Statistics Project

Amin Hashemi

Summer 2023

Phase 1: Load and Process data

Loading Libraries

Let's load required libraries

```
suppressPackageStartupMessages({
  library("MASS")
  library("stringr")
  library("dplyr")
  library("moments")
  library("readr")
  library("ggplot2")
  library("reshape2")
  library("corrplot")
  library("psych")
  library("car")  #vif
  library("lmtest")
})
```

Loading and cleaning

```
carprice <- read.csv("CarPrice_Assignment.csv",stringsAsFactors = F)
```

Creating independent variable Car Company from Car Name variable

for a better analyze, and finding outliers, let's extract car company from the car name. (because of the pattern in carnames we are able to do so.)

```
carprice$carcompany <- word(carprice$CarName,1)
```

Changing the type of Categorical variables to factor.

Using the `as.factor()` command

```
carprice$symboling <- as.factor(carprice$symboling)
carprice$fueltype <- as.factor(carprice$fueltype)
carprice$aspiration <- as.factor(carprice$aspiration)
carprice$doornumber <- as.factor(carprice$doornumber)
carprice$carbody <- as.factor(carprice$carbody)
carprice$drivewheel <- as.factor(carprice$drivewheel)
carprice$enginelocation <- as.factor(carprice$enginelocation)
carprice$enginetype <- as.factor(carprice$enginetype)
```

```r
carprice$cylindernumber <- as.factor(carprice$cylindernumber)
carprice$fuelsystem <- as.factor(carprice$fuelsystem)
carprice$carcompany <- as.factor(carprice$carcompany)
```

Removing duplicate values (if any) in the dataset.

Using the `unique()` command

```r
unique(carprice)
```

We observe that the number of observations doesn't change thus no duplicates are found in the dataset.

Checking for missing values and treat if any.

Using `sum(is.na())` to check if there are any missing values

```r
sum(is.na(carprice))
```

```
## [1] 41
```

```r
sum(carprice == "", na.rm = TRUE)
```

```
## [1] 37
```

We do have empty data. so let's handle them

We have two types of data. Numeric and Categorical.

- We replace Numeric data with *normal distribution* with **mean**
- We replace Numeric data with *skewed distribution* with **median**

```r
# Function to decide whether to use mean or median based on skewness
impute_mean_median <- function(x) {
  if (is.numeric(x)) {
    if(abs(skewness(x, na.rm = TRUE)) < 1) {
      return(ifelse(is.na(x), mean(x, na.rm = TRUE), x))
    } else {
      return(ifelse(is.na(x), median(x, na.rm = TRUE), x))
    }
  } else {
    return(x)
  }
}


# Apply the function to each column
carprice <- carprice %>% mutate(across(everything(), impute_mean_median))
```

We replace Categorical data with **mode**.

```r
# Identify categorical columns
categorical_columns <- names(carprice)[sapply(carprice, is.factor)]

for (col in categorical_columns) {
  # Calculate mode
  mode_value <- names(which.max(table(carprice[[col]])))

  # Replace empty strings with mode value
```

```r
  carprice[[col]][carprice[[col]] == ""] <- mode_value
}
```

Now let's check again for NA values:

```r
sum(is.na(carprice))
```

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

```r
sum(carprice == "", na.rm = TRUE)
```

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

So we have no more empty values. we handled our unavailable data

Checking levels for various categorical variables

```r
summary(carprice$symboling)
```

```
## -2 -1  0  1  2  3
##  3 22 67 54 32 27
```

```r
summary(carprice$fueltype)
```

```
## diesel    gas
##     20    185
```

```r
summary(carprice$aspiration)
```

```
##   std turbo
##   168    37
```

```r
summary(carprice$doornumber)
```

```
## four  two
##  115   90
```

```r
summary(carprice$carbody)
```

```
##             convertible     hardtop   hatchback       sedan       wagon
##           0           6           5          68         104          22
```

```r
summary(carprice$drivewheel)
```

```
## 4wd fwd rwd
##   9 120  76
```

```r
summary(carprice$enginelocation)
```

```
## front  rear
##   202     3
```

```r
summary(carprice$enginetype)
```

```
##  dohc dohcv     l   ohc  ohcf  ohcv rotor
##    12     1    12   148    15    13     4
```

```r
summary(carprice$cylindernumber)
```

```
##         eight   five   four    six  three twelve    two
##      0      5      9    164     21      1      1      4
```

```r
summary(carprice$fuelsystem)
```

```
## 1bbl 2bbl 4bbl  idi  mfi mpfi spdi spfi
##   11   66    3   20    1   94    9    1
```

```r
summary(carprice$carcompany)
```

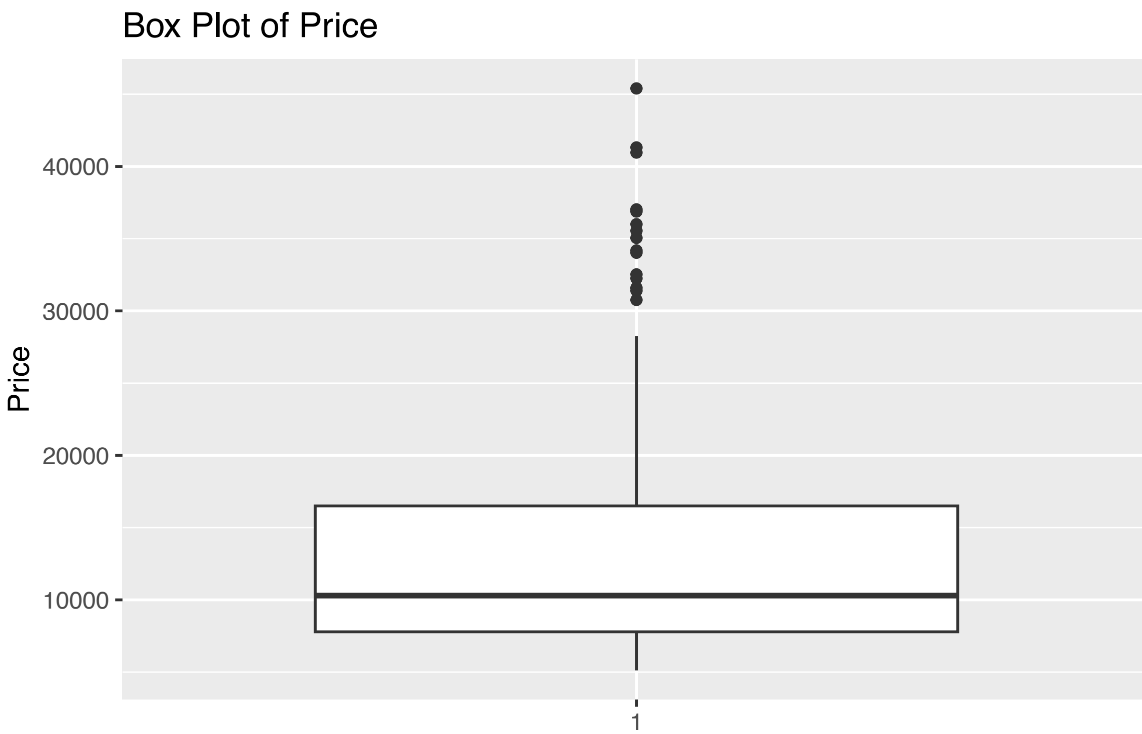
```
## alfa-romero        audi         bmw       buick   chevrolet       dodge
##           3           7           8           8           3           9
##       honda       isuzu      jaguar       maxda       mazda     mercury
##          13           4           3           2          15           1
##  mitsubishi      nissan      Nissan     peugeot    plymouth    porcshce
##          13          17           1          11           7           1
##     porsche     renault        saab      subaru      toyota     toyouta
##           4           2           6          12          31           1
##   vokswagen  volkswagen       volvo          vw
##           1           9          11           2
```

We Identified issues in carcompany variable levels. Now resolving them:

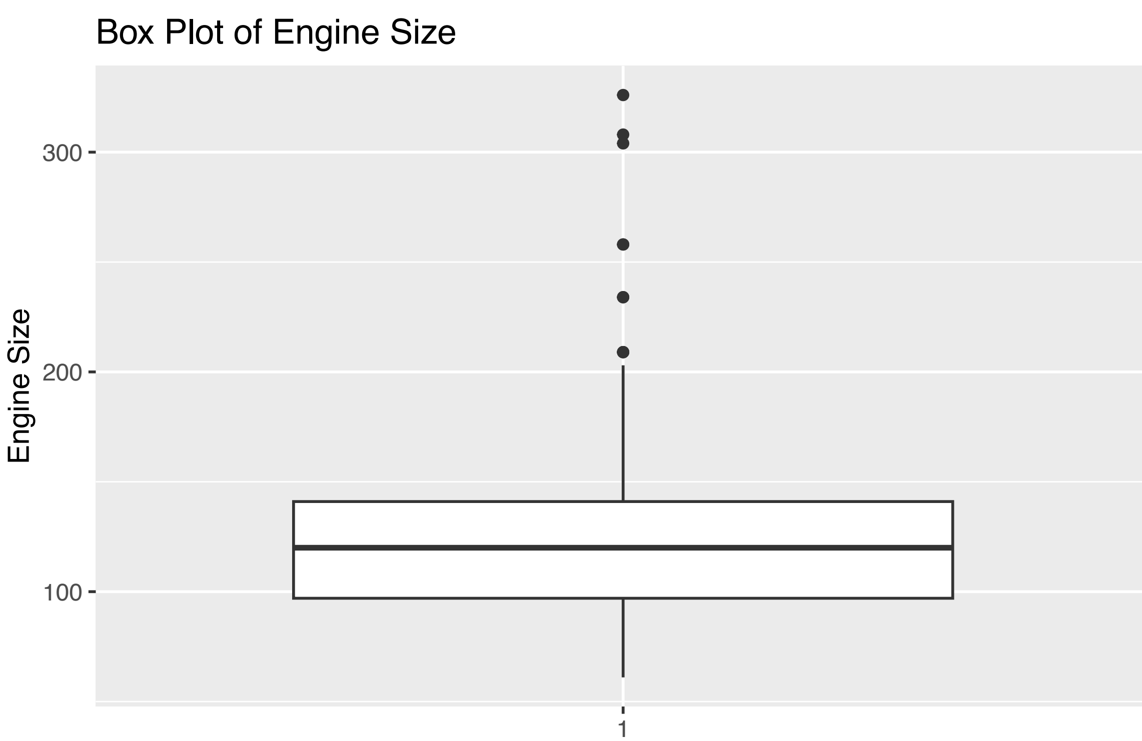
```r
carprice$carcompany[carprice$carcompany == "maxda"] <- "mazda"
carprice$carcompany[carprice$carcompany == "Nissan"] <- "nissan"
carprice$carcompany[carprice$carcompany == "porcshce"] <- "porsche"
carprice$carcompany[carprice$carcompany == "toyouta"] <- "toyota"
carprice$carcompany[carprice$carcompany == "vokswagen"
                    | carprice$carcompany == "vw" ] <- "volkswagen"

levels(carprice$carcompany)[10] <- "mazda"
levels(carprice$carcompany)[14] <- "nissan"
levels(carprice$carcompany)[17] <- "porcshce"
levels(carprice$carcompany)[21] <- "toyota"
levels(carprice$carcompany)[21] <- "volkswagen"
levels(carprice$carcompany)[23] <- "volkswagen"
```

Create box plots using ggplot2

```r
boxplot_data <- data.frame(
  Price = carprice$price,
  EngineSize = carprice$enginesize,
  Horsepower = carprice$horsepower
)

# Draw box plots
scale_factor <- 0.7

ggplot(boxplot_data, aes(x = factor(1), y = Price)) +
  geom_boxplot() +
  labs(x = "", y = "Price") +
  ggtitle("Box Plot of Price") +
  theme(plot.margin = unit(c(1, 1, 1, 1) * scale_factor, "cm"))
```

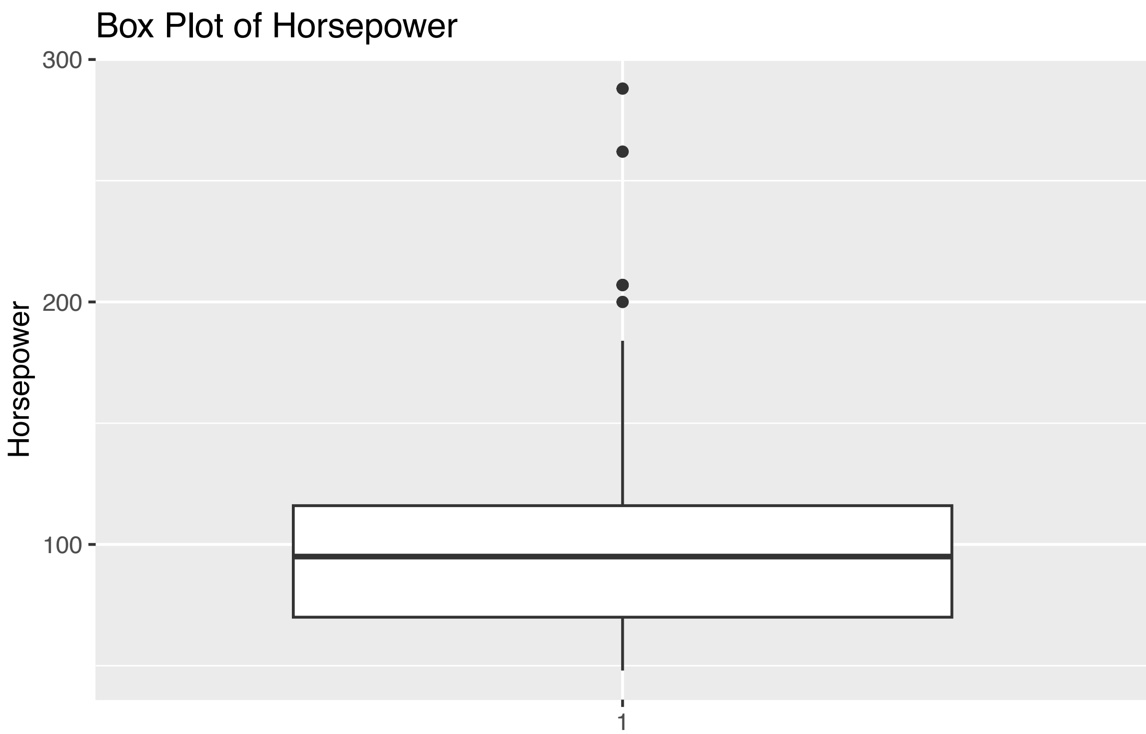



```r
ggplot(boxplot_data, aes(x = factor(1), y = EngineSize)) +
  geom_boxplot() +
  labs(x = "", y = "Engine Size") +
  ggtitle("Box Plot of Engine Size") +
  theme(plot.margin = unit(c(1, 1, 1, 1) * scale_factor, "cm"))
```




```r
ggplot(boxplot_data, aes(x = factor(1), y = Horsepower)) +
```

```
  geom_boxplot() +
  labs(x = "", y = "Horsepower") +
  ggtitle("Box Plot of Horsepower") +
  theme(plot.margin = unit(c(1, 1, 1, 1) * scale_factor, "cm"))
```

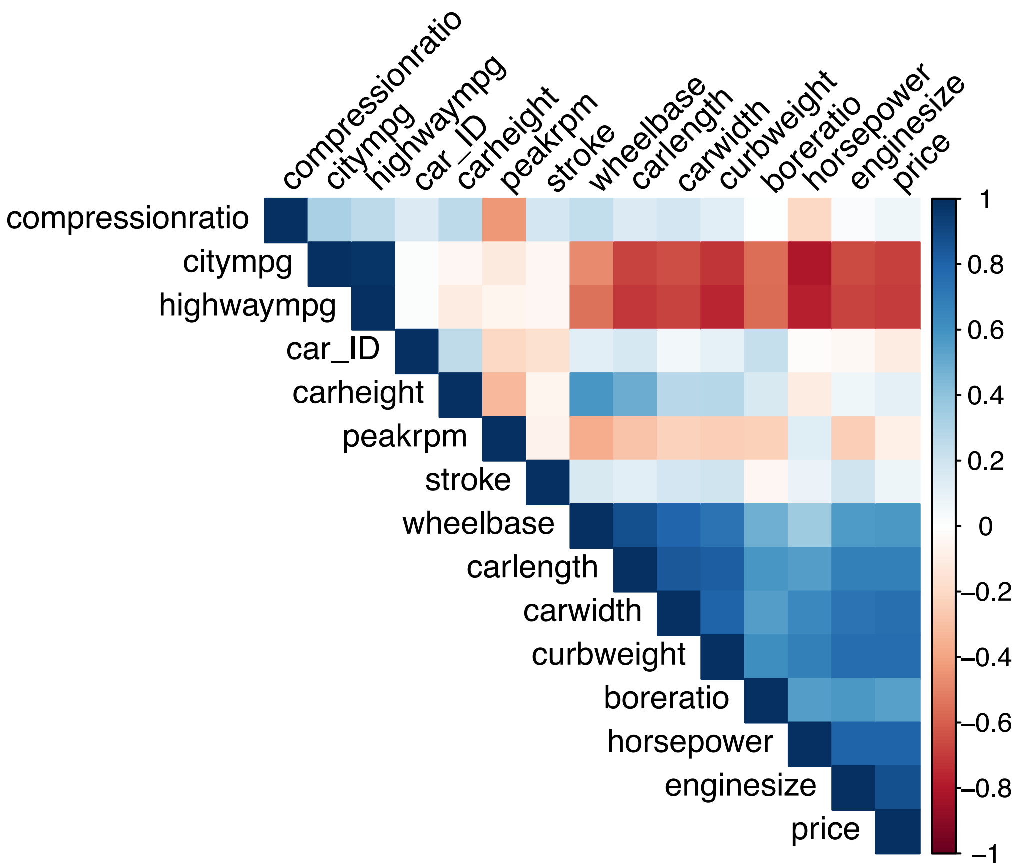


in above diagrams, discrete dots, represent outliers, and if the upper tail is bigger than lowewr tail, then we have a left skewed distribution and vice versa. Heavy line in the box, represents median and box boundary, ranges from $IQR1$ to $IQR3$

Making correlation map

Correlation is a **measure of the linear relationship between variables**, and it is not applicable to categorical variables, including dummy variables. Dummy variables represent categorical information in a binary format and do not convey the same information as numeric variables. Therefore, correlation analysis is not meaningful for categorical variables, even after they are replaced with dummies.

```
# Select only numeric columns
numeric_cols <- carprice %>% select_if(is.numeric)

# Calculate correlation matrix
cor_matrix <- cor(numeric_cols)

# Create correlation map
corrplot(cor_matrix, method = "color", type = "upper", order = "hclust",
         tl.col = "black", tl.srt = 45)
```



Identifying effective and ineffective factors

```
# Define a threshold for determining effective and ineffective factors
threshold <- 0.5

# Find effective factors on price
effective_factors <- colnames(cor_matrix)[cor_matrix[, "price"] >= threshold]

# Find ineffective factors on price
ineffective_factors <- colnames(cor_matrix)[cor_matrix[, "price"] < threshold]

# Print the effective and ineffective factors on price
cat("Effective Factors on Price:\n")
```

```
## Effective Factors on Price:
```

```
cat(effective_factors, sep = ", ")
```

```
## wheelbase, carlength, carwidth, curbweight, enginesize, boreratio, horsepower, price
```

```
cat("\n\nIneffective Factors on Price:\n")
```

```
##
##
## Ineffective Factors on Price:
```

```
cat(ineffective_factors, sep = ", ")
```

```
## car_ID, carheight, stroke, compressionratio, peakrpm, citympg, highwaympg
```

Hypothesis testing

The null hypothesis (H_0) in each case is that *there is no correlation between the two variables*, while the alternative hypothesis (H_1) is that *there is a correlation.*

```r
# Hypothesis test for correlation between price and enginesize
test1 <- cor.test(carprice$price, carprice$enginesize)
print(test1)
```

```
##
##  Pearson's product-moment correlation
##
## data:  carprice$price and carprice$enginesize
## t = 25.645, df = 203, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.8374234 0.9030097
## sample estimates:
##       cor
## 0.8741448
```

Since the p-value is extremely small (less than the significance level of 0.05), we *reject the null hypothesis* (H_0) of no correlation between the variables. The results indicate that there is a significant correlation between the `carprice$price` and `carprice$enginesize` variables. The correlation coefficient estimate of 0.8741448 suggests a **strong positive correlation** between the two variables.

```r
# Hypothesis test for correlation between price and horsepower
test2 <- cor.test(carprice$price, carprice$horsepower)
print(test2)
```

```
##
##  Pearson's product-moment correlation
##
## data:  carprice$price and carprice$horsepower
## t = 19.549, df = 203, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.7546795 0.8509381
## sample estimates:
##       cor
## 0.8081388
```

Since the p-value is extremely small (less than the significance level of 0.05), we *reject the null hypothesis* (H_0) of no correlation between the variables. The results indicate that there is a significant correlation between the `carprice$price` and `carprice$horsepower` variables. The correlation coefficient estimate of 0.8081388 suggests a **strong positive correlation** between the two variables.

```r
# Hypothesis test for correlation between price and carlength
test3 <- cor.test(carprice$price, carprice$carlength)
print(test3)
```

```
##
##  Pearson's product-moment correlation
##
## data:  carprice$price and carprice$carlength
## t = 13.32, df = 203, p-value < 2.2e-16
```

```
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6022455 0.7497870
## sample estimates:
##     cor
## 0.68292
```

Since the p-value is extremely small (less than the significance level of 0.05), we *reject the null hypothesis* (H_0) of no correlation between the variables. The results indicate that there is a significant correlation between the `carprice$price` and `carprice$carlength` variables. The correlation coefficient estimate of 0.68292 suggests a **moderate positive correlation** between the two variables.

```r
# Hypothesis test for correlation between price and carwidth
test4 <- cor.test(carprice$price, carprice$carwidth)
print(test4)
```

```
##
##  Pearson's product-moment correlation
##
## data:  carprice$price and carprice$carwidth
## t = 16.626, df = 203, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6945624 0.8118807
## sample estimates:
##       cor
## 0.7593253
```

Since the p-value is extremely small (less than the significance level of 0.05), we *reject the null hypothesis* (H_0) of no correlation between the variables. The results indicate that there is a significant correlation between the `carprice$price` and `carprice$carwidth` variables. The correlation coefficient estimate of 0.7593253 suggests a **strong positive correlation** between the two variables.

Creating dummy variables to convert the categorical variables to numerical.

let's use `model.matrix()`

```r
# Summary of the 'symboling' column
summary(carprice$symboling)
```

```
## -2 -1  0  1  2  3
##  3 22 67 54 32 27
```

```r
# Create dummy variables for 'symboling'
dummy <- data.frame(model.matrix(~ symboling, data = carprice))
# Remove the first column from the dummy dataframe
dummy <- dummy[, -1]
# Add the dummy variables to the carprice dataframe
carprice <- cbind(carprice, dummy)

# Summary of the 'fueltype' column
summary(carprice$fueltype)
```

```
## diesel    gas
##     20    185
```

```r
# Create dummy variables for 'fueltype'
dummy <- data.frame(model.matrix(~ fueltype, data = carprice))
```

```r
 # Remove the first column from the dummy dataframe
dummy <- dummy[, -1]
# Add the dummy variables to the carprice dataframe
carprice <- cbind(carprice, dummy)
# Rename the column corresponding to 'fueltype'
colnames(carprice)[33] <- "fueltype"

summary(carprice$aspiration)
```

```
##   std turbo
##   168    37
```

```r
dummy <- data.frame(model.matrix(~ aspiration, data = carprice))
dummy <- dummy[, -1]
carprice <- cbind(carprice, dummy)
colnames(carprice)[34] <- "aspiration"

summary(carprice$doornumber)
```

```
## four  two
##  115   90
```

```r
dummy <- data.frame(model.matrix(~ doornumber, data = carprice))
dummy <- dummy[, -1]
carprice <- cbind(carprice, dummy)
colnames(carprice)[35] <- "doornumber"

summary(carprice$carbody)
```

```
##              convertible     hardtop   hatchback       sedan       wagon
##           0            6           5          68         104          22
```

```r
dummy <- data.frame(model.matrix(~ carbody, data = carprice))
dummy <- dummy[, -1]
carprice <- cbind(carprice, dummy)

summary(carprice$drivewheel)
```

```
## 4wd fwd rwd
##   9 120  76
```

```r
dummy <- data.frame(model.matrix(~ drivewheel, data = carprice))
dummy <- dummy[, -1]
carprice <- cbind(carprice, dummy)

summary(carprice$enginelocation)
```

```
## front  rear
##   202     3
```

```r
dummy <- data.frame(model.matrix(~ enginelocation, data = carprice))
dummy <- dummy[, -1]
carprice <- cbind(carprice, dummy)
colnames(carprice)[42] <- "enginelocation"

summary(carprice$enginetype)
```

```
## dohc dohcv     l   ohc  ohcf  ohcv rotor
##    12     1    12   148    15    13     4
```

```
dummy <- data.frame(model.matrix(~ enginetype, data = carprice))
dummy <- dummy[, -1]
carprice <- cbind(carprice, dummy)

summary(carprice$cylindernumber)
```

```
##         eight   five   four    six  three twelve    two
##      0      5      9    164     21      1      1      4
```

```
dummy <- data.frame(model.matrix(~ cylindernumber, data = carprice))
dummy <- dummy[, -1]
carprice <- cbind(carprice, dummy)

summary(carprice$fuelsystem)
```

```
## 1bbl 2bbl 4bbl  idi  mfi mpfi spdi spfi
##   11   66    3   20    1   94    9    1
```

```
dummy <- data.frame(model.matrix(~ fuelsystem, data = carprice))
dummy <- dummy[, -1]
carprice <- cbind(carprice, dummy)

summary(carprice$carcompany)
```

```
## alfa-romero        audi         bmw       buick   chevrolet       dodge
##           3           7           8           8           3           9
##       honda       isuzu      jaguar       mazda     mercury  mitsubishi
##          13           4           3          17           1          13
##      nissan     peugeot    plymouth    porcshce     renault        saab
##          18          11           7           5           2           6
##      subaru      toyota  volkswagen       volvo
##          12          32          12          11
```

```
dummy <- data.frame(model.matrix(~ carcompany, data = carprice))
dummy <- dummy[, -1]
carprice <- cbind(carprice, dummy)
```

Preparing the dataset for modeling by removing unneccesary data and only keeping dummy variables

```
carprice<-carprice[,-1:-25]
carprice<-carprice[,-2]
```

Dividing into training and test data set

We will divide data in a ratio of 70 : 30. If we use higher ratio for training, it would lead to better results but because we don't have a lot of data for prediction, we prefer to use 70 : 30

```
set.seed(100) # to make the same random numbers each time.

# Randomly generating row indices for train dataset
trainindices = sample(1:nrow(carprice), 0.7*nrow(carprice))
```

Generating the train data set

```
train = carprice[trainindices,]
train_final <- train # keeping a copy for later use
```

Similarly storing the rest of the observations into an object `test`:

```
test = carprice[-trainindices,]
test_final <- test # keeping a copy for later use
```

Phase 2: Data processing with multiple regression model

Executing the first model `model_1` in the training set

```
model_1<-lm(price~.,data=train)
summary_data <- summary(model_1)
residuals <- residuals(model_1)

RSS <- sum(residuals^2)
TSS <- sum((train$price - mean(train$price))^2)
MSE <- sum(residuals^2) / length(residuals)
r_squared <- summary_data$r.squared
adjusted_r_squared <- summary_data$adj.r.squared
# Print the values
print(paste("RSS:", RSS))
```

```
## [1] "RSS: 575313514.606226"
```

```
print(paste("TSS:", TSS))
```

```
## [1] "TSS: 9969378040.93547"
```

```
print(paste("MSE:", MSE))
```

```
## [1] "MSE: 4023171.43081277"
```

```
print(paste("R-squared:", r_squared))
```

```
## [1] "R-squared: 0.942291935139392"
```

```
print(paste("Adjusted R-squared:", adjusted_r_squared))
```

```
## [1] "Adjusted R-squared: 0.910928856410801"
```

predict for test data using model 1

```
# Predict the test data using the trained model
predict_test <- predict(model_1, newdata = test)

# Calculate residuals
residuals <- test$price - predict_test

# Calculate RSS
RSS <- sum(residuals^2)

# Calculate TSS
TSS <- sum((test$price - mean(train$price))^2)
```

```r
# Calculate MSE
MSE <- mean(residuals^2)

# Calculate R-squared
r_squared <- 1 - (RSS / TSS)

# Calculate Adjusted R-squared
n <- nrow(test)
p <- length(coef(model_1)) - 1  # Number of predictors
adjusted_r_squared <- 1 - (RSS / (n - p - 1)) / (TSS / (n - 1))

# Print the values
print(paste("RSS:", RSS))
```

```
## [1] "RSS: 684285652.205707"
```

```r
print(paste("TSS:", TSS))
```

```
## [1] "TSS: 3103393440.05477"
```

```r
print(paste("MSE:", MSE))
```

```
## [1] "MSE: 11036865.3581566"
```

```r
print(paste("R-squared:", r_squared))
```

```
## [1] "R-squared: 0.779504060499132"
```

```r
print(paste("Adjusted R-squared:", adjusted_r_squared))
```

```
## [1] "Adjusted R-squared: -2.36256307738823"
```

- R-Squared:

  measures the proportion of the total variation in the target variable (`price`) that is **explained** by the linear regression model.

  It ranges between 0 and 1, where 0 indicates that the model explains none of the variation and 1 indicates that the model explains all the variation.

  R-squared is used to evaluate *how well the model fits the data* and provides an indication of the model's predictive power.

- Adjusted R_Square:

  a modified version of R-squared that takes into account the **number of predictors (variables)** in the model and adjusts for the *degrees of freedom*.

  it is useful when *comparing models with different numbers of predictors*, as it accounts for model complexity.

- RSS:

  represents the sum of the squared differences between the *observed values (actual target variable values) and the predicted values* from the linear regression model.

  It measures the **unexplained variance** or the error of the model.

  A *lower RSS indicates a better fit* of the model to the data.

- TSS: represents the sum of the squared differences between the *observed values (actual target variable values) and the mean of the target variable*.

It quantifies the **total variation** in the target variable.

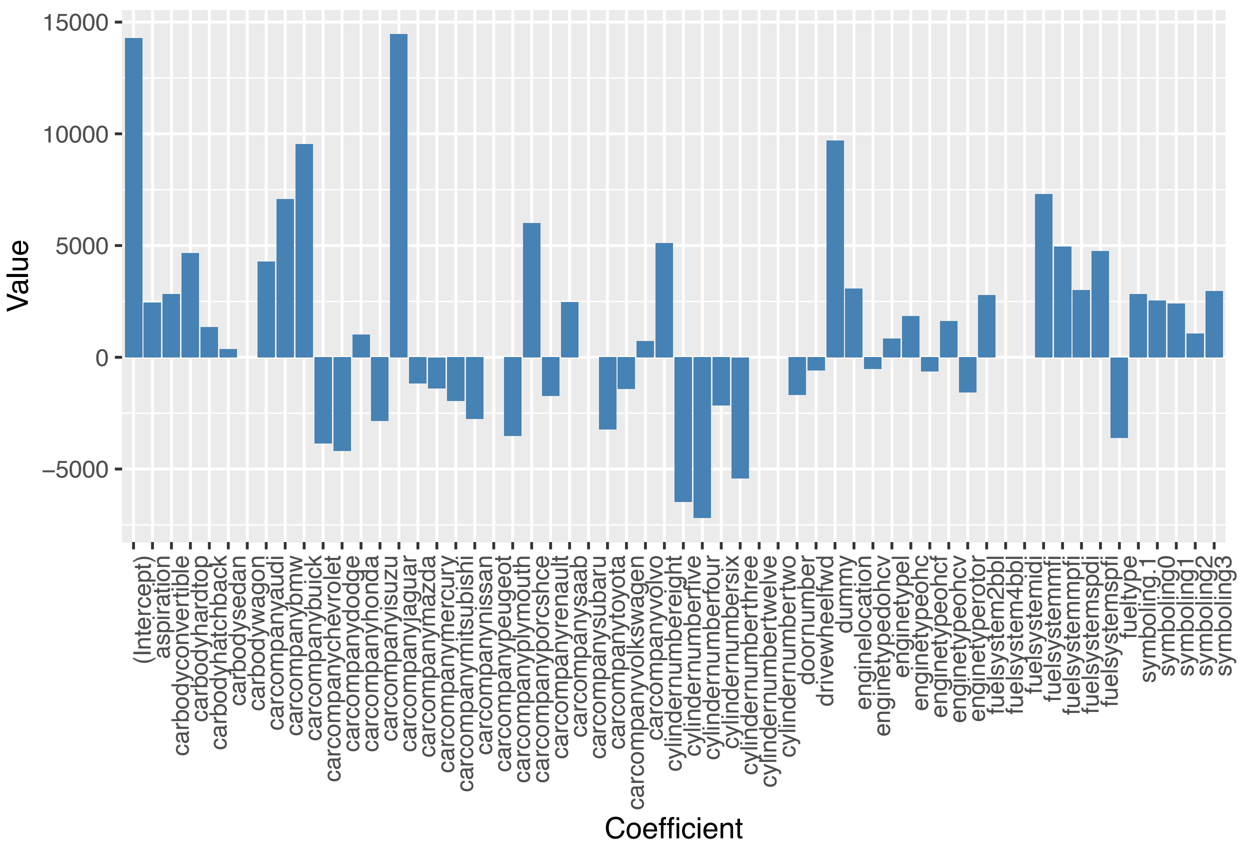
TSS is used to calculate the *proportion of the variation explained by the model* (R-squared).

- MSE:

  It is a measure of the **average squared deviation between the predicted and actual values**.

  MSE is commonly used to assess the *accuracy of the model's predictions*.

```
coef_data <- data.frame(Coefficient = names(coef(model_1)), Value = coef(model_1))
ggplot(coef_data, aes(x = Coefficient, y = Value)) +
  geom_col(fill = "steelblue") +
  labs(x = "Coefficient", y = "Value") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



a higher coefficient does not necessarily mean it is more important. The importance of a coefficient depends on the *scale of the corresponding predictor variable*. When the data scales are similar, it becomes easier to compare the coefficients directly. Additionally, it is crucial to evaluate the **statistical significance of the coefficients**, which can be determined using the *t-values* or *p-values* associated with each coefficient.

- Prediction Efficiency:
  - RSS on the test data is $684,285,652.21$, which indicates the model has some prediction errors.
  - A lower ratio of RSS/TSS indicates a better prediction. the ratio is around 0.22, suggesting that the model explains approximately 78% of the variability in the test data.
  - A lower MSE indicates better prediction accuracy. the MSE is $11,036,865.36$, which means, on average, the predicted values deviate by approximately $11,036,865.36$ from the actual values.
- Interpretation Efficiency:

  - A higher R-squared value indicates a better fit. the R-squared on the test data is 0.94, indicating that approximately 94% of the variability in the test data is explained by the model.
  - A higher adjusted R-squared indicates a better balance between model complexity and fit. the adjusted R-squared on the test data is 0.91, which is interpretable.

To improve the model:

- Feature Selection: Evaluate the relevance and importance of the predictor variables in your model. Remove any irrelevant or redundant variables that might be contributing noise to the model.
- Model Complexity: Assess if the model is overly complex for the available data. Simplify the model if necessary to avoid overfitting and improve generalization to new data.
- Data Quality and Quantity: Evaluate the quality and quantity of the available data. More data, especially if it includes a diverse range of observations, can potentially improve the model's performance. Additionally, ensure the data is clean, free from outliers, and properly preprocessed.

Pase 3: Feature selection and analysis

feature selection basaed on p-value (0.05 significance level)

In a stepwise algorithm, we remove the factor with highest p-value and check the effect.

```
initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing carbodysedan due to insignificance

train <- train[, -which(names(train) == "carbodysedan")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing cylindernumberthree due to insignificance

train <- train[, -which(names(train) == "cylindernumberthree")]
#initial_model <- lm(price ~ ., data = train)
# summary(initial_model) # removing carcompanypeugeot due to insignificance

train <- train[, -which(names(train) == "carcompanypeugeot")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing cylindernumberfive due to insignificance

train <- train[, -which(names(train) == "cylindernumberfive")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing enginetypeohcf due to insignificance

train <- train[, -which(names(train) == "enginetypeohcf")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing carcompanysubaru due to insignificance

train <- train[, -which(names(train) == "carcompanysubaru")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing enginetypedohcv due to insignificance

train <- train[, -which(names(train) == "enginetypedohcv")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing carcompanymercury due to insignificance

train <- train[, -which(names(train) == "carcompanymercury")]
#initial_model <- lm(price ~ ., data = train)
```

```r
#summary(initial_model) # removing carcompanymitsubishi due to insignificance

train <- train[, -which(names(train) == "carcompanymitsubishi")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing carcompanymazda due to insignificance

train <- train[, -which(names(train) == "carcompanymazda")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing carcompanyvolkswagen due to insignificance

train <- train[, -which(names(train) == "carcompanyvolkswagen")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing carcompanyrenault due to insignificance

train <- train[, -which(names(train) == "carcompanyrenault")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing symboling2 due to insignificance

train <- train[, -which(names(train) == "symboling2")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing carbodywagon due to insignificance

train <- train[, -which(names(train) == "carbodywagon")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing drivewheelfwd due to insignificance

train <- train[, -which(names(train) == "drivewheelfwd")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing enginetypeohcv due to insignificance

train <- train[, -which(names(train) == "enginetypeohcv")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing carcompanyisuzu due to insignificance

train <- train[, -which(names(train) == "carcompanyisuzu")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing carcompanynissan due to insignificance

train <- train[, -which(names(train) == "carcompanynissan")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing enginetypel due to insignificance

train <- train[, -which(names(train) == "enginetypel")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing carcompanychevrolet due to insignificance

train <- train[, -which(names(train) == "carcompanychevrolet")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing cylindernumberfour due to insignificance

train <- train[, -which(names(train) == "cylindernumberfour")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing enginetypeohc due to insignificance
```

```r
train <- train[, -which(names(train) == "enginetypeohc")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing carcompanyplymouth due to insignificance

train <- train[, -which(names(train) == "carcompanyplymouth")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing fuelsystemspfi due to insignificance

train <- train[, -which(names(train) == "fuelsystemspfi")]
initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing fuelsystemspdi due to insignificance

train <- train[, -which(names(train) == "fuelsystemspdi")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing fuelsystem2bbl due to insignificance

train <- train[, -which(names(train) == "fuelsystem2bbl")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing enginetyperotor due to insignificance

train <- train[, -which(names(train) == "enginetyperotor")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing cylindernumbertwo due to insignificance

train <- train[, -which(names(train) == "cylindernumbertwo")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing fuelsystem4bbl due to insignificance

train <- train[, -which(names(train) == "fuelsystem4bbl")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing carcompanyhonda due to insignificance

train <- train[, -which(names(train) == "carcompanyhonda")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing carbodyconvertible due to insignificance

train <- train[, -which(names(train) == "carbodyconvertible")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing fuelsystemmfi due to insignificance

train <- train[, -which(names(train) == "fuelsystemmfi")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing carcompanydodge due to insignificance

train <- train[, -which(names(train) == "carcompanydodge")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing carbodyhatchback due to insignificance

train <- train[, -which(names(train) == "carbodyhatchback")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing doornumber due to insignificance

train <- train[, -which(names(train) == "doornumber")]
```

```r
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing symboling1 due to insignificance

train <- train[, -which(names(train) == "symboling1")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing fueltype due to insignificance

train <- train[, -which(names(train) == "fueltype")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing fuelsystemidi due to insignificance

train <- train[, -which(names(train) == "fuelsystemidi")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing carcompanyvolvo due to insignificance

train <- train[, -which(names(train) == "carcompanyvolvo")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing cylindernumbertwelve due to insignificance


train <- train[, -which(names(train) == "cylindernumbertwelve")]
#initial_model <- lm(price ~ ., data = train)
#summary(initial_model) # removing carcompanysaab due to insignificance

train <- train[, -which(names(train) == "carcompanysaab")]
initial_model <- lm(price ~ ., data = train)
summary(initial_model)
```

```
##
## Call:
## lm(formula = price ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6601.2 -1068.6    25.9  1064.4 10305.2
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)             7103.1      385.8  18.410  < 2e-16 ***
## symboling.1             2845.9      833.4   3.415 0.000859 ***
## symboling0              1763.3      505.4   3.489 0.000668 ***
## symboling3              2360.5      761.4   3.100 0.002387 **
## aspiration              2967.1      617.2   4.807 4.28e-06 ***
## carbodyhardtop          3441.5     1650.2   2.085 0.039046 *
## enginelocation          4011.9      595.7   6.735 5.22e-10 ***
## dummy                   8305.4     2737.6   3.034 0.002934 **
## cylindernumbereight    11931.1     1702.3   7.009 1.29e-10 ***
## cylindernumbersix       3048.6      825.5   3.693 0.000329 ***
## fuelsystemmpfi          2327.9      553.6   4.205 4.91e-05 ***
## carcompanyaudi          6800.8     1234.5   5.509 1.95e-07 ***
## carcompanybmw           8320.1     1260.0   6.603 1.01e-09 ***
## carcompanybuick        10899.8     1483.2   7.349 2.21e-11 ***
## carcompanyjaguar       17361.5     1602.2  10.836  < 2e-16 ***
## carcompanyporcshce      6120.6     1989.0   3.077 0.002563 **
```

```
## carcompanytoyota     -2011.2      612.0  -3.287 0.001314 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2469 on 126 degrees of freedom
## Multiple R-squared:  0.9229, Adjusted R-squared:  0.9131
## F-statistic:  94.3 on 16 and 126 DF,  p-value: < 2.2e-16
```

```
# Predicting the car prices in the testing dataset

predict_1 <- predict(initial_model,test[,-1])
test$test_price <- predict_1

# Accuracy of the predictions

# Calculating correlation
r <- cor(test$price,test$test_price)

# Calculating R squared by squaring correlation
rsquared <- cor(test$price,test$test_price)^2

# Checking R-squared
rsquared
```

```
## [1] 0.769138
```

we find accuracy of 76.91% for the final model with all selected features being significant.

F-statistics feature selection

now it's time to select top 10 features based on F-Statistics:

```
anova_table <- anova(initial_model)
sorted_table <- anova_table[order(anova_table$`F value`, decreasing = TRUE), ]
top_features <- rownames(sorted_table)[1:10]
top_features
```

```
##  [1] "enginelocation"      "cylindernumbereight" "carbodyhardtop"
##  [4] "carcompanyjaguar"    "symboling3"          "symboling.1"
##  [7] "carcompanybuick"     "cylindernumbersix"   "dummy"
## [10] "carcompanyaudi"
```

Why I chose this method?

The initial model might have contained features that were not significantly related to the target variable, leading to an insignificant overall model. However, by removing the least significant features iteratively, the resulting model has improved in terms of significance.

Our method quantifies the relative quality of different models based on the maximum likelihood estimation. We aim to find a model with good fit to the data while avoiding overfitting.

finding synergy pairs

Let's consider an interaction factor in our linear model which is the miltiplication of pairs. If the interaction term has an adjusted R square more than $-\infty$ then we will introduce it as a synergy pair.

The adjusted R-squared is a statistical measure that indicates the *proportion of variance in the dependent variable* (in this case, `price`) that is *explained by the independent variables* (features) in the model.

By setting the condition that the interaction term's adjusted R-squared should be greater than $-\infty$, we are essentially ensuring that any synergy pair identified **must provide some improvement over the baseline model** (without the interaction term). Since the adjusted R-squared ranges from negative infinity to 1, setting a condition greater than $-\infty$ implies that we want the interaction term to contribute positively to the model's explanatory power.

```r
# Initialize an empty list to store the synergy pairs
synergy_pairs <- list()

for (i in 1:length(top_features)) {
  # Extract the top feature
  top_feature <- top_features[i]

  # Initialize variables to store the best synergy feature and its adjusted R-squared value
  best_synergy_feature <- ""
  best_adjusted_r_squared <- -Inf

  # Iterate through all the features in the dataset (excluding the dependent variable)
  for (feature in colnames(train)[colnames(train) != "price"]) {
    # Skip the top feature if it's the current feature being tested
    if (feature == top_feature) {
      next
    }

    # Create an interaction term
    train$interaction <- train[[top_feature]] * train[[feature]]
    test$interaction <- test[[top_feature]] * test[[feature]]

    # Fit the linear regression model with the interaction term
    model_formula <- as.formula(paste("price ~ . + interaction"))
    model <- lm(model_formula, data = train)

    # Calculate the adjusted R-squared value
    adjusted_r_squared <- summary(model)$adj.r.squared

    # Check if the current feature has a higher adjusted R-squared value than the best one
    if (adjusted_r_squared > best_adjusted_r_squared) {
      best_adjusted_r_squared <- adjusted_r_squared
      best_synergy_feature <- feature
    }
  }

  # Add the synergy pair to the list
  synergy_pairs[[top_feature]] <- best_synergy_feature
}

# Print the synergy pairs
synergy_pairs
```

```
## $enginelocation
## [1] "carcompanytoyota"
##
## $cylindernumbereight
## [1] "carbodyhardtop"
##
```

```
## $carbodyhardtop
## [1] "cylindernumbereight"
##
## $carcompanyjaguar
## [1] "cylindernumbersix"
##
## $symboling3
## [1] "cylindernumbereight"
##
## $symboling.1
## [1] "cylindernumbereight"
##
## $carcompanybuick
## [1] "carbodyhardtop"
##
## $cylindernumbersix
## [1] "carcompanybmw"
##
## $dummy
## [1] "symboling.1"
##
## $carcompanyaudi
## [1] "symboling0"
```

some of the features have appeared several times which will result into multicollinearity. So we won't consider them. we may consider adding the features have appeared once but obviously they have already appeared in the model. to prevent insignificance or multicollunearity, we must decide basaed on p-values. (if there were any added features)

To fit a better model, we use Variance Inflation Factor (VIF)[1] to improve the `initial_model` and remove the multicollinearity.

```r
vif_model <- initial_model

#alias(vif_model) # remove interaction due to complete multicollinearity

train <- train[, -which(names(train) == "interaction")]
vif_model <- lm(price ~ ., data = train)
```

all vif values is less than 2.5 which is a reasonable threshold and also all factors are significanct. so we stop here. (`carcompanybuick` has a vif of 2.7 but if we remove it, our r-square will decrease. So we leave it)

```r
# Predicting the car prices in the testing dataset

predict_1 <- predict(vif_model,test[,-1])
test$test_price <- predict_1

# Accuracy of the predictions

# Calculating correlation
r <- cor(test$price,test$test_price)

# Calculating R squared by squaring correlation
rsquared <- cor(test$price,test$test_price)^2
```

[1] a measure used to assess multicollinearity by quantifying how much the variance of the estimated regression coefficient is inflated due to correlation with other independent variables.

```r
# Checking R-squared
rsquared # 0.769138
```

```
## [1] 0.769138
```

Phase 4: (Extra) Fitting a Decision Tree model

A decision tree is a tree-like flowchart structure that helps make decisions by mapping possible inputs to predicted outputs based on a sequence of logical conditions or rules.

My decision tree is splitting the data based on variables and values that **minimize the sum of squared errors (SSE)** in each split.

```r
# Recursive function to build the decision tree.
build_tree <- function(data, depth, max_depth, used_variables = c()) {
  # Create a node
  node <- list()
  node$leaf <- FALSE

  # Check if max depth is reached or all target values are the same
  if (depth >= max_depth || length(unique(data$price)) == 1) {
    node$leaf <- TRUE
    node$prediction <- mean(data$price)
    node$used_variables <- used_variables
    return(node)
  }

  # Find the best splitting variable and value
  best_sse <- Inf
  best_variable <- NULL
  best_value <- NULL

  for (variable in names(data)) {
    if (variable != "price" && !(variable %in% used_variables)) {
      unique_values <- sort(unique(data[[variable]]))
      for (value in unique_values) {
        left_data <- data[data[[variable]] <= value, ]
        right_data <- data[data[[variable]] > value, ]

        if (nrow(left_data) > 0 && nrow(right_data) > 0) {
          left_sse <- sum((left_data$price - mean(left_data$price))^2)
          right_sse <- sum((right_data$price - mean(right_data$price))^2)
          total_sse <- left_sse + right_sse

          if (total_sse < best_sse) {
            best_sse <- total_sse
            best_variable <- variable
            best_value <- value
          }
        }
      }
    }
  }
```

```r
  # Check if no best split was found
  if (is.null(best_variable) || is.null(best_value)) {
    node$leaf <- TRUE
    node$prediction <- mean(data$price)
    node$used_variables <- used_variables
    return(node)
  }

  # Create left and right subtrees
  left_data <- data[data[[best_variable]] <= best_value, ]
  right_data <- data[data[[best_variable]] > best_value, ]

  node$split_variable <- best_variable
  node$split_value <- best_value
  node$left <- build_tree(left_data, depth + 1,
                          max_depth, c(used_variables, best_variable))
  node$right <- build_tree(right_data, depth + 1,
                          max_depth, c(used_variables, best_variable))

  return(node)
}

# Build the decision tree
max_depth <- 3
tree <- build_tree(train_final, depth = 0, max_depth = max_depth)

# Function to extract the factors used in the tree
extract_factors <- function(tree) {
  factors <- character()

  if (!tree$leaf) {
    factors <- c(factors, tree$split_variable)
    factors <- c(factors, extract_factors(tree$left))
    factors <- c(factors, extract_factors(tree$right))
  }

  factors <- unique(factors)
  return(factors)
}

# Extract the factors used in the tree
used_factors <- extract_factors(tree)
used_factors
```

```
## [1] "cylindernumberfour" "enginelocation"     "symboling2"
## [4] "carcompanytoyota"   "fuelsystemmpfi"
```

```r
# Make predictions
# Prediction function for a decision tree
predict_tree <- function(tree, data) {
  predictions <- numeric(nrow(data))

  for (i in seq_len(nrow(data))) {
    node <- tree
```

```r
    while (!node$leaf) {
      if (data[[node$split_variable]][i] <= node$split_value) {
        node <- node$left
      } else {
        node <- node$right
      }
    }
    predictions[i] <- node$prediction
  }

  return(predictions)
}


predictions <- predict_tree(tree, test_final)

# Evaluate the model
mse <- mean((predictions - test_final$price)^2)
rmse <- sqrt(mse)

# Calculate R-squared value
ss_total <- sum((test_final$price - mean(test_final$price))^2)
ss_residual <- sum((test_final$price - predictions)^2)
r_squared <- 1 - (ss_residual / ss_total)

# Calculate adjusted R-squared value
n <- nrow(test_final)
p <- length(attributes) - 1  # Number of predictors excluding the intercept
adjusted_r_squared <- 1 - (1 - r_squared) * ((n - 1) / (n - p - 1))

print(paste("Adjusted R-squared:", adjusted_r_squared))
```

```
## [1] "Adjusted R-squared: 0.0857733606662846"
```

```r
print(paste("Root Mean Squared Error (RMSE):", rmse))
```

```
## [1] "Root Mean Squared Error (RMSE): 6570.45529684101"
```