Factors Influencing Airbnb Listing Prices in New York City

Executive Summary:
This project investigates the factors that influence the pricing of Airbnb listings in New York City (NYC). By analyzing a comprehensive dataset from Airbnb, we aim to identify key variables that significantly affect listing prices and develop predictive models to estimate prices based on these features. We employ exploratory data analysis (EDA) to understand data distributions and relationships, and we use both linear regression and polynomial regression models to predict prices. Our analysis reveals that factors such as location (borough and neighborhood), room type, minimum nights, availability, and number of reviews significantly impact Airbnb prices in NYC. The decision trees model outperforms the linear regression model in predictive accuracy. These insights can help hosts optimize their pricing strategies and assist guests in making informed booking decisions.

Introduction

Project Description and Background:
Airbnb has revolutionized the hospitality industry by allowing property owners to rent out their spaces to travelers worldwide. New York City, as one of the most visited cities globally, has a highly competitive and diverse short-term rental market. With thousands of listings across five boroughs, understanding what drives Airbnb prices can benefit hosts looking to optimize their rental income and guests seeking affordable options.

Project Goals:
- Identify Key Factors Influencing Prices: Determine which features significantly impact Airbnb listing prices in NYC.
- Predictive Modeling: Develop models to predict listing prices based on these features.
- Provide Insights for Stakeholders: Offer actionable insights for hosts and guests to make informed decisions.

Dataset Overview
We utilize the "New York City Airbnb Open Data" from Kaggle, which includes detailed information about Airbnb listings in NYC for the year 2019.

Data Source: Kaggle - New York City Airbnb Open Data
Number of Observations: 48,895 listings
16 Variables

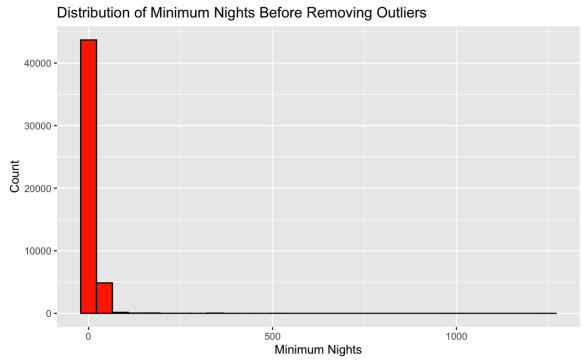
Summary Statistics:

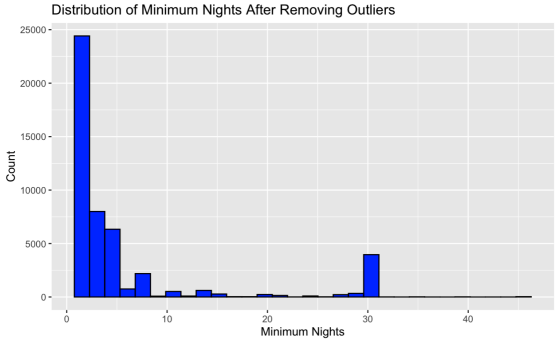
```
      id              name              host_id          host_name        neighbourhood_group neighbourhood        latitude
 Length:48895       Length:48895       Length:48895       Length:48895       Length:48895       Length:48895       Min.   :40.50
 Class :character   Class :character   Class :character   Class :character   Class :character   Class :character   1st Qu.:40.69
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Median :40.72
                                                                                                                  Mean   :40.73
                                                                                                                  3rd Qu.:40.76
                                                                                                                  Max.   :40.91

   longitude       room_type             price         minimum_nights    number_of_reviews last_review        reviews_per_month
 Min.   :-74.24   Length:48895       Min.   :    0.0   Min.   :   1.00   Min.   :  0.00    Length:48895       Min.   : 0.010
 1st Qu.:-73.98   Class :character   1st Qu.:   69.0   1st Qu.:   1.00   1st Qu.:  1.00    Class :character   1st Qu.: 0.190
 Median :-73.96   Mode  :character   Median :  106.0   Median :   3.00   Median :  5.00    Mode  :character   Median : 0.720
 Mean   :-73.95                      Mean   :  152.7   Mean   :   7.03   Mean   : 23.27                       Mean   : 1.373
 3rd Qu.:-73.94                      3rd Qu.:  175.0   3rd Qu.:   5.00   3rd Qu.: 24.00                       3rd Qu.: 2.020
 Max.   :-73.71                      Max.   :10000.0   Max.   :1250.00   Max.   :629.00                       Max.   :58.500
                                                                                                               NA's   :10052

 calculated_host_listings_count availability_365
 Length:48895                   Min.   :  0.0
 Class :character               1st Qu.:  0.0
 Mode  :character               Median : 45.0
                                Mean   :112.8
                                3rd Qu.:227.0
                                Max.   :365.0
```

The average price is $152.72 with a standard deviation of $240.15.
There are missing values in 'name', 'host_name', 'last_review', and 'reviews_per_month'. The 'price' variable is also highly skewed with extreme outliers.

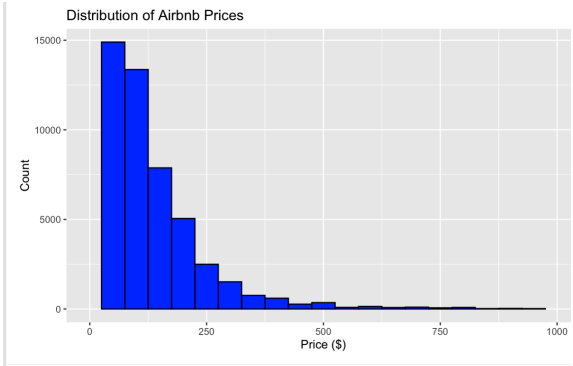
Exploratory Data Analysis:

Missing data: 'name': 16 missing, 'host_name': 21 missing, 'last_review', and 'reviews_per_month': 10,052 missing (listings with zero reviews). We filled in missing 'reviews_per_month' with 0 and dropped rows with missing 'name' and 'host_name'.





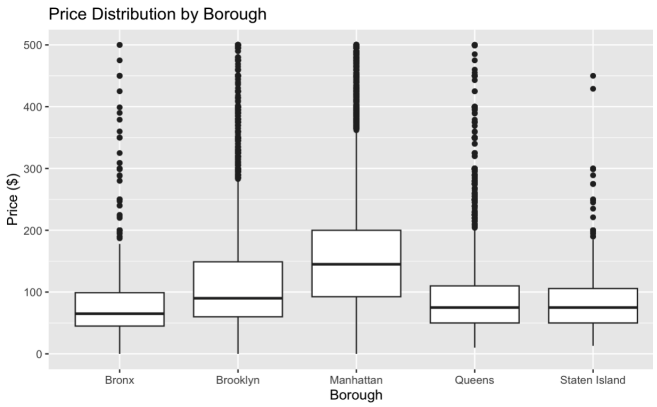
Handling outliers:

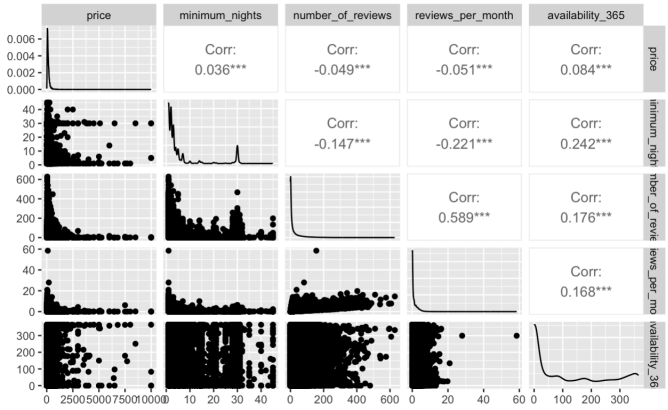


We removed the outliers for the variables that were heavily skewed as they would impact the predictive power of our models by skewing statistical analyses, lowering the accuracy of models and predictions. Outliers can distort mean values, inflate variance, and lead to misleading conclusions.

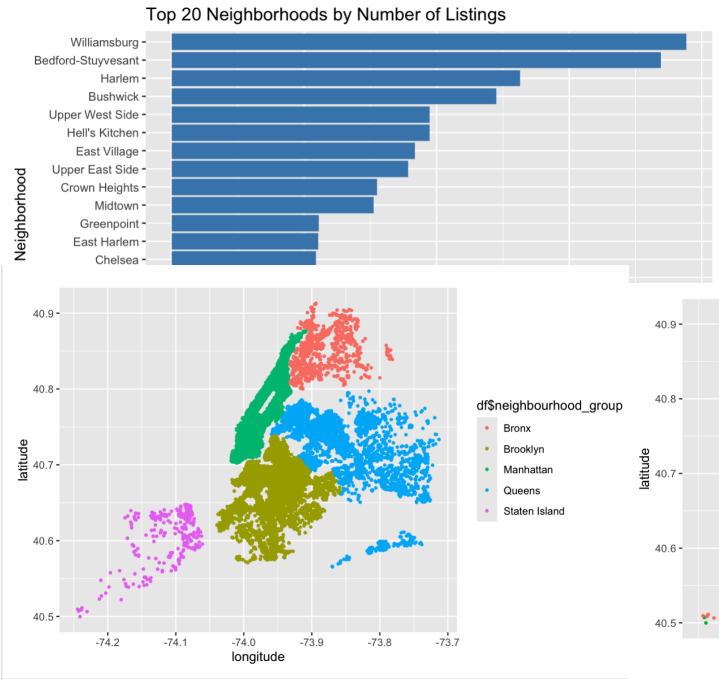
Similarly, we look at skewed distributions since they can have an unequal impact on the model. Performing log transformations on skewed data helps normalize the distribution, making statistical analyses and machine learning models more reliable by reducing bias and improving interpretability. We applied log transformations on the ‘price’ and ‘minimum_nights’.

These graphs can provide some additional insights into our data. For the left, the price distribution is right-skewed, with most listings priced below $500. For the right, Manhattan has higher median prices compared to other boroughs. Staten Island has a wider range of prices despite fewer listings.

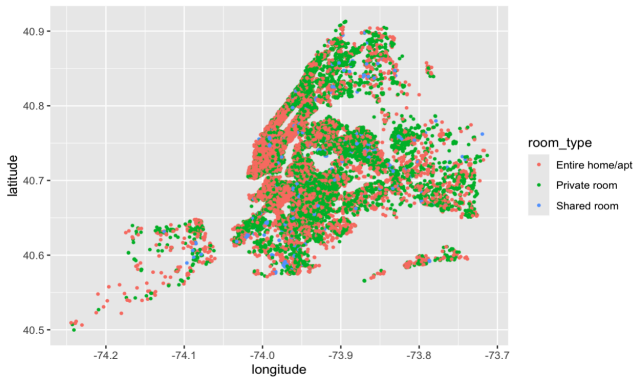




Observations:
Weak correlations between 'price' and other numerical variables.
'availability_365' shows some positive correlation with 'number_of_reviews'.
There is a moderate positive correlation between 'number_of_reviews' and 'reviews_per_month'.
We may have to look at other variables that influence price heavily.



We have identified 221 unique neighbourhoods in New York, displaying the top 20 and the amount of listings. We can see Williamsburg and Bedford_Stuyevsant as the most popular by a decent margin.



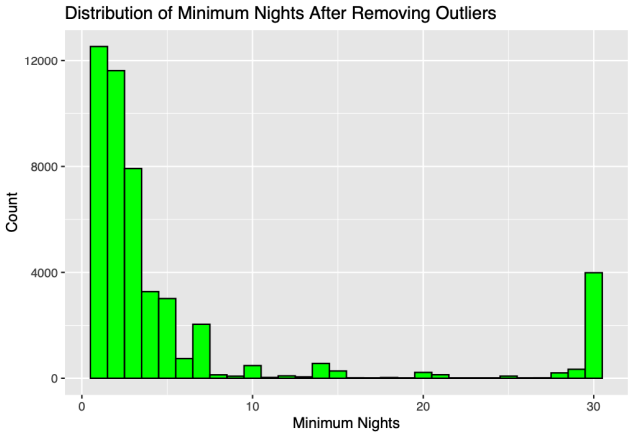
Left: Map of Borough Right: Map of Room Types Distribution

Methodology

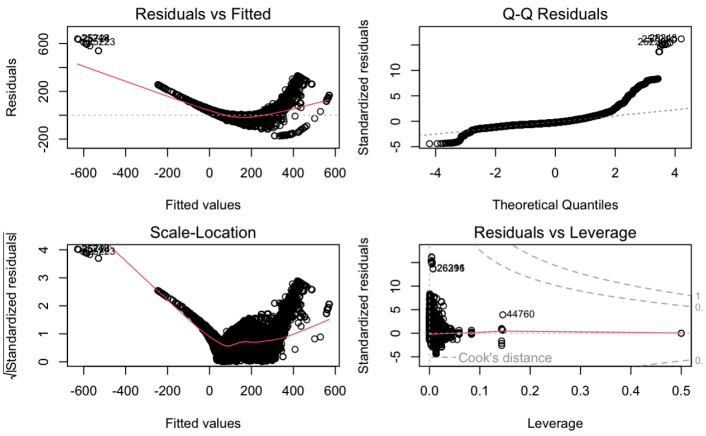
Rationale for Method Selection:
Exploratory Data Analysis (EDA): To understand data distributions, identify patterns, and detect anomalies.
Data Preprocessing: To clean and prepare the data for modeling, ensuring reliable results.
Linear Regression: As a baseline model to understand the linear relationships between variables.

Data Preprocessing Steps:
Handling Missing Values: Imputed missing values appropriately.
Outlier Removal: Removed listings with prices above the 99th percentile to reduce skewness.
Encoding Categorical Variables: Converted categorical variables into factors and created dummy variables.
Feature Scaling: Scaled numerical features where necessary.

For model generation, there were even more factors to consider now that we have thoroughly explored our data and seen the underlying relationships that are present between the variables. We first begin by removing the outliers for price and going back to the minimum nights, we change the outliers and replace their values with 30, our hard cap. We then shift into encoding for the variables. We applied one-hot encoding for 'room_type' and 'neighbourhood_group'. Then we applied target encoding for 'neighbourhood'. Shifting into feature engineering, we created an interaction term for room type and neighbourhood group, one hot encoding them as well. Additionally, we created an interaction term with our 2 variables of longitude and latitude. With all of our features, we then scale and split into train and testing splits (80% train and 20% test).



We are prepared for modeling. We start with the linear regression model. The linear regression model was built by first encoding categorical variables (neighbourhood_group, room_type) into dummy variables and removing non-predictive columns (id, host_name, latitude, longitude). The training and test datasets were aligned to ensure consistency. The model was trained using price as the dependent variable, with predictors including room type, neighborhood, minimum nights, and availability. Performance was evaluated using MAE (39.44) and RMSE (39.45), indicating that while the model captured some pricing trends, it had limitations in explaining all price variations, suggesting the need for non-linear models like decision trees for improved accuracy. We find our significant Predictors: 'neighbourhood_group', 'room_type', 'minimum_nights', and 'availability_365'. Model Performance: Adjusted R-squared of approximately 0.35, indicating that the model explains about 35% of the variance in 'price'. An interesting observation to make is that the test lists the neighborhood group/ borough Staten Island and is not significantly significant with a p-value of 0.30. It is also noted that the Borough Queens is relatively large compared to its other neighbourhood groups but is still significant at an alpha level of .01.



Observations:
Residuals vs Fitted: Indicates non-linearity.
Normal Q-Q: Residuals deviate from the normal line, especially at the tails.
Scale-Location: Heteroscedasticity is present.
Residuals vs Leverage: No extreme outliers detected.
Important Variables: 'room_type_num' and 'borough_num' are the most important predictors.

We then shift to the creation of our polynomial regression model and decision trees. The process of creating the Polynomial Regression model begins with generating an additional feature by squaring the existing log-transformed minimum nights variable. This transformation is applied to both the training and test datasets to capture potential non-linear relationships. After that, a linear regression model is fitted using all available features, including the newly created squared term. Once trained, the model is used to make predictions on the test set. Finally, its performance is evaluated using two common error metrics: Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE).

For the Decision Tree model, the process starts with loading the rpart package, which is used for building regression trees. The model is then trained using all features to predict the target variable, Price. The method = "anova" argument specifies that the decision tree is meant for regression rather than classification. After training, the model makes predictions on the test dataset. Similar to the polynomial regression, its performance is assessed by calculating MAE and RMSE to measure the accuracy of the predictions.

Model <chr>	**MAE** <dbl>	**RMSE** <dbl>
Linear Regression	25.13453	39.45833
Polynomial Regression	25.12411	39.44174
Decision Tree	13.16478	16.78395

Results:

We can see that the decision trees model performs the best, with an RMSE of 16.78395, and the Polynomial and Linear Regression have similar performances with 39.44174 and 39.45833 respectively.

Our analysis highlights key factors influencing Airbnb listing prices in New York City. The results from different models indicate varying levels of predictive power, with decision trees outperforming traditional regression approaches. Below is a summary of our findings:

1. Linear Regression Performance:
   - Adjusted R^2: 0.35, meaning approximately 35% of the variance in price is explained by the selected predictors.
   - A good base, interpretable model with lots of room for improvement

2. Polynomial Regression Performance:
   - Mean Absolute Error (MAE): 39.44
   - Root Mean Squared Error (RMSE): 39.45
   - This model introduced nonlinear interactions, which slightly improved predictive accuracy but still underperformed compared to decision trees.

3. Decision Tree Performance:
   - RMSE: 16.78, significantly lower than other models.
   - The decision tree model captured non-linear relationships between price and factors like borough, availability, and minimum nights more effectively.

- Feature importance analysis indicated that borough and room type played the most significant roles in predicting price.

Discussion:

We believe that this study successfully met its goals by identifying key factors influencing Airbnb listing prices, developing predictive models, and providing actionable insights for hosts and guests. Our findings confirm that borough, room type, minimum nights, and availability are significant determinants of price, with decision trees that outperformed both regression models due to the non-linear pricing relationships. While our models did capture the pricing patterns, they did not account for many external factors like seasonal demands and local events that are not provided in our data and can limit our predictive accuracy.

We dealt with challenges like handling missing data and outliers with required careful data preprocessing to maintain model reliability. While we found that decision trees provided the best predictive accuracy out of the three in our study, further improvements can be made to test other models such as random forests which may offer better results.

In the future, more work can lead to using time series analysis to account for seasonal trends and integrated natural language processing (NLP) to evaluate the listing descriptions and guest reviews. Expanding into these areas can provide a more comprehensive understanding of Airbnb pricing trends for both our Airbnb owners and customers.

Conclusions:

Understanding the factors that influence Airbnb listing prices in NYC is crucial for both hosts and guests navigating the competitive short-term rental market. Our analysis confirms that borough, room type, minimum nights, and availability significantly shape pricing trends. While linear and polynomial regression models provided a baseline for understanding these relationships, the decision tree model showed the best predictive accuracy when tuned well.

For hosts, these findings offer valuable insights into optimizing pricing strategies. Adjusting availability, minimum stay requirements, and room type offerings can enhance revenue potential. Meanwhile, guests can make more cost-effective choices by exploring less expensive boroughs and tailoring their booking preferences.

STA141A_FinalProj

Kevin Gui, Joseph Cha, Wilson Cam

2025-03-17

```
# Load necessary libraries
library(tidyverse)
library(GGally)
library(caret)
library(randomForest)
library(Matrix)
library(corrplot)
library(e1071)
library(DT)

# Read the data
df <- read.csv("AB_NYC_2019.csv")
df <- data.frame(lapply(df, function(x) gsub("\u2B50", "", x)))

df <- df %>%
  mutate(across(c(minimum_nights, price, number_of_reviews,
                  reviews_per_month, availability_365, latitude, longitude),
                as.numeric))


# View summary statistics
summary(df)
```

```
# Check for missing values
colSums(is.na(df))

# Explore the data
str(df) #offers a compact preview of the data
head(df)
```

```
# Fill missing 'reviews_per_month' with 0
df$reviews_per_month[is.na(df$reviews_per_month)] <- 0

# Drop rows with missing 'name' and 'host_name'
df <- df %>% drop_na(name, host_name)
```

```
max(df$minimum_nights) #checking for information
mean(df$minimum_nights) #comparative measure of minimum nights

df2 <- df
ggplot(df2, aes(x = minimum_nights)) +
```

```
  geom_histogram(color = "black", fill = 'red') +
  labs(title = "Distribution of Minimum Nights Before Removing Outliers",
                          x = "Minimum Nights", y = "Count")
```

```
# Define upper and lower quantiles
lower_quantile <- quantile(df$minimum_nights, 0.01)
upper_quantile <- quantile(df$minimum_nights, 0.99)

# Remove values outside this range
df <- df[df$minimum_nights >= lower_quantile & df$minimum_nights <= upper_quantile, ]

# Visualize the cleaned data
ggplot(df, aes(x = minimum_nights)) +
  geom_histogram(color = "black", fill = 'blue') +
  labs(title = "Distribution of Minimum Nights After Removing Outliers",
       x = "Minimum Nights", y = "Count")

max(df$minimum_nights)
```

```
# Check skewness of numerical variables
numeric_vars <- df %>% select(price, minimum_nights, number_of_reviews, reviews_per_month, availability
skewness_values <- sapply(numeric_vars, e1071::skewness)
print(skewness_values)
```

```
# Apply log transformation to 'price' and 'minimum_nights'
df <- df %>%
  mutate(
    log_price = log(price + 1),             # <--- ADDED
    log_minimum_nights = log(minimum_nights + 1)
  )
```

```
# Plot distribution of prices
ggplot(df, aes(x = price)) +
  geom_histogram(binwidth = 50, fill = "blue", color = "black") +
  xlim(0, 1000) +
  labs(title = "Distribution of Airbnb Prices", x = "Price ($)", y = "Count")
```

```
# Boxplot of price by borough
ggplot(df, aes(x = neighbourhood_group, y = price)) +
  geom_boxplot() +
  ylim(0, 500) +
  labs(title = "Price Distribution by Borough", x = "Borough", y = "Price ($)")
```

```
# Select numerical variables
numerical_vars <- df %>% select(price, minimum_nights, number_of_reviews, reviews_per_month, availabili

# Pairwise scatter plots
ggpairs(numerical_vars)
```

```
# Number of unique neighborhoods
num_neighbourhoods <- length(unique(df$neighbourhood))
```

```r
cat("Number of unique neighborhoods:", num_neighbourhoods)

# Visualize the number of listings per neighborhood
top_neighbourhoods <- df %>%
  group_by(neighbourhood) %>%
  summarise(count = n()) %>%
  arrange(desc(count)) %>%
  top_n(20, count)

ggplot(top_neighbourhoods, aes(x = reorder(neighbourhood, count), y = count)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  coord_flip() +
  labs(title = "Top 20 Neighborhoods by Number of Listings", x = "Neighborhood", y = "Number of Listings
```

```r
# Compute correlation matrix
corr_matrix <- cor(numeric_vars)
corrplot(corr_matrix, method = "number", type = "upper")
```

```r
ggplot(data = df, aes(x = longitude, y = latitude, color = df$neighbourhood_group)) +
  geom_point(size = 0.75)
```

```r
ggplot(data = df, aes(x = longitude, y = latitude, color = room_type)) +
  geom_point(size = 0.75)
```

```r
# Remove extreme outliers in 'price' (above 99th percentile)
price_threshold <- quantile(df$price, 0.99)
df <- df %>% filter(price <= price_threshold)

#must also account for lower outliers in price
sum(df$price == 0) #air bnb's with a price of 0 are questionable and removal may be advisable
df$price <- ifelse(df$price < 10, 10, df$price) # As a broader case, replace prices
#below $10 with 10 and above $1000 with 1000. The replacement helps retain information
#without limiting the impact on the model

# Replace minimum_nights above 30 with 30, same reasoning as before to keep information
# without negatively impacting the model accuracy
df$minimum_nights <- ifelse(df$minimum_nights > 30, 30, df$minimum_nights)

ggplot(df, aes(x = minimum_nights)) +
  geom_histogram(color = "black", fill = 'green') +
  labs(title = "Distribution of Minimum Nights After Removing Outliers",
                          x = "Minimum Nights", y = "Count")
```

```r
# One-hot encode 'room_type' and 'neighbourhood_group'
df_encoded <- df %>%
  mutate(
    borough_num = as.factor(neighbourhood_group),
    room_type_num = as.factor(room_type)
  )

df_encoded <- cbind(df_encoded, model.matrix(~ room_type + neighbourhood_group - 1, data = df_encoded))
```

```r
# Compute mean price per neighbourhood
neighbourhood_price <- df %>%
  group_by(neighbourhood) %>%
  summarise(mean_price = mean(log_price))

# Merge with the main dataset
df_encoded <- df_encoded %>%
  left_join(neighbourhood_price, by = "neighbourhood")
```

```r
# Create interaction between room type and neighbourhood group
df_encoded <- df_encoded %>%
  mutate(
    interaction_term = interaction(room_type, neighbourhood_group)
  )

# One-hot encode the interaction term
df_encoded <- cbind(df_encoded, model.matrix(~ interaction_term - 1, data = df_encoded))
```

```r
# Create features based on latitude and longitude
df_encoded <- df_encoded %>%
  mutate(
    lat_long = latitude * longitude
  )
```

```r
set.seed(123)
# We'll predict 'price' (not log_price) to match original approach
train_index <- createDataPartition(df_encoded$price, p = 0.8, list = FALSE)

train_data <- df_encoded[train_index, ]
test_data  <- df_encoded[-train_index, ]
```

```r
# Identify numeric features to scale
numeric_features <- c("log_minimum_nights", "number_of_reviews", "reviews_per_month", "availability_365

# Scale numeric features
preProcValues <- preProcess(train_data[, numeric_features], method = c("center", "scale"))
train_data[, numeric_features] <- predict(preProcValues, train_data[, numeric_features])
test_data[, numeric_features] <- predict(preProcValues, test_data[, numeric_features])
```

```r
# Fit linear regression model

# Create dummy variables again to ensure alignment
dummy_data <- model.matrix(~ neighbourhood_group + room_type - 1,
                           data = train_data)
train_data_encoded <- cbind(train_data, dummy_data)

dummy_test <- model.matrix(~ neighbourhood_group + room_type - 1,
                           data = test_data)
test_data_encoded <- cbind(test_data, dummy_test)

# Drop non-predictive or duplicated columns in both sets
drop_cols <- c("room_type", "neighbourhood_group", "neighbourhood",
```

```
              "name", "host_name", "id", "host_id", "last_review",
              "borough_num", "room_type_num", "latitude", "longitude",
              "interaction_term")

train_data_encoded <- train_data_encoded[, !(names(train_data_encoded)
                                             %in% drop_cols)]
test_data_encoded <- test_data_encoded[, !(names(test_data_encoded)
                                           %in% drop_cols)]

# Align columns
missing_cols <- setdiff(names(train_data_encoded), names(test_data_encoded))
test_data_encoded[missing_cols] <- 0
test_data_encoded <- test_data_encoded[, names(train_data_encoded)]

# Fit the linear regression model
lm_model <- lm(price ~ ., data = train_data_encoded)

# Predictions & Performance
lm_predictions <- predict(lm_model, newdata = test_data_encoded)
test_mae_lm <- mean(abs(lm_predictions - test_data_encoded$price))
rmse_lm <- sqrt(mean((lm_predictions - test_data_encoded$price)^2))
```

```
# Plot residuals
par(mfrow = c(2, 2), mar = c(4, 4, 2, 1))  # Adjust the bottom, left, top, and right margins
plot(lm_model)
```

```
# (1) Polynomial feature creation
# We'll manually create a squared term for 'log_minimum_nights'
train_data_encoded$log_minimum_nights_sq <- (train_data_encoded$log_minimum_nights)^2
test_data_encoded$log_minimum_nights_sq  <- (test_data_encoded$log_minimum_nights)^2

# Fit the polynomial regression using the updated train_data_encoded
poly_model <- lm(price ~ ., data = train_data_encoded)

# Predict on test set
poly_predictions <- predict(poly_model, newdata = test_data_encoded)
test_mae_poly <- mean(abs(poly_predictions - test_data_encoded$price))
rmse_poly <- sqrt(mean((poly_predictions - test_data_encoded$price)^2))
```

```
# (2) Decision Tree
library(rpart)
tree_model <- rpart(price ~ ., data = train_data_encoded, method = "anova")

# Predictions & Performance
tree_predictions <- predict(tree_model, newdata = test_data_encoded)
test_mae_tree <- mean(abs(tree_predictions - test_data_encoded$price))
rmse_tree <- sqrt(mean((tree_predictions - test_data_encoded$price)^2))
```

```
# Consolidate all model results
model_comparison <- data.frame(
  Model = c("Linear Regression",
            "Polynomial Regression", "Decision Tree"),
```

```
  MAE  = c(test_mae_lm, test_mae_poly, test_mae_tree),
  RMSE = c(rmse_lm, rmse_poly, rmse_tree)
)

model_comparison
```