title: “Project 2: Used Cars Model Building”

author: “Safal Thapa”

date: “2024-12-15”

output:

pdf_document: default

word_document: default

html_document: default

```
suppressMessages(library(data.table))
suppressMessages(library(stats))
suppressMessages(library(readr))
suppressMessages(library(car))
suppressMessages(library(lmtest))
suppressMessages(library(caret))
suppressMessages(library(caTools))
suppressMessages(library(corrplot))
suppressMessages(library(ggplot2))
suppressMessages(library(scales))
suppressMessages(library(tidyverse))
suppressMessages(library(gridExtra))
suppressMessages(library(lubridate))
```

PART 1: EDA AND DATA PREPARATION

Step 1: Reading and Understanding the Data

- Importing the data
- Understanding the structure of the data (identify the numerical and categorical variables)

```
set.seed(123)
library(data.table)
cars_data_original <-
  fread("/Users/shuffle/Desktop/DSE 1030/Project II/used_cars_data.csv",
        header = TRUE)

cars_data <- cars_data_original
```

First Step: Exploratory Data Analysis

- Is there a relationship between price to to Seller Rating?
- Relationship between mileage to Seller Rating?
- Relationship between City_fuel_economy or Highway_fuel_economy to Price?
- Relationship between Body_type and Price?
- Relationship between Age and Price?

Understanding the structure

```
set.seed(123)
# head(cars_data)

 # glimpse(cars_data)

colSums(is.na(cars_data))

##                     vin             back_legroom                      bed
##                       0                        0                        0
##              bed_height               bed_length                body_type
##                       0                        0                        0
##                   cabin                     city        city_fuel_economy
##                       0                        0                   491285
##   combine_fuel_economy            daysonmarket               dealer_zip
##                 3000040                        0                        0
##             description         engine_cylinders      engine_displacement
##                       0                        0                   172386
##             engine_type           exterior_color                    fleet
##                       0                       17                  1426595
##           frame_damaged         franchise_dealer           franchise_make
##                 1426595                        0                        0
##           front_legroom         fuel_tank_volume                fuel_type
##                       0                        0                        0
##           has_accidents                   height     highway_fuel_economy
##                 1426595                        0                   491285
##              horsepower           interior_color                    isCab
##                  172386                        0                  1426595
##            is_certified                   is_cpo                   is_new
##                 3000040                  2817142                        0
##              is_oemcpo                 latitude                   length
##                 2864678                        0                        0
##             listed_date            listing_color               listing_id
##                       0                        0                        0
##               longitude         main_picture_url            major_options
##                       0                        0                        0
##               make_name          maximum_seating                  mileage
##                       0                        0                   144387
##              model_name              owner_count                    power
##                       0                  1517013                        0
##                   price                  salvage           savings_amount
##                       0                  1426595                        0
##           seller_rating                    sp_id                  sp_name
##                   40872                       96                        0
##            theft_title                   torque             transmission
##                 1426595                        0                        0
##    transmission_display                   trimId                trim_name
##                       0                        0                        0
## vehicle_damage_category             wheel_system     wheel_system_display
##                 3000040                        0                        0
```

```
##                 wheelbase                        width                          year
##                         0                            0                             0
```

Analysing Data set:

After seeing the glimpse of the data set, I thought few columns could be turned into numerical values for further exploration by deleting string characters in their data. Chosen Variables are:front_legroom (in), fuel_tank_volume (gal), height (in), length (in), width (in), wheelbase (in).

Solution: Create a function to clean the string-based above columns to numeric columns.

Range Constraints:

Now let work on Range constraints.. I want to check if the seller_rating is our of range. If it is than we need to clean the data.

```
sum(is.na(cars_data$seller_rating))

## [1] 40872

sum(cars_data$seller_rating == "NA", na.rm = TRUE)

## [1] 0

max(cars_data$seller_rating, na.rm = TRUE)

## [1] 5

min(cars_data$seller_rating, na.rm = TRUE)

## [1] 1
```

Analysing seller_rating

The values in seller_rating are not out of range.It does have 40872 null values which needs addressing if it is used in the model later.

Since we have a huge dataset, I think it would be ideal to delete them rather than trying to fill it with values.

```
cars_data <- cars_data %>%
  filter(!is.na(seller_rating) & seller_rating != "NA")
```

No null values in seller's rating now.

```
nrow(cars_data)

## [1] 2959168
```

Out of Range listed

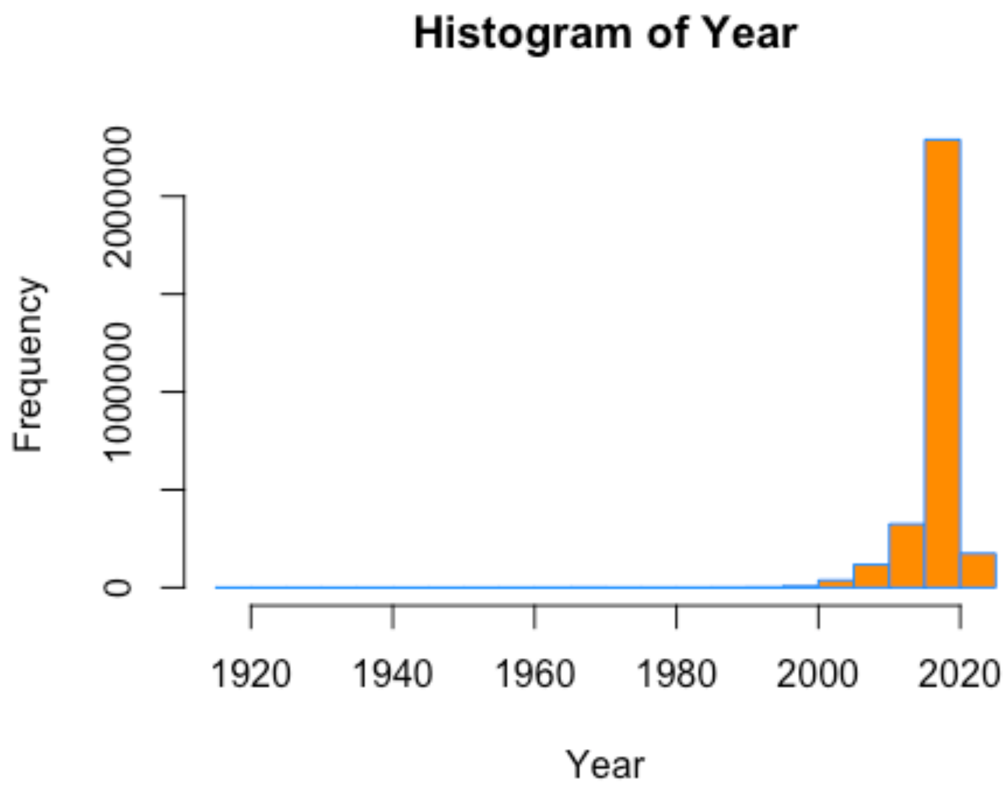
Now lets make sure listed_date is not out of range.

```r
cars_data %>% filter(listed_date > today())

## Empty data.table (0 rows and 66 cols):
vin,back_legroom,bed,bed_height,bed_length,body_type...

numeric_vars <- sapply(cars_data, is.numeric)
cat_vars <- sapply(cars_data, is.factor)

# Printing variable types
print("Numerical Variables:")

## [1] "Numerical Variables:"

print(names(cars_data)[numeric_vars])

##  [1] "city_fuel_economy"    "daysonmarket"          "engine_displacement"
##  [4] "highway_fuel_economy" "horsepower"            "latitude"
##  [7] "listing_id"           "longitude"             "mileage"
## [10] "owner_count"          "price"                 "savings_amount"
## [13] "seller_rating"        "sp_id"                 "year"

print("Categorical Variables:")

## [1] "Categorical Variables:"

print(names(cars_data)[cat_vars])

## character(0)

summary(cars_data$price)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     165   18500   26500   30003   38309 3299995
```

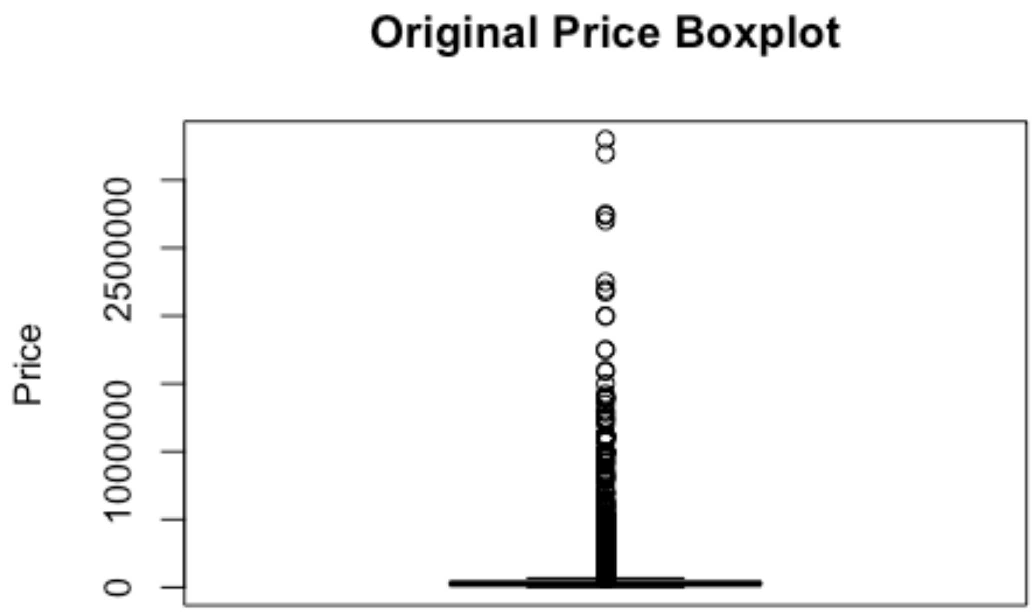
Analysing Price Values:

We also need to take care of the outliers in the prices. Some of the prices are way off. We can use a function to clear the outliers in the prices column.

```r
hist(cars_data$year,
xlab = "Year",
main = "Histogram of Year",
col = "darkorange",
border = "dodgerblue",
breaks = 20)
```



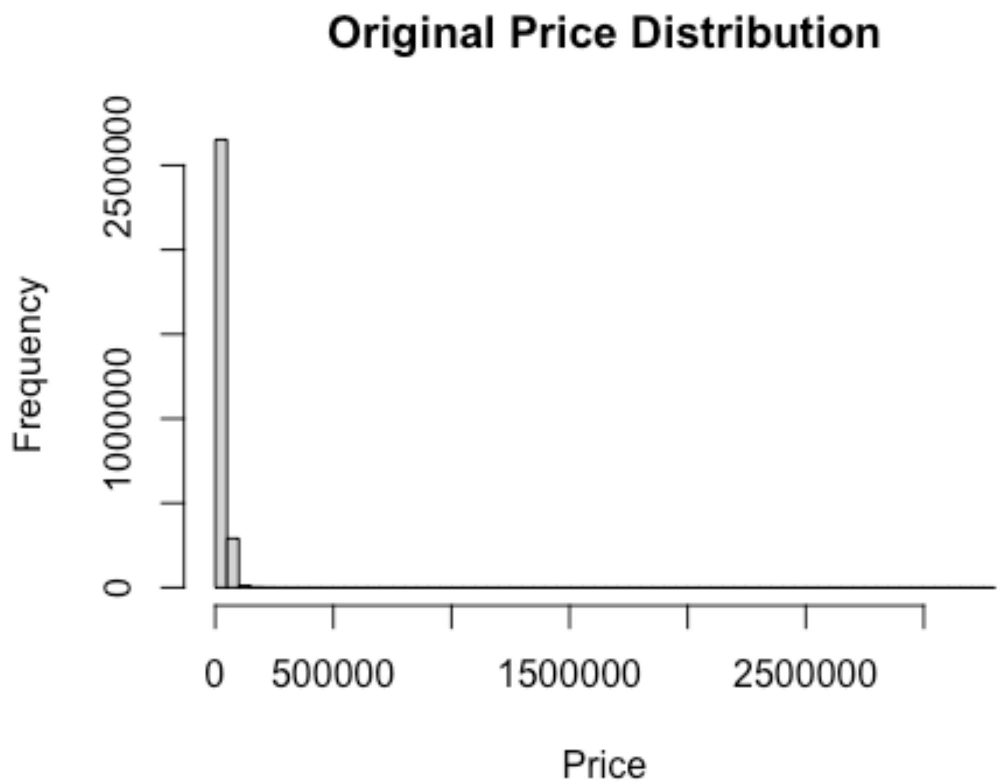

```r
boxplot(cars_data$price,
        main="Original Price Boxplot",
        ylab="Price")
```



Price

Outliers:
Too many outliers; price goes beyond 2.5 million. These are extreme prices. We need to select only those cars between certain price ranges for the model building.

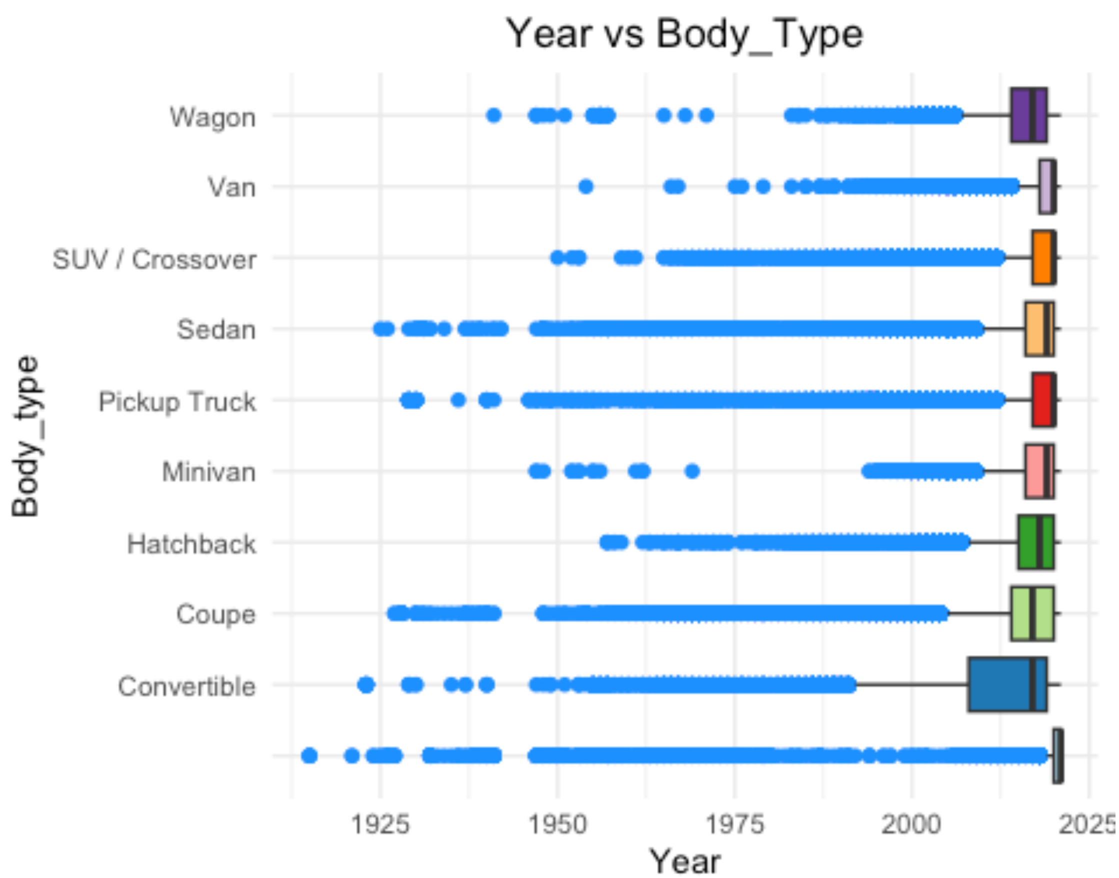
```
hist(cars_data$price,
     breaks=50,
     main="Original Price Distribution",
     xlab="Price")
```



In the above graph, it is clear the values of car price are skewed to the left.

```
cars_data %>%
  group_by(body_type) %>%
  summarize(
    mean_price = mean(price),
    count = n()
  ) %>%
  arrange(desc(count))

## # A tibble: 10 × 3
##    body_type          mean_price   count
##    <chr>                   <dbl>   <int>
##  1 "SUV / Crossover"      30776. 1398664
##  2 "Sedan"                22495.  730781
##  3 "Pickup Truck"         40644.  467932
##  4 "Hatchback"            17450.   87085
##  5 "Minivan"              24885.   78517
##  6 "Coupe"                40855.   70827
##  7 "Van"                  29884.   46566
##  8 "Wagon"                20401.   39811
##  9 "Convertible"          46350.   25673
## 10 ""                     39164.   13312
```

```r
# Boxplot for year vs body_type
ggplot(cars_data, aes(x = year, y = body_type, fill = body_type)) +
  geom_boxplot(outlier.color = "dodgerblue", outlier.shape = 16, outlier.size
= 2) +
  labs(
    title = "Year vs Body_Type",
    y = "Body_type",
    x = "Year"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5),
    legend.position = "none"
  ) +
  scale_fill_brewer(palette = "Paired")
```

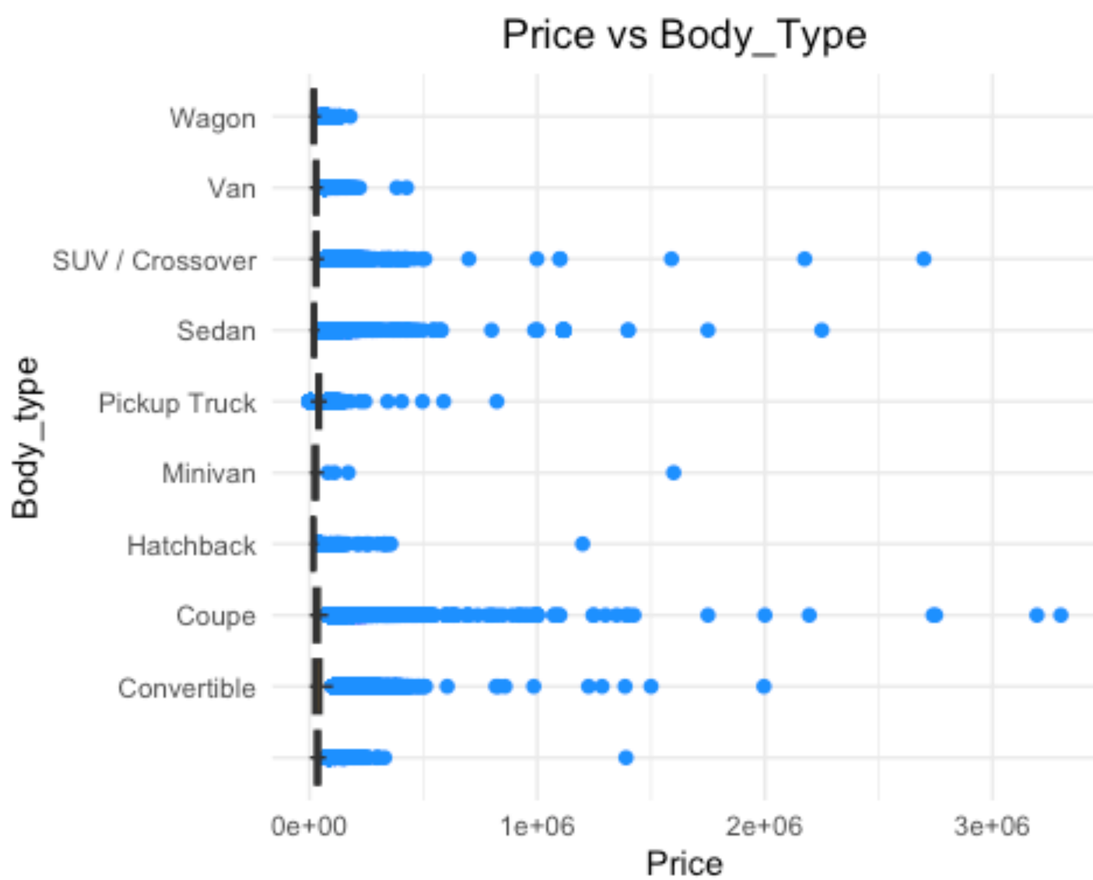


###

Dealing with outliers:
There are so many outliers. We need select cars between certain time period before building the model.

```r
# Boxplot for year vs body_type
ggplot(cars_data, aes(x = price, y = body_type, fill = body_type)) +
  geom_boxplot(outlier.color = "dodgerblue", outlier.shape = 16, outlier.size
= 2) +
```

```r
  labs(
    title = "Price vs Body_Type",
    y = "Body_type",
    x = "Price"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5),
    legend.position = "none"
  )
```



###

More data cleaning..
There are a lot of null values in body_type. Lets delete it as it can skew analysis and visualization.

```r
cars_data <- cars_data %>%
  filter(!is.na(body_type) & body_type != "")
```

Combined_fuel_economy column is empty:

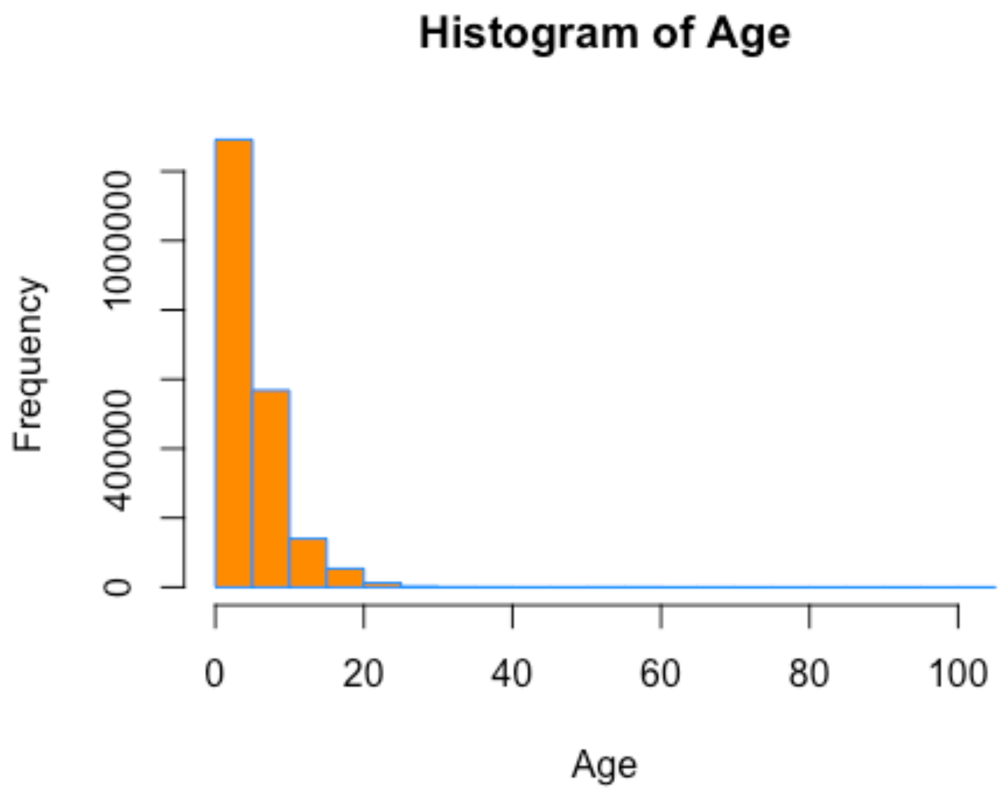
We can fill this column by utilizing average of city_fuel_economy and highway_fuel_economy.

New Car Age column:

There is no age column in the dataset. It might be beneficial to derive it from the make_year column.

```
# Function to clean the columns that are string-based to numeric columns
string_numeric <- function(data, column) {
  numeric_values <- as.numeric(gsub("[^0-9.]", "", data[[column]]))
  data[[paste0(column, "_numeric")]] <- numeric_values
  data[[column]] <- NULL
  return(data)
}

# Function to handle the outliers in our data set using IQR method
remove_outliers <- function(data, column){
  Q1 <- quantile(data[[column]], 0.25)
  Q3 <- quantile(data[[column]], 0.75)
  IQR <- Q3 - Q1

  lower_bound <- Q1 - 1.5 * IQR
  upper_bound <- Q3 + 1.5 * IQR

  data %>%
    filter(
      .data[[column]] >= lower_bound,
      .data[[column]] <= upper_bound
    )
}

# Main data preparation function
prepare_car_data <- function(file, seed = 123) {
  set.seed(seed)

  # 2. Clean String-based Numeric Columns
  columns_to_clean <- c(
    "maximum_seating", "front_legroom", "fuel_tank_volume",
    "height", "length", "width", "wheelbase"
  )

  for (col in columns_to_clean) {
    cars_data <- string_numeric(cars_data, col)
  }

  # 5. Outlier Handling
  cars_data <- remove_outliers(cars_data, "price")

  # 7. Feature Engineering
  cars_data <- cars_data %>%
    # Calculate car age
```

```r
    mutate(
      age = 2024 - year,
      # Combine fuel economy
      combine_fuel_economy = (city_fuel_economy + highway_fuel_economy) / 2
    )

  # 8. Subset to Top 10 Makes
  top_10_makes <- c('Ford', 'Chevrolet', 'Toyota', 'Nissan', 'Honda',
                    'Jeep', 'Hyundai', 'Kia', 'RAM', 'GMC')
  cars_data_subset <- cars_data %>%
    filter(make_name %in% top_10_makes)

  return(cars_data_subset)
}

set.seed(123)
df_cars_data <- prepare_car_data(cars_data)

hist(df_cars_data$age,
xlab = "Age",
main = "Histogram of Age",
col = "darkorange",
border = "dodgerblue",
breaks = 20)
```



Age

Histogram
The age histogram in not normally distributed. Let's check with less than 5 years old cars and see how the result comes.

```
library(dplyr)
df_cars_data %>%
    summarise(across(everything(), ~sum(is.na(.))))

##   vin back_legroom bed bed_height bed_length body_type cabin city
## 1   0            0   0          0          0         0     0    0
##   city_fuel_economy combine_fuel_economy daysonmarket dealer_zip
description
## 1            335711               335711            0          0
0
##   engine_cylinders engine_displacement engine_type exterior_color   fleet
## 1                0              108266           0             16 1020763
##   frame_damaged franchise_dealer franchise_make fuel_type has_accidents
## 1       1020763                0              0         0       1020763
##   highway_fuel_economy horsepower interior_color   isCab is_certified
is_cpo
## 1               335711     108266              0 1020763      2071440
1967460
##   is_new is_oemcpo latitude listed_date listing_color listing_id longitude
## 1      0   1967460        0           0             0          0         0
```

```
##   main_picture_url major_options make_name mileage model_name owner_count
power
## 1                0             0         0  102059          0     1082666
0
##   price salvage savings_amount seller_rating sp_id sp_name theft_title
torque
## 1     0 1020763              0             0     0       0     1020763
0
##   transmission transmission_display trimId trim_name
vehicle_damage_category
## 1            0                    0      0         0
2071440
##   wheel_system wheel_system_display year maximum_seating_numeric
## 1            0                    0    0                   94679
##   front_legroom_numeric fuel_tank_volume_numeric height_numeric
length_numeric
## 1                 94819                    95067          94644
94636
##   width_numeric wheelbase_numeric age
## 1         94655             94619   0
```

Column with null values:

combined_fuel_economy, horsepower, mileage, maximum_seating_numeric, front_legroom_numeric, fuel_tank_volume_numeric, height_numeric, length_numeric, width_numeric, wheelbase_numeric, mileage.

```
# Replacing mileage with mean values for cars from 2015 and newer
df_cars_data <- df_cars_data %>%
  group_by(year) %>%
  mutate(mileage = ifelse(year >= 2015,
                         mean(mileage),
                         mileage)) %>%
  ungroup()

# Setting mileage to 0 for new cars with missing mileage
df_cars_data <- df_cars_data %>%
  mutate(mileage = if_else(is.na(mileage) & is_new == TRUE,
                          0,
                          mileage))

# Removing any remaining rows where mileage is NA
df_cars_data <- df_cars_data %>%
  filter(!is.na(mileage))

df_cars_data <- df_cars_data %>%
  filter(!is.na(maximum_seating_numeric),
         !is.na(front_legroom_numeric),
         !is.na(fuel_tank_volume_numeric),
         !is.na(height_numeric),
```

```
         !is.na(length_numeric),
         !is.na(width_numeric),
         !is.na(wheelbase_numeric),
         !is.na(combine_fuel_economy),
         !is.na(horsepower)
         )

df_cars_data %>%
  select(where(is.numeric)) %>%
  summary()

##  city_fuel_economy combine_fuel_economy  daysonmarket
engine_displacement
##  Min.   : 9.00     Min.   : 9.50        Min.   :   0.00   Min.   :1000
##  1st Qu.:18.00     1st Qu.:21.00        1st Qu.:  18.00   1st Qu.:2000
##  Median :22.00     Median :25.50        Median :  43.00   Median :2500
##  Mean   :23.05     Mean   :26.26        Mean   :  90.14   Mean   :2830
##  3rd Qu.:27.00     3rd Qu.:30.00        3rd Qu.: 124.00   3rd Qu.:3500
##  Max.   :58.00     Max.   :58.00        Max.   :2688.00   Max.   :8100
##
##  highway_fuel_economy   horsepower      latitude       listing_id
##  Min.   :10.00        Min.   : 55    Min.   :21.30   Min.   : 58515071
##  1st Qu.:24.00        1st Qu.:170    1st Qu.:33.39   1st Qu.:271699248
##  Median :29.00        Median :203    Median :37.68   Median :277807690
##  Mean   :29.47        Mean   :237    Mean   :36.91   Mean   :274418431
##  3rd Qu.:34.00        3rd Qu.:295    3rd Qu.:41.03   3rd Qu.:280154232
##  Max.   :61.00        Max.   :662    Max.   :61.16   Max.   :282019143
##
##    longitude          mileage         owner_count         price
##  Min.   :-157.93   Min.   :      0   Min.   : 1.0     Min.   :  350
##  1st Qu.: -96.89   1st Qu.:      0   1st Qu.: 1.0     1st Qu.:20413
##  Median : -87.28   Median :      0   Median : 2.0     Median :27206
##  Mean   : -90.46   Mean   : 20988   Mean   : 2.2     Mean   :29148
##  3rd Qu.: -80.88   3rd Qu.:      0   3rd Qu.: 3.0     3rd Qu.:38440
##  Max.   : -67.23   Max.   :397322   Max.   :15.0     Max.   :67840
##                                      NA's   :874519
##  savings_amount  seller_rating       sp_id             year
##  Min.   :    0   Min.   :1.000   Min.   : 42627   Min.   :1988
##  1st Qu.:    0   1st Qu.:4.000   1st Qu.: 59024   1st Qu.:2020
##  Median :    0   Median :4.309   Median :277941   Median :2020
##  Mean   :  150   Mean   :4.236   Mean   :216166   Mean   :2018
##  3rd Qu.:    0   3rd Qu.:4.571   3rd Qu.:327033   3rd Qu.:2020
##  Max.   :25226   Max.   :5.000   Max.   :440591   Max.   :2021
##
##  maximum_seating_numeric front_legroom_numeric fuel_tank_volume_numeric
##  Min.   : 2.000          Min.   :38.00         Min.   : 7.00
##  1st Qu.: 5.000          1st Qu.:41.10         1st Qu.:14.00
##  Median : 5.000          Median :42.10         Median :16.40
##  Mean   : 5.445          Mean   :42.25         Mean   :17.98
##  3rd Qu.: 6.000          3rd Qu.:43.10         3rd Qu.:21.50
```

```
##  Max.   :15.000          Max.   :54.40         Max.   :42.00
##
##  height_numeric   length_numeric  width_numeric   wheelbase_numeric
##  Min.   : 46.00   Min.   :143.1   Min.   :57.00   Min.   : 86.6
##  1st Qu.: 58.50   1st Qu.:180.9   1st Qu.:72.40   1st Qu.:106.3
##  Median : 66.50   Median :189.8   Median :74.90   Median :111.2
##  Mean   : 66.35   Mean   :193.3   Mean   :77.91   Mean   :115.4
##  3rd Qu.: 70.70   3rd Qu.:201.8   3rd Qu.:82.10   3rd Qu.:119.1
##  Max.   :109.40   Max.   :263.9   Max.   :98.60   Max.   :164.6
##
##       age
##  Min.   : 3.000
##  1st Qu.: 4.000
##  Median : 4.000
##  Mean   : 5.625
##  3rd Qu.: 4.000
##  Max.   :36.000
##
```

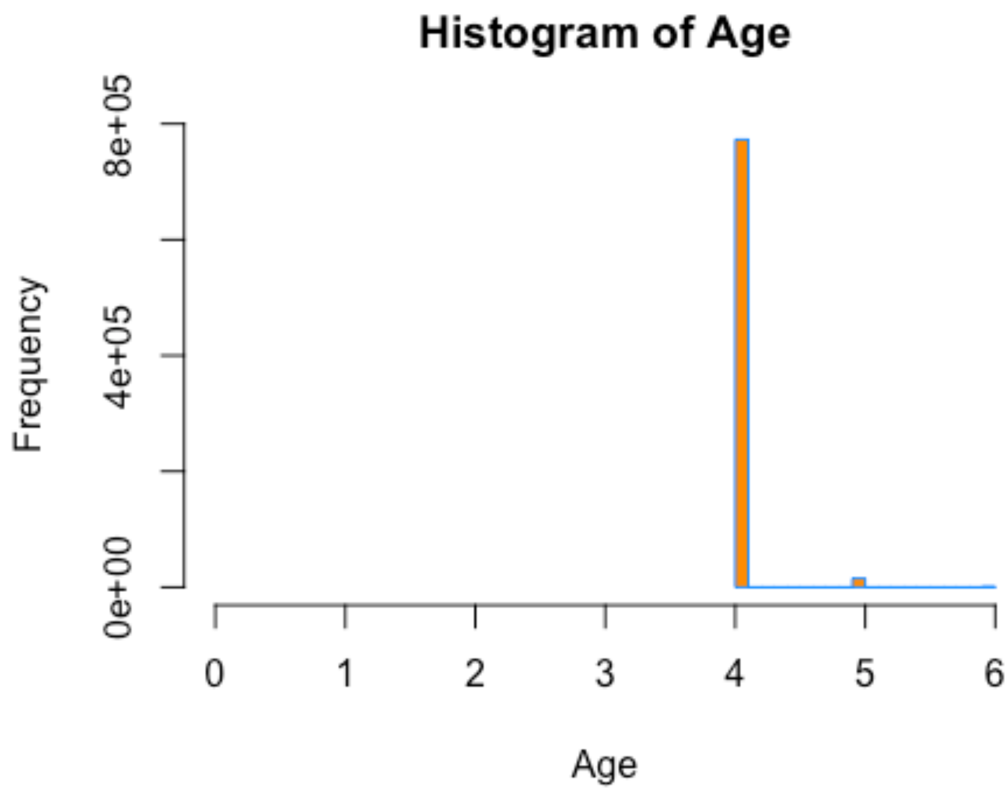
Data Filtering:

Based on the above observations and identifying the outliers present in our predicted variable; price, I decided to filter the dataset by removing the outliers based on the below criteria.

For Price: Selecting those price which is greater than or equal to 1,000 and less than or equal to 60,000. This removes any outliers in price entries.

For year: We also narrowed the dataset by selecting from a 6-year period between 2015 to 2020. There were outliers outside these years.

```
df_cars_age <- df_cars_data %>%
  filter(year > 2015, year <= 2020)

hist(df_cars_age$age,
xlab = "Age",
main = "Histogram of Age",
col = "darkorange",
border = "dodgerblue",
#breaks = c(1, 2, 3, 4, 5),
xlim = c(0, 6))
```



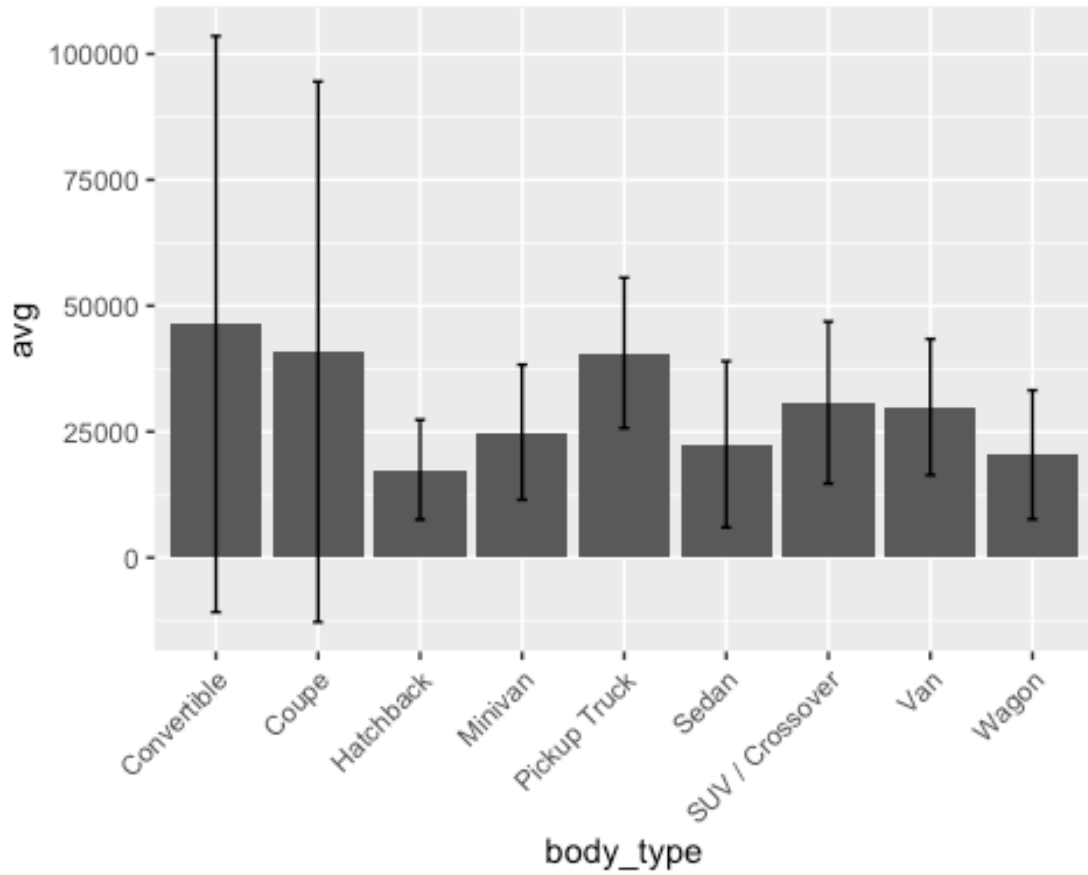
###

Above Histogram conclusion
We can conclude that age column is not great for our model.

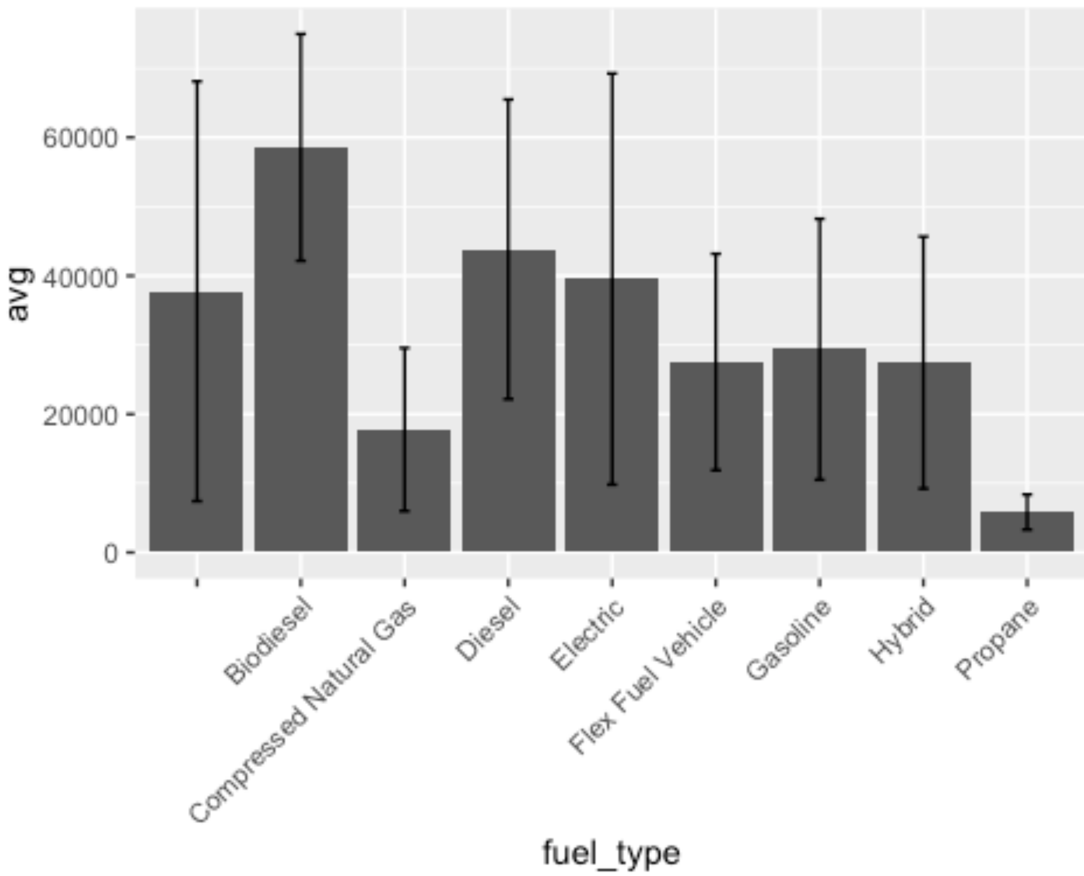
```
df_cars_data <- df_cars_data %>%
  filter(price >= 1000, price <= 60000)

boxplot(df_cars_data$price,
        main="Original Price Boxplot",
        ylab="Price")
```


```
# Now lets sure listed_date is not out of range
df_cars_data %>% filter(listed_date > today())

## # A tibble: 0 × 67
## # ℹ 67 variables: vin <chr>, back_legroom <chr>, bed <chr>, bed_height
<chr>,
## #   bed_length <chr>, body_type <chr>, cabin <chr>, city <chr>,
## #   city_fuel_economy <dbl>, combine_fuel_economy <dbl>, daysonmarket
<int>,
## #   dealer_zip <chr>, description <chr>, engine_cylinders <chr>,
## #   engine_displacement <dbl>, engine_type <chr>, exterior_color <chr>,
## #   fleet <lgl>, frame_damaged <lgl>, franchise_dealer <lgl>,
## #   franchise_make <chr>, fuel_type <chr>, has_accidents <lgl>, …

nrow(df_cars_data)

## [1] 1044672

cars_body_desc <- cars_data %>%
  select(body_type, price) %>%
  pivot_longer(!body_type, names_to = "key",
               values_to = "value") %>%
  group_by(body_type) %>%
  summarise(avg = mean(value),
```

```r
            stdev = sd(value))

ggplot(cars_body_desc, aes(x=body_type, y = avg)) +
  geom_col() +
  geom_errorbar(aes(ymin = avg - stdev,
                    ymax = avg + stdev),
                width = 0.1) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```r
cars_fuel_desc <- cars_data %>%
  select(fuel_type, price) %>%
  pivot_longer(!fuel_type, names_to = "key",
               values_to = "value") %>%
  group_by(fuel_type) %>%
  summarise(avg = mean(value),
            stdev = sd(value))

ggplot(cars_fuel_desc, aes(x=fuel_type, y = avg)) +
  geom_col() +
  geom_errorbar(aes(ymin = avg - stdev,
                    ymax = avg + stdev),
                width = 0.1)+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
df_cars_data %>%
  group_by(body_type) %>%
  summarize(
    mean_price = mean(price),
    count = n()
  ) %>%
  arrange(desc(count))

## # A tibble: 9 × 3
##   body_type        mean_price  count
##   <chr>                 <dbl>  <int>
## 1 SUV / Crossover      28506. 519328
## 2 Sedan                20699. 238898
## 3 Pickup Truck         40077. 197633
## 4 Hatchback            17830.  28756
## 5 Minivan              29622.  23247
## 6 Coupe                24508.  17273
## 7 Van                  23769.  12493
## 8 Convertible          27596.   3787
## 9 Wagon                 8178.   3257
```

Body Type Summary

From above can we predict SUV/Crossover is the preffered body type?
Does convertible has highest average price than hatchback?

```
# Summary statistics by make (top 10)
make_summary <- df_cars_data %>%
  group_by(make_name) %>%
  summarise(
    mean_price = mean(price),
    median_price = median(price),
    count = n(),
    mean_mileage = mean(mileage, na.rm = TRUE),
    mean_combine_fuel_economy = mean(combine_fuel_economy, na.rm = TRUE)
  ) %>%
  arrange(desc(count))

make_summary

## # A tibble: 10 × 6
##    make_name mean_price median_price  count mean_mileage
mean_combine_fuel_eco…¹
##    <chr>          <dbl>        <dbl>  <int>        <dbl>
<dbl>
##  1 Ford          32113.       31320. 233768       19771.
24.0
##  2 Chevrolet     28046.       26023  169537       21385.
24.4
##  3 Honda         26424.       26387  142996       23105.
29.8
##  4 Toyota        27022.       26594. 121474       30285.
31.2
##  5 Nissan        23140.       22508  107780       19251.
29.0
##  6 Jeep          31420.       29882.  83310       19345.
22.5
##  7 Hyundai       22997.       24070   63413       16851.
29.2
##  8 Kia           22352.       22340.  48066       17414.
29.0
##  9 GMC           32227.       32612   37283       31065.
22.3
## 10 RAM           41942.       43428   37045       10218.
19.0
## # ℹ abbreviated name: ¹mean_combine_fuel_economy
```

Car Make Summary

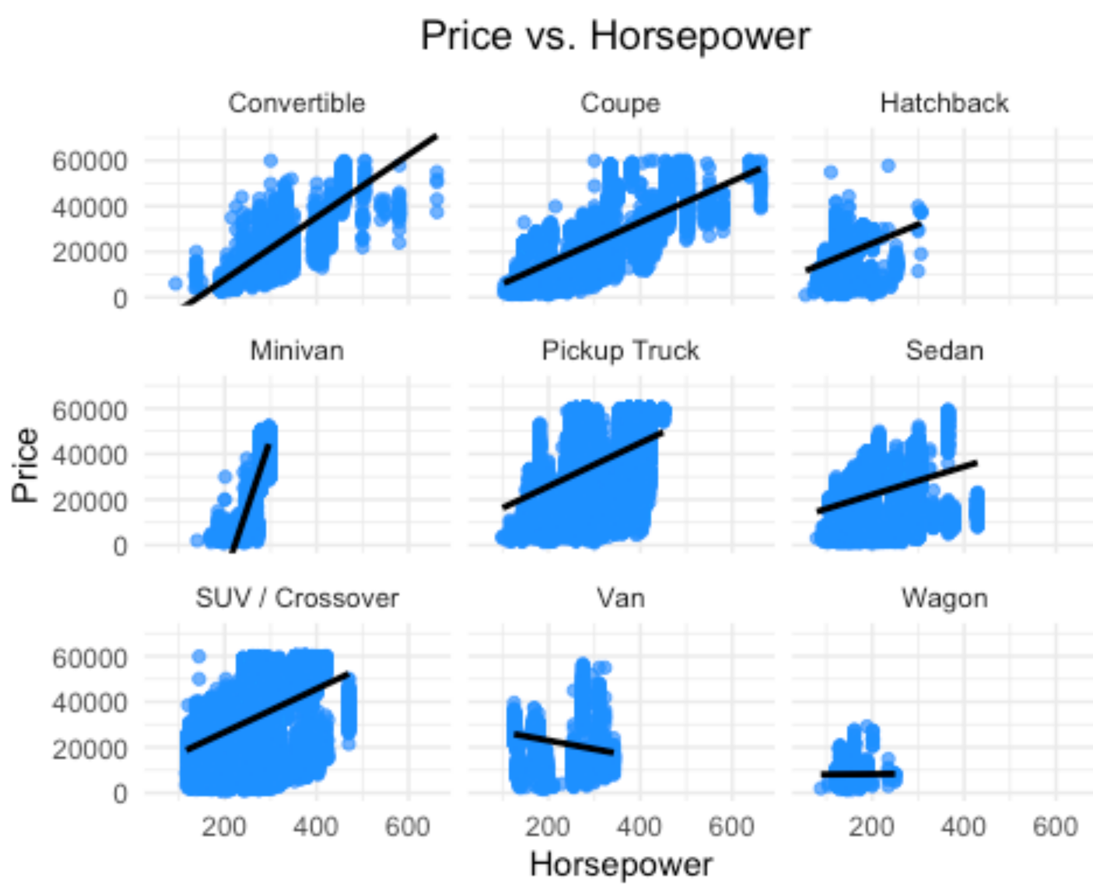
From above, we can conclude that most selling companies are:
Ford, Chevrolet, Toyota, Nissan, Honda, Jeep, Hyundai, Kia, RMC and GMC

```r
set.seed(123)
cars_data_subset <- df_cars_data
top_10_makes <- c('Ford', 'Chevrolet', 'Toyota', 'Nissan', 'Honda',
                  'Jeep', 'Hyundai', 'Kia', 'RAM', 'GMC')
df_cars_data <- cars_data_subset[cars_data_subset$make_name %in%
                                       top_10_makes,
  ]

# glimpse(df_cars_data)

find_missing_columns <- function(df, missing_threshold = 0.4) {
  cols_to_drop <- c()
  for (c in names(df)) {
    count_nulls <- sum(is.na(df[[c]]))
    null_rate <- count_nulls / nrow(df)
    if (null_rate > missing_threshold) {
      cols_to_drop <- c(cols_to_drop, c)
    }
  }
  return(cols_to_drop)
}

cols_to_drop <- find_missing_columns(df_cars_data)
cols_to_drop

##  [1] "fleet"                   "frame_damaged"
##  [3] "has_accidents"           "isCab"
##  [5] "is_certified"            "is_cpo"
##  [7] "is_oemcpo"               "owner_count"
##  [9] "salvage"                 "theft_title"
## [11] "vehicle_damage_category"

df_cars_data <- df_cars_data[, !names(df_cars_data) %in% cols_to_drop]

numerical_cols <- names(df_cars_data[sapply(df_cars_data, is.numeric)])
pros.cor <- cor(df_cars_data[numerical_cols])
pros.cor <- round(pros.cor,3)

price_correlations <- pros.cor[,"price"]
price_correlations <- sort(abs(price_correlations), decreasing = TRUE)
price_correlations

##                   price              horsepower                 mileage
##                   1.000                   0.637                   0.615
##                    year                     age       wheelbase_numeric
##                   0.609                   0.609                   0.552
##          length_numeric          height_numeric           width_numeric
##                   0.541                   0.534                   0.487
## fuel_tank_volume_numeric     engine_displacement          savings_amount
##                   0.453                   0.406                   0.373
##     highway_fuel_economy     combine_fuel_economy maximum_seating_numeric
##                   0.352                   0.311                   0.302
```

```
##        city_fuel_economy                       sp_id    front_legroom_numeric
##                    0.260                       0.192                    0.109
##            seller_rating                daysonmarket                longitude
##                    0.014                       0.008                    0.006
##               listing_id                    latitude
##                    0.005                       0.001
```

```r
ggplot(df_cars_data, aes(x = horsepower, y = price)) +
  geom_point(color = "dodgerblue", alpha = 0.7) +
  geom_smooth(method = "lm", color = "black", se = F) +
  labs(title = "Price vs. Horsepower ",
       x = "Horsepower",
       y = "Price") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5)) +
  facet_wrap(~body_type) +
  coord_cartesian(ylim = c(0, NA))
```

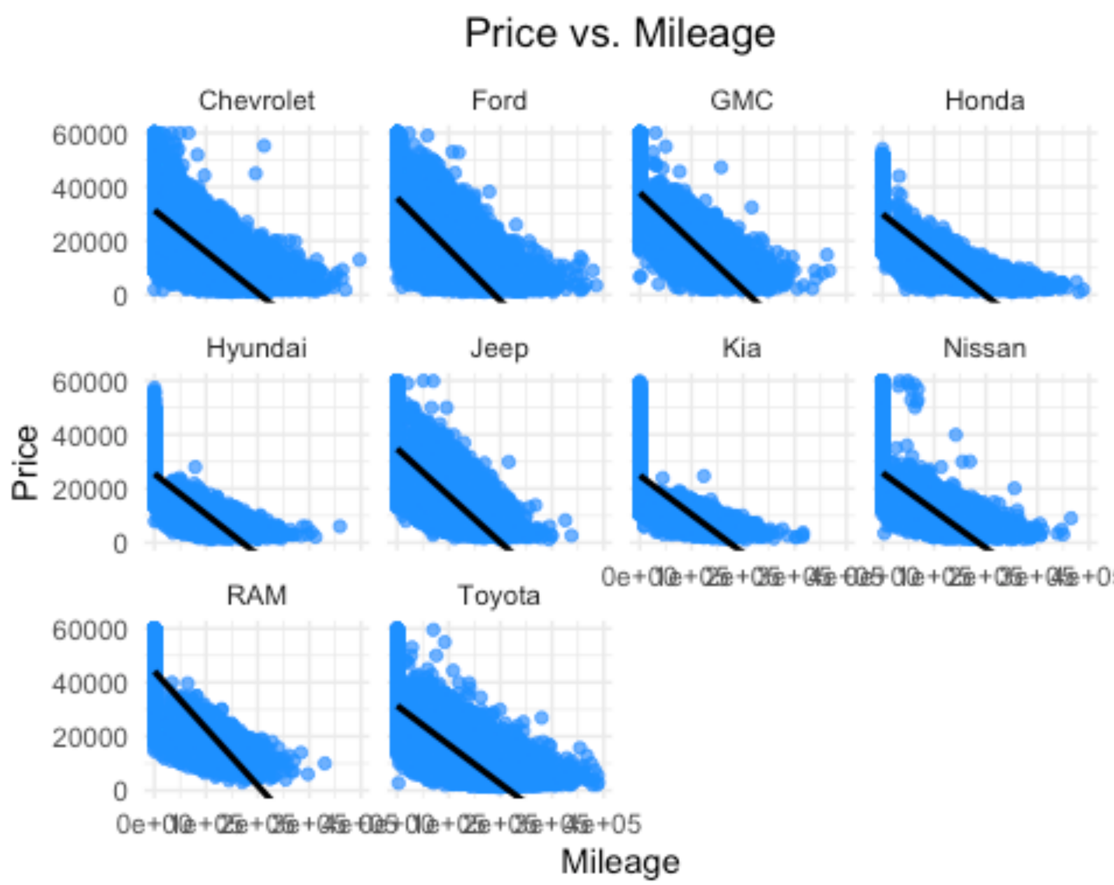
```
## `geom_smooth()` using formula = 'y ~ x'
```



Price vs Horsepower by Body_Type

Almost all body_types have positive correlation between price and horsepower in a car. Only van shows negative correlation which may be odd.

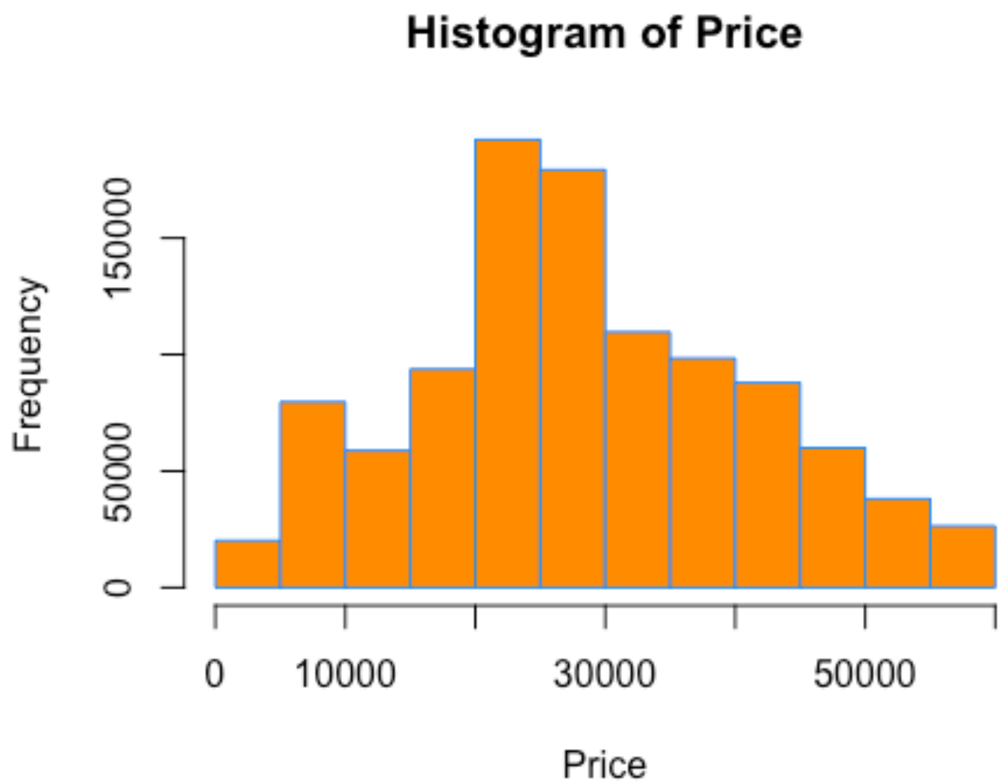
```r
ggplot(df_cars_data, aes(x = mileage, y = price)) +
  geom_point(color = "dodgerblue", alpha = 0.7) +
  geom_smooth(method = "lm", color = "black", se = F) +
  labs(title = "Price vs. Mileage",
       x = "Mileage",
       y = "Price") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5)) +
  facet_wrap(~make_name) +
  coord_cartesian(ylim = c(0, NA))

## `geom_smooth()` using formula = 'y ~ x'
```



Price

Vs Mileage by make_name
Almost all make_name have negative correlation between price and mileage.

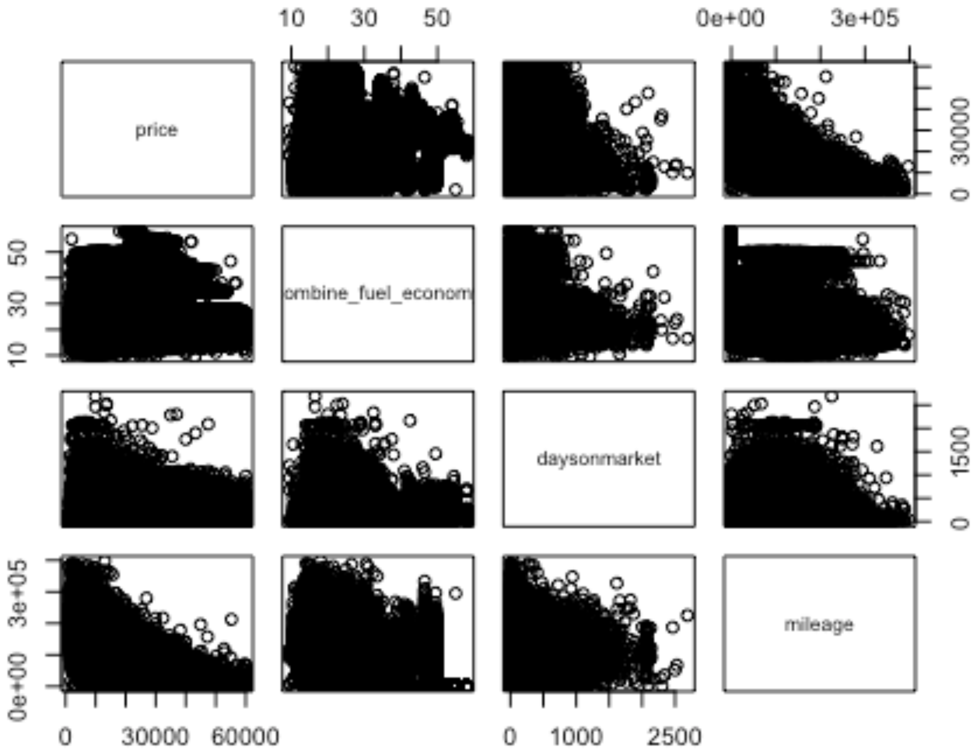
```r
hist(df_cars_data$price,
xlab = "Price",
main = "Histogram of Price",
col = "darkorange",
border = "dodgerblue",
breaks = 20)
```



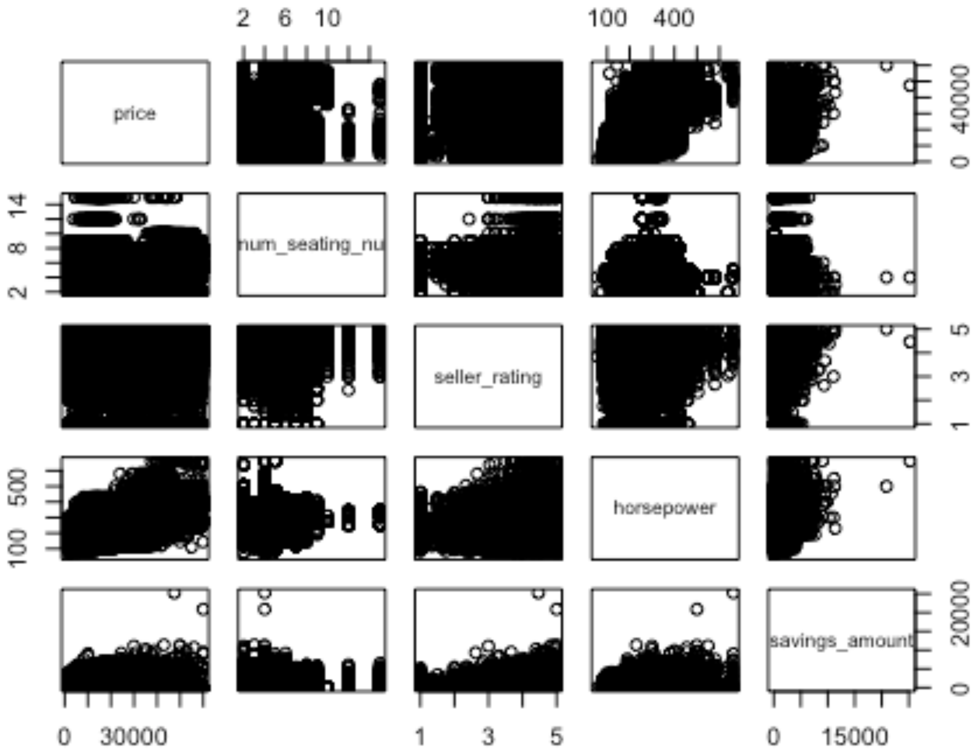
###

Right Skewed Price Values:
The histogram of price appears to be right-skewed. The distribution has a longer tail extending towards the higher price values. We may need to do log transformation of the price variable as it may be beneficial for building a linear regression model.

```
pairs(~ price + combine_fuel_economy + daysonmarket +
        mileage, data=df_cars_data)
```




```r
pairs(~ price + maximum_seating_numeric + seller_rating +
        horsepower + savings_amount, data=df_cars_data)
```



###
Features Selection For further process, I am selecting following variables after analysing the above correlations with prices.

Target variable:

```
price
```

Features:

```
combined_fuel_economy
daysonmarket
mileage
body_type
maximum_seating_seats
seller_rating
horsepower
savings_amount
```

I have decided on this columns after a thorough EDA of the data set.

```
columns_to_keep <- c("price", "combine_fuel_economy", "daysonmarket",
                     "mileage", "body_type", "fuel_type",
                     "maximum_seating_numeric", "make_name", "seller_rating",
                     "horsepower", "savings_amount")
cars_data_subset <- df_cars_data %>% select(all_of(columns_to_keep))
```

```
# glimpse(cars_data_subset)
```

Step 4: Hypothesis Testing

Hypothesis 1:

```
# Hypothesis 1: Is there a significant difference in prices between different
body types?
body_type_anova <- aov(price ~ body_type, data = cars_data_subset)
summary(body_type_anova)
```

```
##                  Df    Sum Sq   Mean Sq F value Pr(>F)    
## body_type         8 4.623e+13 5.779e+12   48044 <2e-16 ***
## Residuals   1044663 1.257e+14 1.203e+08                   
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Post-hoc test if ANOVA is significant
tukey_results <- TukeyHSD(body_type_anova)
print("Tukey's HSD test results:")
```

```
## [1] "Tukey's HSD test results:"
```

```
print(tukey_results)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
## 
## Fit: aov(formula = price ~ body_type, data = cars_data_subset)
## 
## $body_type
##                                    diff         lwr         upr    p adj
## Coupe-Convertible            -3088.0150  -3698.3937  -2477.6362 0.00e+00
## Hatchback-Convertible        -9766.4688 -10354.5247  -9178.4128 0.00e+00
## Minivan-Convertible           2025.6495   1429.5404   2621.7587 0.00e+00
## Pickup Truck-Convertible     12480.5720  11922.5188  13038.6253 0.00e+00
## Sedan-Convertible            -6897.3108  -7454.4572  -6340.1644 0.00e+00
## SUV / Crossover-Convertible    909.8591    355.0651   1464.6532 1.29e-05
## Van-Convertible              -3827.0293  -4458.0563  -3196.0023 0.00e+00
## Wagon-Convertible           -19418.4714 -20231.4047 -18605.5381 0.00e+00
## Hatchback-Coupe              -6678.4538  -7005.9222  -6350.9854 0.00e+00
## Minivan-Coupe                 5113.6645   4771.9455   5455.3835 0.00e+00
## Pickup Truck-Coupe           15568.5870  15298.6811  15838.4929 0.00e+00
## Sedan-Coupe                  -3809.2958  -4077.3216  -3541.2700 0.00e+00
## SUV / Crossover-Coupe         3997.8741   3734.7730   4260.9752 0.00e+00
## Van-Coupe                     -739.0143  -1138.5400   -339.4886 3.00e-07
## Wagon-Coupe                 -16330.4564 -16980.2920 -15680.6209 0.00e+00
## Minivan-Hatchback            11792.1183  11492.0858  12092.1508 0.00e+00
## Pickup Truck-Hatchback       22247.0408  22032.3391  22461.7425 0.00e+00
## Sedan-Hatchback               2869.1580   2656.8247   3081.4913 0.00e+00
## SUV / Crossover-Hatchback    10676.3279  10470.2459  10882.4100 0.00e+00
## Van-Hatchback                 5939.4395   5574.9283   6303.9507 0.00e+00
```

```
## Wagon-Hatchback                   -9652.0026 -10280.9174  -9023.0878 0.00e+00
## Pickup Truck-Minivan             10454.9225  10219.0557  10690.7893 0.00e+00
## Sedan-Minivan                     -8922.9603  -9156.6733  -8689.2473 0.00e+00
## SUV / Crossover-Minivan           -1115.7904  -1343.8389   -887.7418 0.00e+00
## Van-Minivan                       -5852.6788  -6230.0443  -5475.3133 0.00e+00
## Wagon-Minivan                    -21444.1209 -22080.5721 -20807.6697 0.00e+00
## Sedan-Pickup Truck               -19377.8828 -19481.3192 -19274.4464 0.00e+00
## SUV / Crossover-Pickup Truck -11570.7129 -11660.6210 -11480.8048 0.00e+00
## Van-Pickup Truck                 -16307.6013 -16621.4198 -15993.7828 0.00e+00
## Wagon-Pickup Truck               -31899.0434 -32499.9989 -31298.0879 0.00e+00
## SUV / Crossover-Sedan              7807.1699   7723.0742   7891.2657 0.00e+00
## Van-Sedan                          3070.2815   2758.0786   3382.4844 0.00e+00
## Wagon-Sedan                      -12521.1606 -13121.2740 -11921.0472 0.00e+00
## Van-SUV / Crossover               -4736.8884  -5044.8739  -4428.9029 0.00e+00
## Wagon-SUV / Crossover            -20328.3305 -20926.2607 -19730.4004 0.00e+00
## Wagon-Van                        -15591.4421 -16260.7097 -14922.1745 0.00e+00
```

The analysis shows that car body type significantly affects price (p < 2e-16). Tukey's test confirms notable price differences between body types, such as pickup trucks being costlier than sedans and wagons being cheaper than SUVs, all highly significant.

Hypothesis 2:

```
# Hypothesis 2: Is there a significant correlation between price and mileage?
cor_test_mileage <- cor.test(cars_data_subset$price,
cars_data_subset$mileage)
print("Correlation test between price and mileage:")

## [1] "Correlation test between price and mileage:"

print(cor_test_mileage)

##
##  Pearson's product-moment correlation
##
## data:  cars_data_subset$price and cars_data_subset$mileage
## t = -797.7, df = 1044670, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.6164466 -0.6140632
## sample estimates:
##        cor
## -0.6152563
```

The correlation test reveals a significant negative relationship between car price and mileage (correlation = -0.615, p < 2.2e-16). Higher mileage is associated with lower prices, with a 95% confidence interval for the correlation ranging from -0.616 to -0.614.

Hypothesis 3:

```r
# Hypothesis 3: Are Toyota prices significantly different from the
# mean price of all other makes?
toyota_prices <- cars_data_subset$price[cars_data_subset$make_name ==
"Toyota"]
other_prices <- cars_data_subset$price[cars_data_subset$make_name !=
"Toyota"]
t_test_toyota <- t.test(toyota_prices, other_prices)
print("T-test comparing Toyota prices with other makes:")

## [1] "T-test comparing Toyota prices with other makes:"

print(t_test_toyota)

##
##  Welch Two Sample t-test
##
## data:  toyota_prices and other_prices
## t = -43.606, df = 160474, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1690.327 -1544.911
## sample estimates:
## mean of x mean of y
##  27021.80  28639.42
```

The t-test indicates a significant difference in mean prices between Toyota vehicles and other makes (p < 2.2e-16). Toyota vehicles have a lower average price ($27,021.80) compared to other makes ($28,639.42), with a 95% confidence interval for the mean difference between -1,690.33 and -1,544.91.

PART 2: MODEL BUILDING AND DIAGNOSTICS

```r
# Splitting the dataset into training and testing sets
split <- sample.split(cars_data_subset$price, SplitRatio = 0.7)
df_train <- subset(cars_data_subset, split == TRUE)
df_test <- subset(cars_data_subset, split == FALSE)

# Initializing K-Fold Cross-Validation (k = 5)
train_control <- trainControl(method = "cv", number = 5)

colnames(df_train)

##  [1] "price"                  "combine_fuel_economy"
##  [3] "daysonmarket"           "mileage"
##  [5] "body_type"              "fuel_type"
##  [7] "maximum_seating_numeric" "make_name"
##  [9] "seller_rating"          "horsepower"
## [11] "savings_amount"
```

Building the model

```
df_train$log_price <- log(df_train$price)
df_test$log_price <- log(df_test$price)
```

Model 1: Model without Interaction

First we will evaluate our model without interaction:

```
model_no_interaction <- train(
  log_price ~ mileage + horsepower + combine_fuel_economy + daysonmarket +
seller_rating +  maximum_seating_numeric + savings_amount,
  data = df_train,
  method = 'lm',
  trControl = train_control
  )
model_no_interaction
```

```
## Linear Regression
##
## 732175 samples
##      7 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 585741, 585740, 585741, 585739, 585739
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.2225011  0.8475355  0.1642396
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

K-Fold Cross Validation

K-fold cross-validation was used with 732,175 samples and 7 predictors, splitting the data into 5 folds. Each fold trained on about 585,740 samples. The model's performance was evaluated using metrics: RMSE (0.2225011), R-squared (0.8475355), and MAE (0.1642396). These results indicate the model has a good fit, explaining about 84% of the variance in the data, though some errors in predictions remain.

```
summary(model_no_interaction)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.93378 -0.12543  0.00356  0.13183  2.39106
##
```

```
## Coefficients:
##                           Estimate Std. Error  t value Pr(>|t|)    
## (Intercept)              9.268e+00  3.427e-03  2704.60   <2e-16 ***
## mileage                 -9.113e-06  6.879e-09 -1324.67   <2e-16 ***
## horsepower               3.162e-03  4.637e-06   681.88   <2e-16 ***
## combine_fuel_economy     1.073e-03  5.669e-05    18.93   <2e-16 ***
## daysonmarket            -6.419e-05  2.337e-06   -27.47   <2e-16 ***
## seller_rating            2.323e-02  5.020e-04    46.27   <2e-16 ***
## maximum_seating_numeric  3.464e-02  2.719e-04   127.42   <2e-16 ***
## savings_amount           1.090e-05  6.695e-07    16.28   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.2225 on 732167 degrees of freedom
## Multiple R-squared:  0.8475, Adjusted R-squared:  0.8475 
## F-statistic: 5.814e+05 on 7 and 732167 DF,  p-value: < 2.2e-16
```

Key takeaway from above summary:

- A linear regression model was built with seven predictors, showing strong performance ($R^2 = 0.8475$).
- Significant predictors include mileage (-9.113e-06), horsepower (3.162e-03), days on market (-6.419e-05), seller rating (2.323e-02), maximum seating (3.464e-02), and savings amount (1.090e-05).
- The residual standard error is 0.2225, and the F-statistic (5.814e+05, $p < 2.2e-16$) confirms the model's overall significance.

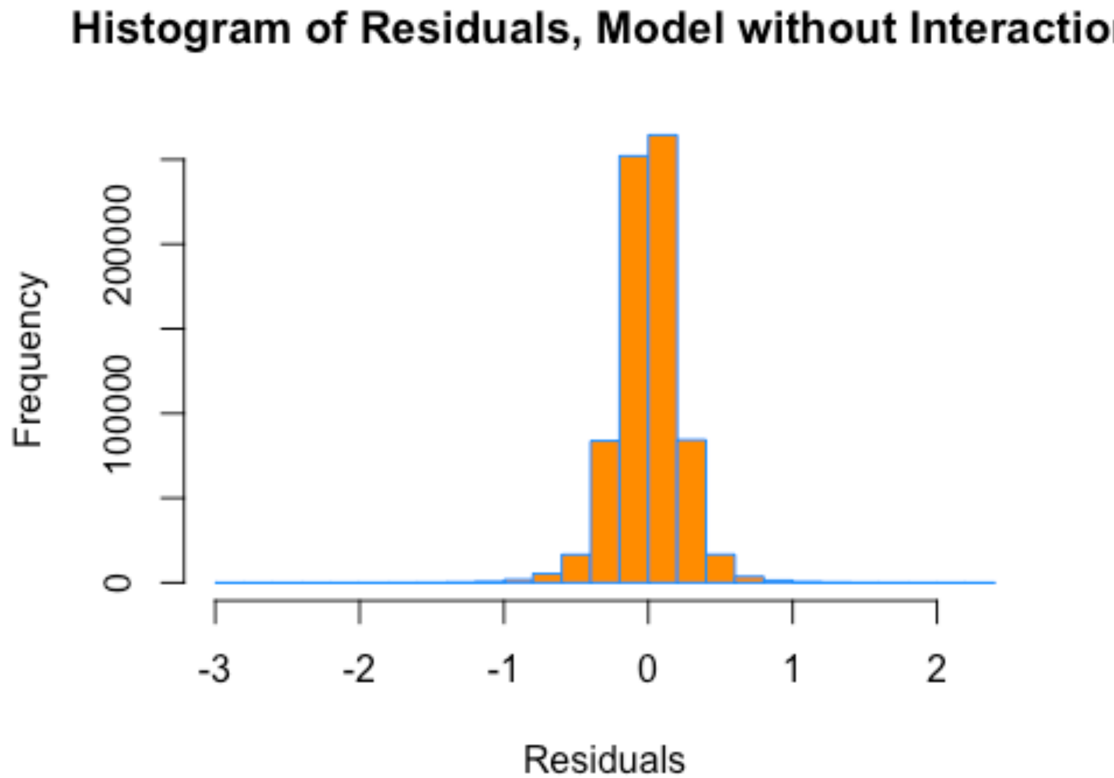
MODEL 1: (Model with no Interaction) Evaluation and Diagnostics.

Plotting residual histogram:

Histograms

We have a number of tools for assessing the normality assumption. The most obvious would be to make a histogram of the residuals. If it appears roughly normal, then we'll believe the errors could truly be normal.

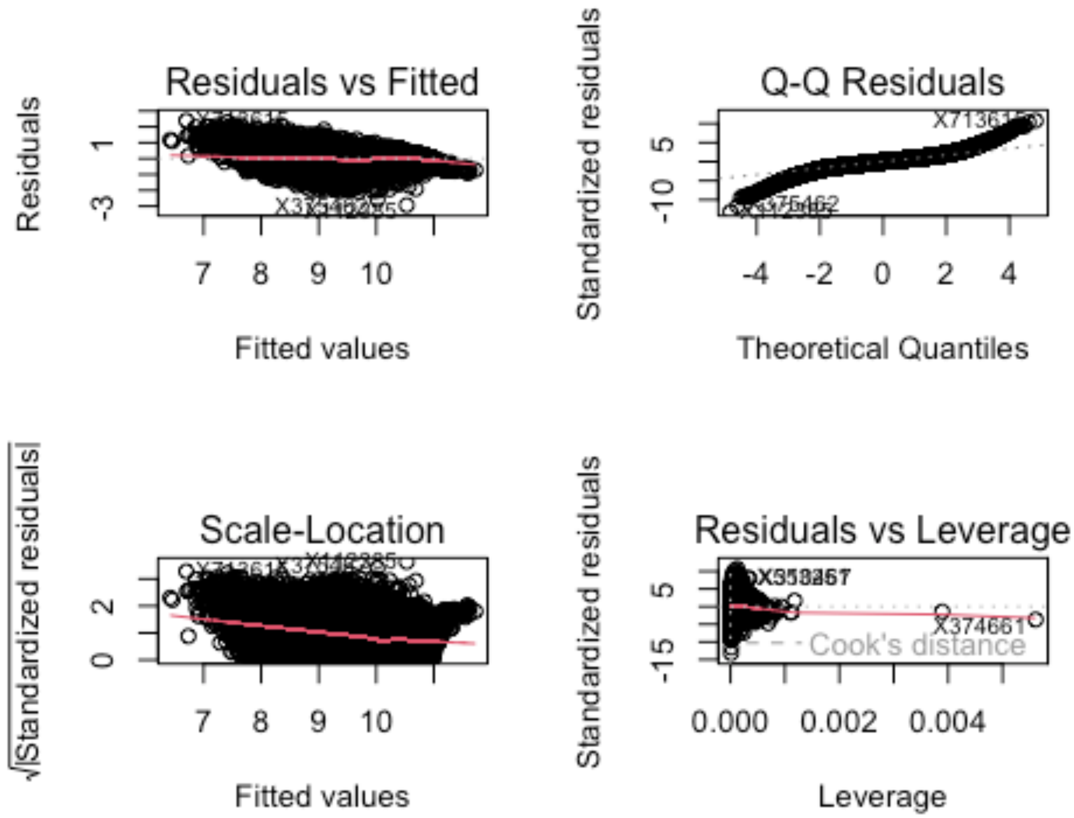
```
hist(resid(model_no_interaction),
xlab = "Residuals",
main = "Histogram of Residuals, Model without Interaction",
col = "darkorange",
border = "dodgerblue",
breaks = 20)
```



###

Plotting residual histogram:
Plot 1: Residuals Vs Fitted - The points are scattered fairly evenly around the horizontal line at 0, suggesting the assumption of homoscedasticity (constant variance) is likely met. Plot 2: Q-Q Residuals - There may be some slight deviations from the line at the tails, but overall the residuals appear to be reasonably close to a normal distribution. Plot 3: Scale-Location - The points are fairly evenly spread around the horizontal line, indicating the assumption of homoscedasticity is likely met. Plot 4: Residuals vs Leverage - The residuals are fairly evenly distributed across the range of leverage values.

```
model_no_in <- model_no_interaction$finalModel
par(mfrow = c(2, 2))
plot(model_no_in)
```



###

Breusch-Pagan Test
Testing for homoscedasticity; the test for constant variance. • Null Hypothesis (Ho): Homoscedasticity. The errors have constant variance about the true model.
• Alternative Hypothesis(HA): Heteroscedasticity. The errors have non-constant variance about the true model.

```
base_model <- model_no_interaction$finalModel
bptest(base_model)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  base_model
## BP = 109417, df = 7, p-value < 2.2e-16
```

Here, the test statistic (BP = 109417) with 7 degrees of freedom and a p-value < 2.2e-16 indicates strong evidence of heteroscedasticity, meaning the residuals' variance is not constant.

Shapiro-Wilk Test

Null Hypothesis (Ho): The data (residuals) follow a normal distribution
Alternative Hypothesis(HA): The data(residuals) does not follow a normal distribution

```r
sample_residuals <- sample(resid(model_no_interaction), size = 5000)
shapiro.test(sample_residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  sample_residuals
## W = 0.96435, p-value < 2.2e-16
```

Above Shapiro-Wilk normality test result interpretation:

- The W-statistic is 0.96435 (close to 1) which indicate the data are somewhat close to a normal distribution but not sure.
- The p-value, however, is very small (p = 2.2e-16) which < 0.005. So we reject the null hypothesis suggesting that the residuals do not follow a normal distribution.

MODEL 2: Model with Interaction

```r
model_with_interaction <- train(
  log_price ~ mileage + horsepower + body_type + mileage:combine_fuel_economy
+
    body_type:daysonmarket:seller_rating + mileage:savings_amount,
  data = df_train,
  method = 'lm',
  trControl = train_control
  )
model_with_interaction
```

```
## Linear Regression
##
## 732175 samples
##      7 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 585741, 585741, 585739, 585739, 585740
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.2122681  0.8612358  0.1550252
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

Model Result:

Model was evaluated using 5-fold cross-validation on 732,175 samples with 7 predictors.

Result:

- Root Mean Square Error (RMSE): 0.2122681 (average prediction error magnitude).
- R-squared: 0.8612358 (model explains ~86% of the variance in the data).

- Mean Absolute Error (MAE): 0.1550252 (average absolute difference between predicted and actual values).

```
summary(model_with_interaction)

##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.95954 -0.11450  0.00553  0.11952  2.42063
##
## Coefficients:
##                                                       Estimate Std.
Error
## (Intercept)                                          9.347e+00  4.918e-
03
## mileage                                             -9.142e-06  1.989e-
08
## horsepower                                           3.153e-03  4.331e-
06
## body_typeCoupe                                      -2.189e-02  5.202e-
03
## body_typeHatchback                                   1.491e-01  5.174e-
03
## body_typeMinivan                                     2.503e-01  5.120e-
03
## `body_typePickup Truck`                              2.397e-01  4.734e-
03
## body_typeSedan                                       1.435e-01  4.785e-
03
## `body_typeSUV / Crossover`                           2.942e-01  4.734e-
03
## body_typeVan                                         2.742e-01  5.857e-
03
## body_typeWagon                                       2.763e-02  7.052e-
03
## `mileage:combine_fuel_economy`                      -1.267e-08  8.200e-
10
## `mileage:savings_amount`                             6.635e-10  5.646e-
12
## `body_typeConvertible:daysonmarket:seller_rating`   -1.253e-05  6.561e-
06
## `body_typeCoupe:daysonmarket:seller_rating`          1.098e-05  4.012e-
06
## `body_typeHatchback:daysonmarket:seller_rating`      2.646e-05  3.212e-
06
## `body_typeMinivan:daysonmarket:seller_rating`       -1.563e-05  3.621e-
06
## `body_typePickup Truck:daysonmarket:seller_rating`  -1.827e-05  1.204e-
```

```
06
## `body_typeSedan:daysonmarket:seller_rating`           -9.477e-06  1.108e-
06
## `body_typeSUV / Crossover:daysonmarket:seller_rating` -3.468e-05  7.361e-
07
## `body_typeVan:daysonmarket:seller_rating`              9.469e-06  3.482e-
06
## `body_typeWagon:daysonmarket:seller_rating`           -7.959e-06  7.823e-
06
##                                                        t value Pr(>|t|)
## (Intercept)                                           1900.732  < 2e-16
***
## mileage                                               -459.545  < 2e-16
***
## horsepower                                             727.997  < 2e-16
***
## body_typeCoupe                                          -4.209 2.57e-05
***
## body_typeHatchback                                      28.824  < 2e-16
***
## body_typeMinivan                                        48.898  < 2e-16
***
## `body_typePickup Truck`                                 50.641  < 2e-16
***
## body_typeSedan                                          29.982  < 2e-16
***
## `body_typeSUV / Crossover`                              62.147  < 2e-16
***
## body_typeVan                                            46.821  < 2e-16
***
## body_typeWagon                                           3.917 8.95e-05
***
## `mileage:combine_fuel_economy`                         -15.453  < 2e-16
***
## `mileage:savings_amount`                               117.513  < 2e-16
***
## `body_typeConvertible:daysonmarket:seller_rating`       -1.910  0.05615 .
## `body_typeCoupe:daysonmarket:seller_rating`              2.736  0.00621 **
## `body_typeHatchback:daysonmarket:seller_rating`          8.239  < 2e-16
***
## `body_typeMinivan:daysonmarket:seller_rating`           -4.316 1.59e-05
***
## `body_typePickup Truck:daysonmarket:seller_rating`     -15.179  < 2e-16
***
## `body_typeSedan:daysonmarket:seller_rating`             -8.556  < 2e-16
***
## `body_typeSUV / Crossover:daysonmarket:seller_rating`  -47.110  < 2e-16
***
## `body_typeVan:daysonmarket:seller_rating`                2.719  0.00654 **
## `body_typeWagon:daysonmarket:seller_rating`             -1.017  0.30896
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2123 on 732153 degrees of freedom
## Multiple R-squared:  0.8613, Adjusted R-squared:  0.8612
## F-statistic: 2.164e+05 on 21 and 732153 DF,  p-value: < 2.2e-16
```

Key takeaway from above summary:

- The Residual Standard Error is 0.2123 on 732153 degrees of freedom which is small and indicates better fit.

- The adjusted R-squared value is 0.8612 which shows significant variability in response variable.

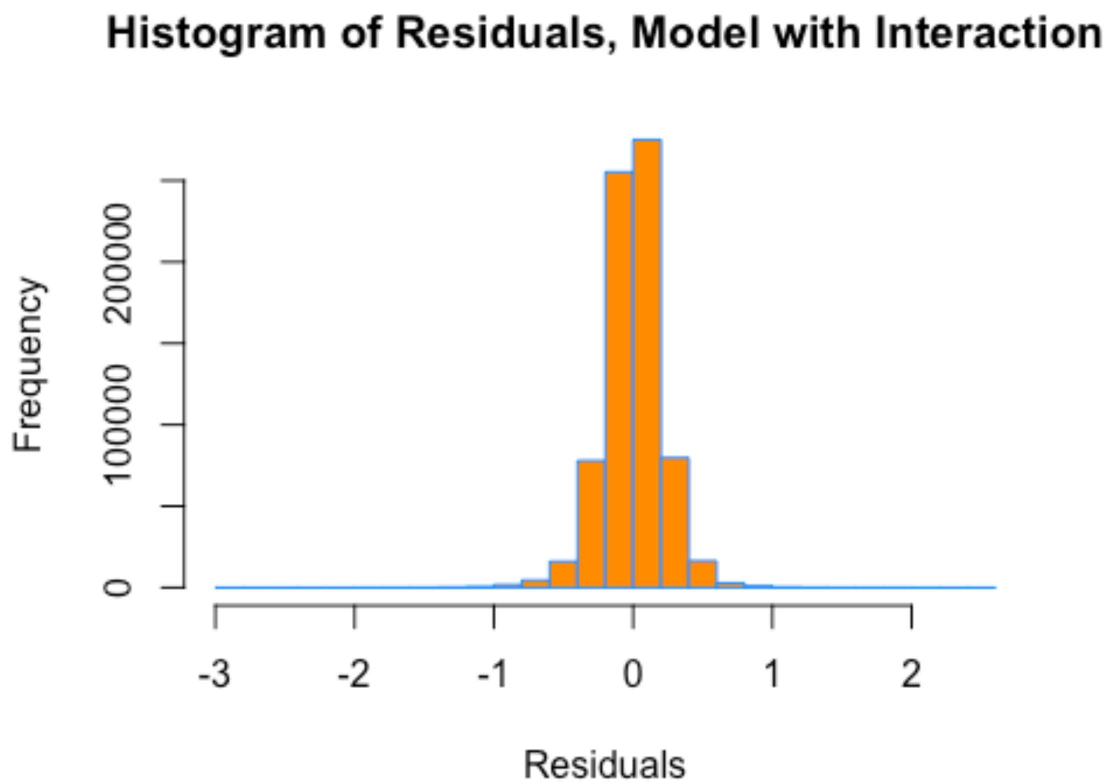
- The p-value is (< 2.2e-16) which indicate the model is highly significant i.e it explains variability in response variable significantly better than model with no predictors.

MODEL 2: (Model with Interaction) Evaluation and Diagnostics.

Histograms

We have a number of tools for assessing the normality assumption. The most obvious would be to make a histogram of the residuals. If it appears roughly normal, then we'll believe the errors could truly be normal.

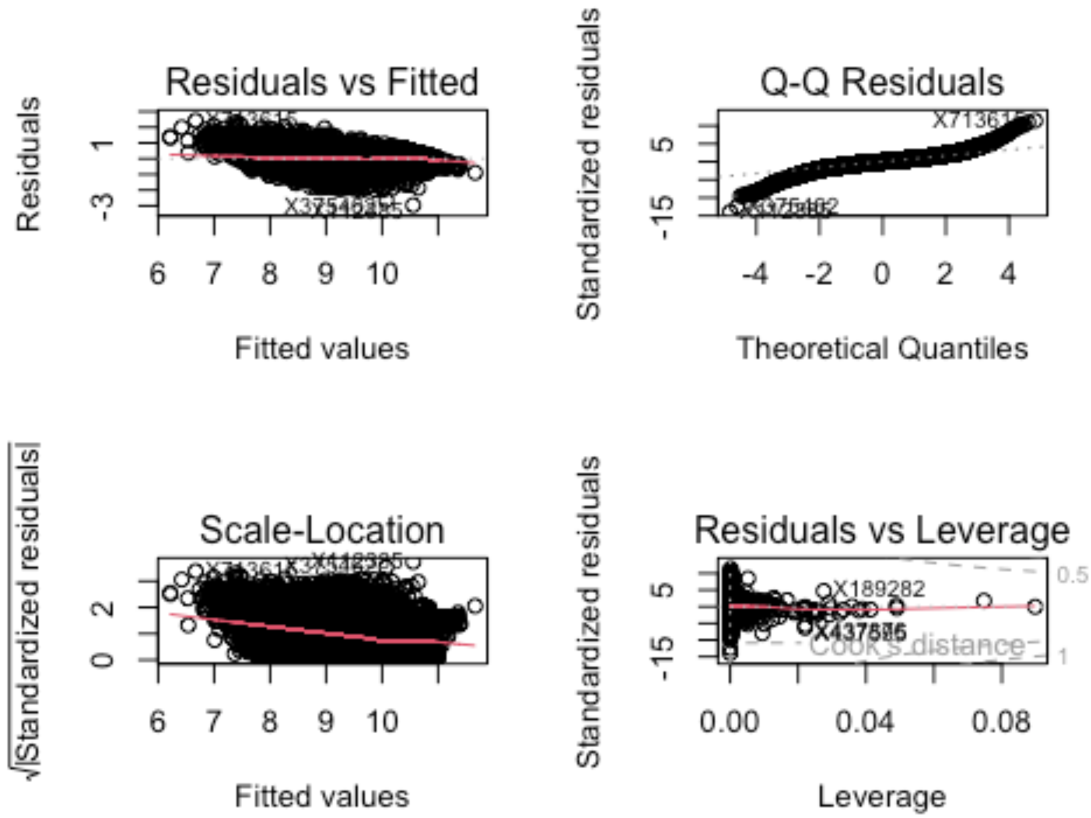
```
hist(resid(model_with_interaction),
xlab = "Residuals",
main = "Histogram of Residuals, Model with Interaction",
col = "darkorange",
border = "dodgerblue",
breaks = 20)
```



###

Plotting residual histogram:
Plot 1: Residuals Vs Fitted - The plot appears to show a fairly even spread of residuals around the horizontal line at 0, indicating the assumption of homoscedasticity (constant variance) is likely met. Plot 2: Q-Q Residuals
- The points generally follow the straight diagonal line, indicating the residuals are approximately normally distributed. We will run further tests to confirm it. Plot 3: Scale-Location - The plot exhibits a fairly even spread of points around the horizontal line. Plot 4: Residuals vs Leverage - The plot does not appear to show any concerning outliers or high-leverage points that could significantly influence the model.

```r
model_with_in <- model_with_interaction$finalModel
par(mfrow = c(2, 2))
plot(model_with_in)
```



###

Breusch-Pagan Test
Testing for homoscedasticity; the test for constant variance.

Null Hypothesis (Ho): Homoscedasticity. The errors have constant variance about the true model.
Alternative Hypothesis(HA): Heteroscedasticity. The errors have non-constant variance about the true model.

```
interaction_model <- model_with_interaction$finalModel
bptest(interaction_model)

##
##  studentized Breusch-Pagan test
##
## data:  interaction_model
## BP = 105243, df = 21, p-value < 2.2e-16
```

Test Result Summary:

- BP statistic: 105243 (indicating a large deviation from homoscedasticity)
- For the case of model with interaction, the p-value is equals to 2.2e-16 meaning that it is much lesser than the typical threshold of 0.05. In this case we reject the null hypothesis; meaning there is evidence of heteroscedascity in the interaction model.

Shapiro-Wilk Test

Null Hypothesis (Ho): The data (residuals) follow a normal distribution
Alternative Hypothesis(HA): The data(residuals) does not follow a normal distribution

```
sample_residuals <- sample(resid(model_with_interaction), size = 5000)
shapiro.test(sample_residuals)

##
##  Shapiro-Wilk normality test
##
## data:  sample_residuals
## W = 0.96146, p-value < 2.2e-16
```

Above: model with interaction test result interpretation:

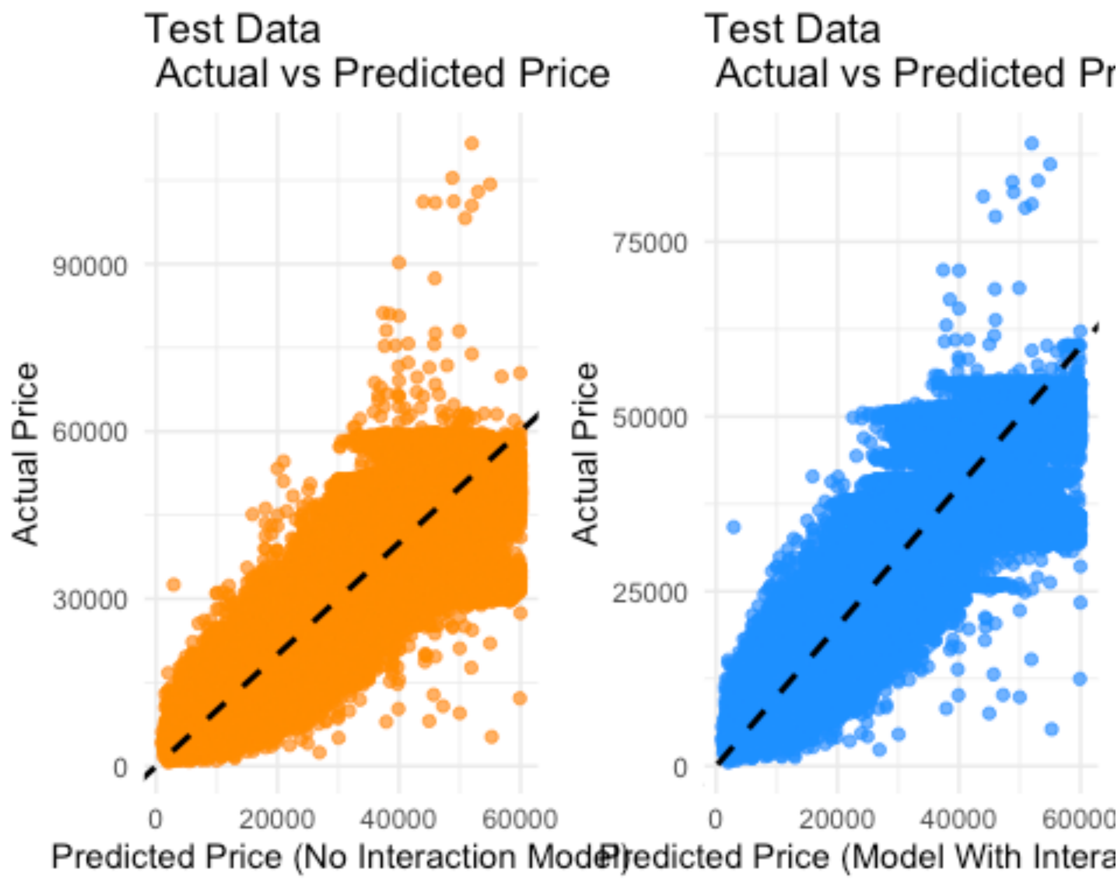
- The W-statistic (0.96146) is close to 1 which indicate the data are somewhat close to a normal distribution but not sure.

- The p-value p = 2.2e-16 which is extremely small and much smaller than the common threshold of 0.005.
  So we reject the null hypothesis suggesting that the residuals slightly deviate from normality.

Part 3: Predictions and Evaluation of the Model.

```
df_test2 <- df_test
df_test$predicted_price <- exp(predict(model_no_interaction, newdata =
df_test))
df_test2$predicted_price <- exp(predict(model_with_interaction, newdata =
df_test2))

par(mfrow=c(2,2), mar=c(4,4,2,0.5))

# first model
plot1 <- ggplot(df_test, aes(x = price, y = predicted_price)) +
  geom_point(alpha = 0.7, color = "darkorange") +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "black",
linewidth=0.9) +
  labs(
    title = "Test Data \n Actual vs Predicted Price",
    x = "Predicted Price (No Interaction Model)",
    y = "Actual Price"
  ) +
  theme_minimal()

# second model
plot2 <- ggplot(df_test2, aes(x = price, y = predicted_price)) +
  geom_point(alpha = 0.7, color = "dodgerblue") +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "black",
```

```r
linewidth=0.9) +
  labs(
    title = "Test Data \n Actual vs Predicted Price",
    x = "Predicted Price (Model With Interaction)",
    y = "Actual Price"
  ) +
  theme_minimal()

grid.arrange(plot1, plot2, ncol=2)
```



The left plot represents the residuals from Model 1 (without interaction)
The right plot represents the residuals from Model 2 (with interaction)
Adding interaction terms enhances the model performance slightly. The interaction terms reduces the residuals making it better for the data.

Overall the model with interaction is the better model.
The orange points in the scatter plot represent residuals from Model 1 (without interaction) and blue points represent residuals from Model 2 (With interaction)

Conclusion:

- Model 1 (no interaction) has an R^2 of 0.8475 and Model 2 (with interaction) has a slightly higher adjusted R^2 of 0.8612. This suggests Model 2 explains slightly more variability in the response variable
- Residual Standard Erro in Model 1 is 0.2225 and Model 2 is 0.2123. Model 2 has a lower residual standard error which indicate a marginally better fit. The differences are relatively small, suggesting both models are robust and perform well.