HW3

Jadon Fowler jaf582 5778191

2022-10-18

Instructions

Download a copy of this markdown. Change the `author:` tag above to have your name and NAU's ID. Fill in the file with your solution to the proposed problems. Knit your final document to PDF and submit it through BBLearn in the assignment `[HW] Homework 3: review and cleaning` by the end of the day on **Tuesday, October 18 (11:59 PM)**.

Note: to knit your homework to PDF, you need to have MikTex installed. You can download and install MikTex from here. If you don't want to install MikTex, you can knit to HTML, open the file in the browser and print the page to PDF.

For all the problems, please import the useful libraries in this chunk of code:

Problem 1: Load the data

The dataset used in this assignment is the Automobile Data Set by Jeffrey C. Schlimmer. The goal is to use this data to predict **prices** of automobiles.

The following attributes are provided:

```
1. symboling: -3, -2, -1, 0, 1, 2, 3.
2. normalized-losses: continuous from 65 to 256.
3. make: alfa-romero, audi, bmw, chevrolet, dodge, honda, isuzu, jaguar, mazda, mercedes-benz, mercury,
4. fuel-type: diesel, gas.
5. aspiration: std, turbo.
6. num-of-doors: four, two.
7. body-style: hardtop, wagon, sedan, hatchback, convertible.
8. drive-wheels: 4wd, fwd, rwd.
9. engine-location: front, rear.
10. wheel-base: continuous from 86.6 120.9.
11. length: continuous from 141.1 to 208.1.
12. width: continuous from 60.3 to 72.3.
13. height: continuous from 47.8 to 59.8.
14. curb-weight: continuous from 1488 to 4066.
15. engine-type: dohc, dohcv, l, ohc, ohcf, ohcv, rotor.
16. num-of-cylinders: eight, five, four, six, three, twelve, two.
17. engine-size: continuous from 61 to 326.
18. fuel-system: 1bbl, 2bbl, 4bbl, idi, mfi, mpfi, spdi, spfi.
19. bore: continuous from 2.54 to 3.94.
20. stroke: continuous from 2.07 to 4.17.
21. compression-ratio: continuous from 7 to 23.
22. horsepower: continuous from 48 to 288.
```

```
23. peak-rpm: continuous from 4150 to 6600.
24. city-mpg: continuous from 13 to 49.
25. highway-mpg: continuous from 16 to 54.
26. price: continuous from 5118 to 45400.
```

Load the `.csv` file and show the `head()`. What do you notice about variable names? Solve the issue and explain your solution.

> The csv file doesn't have a line for column names. We can fix this by setting `col_names = FALSE` in `read_csv()` add we can add our own column names to the dataframe.

```r
real.col.names <- c(
  "symboling",
  "normalized.losses",
  "make",
  "fuel.type",
  "aspiration",
  "num.of.doors",
  "body.style",
  "drive.wheels",
  "engine.location",
  "wheel.base",
  "length",
  "width",
  "height",
  "curb.weight",
  "engine.type",
  "num.of.cylinders",
  "engine.size",
  "fuel.system",
  "bore",
  "stroke",
  "compression.ratio",
  "horsepower",
  "peak.rpm",
  "city.mpg",
  "highway.mpg",
  "price"
)
auto <- as.data.frame(read_csv("auto.csv", col_names = FALSE))
```

```
## Rows: 205 Columns: 26
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (14): X2, X3, X4, X5, X6, X7, X8, X9, X15, X16, X18, X19, X22, X23
## dbl (12): X1, X10, X11, X12, X13, X14, X17, X20, X21, X24, X25, X26
```

```
##
## 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.
```

```
colnames(auto) <- real.col.names
head(auto)
```

```
##   symboling normalized.losses        make fuel.type aspiration num.of.doors
## 1         3                 ? alfa-romero       gas        std          two
## 2         3                 ? alfa-romero       gas        std          two
## 3         1                 ? alfa-romero       gas        std          two
## 4         2               164        audi       gas        std         four
## 5         2               164        audi       gas        std         four
## 6         2                 ?        audi       gas        std          two
##    body.style drive.wheels engine.location wheel.base length width height
## 1 convertible          rwd           front       88.6  168.8  64.1   48.8
## 2 convertible          rwd           front       88.6  168.8  64.1   48.8
## 3   hatchback          rwd           front       94.5  171.2  65.5   52.4
## 4       sedan          fwd           front       99.8  176.6  66.2   54.3
## 5       sedan        allwd           front       99.4  176.6  66.4   54.3
## 6       sedan          fwd           front       99.8  177.3  66.3   53.1
##   curb.weight engine.type num.of.cylinders engine.size fuel.system bore stroke
## 1        2548        dohc             four         130        mpfi 3.47   2.68
## 2        2548        dohc             four         130        mpfi 3.47   2.68
## 3        2823        ohcv              six         152        mpfi 2.68   3.47
## 4        2337         ohc             four         109        mpfi 3.19   3.40
## 5        2824         ohc             five         136        mpfi 3.19   3.40
## 6        2507         ohc             five         136        mpfi 3.19   3.40
##   compression.ratio horsepower peak.rpm city.mpg highway.mpg price
## 1               9.0        111     5000       21          27 13495
## 2               9.0        111     5000       21          27 16500
## 3               9.0        154     5000       19          26 16500
## 4              10.0        102     5500       24          30 13950
## 5               8.0        115     5500       18          22 17450
## 6               8.5        110     5500       19          25 15250
```

Problem 2: Identifying the missing data

Inspect the dataset and the columns individually. Can you tell what codification is being used to address missing data?

> Some columns use `?`, some use `-`, and some use `NA` for missing data. We can normalize this by replacing weird values with `NA`.

```
auto$normalized.losses[auto$normalized.losses == '?'] <- NA
auto$num.of.doors[auto$num.of.doors == '-'] <- NA
auto$bore[auto$bore == '-'] <- NA
auto$horsepower[auto$horsepower == '?'] <- NA
auto$peak.rpm[auto$peak.rpm == '?'] <- NA
```

Problem 3: Identifying typos

Several columns in this dataframe have typos (text that was misspelled by mistake). Identify and fix the typos. Explain your reasoning.

We can replace specific strings with other strings in the bad columns.

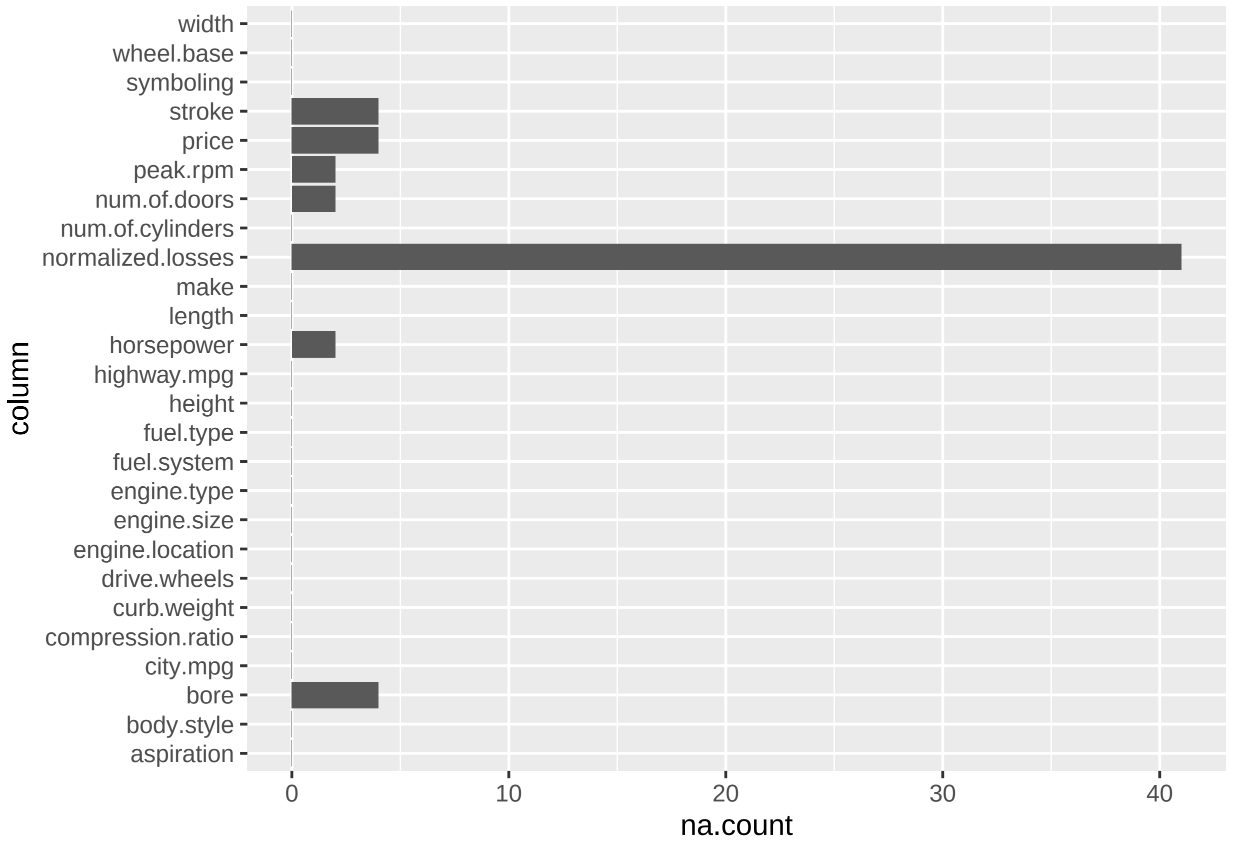
```r
auto$make[auto$make == 'cherolet'] <- 'chevrolet'
auto$make[auto$make == 'plymouht'] <- 'plymouth'
```

Problem 4: dealing with the missing data

Plot the missing data, is there any columns/rows that we would benefit from droping? Remember that the goal of the analysis is to predict the auto's prices. In case we have to deal with the missing values, which strategy would you use? Use functions and plots to demonstrate to argue on whether it would be interesting replace by the most frequent class or the mean value of the column. Explain your reasoning. Then, write a code that perform the changes you suggested.

Number of doors & number of cylinders can be replaced by the median Normalized.losses is missing over 40 items so it might be useful to just remove it. The rest of the columns can be replaced with the mean.

```r
# plot na values vs column
na.values <- as.data.frame(sapply(auto, function (x) sum(is.na(x))))
na.values <- cbind(na.values,rownames(na.values))
rownames(na.values) <- NULL
colnames(na.values) <- c("na.count", "column")
ggplot(na.values) +
  geom_col(aes(x = na.count, y = column))
```



```r
# replace NAs with medians
auto$num.of.cylinders[is.na(auto$num.of.cylinders)] <- median(auto$num.of.cylinders[!is.na(auto$num.of.
auto$num.of.doors[is.na(auto$num.of.doors)] <- median(auto$num.of.doors[!is.na(auto$num.of.doors)])

# replace NAs with means
auto$stroke[is.na(auto$stroke)] <- mean(as.numeric(auto$stroke[!is.na(auto$stroke)]))
auto$price[is.na(auto$price)] <- mean(as.numeric(auto$price[!is.na(auto$price)]))
auto$peak.rpm[is.na(auto$peak.rpm)] <- mean(as.numeric(auto$peak.rpm[!is.na(auto$peak.rpm)]))
auto$horsepower[is.na(auto$horsepower)] <- mean(as.numeric(auto$horsepower[!is.na(auto$horsepower)]))
auto$bore[is.na(auto$bore)] <- mean(as.numeric(auto$bore[!is.na(auto$bore)]))
```

Problem 5: Type conversion

Look at the structure of your file. Are the variables represented in the appropriate data type? Convert the types for the ones that are not!

```r
auto$symboling <- as.factor(auto$symboling)

number.lookup <- c(
  "one" = 1,
  "two" = 2,
  "three" = 3,
  "four" = 4,
  "five" = 5,
  "six" = 6,
  "seven" = 7,
  "eight" = 8,
  "nine" = 9,
  "ten" = 10
)

auto$num.of.cylinders <- number.lookup[auto$num.of.cylinders]
auto$num.of.doors <- number.lookup[auto$num.of.doors]
```

Extra credit (10 points toward HW rubric): can you standardize the data?

Sometimes we get data from a source that uses a different measuring unit from what we expect. For example, in our in-class examples, we needed date in the `mm/dd/yyyy` format, but we got it in seconds. We had to transform the data into a more appropriate format/measuring unit. In data wrangling, this is called **Transformation**.

1. Transform the variables that stores the fuel consumption (variables `24. city-mpg` and `25. highway-mpg`) to store the values in `liter per Km`.

```r
# convert mpg to kilometers per liter by diving the value by 2.352
# and then turn that into liter per Km by flipping it
lpk <- function (x) 1 / (x / 2.352)
auto$city.mpg <- lpk(auto$city.mpg)
auto$highway.mpg <- lpk(auto$highway.mpg)
```

2. Transform the length measurements from inches to centimeters (variables `11. length`, `12. width`, `13. height`, and `14. curb-weight`).

```r
auto$length <- auto$length * 2.54
auto$width <- auto$width * 2.54
auto$height <- auto$height * 2.54
auto$curb.weight <- auto$curb.weight * 2.54
```