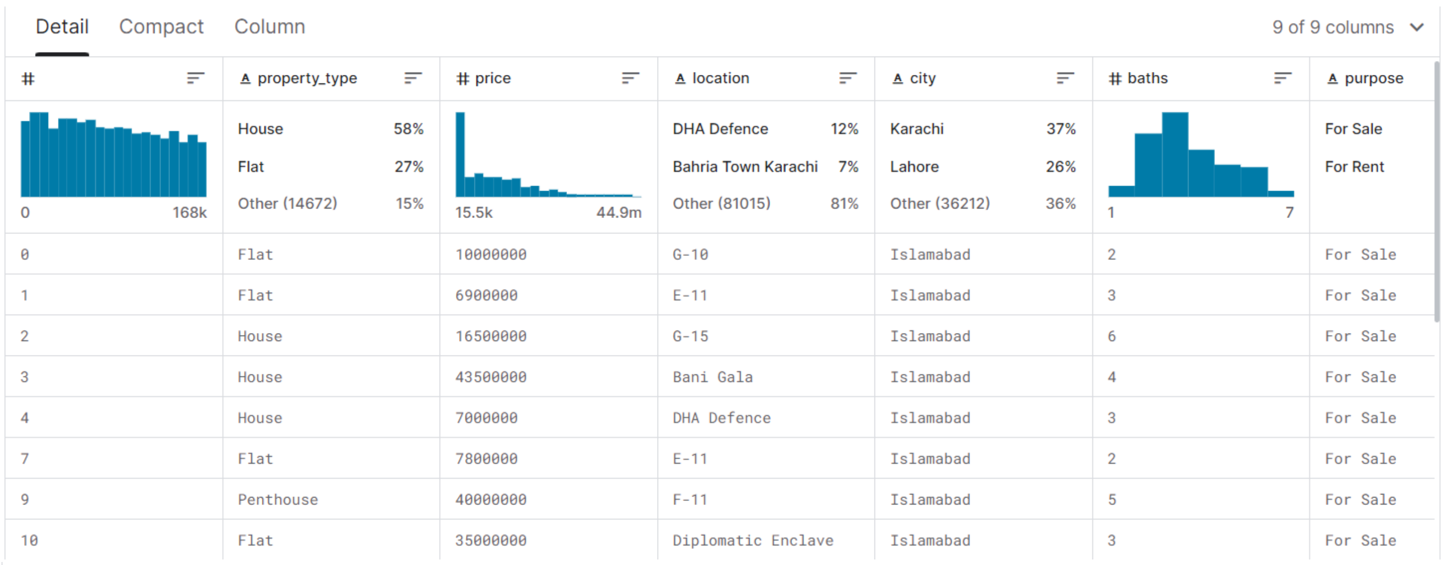
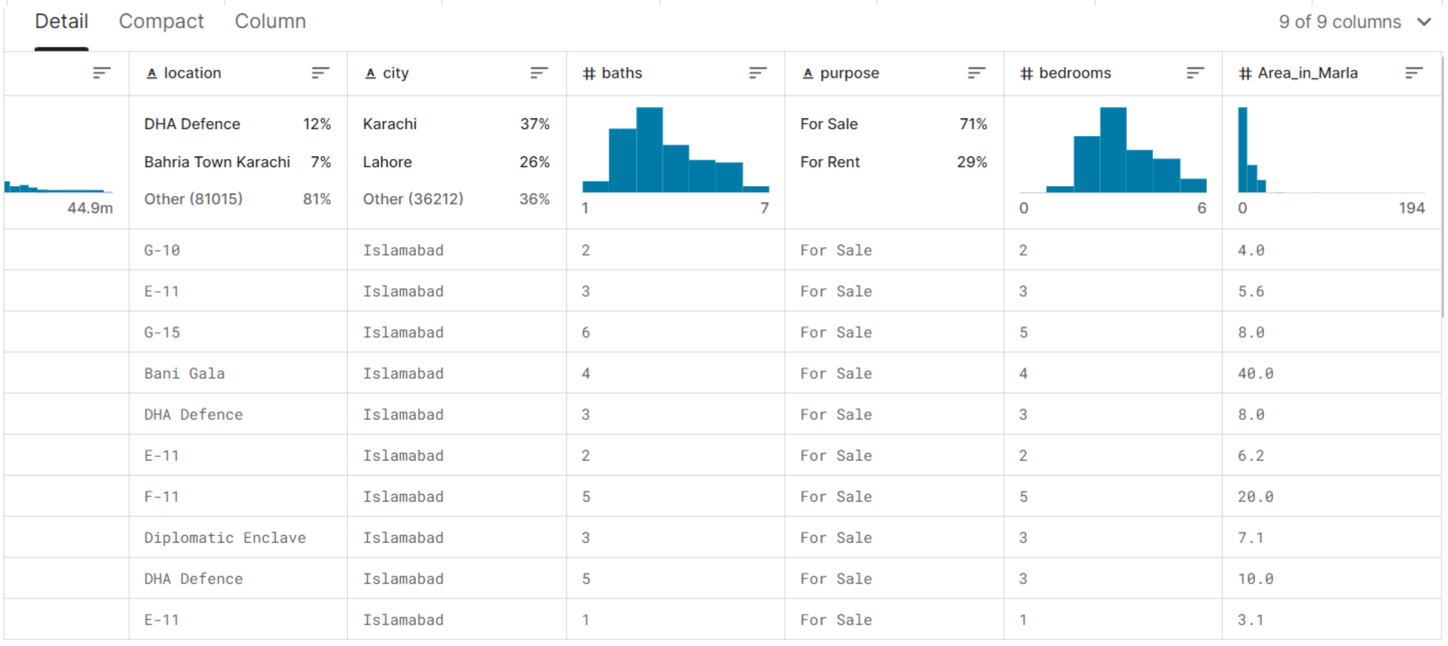
Darryl Babar, Christian Henry Miguel Caruz, Miguel Soniel, Jan Edgar Tupas CS174 BM1

M1 – SA CART and Decision Tree (Chapter Project)

The project uses a CART decision tree to predict the prices of houses from the House Prices 2023 Dataset retrieved from Kaggle. The dataset contains data of housing features and prices from certain cities in Pakistan. The features include **property_type** with six types, **location** describing the different locations in every city, **city** which include five cities in Pakistan, **baths**, **purpose**, **bedrooms**, and **Area_in_Marla**.

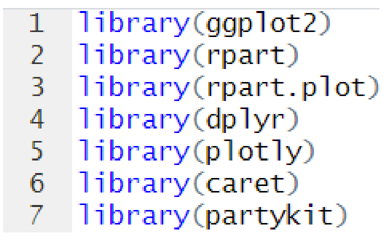


#	property_type	price	location	city	baths	purpose
0 – 168k	House 58%; Flat 27%; Other (14672) 15%	15.5k – 44.9m	DHA Defence 12%; Bahria Town Karachi 7%; Other (81015) 81%	Karachi 37%; Lahore 26%; Other (36212) 36%	1 – 7	For Sale; For Rent
0	Flat	10000000	G-10	Islamabad	2	For Sale
1	Flat	6900000	E-11	Islamabad	3	For Sale
2	House	16500000	G-15	Islamabad	6	For Sale
3	House	43500000	Bani Gala	Islamabad	4	For Sale
4	House	7000000	DHA Defence	Islamabad	3	For Sale
7	Flat	7800000	E-11	Islamabad	2	For Sale
9	Penthouse	40000000	F-11	Islamabad	5	For Sale
10	Flat	35000000	Diplomatic Enclave	Islamabad	3	For Sale



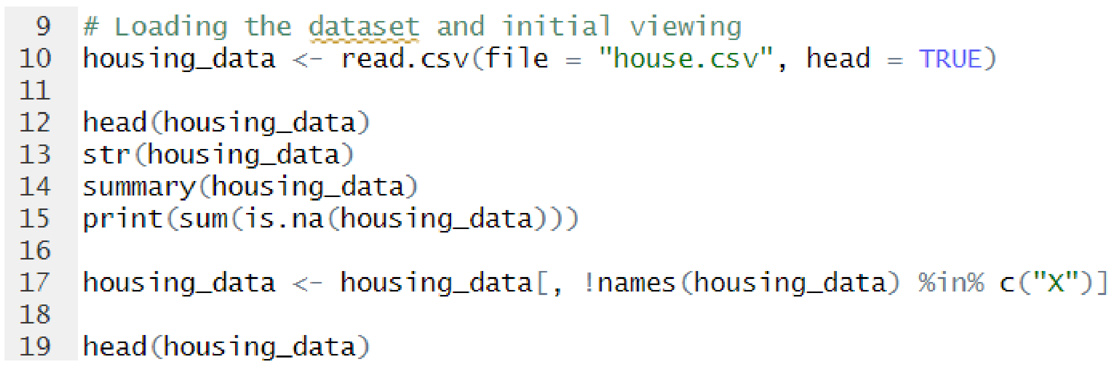
	location	city	baths	purpose	bedrooms	Area_in_Marla
44.9m	DHA Defence 12%; Bahria Town Karachi 7%; Other (81015) 81%	Karachi 37%; Lahore 26%; Other (36212) 36%	1 – 7	For Sale 71%; For Rent 29%	0 – 6	0 – 194
	G-10	Islamabad	2	For Sale	2	4.0
	E-11	Islamabad	3	For Sale	3	5.6
	G-15	Islamabad	6	For Sale	5	8.0
	Bani Gala	Islamabad	4	For Sale	4	40.0
	DHA Defence	Islamabad	3	For Sale	3	8.0
	E-11	Islamabad	2	For Sale	2	6.2
	F-11	Islamabad	5	For Sale	5	20.0
	Diplomatic Enclave	Islamabad	3	For Sale	3	7.1
	DHA Defence	Islamabad	5	For Sale	3	10.0
	E-11	Islamabad	1	For Sale	1	3.1

The packages we need for the project are installed and unpacked. We used ggplot2, rpart, rpart.plot, dplyr, plotly, caret, and partykit.



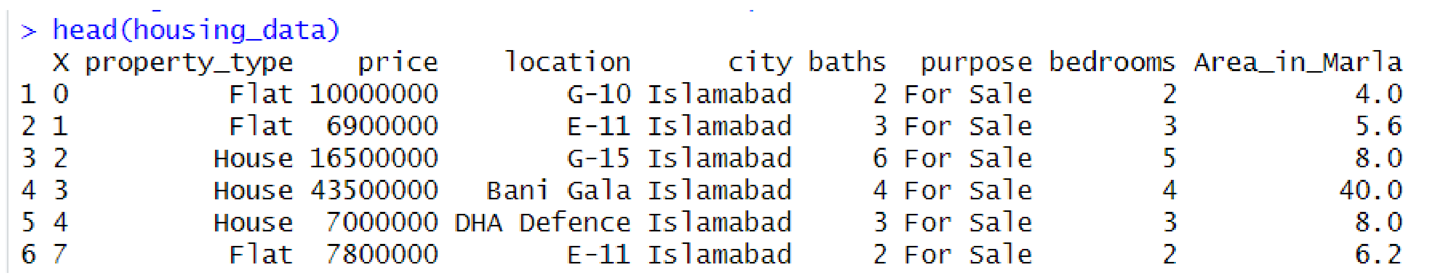
```r
library(ggplot2)
library(rpart)
library(rpart.plot)
library(dplyr)
library(plotly)
library(caret)
library(partykit)
```

We create a dataframe named housing_data and nitialize it with the housing prices 2023 dataset in the house.csv file. We can initially view and describe the nature of the dataset through head, str, and summary methods. We also check if there are missing values in the dataset. Since the X column just contains the index of each data, it is dropped.



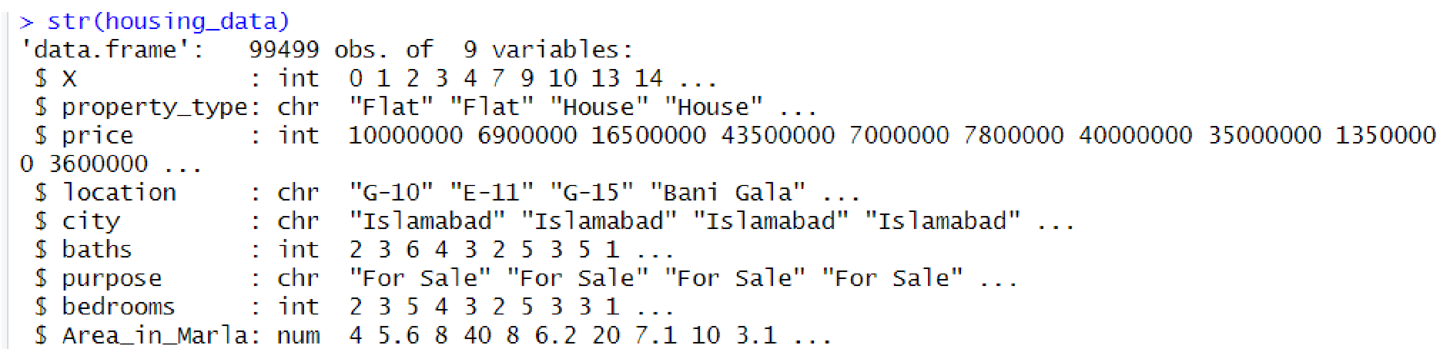
```r
# Loading the dataset and initial viewing
housing_data <- read.csv(file = "house.csv", head = TRUE)

head(housing_data)
str(housing_data)
summary(housing_data)
print(sum(is.na(housing_data)))

housing_data <- housing_data[, !names(housing_data) %in% c("X")]

head(housing_data)
```

The head method shows the first 6 instances in the dataset. We can see some samples of the dataset.



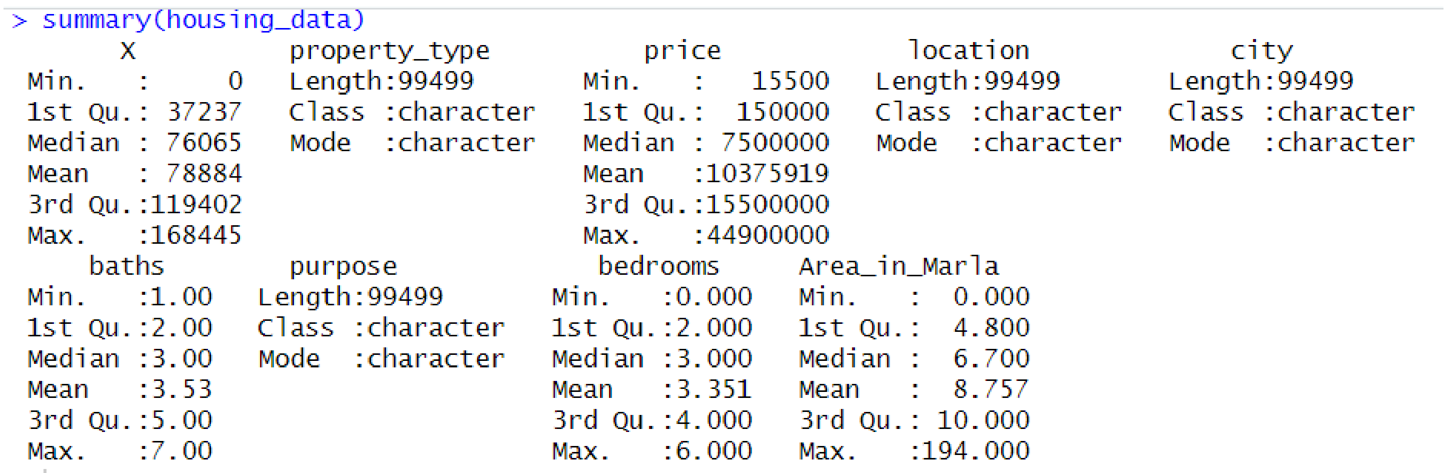
```
> head(housing_data)
  X property_type    price        location      city baths  purpose bedrooms Area_in_Marla
1 0          Flat 10000000            G-10 Islamabad     2 For Sale        2           4.0
2 1          Flat  6900000            E-11 Islamabad     3 For Sale        3           5.6
3 2         House 16500000            G-15 Islamabad     6 For Sale        5           8.0
4 3         House 43500000       Bani Gala Islamabad     4 For Sale        4          40.0
5 4         House  7000000 DHA Defence Islamabad     3 For Sale        3           8.0
6 7          Flat  7800000            E-11 Islamabad     2 For Sale        2           6.2
```

We used the Str method to show the datatypes and values of each feature in the dataset.



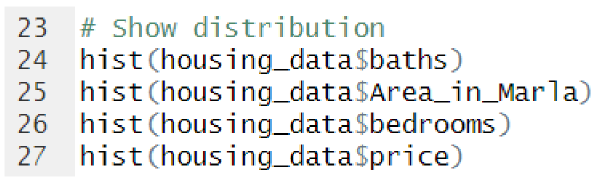
```
> str(housing_data)
'data.frame':	99499 obs. of  9 variables:
 $ X            : int  0 1 2 3 4 7 9 10 13 14 ...
 $ property_type: chr  "Flat" "Flat" "House" "House" ...
 $ price        : int  10000000 6900000 16500000 43500000 7000000 7800000 40000000 35000000 1350000
0 3600000 ...
 $ location     : chr  "G-10" "E-11" "G-15" "Bani Gala" ...
 $ city         : chr  "Islamabad" "Islamabad" "Islamabad" "Islamabad" ...
 $ baths        : int  2 3 6 4 3 2 5 3 5 1 ...
 $ purpose      : chr  "For Sale" "For Sale" "For Sale" "For Sale" ...
 $ bedrooms     : int  2 3 5 4 3 2 5 3 3 1 ...
 $ Area_in_Marla: num  4 5.6 8 40 8 6.2 20 7.1 10 3.1 ...
```

We used the summary method to show a summary of the dataset. This includes the summary of each feature in the dataset, their datatype, and the minimum, 1st quartile, median, mean, 3rd quartile, and max of numerical features.



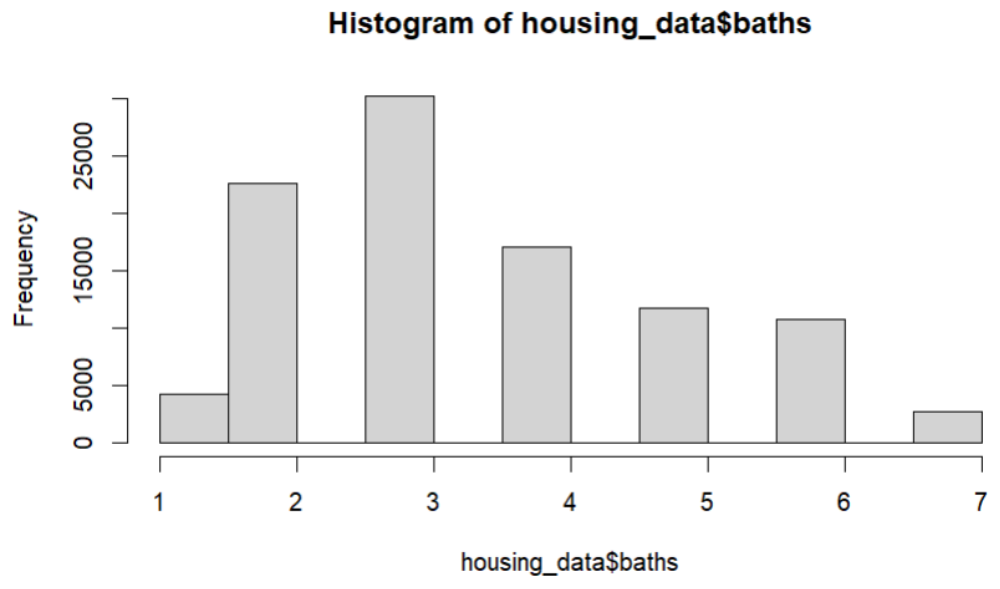
```
> summary(housing_data)
       X          property_type          price             location             city          
 Min.   :     0   Length:99499       Min.   :   15500   Length:99499       Length:99499      
 1st Qu.: 37237   Class :character   1st Qu.:  150000   Class :character   Class :character  
 Median : 76065   Mode  :character   Median : 7500000   Mode  :character   Mode  :character  
 Mean   : 78884                      Mean   :10375919                                        
 3rd Qu.:119402                      3rd Qu.:15500000                                        
 Max.   :168445                      Max.   :44900000                                        
     baths         purpose             bedrooms      Area_in_Marla   
 Min.   :1.00   Length:99499       Min.   :0.000   Min.   :  0.000  
 1st Qu.:2.00   Class :character   1st Qu.:2.000   1st Qu.:  4.800  
 Median :3.00   Mode  :character   Median :3.000   Median :  6.700  
 Mean   :3.53                      Mean   :3.351   Mean   :  8.757  
 3rd Qu.:5.00                      3rd Qu.:4.000   3rd Qu.: 10.000  
 Max.   :7.00                      Max.   :6.000   Max.   :194.000  
```

We perform exploratory data analysis. This allows us to have a better understanding of the dataset. Here we show the distributions of numerical features including baths, Area_in_Marla, bedrooms and price. We use histogram.

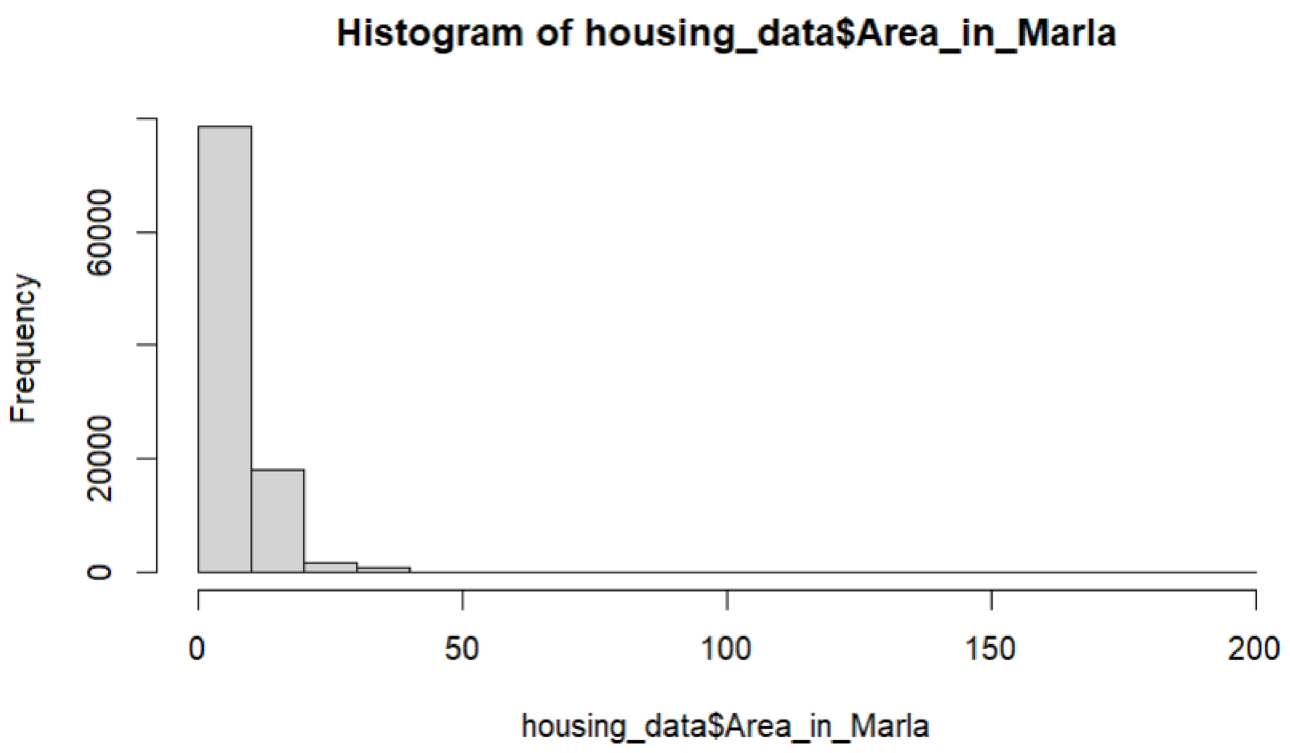


```
# Show distribution
hist(housing_data$baths)
hist(housing_data$Area_in_Marla)
hist(housing_data$bedrooms)
hist(housing_data$price)
```

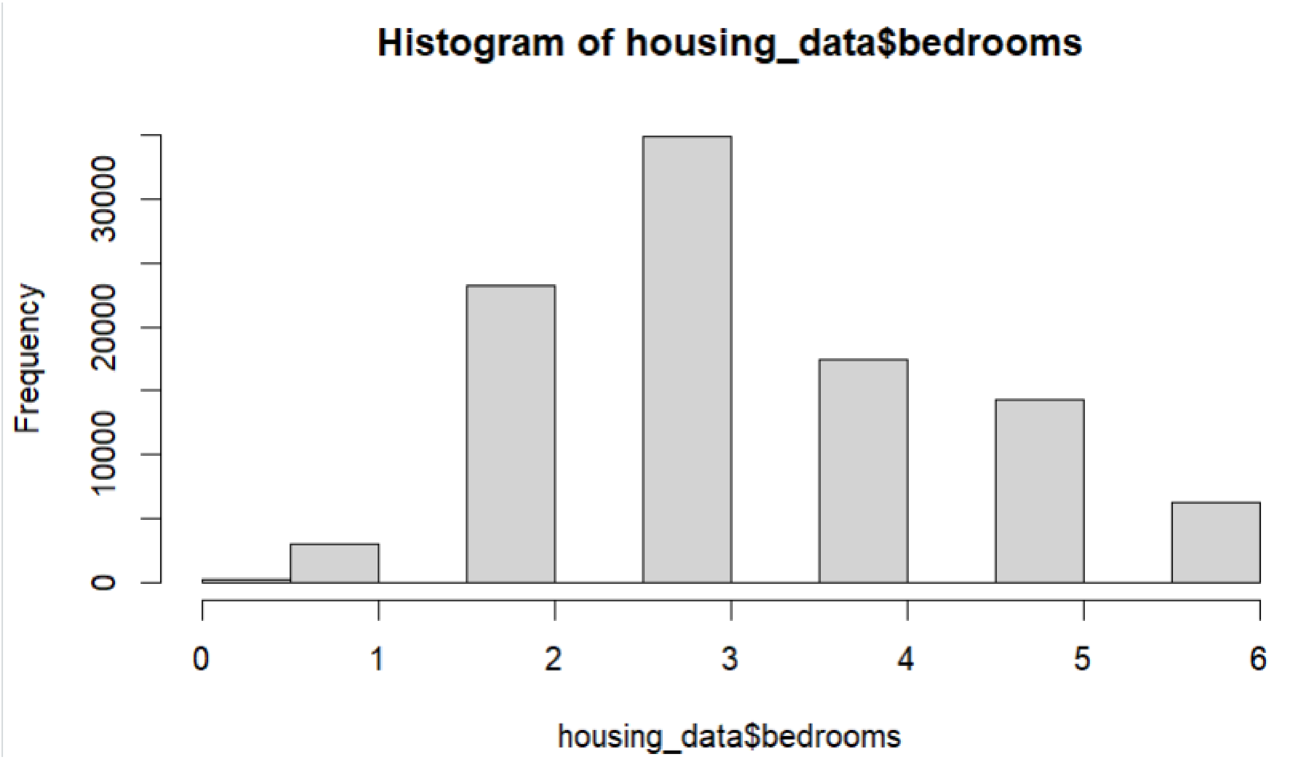
We show the distribution of the baths features. We can see that most number of houses have 3 baths, followed by 2 and 4 respectively. Houses with 7 and 1 baths are the last.



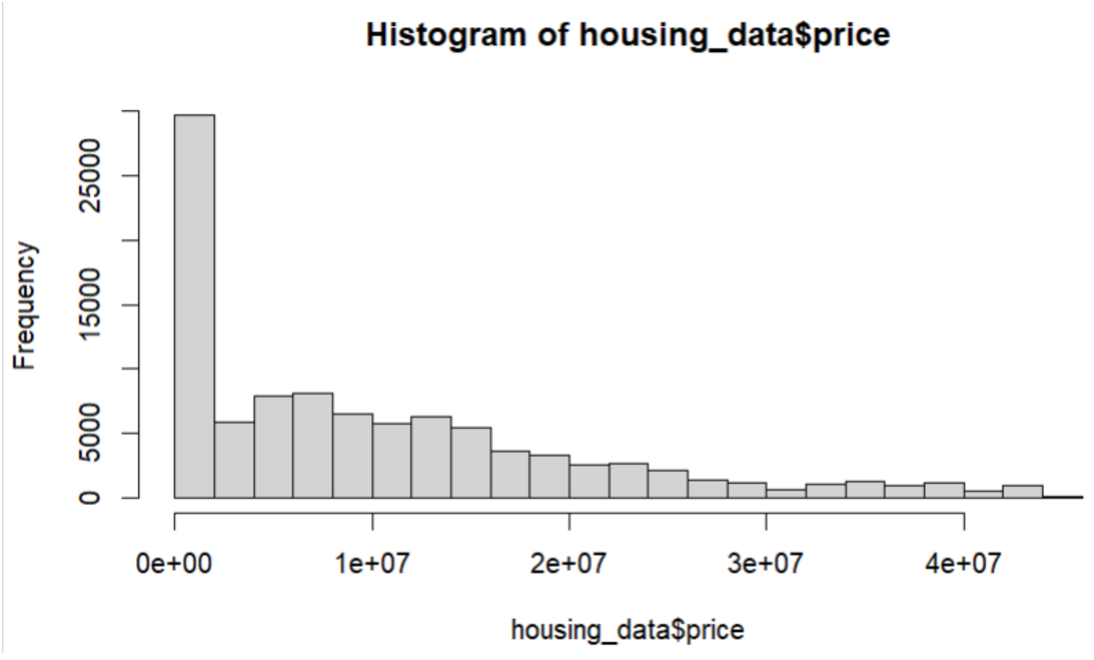
Next we show the distribution of the area_in_marla feature. We can see that most houses have an area below 10.



Next we show the distribution of the bedrooms feature. We can observe that the highest frequency are houses with 3 bedrooms, followed by 2 and 4. The least are houses with 0 bathrooms.

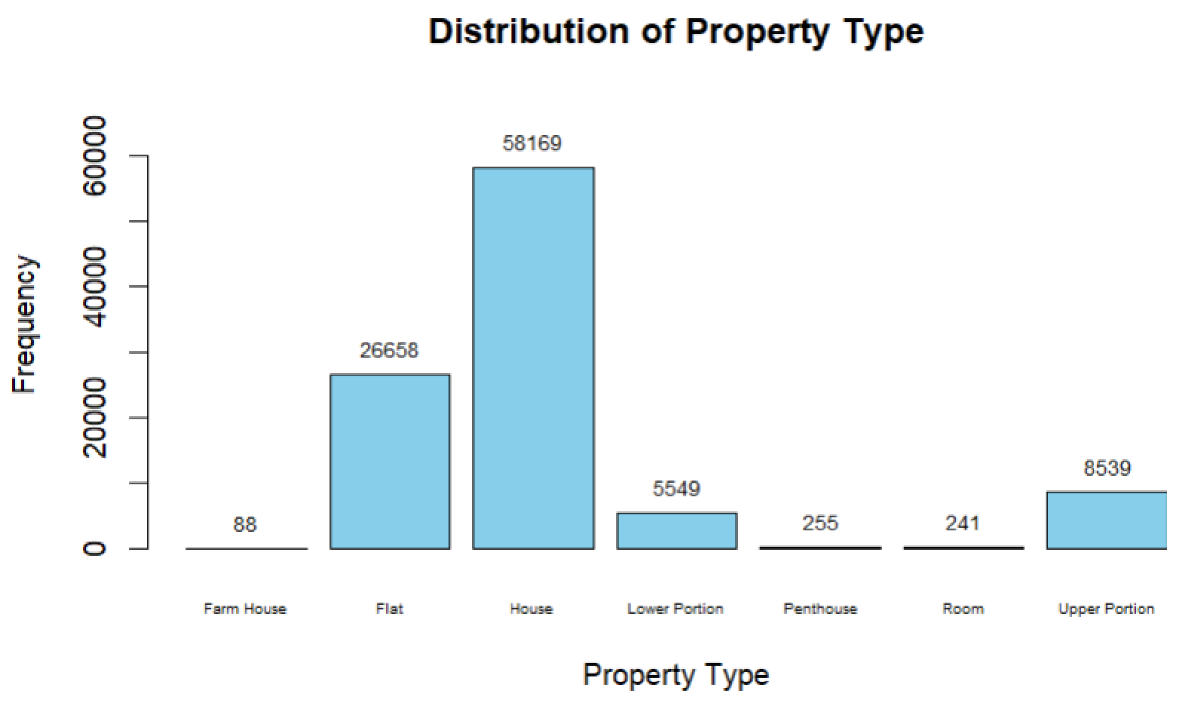


We also show the distribution of the prices of houses.

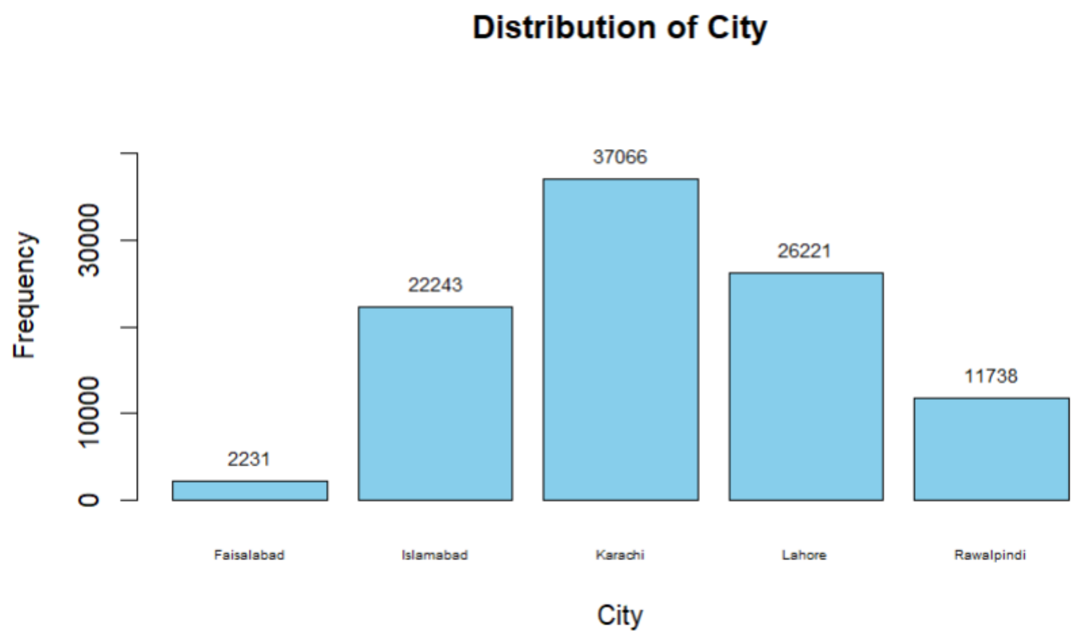


Now we show the distribution of categorical features. We use barplot to visualize the features of property_type, city, and purpose.

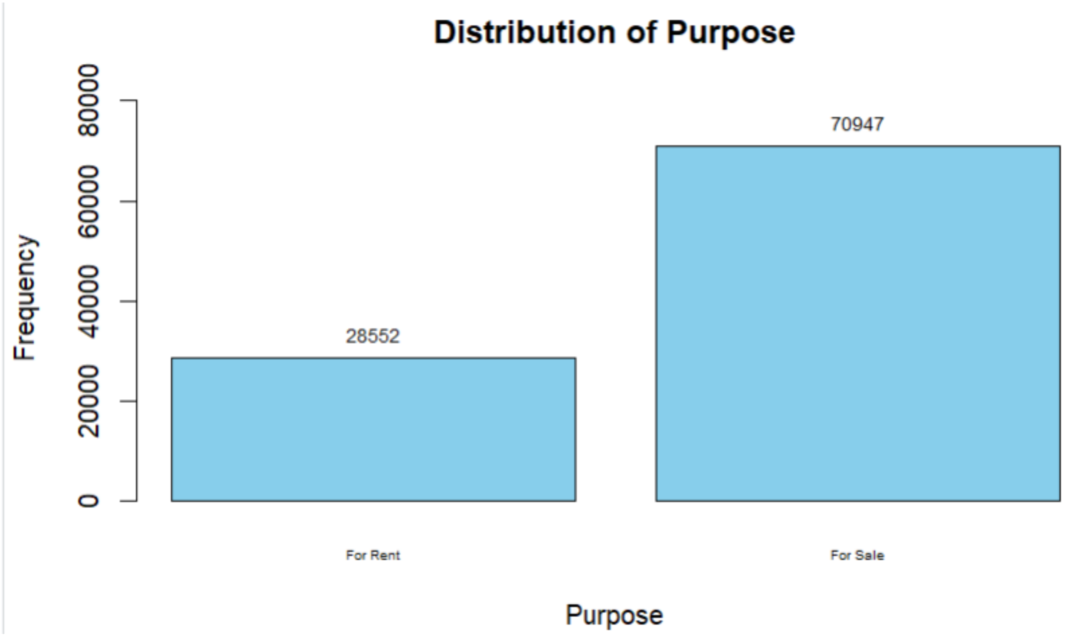
We show the distribution for property type and their total instances. We can observe that house properties have the highest frequency, followed by flat. Farm house, penthouse, and room have the lowest frequencies.



Next we show the distribution of city and their total count. We can see that the most houses are in Karachi city followed by Lahore. The least amount of houses are located in Faissalabad city.

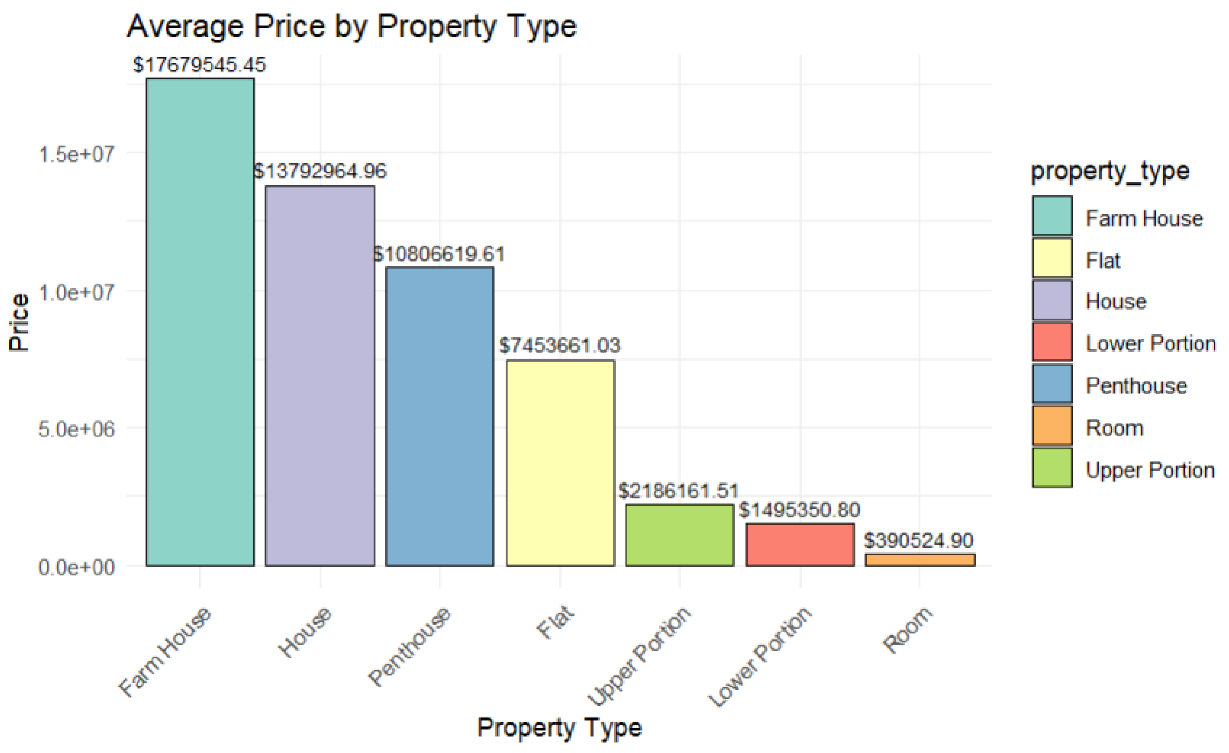


Here we show the distribution of the purpose and their total instance. We can see that there are more houses for sale then there are for rent.

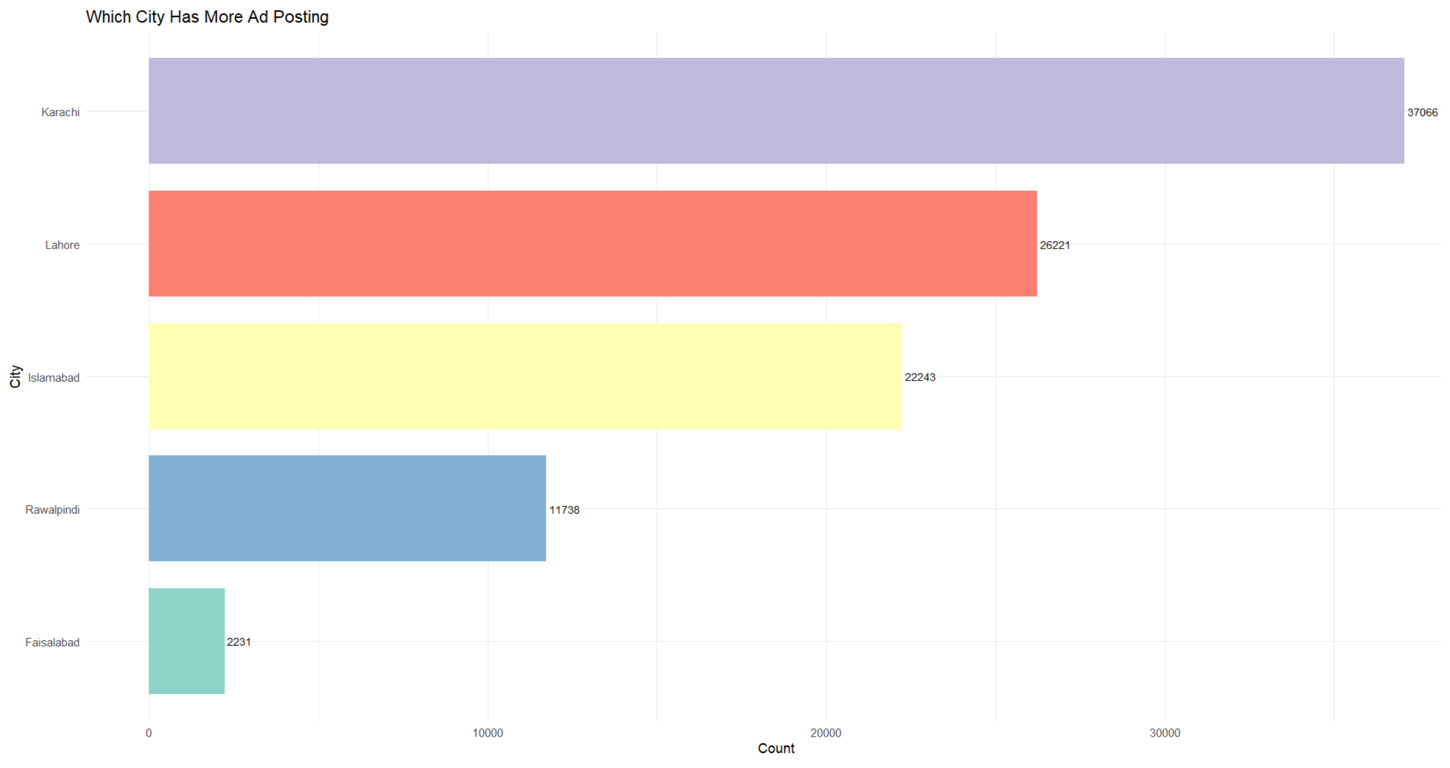


We further show the average price per property type. We can observe that the most expensive property type is farmhouse, followed by house and penthouse. The cheapest type of property is room.

```
# Show average price by property type
df_summary <- housing_data %>%
  group_by(property_type) %>%
  summarise(mean_price = mean(price)) %>%
  arrange(desc(mean_price)) %>%
  head(10)

ggplot(df_summary, aes(x = reorder(property_type, -mean_price), y = mean_price, fill = proper
  geom_bar(stat = "identity", color = "black") +
  labs(x = "Property Type", y = "Price", title = "Average Price by Property Type") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  geom_text(aes(label = sprintf("$%.2f", mean_price)), vjust = -0.5, size = 3) +
  scale_fill_brewer(palette = "Set3")
```



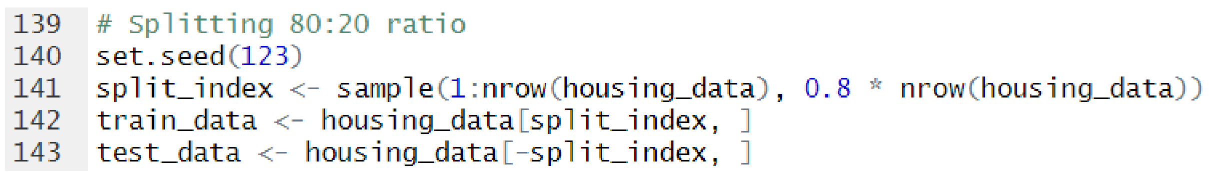
Here we show the ad posting per city. We can observe that Karachi city has the most ad posting with 37066 instances and the lowest is in Faisalabad city with 2231 instances.



Now we can begin data preparation for modelling. We use the as.factor method to convert categorical features of char datatype into factor datatype.

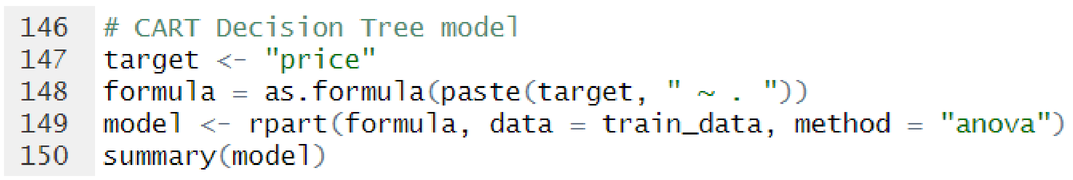
```
# Prepare the data
housing_data$property_type <- as.factor(housing_data$property_type)
housing_data$location <- as.factor(housing_data$location)
housing_data$city <- as.factor(housing_data$city)
```

We split the data into training and testing sets. We use the standard 80:20 ratio.

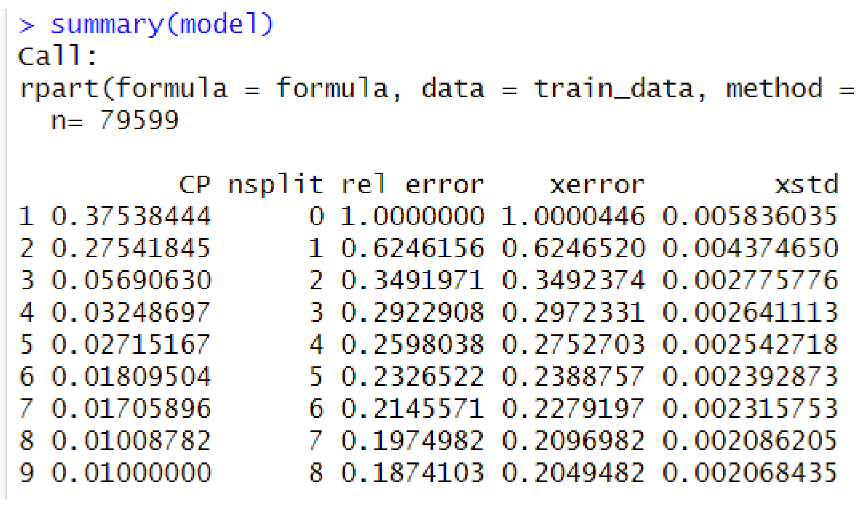


```
# Splitting 80:20 ratio
set.seed(123)
split_index <- sample(1:nrow(housing_data), 0.8 * nrow(housing_data))
train_data <- housing_data[split_index, ]
test_data <- housing_data[-split_index, ]
```

We create the model for our project. Here we use the CART decision tree model. We specify the target as price since this is the dependent variable which we will predict. The remaining features are the independent variable which will be used to predict our dependent variable. We train our model with our training set and through anova. We then show the summary of the model.



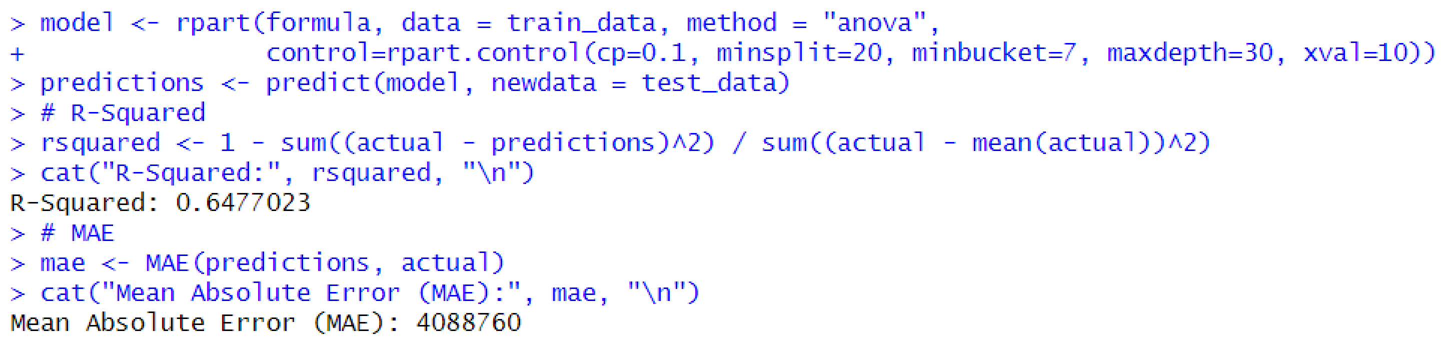
```
# CART Decision Tree model
target <- "price"
formula = as.formula(paste(target, " ~ . "))
model <- rpart(formula, data = train_data, method = "anova")
summary(model)
```



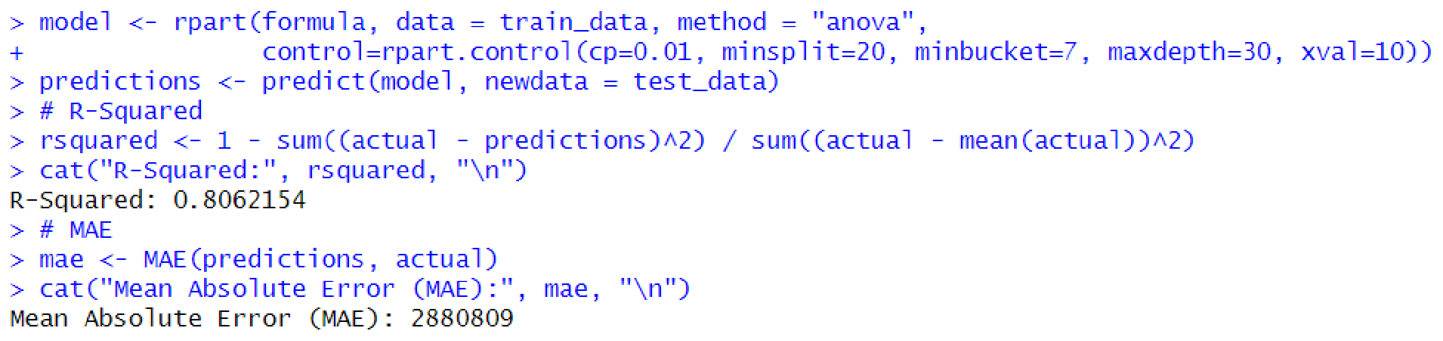
```
> summary(model)
Call:
rpart(formula = formula, data = train_data, method = "anova")
  n= 79599

          CP nsplit rel error    xerror        xstd
1 0.37538444      0 1.0000000 1.0000446 0.005836035
2 0.27541845      1 0.6246156 0.6246520 0.004374650
3 0.05690630      2 0.3491971 0.3492374 0.002775776
4 0.03248697      3 0.2922908 0.2972331 0.002641113
5 0.02715167      4 0.2598038 0.2752703 0.002542718
6 0.01809504      5 0.2326522 0.2388757 0.002392873
7 0.01705896      6 0.2145571 0.2279197 0.002315753
8 0.01008782      7 0.1974982 0.2096982 0.002086205
9 0.01000000      8 0.1874103 0.2049482 0.002068435
```

Now we evaluate the model based on the testing data. Here the data performs predictions on data it has not yet seen after it has been trained using the train dataset. We evaluate the performance of the model using the mean absolute error (MAE), mean squared error (MSE), and R-squared. We specify 5 hyperparameters for the model namely cost complexity pruning (CP), minsplit, minbucket, maxdepth, and xval. For CP, we compare the values of 0.1, 0.01 and 0.001. For minsplit, we compare the values 10 and 20. We compare the values of 7, 15, and 20 in evaluating the minbucket. We also evaluate 10, 20, and 0 for maxdepth. We also compare 5, 10, and 20 for xval parameter.

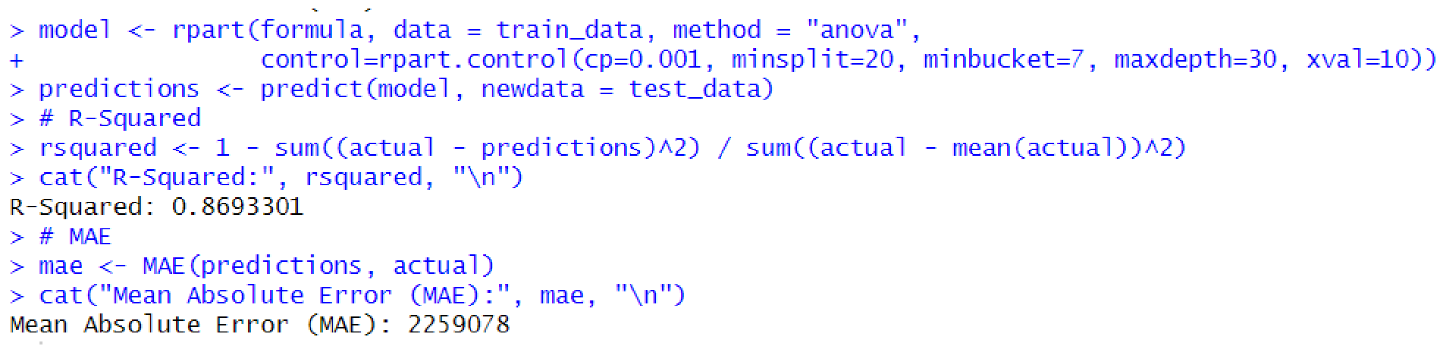
The CP value describes the cost of adding extra nodes to a tree and is used so we can prune the tree. We used 0.1, 0.01 and 0.001. We can see that in decreasing the CP value, the model performs better. So the optimal value is 0.001



```
> model <- rpart(formula, data = train_data, method = "anova",
+                control=rpart.control(cp=0.1, minsplit=20, minbucket=7, maxdepth=30, xval=10))
> predictions <- predict(model, newdata = test_data)
> # R-Squared
> rsquared <- 1 - sum((actual - predictions)^2) / sum((actual - mean(actual))^2)
> cat("R-Squared:", rsquared, "\n")
R-Squared: 0.6477023
> # MAE
> mae <- MAE(predictions, actual)
> cat("Mean Absolute Error (MAE):", mae, "\n")
Mean Absolute Error (MAE): 4088760
```

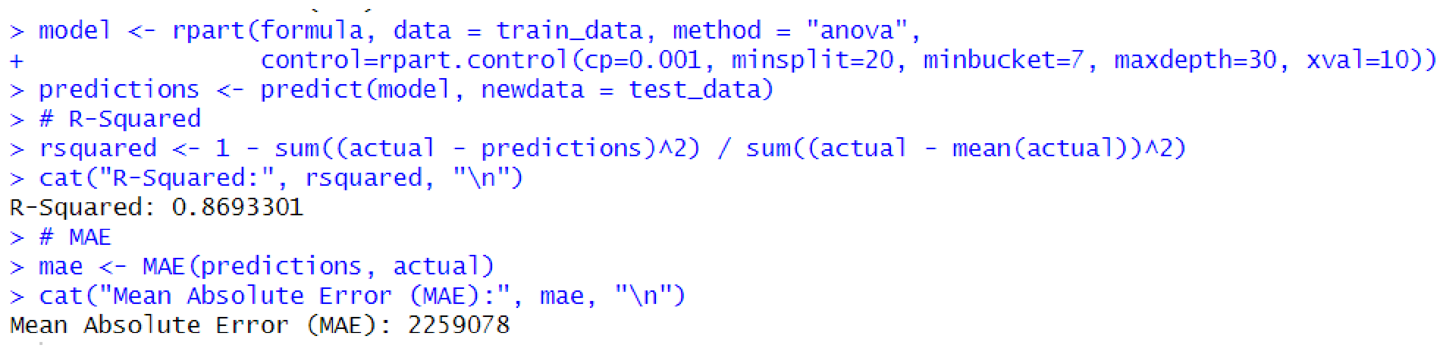


```
> model <- rpart(formula, data = train_data, method = "anova",
+                control=rpart.control(cp=0.01, minsplit=20, minbucket=7, maxdepth=30, xval=10))
> predictions <- predict(model, newdata = test_data)
> # R-Squared
> rsquared <- 1 - sum((actual - predictions)^2) / sum((actual - mean(actual))^2)
> cat("R-Squared:", rsquared, "\n")
R-Squared: 0.8062154
> # MAE
> mae <- MAE(predictions, actual)
> cat("Mean Absolute Error (MAE):", mae, "\n")
Mean Absolute Error (MAE): 2880809
```



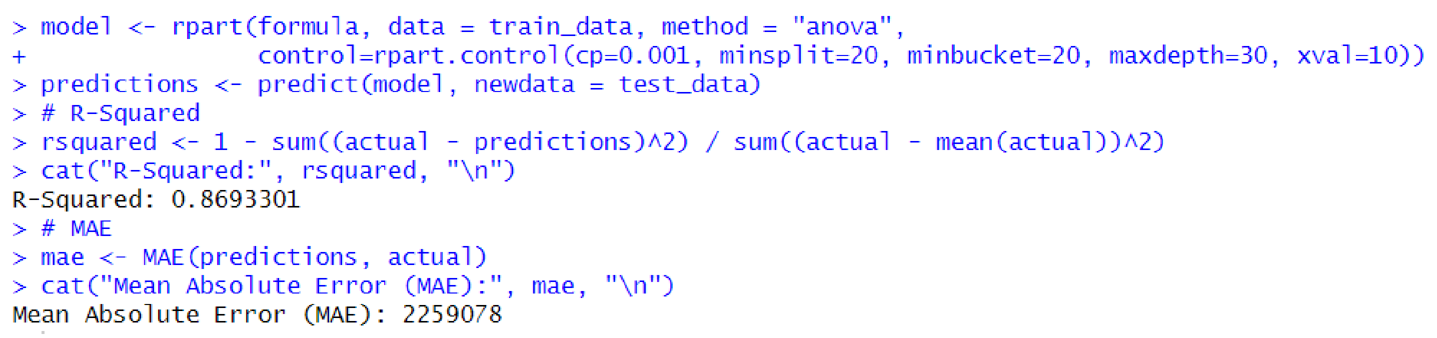
```
> model <- rpart(formula, data = train_data, method = "anova",
+                control=rpart.control(cp=0.001, minsplit=20, minbucket=7, maxdepth=30, xval=10))
> predictions <- predict(model, newdata = test_data)
> # R-Squared
> rsquared <- 1 - sum((actual - predictions)^2) / sum((actual - mean(actual))^2)
> cat("R-Squared:", rsquared, "\n")
R-Squared: 0.8693301
> # MAE
> mae <- MAE(predictions, actual)
> cat("Mean Absolute Error (MAE):", mae, "\n")
Mean Absolute Error (MAE): 2259078
```

The minsplit parameter defines the minimum number of observations that a node should have to split it during the creation of the tree. We evaluate the values of 10 and 20 while keeping the other parameters constant. There is no change in model performance upon changing the values. Thus, a value of 20 is the default and optimal value since our dataset is large.



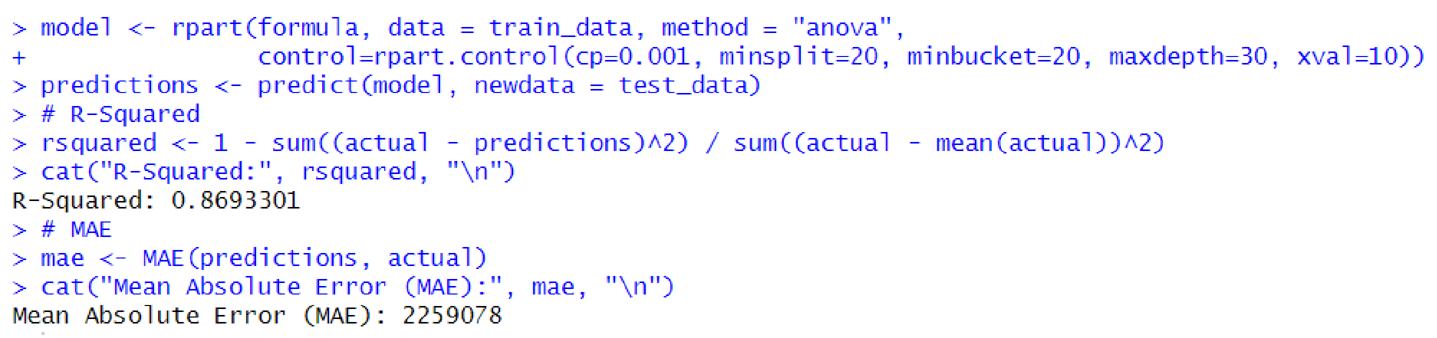
```
> model <- rpart(formula, data = train_data, method = "anova",
+                control=rpart.control(cp=0.001, minsplit=20, minbucket=7, maxdepth=30, xval=10))
> predictions <- predict(model, newdata = test_data)
> # R-Squared
> rsquared <- 1 - sum((actual - predictions)^2) / sum((actual - mean(actual))^2)
> cat("R-Squared:", rsquared, "\n")
R-Squared: 0.8693301
> # MAE
> mae <- MAE(predictions, actual)
> cat("Mean Absolute Error (MAE):", mae, "\n")
Mean Absolute Error (MAE): 2259078
```

The minbucket parameter defines the minimum number of observations that should exist in any leaf node of the decision tree. We evaluate the values of 7, 15, and 20. Upon evaluating, the performance of the model does not change according to the specified values. So 20 is our optimal and default parameter.



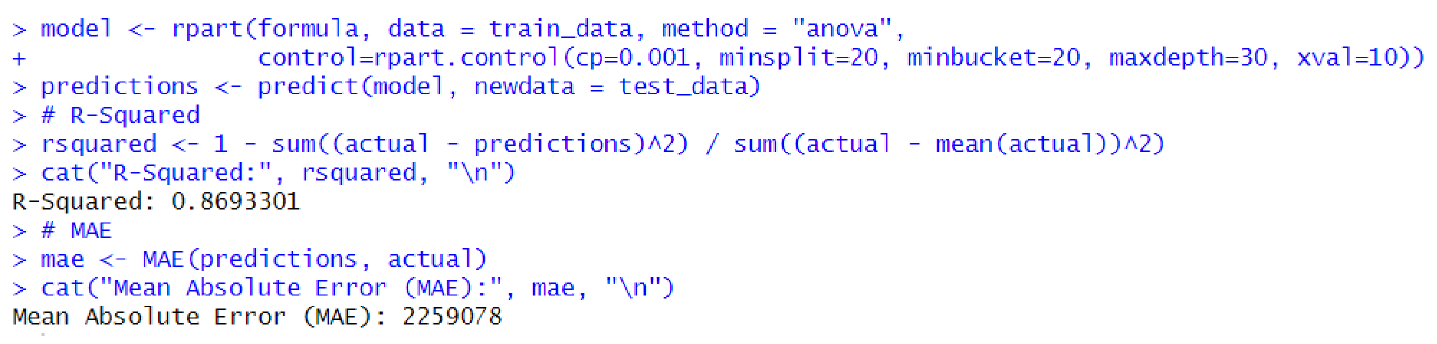
```
> model <- rpart(formula, data = train_data, method = "anova",
+                control=rpart.control(cp=0.001, minsplit=20, minbucket=20, maxdepth=30, xval=10))
> predictions <- predict(model, newdata = test_data)
> # R-Squared
> rsquared <- 1 - sum((actual - predictions)^2) / sum((actual - mean(actual))^2)
> cat("R-Squared:", rsquared, "\n")
R-Squared: 0.8693301
> # MAE
> mae <- MAE(predictions, actual)
> cat("Mean Absolute Error (MAE):", mae, "\n")
Mean Absolute Error (MAE): 2259078
```

The maxdepth parameter shows the number of nodes between the root node to the farthest node of the tree. It controls the maximum levels of the tree. We use and compare 10, 20, and 30. Upon evaluating, there is also no change in the performance of the model with the specified values. Thus, we use the maximum depth parameter of 30 as the optimal value.



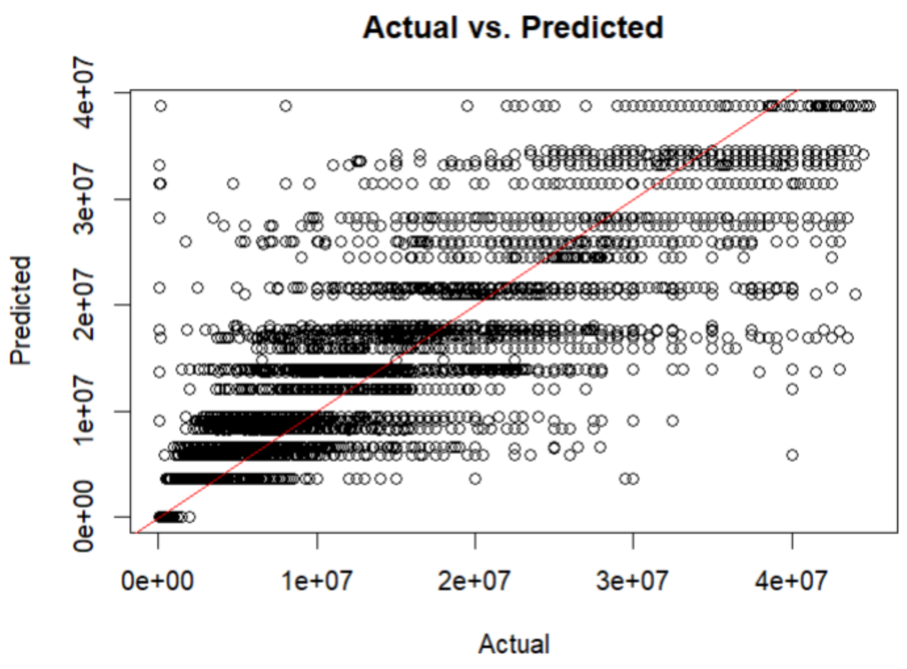
```
> model <- rpart(formula, data = train_data, method = "anova",
+                control=rpart.control(cp=0.001, minsplit=20, minbucket=20, maxdepth=30, xval=10))
> predictions <- predict(model, newdata = test_data)
> # R-Squared
> rsquared <- 1 - sum((actual - predictions)^2) / sum((actual - mean(actual))^2)
> cat("R-Squared:", rsquared, "\n")
R-Squared: 0.8693301
> # MAE
> mae <- MAE(predictions, actual)
> cat("Mean Absolute Error (MAE):", mae, "\n")
Mean Absolute Error (MAE): 2259078
```

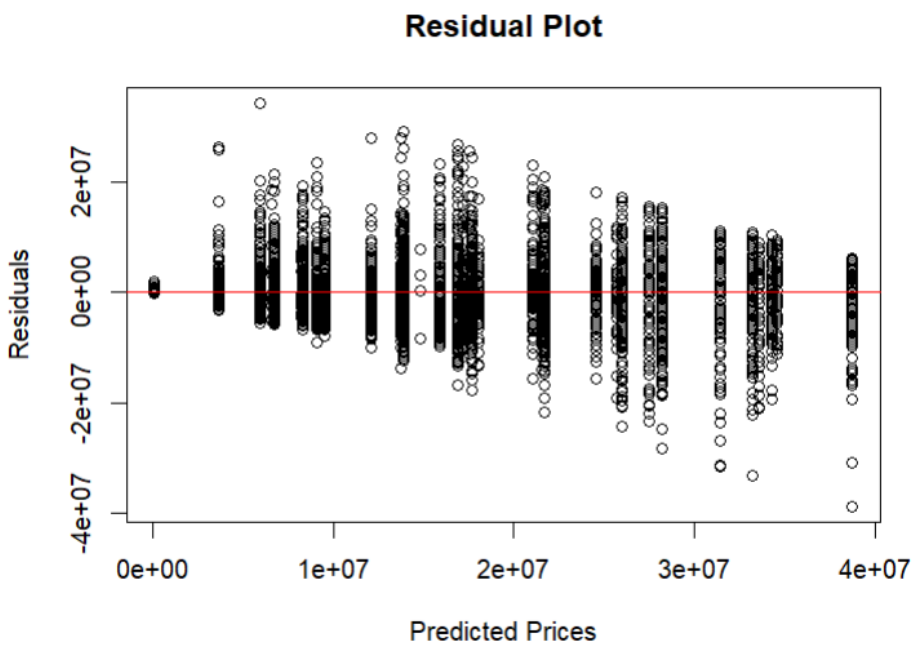
The xval parameter defines the number of cross-validations that should be done for choosing the optimal tree. Here we evaluate the values of 5, 10, and 20. After testing, the model performance does not change when updating the values. So we use the default value of 10 as the optimal value for this parameter.

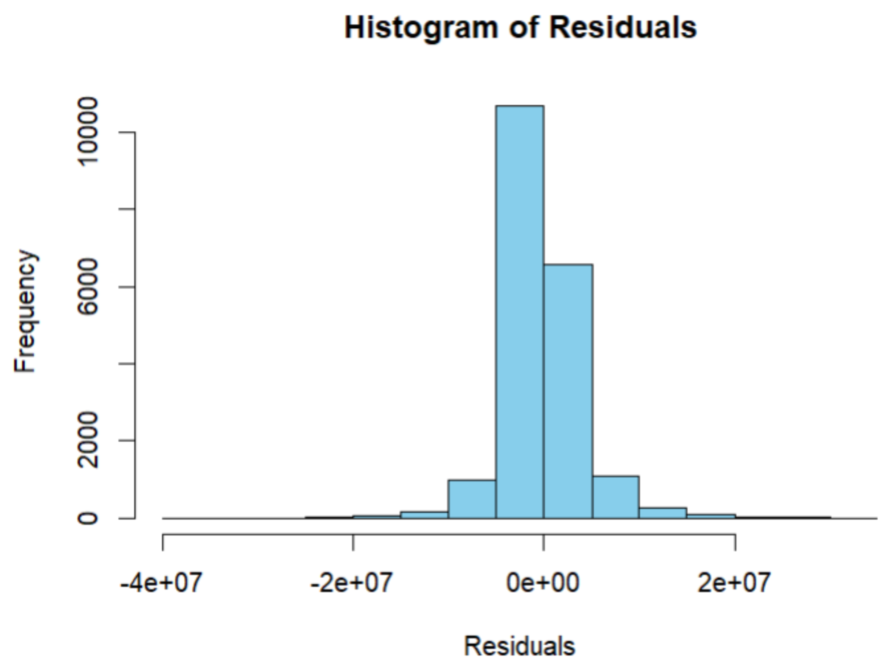


```
> model <- rpart(formula, data = train_data, method = "anova",
+                control=rpart.control(cp=0.001, minsplit=20, minbucket=20, maxdepth=30, xval=10))
> predictions <- predict(model, newdata = test_data)
> # R-Squared
> rsquared <- 1 - sum((actual - predictions)^2) / sum((actual - mean(actual))^2)
> cat("R-Squared:", rsquared, "\n")
R-Squared: 0.8693301
> # MAE
> mae <- MAE(predictions, actual)
> cat("Mean Absolute Error (MAE):", mae, "\n")
Mean Absolute Error (MAE): 2259078
```

With our cart model and our optimized hyperparameters, we can see that the R-Squared performance of the model is 0.8693 or 86.93%. This suggests that the model performs well in explaining the variance in the dependent variable using the independent variables. In this case, it can be approximately 86.93% of the variance in the dependent variable can be explained. We can also view the performance of the model through the scatterplot, residual plot, and residuals distribution. The visualizations provide a better way for understanding the performance of the CART decision tree model. It shows the actual price against the prices predicted by the model. The residuals are also shown which show the differences between the observed values and values predicted by the model.







REFERENCES

Avjyan, H. (2018). *CART*. Medium. https://medium.com/@hakobavjyan/cart-9e21862ee3d

Dobilas, S. (2021). *CART: Classification and Regression Trees for Clean but Powerful Models*. Towards Data Science. https://towardsdatascience.com/cart-classification-and-regression-trees-for-clean-but-powerful-models-cc89e60b7a85

Gedik, R. (2021). *Decision Trees in R Rpart Library*. YouTube. https://www.youtube.com/watch?v=oSP7uV2Hi2I

Koli, S. (2023). *Decision Trees: A Complete Introduction With Examples*. Medium. https://medium.com/@MrBam44/decision-trees-91f61a42c724

Kuhn, M. (2019). *The caret Package*. GitHub. https://topepo.github.io/caret/