = Gain)) +
  geom_bar(stat = "identity", fill = "indianred1") +
  coord_flip() +
  xlab("Features") +
  ylab("Importance (Gain)") +
  ggtitle("Top 20 Important Features") +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold")
  )
```

**Actual vs. Predicted Values of XGBoost**

```
# prepare data for scatter plot
xg_graph_data <- data.frame(
  Actual = test_label,
  Predicted = xgb_test_predictions,
  Residuals = test_label - xgb_test_predictions
)

range_vals_xg <- range(c(xg_graph_data$Actual, xg_graph_data$Predicted))

# predicted vs. actual
p3 <- ggplot(xg_graph_data, aes(x = Actual, y = Predicted)) +
  geom_point(shape = 21, color = "indianred1", fill = "mistyrose1", size = 1.5, alpha = 0.15) +
  geom_abline(intercept = 0, slope = 1, color = "black", linewidth = 0.6) +
  labs(title = "Actual VS. Predicted (XGBoost)",
       x = "Actual Value (log_price)", y = "XGBoost Predicted Value (log_price)") +
  xlim(range_vals_xg) + ylim(range_vals_xg) +
  theme_minimal() +
  coord_fixed(ratio = 1) +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold")
  )

# residual plot
p4 <- ggplot(xg_graph_data, aes(x = Predicted, y = Residuals)) +
  geom_point(shape = 21, color = "indianred1", fill = "mistyrose1", size = 1.5, alpha = 0.15) +
  geom_hline(yintercept = 0, color = "black", linewidth = 0.6) +
  labs(title = "Residual Plot of XGBoost",
       x = "Predicted Value (XGBoost) (log_price)", y = "Residual") +
  theme_minimal() +
  coord_fixed(ratio = 1) +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold")
  )

grid.arrange(p3, p4, ncol = 2)
```