A Statistical Approach to Used Car Price Prediction

12/09/2024

Steven Qie
Statistics and Computer Science
University of Illinois Urbana-Champaign
qie2@illinois.edu

Brian Gong
Statistics and Computer Science
University of Illinois Urbana-Champaign
brianhg2@illinois.edu

William Yeh
Statistics and Computer Science
University of Illinois Urbana-Champaign
wy16@illinois.edu

Introduction

With the used car market being significantly larger than the new car market, many consumers are realizing that used cars provide a more affordable option. It plays a significant role in the growth and stability of the U.S. economy, driven by changing consumer preferences, economic factors, and the availability of certain cars. Accurately predicting the price of a used car is a challenging but essential task for buyers, sellers, and market analysts/economists alike.

This report aims to develop various predictive models for used car prices using the Used Car Price Prediction Dataset from Kaggle. This dataset comprises of 4,009 data points, representing unique vehicle listings, as well as nine distinct features that serve as key indicators influencing the value of a used car. We follow a very structured and standard approach, including data exploration, preprocessing, model training, and evaluation using relevant performance metrics. By leveraging these methods, we aim to uncover valuable insights into the world of automobiles and the various factors that are driving used car prices.

Need a section on key findings. . . .

Abstract—

white space

white space

white space

white space

white space

white space

white space

white space

white space

We utilized AI tools in this report to enhance and assist in our writing. These tools helped play a big role in ensuring clarity, conciseness, and professionalism. We also utilized AI tools to help us with syntax help when writing code in R, as well as discovering potential bugs in our code.

Literature Review

This literature review aims to summarize key findings and approaches from a few noteworthy research papers focused on used car price prediction.

"Price Prediction of Used Cars Using Machine Learning", written by Chuyang Jin of the University of Sydney, presents a model that can predict a used vehicle's price given their year of production, mileage, tax, miles per gallon, He hopes that his model can benefit and save time for both sellers and buyers who are looking to sell or serach for second-hand vehicles. Jin used a CSV dataset containing 100,000 records of used cars in the UK, focusing specifically on the Mercedes brand. The nine factors that he considered were the following: model, year, selling price, transmission, mileage, fuel type, tax, miles per gallon (mpg), and engine size. While doing exploratory data analysis and preprocessing, Jin noted that many many predictors had skewed distributions. For example, the overwhelming majority of prices fell in the 0-75,000 range, limiting the model's potential effectiveness for higher price ranges. Jin deemed these data points as outliers and excluded them to ensure that the model would be more accurate and usable. After testing various forms of regression, namely linear, polynomial, SVR, Decision Trees, and Random Forests, Jin found Random Forest Regression yielded the best R squared value of 0.90416.

"Used Car Price Prediction using Machine Learning: A Case Study", written by Mustapha Hankar, Marouane Birjali, and Abderrahim Beni-Hssane, applies several supervised machine learning algorithms to predict used car price prices based on features from a dataset collected from an online eCommerce website called Avito. During preprocessing, the authors of this paper performed recursive feature elimination to maintain only the most relevant features to car prices: year of manufacture, mileage, mark, fuel type, fiscal power, and model. Along with a baseline multiple linear regression model, the study also looked at K-nearest neighbors, Random Forest, Gradient Boosting, and Artificial Neural Networks. The study utilized 2 different performance metrics, R^2 and RMSE, and concluded that the Gradient Boosting Regression Model achieved the best results, with a R^2 of 0.8 and RMSE of 44516.20.

"Car Price Prediction using Supervised and Unsupervised Learning Models and Deep Learning" by Thomas Nsiah approached the problem of car price prediction from a supervised and unsupervised lenses. While supervised models allow a consumer to understand the key factors and predictors that influence pricing of used cars, unsupervised learning oftentimes uncovers hidden connections and patterns within the data. In his paper, Nsiah used a mock dataset of 50,000 UK second hand car sales with features similar to the previous 2 studies, such as model, engine size, fuel type, year, and mileage. Supervised learning models that Nsiah tried included simple linear regression, polynomial regression, and random forest, evaluated using mean absolute error (MAE) and R-squared metrics. He concluded that out of the supervised models, random forest performed best with an R-squared of 0.99849 and a MAE of 289.0691. For unsupervised learning techniques, Nsiah applied K-Means and DBSCAN clustering to identify price patterns, evaluated using the Davis Boudlin Index and the Silhouette Coefficient. He concluded that K-Means clustering for the year of manufacture vs price produced the best clustering results.

Overall, these three studies demonstrate the effectiveness that machinie learning can have on accurately predicting used car prices. The next section will outline our own approach and findings.

Citations:

- C. Jin, "Price Prediction of Used Cars Using Machine Learning," in 2021 IEEE International Conference on Emergency Science and Information Technology (ICESIT), Chongqing, China, 2021, pp. 223-230, doi: 10.1109/ICESIT53460.2021.9696839.
- M. Hankar, M. Birjali, and A. Beni-Hssane, "Used Car Price Prediction using Machine Learning: A Case Study," in 2022 11th International Symposium on Signal, Image, Video and Communications (ISIVC), El Jadida, Morocco, 2022, pp. 1-4, doi: 10.1109/ISIVC54825.2022.9800719.
- T. Nsiah, "Car Price Prediction using Supervised and Unsupervised Learning Models and Deep Learning," unpublished, 2024.

Data Processing and Summary Statistics

First, we will import the dataset and libraries into our workspace

```r
library(caret)
```

```
## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.2.3

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 4.2.3
```

```r
library(ggplot2)
library(MASS)
library(randomForest)
```

```
## randomForest 4.7-1.1

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

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(kernlab)
```

```
##
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```r
library(stringr)
```

```
## Warning: package 'stringr' was built under R version 4.2.3
```

```r
library(cluster)
```

```
## Warning: package 'cluster' was built under R version 4.2.3
```

```r
library(glmnet)
```

```
## Loading required package: Matrix

## Warning: package 'Matrix' was built under R version 4.2.3

## Loaded glmnet 4.1-8
```

```r
library(stats)
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.2.3

##
## Attaching package: 'dplyr'

## The following object is masked from 'package:randomForest':
##
##     combine

## The following object is masked from 'package:MASS':
##
##     select

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

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

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

data <- read.csv("used_cars.csv")
```

#Preliminary Data Cleaning/Modifications First, we will removed the dollar sign and comma in price to enable numeric operations

```r
data$price <- as.numeric(gsub("[$,]", "", data$price))
```

Corrected the spelling of mileage from milage to mileage. Removed mi. and , to enable numeric operations. Renamed transmission to transmission_ for readability of one hot encoded dummy variables later.

```r
colnames(data)[colnames(data) == "milage"] <- "mileage"
data$mileage <- as.numeric(gsub("[,]| mi\\.", "", data$mileage))
```

The Engine columns contains very useful information such as the horsepower, displacement, cylinders, engine type, and fuel type. We turn these all into new columns.

```r
# Extract Horsepower (HP)
data$horsepower <- as.numeric(str_extract(data$engine, "\\d+\\.\\d+(?=HP)"))

# Extract Displacement
data$displacement <- as.numeric(str_extract(data$engine, "\\d+\\.\\d+(?=L)"))

# Extract Cylinders
data$cylinders <- str_extract(data$engine, "\\d+ Cylinder")
#data$cylinders_factor <- factor(str_extract(data$cylinders, "\\d+"))

# Extract Engine Type
data$engine_type <- str_extract(data$engine, "DOHC|SOHC|Turbo|Twin Turbo|Electric Motor")

# Extract Fuel Type
data$fuel_type <- str_extract(data$engine, "Gasoline|Diesel|Electric|Hybrid|Flex Fuel|Plug-In Electric/(
#data$fuel_type_factor <- factor(data$fuel_type)

#we are done with engine column since we have extracted all the information out
data$engine = NULL
head(data)
```

```
##      brand                           model model_year mileage fuel_type
## 1     Ford Utility Police Interceptor Base       2013   51000 Flex Fuel
## 2  Hyundai                    Palisade SEL       2021   34742      <NA>
## 3    Lexus                   RX 350 RX 350       2022   22372      <NA>
## 4 INFINITI                Q50 Hybrid Sport       2015   88900  Electric
## 5     Audi       Q3 45 S line Premium Plus       2021    9835      <NA>
## 6    Acura                        ILX 2.4L       2016  136397      <NA>
##        transmission                ext_col int_col
## 1       6-Speed A/T                  Black   Black
## 2 8-Speed Automatic        Moonlight Cloud    Gray
## 3         Automatic                   Blue   Black
## 4       7-Speed A/T                  Black   Black
## 5 8-Speed Automatic Glacier White Metallic   Black
## 6                 F                 Silver  Ebony.
##                                 accident clean_title price horsepower
## 1 At least 1 accident or damage reported         Yes 10300        300
## 2 At least 1 accident or damage reported         Yes 38005         NA
## 3                          None reported             54598         NA
## 4                          None reported         Yes 15500        354
## 5                          None reported             34999         NA
## 6                          None reported             14798         NA
##   displacement  cylinders engine_type
## 1          3.7 6 Cylinder        <NA>
## 2          3.8       <NA>        DOHC
## 3           NA       <NA>        DOHC
## 4          3.5 6 Cylinder        <NA>
## 5          2.0       <NA>        DOHC
## 6           NA       <NA>        <NA>
```

Looking at each column's type and unique count

```r
sapply(data, class)
```

```
##        brand        model   model_year      mileage    fuel_type transmission
##  "character"  "character"    "integer"    "numeric"  "character"  "character"
##      ext_col      int_col     accident  clean_title        price   horsepower
##  "character"  "character"  "character"  "character"    "numeric"    "numeric"
## displacement    cylinders  engine_type
##    "numeric"  "character"  "character"
```

```r
sapply(data, function(col) {
  if (is.character(col)) {
    length(unique(col))
  } else {
    NA  # Return NA for non-character columns
  }
})
```

```
##        brand        model   model_year      mileage    fuel_type transmission
##           57         1898           NA           NA            7           62
##      ext_col      int_col     accident  clean_title        price   horsepower
##          319          156            3            2           NA           NA
## displacement    cylinders  engine_type
##           NA            8            6
```

Let's examine columns that include NA or Empty String entries.

```r
na_columns <- colSums(is.na(data)) > 0
empty_string_columns <- colSums(data == "") > 0
columns_with_na_or_empty <- na_columns | empty_string_columns
print(names(data)[columns_with_na_or_empty])
```

```
## [1] "fuel_type"    "accident"     "clean_title"  "horsepower"   "displacement"
## [6] "cylinders"    "engine_type"
```

#Analyzing categorical variables Categorical variables with various unique values include brand, model, transmission, ext_col, int_col. Let's examine all of them

First, we look at the "brand" and the "model" columns. Through analysis shown below, we have decided to omit both of these columns. Our reasoning and visualizations are shown below.

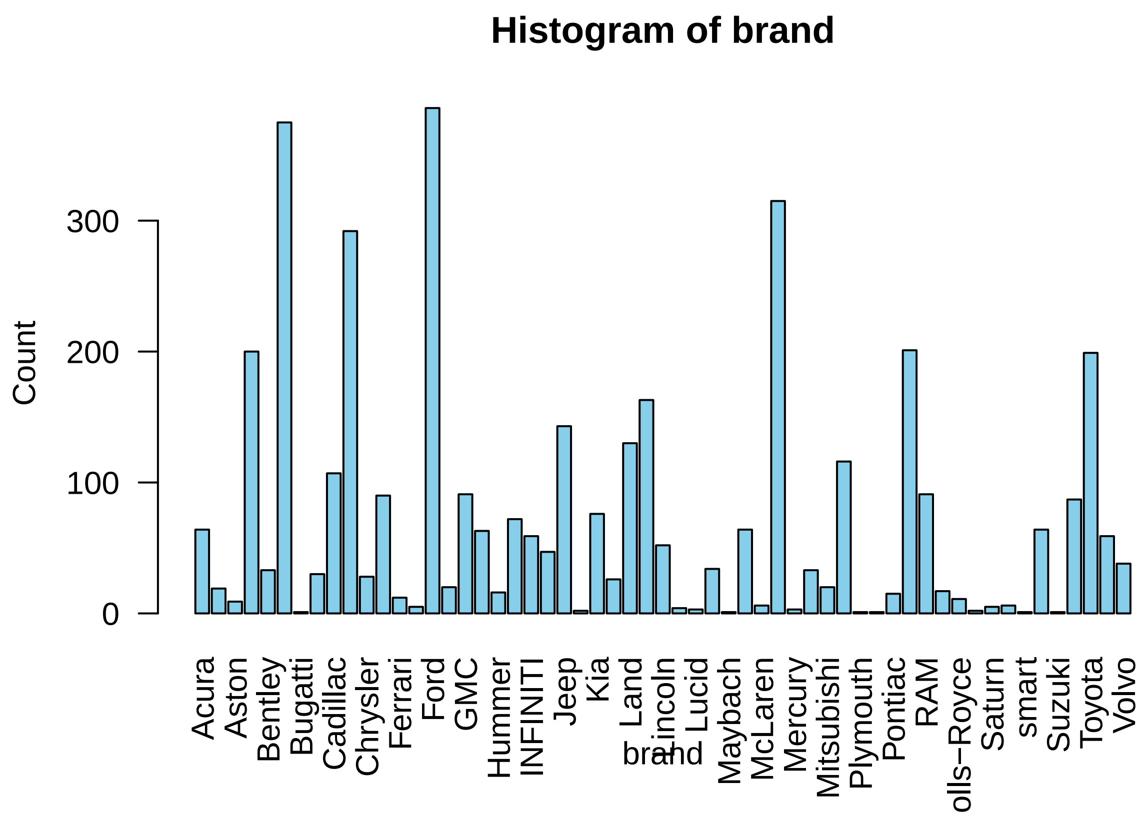
There are 57 unique brands with the frequency histogram not showing much dominance in a certain brand. To reduce the dimensionality, we will just omit this column

```r
length(unique(data$brand))
```

```
## [1] 57
```

```r
# calculate the counts for brand
brandcounts <- table(data$brand)
barplot(brandcounts,
  main = "Histogram of brand",
```

```r
  xlab = "brand",
  ylab = "Count",
  col = "skyblue",
  las = 2)
```



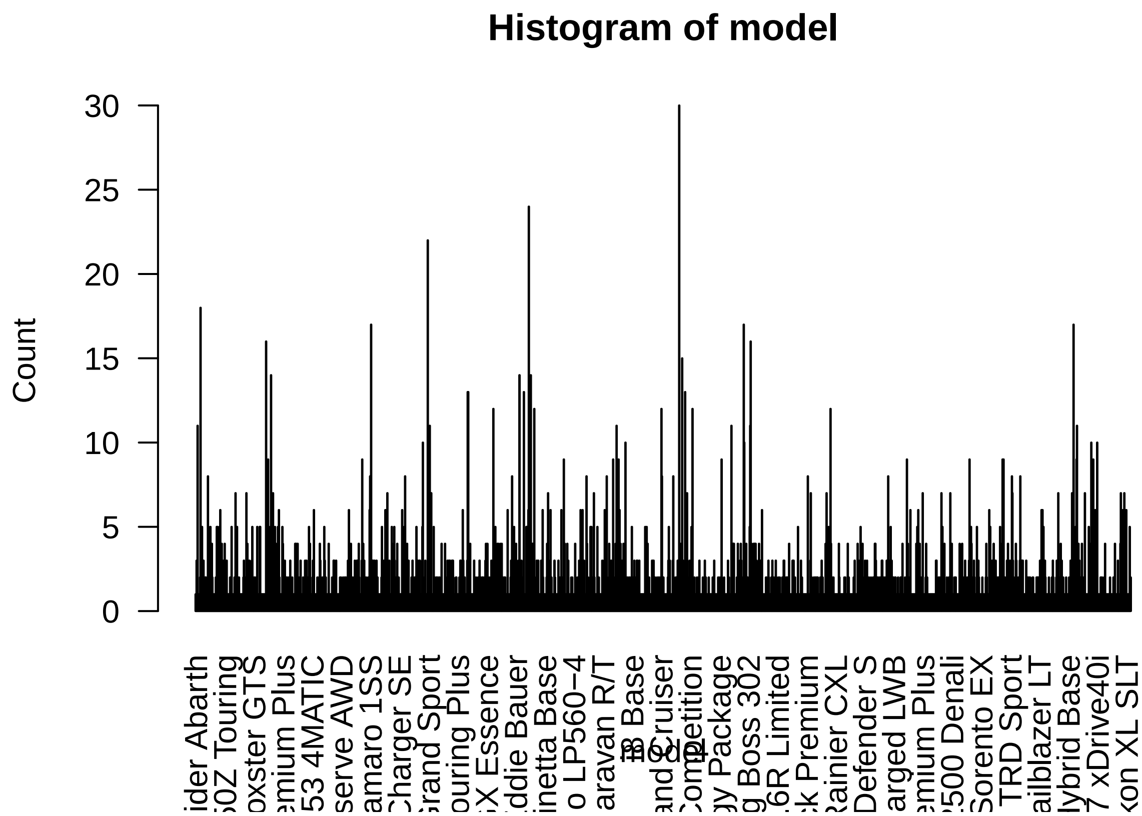

```r
#omit this column
data$brand = NULL
```

This problem is seen even more in the model column. We also omit this column from the dataset

```r
length(unique(data$model))
```

```
## [1] 1898
```

```r
modelcounts <- table(data$model)
barplot(modelcounts,
  main = "Histogram of model",
  xlab = "model",
  ylab = "Count",
  col = "skyblue",
  las = 2)
```




```r
#omit this columm
data$model = NULL
```

Now, let's examine colors. There are both intcol and extcol variables. Having too many unique color names can introduce noise into your classification model and make it harder for the model to generalize effectively. Grouping the colors into broader, more general categories can help improve model performance by reducing the dimensionality of the feature and making patterns more apparent.

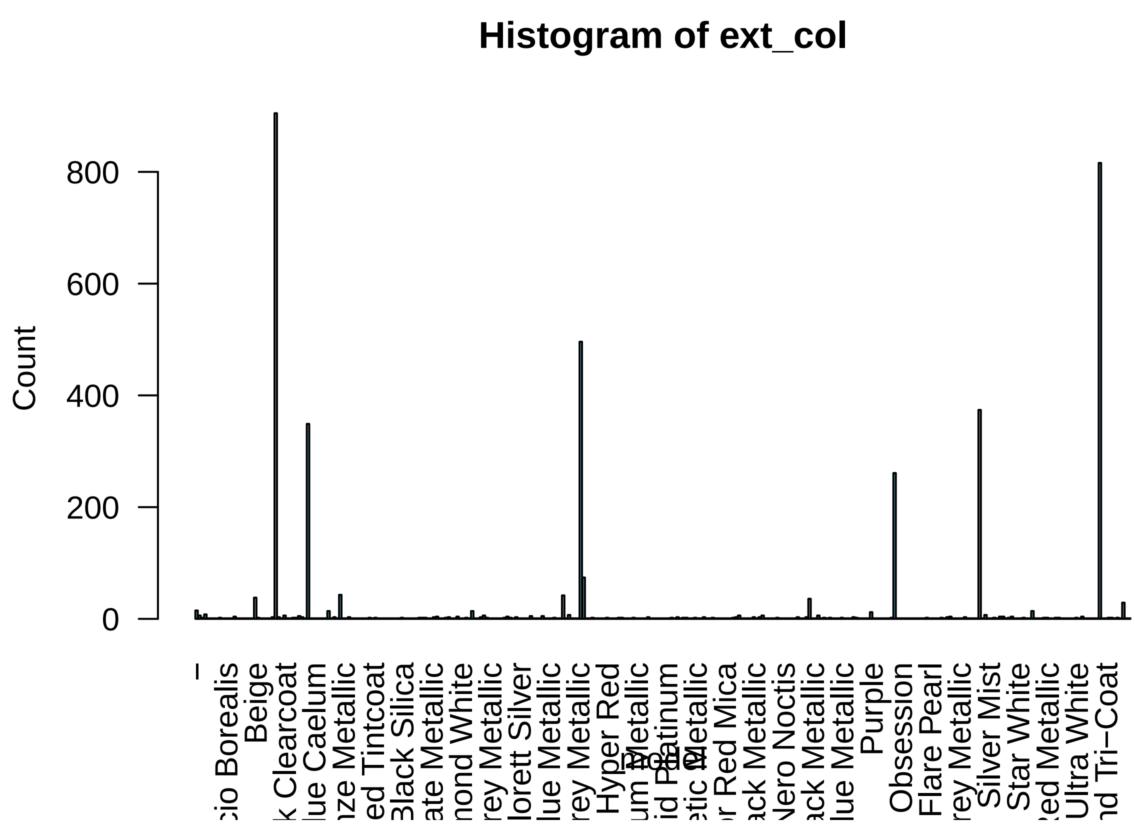
```r
# Define the mapping function
generalize_colors <- function(color_name) {
  # Convert to lowercase for uniformity
  color_lower <- tolower(color_name)

  # Define patterns for each general category
  if (str_detect(color_lower, "black")) {
    return("Black")
  } else if (str_detect(color_lower, "white|ivory|platinum")) {
    return("White")
  } else if (str_detect(color_lower, "gray|grey|silver|slate|charcoal|mica|metallic|graphite")) {
    return("Gray")
  } else if (str_detect(color_lower, "brown|beige|tan|camel|mocha|walnut|chestnut|saddle|cappuccino|coc
    return("Brown")
  } else if (str_detect(color_lower, "silver")) {
    return("Silver")
  } else if (str_detect(color_lower, "gold")) {
    return("Gold")
```

```r
  } else {
    return("Other")  # For colors that don't match any category
  }
}
```

```r
length(unique(data$ext_col))
```

```
## [1] 319
```

```r
extcolorcounts <- table(data$ext_col)
barplot(extcolorcounts,
  main = "Histogram of ext_col",
  xlab = "model",
  ylab = "Count",
  col = "skyblue",
  las = 2)
```



Let's apply the generalization function to simply the different colors

```r
data$ext_col <- sapply(data$ext_col, generalize_colors)
unique(data$ext_col)
```

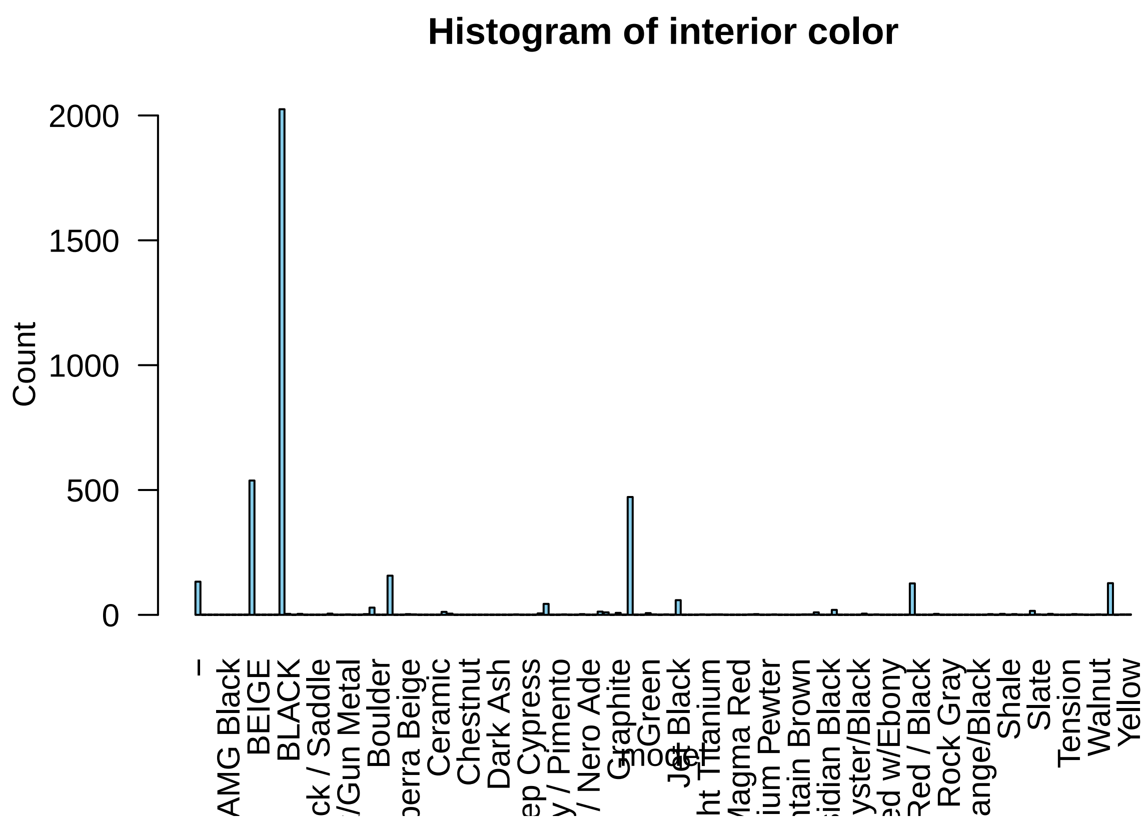
```
## [1] "Black" "Other" "White" "Gray"  "Gold"  "Brown"
```

The same thing happens to int_col, but looking at the dataset we will have 4 categories.

```r
length(unique(data$int_col))
```

```
## [1] 156
```

```r
intcolorcounts <- table(data$int_col)
barplot(intcolorcounts,
  main = "Histogram of interior color",
  xlab = "model",
  ylab = "Count",
  col = "skyblue",
  las = 2)
```




```r
# Grouping less frequent categories
data$int_col <- ifelse(
  data$int_col %in% c("Black", "Jet Black", "AMG Black"),
  "Black",
  ifelse(data$int_col %in% c("Beige", "Ivory"), "Beige/Ivory",
         ifelse(data$int_col %in% c("Gray", "Graphite"), "Gray",
                "Other"))
)

unique(data$int_col)
```

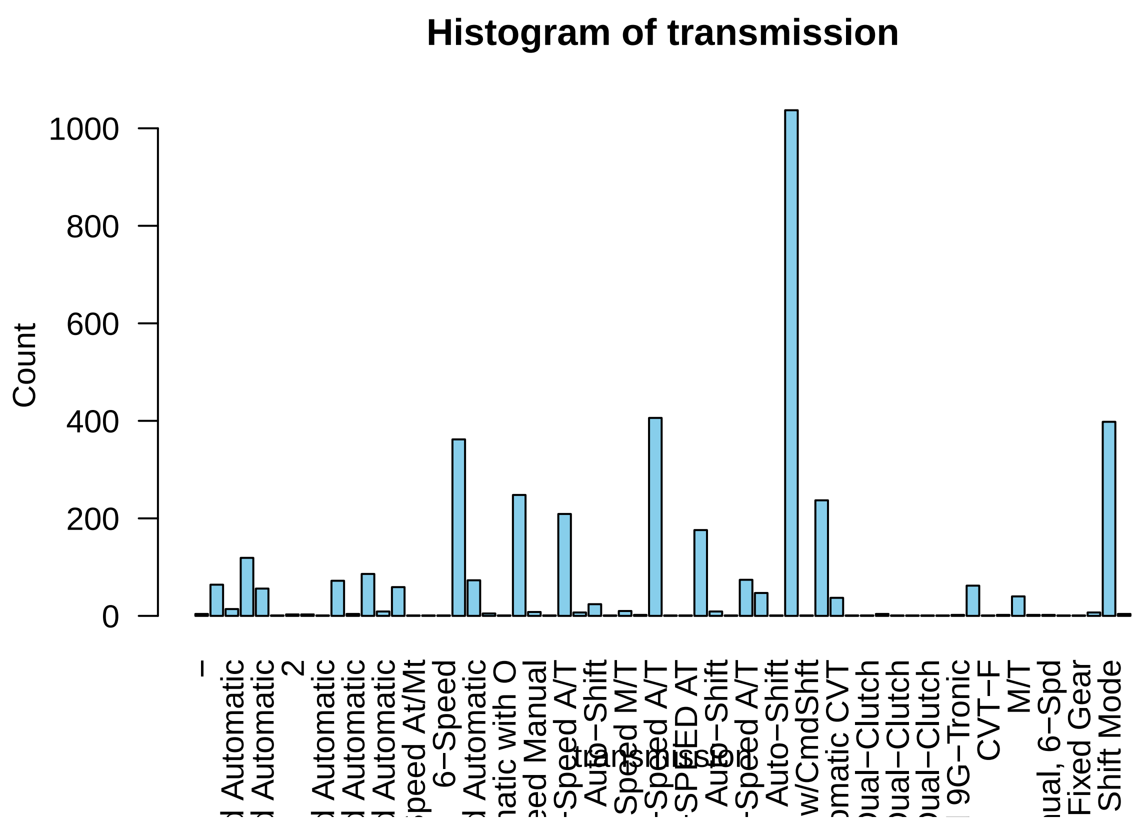
```
## [1] "Black"       "Gray"        "Other"       "Beige/Ivory"
```

Examining the transmission column now

```r
length(unique(data$transmission))
```

```
## [1] 62
```

```r
# calculate the counts for transmission
trancounts <- table(data$transmission)

barplot(trancounts,
  main = "Histogram of transmission",
  xlab = "transmission",
  ylab = "Count",
  col = "skyblue",
  las = 2)
```




```r
result <- data %>%
  group_by(transmission) %>%
  summarise(
    medianprice = median(price),
    averageprice = mean(price),
    count = n()
  )
result_sorted <- result %>%
  arrange(desc(count))
print(result_sorted)
```

```
## # A tibble: 62 x 4
##    transmission                      medianprice averageprice count
##    <chr>                                   <dbl>        <dbl> <int>
##  1 A/T                                     20500       31508.  1037
##  2 8-Speed A/T                             39625       51126.   406
##  3 Transmission w/Dual Shift Mode          34000       54711.   398
##  4 6-Speed A/T                             20900       25450.   362
##  5 6-Speed M/T                             26450       39282.   248
##  6 Automatic                               47541       63105.   237
##  7 7-Speed A/T                             32999       47250.   209
##  8 8-Speed Automatic                       41599       66072.   176
##  9 10-Speed A/T                            57000       60915.   119
## 10 5-Speed A/T                             15000       17607.    86
## # i 52 more rows
```

```r
threshold <- quantile(result_sorted$count, 0.9) # 205.7
significant_transmissions <- result_sorted$transmission[result_sorted$count > threshold]
print(significant_transmissions)
```

```
## [1] "A/T"                            "8-Speed A/T"
## [3] "Transmission w/Dual Shift Mode" "6-Speed A/T"
## [5] "6-Speed M/T"                    "Automatic"
## [7] "7-Speed A/T"
```

Now, we have looked at all the categorical variables with many many unique values, we will now one-hot encode the cagtegorical vairables. # One-hot encoding categorical variables

After looking at histograms for both Brand and Transmission, it seems Brand is more uniformly distributed while Transmission has a few salient categories. After exploring the categories of transmissions we found that the top 7 most frequent transmissions account for approximately 67-70% of the data points. Therefore we will one hot encode these 7 categories + an "Other" category for Transmission for a total of 8 transmission categories. We will also one hot encode "fuel type" and "cylinders" since those are categorical variables as well.

Transmission

```r
# map transmissions to just the top 7 or Other. could make this function take in significant_transmissi
map_transmission <- function(transmission) {
  primary_transmissions <- c(
    "A/T",
    "8-Speed A/T",
    "Transmission w/Dual Shift Mode",
    "6-Speed A/T",
    "6-Speed M/T",
    "Automatic",
    "7-Speed A/T"
  )

  if (transmission %in% primary_transmissions) {
    return(transmission)
  } else {
    return("Other")
```

```
  }
}

# Apply the mapping function
data$transmission <- sapply(data$transmission, map_transmission)
```

#Analyzing Null/Empty Values We will first look at the problem with NA and Empty values, something that this dataset has a lot of. We will first handle both NA and Empty " " values by replacing them to "NA" to make it easier to preprocess and analyze.

```
na_columns <- colSums(is.na(data)) > 0
empty_string_columns <- colSums(data == "") > 0
columns_with_na_or_empty <- na_columns | empty_string_columns
print(names(data)[columns_with_na_or_empty])
```

```
## [1] "fuel_type"    "accident"     "clean_title"  "horsepower"   "displacement"
## [6] "cylinders"    "engine_type"
```

```
# data[data == "" | is.na(data)] <- "NA"
# summary(data)
# unique(data$horsepower)
# unique(data$displacement)
# unique(data$cylinders_numeric)
unique(data$fuel_type_factor)
```

```
## NULL
```

```
unique(data$fuel_type_numeric)
```

```
## NULL
```

```
unique(data$horsepower)
```

```
##   [1] 300  NA 354 292 282 311 534 715 382 400 375 305 287 550 120
##  [16] 355 276 445 362 345 383 180 211 173 240 552 536 310 228 268
##  [31] 503 325 208 250 200 420 302 306 237 248 425 582 444 335 424
##  [46] 340 225 365 315 199 560 326 165 835 241 215 130 288 369 195
##  [61] 285 485 132 416 360 280 620 265 469 169 330 275 303 450 651
##  [76] 255 455 182 236 370 212 565 230 171 252 220 188 235 320 138
##  [91] 291 523 440 181 429 263 210 404 670 563 283 150 266 328 304
## [106] 381 493 641 760 329 239 160 402 166 390 147 357 271 350 611
## [121] 295 603 454 490 301 395 272 437 323 256 140 600 409 640 204
## [136] 316 591 219 505 403 170 115 562 106 201 496 475 184 407 543
## [151] 333 553 471 380 247 349 190 410 260 245 332 261 107 577 290
## [166] 453 293 139 389 567 221 518 630 218 385 174 134 273 172 542
## [181] 571 601 500 270 161 394 520 164 205 308 226 227 412 158 414
## [196] 177 346 111 573 277 191 318 411 244 605 192 207 155 189 185
## [211] 162 187 313 557 281 463 186 797 214 449 153 296 650 759 286
## [226] 525 246 526 397 645 575 401 348 510 122 179 167 691 202 136
## [241] 151 617 146 294 317 175 717 435 405 616 137 152 206 415 460
```

```
## [256]  707  319  426  555  480  121  430  159  378  321  344  133  232  142  278
## [271]   78  258  264  118   76  788  131  148  203  253  312  467  168  156  353
## [286]  545  422  451  197  386  778  521  495  621  456  279  540  104  372  366
## [301]  284  556  193  393  198  298  145  242  243   70  610  141  217  533  262
## [316]  342  483  109  231  473  324  443  101  322  126  638  710  154  808  143
## [331]  602  363  178  580  624  379  502  470 1020  572  702  660  341  222  729
## [346]  417  482  224  176
```

```r
unique(data$displacement)
```

```
##  [1] 3.70 3.80   NA 3.50 2.00 4.40 5.20 3.00 5.00 3.60 2.20 5.30 5.70 2.40 2.70
## [16] 6.00 4.00 1.50 6.10 1.60 2.90 3.30 3.40 2.50 1.80 6.20 4.30 6.75 5.50 5.60
## [31] 6.30 5.40 6.70 4.60 4.50 4.70 1.30 2.30 3.20 5.80 6.80 6.40 8.00 4.20 1.20
## [46] 3.90 1.70 7.00 2.80 6.60 1.40 4.80 7.40 5.90 8.10 6.50 8.40 0.65 8.30 2.10
## [61] 7.30 1.00
```

```r
unique(data$cylinders_factor)
```

```
## NULL
```

```r
unique(data$cylinders_numeric)
```

```
## NULL
```

```r
sum(is.na(data$horsepower))
```

```
## [1] 810
```

```r
sum(is.na(data$displacement))
```

```
## [1] 396
```

```r
table(data$displacement)
```

```
## 
## 0.65    1  1.2  1.3  1.4  1.5  1.6  1.7  1.8    2  2.1  2.2  2.3  2.4  2.5  2.7 
##    5    1    3    8   16   38   59    1   46  471    2    5   35   99  175   46 
##  2.8  2.9    3  3.2  3.3  3.4  3.5  3.6  3.7  3.8  3.9    4  4.2  4.3  4.4  4.5 
##    5   16  432   31   26   30  333  235   62  105   15  182   26   15   82    2 
##  4.6  4.7  4.8    5  5.2  5.3  5.4  5.5  5.6  5.7  5.8  5.9    6  6.1  6.2  6.3 
##   70   54   28  112   29  104   23   28   35  129    3    4   67    4  173    6 
##  6.4  6.5  6.6  6.7 6.75  6.8    7  7.3  7.4    8  8.1  8.3  8.4 
##   28    7   26   52    2    7    3    4    1    1    2    3    1
```

```r
summary(data)
```

```
##    model_year      mileage        fuel_type         transmission
##  Min.   :1974   Min.   :   100   Length:4009        Length:4009
##  1st Qu.:2012   1st Qu.: 23044   Class :character   Class :character
##  Median :2017   Median : 52775   Mode  :character   Mode  :character
##  Mean   :2016   Mean   : 64718
##  3rd Qu.:2020   3rd Qu.: 94100
##  Max.   :2024   Max.   :405000
##
##    ext_col            int_col            accident          clean_title
##  Length:4009        Length:4009        Length:4009        Length:4009
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##      price           horsepower      displacement    cylinders
##  Min.   :   2000   Min.   :  70.0   Min.   :0.650   Length:4009
##  1st Qu.:  17200   1st Qu.: 248.0   1st Qu.:2.500   Class :character
##  Median :  31000   Median : 310.0   Median :3.500   Mode  :character
##  Mean   :  44553   Mean   : 332.3   Mean   :3.711
##  3rd Qu.:  49990   3rd Qu.: 400.0   3rd Qu.:4.700
##  Max.   :2954083   Max.   :1020.0   Max.   :8.400
##                    NA's   :810      NA's   :396
##  engine_type
##  Length:4009
##  Class :character
##  Mode  :character
##
##
##
##
```

There are five columns with empty strings/NA values. Let's examine all five of them to discover if we can find any patterns.

horsepower

```r
# number of unique values in horsepower
length(table(data$horsepower))
```

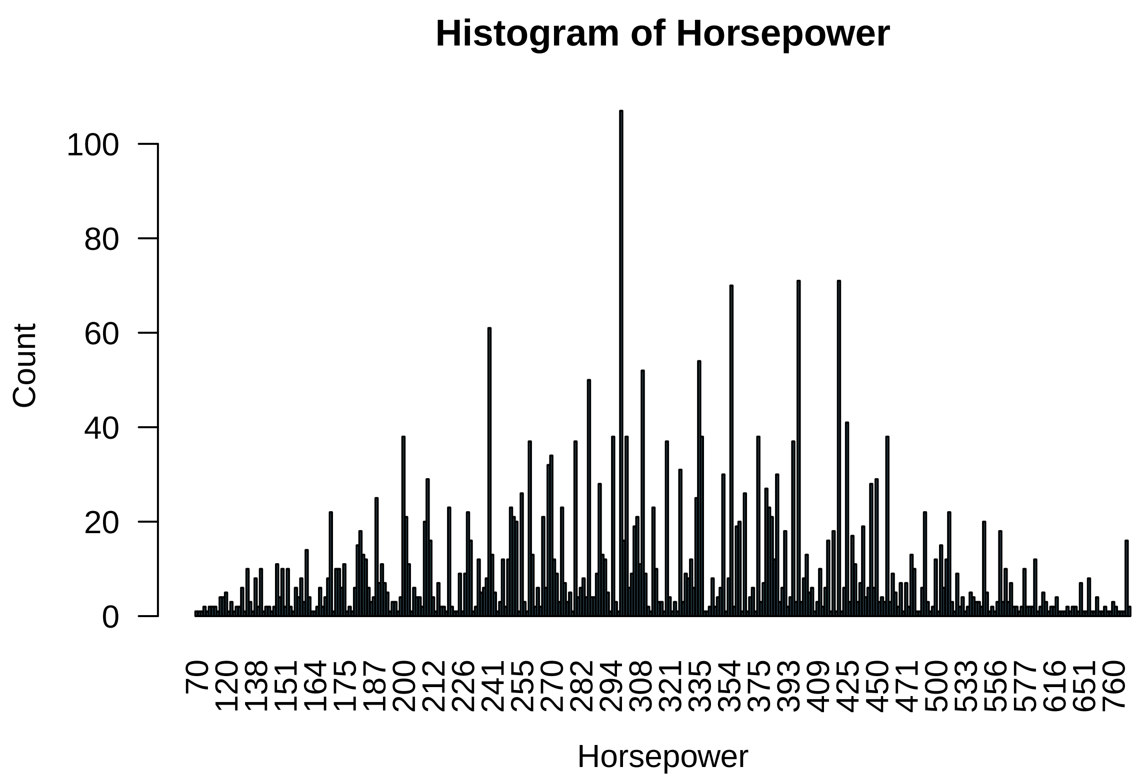
```
## [1] 348
```

```r
# number of null values in horsepower
sum(is.na(data$horsepower))
```

```
## [1] 810
```

```r
# calculate the counts for horsepower
horsepower_counts <- table(data$horsepower)
```

```r
barplot(horsepower_counts,
  main = "Histogram of Horsepower",
  xlab = "Horsepower",
  ylab = "Count",
  col = "skyblue",
  las = 2)
```




```r
# median imputation
data$horsepower[is.na(data$horsepower)] <- median(data$horsepower, na.rm = TRUE)

sum(is.na(data$horsepower))
```

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

Since there are 348 unique values in horsepower, we can consider horsepower as a continuous variable rather than categorical. However, there are 810 null values in a dataset with 4009 entries which is over 20% null values. This is too many to simply drop, so we want to perform some form of imputation. Looking at the distribution of horsepowers, we can see that the median is a good representative approximation for the distribution so we will use **median imputation**.

displacement (engine size)

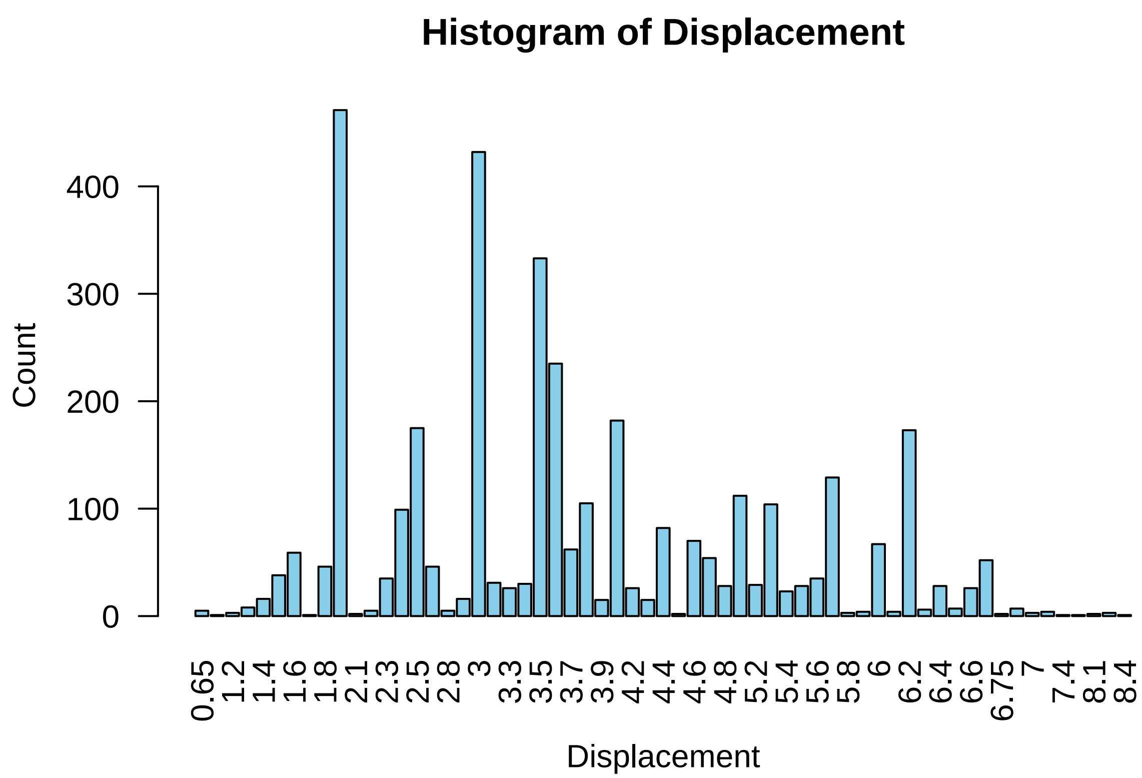
```r
# number of unique values in displacement
# table(data$displacement)
length(table(data$displacement))
```

```
## [1] 61
```

```r
# number of null values in displacement
sum(is.na(data$displacement))
```

```
## [1] 396
```

```r
# calculate the counts for horsepower
displacement_counts <- table(data$displacement)

barplot(displacement_counts,
  main = "Histogram of Displacement",
  xlab = "Displacement",
  ylab = "Count",
  col = "skyblue",
  las = 2)
```




```r
# median imputation
data$displacement[is.na(data$displacement)] <- median(data$displacement, na.rm = TRUE)
```

```r
sum(is.na(data$displacement))
```

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

There are 61 unique values in displacement (engine size). Although these appear to be discretized measurements (ex: size = 0.8 or size = 3.71 may not make sense), we can treat it as a more continuous predictor for now. There are 396 null values in displacement which is just under 10% null values, so we could consider dropping these. However since the median already exists in the dataset (median = 3.5) we can also proceed with median imputation which is what we did.

```r
result <- data %>%
  group_by(fuel_type) %>%
  summarise(
    medianprice = median(price),
    averageprice = mean(price),
    count = n()
  )
print(result)
```

```
## # A tibble: 7 x 4
##   fuel_type            medianprice averageprice count
##   <chr>                      <dbl>        <dbl> <int>
## 1 Diesel                     45450       48878.   114
## 2 Electric                   42000.      46884.   238
## 3 Flex Fuel                  18650       22156.   128
## 4 Gasoline                   27000       38733.  2731
## 5 Hybrid                     37999       45063.    17
## 6 Plug-In Electric/Gas       44945       45946.    34
## 7 <NA>                       41599       68192.   747
```

```r
data$fuel_type[is.na(data$fuel_type)] <- "NA"
```

The NA values for fuel_type have a higher median price and average price than other types, and makes up a significant count of observations so we are going to treat it as a separate category.

#cylinder

```r
result <- data %>%
  group_by(cylinders) %>%
  summarise(
    medianprice = median(price),
    averageprice = mean(price),
    count = n()
  )
print(result)
```

```
## # A tibble: 8 x 4
##   cylinders   medianprice averageprice count
##   <chr>             <dbl>        <dbl> <int>
## 1 10 Cylinder      100000       166530.    23
## 2 12 Cylinder       81330       140259.    37
```

```
## 3 3 Cylinder       32000        45281.    13
## 4 4 Cylinder       19000        22476.   739
## 5 5 Cylinder       10150.       18584.    20
## 6 6 Cylinder       27999        35935.  1225
## 7 8 Cylinder       34500        46401.  1007
## 8 <NA>             42599        64844.   945
```

```
data$cylinders[is.na(data$cylinders)] <- "NA"
```

#accident

```
result <- data %>%
  group_by(accident) %>%
  summarise(
    medianprice = median(price),
    averageprice = mean(price),
    count = n()
  )
print(result)
```

```
## # A tibble: 3 x 4
##   accident                                   medianprice averageprice count
##   <chr>                                            <dbl>        <dbl> <int>
## 1 ""                                               36500        50788.   113
## 2 "At least 1 accident or damage reported"         20900        28832.   986
## 3 "None reported"                                  35668.       49638.  2910
```

The NA/Empty values for accident exhibit very similar properties to the None reported category, with median price and average price being pretty similar, not to mention a very small percentage of data is represented by this value. Therefore, we replace and combine these observations with the None reported category. Because accident only has 2 unique values now, no accidents and 1 or more accidents, we changed it to 1,0 to be useful for models.

```
data$accident[data$accident == "NA"] <- "None reported"
#unique(data$accident)
data$accident <- ifelse(data$accident == "At least 1 accident or damage reported", 1, 0)
# unique(data$accident)
```

```
result <- data %>%
  group_by(clean_title) %>%
  summarise(
    medianprice = median(price),
    averageprice = mean(price),
    count = n()
  )
print(result)
```

```
## # A tibble: 2 x 4
##   clean_title medianprice averageprice count
##   <chr>             <dbl>        <dbl> <int>
## 1 ""               42996.       60695.   596
## 2 "Yes"            29000        41734.  3413
```

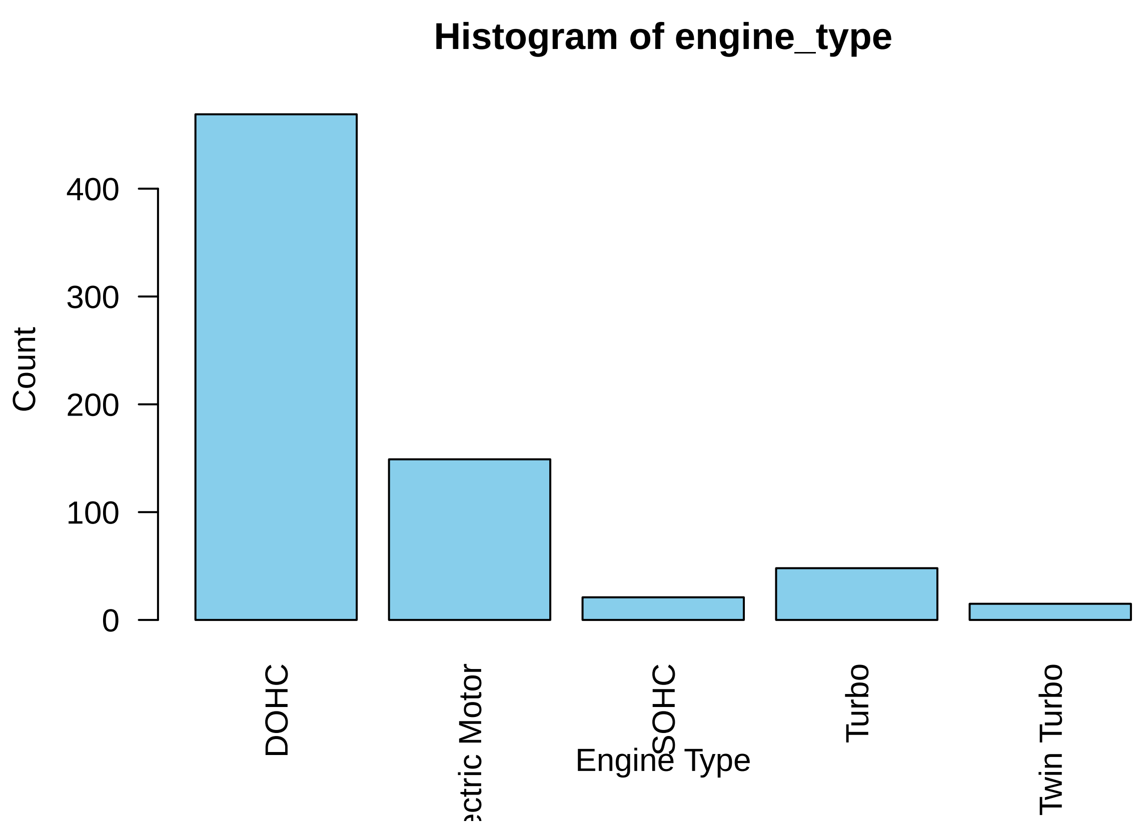
The NA values for clean_title clearly have a significantly higher median price and will be treated as a separate category. We apply similar reasoning from accident to clean_title. Since there is only "Yes" and NA, we treat all the yes's to 1 and all the NA values to 0.

```
data$clean_title <-ifelse(data$clean_title == "Yes", 1, 0)
unique(data$clean_title)
```

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

#engine type

```
# calculate the counts for horsepower
engine_counts <- table(data$engine_type)

barplot(engine_counts,
  main = "Histogram of engine_type",
  xlab = "Engine Type",
  ylab = "Count",
  col = "skyblue",
  las = 2)
```




```
result <- data %>%
  group_by(engine_type) %>%
  summarise(
    medianprice = median(price),
```

```r
    averageprice = mean(price),
    count = n()
  )
print(result)
```

```
## # A tibble: 6 x 4
##   engine_type    medianprice averageprice count
##   <chr>                <dbl>        <dbl> <int>
## 1 DOHC                39244        77951.   469
## 2 Electric Motor      47800        54439.   149
## 3 SOHC                38998        38676.    21
## 4 Turbo               49940.       51767.    48
## 5 Twin Turbo          85998        89258.    15
## 6 <NA>                28250        39101.  3307
```

```r
data$engine_type[is.na(data$engine_type)] <- "NA"
```

#Removing Outliers We remove outliers with 1.5*IQR value.

```r
Q1 <- quantile(data$price, 0.25, na.rm = TRUE)
Q3 <- quantile(data$price, 0.75, na.rm = TRUE)
IQR_value <- IQR(data$price, na.rm = TRUE)

# Identify outliers using IQR
lower_bound <- Q1 - 1.5 * IQR_value
upper_bound <- Q3 + 1.5 * IQR_value

outliers <- data[data$price < lower_bound | data$price > upper_bound, ]
print(paste("Number of outliers: ", nrow(outliers), "and average price of these cars: ", round(mean(out
```

```
## [1] "Number of outliers:  244 and average price of these cars:  214826.76"
```

```r
#removing these rows from the dataset
data <- data[!(data$price < lower_bound | data$price > upper_bound), ]
```

```r
summary(data)
```

```
##    model_year      mileage        fuel_type         transmission
##  Min.   :1992   Min.   :   100   Length:3765        Length:3765
##  1st Qu.:2012   1st Qu.: 26600   Class :character   Class :character
##  Median :2017   Median : 57237   Mode  :character   Mode  :character
##  Mean   :2015   Mean   : 68075
##  3rd Qu.:2020   3rd Qu.: 97000
##  Max.   :2024   Max.   :405000
##    ext_col            int_col             accident       clean_title
##  Length:3765        Length:3765        Min.   :0.0000   Min.   :0.0000
##  Class :character   Class :character   1st Qu.:0.0000   1st Qu.:1.0000
##  Mode  :character   Mode  :character   Median :0.0000   Median :1.0000
##                                        Mean   :0.2595   Mean   :0.8608
##                                        3rd Qu.:1.0000   3rd Qu.:1.0000
##                                        Max.   :1.0000   Max.   :1.0000
```

```
##      price          horsepower       displacement     cylinders
##  Min.   : 2000   Min.   :  70.0   Min.   :0.650   Length:3765
##  1st Qu.:16500   1st Qu.: 263.0   1st Qu.:2.500   Class :character
##  Median :29600   Median : 310.0   Median :3.500   Mode  :character
##  Mean   :33518   Mean   : 320.9   Mean   :3.648
##  3rd Qu.:45500   3rd Qu.: 375.0   3rd Qu.:4.400
##  Max.   :99000   Max.   :1020.0   Max.   :8.300
##  engine_type
##  Length:3765
##  Class :character
##  Mode  :character
##
##
##
```

#turning each categorial column into a factor type

```r
data[sapply(data, is.character)] <- lapply(data[sapply(data, is.character)], as.factor)
```

#one hot encoding Some models will require one hot encoding. For these models, we create a new dataset and apply this one hot encoding

```r
dummy_model <- dummyVars(~ ., data = data)
data_one_hot <- as.data.frame(predict(dummy_model, newdata = data))
```

#Final Summary Statistics

```r
dim(data_one_hot)
```

```
## [1] 3765   46
```

```r
summary(data_one_hot)
```

```
##    model_year      mileage       fuel_type.Diesel  fuel_type.Electric
##  Min.   :1992   Min.   :   100   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:2012   1st Qu.: 26600   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :2017   Median : 57237   Median :0.00000   Median :0.00000
##  Mean   :2015   Mean   : 68075   Mean   :0.02895   Mean   :0.06135
##  3rd Qu.:2020   3rd Qu.: 97000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :2024   Max.   :405000   Max.   :1.00000   Max.   :1.00000
##  fuel_type.Flex Fuel fuel_type.Gasoline fuel_type.Hybrid   fuel_type.NA
##  Min.   :0.000       Min.   :0.0000     Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.000       1st Qu.:0.0000     1st Qu.:0.00000   1st Qu.:0.0000
##  Median :0.000       Median :1.0000     Median :0.00000   Median :0.0000
##  Mean   :0.034       Mean   :0.6887     Mean   :0.00425   Mean   :0.1737
##  3rd Qu.:0.000       3rd Qu.:1.0000     3rd Qu.:0.00000   3rd Qu.:0.0000
##  Max.   :1.000       Max.   :1.0000     Max.   :1.00000   Max.   :1.0000
##  fuel_type.Plug-In Electric/Gas transmission.6-Speed A/T
##  Min.   :0.00000                Min.   :0.00000
##  1st Qu.:0.00000                1st Qu.:0.00000
##  Median :0.00000                Median :0.00000
```

```
##  Mean   :0.00903               Mean   :0.09535
##  3rd Qu.:0.00000               3rd Qu.:0.00000
##  Max.   :1.00000               Max.   :1.00000
##  transmission.6-Speed M/T transmission.7-Speed A/T transmission.8-Speed A/T
##  Min.   :0.00000          Min.   :0.00000          Min.   :0.00000
##  1st Qu.:0.00000          1st Qu.:0.00000          1st Qu.:0.00000
##  Median :0.00000          Median :0.00000          Median :0.00000
##  Mean   :0.06348          Mean   :0.05206          Mean   :0.09854
##  3rd Qu.:0.00000          3rd Qu.:0.00000          3rd Qu.:0.00000
##  Max.   :1.00000          Max.   :1.00000          Max.   :1.00000
##  transmission.A/T transmission.Automatic transmission.Other
##  Min.   :0.0000   Min.   :0.00000       Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.00000       1st Qu.:0.0000
##  Median :0.0000   Median :0.00000       Median :0.0000
##  Mean   :0.2667   Mean   :0.05657       Mean   :0.2709
##  3rd Qu.:1.0000   3rd Qu.:0.00000       3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.00000       Max.   :1.0000
##  transmission.Transmission w/Dual Shift Mode ext_col.Black   ext_col.Brown
##  Min.   :0.00000                             Min.   :0.000   Min.   :0.00000
##  1st Qu.:0.00000                             1st Qu.:0.000   1st Qu.:0.00000
##  Median :0.00000                             Median :0.000   Median :0.00000
##  Mean   :0.09641                             Mean   :0.255   Mean   :0.02125
##  3rd Qu.:0.00000                             3rd Qu.:1.000   3rd Qu.:0.00000
##  Max.   :1.00000                             Max.   :1.000   Max.   :1.00000
##   ext_col.Gold      ext_col.Gray    ext_col.Other    ext_col.White
##  Min.   :0.00000   Min.   :0.0000   Min.   :0.0000   Min.   :0.000
##  1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000
##  Median :0.00000   Median :0.0000   Median :0.0000   Median :0.000
##  Mean   :0.01116   Mean   :0.2653   Mean   :0.2133   Mean   :0.234
##  3rd Qu.:0.00000   3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.000
##  Max.   :1.00000   Max.   :1.0000   Max.   :1.0000   Max.   :1.000
##  int_col.Beige/Ivory int_col.Black     int_col.Gray    int_col.Other
##  Min.   :0.0000      Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000      1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000      Median :1.0000   Median :0.0000   Median :0.0000
##  Mean   :0.1392      Mean   :0.5214   Mean   :0.1246   Mean   :0.2149
##  3rd Qu.:0.0000      3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.0000
##  Max.   :1.0000      Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##     accident       clean_title         price         horsepower
##  Min.   :0.0000   Min.   :0.0000   Min.   : 2000   Min.   :  70.0
##  1st Qu.:0.0000   1st Qu.:1.0000   1st Qu.:16500   1st Qu.: 263.0
##  Median :0.0000   Median :1.0000   Median :29600   Median : 310.0
##  Mean   :0.2595   Mean   :0.8608   Mean   :33518   Mean   : 320.9
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:45500   3rd Qu.: 375.0
##  Max.   :1.0000   Max.   :1.0000   Max.   :99000   Max.   :1020.0
##   displacement   cylinders.10 Cylinder cylinders.12 Cylinder
##  Min.   :0.650   Min.   :0.000000      Min.   :0.000000
##  1st Qu.:2.500   1st Qu.:0.000000      1st Qu.:0.000000
##  Median :3.500   Median :0.000000      Median :0.000000
##  Mean   :3.648   Mean   :0.002922      Mean   :0.005578
##  3rd Qu.:4.400   3rd Qu.:0.000000      3rd Qu.:0.000000
##  Max.   :8.300   Max.   :1.000000      Max.   :1.000000
##  cylinders.3 Cylinder cylinders.4 Cylinder cylinders.5 Cylinder
##  Min.   :0.000000     Min.   :0.0000       Min.   :0.000000
```

```
##  1st Qu.:0.000000     1st Qu.:0.0000       1st Qu.:0.000000
##  Median :0.000000     Median :0.0000       Median :0.000000
##  Mean   :0.003453     Mean   :0.1958       Mean   :0.005312
##  3rd Qu.:0.000000     3rd Qu.:0.0000       3rd Qu.:0.000000
##  Max.   :1.000000     Max.   :1.0000       Max.   :1.000000
##  cylinders.6 Cylinder cylinders.8 Cylinder  cylinders.NA    engine_type.DOHC
##  Min.   :0.0000       Min.   :0.000        Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000       1st Qu.:0.000        1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000       Median :0.000        Median :0.0000   Median :0.0000
##  Mean   :0.3118       Mean   :0.251        Mean   :0.2242   Mean   :0.1039
##  3rd Qu.:1.0000       3rd Qu.:1.000        3rd Qu.:0.0000   3rd Qu.:0.0000
##  Max.   :1.0000       Max.   :1.000        Max.   :1.0000   Max.   :1.0000
##  engine_type.Electric Motor engine_type.NA   engine_type.SOHC
##  Min.   :0.00000            Min.   :0.0000   Min.   :0.000000
##  1st Qu.:0.00000            1st Qu.:1.0000   1st Qu.:0.000000
##  Median :0.00000            Median :1.0000   Median :0.000000
##  Mean   :0.03825            Mean   :0.8369   Mean   :0.005578
##  3rd Qu.:0.00000            3rd Qu.:1.0000   3rd Qu.:0.000000
##  Max.   :1.00000            Max.   :1.0000   Max.   :1.000000
##  engine_type.Turbo engine_type.Twin Turbo
##  Min.   :0.00000   Min.   :0.000000
##  1st Qu.:0.00000   1st Qu.:0.000000
##  Median :0.00000   Median :0.000000
##  Mean   :0.01275   Mean   :0.002656
##  3rd Qu.:0.00000   3rd Qu.:0.000000
##  Max.   :1.00000   Max.   :1.000000
```

Unsupervised Learning

Apply at least three clustering algorithms to the processed dataset. Determine the appropriate number of clusters and discuss the interpretability of these clusters. Do they hold any meaningful distinctions? Examine whether the clustering results are associated with your outcome variable.

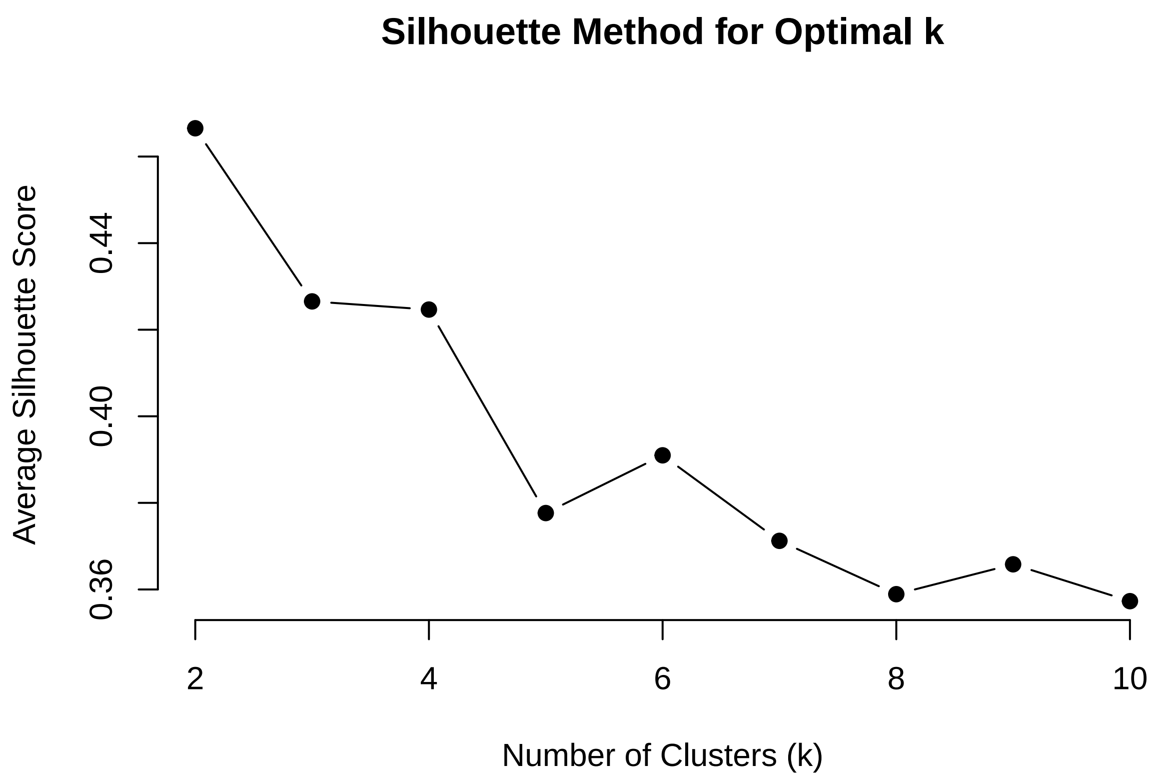
1. KMeans Clustering

```r
data_subset <- data[, c("model_year", "price")]
data_subset <- na.omit(data_subset)
data_subset_scaled <- scale(data_subset)
```

We decided to use kmeans to examine the relation between model_year and price, as we noticed a similar examination in one of the papers while doing the literature review. Because K-means utilizes distance metrics, we scale the data before clustering.

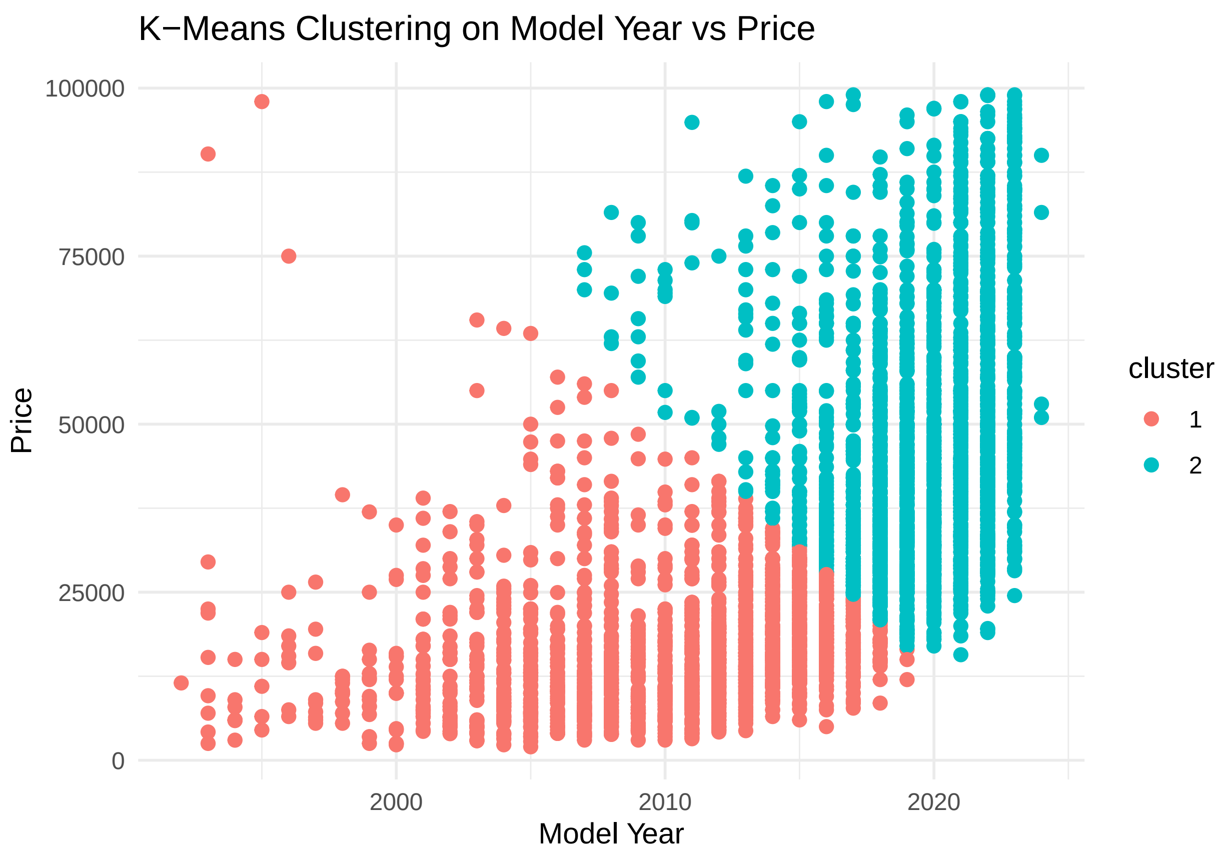
```r
set.seed(1)

sil_scores <- sapply(2:10, function(k) {
  km <- kmeans(scale(data_subset_scaled[, c("model_year", "price")]), centers = k, nstart = 10)
  silhouette(km$cluster, dist(scale(data_subset[, c("model_year", "price")]))) %>%
    summary() %>%
    .$avg.width
})
```

```r
plot(2:10, sil_scores, type = "b", pch = 19, frame = FALSE,
     xlab = "Number of Clusters (k)",
     ylab = "Average Silhouette Score",
     main = "Silhouette Method for Optimal k")
```



We decided to use the Silhouette Method to determine the optimal number of clusters. This method essentially uses distance measures calculating how close clusters are to themselves and how far away they are to other clusters to judge the optimal number of clusters. In this case, 2 has the highest average silhouette score so we will use k=2.

```r
set.seed(1)

kmeans_result <- kmeans(data_subset_scaled, centers = 2)
data_subset$cluster <- as.factor(kmeans_result$cluster)

ggplot(data_subset, aes(x = model_year, y = price, color = cluster)) +
  geom_point(size = 2) +
  labs(title = "K-Means Clustering on Model Year vs Price",
       x = "Model Year",
       y = "Price") +
  theme_minimal()
```



There seems to be a pretty solid relationship between a more recent model_year and higher price. Although the 2 clusters seem to be mostly dominated by model year, it's clear that the average price of cluster 2 is higher than cluster 1.

2. Hierarchical Clustering

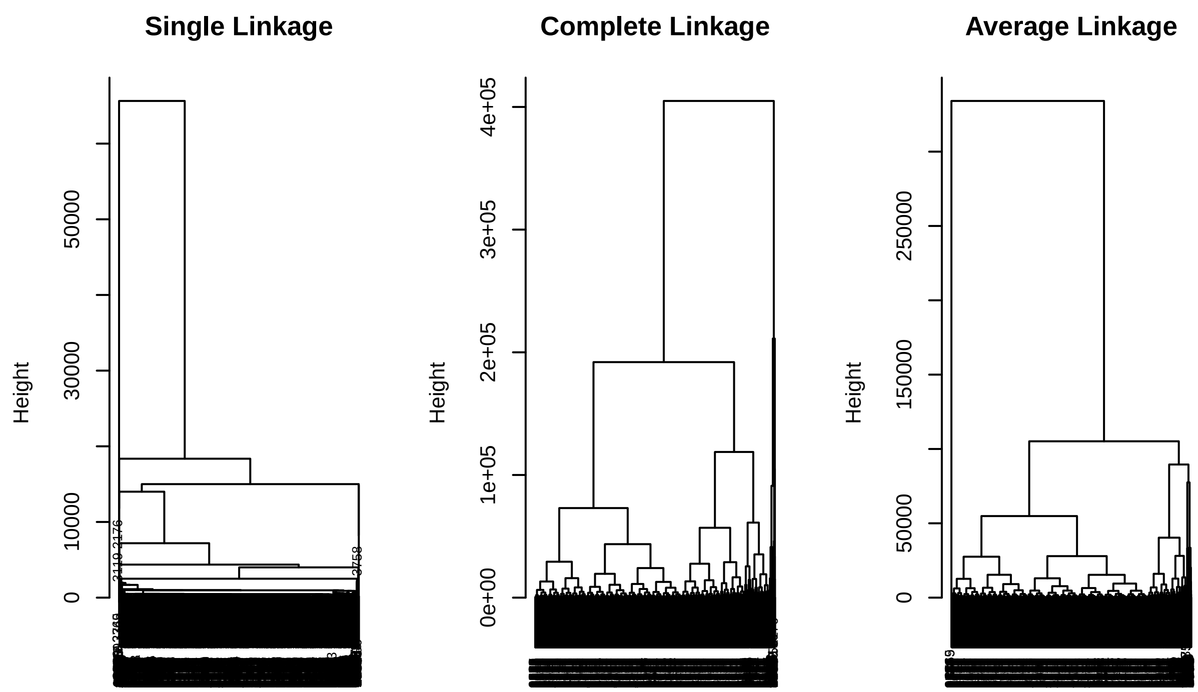
Next, we will try hierarchical clustering with three different linkage methods(single, complete, and average) using euclidean distance. Hierarchical Clustering begins with each data point starting as its own cluster. The goal is to progressively group them together until there is only one group. The process involves choosing the closest two groups, calculated through a specific distance metric.

```
#numeric_data <- data[, c("model_year", "price")]
numeric_data <- data[, sapply(data, is.numeric)]
numeric_data_without_price <- numeric_data[, !colnames(numeric_data) %in% "price"]
```

Removing non-numeric features as clustering requires numeric features. Also, removed the target feature price.

```
# Perform hierarchical clustering with scaled data
hclust_single <- hclust(dist(numeric_data_without_price, method = "euclidean"), method = "single")
hclust_complete <- hclust(dist(numeric_data_without_price, method = "euclidean"), method = "complete")
hclust_average <- hclust(dist(numeric_data_without_price, method = "euclidean"), method = "average")

# Plot the dendrograms
par(mfrow = c(1, 3))  # Arrange plots side by side
plot(hclust_single, main = "Single Linkage", xlab = "", sub = "", cex = 0.6)
```

```r
plot(hclust_complete, main = "Complete Linkage", xlab = "", sub = "", cex = 0.6)
plot(hclust_average, main = "Average Linkage", xlab = "", sub = "", cex = 0.6)
```




```r
par(mfrow = c(1, 1))  # Reset plotting layout
```

```r
# Cut the dendrogram into 2 clusters using complete linkage
clusters_complete <- cutree(hclust_complete, k = 2)

# Assign cluster labels to the dataset
data$cluster <- as.factor(clusters_complete)

# View the cluster sizes
table(data$cluster)
```

```
##
##    1    2
## 3674   91
```

```r
# Summarise cluster statistics
library(dplyr)

cluster_summary <- data %>%
  group_by(cluster) %>%
  summarise(
```

```
    avg_price = mean(price, na.rm = TRUE),
    avg_model_year = mean(model_year, na.rm = TRUE),
    avg_accident = mean(as.numeric(accident), na.rm = TRUE),  # Convert accident to numeric if necessar
    avg_mileage = mean(mileage, na.rm = TRUE),
    avg_horsepower = mean(horsepower, na.rm = TRUE),
    count = n()
  )

# Print the cluster summary
print(cluster_summary)
```

```
## # A tibble: 2 x 7
##   cluster avg_price avg_model_year avg_accident avg_mileage avg_horsepower count
##   <fct>       <dbl>          <dbl>        <dbl>       <dbl>          <dbl> <int>
## 1 1          34086.          2015.        0.254      64111.           322.  3674
## 2 2          10588.          2007.        0.484     228100.           267.    91
```

There are a lot of correlations here that make sense between the 2 clusters. Cluster 1, with a more recent avg_model_year, also has a lower avg_mileage and a lower avg_accident rate, probably because the car has been driven for less time, this cluster also has a much higher avg_price in comparison to cluster 2. The data isn't distributed very well however as a vast majority of the points sit in cluster 1, perhaps suggesting that hierarchical clustering isn't suitable for this dataset.

3. Spectral Clustering

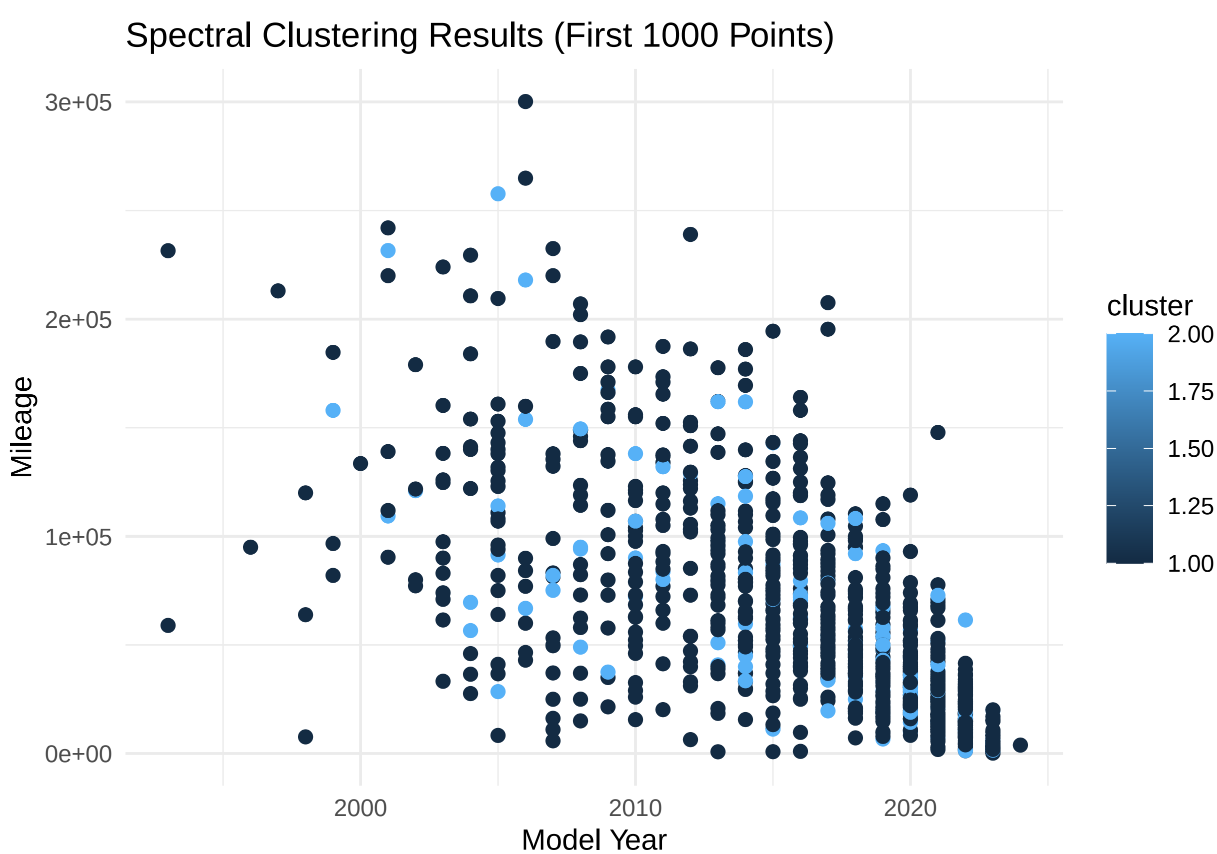
Finally, we will try spectral clustering, which aims to group observations based on their proximity information. This method involves 2 main steps, the first being using the eigenvalues of a similarity matrix to perform dimension reduction, followed by applying a clustering algorithm like K-means.

```
# Step 1: Prepare and scale data
# Select numeric columns only
numeric_data <- data[, sapply(data, is.numeric)]
numeric_data <- numeric_data[sample(nrow(numeric_data)), ]
numeric_data_without_price <- numeric_data[, !colnames(numeric_data) %in% "price"]  # Exclude 'price' c
numeric_data_without_price_scaled <- scale(numeric_data_without_price)  # Scale the data

# Step 2: Subset the first 1000 points
subset_data <- numeric_data_without_price_scaled[1:1000, ]

# Step 3: Perform spectral clustering
set.seed(1)  # For reproducibility
n_clusters <- 2  # Number of clusters
specc_result <- specc(as.matrix(subset_data), centers = n_clusters, kernel = "rbfdot")

# Step 4: Add cluster assignments to the original dataset
data$cluster <- NA  # Initialize cluster column
data$cluster[1:1000] <- as.factor(specc_result@.Data)  # Assign clusters to the first 1000 points

# Step 5: Summarize the clusters
cluster_summary <- data %>%
  filter(!is.na(cluster)) %>%
  group_by(cluster) %>%
```

```r
  summarise(
    avg_model_year = mean(model_year, na.rm = TRUE),
    avg_mileage = mean(mileage, na.rm = TRUE),
    avg_accident = mean(as.numeric(accident), na.rm = TRUE),  # Convert 'accident' to numeric if necess
    avg_horsepower = mean(horsepower, na.rm = TRUE),
    count = n()
  )

# Print the cluster summary
print(cluster_summary)
```

```
## # A tibble: 2 x 6
##   cluster avg_model_year avg_mileage avg_accident avg_horsepower count
##     <int>          <dbl>       <dbl>        <dbl>          <dbl> <int>
## 1       1          2016.      60899.        0.246           325.   859
## 2       2          2016.      60429.        0.220           329.   141
```

```r
# Step 6: Visualize the clusters (optional)
ggplot(data %>% filter(!is.na(cluster)), aes(x = model_year, y = mileage, color = cluster)) +
  geom_point(size = 2) +
  labs(title = "Spectral Clustering Results (First 1000 Points)", x = "Model Year", y = "Mileage") +
  theme_minimal()
```



Similar to Cluster 1, with a more recent avg_model_year, also has a lower avg_mileage and a lower avg_accident rate, this cluster also has a much higher avg_price in comparison to cluster 2. The distribution of data points between the 2 clusters seem to be more even in comparison to heirarchicaly clustering, meaning that perhaps spectral clustering is more suitable for this dataset.

Prediction Models

For all the supervised models below, we will split the data into training sets for model training and testing sets to evaluate performance and accuracy

```r
y = data_one_hot$price
X <- data_one_hot[, !(colnames(data_one_hot) %in% "price")]
#test_idx = sample(nrow(data), size = 0.2 * nrow(data))
#xtrain = X[ -test_idx, ]
#xtest  = X[ test_idx, ]
#ytrain = y[ -test_idx ]
#ytest  = y[ test_idx ]

sample <- sample(c(TRUE, FALSE), nrow(data), replace=TRUE, prob=c(0.8, 0.2))
xtrain <- X[sample, ]
xtest <- X[!sample, ]
ytrain = y[sample]
ytest = y[!sample]

#as.matrix(xtrain)
#sum(is.na(xtrain)
#colnames(xtrain)
head(xtrain)
```

```
##   model_year mileage fuel_type.Diesel fuel_type.Electric fuel_type.Flex Fuel
## 1       2013   51000                0                  0                   1
## 2       2021   34742                0                  0                   0
## 3       2022   22372                0                  0                   0
## 4       2015   88900                0                  1                   0
## 5       2021    9835                0                  0                   0
## 6       2016  136397                0                  0                   0
##   fuel_type.Gasoline fuel_type.Hybrid fuel_type.NA
## 1                  0                0            0
## 2                  0                0            1
## 3                  0                0            1
## 4                  0                0            0
## 5                  0                0            1
## 6                  0                0            1
##   fuel_type.Plug-In Electric/Gas transmission.6-Speed A/T
## 1                              0                        1
## 2                              0                        0
## 3                              0                        0
## 4                              0                        0
## 5                              0                        0
## 6                              0                        0
##   transmission.6-Speed M/T transmission.7-Speed A/T transmission.8-Speed A/T
## 1                        0                        0                        0
## 2                        0                        0                        0
## 3                        0                        0                        0
## 4                        0                        1                        0
```

```
## 5                        0                     0                         0
## 6                        0                     0                         0
##   transmission.A/T transmission.Automatic transmission.Other
## 1                0                      0                  0
## 2                0                      0                  1
## 3                0                      1                  0
## 4                0                      0                  0
## 5                0                      0                  1
## 6                0                      0                  1
##   transmission.Transmission w/Dual Shift Mode ext_col.Black ext_col.Brown
## 1                                           0             1             0
## 2                                           0             0             0
## 3                                           0             0             0
## 4                                           0             1             0
## 5                                           0             0             0
## 6                                           0             0             0
##   ext_col.Gold ext_col.Gray ext_col.Other ext_col.White int_col.Beige/Ivory
## 1            0            0             0             0                   0
## 2            0            0             1             0                   0
## 3            0            0             1             0                   0
## 4            0            0             0             0                   0
## 5            0            0             0             1                   0
## 6            0            1             0             0                   0
##   int_col.Black int_col.Gray int_col.Other accident clean_title horsepower
## 1             1            0             0        1           1        300
## 2             0            1             0        1           1        310
## 3             1            0             0        0           0        310
## 4             1            0             0        0           1        354
## 5             1            0             0        0           0        310
## 6             0            0             1        0           0        310
##   displacement cylinders.10 Cylinder cylinders.12 Cylinder cylinders.3 Cylinder
## 1          3.7                     0                     0                    0
## 2          3.8                     0                     0                    0
## 3          3.5                     0                     0                    0
## 4          3.5                     0                     0                    0
## 5          2.0                     0                     0                    0
## 6          3.5                     0                     0                    0
##   cylinders.4 Cylinder cylinders.5 Cylinder cylinders.6 Cylinder
## 1                    0                    0                    1
## 2                    0                    0                    0
## 3                    0                    0                    0
## 4                    0                    0                    1
## 5                    0                    0                    0
## 6                    0                    0                    0
##   cylinders.8 Cylinder cylinders.NA engine_type.DOHC engine_type.Electric Motor
## 1                    0            0                0                         0
## 2                    0            1                1                         0
## 3                    0            1                1                         0
## 4                    0            0                0                         0
## 5                    0            1                1                         0
## 6                    0            1                0                         0
##   engine_type.NA engine_type.SOHC engine_type.Turbo engine_type.Twin Turbo
## 1              1                0                 0                      0
## 2              0                0                 0                      0
```

```
## 3                 0                0                 0                      0
## 4                 1                0                 0                      0
## 5                 0                0                 0                      0
## 6                 1                0                 0                      0
```

1. Linear Model. There are mainly three possible linear models: Lasso, Ridge, and Elastic Net. We will try all three models and see which one performs the best. Lasso, Ridge, and Elastic Net all benefit from feature scaling because these models involve regularization. which will penalize the size of coefficients of the model to avoid overfitting. All 3 models also involving a tuning parameter, and so we will use k-fold cross validation to find the best parameters. cv.glmnet will automatically scale and center the data as well.

Training our ridge model

```r
ridgemodel = cv.glmnet(x = as.matrix(xtrain), y = ytrain, nfolds = 10, alpha = 0)
```

```r
ridgemodel$lambda.min
```

```
## [1] 1368.33
```

```r
pred = predict(ridgemodel, newx = as.matrix(xtest), s = "lambda.min")
sqrt(mean((pred - ytest)^2))
```

```
## [1] 12017.06
```

Training our lasso model

```r
lassomodel = cv.glmnet(x = as.matrix(xtrain), y = ytrain, nfolds = 10, alpha = 1)
```

```r
lassomodel$lambda.min
```

```
## [1] 82.02921
```

```r
pred = predict(lassomodel, newx = as.matrix(xtest), s = "lambda.min")
sqrt(mean((pred - ytest)^2))
```

```
## [1] 11992.87
```

Training our elastic net model

```r
elastic_net_model <- cv.glmnet(x = as.matrix(xtrain), y = ytrain, nfolds = 10, alpha = 0.5)
```

```r
elastic_net_model$lambda.min
```

```
## [1] 149.4839
```

```r
pred1 = predict(elastic_net_model, newx = as.matrix(xtest), s = "lambda.min")
sqrt(mean((pred1 - ytest)^2))
```

```
## [1] 11993.98
```

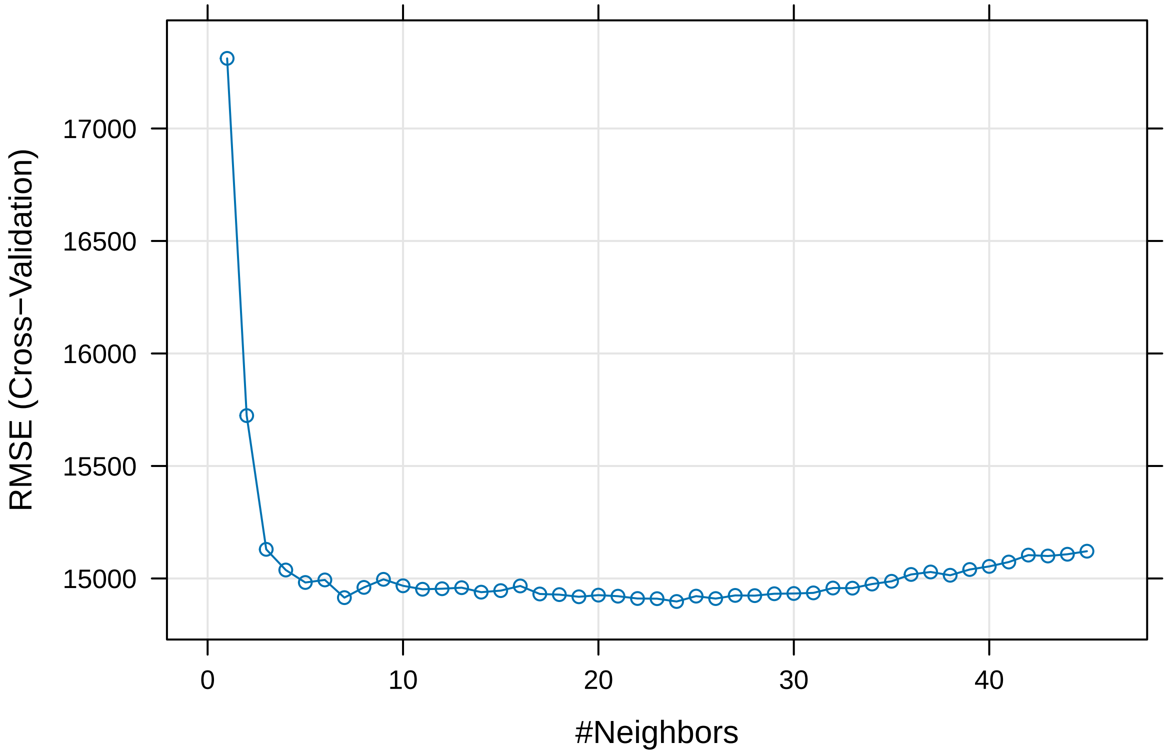
Out of our 3 linear models, Ridge performed the best, with a RMSE of 12261.19

2. K Nearest Neighbors(KNN) regression works by calculating the k nearest training set data points to the test point and predicting the target value by taking the average of their target values. KNN is sensitive to feature scaling, so we will need to scale the data. The reason behind this is for example, if one feature has ranges from 1-10 and another one has 1-10000, distance calcualtions will be biased and results will suffer as a result. KNN is also sensitive to the choice of k. To find the optimal value of k, we will perform k-fold cross validation.

```r
set.seed(1)

# 2. Scale the numeric columns
preProcValues <- preProcess(xtrain, method = c("center", "scale"))

# Apply scaling and centering to the training and test data
xtrain_processed <- predict(preProcValues, xtrain)
xtest_processed <- predict(preProcValues, xtest)

pca_model <- prcomp(xtrain_processed, center = TRUE, scale. = TRUE)
explained_variance <- summary(pca_model)$importance[3, ]  # Cumulative variance
num_components <- which(explained_variance >= 0.95)[1]   # First component to reach 95%

# Print the number of components
cat("Number of components to retain:", num_components, "\n")
```

```
## Number of components to retain: 32
```

```r
# Transform the training and testing data
xtrain_pca <- pca_model$x[, 1:num_components]  # Retain the first num_components
xtest_pca <- as.matrix(xtest_processed) %*% pca_model$rotation[, 1:num_components]

# 3. Train the KNN model using the processed data
control <- trainControl(method = "cv", number = 10)
knn.cvfit <- train(ytrain ~ ., method = "knn",
                   data = data.frame(xtrain_pca, ytrain),
                   tuneGrid = data.frame(k = seq(1, 45, 1)),
                   trControl = control)

# 4. Plot the cross-validation results
plot(knn.cvfit)
```




```r
# 6. Print the best value of k based on cross-validation
print(paste("The best value of k based on cross-validation is: ", knn.cvfit$bestTune$k))
```

```
## [1] "The best value of k based on cross-validation is:  24"
```

```r
# Train the final model using the best value of k and find the predictions
best_k <- knn.cvfit$bestTune$k
knn_predictions <- knn(train = xtrain_processed, test = xtest_processed, cl = ytrain, k = best_k)

# Calculate prediction error
print(paste("Prediction errort: ", sqrt(mean((as.numeric(knn_predictions) - ytest)^2))))
```

```
## [1] "Prediction errort:  39760.6901508748"
```

3. Random Forest
4. SVM? does this count as a linear model
5. Gradient Boosting Regressor

Open-Ended Question/Conclusion

A researcher is interested in estimating the original price of the cars in your dataset as if they were brand new.

Since you are predicting prices without direct historical data for new cars, you may be extrapolating beyond the range of your training data, which can lead to inaccuracies. External factors such as changes in market demand, economic conditions, or new models being released can affect car prices but may not be captured in your model.