# CO2 Prediction Analysis

Shubhan Tamhane

2025-01-18

Including Plots

Importing necessary libraries

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(car)
```

```
## Loading required package: carData

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode
```

```r
library(randomForest)
```

```
## randomForest 4.7-1.2

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':
##
##     margin
```

Importing data

```r
d1 <- read.csv("/Users/shubhantamhane/Downloads/CO2 Emissions.csv")
colnames(d1)
```

```
##  [1] "Make"                             "Model"
##  [3] "Vehicle.Class"                    "Engine.Size.L."
##  [5] "Cylinders"                        "Transmission"
##  [7] "Fuel.Type"                        "Fuel.Consumption.City..L.100.km."
##  [9] "Fuel.Consumption.Hwy..L.100.km."  "Fuel.Consumption.Comb..L.100.km."
## [11] "Fuel.Consumption.Comb..mpg."      "CO2.Emissions.g.km."
```

Checking for null values

```r
sum(is.na.data.frame(d1))
```

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

Exploring data

```r
head(d1)
```

```
##    Make      Model Vehicle.Class Engine.Size.L. Cylinders Transmission
## 1 ACURA        ILX       COMPACT            2.0         4          AS5
## 2 ACURA        ILX       COMPACT            2.4         4           M6
## 3 ACURA ILX HYBRID       COMPACT            1.5         4          AV7
## 4 ACURA    MDX 4WD   SUV - SMALL            3.5         6          AS6
## 5 ACURA    RDX AWD   SUV - SMALL            3.5         6          AS6
## 6 ACURA        RLX      MID-SIZE            3.5         6          AS6
##   Fuel.Type Fuel.Consumption.City..L.100.km. Fuel.Consumption.Hwy..L.100.km.
## 1         Z                              9.9                             6.7
## 2         Z                             11.2                             7.7
## 3         Z                              6.0                             5.8
## 4         Z                             12.7                             9.1
## 5         Z                             12.1                             8.7
## 6         Z                             11.9                             7.7
##   Fuel.Consumption.Comb..L.100.km. Fuel.Consumption.Comb..mpg.
## 1                              8.5                          33
## 2                              9.6                          29
## 3                              5.9                          48
## 4                             11.1                          25
```

```
## 5                             10.6                            27
## 6                             10.0                            28
##   CO2.Emissions.g.km.
## 1                 196
## 2                 221
## 3                 136
## 4                 255
## 5                 244
## 6                 230
```

```r
str(d1)
```

```
## 'data.frame':    7385 obs. of  12 variables:
##  $ Make                            : chr  "ACURA" "ACURA" "ACURA" "ACURA" ...
##  $ Model                           : chr  "ILX" "ILX" "ILX HYBRID" "MDX 4WD" ...
##  $ Vehicle.Class                   : chr  "COMPACT" "COMPACT" "COMPACT" "SUV - SMALL" ...
##  $ Engine.Size.L.                  : num  2 2.4 1.5 3.5 3.5 3.5 3.5 3.7 3.7 2.4 ...
##  $ Cylinders                       : int  4 4 4 6 6 6 6 6 6 4 ...
##  $ Transmission                    : chr  "AS5" "M6" "AV7" "AS6" ...
##  $ Fuel.Type                       : chr  "Z" "Z" "Z" "Z" ...
##  $ Fuel.Consumption.City..L.100.km.: num  9.9 11.2 6 12.7 12.1 11.9 11.8 12.8 13.4 10.6 ...
##  $ Fuel.Consumption.Hwy..L.100.km. : num  6.7 7.7 5.8 9.1 8.7 7.7 8.1 9 9.5 7.5 ...
##  $ Fuel.Consumption.Comb..L.100.km.: num  8.5 9.6 5.9 11.1 10.6 10 10.1 11.1 11.6 9.2 ...
##  $ Fuel.Consumption.Comb..mpg.     : int  33 29 48 25 27 28 28 25 24 31 ...
##  $ CO2.Emissions.g.km.             : int  196 221 136 255 244 230 232 255 267 212 ...
```

```r
unique(d1$Vehicle.Class)
```

```
##  [1] "COMPACT"                  "SUV - SMALL"
##  [3] "MID-SIZE"                 "TWO-SEATER"
##  [5] "MINICOMPACT"              "SUBCOMPACT"
##  [7] "FULL-SIZE"                "STATION WAGON - SMALL"
##  [9] "SUV - STANDARD"           "VAN - CARGO"
## [11] "VAN - PASSENGER"          "PICKUP TRUCK - STANDARD"
## [13] "MINIVAN"                  "SPECIAL PURPOSE VEHICLE"
## [15] "STATION WAGON - MID-SIZE" "PICKUP TRUCK - SMALL"
```

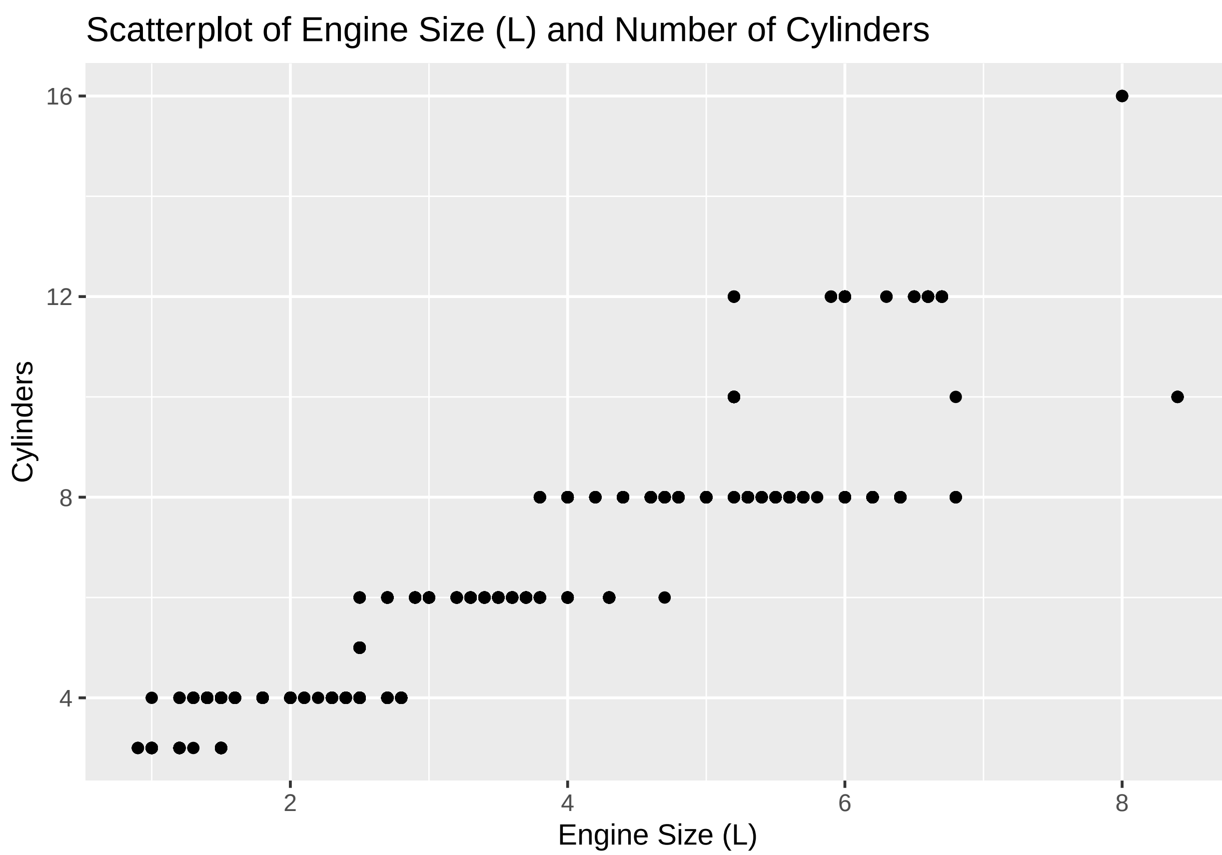
Data Preprocessing

High correlations between features, could cause collinearity issues

```r
cor(d1$Engine.Size.L.,d1$Cylinders)
```

```
## [1] 0.9276529
```

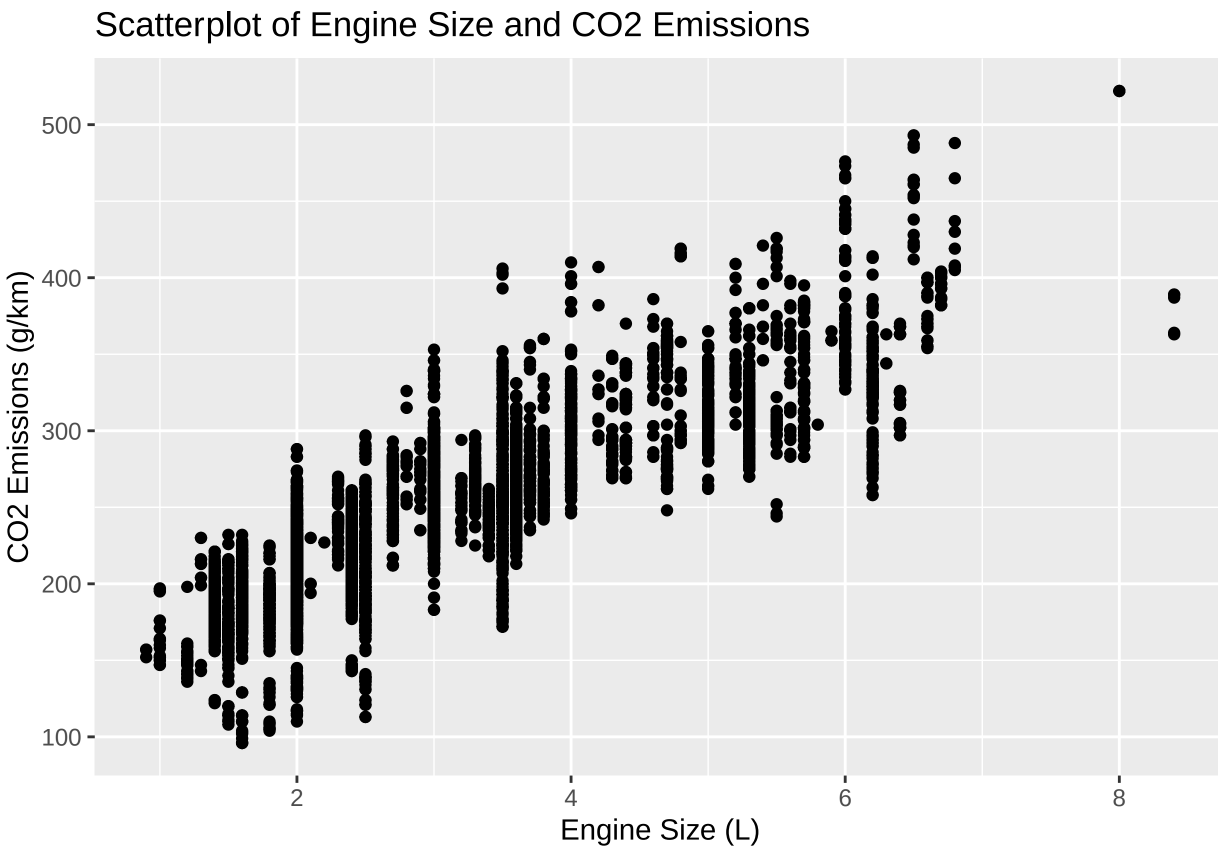
```r
ggplot(d1, aes(x = Engine.Size.L., y = Cylinders)) +
  geom_point() +
  ggtitle("Scatterplot of Engine Size (L) and Number of Cylinders") +
  xlab("Engine Size (L)") +
  ylab("Cylinders")
```




```
cor(d1$Engine.Size.L.,d1$CO2.Emissions.g.km.)
```

```
## [1] 0.8511446
```

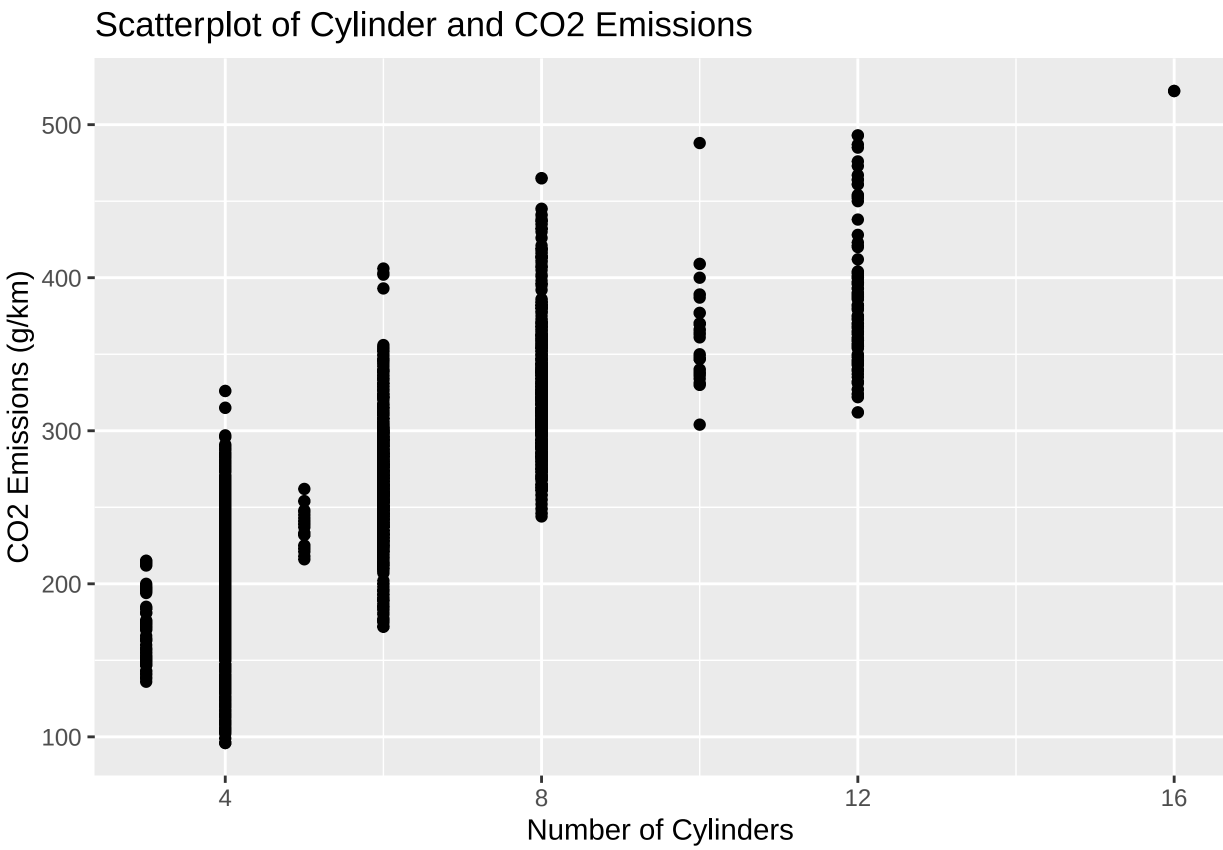
```
ggplot(d1, aes(x = Engine.Size.L., y = CO2.Emissions.g.km.)) +
  geom_point() +
  ggtitle("Scatterplot of Engine Size and CO2 Emissions ") +
  xlab("Engine Size (L)") +
  ylab("CO2 Emissions (g/km)")
```




```
cor(d1$Cylinders, d1$CO2.Emissions.g.km.)
```

```
## [1] 0.8326436
```

```
ggplot(d1, aes(x = Cylinders, y = CO2.Emissions.g.km.)) +
  geom_point() +
  ggtitle("Scatterplot of Cylinder and CO2 Emissions") +
  xlab("Number of Cylinders") +
  ylab("CO2 Emissions (g/km)")
```




Since I want to avoid multicollinearity and engine size has a higher correlation with the target variable, I chose to drop the cylinders feature.

```
d1 <- d1 %>% select(-Cylinders)
```

```
unique(d1$Transmission)
```

```
##  [1] "AS5"  "M6"   "AV7"  "AS6"  "AM6"  "A6"   "AM7"  "AV8"  "AS8"  "A7"
## [11] "A8"   "M7"   "A4"   "M5"   "AV"   "A5"   "AS7"  "A9"   "AS9"  "AV6"
## [21] "AS4"  "AM5"  "AM8"  "AM9"  "AS10" "A10"  "AV10"
```

Since there were too many models and it would be too difficult to encode them, I decided to drop the feature.

```
d1 <- d1 %>% select(-Model)
```

```
colnames(d1)
```

```
##  [1] "Make"                             "Vehicle.Class"
##  [3] "Engine.Size.L."                   "Transmission"
##  [5] "Fuel.Type"                        "Fuel.Consumption.City..L.100.km."
##  [7] "Fuel.Consumption.Hwy..L.100.km."  "Fuel.Consumption.Comb..L.100.km."
##  [9] "Fuel.Consumption.Comb..mpg."      "CO2.Emissions.g.km."
```

```r
unique(d1$Fuel.Type)
```

```
## [1] "Z" "D" "X" "E" "N"
```

Drop all types of gas mileage except for miles per gallon, as this is the industry norm.

```r
d1 <- d1 %>% select(-Fuel.Consumption.City..L.100.km., -Fuel.Consumption.Hwy..L.100.km., -Fuel.Consumpt
```

```r
colnames(d1)
```

```
## [1] "Make"                         "Vehicle.Class"
## [3] "Engine.Size.L."               "Transmission"
## [5] "Fuel.Type"                    "Fuel.Consumption.Comb..mpg."
## [7] "CO2.Emissions.g.km."
```

```r
unique(d1$Make)
```

```
##  [1] "ACURA"         "ALFA ROMEO"    "ASTON MARTIN"  "AUDI"
##  [5] "BENTLEY"       "BMW"           "BUICK"         "CADILLAC"
##  [9] "CHEVROLET"     "CHRYSLER"      "DODGE"         "FIAT"
## [13] "FORD"          "GMC"           "HONDA"         "HYUNDAI"
## [17] "INFINITI"      "JAGUAR"        "JEEP"          "KIA"
## [21] "LAMBORGHINI"   "LAND ROVER"    "LEXUS"         "LINCOLN"
## [25] "MASERATI"      "MAZDA"         "MERCEDES-BENZ" "MINI"
## [29] "MITSUBISHI"    "NISSAN"        "PORSCHE"       "RAM"
## [33] "ROLLS-ROYCE"   "SCION"         "SMART"         "SRT"
## [37] "SUBARU"        "TOYOTA"        "VOLKSWAGEN"    "VOLVO"
## [41] "GENESIS"       "BUGATTI"
```

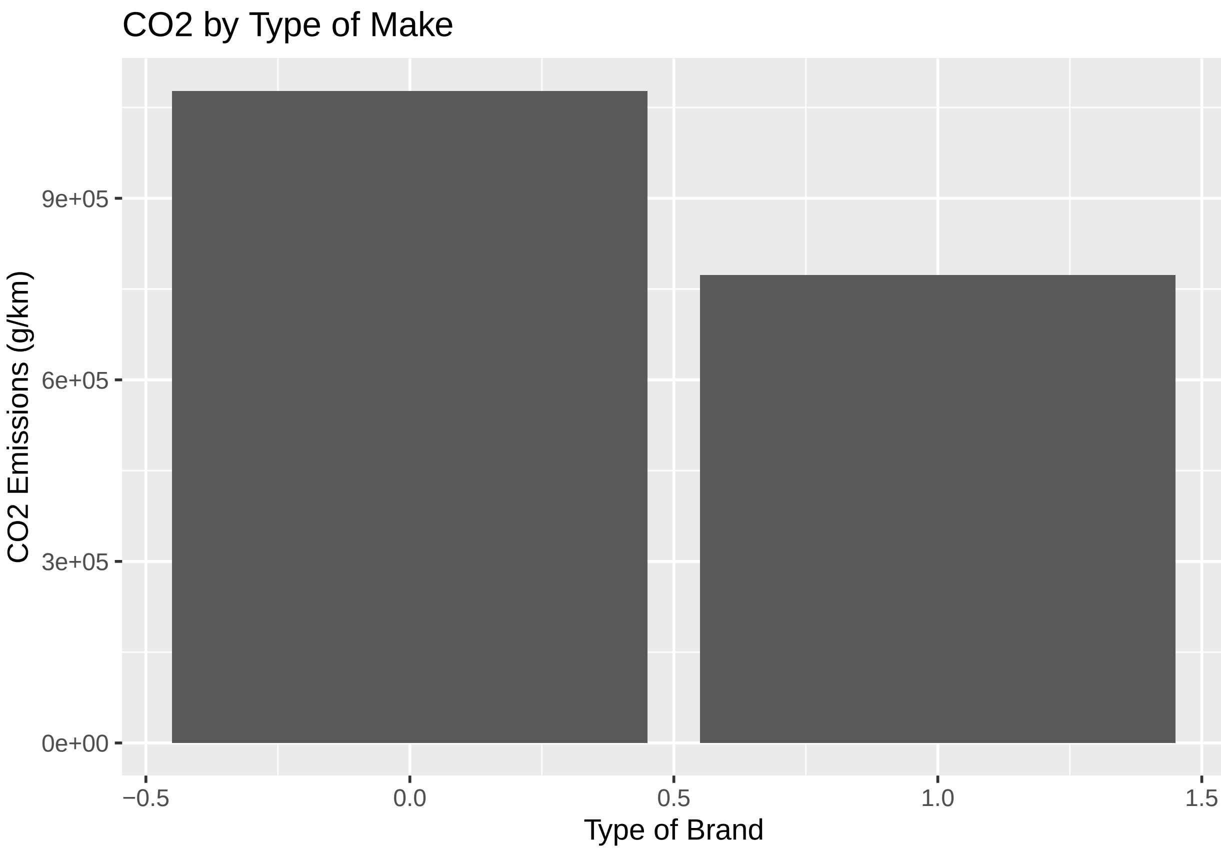
Separate all car makes into either economy or luxury brands. A value of 0 is given to all economy cars and a 1 to all luxury brands. Add this column to the d1 dataframe.

```r
economy_brands <- c("BUICK", "CHEVROLET", "CHRYSLER", "DODGE", "FIAT", "FORD", "GMC", "HONDA", "HYUNDAI


luxury_brands <- c("ACURA", "ALFA ROMEO", "ASTON MARTIN", "AUDI", "BENTLEY", "BMW", "CADILLAC", "CADILL

d1$brand_encoded <- ifelse(d1$Make %in% economy_brands, 0, 1)
```

```r
ggplot(d1, aes(x = brand_encoded, y = CO2.Emissions.g.km.)) +
  geom_col() +
  ggtitle("CO2 by Type of Make") +
  xlab("Type of Brand") +
  ylab("CO2 Emissions (g/km)")
```




```
unique(d1$Transmission)
```

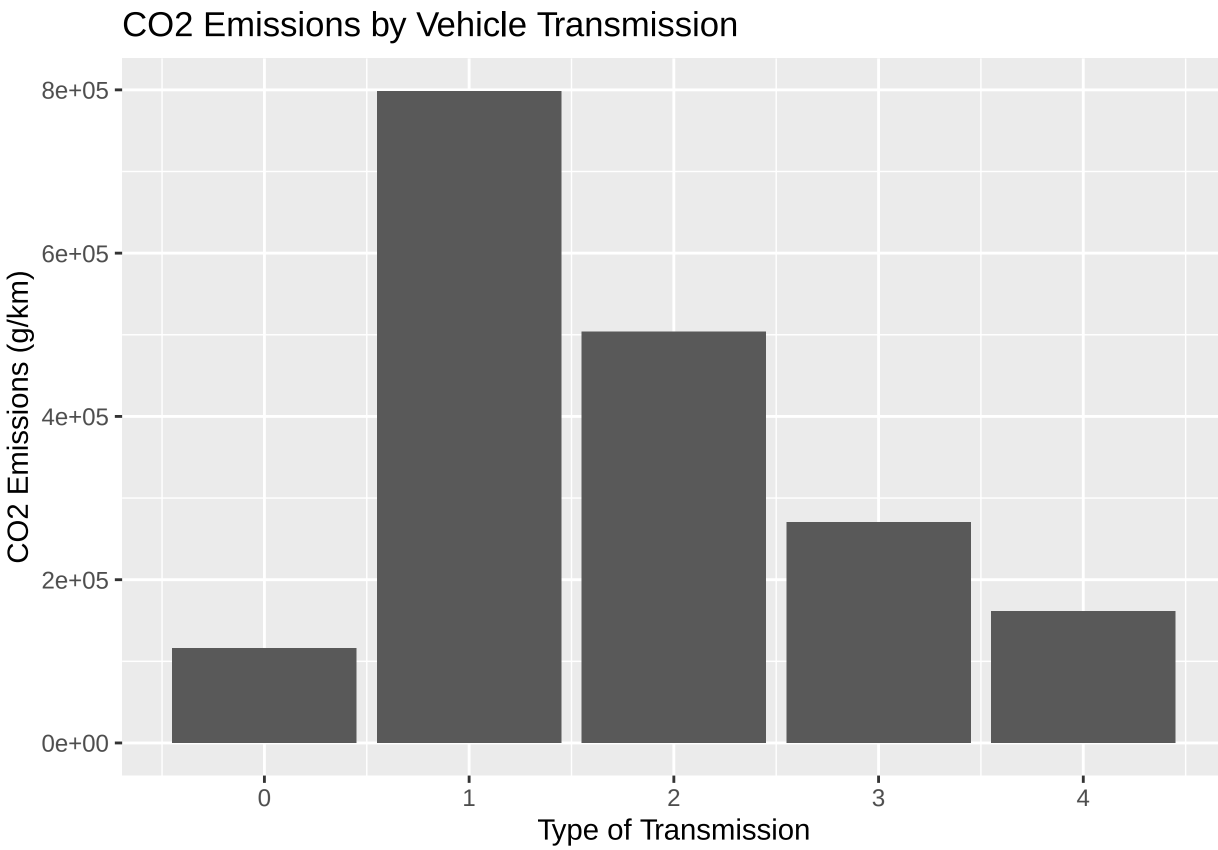
```
##  [1] "AS5"  "M6"   "AV7"  "AS6"  "AM6"  "A6"   "AM7"  "AV8"  "AS8"  "A7"
## [11] "A8"   "M7"   "A4"   "M5"   "AV"   "A5"   "AS7"  "A9"   "AS9"  "AV6"
## [21] "AS4"  "AM5"  "AM8"  "AM9"  "AS10" "A10"  "AV10"
```

To encode the different types of transmissions, I separated them into the following columns: Automatic sequential, automatic, manual, automated manual, and continuously variable transmission. I then encoded these and added them to the d1 dataframe.

```
Automatic_seq <- c("AS5", "AS6", "AS8", "AS7", "AS9", "AS4", "AS10")
Automatic <- c("A6", "A7", "A8", "A4", "A5", "A9")
Manual <- c("M6", "M7", "M5")
Automated_Manual <- c("AM6", "AM7", "AM5", "AM8", "AM9")

d1$Transmission_encoded <- ifelse(d1$Transmission %in% Automatic_seq, 1,
                                  ifelse(d1$Transmission %in% Automatic, 2,
                                         ifelse(d1$Transmission %in% Manual, 3,
                                                ifelse(d1$Transmission %in% Automated_Manual, 4, 0))))
```

```
ggplot(d1, aes(x = Transmission_encoded, y = CO2.Emissions.g.km.)) +
  geom_col() +
  ggtitle("CO2 Emissions by Vehicle Transmission") +
  xlab("Type of Transmission")+
  ylab("CO2 Emissions (g/km)")
```




```
unique(d1$Vehicle.Class)
```

```
##  [1] "COMPACT"                  "SUV - SMALL"
##  [3] "MID-SIZE"                 "TWO-SEATER"
##  [5] "MINICOMPACT"              "SUBCOMPACT"
##  [7] "FULL-SIZE"                "STATION WAGON - SMALL"
##  [9] "SUV - STANDARD"           "VAN - CARGO"
## [11] "VAN - PASSENGER"          "PICKUP TRUCK - STANDARD"
## [13] "MINIVAN"                  "SPECIAL PURPOSE VEHICLE"
## [15] "STATION WAGON - MID-SIZE" "PICKUP TRUCK - SMALL"
```

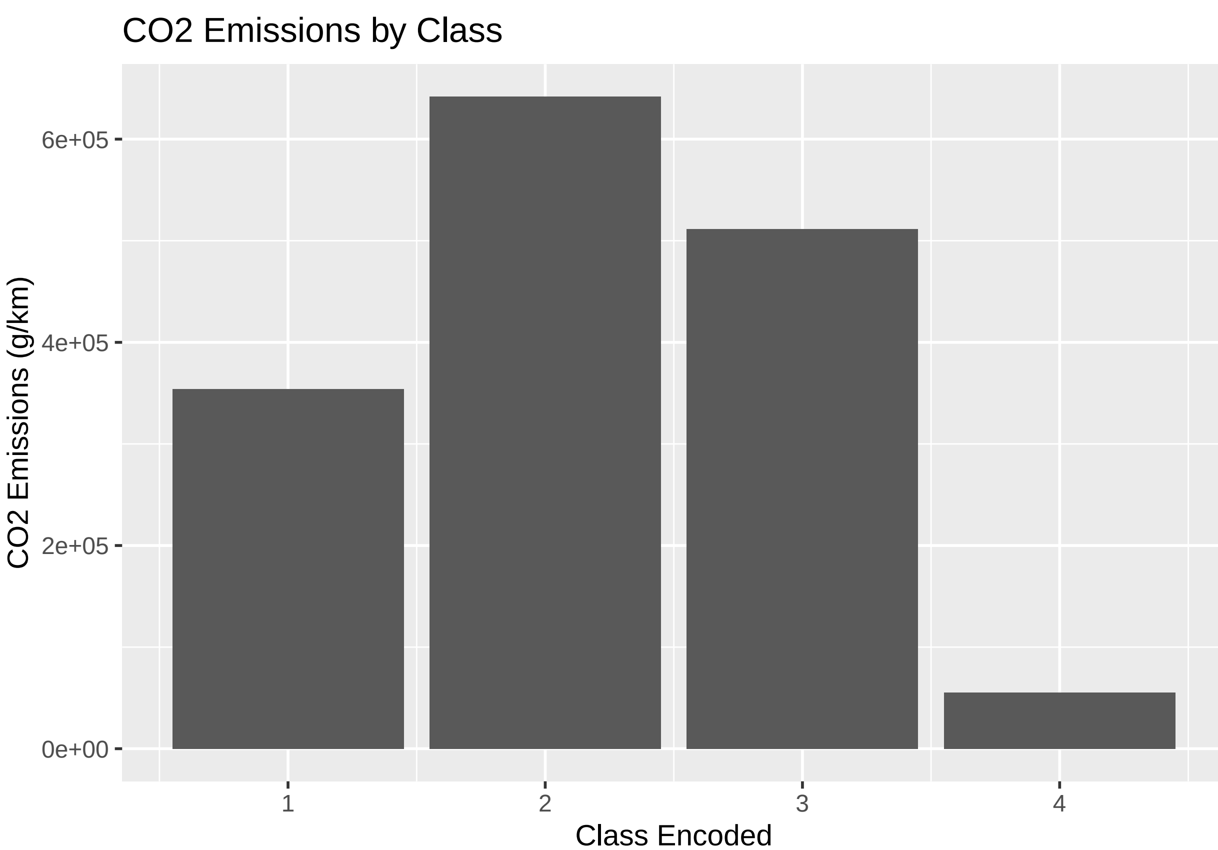
I separated the cars into coupes, sedans, SUVs, vans and trucks as these are the most common types of cars. I then encoded these and added them to the d1 dataframe.

```
coupe <- c("TWO-SEATER", "MINICOMPACT", "SUBCOMPACT")
sedan <- c("COMPACT", "MID-SIZE", "FULL-SIZE")
SUV <- c("SUV - SMALL", "SUV - STANDARD")
VAN <- c("VAN - CARGO", "VAN - PASSENGER", "MINIVAN")
Truck <- c("PICKUP TRUCK - SMALL", "PICKUP TRUCK - STANDARD")

d1$Class_encoded <- ifelse(d1$Vehicle.Class %in% coupe, 1,
                           ifelse(d1$Vehicle.Class %in% sedan, 2,
                                  ifelse(d1$Vehicle.Class %in% SUV, 3,
                                         ifelse(d1$Vehicle.Class %in% VAN, 4,
                                                ifelse(d1$Transmission %in% Truck,5, 0)))))
```

```r
d1 <- subset(d1, Class_encoded != 0)
```

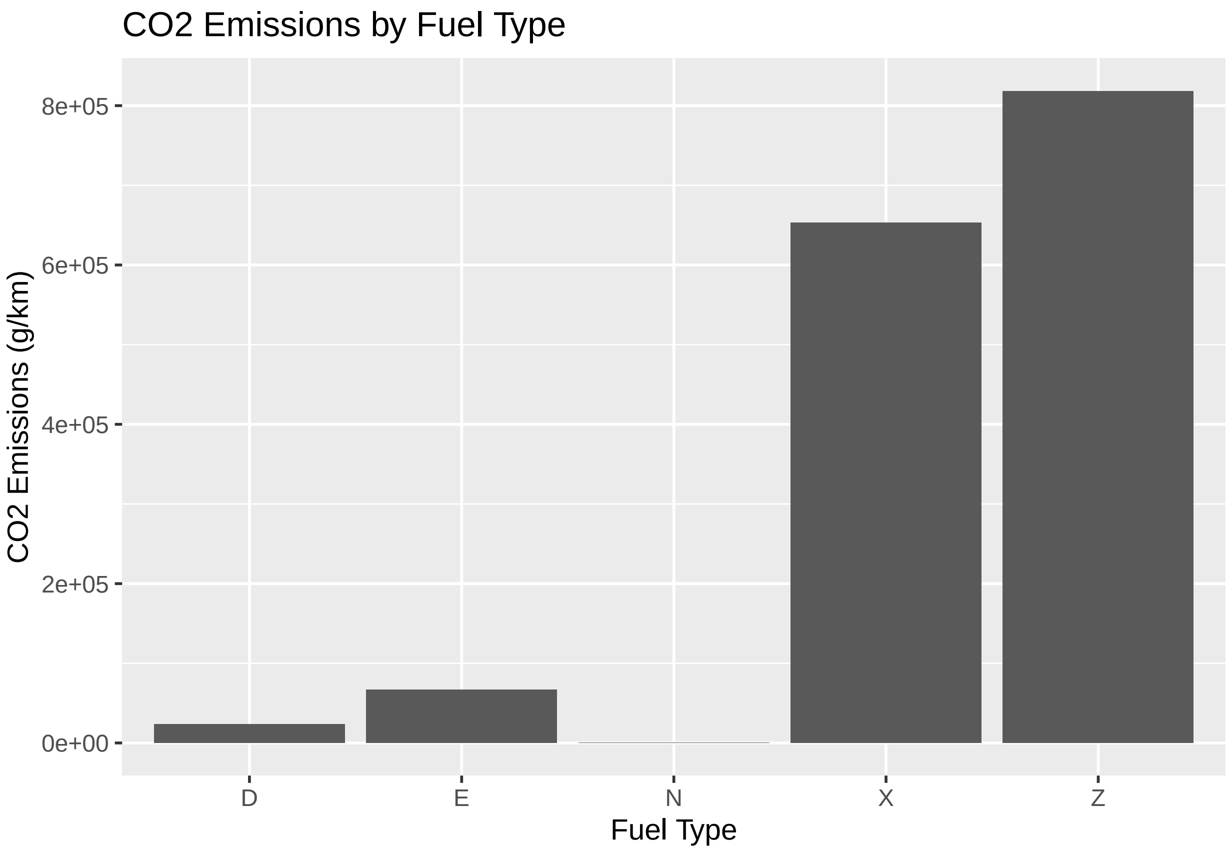
```r
ggplot(d1, aes(x = Class_encoded, y = CO2.Emissions.g.km.)) +
  geom_col() +
  ggtitle("CO2 Emissions by Class") +
  xlab("Class Encoded")+
  ylab("CO2 Emissions (g/km)")
```




I changed the fuel type from a string to a factor variable to encode it. The fuel had many types including normal gas, premium gas, no gas (electric), diesel, and ethanol.

```r
d1$Fuel.Type <- as.factor(d1$Fuel.Type)
```

```r
ggplot(d1, aes(x=Fuel.Type, y = CO2.Emissions.g.km.)) +
  geom_col() +
  ggtitle("CO2 Emissions by Fuel Type") +
  xlab("Fuel Type") +
  ylab("CO2 Emissions (g/km)")
```




Fitting linear model

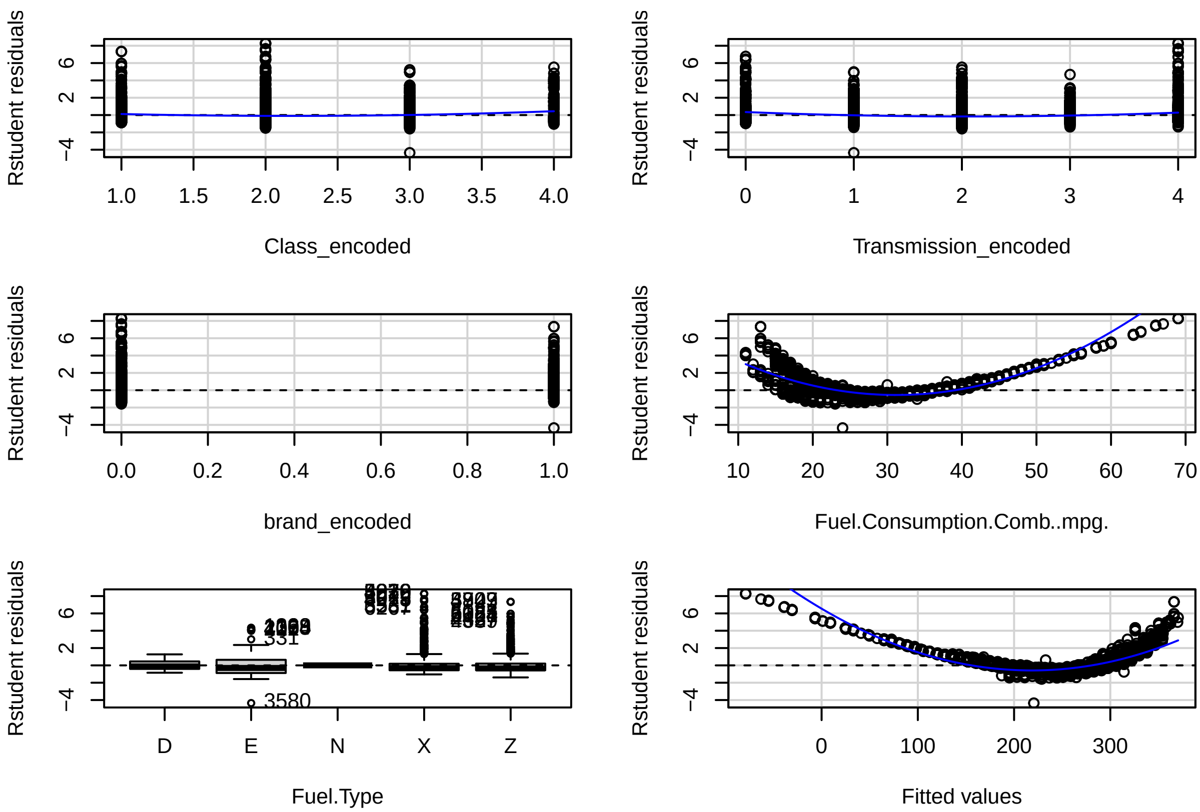
```
m1 <- lm(CO2.Emissions.g.km. ~ Class_encoded + Transmission_encoded + brand_encoded + Fuel.Consumption.C
```

```
summary(m1)
```

```
##
## Call:
## lm(formula = CO2.Emissions.g.km. ~ Class_encoded + Transmission_encoded +
##     brand_encoded + Fuel.Consumption.Comb..mpg. + Fuel.Type,
##     data = d1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -92.533 -12.090  -6.842   4.767 174.884
##
## Coefficients:
##                               Estimate Std. Error  t value Pr(>|t|)
## (Intercept)                  498.17024    3.23313  154.083  < 2e-16 ***
## Class_encoded                  1.67897    0.42519    3.949 7.94e-05 ***
## Transmission_encoded          -0.11372    0.25725   -0.442    0.658
## brand_encoded                 -0.26713    0.76948   -0.347    0.728
## Fuel.Consumption.Comb..mpg.   -7.96035    0.04574 -174.052  < 2e-16 ***
## Fuel.TypeE                   -91.24529    2.63596  -34.616  < 2e-16 ***
## Fuel.TypeN                  -113.28668   21.46294   -5.278 1.35e-07 ***
## Fuel.TypeX                   -30.69284    2.15742  -14.227  < 2e-16 ***
```

```
## Fuel.TypeZ                    -29.42117    2.17885  -13.503  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.34 on 6297 degrees of freedom
## Multiple R-squared:  0.8695, Adjusted R-squared:  0.8694
## F-statistic:  5245 on 8 and 6297 DF,  p-value: < 2.2e-16
```

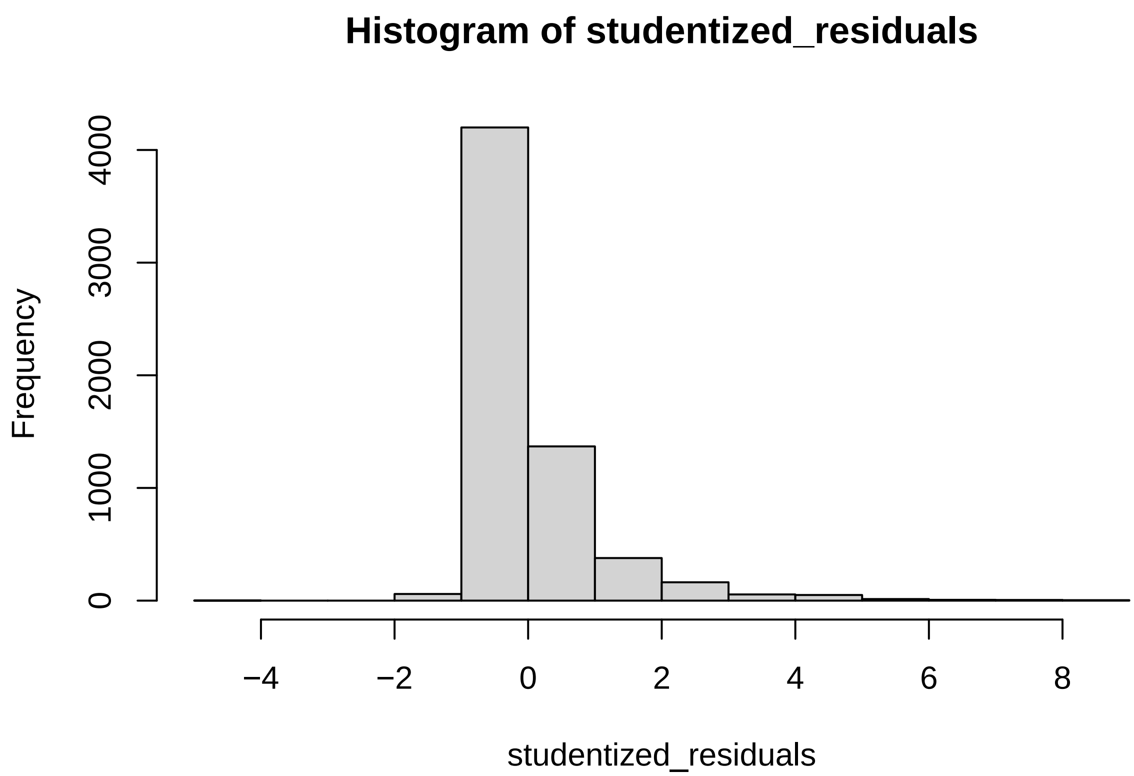
```r
residualPlots(m1, type = "rstudent")
```




```
##                             Test stat Pr(>|Test stat|)
## Class_encoded                  9.4850        <2e-16 ***
## Transmission_encoded          13.6601        <2e-16 ***
## brand_encoded                  0.9395        0.3475
## Fuel.Consumption.Comb..mpg.  161.7271        <2e-16 ***
## Fuel.Type
## Tukey test                   152.8104        <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Since the graph displays heavy curvature, a linear model does not seem to be appropriate here.

```r
studentized_residuals <- rstudent(m1)
hist(studentized_residuals)
```




```
model_rf <- randomForest(d1$CO2.Emissions.g.km. ~ Class_encoded + Transmission_encoded + brand_encoded
summary(model_rf)
```

```
##                 Length Class  Mode
## call               4   -none- call
## type               1   -none- character
## predicted       6306   -none- numeric
## mse              500   -none- numeric
## rsq              500   -none- numeric
## oob.times       6306   -none- numeric
## importance         5   -none- numeric
## importanceSD       0   -none- NULL
## localImportance    0   -none- NULL
## proximity          0   -none- NULL
## ntree              1   -none- numeric
## mtry               1   -none- numeric
## forest            11   -none- list
## coefs              0   -none- NULL
## y               6306   -none- numeric
## test               0   -none- NULL
## inbag              0   -none- NULL
## terms              3   terms  call
```

```
predictions <- predict(model_rf, d1)
y_actual <- d1$CO2.Emissions.g.km.
rss <- sum((y_actual - predictions)^2)
tss <- sum((y_actual - mean(y_actual))^2)
print(1 - (rss / tss))
```

```
## [1] 0.8068461
```

The r^2 value that we are getting here is 81%. This is a decent value, but we can try to enhance it with hyperparameter tuning.

```r
install.packages("randomForest")
```

```
##
## The downloaded binary packages are in
##  /var/folders/c3/2w85_lw50qdgkzcnmbnz057m0000gn/T//Rtmp4eGXuX/downloaded_packages
```

```r
install.packages("caret")
```

```
##
## The downloaded binary packages are in
##  /var/folders/c3/2w85_lw50qdgkzcnmbnz057m0000gn/T//Rtmp4eGXuX/downloaded_packages
```

```r
library(randomForest)
library(caret)
```

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

```r
set.seed(123)

# Define a tuning grid for mtry
tune_grid <- expand.grid(
  mtry = seq(2, 6, by = 1)
)
```

Using grid search to find best possible parameters for random forest model.

```r
control <- trainControl(
  method = "cv",
  number = 5,
  search = "grid"
)
```

Using the caret package, we can use the optimal parameters to

```r
tuned_model <- train(
  CO2.Emissions.g.km. ~ Class_encoded + Transmission_encoded + brand_encoded + Fuel.Consumption.Comb..mp
  data = d1,
  method = "rf",
  tuneGrid = tune_grid,
  trControl = control,
  ntree = 500
)

print(tuned_model$bestTune)
```

```
##   mtry
## 5    6
```

```r
print(tuned_model)
```

```
## Random Forest
##
## 6306 samples
##    5 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 5045, 5046, 5043, 5045, 5045
## Resampling results across tuning parameters:
##
##   mtry  RMSE       Rsquared   MAE
##   2     21.812118  0.9293333  14.824141
##   3     10.847032  0.9786125   7.049744
##   4      6.007994  0.9909151   4.027598
##   5      4.768108  0.9935391   3.327970
##   6      4.558736  0.9940332   3.204856
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 6.
```