# Exploring Factors Affecting Airbnb Rental Prices in New York City: A Data Analysis and Modeling Approach

Syed Ali Irtaza

## Introduction

Airbnb is a popular alternative to traditional hotel stays, offering travelers the opportunity to stay in unique accommodations and experience new neighborhoods. With growing popularity, the prices of Airbnb rentals have become an important consideration for hosts and guests. In this project, we aim to identify key factors that affect the price of Airbnb rentals using a dataset of listings in New York City. By analyzing these factors, we can gain insights into what drives rental prices in different markets and develop strategies for setting competitive rental rates.

## Exploratory Data Analysis (EDA)

After reading the data, one of the most important steps in any data analysis project is data cleaning and wrangling. Data needs to be checked for any missing values or duplicate values and delete rows that have missing values.

```
my_data <- read.csv("~/Desktop/Classes/STA 4102/AB_NYC_2019.csv")

# Calculating the number of rows
num_rows <- nrow(my_data)
print(num_rows)
```

```
## [1] 48895
```

```
# Checking for missing values and removing those rows
missing_values <- any(is.na(my_data))
print(missing_values)
```

```
## [1] TRUE
```

```
my_data <- na.omit(my_data)

# Checking for duplicate values and removing those rows
duplicate <- duplicated(my_data)
print(sum(duplicate))
```

```
## [1] 0
```

```r
# Calculating the number of rows after removing the duplicates and missing rows
n_rows <- nrow(my_data)
print(n_rows)
```

```
## [1] 38843
```

In our dataset, we did not have any duplicates but we found some missing values which need to be removed as it might affect the accuracy and reliability of the analysis and the predictive model that we build using the data. By doing so we are improving the quality of the data.

The next step is to perform the Exploratory Data Analysis (EDA) to gain insights into the dataset. This will help us better understand our dataset by summarizing its main characteristics and identifying any patterns or trends that may exist.

```r
summary(my_data)
```

```
##        id                 name              host_id           host_name
##  Min.   :    2539   Length:38843       Min.   :     2438   Length:38843
##  1st Qu.: 8720027   Class :character   1st Qu.:  7033824   Class :character
##  Median :18871455   Mode  :character   Median : 28371926   Mode  :character
##  Mean   :18096462                      Mean   : 64239145
##  3rd Qu.:27554820                      3rd Qu.:101846466
##  Max.   :36455809                      Max.   :273841667
##  neighbourhood_group neighbourhood         latitude       longitude
##  Length:38843        Length:38843       Min.   :40.51   Min.   :-74.24
##  Class :character    Class :character   1st Qu.:40.69   1st Qu.:-73.98
##  Mode  :character    Mode  :character   Median :40.72   Median :-73.95
##                                         Mean   :40.73   Mean   :-73.95
##                                         3rd Qu.:40.76   3rd Qu.:-73.94
##                                         Max.   :40.91   Max.   :-73.71
##   room_type             price         minimum_nights     number_of_reviews
##  Length:38843       Min.   :    0.0   Min.   :   1.000   Min.   :  1.0
##  Class :character   1st Qu.:   69.0   1st Qu.:   1.000   1st Qu.:  3.0
##  Mode  :character   Median :  101.0   Median :   2.000   Median :  9.0
##                     Mean   :  142.3   Mean   :   5.868   Mean   : 29.3
##                     3rd Qu.:  170.0   3rd Qu.:   4.000   3rd Qu.: 33.0
##                     Max.   :10000.0   Max.   :1250.000   Max.   :629.0
##  last_review        reviews_per_month calculated_host_listings_count
##  Length:38843       Min.   : 0.010    Min.   :  1.000
##  Class :character   1st Qu.: 0.190    1st Qu.:  1.000
##  Mode  :character   Median : 0.720    Median :  1.000
##                     Mean   : 1.373    Mean   :  5.165
##                     3rd Qu.: 2.020    3rd Qu.:  2.000
##                     Max.   :58.500    Max.   :327.000
##  availability_365
##  Min.   :  0.0
##  1st Qu.:  0.0
##  Median : 55.0
##  Mean   :114.9
##  3rd Qu.:229.0
##  Max.   :365.0
```

```
str(my_data)
```

```
## 'data.frame':    38843 obs. of  16 variables:
##  $ id                            : int  2539 2595 3831 5022 5099 5121 5178 5203 5238 5295 ...
##  $ name                          : chr  "Clean & quiet apt home by the park" "Skylit Midtown Castle"
##  $ host_id                       : int  2787 2845 4869 7192 7322 7356 8967 7490 7549 7702 ...
##  $ host_name                     : chr  "John" "Jennifer" "LisaRoxanne" "Laura" ...
##  $ neighbourhood_group           : chr  "Brooklyn" "Manhattan" "Brooklyn" "Manhattan" ...
##  $ neighbourhood                 : chr  "Kensington" "Midtown" "Clinton Hill" "East Harlem" ...
##  $ latitude                      : num  40.6 40.8 40.7 40.8 40.7 ...
##  $ longitude                     : num  -74 -74 -74 -73.9 -74 ...
##  $ room_type                     : chr  "Private room" "Entire home/apt" "Entire home/apt" "Entire h
##  $ price                         : int  149 225 89 80 200 60 79 79 150 135 ...
##  $ minimum_nights                : int  1 1 1 10 3 45 2 2 1 5 ...
##  $ number_of_reviews             : int  9 45 270 9 74 49 430 118 160 53 ...
##  $ last_review                   : chr  "10/19/18" "5/21/19" "7/5/19" "11/19/18" ...
##  $ reviews_per_month             : num  0.21 0.38 4.64 0.1 0.59 0.4 3.47 0.99 1.33 0.43 ...
##  $ calculated_host_listings_count: int  6 2 1 1 1 1 1 1 4 1 ...
##  $ availability_365              : int  365 355 194 0 129 0 220 0 188 6 ...
##  - attr(*, "na.action")= 'omit' Named int [1:10052] 3 20 27 37 39 194 205 261 266 268 ...
##   ..- attr(*, "names")= chr [1:10052] "3" "20" "27" "37" ...
```

```
head(my_data, n=5)
```

```
##     id                                              name host_id   host_name
## 1 2539               Clean & quiet apt home by the park    2787        John
## 2 2595                           Skylit Midtown Castle    2845    Jennifer
## 4 3831                 Cozy Entire Floor of Brownstone    4869 LisaRoxanne
## 5 5022 Entire Apt: Spacious Studio/Loft by central park    7192       Laura
## 6 5099        Large Cozy 1 BR Apartment In Midtown East    7322       Chris
##   neighbourhood_group neighbourhood latitude longitude       room_type price
## 1            Brooklyn    Kensington 40.64749 -73.97237    Private room   149
## 2           Manhattan       Midtown 40.75362 -73.98377 Entire home/apt   225
## 4            Brooklyn  Clinton Hill 40.68514 -73.95976 Entire home/apt    89
## 5           Manhattan   East Harlem 40.79851 -73.94399 Entire home/apt    80
## 6           Manhattan   Murray Hill 40.74767 -73.97500 Entire home/apt   200
##   minimum_nights number_of_reviews last_review reviews_per_month
## 1              1                 9    10/19/18              0.21
## 2              1                45     5/21/19              0.38
## 4              1               270      7/5/19              4.64
## 5             10                 9    11/19/18              0.10
## 6              3                74     6/22/19              0.59
##   calculated_host_listings_count availability_365
## 1                              6              365
## 2                              2              355
## 4                              1              194
## 5                              1                0
## 6                              1              129
```

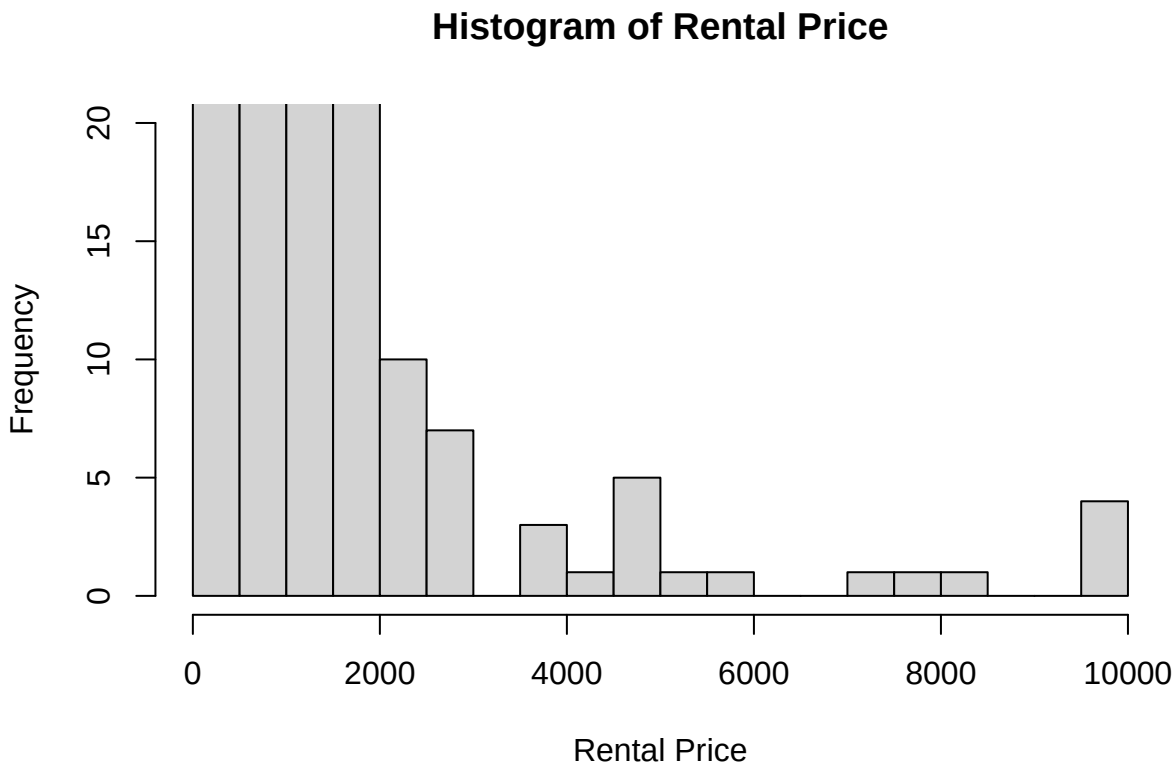
These commands give us some key information about the structure of the dataset. The summary function gives us the summary of the numerical variables such as minimum, maximum, mean, median, and quartiles. The str function gives us the structure of the dataset such as data types, variables, and their names. The

head function gives us the first five rows of the dataset giving us the idea of the format and values of the dataset.

### Histogram and Boxplot

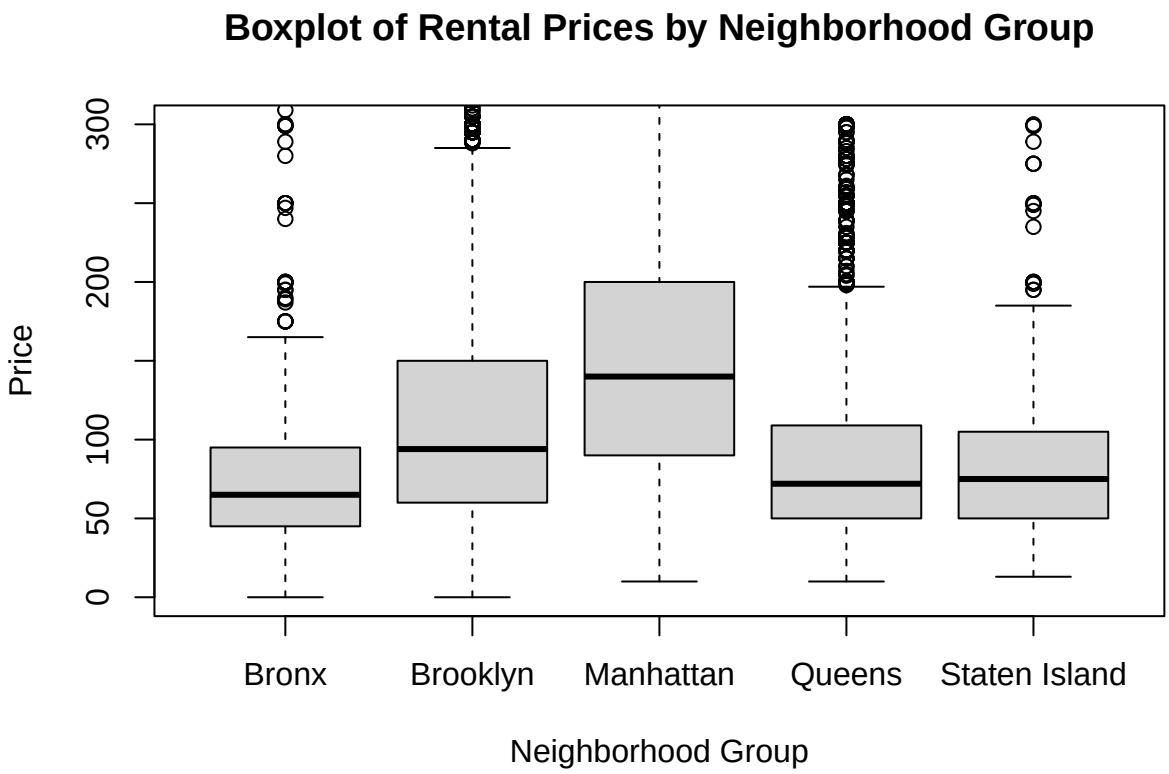
After having an overview of the data, we further need to create some visualizations to identify any patterns or trends that may exist.

```
# Creating histogram for rental price
hist(my_data$price,
     main = "Histogram of Rental Price",
     xlab = "Rental Price",
     ylab = "Frequency", ylim=c(0,20))
```



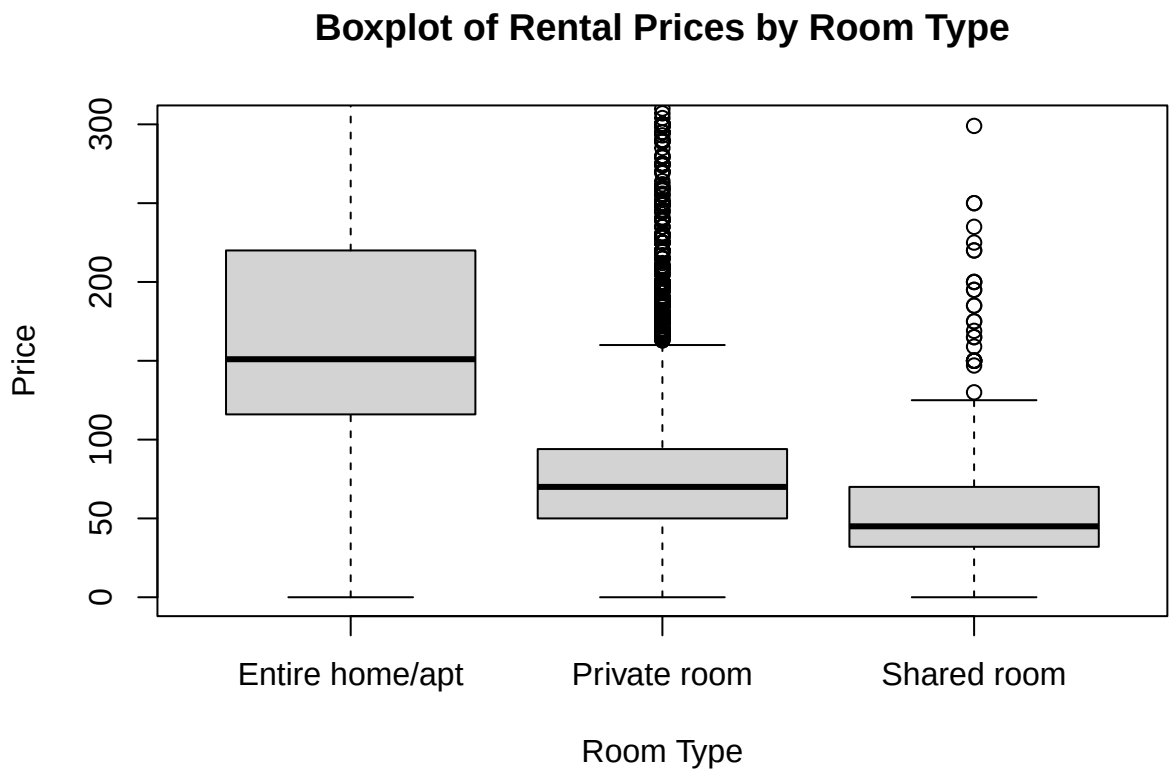

The above histogram shows the frequency distribution of price values in different price ranges. As we can see, most Airbnb listings fall within the price range of $0-$200 per night, with a few listings in the higher price ranges. The histogram also shows that the distribution of prices is skewed to the right, indicating that there are a few listings with very high prices.

```
# Creating boxplot for rental price by neighborhood
boxplot(my_data$price ~ my_data$neighbourhood_group, main="Boxplot of Rental Prices by Neighborhood Grou
```




The above boxplot is created for the rental price by neighborhood. The boxplot shows that the rental prices vary widely across different neighborhoods, with some neighborhoods having much higher prices than others such as Manhattan.

```
# Creating boxplot for rental price by room type
boxplot(my_data$price ~ my_data$room_type, main="Boxplot of Rental Prices by Room Type",xlab="Room Type
```




The above boxplot shows the rental price with the type of rental it is. The boxplot shows that the rental prices vary widely across different room types, with entire homes/apt having higher prices than private rooms and shared rooms.

## Statistical Analysis

### Hypothesis Testing

Based on our EDA findings, we further analyzed the data by performing hypothesis testing to determine whether there is a significant difference in the mean rental prices between different neighborhood groups. We use a t-test to compare the means of two or more groups. We pose the following hypothesis:

- Null hypothesis: There is no significant difference in the mean rental prices across neighborhoods
- Alternative hypothesis: There is a significant difference in the mean rental prices across the neighborhood

```
# Subset the data to include only Manhattan and Brooklyn rentals
subset_data <- my_data[my_data$neighbourhood_group %in% c("Manhattan", "Brooklyn"),]

# Conduct a two-sample t-test
t.test(price ~ neighbourhood_group, data = subset_data)
```

```
##
##  Welch Two Sample t-test
```

```
##
## data:  price by neighbourhood_group
## t = -26.649, df = 30845, p-value < 2.2e-16
## alternative hypothesis: true difference in means between group Brooklyn and group Manhattan is not e
## 95 percent confidence interval:
##  -62.91411 -54.29344
## sample estimates:
##  mean in group Brooklyn mean in group Manhattan
##                121.4487                180.0525
```

A p-value of $< 2.2e-16$ suggests that there is a very significant difference in the mean rental prices between the "Manhattan" and "Brooklyn" neighborhood groups. This means that the difference in mean rental prices between these two groups is unlikely to have occurred by chance alone and is likely due to some underlying factor(s) that differentiate these two neighborhoods. Therefore, we can infer that neighborhood is a significant factor that affects rental prices.

### Regression Analysis

Based on our EDA findings, we further analyzed the data by performing the regression analysis to determine whether there is a significant difference in the mean rental prices between different room types. By fitting a linear regression model to the data, we can estimate the relationship between rental price (the dependent variable) and room type (the independent variable), controlling for any other relevant variables. The coefficients of the regression model provide estimates of the impact of each type of room (private, shared, entire) on the rental price while controlling for other factors. The results of the analysis can provide insights into how the type of room affects the rental price.

```
my_data$entire_apt <- ifelse(my_data$room_type == "Entire home/apt", 1, 0)
my_data$private_room <- ifelse(my_data$room_type == "Private room", 1, 0)
my_data$shared_room <- ifelse(my_data$room_type == "Shared room", 1, 0)
model <- lm(price ~ entire_apt + private_room + shared_room, data = my_data)
summary(model)
```

```
##
## Call:
## lm(formula = price ~ entire_apt + private_room + shared_room,
##     data = my_data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -196.3  -47.0  -21.3   11.0 9916.0
##
## Coefficients: (1 not defined because of singularities)
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    63.214      6.485   9.747  < 2e-16 ***
## entire_apt    133.080      6.619  20.107  < 2e-16 ***
## private_room   20.767      6.639   3.128  0.00176 **
## shared_room        NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 188.6 on 38840 degrees of freedom
## Multiple R-squared:  0.08273,    Adjusted R-squared:  0.08269
## F-statistic:  1752 on 2 and 38840 DF,  p-value: < 2.2e-16
```

According to the output, both "entire_apt" and "private_room" have significant coefficients with p-values less than 0.05. This suggests that these room types are associated with a higher mean price compared to shared rooms. The coefficient for "shared_room" is not defined due to singularities in the model, but we can infer that the mean price for shared rooms is significantly lower than the other two room types. The adjusted R-squared value of 0.08269 indicates that only about 8% of the variability in price can be explained by the room type variable in this model. The results suggest that the room type has a small impact on rental prices, it is not the most significant variable. This means that other factors, such as location, amenities, size, or other variables not included in the model, may have a stronger impact on the rental price than the room type variable.

## Modeling and Prediction

### Logistic Regression

Furthermore, we performed Logistic regression to predict the probability of a listing being in a particular price range based on the neighborhood and room type combination. There are three price ranges: low, medium, and high.

```
# Logistic Regression
my_data$price_range <- cut(my_data$price, breaks = c(0, 100, 200, Inf), labels = c("low", "medium", "hig

# Load the required library for logistic regression
library(nnet)

# Fit the logistic regression model
model <- multinom(price_range ~ neighbourhood_group + room_type, data = my_data)
```

```
## # weights:  24 (14 variable)
## initial  value 42662.411006
## iter  10 value 35150.761560
## iter  20 value 28141.086324
## iter  30 value 27921.936318
## final  value 27921.932865
## converged
```

```
# Print the summary of the model
summary(model)
```

```
## Call:
## multinom(formula = price_range ~ neighbourhood_group + room_type,
##     data = my_data)
##
## Coefficients:
##        (Intercept) neighbourhood_groupBrooklyn neighbourhood_groupManhattan
## medium  -0.3464128                   1.100187                      2.254620
## high    -1.9076537                   1.579588                      3.419209
##        neighbourhood_groupQueens neighbourhood_groupStaten Island
## medium                 0.5051847                         0.1359791
## high                   0.7559815                         0.2671365
##        room_typePrivate room room_typeShared room
## medium             -2.958283            -4.045160
## high               -4.171261            -4.564463
```

```
##
## Std. Errors:
##        (Intercept) neighbourhood_groupBrooklyn neighbourhood_groupManhattan
## medium   0.1071100                  0.1091003                   0.1100245
## high     0.2086799                  0.2109322                   0.2108158
##        neighbourhood_groupQueens neighbourhood_groupStaten Island
## medium                 0.1154538                       0.1912290
## high                   0.2210922                       0.3492015
##        room_typePrivate room room_typeShared room
## medium            0.03091514            0.1497586
## high              0.05716825            0.2484856
##
## Residual Deviance: 55843.87
## AIC: 55871.87
```

The coefficients for each neighborhood group and room type indicate how much they contribute to the probability of a listing belonging to a particular price range. The output shows that in reference to the neighborhood, the probability of listing in the medium price range is higher than the probability of it being in the lower or high price range due to the intercept values of -0.3464 for medium price range. A listing in Manhattan is more likely to be in a high price range compared to the reference neighborhood due to the coefficient of 3.4192. A private room is less likely to be in the medium price range or high price range compared to the reference room type due to the negative coefficient of -2.9583 and -4.1713 respectively.

### Cross-Validation

We performed cross-validation to evaluate the accuracy of the logistic regression model

```
# Check for missing values
sum(!complete.cases(my_data))
```

```
## [1] 10
```

```
# Remove rows with missing values
my_data <- my_data[complete.cases(my_data), ]


# Calculating the accuracy

library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
# Set seed for reproducibility
set.seed(123)

# Split data into training and testing sets
trainIndex <- createDataPartition(my_data$price_range, p = 0.8, list = FALSE)
trainData <- my_data[trainIndex, ]
testData <- my_data[-trainIndex, ]
```

```r
# Fit the logistic regression model on the training data
model <- multinom(price_range ~ neighbourhood_group + room_type, data = trainData)
```

```
## # weights:  24 (14 variable)
## initial  value 34131.686584
## iter  10 value 25249.218237
## iter  20 value 22370.365255
## final  value 22302.199599
## converged
```

```r
# Use 10-fold cross-validation to estimate accuracy
fitControl <- trainControl(method = "cv", number = 10)
accuracy <- train(price_range ~ neighbourhood_group + room_type, data = trainData,
                  method = "multinom", trControl = fitControl)$results$Accuracy
```

```
## # weights:  24 (14 variable)
## initial  value 30718.298203
## iter  10 value 23524.122732
## iter  20 value 20202.300681
## iter  30 value 20021.065923
## iter  40 value 20020.984850
## final  value 20020.926642
## converged
## # weights:  24 (14 variable)
## initial  value 30718.298203
## iter  10 value 23549.893137
## iter  20 value 20086.313146
## iter  30 value 20029.436441
## iter  30 value 20029.436424
## final  value 20029.436424
## converged
## # weights:  24 (14 variable)
## initial  value 30718.298203
## iter  10 value 23524.148737
## iter  20 value 20202.437496
## iter  30 value 20021.085638
## iter  40 value 20020.977247
## final  value 20020.933077
## converged
## # weights:  24 (14 variable)
## initial  value 30719.396816
## iter  10 value 23568.465213
## iter  20 value 20147.984138
## iter  30 value 20068.734997
## final  value 20068.732409
## converged
## # weights:  24 (14 variable)
## initial  value 30719.396816
## iter  10 value 23593.723511
## iter  20 value 20170.399649
## iter  30 value 20077.047432
## iter  30 value 20077.047347
```

```
## final  value 20077.047347
## converged
## # weights:  24 (14 variable)
## initial  value 30719.396816
## iter  10 value 23568.490695
## iter  20 value 20147.969585
## iter  30 value 20068.743383
## final  value 20068.740809
## converged
## # weights:  24 (14 variable)
## initial  value 30718.298203
## iter  10 value 23547.178725
## iter  20 value 20105.518724
## final  value 20065.899558
## converged
## # weights:  24 (14 variable)
## initial  value 30718.298203
## iter  10 value 23573.104093
## iter  20 value 20095.833135
## final  value 20074.751044
## converged
## # weights:  24 (14 variable)
## initial  value 30718.298203
## iter  10 value 23547.204885
## iter  20 value 20105.527745
## final  value 20065.908552
## converged
## # weights:  24 (14 variable)
## initial  value 30717.199591
## iter  10 value 23575.443020
## iter  20 value 20105.718332
## final  value 20083.638206
## converged
## # weights:  24 (14 variable)
## initial  value 30717.199591
## iter  10 value 23600.476338
## iter  20 value 20130.025033
## final  value 20091.857372
## converged
## # weights:  24 (14 variable)
## initial  value 30717.199591
## iter  10 value 23575.468272
## iter  20 value 20105.710727
## final  value 20083.646536
## converged
## # weights:  24 (14 variable)
## initial  value 30719.396816
## iter  10 value 23585.957076
## iter  20 value 20184.584281
## iter  30 value 20112.885728
## iter  30 value 20112.885707
## final  value 20112.885707
## converged
## # weights:  24 (14 variable)
```

```
## initial  value 30719.396816
## iter  10 value 23610.907097
## iter  20 value 20154.297757
## final  value 20121.288667
## converged
## # weights:  24 (14 variable)
## initial  value 30719.396816
## iter  10 value 23585.982240
## iter  20 value 20184.792255
## iter  30 value 20112.894243
## iter  30 value 20112.894224
## final  value 20112.894224
## converged
## # weights:  24 (14 variable)
## initial  value 30719.396816
## iter  10 value 23513.569266
## iter  20 value 20130.387161
## iter  30 value 20064.797112
## final  value 20064.790883
## converged
## # weights:  24 (14 variable)
## initial  value 30719.396816
## iter  10 value 23539.738484
## iter  20 value 20220.410522
## iter  30 value 20073.250241
## final  value 20073.247655
## converged
## # weights:  24 (14 variable)
## initial  value 30719.396816
## iter  10 value 23513.595671
## iter  20 value 20130.389035
## iter  30 value 20064.805588
## final  value 20064.797706
## converged
## # weights:  24 (14 variable)
## initial  value 30717.199591
## iter  10 value 23451.264558
## iter  20 value 20090.202117
## final  value 20063.489389
## converged
## # weights:  24 (14 variable)
## initial  value 30717.199591
## iter  10 value 23477.051990
## iter  20 value 20092.350634
## final  value 20071.953764
## converged
## # weights:  24 (14 variable)
## initial  value 30717.199591
## iter  10 value 23451.290579
## iter  20 value 20090.142565
## final  value 20063.497963
## converged
## # weights:  24 (14 variable)
## initial  value 30718.298203
```

```
## iter  10 value 23543.935379
## iter  20 value 20186.157903
## iter  30 value 20084.493211
## iter  40 value 20084.483589
## final  value 20084.481848
## converged
## # weights:  24 (14 variable)
## initial  value 30718.298203
## iter  10 value 23570.085917
## iter  20 value 20287.995430
## iter  30 value 20093.246002
## iter  40 value 20093.227217
## final  value 20093.224325
## converged
## # weights:  24 (14 variable)
## initial  value 30718.298203
## iter  10 value 23543.961767
## iter  20 value 20186.229450
## iter  30 value 20084.501947
## final  value 20084.494336
## converged
## # weights:  24 (14 variable)
## initial  value 30718.298203
## iter  10 value 23541.878137
## iter  20 value 20115.842209
## final  value 20073.817167
## converged
## # weights:  24 (14 variable)
## initial  value 30718.298203
## iter  10 value 23567.635506
## iter  20 value 20132.845465
## iter  30 value 20082.273813
## iter  30 value 20082.273813
## final  value 20082.273813
## converged
## # weights:  24 (14 variable)
## initial  value 30718.298203
## iter  10 value 23541.904132
## iter  20 value 20115.841058
## final  value 20073.825740
## converged
## # weights:  24 (14 variable)
## initial  value 30719.396816
## iter  10 value 23597.641263
## iter  20 value 20135.103339
## iter  30 value 20075.475054
## iter  30 value 20075.475023
## final  value 20075.475023
## converged
## # weights:  24 (14 variable)
## initial  value 30719.396816
## iter  10 value 23623.275670
## iter  20 value 20111.361384
## iter  30 value 20083.984779
```

```
## iter  30 value 20083.984778
## final  value 20083.984778
## converged
## # weights:  24 (14 variable)
## initial  value 30719.396816
## iter  10 value 23597.667128
## iter  20 value 20135.093857
## iter  30 value 20075.483688
## iter  30 value 20075.483655
## final  value 20075.483655
## converged
## # weights:  24 (14 variable)
## initial  value 34131.686584
## iter  10 value 25263.926280
## iter  20 value 22391.875586
## iter  30 value 22310.691477
## iter  30 value 22310.691440
## final  value 22310.691440
## converged
```

```
# Print accuracy
accuracy
```

```
## [1] 0.6849173 0.6849173 0.6849173
```

```
mean(accuracy)
```

```
## [1] 0.6849173
```

The mean(accuracy) value of 0.6849173 suggests that the logistic regression model you fitted has an overall accuracy of about 68.49%

## Conclusion

Based on the analysis and modeling, we can conclude that the logistic regression model using the variables "neighbourhood_group" and "room_type" is able to predict the price range of Airbnb listings with an accuracy of approximately 68-69% cross-validation technique. However, it is important to note that the accuracy may not be sufficient for some business applications where higher precision is required. Therefore, we may need to explore other modeling techniques or consider additional variables to improve the accuracy. Furthermore, the analysis is limited by the data available, as there may be other factors that affect the price range of Airbnb listings. Therefore, future research could involve collecting more comprehensive data and exploring other modeling techniques to identify other important variables that may affect the price range of Airbnb listings. Additionally, it may be useful to examine how the model performs for different cities or geographic regions, as pricing patterns may vary by location.