Airbnb Price Prediction Using Machine Learning Models

Contents

1 Introduction

Determining the optimal price for renting out a space on Airbnb poses a considerable challenge. To address this challenge, the data is extracted from Airbnb listings in three major European cities; Athens, Bologna, and Copenhagen. Initially the data has 5867 observations and 49 variables.

2 Methodology

This section is divided into 3 major sub sections including information about the 'Data Preparation', 'Exploratory Data Analysis' and 'Regression Modelling'.

2.1 Data Preparation

Glancing over the summary of the data, there is a lot of scope of enhancing the data to prepare for it for the regression analysis. The tasks undertaken to address the issues involves dropping variables that are not required, reducing the NA values, removing special characters, removing outliers and creating dummy variables.

The columns dropped on the assumption that they dont affect the price for the listing and also there are only NA values. These include X, host_id, host_location, host_neighbourhood, neighbourhood, neighbourhood_group_cleansed, calendar_updated.

To address the issue of NA values in 'license', the structure of the variable is modified to have value as 1 if there is license information else 0.

For the variables 'beds' and 'bedrooms', the NA values are imputed in both on the basis of the observations that has values. This helped reduce the number of NA significantly. Similarly, the variables 'bathrooms' and 'bathrooms_text' were handled. Post this the variables 'bedrooms' and 'bathrooms_text' were dropped.

The variables with class type logical are converted to integer to have values 0 and 1 for FALSE and TRUE respectively.

Only the 'Year' value is retained from the variables with observation in data/time format. Similarly, variable with special symbols like % and $ signs were removed and only the original numeric value was retained.

[1] 64

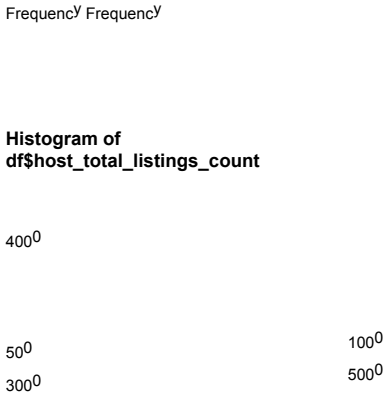
[1] 68

[1] 53

[1] 19


Histogram of df$price
Frequency
0
1000
2000
3000
4000
5000
6000
7000
df$price


Histogram of df$host_listings_count
Frequency
0
50
100
150
200
250
300
df$host_listings_count



Histogram of df$minimum_nights

To address the issue with outliers in the variables ‘price’, ‘host_total_listings_count’,‘host_listing_count’ and ‘minimum_nights’, histogram and boxplots are used to eliminate the bias in the data.

Frequency Distribution of City

Frequency

3000

1000

0

Athens_20_Sep_2022_listings (1).csv Bologna_14_Sep_2022_listings (1).csv

Copenhagen_24_Sep_2022_listings (1).csv City 3

The data is then split in to 2 data frames; df_A and df_B. The data df_A is the prime focus of the report and involved observations from the city ‘Athens’. For the purpose of regression analysis, the character variables are converted to numeric.

2.2 Exploratory Data Analysis

In this section various plots are created to undertand the relation between the variables.

2.2.1 Correlation Plot

While the correlation plot explores the correlation between the entire dataframe, the table provides the information of correlation between price and other variables. The accomodates and bathrooms have the higher postive correlation with the target variable price.

2.2.2 Price and Accommodates

Relationship between Accommodates and Price 1500

1000

Price

500

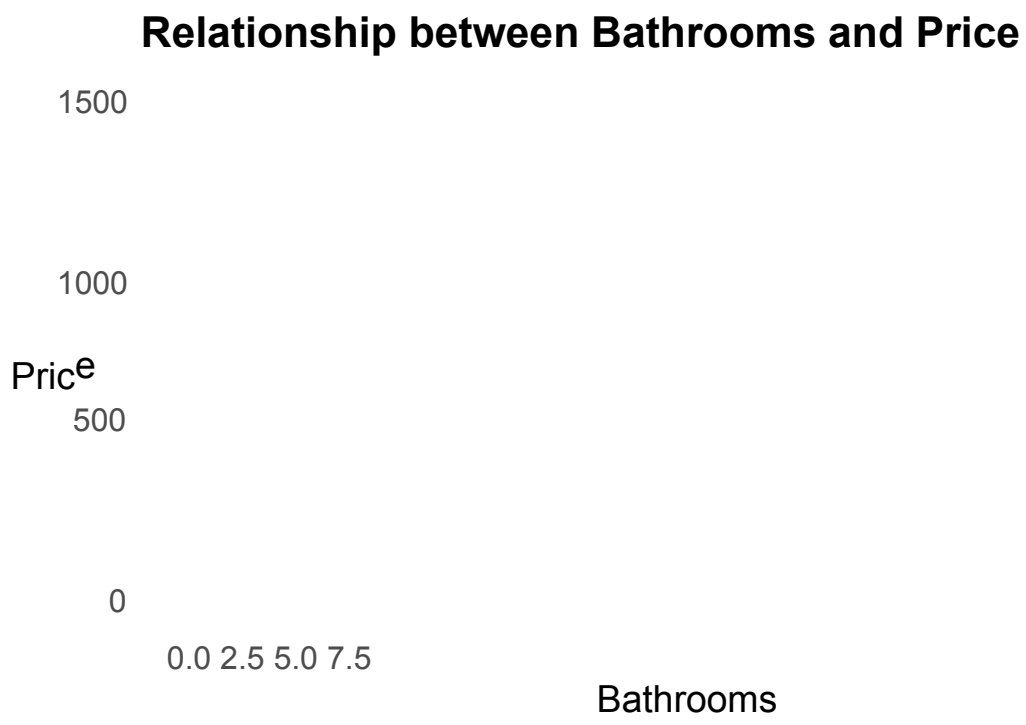
0

4 8 12 16

Accommodates

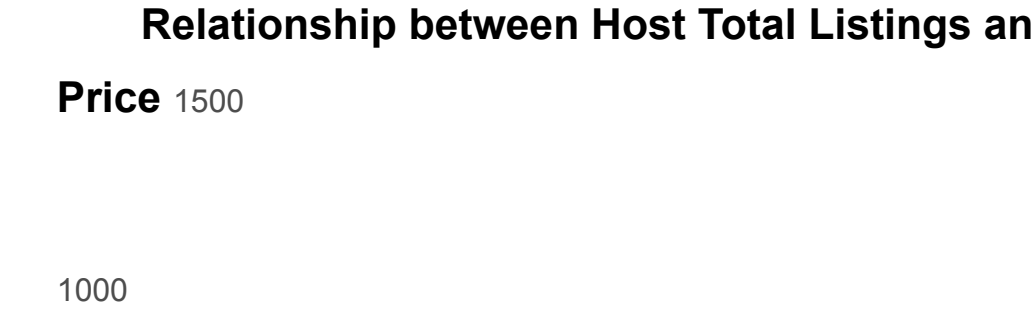
The graph above is a scatter plot representing positive linear relation between price and accommodates. 4

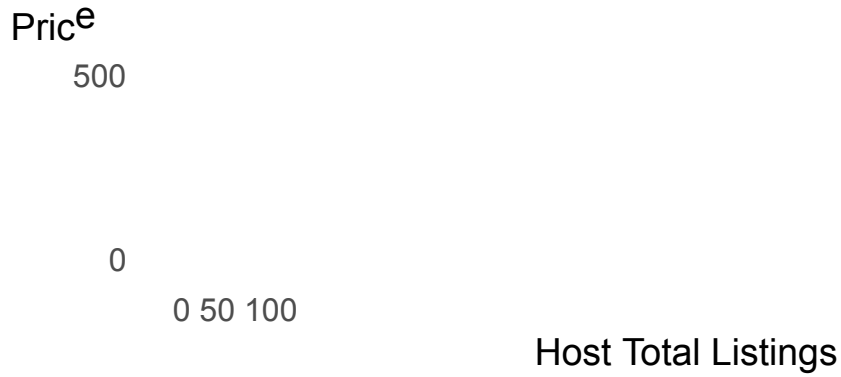
2.2.3 Price and Bathrooms



The scatterplot visualises the positive linear relationship between Bathrooms and Price.

2.2.4 Price and Total Listings





Similar to accommodates and bathrooms, the host_total_listings has positive linear relation with the price however it is not very significant.

2.2.5 Price and Number of Reviews

2.2.6 Price and Host_is_Superhost

1.00 0.75 0.50 0.25 0.00

2.3 Regression Analysis

2.3.1 Pre-Processing

As the foremost step in predictive analytics, the data set is split in to training and test data sets using 70:30 split ratio.

10 Fold cross validation function is created and stored in object ‘ctrlspecs’ to train the regression mod els.Training models using k-fold cross-validation is beneficial because it allows for a more reliable evaluation of the model’s performance by reducing the impact of data variability and providing a more robust esti mate of its generalization ability. Additionally, it helps in utilizing the available data more effectively by maximizing the use of both training and testing data across multiple iterations.

2.3.2 Regression Models

2.3.2.1 LASSO Regression

first_review availability_60 `room_type_Entire home/apt` `host_response_time_a few days or more` `host_response_time_within a few hours` review_scores_accuracy minimum_nights host_identity_verified host_has_profile_pic has_availability host_since host_acceptance_rate host_response_time_Unknown `host_response_time_within a day` amenities license host_response_rate host_listings_count number_of_reviews beds `host_response_time_within an hour` last_review host_is_superhost availability_90 `room_type_Shared room` longitude `room_type_Private room` maximum_nights instant_bookable host_total_listings_count latitude availability_365 availability_30 review_scores_checkin reviews_per_month review_scores_communication review_scores_cleanliness `room_type_Hotel room` review_scores_value review_scores_rating review_scores_location accommodates bathrooms

Feature

RMSE Rsquared
1 72.67702 0.2624872

The LASSO regression model is created and stored in object ‘model1’. LASSO regression was used because it helps in predicting the price of properties by considering various features. It is beneficial because it 50%matically selects the most important features and reduces the impact of less relevant features. This helps in creating a more accurate and efficient model for predicting property prices.

The best lambda value is used as to minimize the mean squared error.

2.3.2.2 Random Forest

RMSE Rsquare
1 67.55725 0.3615867

The Random Forest regression model is created and stored in object ‘model2’. Random Forest regression was used to predict the price of properties by considering various features. It is beneficial because it combines the predictions of multiple decision trees to make more accurate predictions. Each decision tree is trained on a different subset of data, and they work together to improve the overall prediction accuracy. This helps in capturing complex relationships between the features and the price, resulting in a more reliable model for predicting property prices.

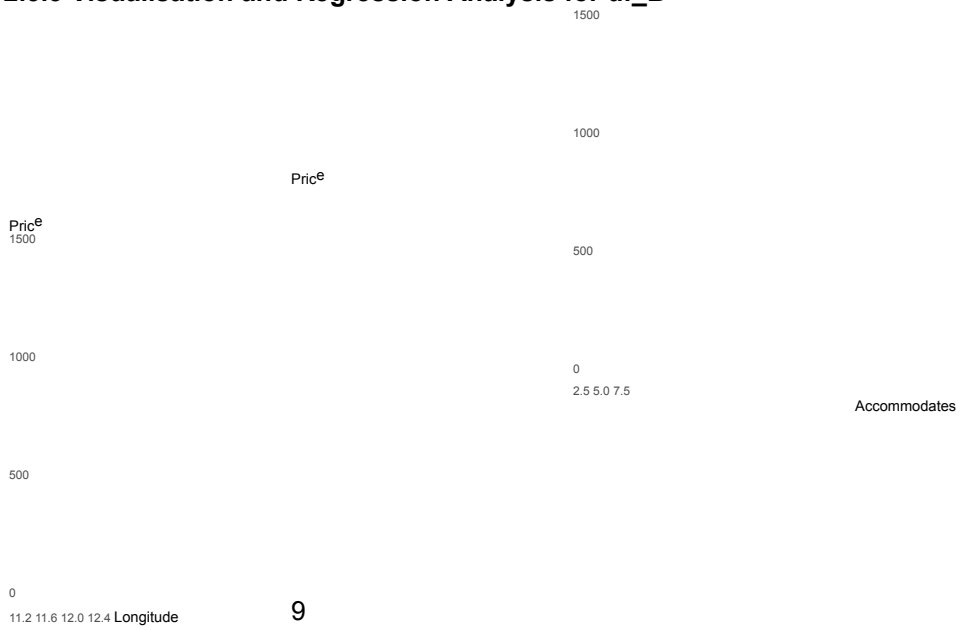
2.3.2.3 Support Vector Regression

RMSE Rsquare

1 69.80202 0.3324607

The Support Vector Regression model is created and stored in object ‘model3’. Support Vector Regression (SVR) was used to predict the price of properties based on different features. SVR is beneficial because it can handle both linear and non-linear relationships between the features and the price. It finds the best-fitting line or curve that maximizes the margin between the predicted values and the actual values. This helps in capturing the patterns and trends in the data, making it a powerful tool for predicting property prices accurately.

2.3.3 Visualisation and Regression Analysis for df_B



9

review_scores_communication `room_type_Hotel room` review_scores_value host_is_superhost host_response_rate `room_type_Private room` host_acceptance_rate latitude bathrooms host_since host_has_profile_pic review_scores_rating first_review beds license host_total_listings_count availability_90 review_scores_checkin host_listings_count has_availability `host_response_time_a few days or more` number_of_reviews `room_type_Shared room` host_response_time_Unknown availability_30 `host_response_time_within a few hours` amenities last_review instant_bookable review_scores_accuracy maximum_nights `host_response_time_within a day` host_identity_verified availability_60 `host_response_time_within an hour` review_scores_cleanliness availability_365 review_scores_location minimum_nights reviews_per_month `room_type_Entire home/apt` accommodates longitude

Feature

0 25 50 75 100

Importance

RMSE Rsquared
1 224.495 0.7949659

The exploratory data and regression analysis is done on df_B which consists data from cities Bologna and Copenhagen. The df_B relativey very small data frame with only 320 observations. From the correlation table and plot, it is evident that variable ‘longitude’ and ‘latitude’ has high positive correlation with price. As opposed to df_A, variables in df_B has higher degree of correlation with the target variable. The impact of this is observed in the R2 value of the LASSO regression model stored in object ‘model4’.

3 Results and Discussions

3.1 LASSO Regression

The model is tuned using the best lambda value to minimize the mean root squared error. The deviation between the actual and predicted values are provided by the RMSE value of 72.67702. This suggests that, on average, the predicted prices by the model deviate from the actual prices by approximately 72.67702 units.

The Rsquared value of 0.2624872 suggests that the regression model explains approximately 26.25% of the variance in the property prices.The remaining 73.75% of the variance is not accounted for by the model and may be attributed to other factors not included in the analysis.

The coefficients estimates for each feature indicates the strength and direction of relation with target vari able.Positive coefficients indicate that as the value of the corresponding feature increases, the predicted price also tends to increase. For example, features like “host_listings_count” and “availability_30” have positive coefficients, suggesting that properties with more host listings and higher availability in the next 30 days tend to have higher prices.

Negative coefficients indicate that as the value of the corresponding feature increases, the predicted price tends to decrease. For example, features like “latitude” and “number_of_reviews” have negative coefficients, indicating that properties located at lower latitudes and with more reviews tend to have lower prices.

Some coefficients are represented by dots (“.”) or zeros, which means that these features have been assigned a coefficient of zero and are not contributing to the prediction of property prices. This is a characteristic of the LASSO regression, which performs variable selection by shrinking some coefficients to zero.

From the variable importance table, “bathrooms”, with a score of 100, suggests that the number of bathrooms in a property has the strongest influence on predicting its price followed by “accommodates” with a score of 47.173472, indicating that the property’s capacity to accommodate guests is an important factor in determining its price.

3.2 Random Forest Regression

The RMSE value of 67.5831 indicates that the predicted prices are deviated from the actual price value by 67.5 units, which is lower than that of the LASSO model. The R2 value 0.3609034, indicated that the proportion of variance explained by the model is 36% approximately and this better than that of the LASSO regression model.

Similar to LASSO model, the variable importance shows that the variables ‘bathrooms’, ‘accommodates’ and ‘latitude’ has high level of impact on the price of the listings for the Random Forest regression model.

3.3 Support Vector Regression

The RMSE value of 69.79567 and the R2 value of 0.3327546, while it is slightly lower than that of Random Forest model but it is definitely better than that of LASSO regression model.

Variable importance output is quite similar to that of the other two models.

3.4 Dataframe B

The RMSE value of 224.65 indicates that there is high deviation between the predicted prices and the actual prices. While the R2 value of approximately 79% suggest a high degree of explanation for the proportion of variance.

From the coefficient estimates, a unit change in ‘longitude’ and ‘accommodates’ results in approximately 358 and 99 units of increase respectively in the price. While a unit increase in ‘minimum nights’ results in approximately 23 units decrease in price.

The output from variable importance indicates that ‘longitude’ has a major impact on predicting prices while ‘bathrooms’ has no impact unlike in the dataframe A. However, the ‘accommodates’ variable is still important in the both the data frames A and B.

4 Conclusion

The Random Forest regression model performed slightly better with an RMSE of 67.5831 and an R-squared of 0.3609034, explaining around 36% of the variance.

Among the three models, the number of bathrooms, accommodates, and latitude were consistently identified as important features influencing property prices. However, other variables such as host_listings_count, availability_30, latitude, and number_of_reviews also had significant impacts in the LASSO and Random Forest models.

Overall, these findings suggest that the Random Forest regression model outperforms the LASSO and Sup port Vector Regression models in predicting Airbnb property prices, providing more accurate predictions and explaining a higher proportion of variance. The identified influential features can help hosts and travelers understand the factors that drive property prices and make informed decisions.

References

1. Gareth, J., Daniela, W., Trevor, H., & Robert, T. (2021). An introduction to statistical learning: with applications in R. Spinger, Second Edition. https://www.statlearning.com/

Appendix

```
setwd("C:/Users/Prayas Sachdeva/Downloads")
library(fastDummies)
library(caret)
library(ggplot2)
library(dplyr)
library(corrplot)
library(glmnet)
library(randomForest)
library(e1071)

# Loading Data
init_ds <- read.csv("dataset-cities- Athens_20_ - Bologna_14 - Copenhagen - student 177 .csv") df <-
as.data.frame(init_ds)
summary(df)
```

```
colSums(is.na(df))

# Data Cleaning

## dropping columns
df <- subset(df, select = -c(X,host_id,host_location,host_neighbourhood,neighbourhood,neighbourhood_gro## dealing

with NA's

### license
df$license <- ifelse(is.na(df$license), 0, 1)

### beds and bedrooms
df$beds <- ifelse(!is.na(df$bedrooms), 1, df$beds)
df$bedrooms <- ifelse(!is.na(df$beds), 1, df$bedrooms)
df <- df %>% select(-bedrooms)
### bathrooms and bathrooms_text
levels(as.factor(df$bathrooms_text))
table(df$bathrooms_text)
df$bathrooms <- ifelse(grepl("\\d+\\.?\\d*", df$bathrooms_text), as.numeric(gsub("[^0-9.]", "", df$bathdf$bathrooms <-
ifelse(df$bathrooms_text %in% c("Half-bath", "Shared half-bath"), 1, df$bathrooms) df <- df[, -which(names(df) ==
"bathrooms_text")]

### dropping rows with NA values
df <- na.omit(df)

## data manipulation

### Converting logical variables to with T/F levels in to 1/0
df <- df %>%
  mutate_if(is.logical, as.integer)
```



```
### Extracting 'Years' from the variables with 'YYYY-MM-DD' values

#### Converting the variables to Date format
df$host_since <- as.Date(df$host_since, format = "%Y-%m-%d")
df$host_since <- format(df$host_since, "%Y")

df$first_review <- as.Date(df$first_review, format = "%Y-%m-%d")
df$first_review <- format(df$first_review, "%Y")

df$last_review <- as.Date(df$last_review, format = "%Y-%m-%d")
df$last_review <- format(df$last_review, "%Y")

### Converting variables with more than 2 levels in to dummy variables
####host_response_time
table(as.factor(df$host_response_time))
df$host_response_time[df$host_response_time == "N/A"] <- "Unknown"
      df <- dummy_cols(df, select_columns = c("host_response_time","room_type"), remove_selected_columns = TR

####property_type
levels(as.factor(df$property_type))
df <- df %>% select(-property_type)
```

```
### Removing special symbols like % and $
#### host_response_rate
table(as.factor(df$host_response_rate))
df$host_response_rate <- gsub("%", "", df$host_response_rate)
df$host_response_rate <- as.numeric(df$host_response_rate)
df$host_response_rate[is.na(df$host_response_rate)] <- mean(df$host_response_rate, na.rm = TRUE)

#### host_acceptance_rate
table(as.factor(df$host_acceptance_rate))
df$host_acceptance_rate <- gsub("%", "", df$host_acceptance_rate)
df$host_acceptance_rate <- as.numeric(df$host_acceptance_rate)
df$host_acceptance_rate[is.na(df$host_acceptance_rate)] <- mean(df$host_acceptance_rate, na.rm = TRUE)

#### price
table(as.factor(df$price))
df$price <- gsub("\\$", "", df$price)
df$price <- gsub(",", "", df$price)
df$price <- as.numeric(df$price)

#### Amenities
df$amenities <- gsub("\\[|\\]|\"", "", df$amenities)
df$amenities <- sapply(strsplit(df$amenities, ","), function(x) length(x))

## Outliers
hist(df$price)
boxplot(df$price)
sum(df$price > 1500)
df <- df[df$price <= 1500,]

hist(df$host_listings_count)
boxplot(df$host_listings_count)
```



```
sum(df$host_listings_count > 150)
df <- df[df$host_listings_count <= 150,]

hist(df$host_total_listings_count)
boxplot(df$host_total_listings_count)
sum(df$host_total_listings_count > 150)
df <- df[df$host_total_listings_count <= 150,]

hist(df$minimum_nights)
boxplot(df$minimum_nights)
sum(df$minimum_nights > 100)
df <- df[df$minimum_nights <= 100,]

# Splitting the data in to two sets
# Create the frequency table
city_freq <- table(as.factor(df$city))

# Plot the frequency distribution
# Create the frequency table
city_freq <- table(as.factor(df$city))
```

```r
# Plot the frequency distribution
barplot(city_freq, col = rainbow(length(city_freq)), cex.names = 0.7)

# Add labels and title
title("Frequency Distribution of City")
xlabel <- "City"
ylabel <- "Frequency"
mtext(xlabel, side = 1, line = 3)
mtext(ylabel, side = 2, line = 3)

df_A <- df[df$city == "Athens_20_Sep_2022_listings (1).csv", ] df_B <-
df[df$city != "Athens_20_Sep_2022_listings (1).csv", ]

# data pre-processing
df_A <- df_A %>%
  select(-id, -city)

df_B <- df_B %>%
  select(-id, -city)

df_A <- df_A %>%
  mutate_if(is.character, as.numeric)
df_B <- df_B %>%
  mutate_if(is.character, as.numeric)
summary(df_B)
# Compute correlation matrix
cor_matrix_A <- cor(df_A)
cor_matrix_B <- cor(df_B)

# Plot correlation matrix
corrplot(cor_matrix_A, method = "color", type = "full", tl.cex = 0.5,
         tl.col = "black", tl.srt = 45,
```



```r
         col = colorRampPalette(c("red", "green"))(100),
         addCoef.col = "white", number.cex = 0.4,
         mar = c(0, 0, 2, 0))
corrplot(cor_matrix_B, method = "color", type = "full",
         tl.cex = 0.5, tl.col = "black", tl.srt = 45,
         col = colorRampPalette(c("red", "green"))(100),
         addCoef.col = "white", number.cex = 0.4,
         mar = c(0, 0, 2, 0))

# Visualizations
# Scatter plot: price vs accommodates
ggplot(df_A, aes(x = accommodates, y = price)) +
  geom_point(size = 3, color = "steelblue", alpha = 0.6) +
  geom_smooth(method = "lm", se = FALSE, color = "darkorange", size = 1.5) +
  labs(x = "Accommodates", y = "Price", title = "Relationship between Accommodates and Price") +
  theme_minimal() +
  theme(plot.title = element_text(size = 16, face = "bold"),
        axis.title = element_text(size = 14),
        axis.text = element_text(size = 12),
        legend.position = "none")
```

```r
# Scatter plot: price vs bathrooms
ggplot(df_A, aes(x = bathrooms, y = price)) +
  geom_point(size = 3, color = "orange", alpha = 0.6) +
  geom_smooth(method = "lm", se = FALSE, color = "darkblue", size = 1.5) +
  labs(x = "Bathrooms", y = "Price", title = "Relationship between Bathrooms and Price") +
  theme_minimal() +
  theme(plot.title = element_text(size = 16, face = "bold"),
        axis.title = element_text(size = 14),
        axis.text = element_text(size = 12),
        legend.position = "none")

# Scatter plot: price vs host_total_listings
ggplot(df_A, aes(x = host_total_listings_count, y = price)) +
  geom_point(size = 3, color = "black", alpha = 0.6) +
  geom_smooth(method = "lm", se = FALSE, color = "darkgreen", size = 1.5) +
  labs(x = "Host Total Listings", y = "Price", title = "Relationship between Host Total Listings and Prtheme_minimal() +
  theme(plot.title = element_text(size = 16, face = "bold"),
        axis.title = element_text(size = 14),
        axis.text = element_text(size = 12),
        legend.position = "none")

# Scatter plot: price vs number_of_reviews
ggplot(df_A, aes(x = number_of_reviews, y = price)) +
  geom_point(size = 3, color = "steelblue", alpha = 0.6) +
  geom_smooth(method = "lm", se = FALSE, color = "darkorange", size = 1.5) +
  labs(x = "Number of Reviews", y = "Price", title = "Relationship between Number of Reviews and
  Price"theme_minimal() +
  theme(plot.title = element_text(size = 16, face = "bold"),
        axis.title = element_text(size = 14),
        axis.text = element_text(size = 12),
        legend.position = "none")
```



```r
# Bar plot: price vs host_is_superhost
ggplot(df_A, aes(x = host_is_superhost, y = price, fill = host_is_superhost)) + geom_bar(stat
  = "summary", fun = mean, position = "dodge") +
  labs(x = "Host Is Superhost", y = "Mean Price", title = "Relationship between Host Is Superhost and Ptheme_minimal()
  +
  theme(plot.title = element_text(size = 16, face = "bold"),
        axis.title = element_text(size = 14),
        axis.text = element_text(size = 12),
        legend.title = element_blank(),
        legend.text = element_text(size = 12))

#Regression Analysis
#Train and Test split
set.seed(40386053)
index_A <- createDataPartition(df_A$price, p = 0.7, list = FALSE)
train_A <- df_A[index_A, ]
test_A <- df_A[-index_A, ]

index_B <- createDataPartition(df_B$price, p = 0.7, list = FALSE)
train_B <- df_B[index_B, ]
test_B <- df_B[-index_B, ]
```

```
#K Fold Cross Validation
ctrlspecs <- trainControl(method = "cv", number = 10,
                          savePredictions = "all")


# Define the model formulas
formula <- price ~ .

#Specify and Train LASSO regression model

# create vector of potential lambda values
lambda_vector <- 10^seq(5,-5, length = 500)

# specify LASSO regression model to be estimated using training dataset and 10 fold cross validation prmodel1 <-
train(price ~ .,data = train_A,
                preProcess = c("center", "scale"),
                        method = "glmnet", tuneGrid=expand.grid(alpha=1, lambda = lambda_vector),
                trControl=ctrlspecs, na.action=na.omit)
# best tuning parameter
model1$bestTune

# LASSO regression coefficients (parameter estimates)
coef(model1$finalModel, model1$bestTune$lambda)

# Plot log(lambda) & RMSE
plot(log(model1$results$lambda),
     model1$results$RMSE,
     xlab = "log(lambda)",
     ylab='RMSE')

plot(log(model1$results$lambda),
```



```
     model1$results$Rsquared,
     xlab = "log(lambda)",
     ylab='R^2')

# Variable Importance
varImp(model1)
ggplot(varImp(model1))

# Model Prediction
predictions1 <- predict(model1, newdata = test_A)

# Model Performance/Accuracy
model1perf <- data.frame(RMSE = RMSE(predictions1, test_A$price),
                         Rsquared = R2(predictions1, test_A$price))
model1perf

library(randomForest)

# Create model using Random Forest
model2 <- train(price ~ ., data = train_A,
                method = "rf",
```

```
                    trControl = ctrlspecs,
                    na.action = na.omit)

# Variable Importance
varImp(model2)
ggplot(varImp(model2))

# Model Prediction
predictions2 <- predict(model2, newdata = test_A)

# Model Performance
model2perf <- data.frame(RMSE = RMSE(predictions2, test_A$price),
                         Rsquare=R2(predictions2, test_A$price))
model2perf

# Create model using Support Vector Regression (SVR)
model3 <- train(price ~ ., data = train_A,
                    method = "svmRadial",
                    trControl = ctrlspecs,
                    na.action = na.omit)

# Model Prediction
predictions3 <- predict(model3, newdata = test_A)

#Model Performance
model3perf <- data.frame(RMSE = RMSE(predictions3, test_A$price),
                         Rsquare=R2(predictions3, test_A$price))
model3perf

# Variable Importance
varImp(model3)
ggplot(varImp(model3))
```



```
#Visualisations on df_B
ggplot(df_B, aes(x = longitude, y = price)) +
  geom_point(color = "#0072B2", alpha = 0.6) + # Customize point color and transparency geom_smooth(method = "lm",
  se = FALSE, color = "#D55E00", linetype = "dashed") + # Add a smooth linelabs(x = "Longitude", y = "Price") + # Set x
  and y axis labels
  theme_minimal() # Apply a minimal theme

ggplot(df_B, aes(x = accommodates, y = price)) +
  geom_jitter(color = "#9072B9", alpha = 0.6) +
  geom_smooth(method = "lm", se = FALSE, color = "#D55E00", linetype = "dashed") + labs(x =
  "Accommodates", y = "Price") +
  theme_minimal()


# LASSO Model on df_B
# specify LASSO regression model to be estimated using training dataset and 10 fold cross validation prmodel4 <-
train(price ~ .,data = train_B,
                    preProcess = c("center", "scale"),
                          method = "glmnet", tuneGrid=expand.grid(alpha=1, lambda = lambda_vector),
                    trControl=ctrlspecs, na.action=na.omit)
# best tuning parameter
```

```
model4$bestTune

# LASSO regression coefficients (parameter estimates)
coef(model4$finalModel, model4$bestTune$lambda)

# Plot log(lambda) & RMSE
plot(log(model4$results$lambda),
     model4$results$RMSE,
     xlab = "log(lambda)",
     ylab='RMSE')

plot(log(model1$results$lambda),
     model1$results$Rsquared,
     xlab = "log(lambda)",
     ylab='R^2')

# Variable Importance
varImp(model4)
ggplot(varImp(model4))

# Model Prediction
predictions4 <- predict(model4, newdata = test_B)

# Model Performance/Accuracy
model4perf <- data.frame(RMSE = RMSE(predictions4, test_B$price),
                         Rsquared = R2(predictions4, test_B$price))
model4perf

# Linear Model
model5 <- lm(formula, data = train_B)
summary(model5)
```