AI-for-cars.R

gagan

2025-02-20

```r
# Load necessary libraries
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(ggplot2)
library(dplyr)
library(readr)

# Read the dataset
df <- read_csv("~/CSV_Data_Sets/car_model.csv")
```

```
## Rows: 428 Columns: 15
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (7): Make, Model, Type, Origin, DriveTrain, MSRP, Invoice
## dbl (8): EngineSize, Cylinders, Horsepower, MPG_City, MPG_Highway, Weight, W...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# View first few rows
head(df)
```

```
## # A tibble: 6 x 15
##   Make  Model         Type  Origin DriveTrain MSRP  Invoice EngineSize Cylinders
##   <chr> <chr>         <chr> <chr>  <chr>      <chr> <chr>        <dbl>     <dbl>
## 1 Acura MDX           SUV   Asia   All        $36,~ $33,337        3.5         6
## 2 Acura RSX Type S 2~ Sedan Asia   Front      $23,~ $21,761        2           4
## 3 Acura TSX 4dr       Sedan Asia   Front      $26,~ $24,647        2.4         4
## 4 Acura TL 4dr        Sedan Asia   Front      $33,~ $30,299        3.2         6
```

```
## 5 Acura 3.5 RL 4dr    Sedan Asia   Front      $43,~ $39,014        3.5         6
## 6 Acura 3.5 RL w/Nav~ Sedan Asia   Front      $46,~ $41,100        3.5         6
## # i 6 more variables: Horsepower <dbl>, MPG_City <dbl>, MPG_Highway <dbl>,
## #   Weight <dbl>, Wheelbase <dbl>, Length <dbl>
```

```
# Check structure and summary
str(df)
```

```
## spc_tbl_ [428 x 15] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Make       : chr [1:428] "Acura" "Acura" "Acura" "Acura" ...
##  $ Model      : chr [1:428] "MDX" "RSX Type S 2dr" "TSX 4dr" "TL 4dr" ...
##  $ Type       : chr [1:428] "SUV" "Sedan" "Sedan" "Sedan" ...
##  $ Origin     : chr [1:428] "Asia" "Asia" "Asia" "Asia" ...
##  $ DriveTrain : chr [1:428] "All" "Front" "Front" "Front" ...
##  $ MSRP       : chr [1:428] "$36,945" "$23,820" "$26,990" "$33,195" ...
##  $ Invoice    : chr [1:428] "$33,337" "$21,761" "$24,647" "$30,299" ...
##  $ EngineSize : num [1:428] 3.5 2 2.4 3.2 3.5 3.5 3.2 1.8 1.8 3 ...
##  $ Cylinders  : num [1:428] 6 4 4 6 6 6 6 4 4 6 ...
##  $ Horsepower : num [1:428] 265 200 200 270 225 225 290 170 170 220 ...
##  $ MPG_City   : num [1:428] 17 24 22 20 18 18 17 22 23 20 ...
##  $ MPG_Highway: num [1:428] 23 31 29 28 24 24 24 31 30 28 ...
##  $ Weight     : num [1:428] 4451 2778 3230 3575 3880 ...
##  $ Wheelbase  : num [1:428] 106 101 105 108 115 115 100 104 105 104 ...
##  $ Length     : num [1:428] 189 172 183 186 197 197 174 179 180 179 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Make = col_character(),
##   ..   Model = col_character(),
##   ..   Type = col_character(),
##   ..   Origin = col_character(),
##   ..   DriveTrain = col_character(),
##   ..   MSRP = col_character(),
##   ..   Invoice = col_character(),
##   ..   EngineSize = col_double(),
##   ..   Cylinders = col_double(),
##   ..   Horsepower = col_double(),
##   ..   MPG_City = col_double(),
##   ..   MPG_Highway = col_double(),
##   ..   Weight = col_double(),
##   ..   Wheelbase = col_double(),
##   ..   Length = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

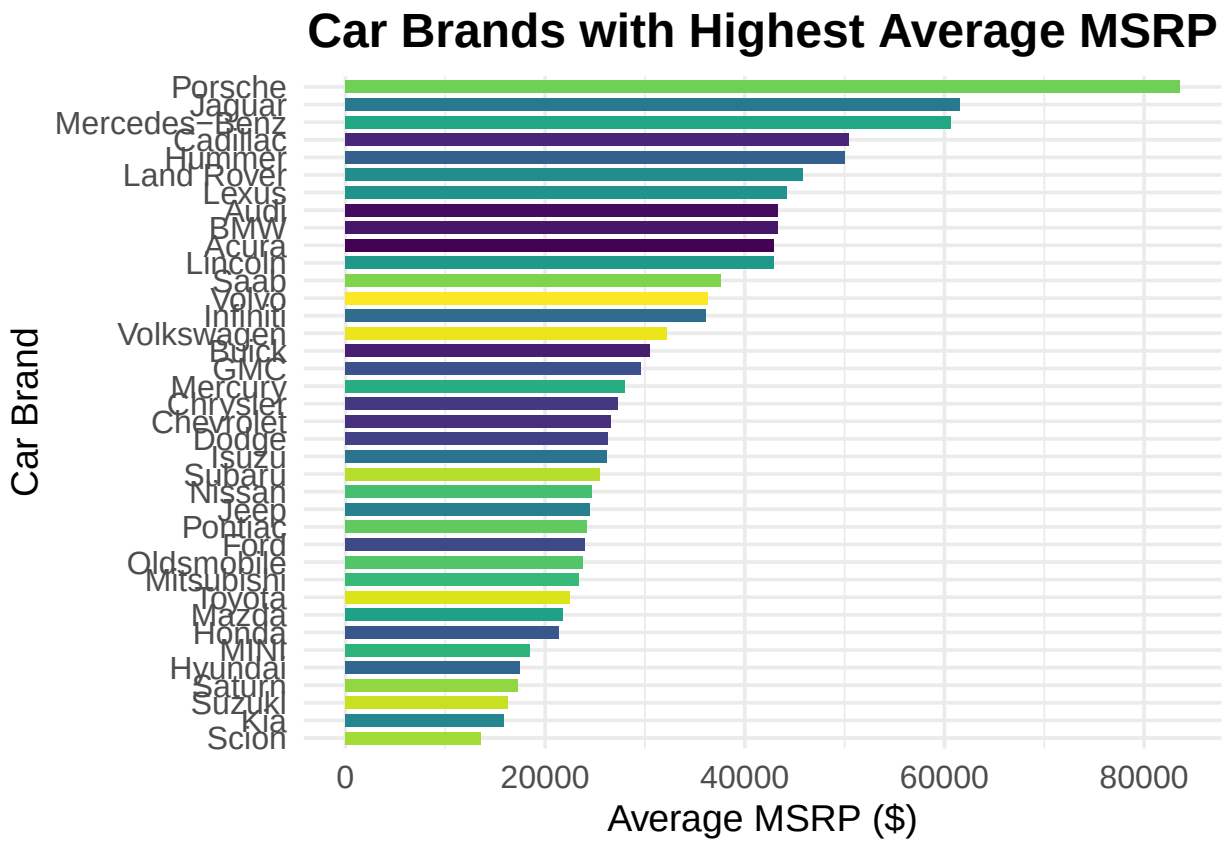
```
summary(df)
```

```
##      Make               Model              Type              Origin
##  Length:428         Length:428         Length:428         Length:428
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
```

```
##
##   DriveTrain            MSRP              Invoice            EngineSize
##  Length:428         Length:428         Length:428         Min.   :1.300
##  Class :character   Class :character   Class :character   1st Qu.:2.375
##  Mode  :character   Mode  :character   Mode  :character   Median :3.000
##                                                           Mean   :3.197
##                                                           3rd Qu.:3.900
##                                                           Max.   :8.300
##
##    Cylinders        Horsepower       MPG_City      MPG_Highway
##  Min.   : 3.000   Min.   : 73.0   Min.   :10.00   Min.   :12.00
##  1st Qu.: 4.000   1st Qu.:165.0   1st Qu.:17.00   1st Qu.:24.00
##  Median : 6.000   Median :210.0   Median :19.00   Median :26.00
##  Mean   : 5.808   Mean   :215.9   Mean   :20.06   Mean   :26.84
##  3rd Qu.: 6.000   3rd Qu.:255.0   3rd Qu.:21.25   3rd Qu.:29.00
##  Max.   :12.000   Max.   :500.0   Max.   :60.00   Max.   :66.00
##  NA's   :2
##      Weight       Wheelbase         Length
##  Min.   :1850   Min.   : 89.0   Min.   :143.0
##  1st Qu.:3104   1st Qu.:103.0   1st Qu.:178.0
##  Median :3474   Median :107.0   Median :187.0
##  Mean   :3578   Mean   :108.2   Mean   :186.4
##  3rd Qu.:3978   3rd Qu.:112.0   3rd Qu.:194.0
##  Max.   :7190   Max.   :144.0   Max.   :238.0
##
```

```r
#Data Cleaning & Preprocessing
# Fill missing values with median for numeric columns
df <- df %>%
  mutate(across(where(is.numeric), ~ifelse(is.na(.), median(., na.rm = TRUE), .)))

#Convert Categorical Columns to Factors
df$Make <- as.factor(df$Make)
df$Model <- as.factor(df$Model)
df$Type <- as.factor(df$Type)
df$Origin <- as.factor(df$Origin)


#Possible Analysis & Visualizations

# Remove the "$" sign and any other unwanted characters, then convert to numeric
df$MSRP <- as.numeric(gsub("[^0-9.]", "", df$MSRP))
df$Invoice <- as.numeric(gsub("[^0-9.]", "", df$Invoice))

# Calculate average MSRP by car brand
msrp_analysis <- df %>%
  group_by(Make) %>%
  summarise(Average_MSRP = mean(MSRP, na.rm = TRUE)) %>%
  arrange(desc(Average_MSRP))  # Sort by average MSRP in descending order


#Which car brands have the highest average MSRP?
ggplot(msrp_analysis, aes(x = reorder(Make, Average_MSRP), y = Average_MSRP, fill = Make)) +
  geom_bar(stat = "identity", width = 0.7) +
```

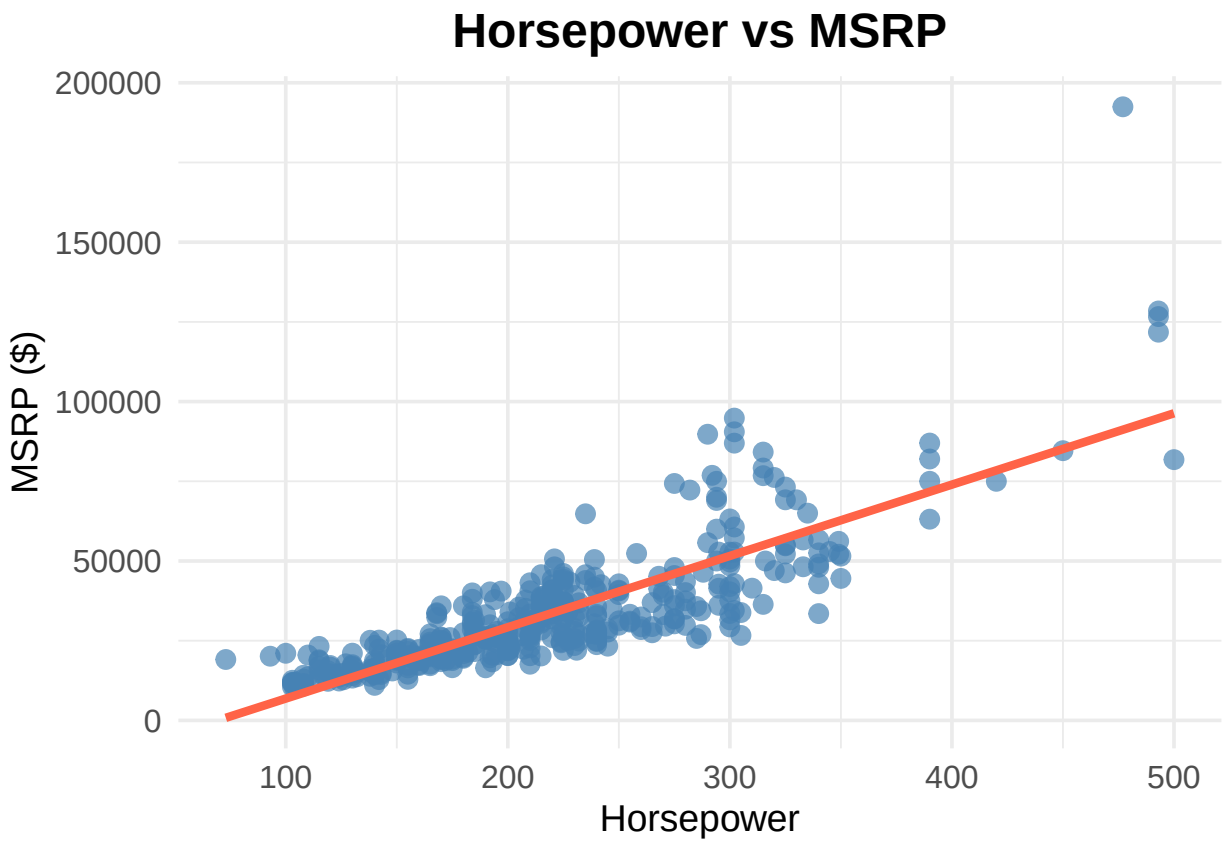
```r
  coord_flip() +
  theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(size = 18, face = "bold", hjust = 0.5),
    axis.title = element_text(size = 14),
    axis.text = element_text(size = 12),
    legend.position = "none"
  ) +
  labs(
    title = "Car Brands with Highest Average MSRP",
    x = "Car Brand",
    y = "Average MSRP ($)"
  ) +
  scale_fill_viridis_d()  # Use a colorblind-friendly palette
```




```r
#HP vs MSRP coherence
ggplot(df, aes(x = Horsepower, y = MSRP)) +
  geom_point(color = "steelblue", alpha = 0.7, size = 3) +
  geom_smooth(method = "lm", se = FALSE, color = "tomato", linewidth = 1.5) +
  theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(size = 18, face = "bold", hjust = 0.5),
    axis.title = element_text(size = 14),
    axis.text = element_text(size = 12)
  ) +
```

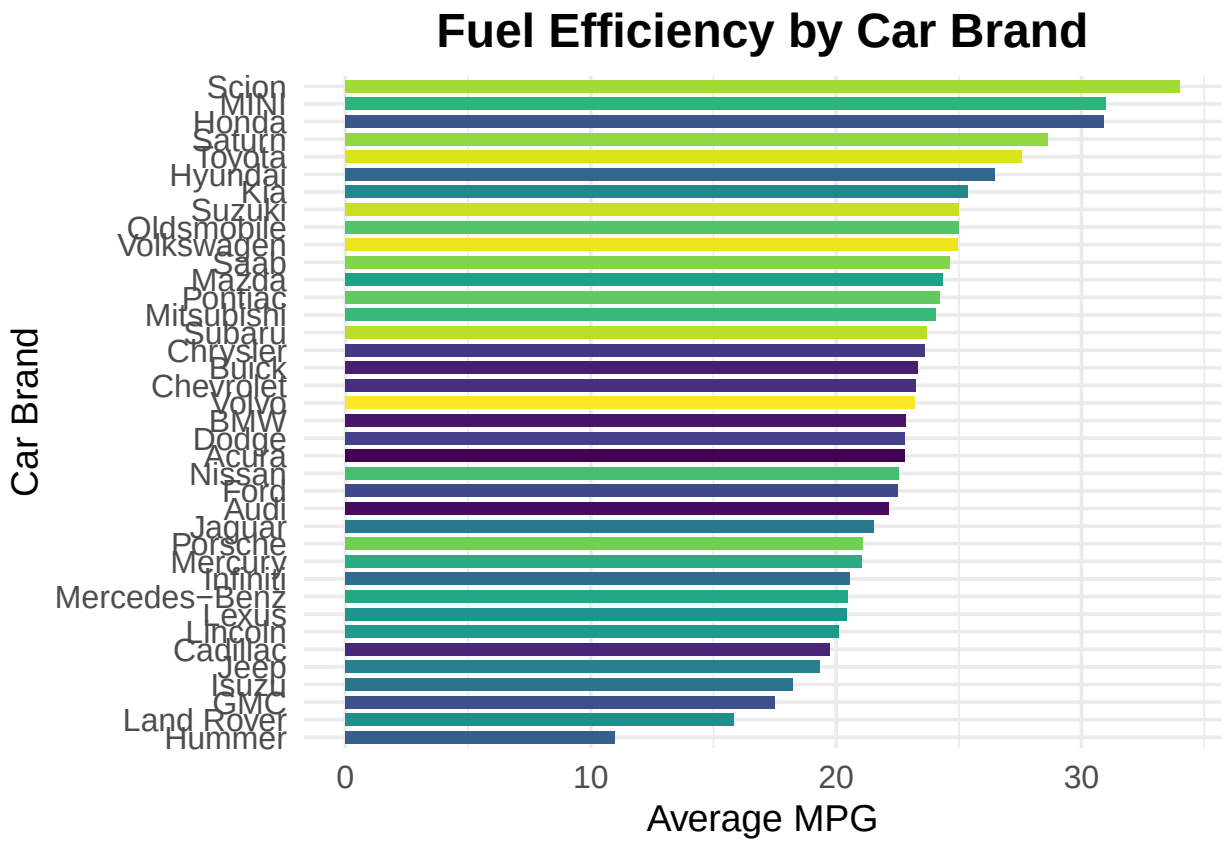
```
  labs(
    title = "Horsepower vs MSRP",
    x = "Horsepower",
    y = "MSRP ($)"
  )
```

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



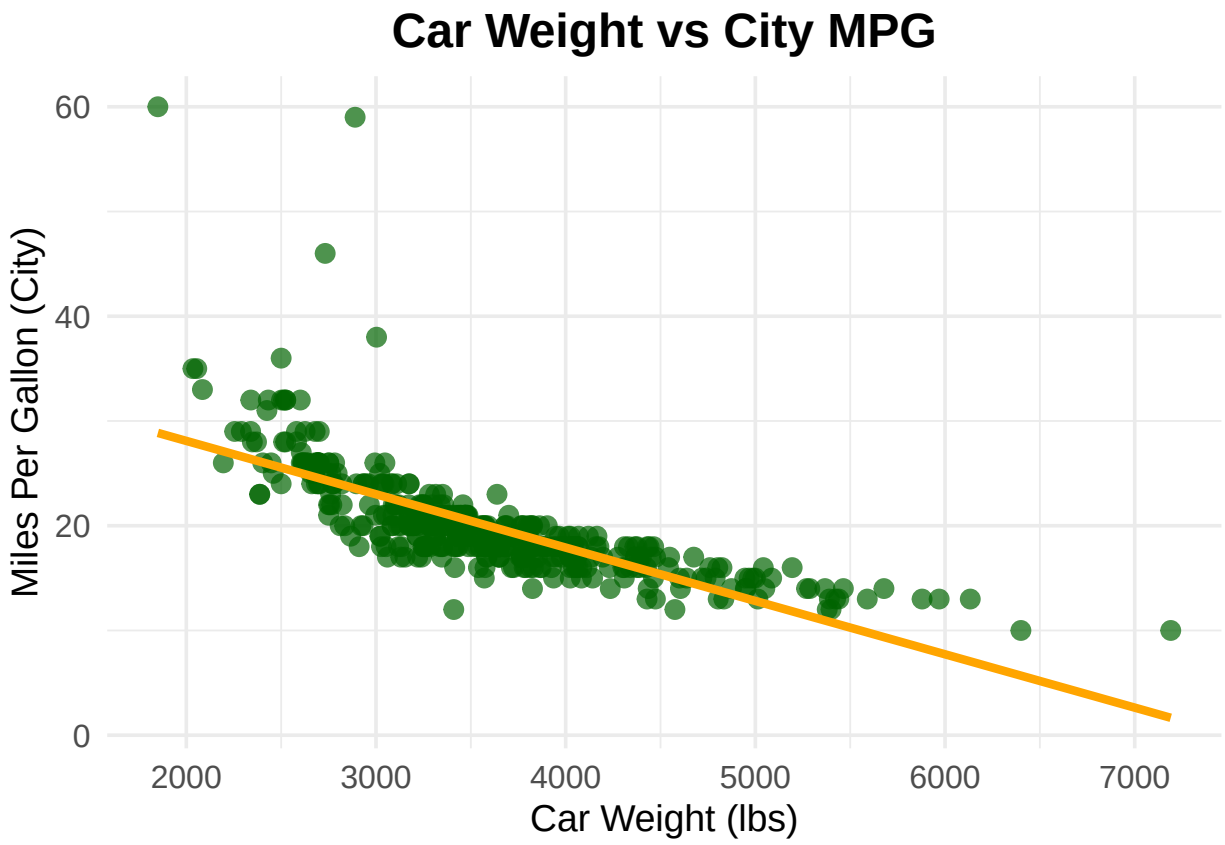

```
# Calculate average MPG by car brand
fuel_efficiency <- df %>%
  group_by(Make) %>%
  summarise(Average_MPG = mean((MPG_City + MPG_Highway) / 2, na.rm = TRUE)) %>%
  arrange(desc(Average_MPG))  # Sort by average MPG in descending order

#Fuel Efficiency Analysis: MPG (Miles Per Gallon)
ggplot(fuel_efficiency, aes(x = reorder(Make, Average_MPG), y = Average_MPG, fill = Make)) +
  geom_bar(stat = "identity", width = 0.7) +
  coord_flip() +
  theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(size = 18, face = "bold", hjust = 0.5),
    axis.title = element_text(size = 14),
    axis.text = element_text(size = 12),
    legend.position = "none"
  ) +
```

```
  labs(
    title = "Fuel Efficiency by Car Brand",
    x = "Car Brand",
    y = "Average MPG"
  ) +
  scale_fill_viridis_d()
```



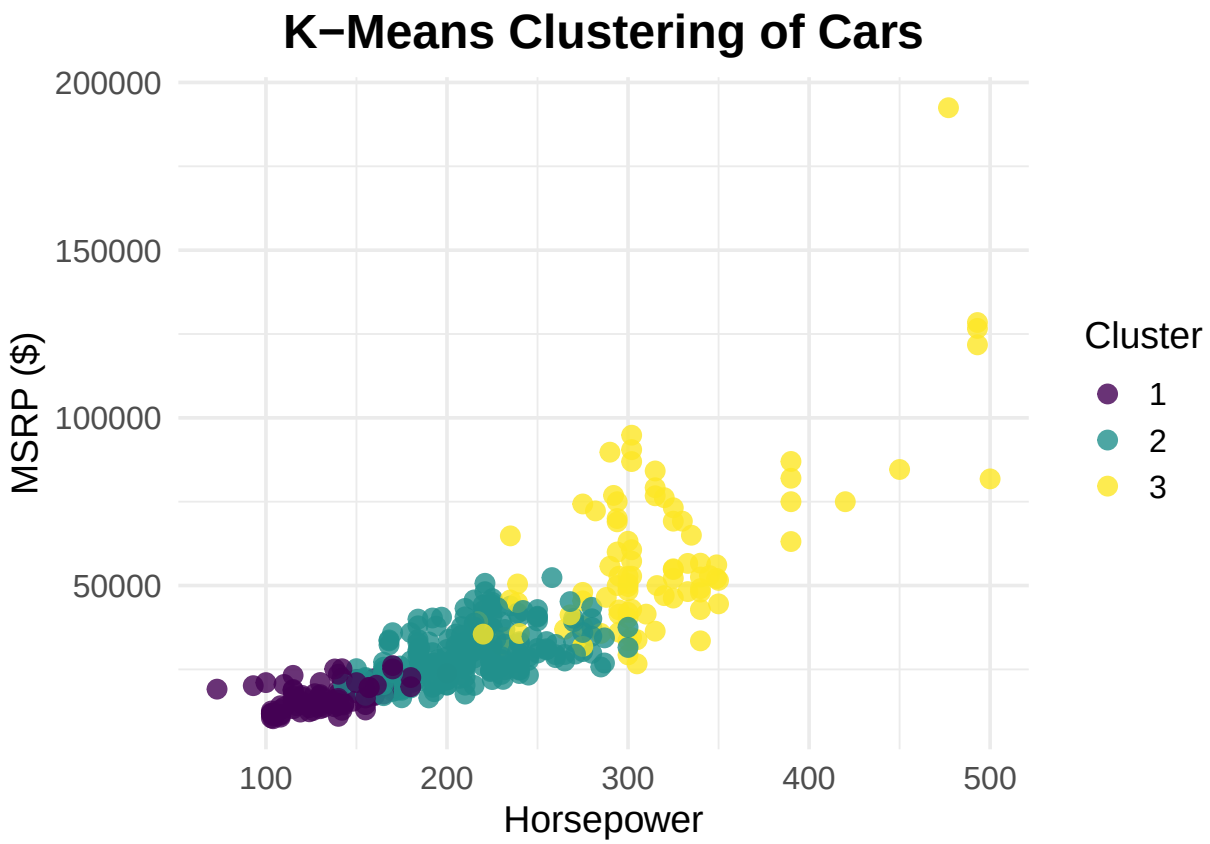

```
#Relationship Between Car Weight & MPG
ggplot(df, aes(x = Weight, y = MPG_City)) +
geom_point(color = "darkgreen", alpha = 0.7, size = 3) +
  geom_smooth(method = "lm", se = FALSE, color = "orange", linewidth = 1.5) +
  theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(size = 18, face = "bold", hjust = 0.5),
    axis.title = element_text(size = 14),
    axis.text = element_text(size = 12)
  ) +
  labs(
    title = "Car Weight vs City MPG",
    x = "Car Weight (lbs)",
    y = "Miles Per Gallon (City)"
  )
```

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




```r
#k means
# Select numeric features for clustering
df_scaled <- df %>%
  select(MSRP, MPG_City, Horsepower, Weight) %>%
  scale()  # Scale the data for K-Means

# Perform K-Means clustering
set.seed(123)  # For reproducibility
kmeans_model <- kmeans(df_scaled, centers = 3)  # 3 clusters

# Add cluster labels to the original data frame
df$Cluster <- as.factor(kmeans_model$cluster)
ggplot(df, aes(x = Horsepower, y = MSRP, color = Cluster)) +
  geom_point(size = 3, alpha = 0.8) +
  theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(size = 18, face = "bold", hjust = 0.5),
    axis.title = element_text(size = 14),
    axis.text = element_text(size = 12),
    legend.title = element_text(size = 14),
    legend.text = element_text(size = 12)
  ) +
  labs(
    title = "K-Means Clustering of Cars",
    x = "Horsepower",
    y = "MSRP ($)"
```

```
  ) +
  scale_color_viridis_d()
```




```
#decision tree
# Install and load the rpart.plot package

library(rpart.plot)
```

```
## Loading required package: rpart
```

```
# Train a decision tree model
library(rpart)
tree_model <- rpart(Type ~ Horsepower + EngineSize + Weight, data = df, method = "class")

# Plot the decision tree
rpart.plot(
  tree_model,
  type = 4,
  extra = 101,
  fallen.leaves = TRUE,
  shadow.col = "gray",
  box.palette = list(
    "Blues",
    "Greens",
```

```
    "Reds"
  ),
  main = "Decision Tree for Car Type Prediction",
  cex = 0.9,
  branch = 0.5,
  under = TRUE,
  tweak = 1.2,
  nn = TRUE,
  pal.thresh = 0.5,
  roundint = FALSE,
  faclen = 0
)
```

```
## Warning: cex and tweak both specified, applying both
```