221030 Homework 8

Rabail Adwani

2022-10-30

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

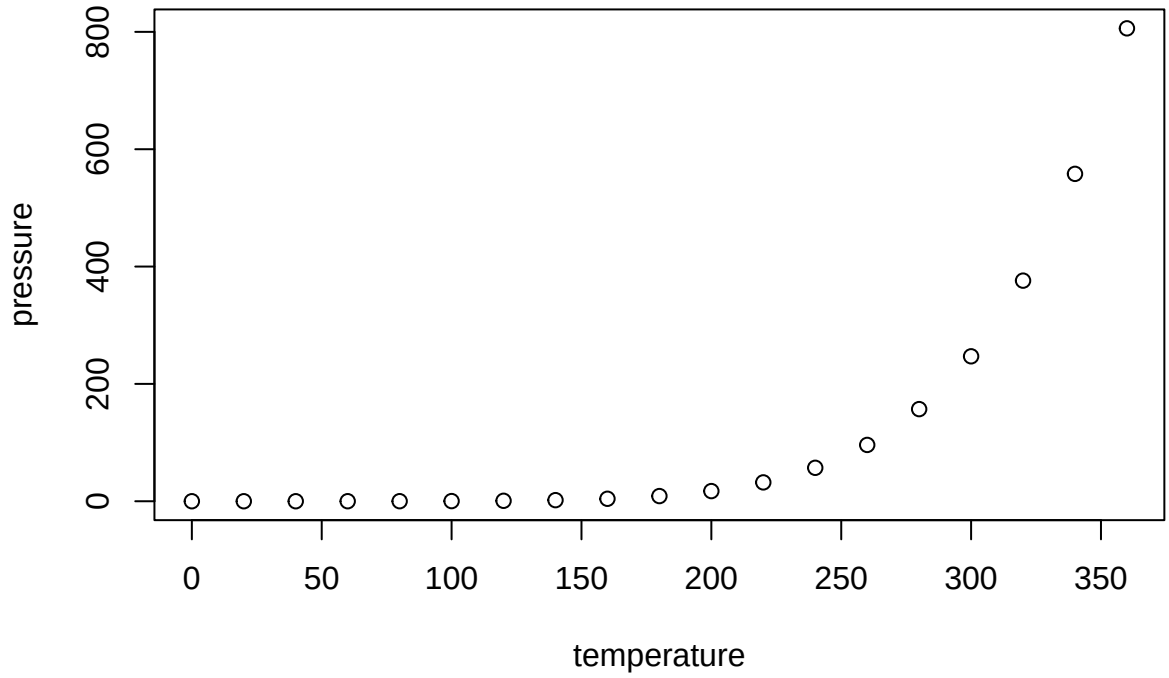
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

Homework 8

Data description:

A Chinese automobile company, named Geely Auto, is targeting the US market to set up a manufacturing unit and compete with their US and European counterparts. Since the dynamics of the American market can be very different in comparison with the Chinese market, they have contracted with an automobile consulting company to provide insights. Basically, they are looking for the factors that are significant in predicting the price of a car in the US. Therefore, the automobile consulting firm has gathered a large dataset of different types of cars across the American market. The data file carprices.csv has information on 205 cars, and p = 24 predictors that may help explain changes in price.

Source: https://archive.ics.uci.edu/ml/datasets/Automobile

Variable information

1. Car_ID: Unique id of each observation (Integer)
2. Symboling: Its assigned insurance risk rating, A value of +3 indicates that the auto is risky, -3 that it is probably pretty safe (Categorical)
3. carCompany: Name of car company (Categorical)
4. fueltype: Car fuel type i.e gas or diesel (Categorical)

5. aspiration: Aspiration used in a car (Categorical)
6. doornumber: Number of doors in a car (Categorical)
7. carbody: body of car (Categorical)
8. drivewheel: type of drive wheel (Categorical)
9. enginelocation: Location of car engine (Categorical)
10. wheelbase: Wheelbase of car (Numeric)
11. carlength: Length of car (Numeric)
12. carwidth: Width of car (Numeric)
13. carheight: height of car (Numeric)
14. curbweight: The weight of a car without occupants or baggage. (Numeric)
15. enginetype: Type of engine. (Categorical)
16. cylindernumber: cylinder placed in the car (Categorical)
17. enginesize: Size of car (Numeric)
18. fuelsystem: Fuel system of car (Categorical)
19. boreratio: Boreratio of car (Numeric)
20. stroke: Stroke or volume inside the engine (Numeric)
21. compressionratio: compression ratio of car (Numeric)
22. horsepower: Horsepower (Numeric)
23. peakrpm: car peak rpm (Numeric)
24. citympg: Mileage in city (Numeric)
25. highwaympg: Mileage on highway (Numeric)
26. price: Price of car (Numeric)

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(car)
```

```
## Loading required package: carData
```

```
library(ppcor)
```

```
## Loading required package: MASS
```

```
library(olsrr)
```

```
##
## Attaching package: 'olsrr'

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

## The following object is masked from 'package:datasets':
##
##     rivers
```

```
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-4
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --

## v tibble  3.1.8     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.1
## v readr   2.1.2     v forcats 0.5.2
## v purrr   0.3.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::expand() masks Matrix::expand()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
## x tidyr::pack()   masks Matrix::pack()
## x dplyr::recode() masks car::recode()
## x dplyr::select() masks MASS::select()
## x purrr::some()   masks car::some()
## x tidyr::unpack() masks Matrix::unpack()
```

```
library(ggplot2)
library(RobStatTM)
```

```
##
## Attaching package: 'RobStatTM'
##
## The following objects are masked from 'package:MASS':
##
##     huber, oats
##
## The following object is masked from 'package:datasets':
##
##     stackloss
```

```
library(ppcor)
library(leaps)

# Reading the data
setwd("F:/MSDS/Applied Statistics for Data Science")
carprices <- read.csv("Data/carprices.csv", header=TRUE)
```

Missing values:

There are no missing values in the carprices.csv dataset. Therefore, we can move forward in data preprocessing.

```
# Checking for missing values
table(is.na(carprices))
```

```
##
## FALSE
##  5330
```

Encoding the categorical data:

Encoding the categorical data refers to transforming the variables from Characters to Factors. Keeping the variable as character can cause many issues in building a regression model. Below, we have transformed 11 columns to factors.

```
# Encoding the categorical data
str(carprices)
```

```
## 'data.frame':    205 obs. of  26 variables:
##  $ car_ID          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ symboling       : int  3 3 1 2 2 2 1 1 1 0 ...
##  $ CarName         : chr  "alfa-romero giulia" "alfa-romero stelvio" "alfa-romero Quadrifoglio" "audi
##  $ fueltype        : chr  "gas" "gas" "gas" "gas" ...
##  $ aspiration      : chr  "std" "std" "std" "std" ...
##  $ doornumber      : chr  "two" "two" "two" "four" ...
##  $ carbody         : chr  "convertible" "convertible" "hatchback" "sedan" ...
##  $ drivewheel      : chr  "rwd" "rwd" "rwd" "fwd" ...
##  $ enginelocation  : chr  "front" "front" "front" "front" ...
##  $ wheelbase       : num  88.6 88.6 94.5 99.8 99.4 ...
##  $ carlength       : num  169 169 171 177 177 ...
##  $ carwidth        : num  64.1 64.1 65.5 66.2 66.4 66.3 71.4 71.4 71.4 67.9 ...
##  $ carheight       : num  48.8 48.8 52.4 54.3 54.3 53.1 55.7 55.7 55.9 52 ...
##  $ curbweight      : int  2548 2548 2823 2337 2824 2507 2844 2954 3086 3053 ...
##  $ enginetype      : chr  "dohc" "dohc" "ohcv" "ohc" ...
##  $ cylindernumber  : chr  "four" "four" "six" "four" ...
##  $ enginesize      : int  130 130 152 109 136 136 136 136 131 131 ...
##  $ fuelsystem      : chr  "mpfi" "mpfi" "mpfi" "mpfi" ...
##  $ boreratio       : num  3.47 3.47 2.68 3.19 3.19 3.19 3.19 3.19 3.13 3.13 ...
##  $ stroke          : num  2.68 2.68 3.47 3.4 3.4 3.4 3.4 3.4 3.4 3.4 ...
##  $ compressionratio: num  9 9 9 10 8 8.5 8.5 8.5 8.3 7 ...
##  $ horsepower      : int  111 111 154 102 115 110 110 110 140 160 ...
##  $ peakrpm         : int  5000 5000 5000 5500 5500 5500 5500 5500 5500 5500 ...
##  $ citympg         : int  21 21 19 24 18 19 19 19 17 16 ...
```

```
##  $ highwaympg      : int  27 27 26 30 22 25 25 25 20 22 ...
##  $ price           : num  13495 16500 16500 13950 17450 ...
```

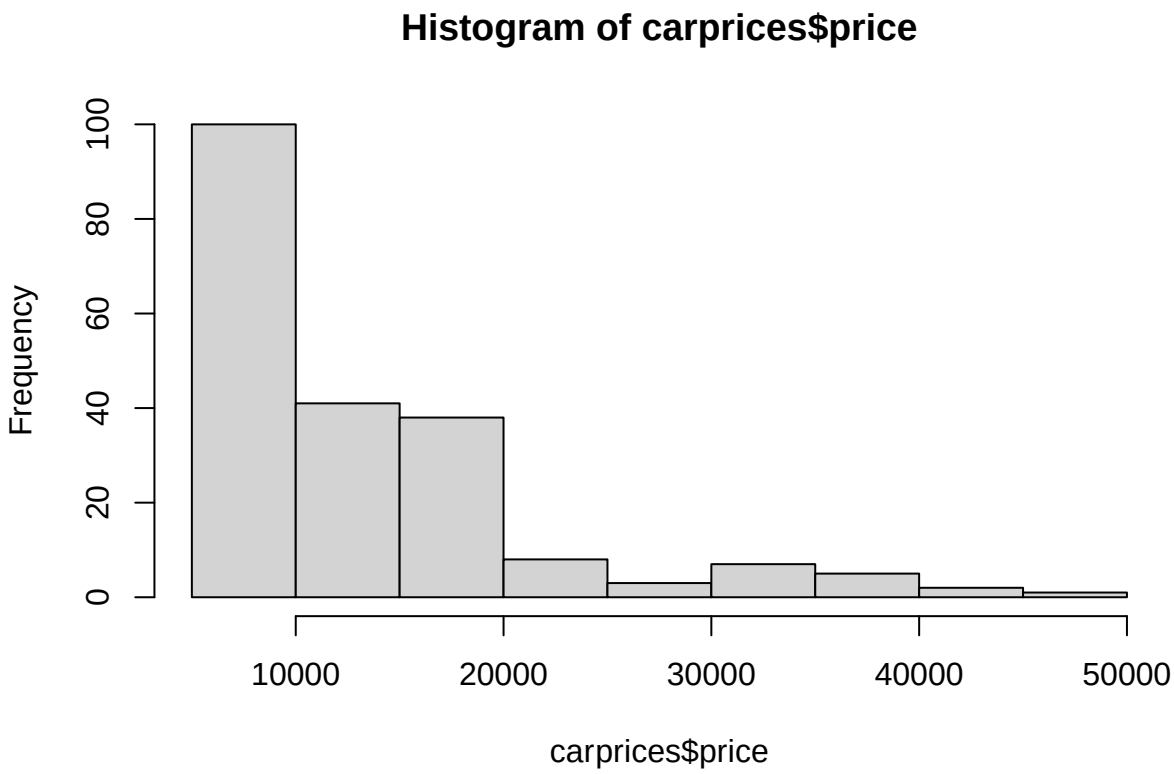
```
cat.cols <- c("symboling", "CarName", "fueltype", "aspiration", "doornumber",
              "carbody", "drivewheel", "enginelocation", "enginetype",
              "cylindernumber", "fuelsystem")
carprices[,cat.cols] <- lapply(carprices[,cat.cols], factor)
sapply(carprices, class)
```

```
##           car_ID        symboling          CarName         fueltype
##        "integer"         "factor"         "factor"         "factor"
##       aspiration       doornumber          carbody       drivewheel
##         "factor"         "factor"         "factor"         "factor"
##   enginelocation        wheelbase        carlength         carwidth
##         "factor"        "numeric"        "numeric"        "numeric"
##        carheight       curbweight       enginetype   cylindernumber
##        "numeric"        "integer"         "factor"         "factor"
##       enginesize       fuelsystem        boreratio           stroke
##        "integer"         "factor"        "numeric"        "numeric"
## compressionratio       horsepower          peakrpm          citympg
##        "numeric"        "integer"        "integer"        "integer"
##       highwaympg            price
##        "integer"        "numeric"
```

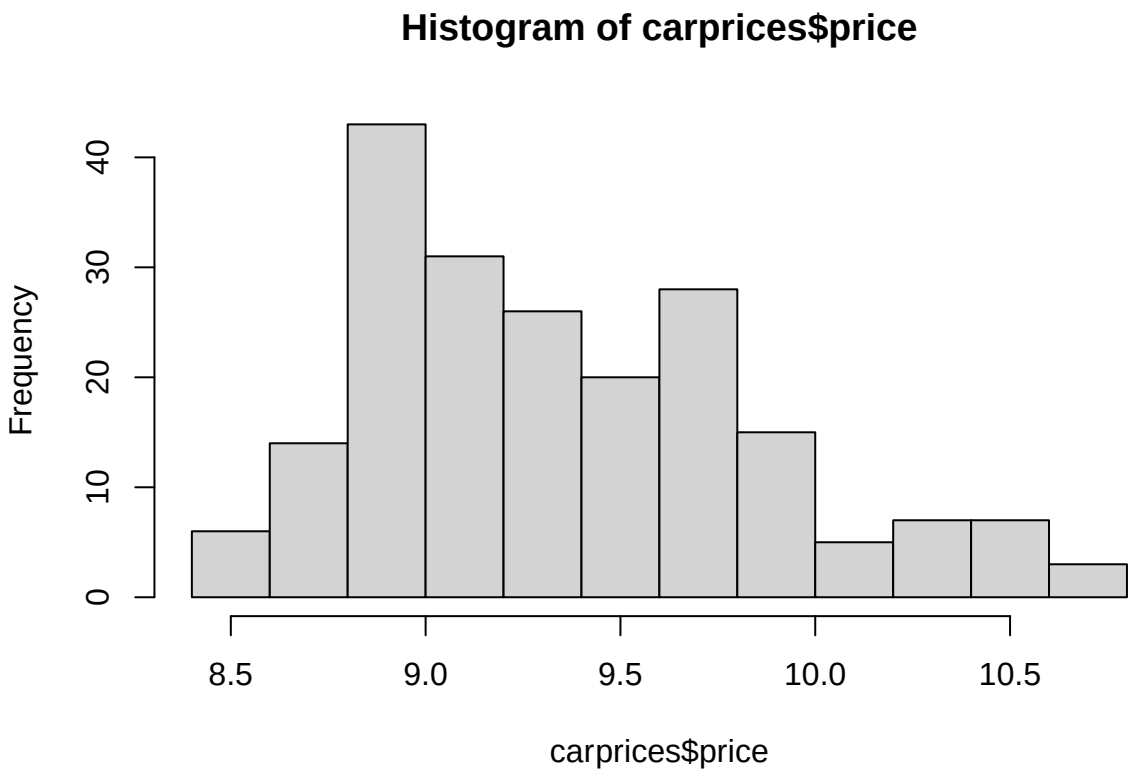
Log transformation of price (target variable):

We have used log transformation of price as the target variable because price is skewed to the right as evident by the histogram. The transformation helps fix a non-linear relationship between X and Y to make it more linear, which is an important assumption of our multiple linear regression model.

```
# Checking target's (Y) distribution
hist(carprices$price)
```




```
# Transforming the target using log()
carprices$price <- log(carprices$price)
hist(carprices$price)
```



MLR fit using all predictor variables:

Looking at the p-values, we can see that some variables are significant in explaining log transformation of price, while others seem to be ineffective. The fitted versus actual plot for the training data shows reasonably good fit, and indicates several outliers. There are a total of 8 outliers whose magnitude of the raw residual is larger than the rest of the cases in the dataset. In the plot 1 of diagnostics (residual vs fitted), the horizontal line shows no distinct patterns which is an indication of linear relationship. In plot 2 (Normal Q-Q), the residuals are following the straight dashed line. Therefore, they are normally distributed. However, some points on the tails do not fall on the straight line owing to outliers. In plot 3 (scale-location), a horizontal line with equally spread points is a good indication of homoscedasticity. However, that does not seem to be true because the line has kinks. As for the CooksD, we will be addressing it later in the report.

Formulas for MLR:

$$price_i^* = \beta_0^* + \beta_1^* symboling_i^* + \tag{1}$$
$$\beta_2^* fueltype_i^* + \beta_3^* aspiration_i^* + \tag{2}$$
$$\beta_4^* doornumber_i^* + \beta_5^* carbody_i^* + \tag{3}$$
$$\beta_6^* drivewheel_i^* + \beta_7^* enginelocation_i^* + \tag{4}$$
$$\beta_8^* enginetype_i^* + \beta_9^* cylindernumber_i^* + \tag{5}$$
$$\beta_{10}^* fuelsystem_i^* + \beta_{11}^* wheelbase_i^* + \tag{6}$$
$$\beta_{12}^* carlength_i^* + \beta_{13}^* carwidth_i^* + \tag{7}$$
$$\beta_{14}^* carheight_i^* + \beta_{15}^* curbweight_i^* + \tag{8}$$
$$\beta_{16}^* enginesize_i^* + \beta_{17}^* boreratio_i^* + \tag{9}$$
$$\beta_{18}^* stroke_i^* + \beta_{19}^* compressionratio_i^* + \tag{10}$$
$$\beta_{20}^* horsepower_i^* + \beta_{21}^* peakrpm_i^* + \tag{11}$$
$$\beta_{22}^* citympg_i^* + \beta_{23}^* highwaympg_i^* + \tag{12}$$
$$\epsilon_i, \tag{13}$$

```
# Making the training and validation split
train.prop <- 0.8
set.seed(123457)
trnset <- sort(sample(1:nrow(carprices), ceiling(nrow(carprices)*train.prop)))
train.set <- carprices[trnset, ]
test.set  <- carprices[-trnset, ]

# Standardizing the continuous predictor variables
contpredcols <- c("wheelbase", "carlength", "carwidth", "carheight",
                  "curbweight", "enginesize", "boreratio", "stroke",
                  "compressionratio", "horsepower", "peakrpm", "citympg",
                  "highwaympg")
normParam <- preProcess(train.set[,contpredcols],
                        method = c("center", "scale"))
data.train <- cbind(train.set[,c("price", "symboling", "fueltype",
                                 "aspiration", "doornumber", "carbody",
                                 "drivewheel", "enginelocation", "enginetype",
                                 "cylindernumber", "fuelsystem")],
                    predict(normParam, train.set[,contpredcols]))
data.test <- cbind(test.set[,c("price", "symboling", "fueltype",
                                 "aspiration", "doornumber", "carbody",
                                 "drivewheel", "enginelocation", "enginetype",
                                 "cylindernumber", "fuelsystem")],
                    predict(normParam, test.set[,contpredcols]))

# Fitting the MLR model
mod.1 <- lm(price ~., data=data.train)
summary(mod.1)
```

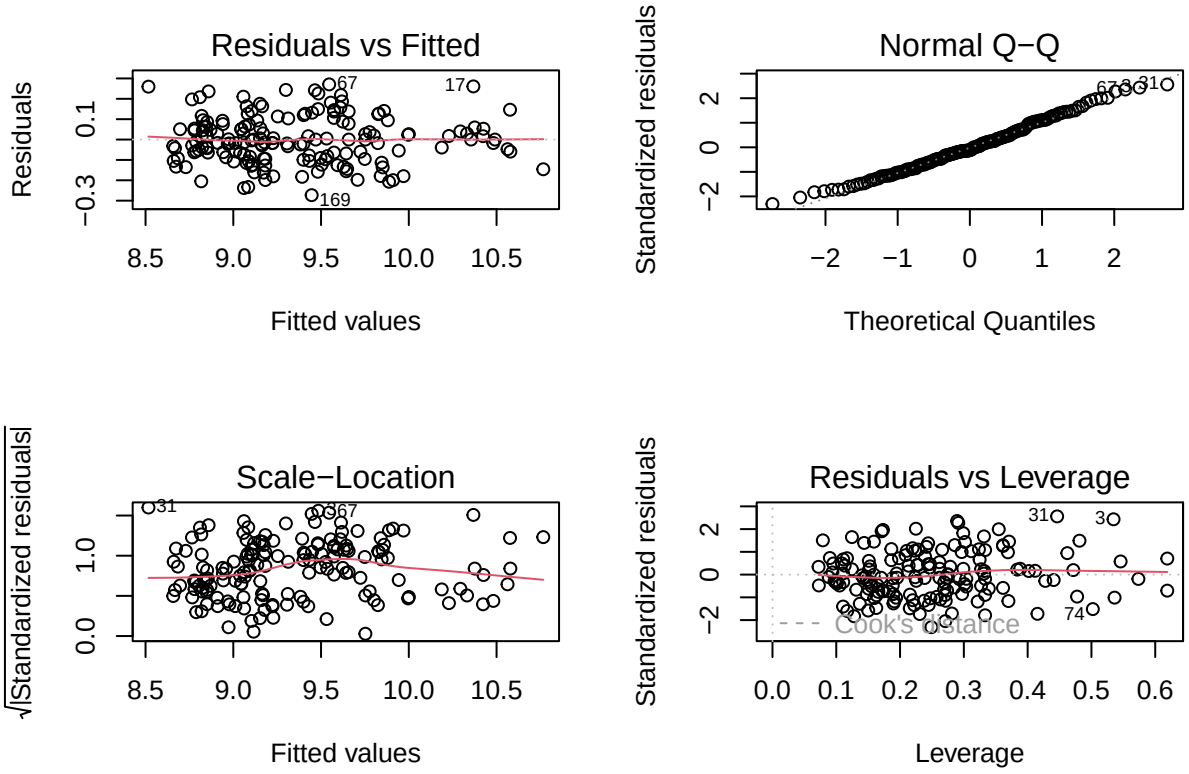
```
##
## Call:
## lm(formula = price ~ ., data = data.train)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
```

```
## -0.273069 -0.085091 -0.007804  0.076371  0.270811
##
## Coefficients: (2 not defined because of singularities)
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)            9.418158   0.537964  17.507  < 2e-16 ***
## symboling-1            0.137629   0.095832   1.436  0.15356
## symboling0             0.220260   0.093874   2.346  0.02060 *
## symboling1             0.136059   0.102991   1.321  0.18899
## symboling2             0.119796   0.102094   1.173  0.24296
## symboling3             0.204663   0.113233   1.807  0.07320 .
## fueltypegas            0.076304   0.486142   0.157  0.87554
## aspirationturbo        0.083430   0.059482   1.403  0.16331
## doornumbertwo          0.009889   0.042729   0.231  0.81737
## carbodyhardtop        -0.192437   0.095301  -2.019  0.04569 *
## carbodyhatchback      -0.250431   0.082196  -3.047  0.00284 **
## carbodysedan          -0.170858   0.090356  -1.891  0.06104 .
## carbodywagon          -0.291128   0.099294  -2.932  0.00403 **
## drivewheelfwd         -0.079084   0.072410  -1.092  0.27695
## drivewheelrwd          0.033789   0.085503   0.395  0.69342
## enginelocationrear     0.388767   0.192899   2.015  0.04610 *
## enginetypedohcv       -0.398564   0.328925  -1.212  0.22800
## enginetypel           -0.134140   0.121313  -1.106  0.27106
## enginetypeohc          0.140677   0.065698   2.141  0.03428 *
## enginetypeohcf        -0.027966   0.116094  -0.241  0.81005
## enginetypeohcv        -0.132525   0.086646  -1.529  0.12877
## enginetyperotor        0.109032   0.363413   0.300  0.76468
## cylindernumberfive    -0.155155   0.219132  -0.708  0.48029
## cylindernumberfour    -0.189966   0.266048  -0.714  0.47660
## cylindernumbersix     -0.116923   0.168888  -0.692  0.49008
## cylindernumbertwelve  -0.571916   0.323453  -1.768  0.07958 .
## cylindernumbertwo            NA         NA      NA       NA
## fuelsystem2bbl        -0.063323   0.064239  -0.986  0.32624
## fuelsystem4bbl        -0.036442   0.177097  -0.206  0.83732
## fuelsystemidi                NA         NA      NA       NA
## fuelsystemmfi         -0.003715   0.167420  -0.022  0.98233
## fuelsystemmpfi         0.058838   0.070739   0.832  0.40720
## fuelsystemspdi        -0.060449   0.101477  -0.596  0.55251
## wheelbase              0.037549   0.042062   0.893  0.37380
## carlength             -0.017090   0.042155  -0.405  0.68590
## carwidth               0.093417   0.036753   2.542  0.01230 *
## carheight              0.013643   0.023177   0.589  0.55721
## curbweight             0.124914   0.061279   2.038  0.04370 *
## enginesize             0.113454   0.094804   1.197  0.23378
## boreratio             -0.018352   0.044878  -0.409  0.68331
## stroke                -0.053306   0.025568  -2.085  0.03920 *
## compressionratio       0.058973   0.155600   0.379  0.70536
## horsepower             0.095439   0.063539   1.502  0.13571
## peakrpm                0.019815   0.021956   0.903  0.36860
## citympg               -0.171641   0.066563  -2.579  0.01113 *
## highwaympg             0.138597   0.064556   2.147  0.03381 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1364 on 120 degrees of freedom
```

```
## Multiple R-squared:  0.9462, Adjusted R-squared:  0.9269
## F-statistic: 49.04 on 43 and 120 DF,  p-value: < 2.2e-16
```

```
# Running diagnostics and looking for outliers (using a cutoff of 2 stdev)
par(mfrow = c(2,2))
plot(mod.1)
```

```
## Warning: not plotting observations with leverage one:
##   23, 39, 48, 104
```




```
plot(data.train$price, predict(mod.1,newdata = data.train),
     col=4, cex=0.3, xlab="Actual", ylab="In.sample fits", axes=FALSE)
```

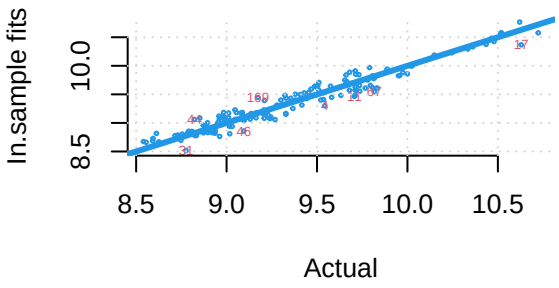
```
## Warning in predict.lm(mod.1, newdata = data.train): prediction from a rank-
## deficient fit may be misleading
```

```
extpts <- which(abs(residuals(mod.1)) > 2*sd(residuals(mod.1)))
text(data.train$price[extpts],
     predict(mod.1,newdata = data.train)[extpts],
     rownames(data.train)[extpts], cex=0.5, col=2)
```

```
## Warning in predict.lm(mod.1, newdata = data.train): prediction from a rank-
## deficient fit may be misleading
```

```
axis(1); axis(2); grid(); abline(0,1, col=4, lwd=3)
extpts
```

```
##   4  11  17  31  44  46  67 169
##   4   8  13  24  36  37  56 134
```



High leverage points:

We have identified 11 high leverage points in the training dataset of carprices. A datapoint is considered high leverage when it is far removed from x-bar or it is an outlier in the x space. Alternatively, we can define these points as extreme values which might be particularly high or low for one or more predictors, or may be "unusual" combinations of predictor values.

```
# High leverage points
n <- nrow(data.train)
p <- ncol(data.train)-1
(hilev <- which(influence(mod.1)$hat > max(2*(p+1)/n,0.5)))
```

```
##   3  30  49  50  59  73  74 128 129 130 156
##   3  23  38  39  48  61  62 102 103 104 124
```

```
length(hilev)
```

```
## [1] 11
```

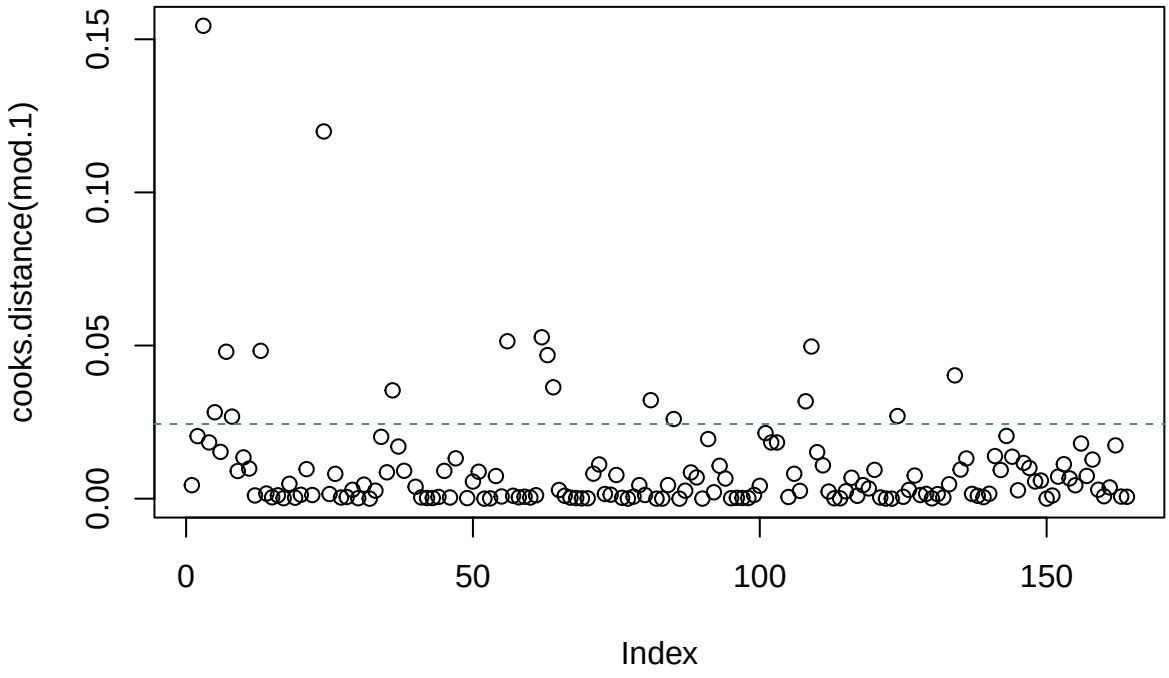
Influential points:

As for CooksD, the points above the 4/n threshold in the scatter are identified as the influential points which need to be investigated further because they may be negatively affecting the regression model. The CooksD measures how much all of the fitted values in the model change when the ith datapoint is deleted. Below, we have identified 17 points using CooksD that are highly influential. With regards to DFFITS, it indicates the datapoints that are influential in changing the in-sample fits in case of their omission. We identified 18 such points.

```
# Influential points
(hiCookD <- which(cooks.distance(mod.1) > min(qf(0.95,p,n-p), 4/n)))
```

```
##   3   5  10  11  17  31  44  67  74  75  76  99 104 137 138 156 169
##   3   5   7   8  13  24  36  56  62  63  64  81  85 108 109 124 134
```

```
plot(cooks.distance(mod.1))
abline(h=4/n, lty=2, col="steelblue")
```




```
(hiDFFITS <- which(dffits(mod.1) > qt(0.975,n-p-1)*sqrt(p/(n-p))))
```

```
##   2   3   4   5   6  11  17  31  41  46  67  75  99 126 128 137 138 203
##   2   3   4   5   6   8  13  24  34  37  56  63  81 101 102 108 109 162
```

Detecting multicollinearity using Variance Inflation Factor (VIF):

A VIF greater than 10 indicates multicollinearity. There are 13 variables that have VIFs greater than 10. The solution to this problem is to either drop one of these predictor variables from the regression model and fit the model again or move towards ridge regression.

```
# Running a check if multicollineartity exists in fitting an MLR model to the
# response Y using all the predictors
contpred.df <- data.train[,contpredcols]
cor.pred <- cor(contpred.df)
off.diag <- function(x) x[col(x) > row(x)]
v <- off.diag(cor.pred)
table(v >=0.95)
```

```
##
## FALSE  TRUE
##    77     1
```

```
# Removing variables that have alias coefficients
attributes(alias(mod.1)$Complete)$dimnames[[1]]
```

```
## [1] "cylindernumbertwo" "fuelsystemidi"
```

```
data.train.1 <- subset(data.train, select = -c(cylindernumber, fuelsystem))

# Fit the MLR model again as mod.2
mod.2 <- lm(price ~., data=data.train.1)
summary(mod.2)
```

```
##
## Call:
## lm(formula = price ~ ., data = data.train.1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.29262 -0.07818 -0.02263  0.08061  0.33135
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         9.7777884  0.3716680  26.308  < 2e-16 ***
## symboling-1         0.1286733  0.0984254   1.307 0.193430
## symboling0          0.2373631  0.0969330   2.449 0.015679 *
## symboling1          0.1451541  0.1041691   1.393 0.165881
## symboling2          0.1807320  0.1037609   1.742 0.083925 .
## symboling3          0.2458876  0.1130709   2.175 0.031482 *
## fueltypegas        -0.4625560  0.3775825  -1.225 0.222790
## aspirationturbo     0.0267556  0.0578484   0.463 0.644493
## doornumbertwo       0.0008425  0.0431159   0.020 0.984440
## carbodyhardtop     -0.2272464  0.0960808  -2.365 0.019513 *
## carbodyhatchback   -0.3101760  0.0796272  -3.895 0.000157 ***
## carbodysedan       -0.2333841  0.0879254  -2.654 0.008947 **
## carbodywagon       -0.3725987  0.0981923  -3.795 0.000226 ***
## drivewheelfwd      -0.0557567  0.0732998  -0.761 0.448244
## drivewheelrwd       0.1147338  0.0817623   1.403 0.162942
```

```
## enginelocationrear  0.4922910  0.1863223   2.642 0.009258 **
## enginetypedohcv    -0.2413118  0.2292552  -1.053 0.294497
## enginetypel        -0.2951656  0.1065379  -2.771 0.006424 **
## enginetypeohc       0.0837006  0.0637882   1.312 0.191795
## enginetypeohcf     -0.0285894  0.1125142  -0.254 0.799826
## enginetypeohcv     -0.0718160  0.0839820  -0.855 0.394061
## enginetyperotor     0.1464244  0.1184432   1.236 0.218615
## wheelbase           0.0713749  0.0422175   1.691 0.093319 .
## carlength          -0.0252588  0.0427110  -0.591 0.555295
## carwidth            0.1137187  0.0353563   3.216 0.001641 **
## carheight           0.0297033  0.0218429   1.360 0.176247
## curbweight          0.1397579  0.0616825   2.266 0.025132 *
## enginesize          0.0572135  0.0569352   1.005 0.316832
## boreratio          -0.0334014  0.0249386  -1.339 0.182814
## stroke             -0.0450848  0.0202897  -2.222 0.028023 *
## compressionratio   -0.1069876  0.1147680  -0.932 0.352970
## horsepower          0.1218144  0.0521009   2.338 0.020923 *
## peakrpm             0.0278622  0.0203169   1.371 0.172637
## citympg            -0.1600272  0.0674522  -2.372 0.019148 *
## highwaympg          0.1271999  0.0659970   1.927 0.056133 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1423 on 129 degrees of freedom
## Multiple R-squared:  0.937,  Adjusted R-squared:  0.9204
## F-statistic: 56.44 on 34 and 129 DF,  p-value: < 2.2e-16
```

```
# Check for multicollinearity in mod.2 using vif
vif(mod.2)
```

```
##                       GVIF Df GVIF^(1/(2*Df))
## symboling         21.400901  5        1.358449
## fueltype         112.892697  1       10.625098
## aspiration         3.947580  1        1.986852
## doornumber         3.746181  1        1.935505
## carbody           12.824262  4        1.375638
## drivewheel         9.183719  2        1.740823
## enginelocation     3.389154  1        1.840966
## enginetype       313.427146  6        1.614401
## wheelbase         14.355971  1        3.788927
## carlength         14.693588  1        3.833222
## carwidth          10.068877  1        3.173149
## carheight          3.842985  1        1.960353
## curbweight        30.645932  1        5.535877
## enginesize        26.110165  1        5.109811
## boreratio          5.009483  1        2.238187
## stroke             3.315901  1        1.820961
## compressionratio 106.093686  1       10.300179
## horsepower        21.864419  1        4.675940
## peakrpm            3.324790  1        1.823401
## citympg           36.647153  1        6.053689
## highwaympg        35.083010  1        5.923091
```

Remedies for Multicollinearity:

There are two solutions to Multicollinearity.

Solution # 1 Dropping predictors from the model:

It is not clear how the omission of a variable will affect the estimates of the remaining model parameters. We refit the model after excluding one of the predictors with the largest VIF. We can repeat this process until we get VIFs less than 10 for all the predictors, indicating that the issue of multicollinearity has been addressed. Below, we excluded a total of 5 variables with high VIFs until we reached a model that is free of multicollinear predictors.

```
# Solution # 1 for multicollinearity
# Fit MLR by omitting some collinear variables

# Drop engine type
data.train.2 <- subset(data.train.1, select = -c(enginetype))
mod.dropenginetype <- lm(price ~., data = data.train.2)
summary(mod.dropenginetype)
```

```
##
## Call:
## lm(formula = price ~ ., data = data.train.2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.33404 -0.08415 -0.01969  0.08182  0.43075
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         9.6021148  0.3834142  25.044  < 2e-16 ***
## symboling-1         0.0957832  0.1053970   0.909 0.365083
## symboling0          0.1071210  0.1006269   1.065 0.288986
## symboling1         -0.0007414  0.1074461  -0.007 0.994505
## symboling2          0.0347671  0.1076414   0.323 0.747203
## symboling3          0.1226566  0.1147216   1.069 0.286902
## fueltypegas         0.0130628  0.3919813   0.033 0.973465
## aspirationturbo     0.0483194  0.0573781   0.842 0.401209
## doornumbertwo       0.0170677  0.0458780   0.372 0.710459
## carbodyhardtop     -0.2062446  0.1023107  -2.016 0.045799 *
## carbodyhatchback   -0.3165080  0.0827978  -3.823 0.000201 ***
## carbodysedan       -0.2412262  0.0903939  -2.669 0.008550 **
## carbodywagon       -0.4103507  0.0994559  -4.126 6.42e-05 ***
## drivewheelfwd      -0.1443307  0.0742932  -1.943 0.054132 .
## drivewheelrwd       0.0200473  0.0762727   0.263 0.793077
## enginelocationrear  0.3680549  0.1478084   2.490 0.013985 *
## wheelbase           0.0494983  0.0417724   1.185 0.238118
## carlength           0.0085635  0.0415530   0.206 0.837033
## carwidth            0.0912431  0.0342342   2.665 0.008631 **
## carheight           0.0257923  0.0217832   1.184 0.238475
## curbweight          0.0547273  0.0639322   0.856 0.393503
## enginesize          0.1010721  0.0454731   2.223 0.027901 *
## boreratio          -0.0336597  0.0212861  -1.581 0.116150
## stroke             -0.0063234  0.0175490  -0.360 0.719166
## compressionratio    0.0456026  0.1169018   0.390 0.697082
## horsepower          0.0964784  0.0437990   2.203 0.029309 *
## peakrpm             0.0367695  0.0209070   1.759 0.080891 .
```

```
## citympg            -0.2184112  0.0695240  -3.142 0.002066 **
## highwaympg          0.1559385  0.0670145   2.327 0.021457 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.155 on 135 degrees of freedom
## Multiple R-squared:  0.9218, Adjusted R-squared:  0.9056
## F-statistic: 56.82 on 28 and 135 DF,  p-value: < 2.2e-16
```

```
anova(mod.dropenginetype)
```

```
## Analysis of Variance Table
##
## Response: price
##                   Df Sum Sq Mean Sq  F value    Pr(>F)
## symboling          5 9.7299  1.9460  81.0396 < 2.2e-16 ***
## fueltype           1 0.1554  0.1554   6.4722   0.01208 *
## aspiration         1 0.7325  0.7325  30.5041 1.652e-07 ***
## doornumber         1 0.5410  0.5410  22.5299 5.206e-06 ***
## carbody            4 5.7860  1.4465  60.2383 < 2.2e-16 ***
## drivewheel         2 9.6015  4.8008 199.9249 < 2.2e-16 ***
## enginelocation     1 0.4473  0.4473  18.6261 3.053e-05 ***
## wheelbase          1 4.7670  4.7670 198.5199 < 2.2e-16 ***
## carlength          1 1.8522  1.8522  77.1326 6.336e-15 ***
## carwidth           1 2.2683  2.2683  94.4609 < 2.2e-16 ***
## carheight          1 0.0017  0.0017   0.0698   0.79197
## curbweight         1 1.1958  1.1958  49.8001 8.084e-11 ***
## enginesize         1 0.0980  0.0980   4.0806   0.04536 *
## boreratio          1 0.0642  0.0642   2.6740   0.10433
## stroke             1 0.0002  0.0002   0.0070   0.93357
## compressionratio   1 0.0088  0.0088   0.3654   0.54656
## horsepower         1 0.6397  0.6397  26.6398 8.574e-07 ***
## peakrpm            1 0.0579  0.0579   2.4128   0.12269
## citympg            1 0.1278  0.1278   5.3232   0.02257 *
## highwaympg         1 0.1300  0.1300   5.4146   0.02146 *
## Residuals        135 3.2417  0.0240
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
car::vif(mod.dropenginetype)
```

```
##                      GVIF Df GVIF^(1/(2*Df))
## symboling         11.051934  5        1.271580
## fueltype         102.534412  1       10.125928
## aspiration         3.272937  1        1.809126
## doornumber         3.574525  1        1.890641
## carbody           10.115622  4        1.335439
## drivewheel         6.283426  2        1.583249
## enginelocation     1.797449  1        1.340690
## wheelbase         11.844693  1        3.441612
## carlength         11.720620  1        3.423539
## carwidth           7.955474  1        2.820545
## carheight          3.220970  1        1.794706
```

```
## curbweight        27.745002  1        5.267352
## enginesize        14.036361  1        3.746513
## boreratio          3.075662  1        1.753756
## stroke             2.090499  1        1.445856
## compressionratio  92.765722  1        9.631496
## horsepower        13.021872  1        3.608583
## peakrpm            2.967067  1        1.722518
## citympg           32.810632  1        5.728057
## highwaympg        30.484775  1        5.521302
```

```
# Drop fueltype
data.train.3 <- subset(data.train.2, select = -c(fueltype))
mod.dropfueltype <- lm(price ~., data = data.train.3)
summary(mod.dropfueltype)
```

```
##
## Call:
## lm(formula = price ~ ., data = data.train.3)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.33423 -0.08458 -0.01976  0.08190  0.43006
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         9.6137800  0.1558741  61.677  < 2e-16 ***
## symboling-1         0.0957761  0.1050090   0.912 0.363343
## symboling0          0.1067686  0.0997015   1.071 0.286119
## symboling1         -0.0009934  0.1067852  -0.009 0.992591
## symboling2          0.0347613  0.1072452   0.324 0.746339
## symboling3          0.1223458  0.1139212   1.074 0.284746
## aspirationturbo     0.0473308  0.0489347   0.967 0.335149
## doornumbertwo       0.0169922  0.0456535   0.372 0.710323
## carbodyhardtop     -0.2059478  0.1015473  -2.028 0.044503 *
## carbodyhatchback   -0.3163456  0.0823501  -3.841 0.000187 ***
## carbodysedan       -0.2411536  0.0900352  -2.678 0.008309 **
## carbodywagon       -0.4100855  0.0987723  -4.152 5.79e-05 ***
## drivewheelfwd      -0.1440637  0.0735881  -1.958 0.052312 .
## drivewheelrwd       0.0203750  0.0753576   0.270 0.787280
## enginelocationrear  0.3678754  0.1471667   2.500 0.013618 *
## wheelbase           0.0493391  0.0413456   1.193 0.234816
## carlength           0.0087729  0.0409243   0.214 0.830580
## carwidth            0.0912876  0.0340824   2.678 0.008309 **
## carheight           0.0257286  0.0216193   1.190 0.236091
## curbweight          0.0546028  0.0635881   0.859 0.392020
## enginesize          0.1010091  0.0452667   2.231 0.027290 *
## boreratio          -0.0337717  0.0209415  -1.613 0.109135
## stroke             -0.0065548  0.0160572  -0.408 0.683758
## compressionratio    0.0417733  0.0214136   1.951 0.053139 .
## horsepower          0.0967571  0.0428347   2.259 0.025484 *
## peakrpm             0.0368880  0.0205264   1.797 0.074539 .
## citympg            -0.2181110  0.0686843  -3.176 0.001850 **
## highwaympg          0.1558421  0.0667057   2.336 0.020939 *
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1544 on 136 degrees of freedom
## Multiple R-squared:  0.9218, Adjusted R-squared:  0.9063
## F-statistic: 59.36 on 27 and 136 DF,  p-value: < 2.2e-16
```

```
anova(mod.dropfueltype)
```

```
## Analysis of Variance Table
##
## Response: price
##                    Df Sum Sq Mean Sq  F value    Pr(>F)
## symboling           5 9.7299  1.9460  81.6392 < 2.2e-16 ***
## aspiration          1 0.8879  0.8879  37.2499 1.019e-08 ***
## doornumber          1 0.5375  0.5375  22.5485 5.131e-06 ***
## carbody             4 5.6421  1.4105  59.1747 < 2.2e-16 ***
## drivewheel          2 9.5254  4.7627 199.8068 < 2.2e-16 ***
## enginelocation      1 0.4403  0.4403  18.4721 3.263e-05 ***
## wheelbase           1 4.6182  4.6182 193.7454 < 2.2e-16 ***
## carlength           1 2.0501  2.0501  86.0072 3.657e-16 ***
## carwidth            1 2.1718  2.1718  91.1116 < 2.2e-16 ***
## carheight           1 0.0219  0.0219   0.9192  0.339375
## curbweight          1 1.2372  1.2372  51.9026 3.623e-11 ***
## enginesize          1 0.1457  0.1457   6.1123  0.014659 *
## boreratio           1 0.0319  0.0319   1.3398  0.249092
## stroke              1 0.0049  0.0049   0.2067  0.650111
## compressionratio    1 0.1816  0.1816   7.6187  0.006574 **
## horsepower          1 0.6613  0.6613  27.7413 5.291e-07 ***
## peakrpm             1 0.0531  0.0531   2.2286  0.137795
## citympg             1 0.1343  0.1343   5.6346  0.019008 *
## highwaympg          1 0.1301  0.1301   5.4581  0.020939 *
## Residuals         136 3.2418  0.0238
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
car::vif(mod.dropfueltype)
```

```
##                       GVIF Df GVIF^(1/(2*Df))
## symboling         9.982000  5        1.258699
## aspiration        2.398174  1        1.548604
## doornumber        3.565824  1        1.888339
## carbody           9.863962  4        1.331240
## drivewheel        6.178650  2        1.576607
## enginelocation    1.795061  1        1.339799
## wheelbase        11.689746  1        3.419027
## carlength        11.452713  1        3.384186
## carwidth          7.943391  1        2.818402
## carheight         3.196176  1        1.787785
## curbweight       27.650224  1        5.258348
## enginesize       14.012112  1        3.743276
## boreratio         2.998905  1        1.731735
## stroke            1.763138  1        1.327832
## compressionratio  3.135629  1        1.770771
```

```
## horsepower       12.546977  1        3.542171
## peakrpm           2.881198  1        1.697409
## citympg          32.259822  1        5.679773
## highwaympg       30.427976  1        5.516156
```

```
# Drop citympg
data.train.4 <- subset(data.train.3, select = -c(citympg))
mod.dropcitympg <- lm(price ~., data = data.train.4)
summary(mod.dropcitympg)
```

```
##
## Call:
## lm(formula = price ~ ., data = data.train.4)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.32334 -0.08163 -0.00690  0.07502  0.43556
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)         9.526052   0.158411  60.135  < 2e-16 ***
## symboling-1         0.132711   0.107767   1.231 0.220263
## symboling0          0.160230   0.101476   1.579 0.116642
## symboling1          0.055287   0.108739   0.508 0.611964
## symboling2          0.104355   0.108407   0.963 0.337430
## symboling3          0.192044   0.115434   1.664 0.098463 .
## aspirationturbo    -0.001021   0.048023  -0.021 0.983077
## doornumbertwo       0.007537   0.047042   0.160 0.872940
## carbodyhardtop     -0.196241   0.104812  -1.872 0.063295 .
## carbodyhatchback   -0.304989   0.084956  -3.590 0.000460 ***
## carbodysedan       -0.220650   0.092733  -2.379 0.018719 *
## carbodywagon       -0.423508   0.101901  -4.156 5.67e-05 ***
## drivewheelfwd      -0.105690   0.074957  -1.410 0.160803
## drivewheelrwd       0.045277   0.077393   0.585 0.559494
## enginelocationrear  0.426391   0.150772   2.828 0.005386 **
## wheelbase           0.038085   0.042537   0.895 0.372185
## carlength           0.019011   0.042128   0.451 0.652513
## carwidth            0.087774   0.035176   2.495 0.013772 *
## carheight           0.027883   0.022314   1.250 0.213581
## curbweight          0.086528   0.064837   1.335 0.184237
## enginesize          0.041220   0.042508   0.970 0.333903
## boreratio          -0.023970   0.021388  -1.121 0.264384
## stroke              0.004087   0.016216   0.252 0.801377
## compressionratio    0.027499   0.021619   1.272 0.205546
## horsepower          0.151157   0.040541   3.729 0.000281 ***
## peakrpm             0.024986   0.020840   1.199 0.232614
## highwaympg         -0.031353   0.032239  -0.973 0.332498
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1594 on 137 degrees of freedom
## Multiple R-squared:  0.916,  Adjusted R-squared:    0.9
## F-statistic: 57.45 on 26 and 137 DF,  p-value: < 2.2e-16
```

```r
anova(mod.dropcitympg)
```

```
## Analysis of Variance Table
##
## Response: price
##                   Df Sum Sq Mean Sq  F value    Pr(>F)
## symboling          5 9.7299  1.9460  76.5625 < 2.2e-16 ***
## aspiration         1 0.8879  0.8879  34.9335 2.573e-08 ***
## doornumber         1 0.5375  0.5375  21.1463 9.576e-06 ***
## carbody            4 5.6421  1.4105  55.4950 < 2.2e-16 ***
## drivewheel         2 9.5254  4.7627 187.3820 < 2.2e-16 ***
## enginelocation     1 0.4403  0.4403  17.3234 5.540e-05 ***
## wheelbase          1 4.6182  4.6182 181.6974 < 2.2e-16 ***
## carlength          1 2.0501  2.0501  80.6589 1.866e-15 ***
## carwidth           1 2.1718  2.1718  85.4458 4.132e-16 ***
## carheight          1 0.0219  0.0219   0.8621  0.354790
## curbweight         1 1.2372  1.2372  48.6751 1.177e-10 ***
## enginesize         1 0.1457  0.1457   5.7322  0.018010 *
## boreratio          1 0.0319  0.0319   1.2565  0.264272
## stroke             1 0.0049  0.0049   0.1938  0.660447
## compressionratio   1 0.1816  0.1816   7.1449  0.008431 **
## horsepower         1 0.6613  0.6613  26.0162 1.106e-06 ***
## peakrpm            1 0.0531  0.0531   2.0900  0.150551
## highwaympg         1 0.0240  0.0240   0.9458  0.332498
## Residuals        137 3.4821  0.0254
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
car::vif(mod.dropcitympg)
```

```
##                      GVIF Df GVIF^(1/(2*Df))
## symboling        9.412889  5        1.251331
## aspiration       2.165994  1        1.471732
## doornumber       3.550657  1        1.884319
## carbody          9.450064  4        1.324126
## drivewheel       5.998865  2        1.565011
## enginelocation   1.766918  1        1.329255
## wheelbase       11.603852  1        3.406443
## carlength       11.381635  1        3.373668
## carwidth         7.935020  1        2.816917
## carheight        3.193029  1        1.786905
## curbweight      26.959076  1        5.192213
## enginesize      11.588039  1        3.404121
## boreratio        2.933751  1        1.712820
## stroke           1.686339  1        1.298591
## compressionratio 2.997455  1        1.731316
## horsepower      10.540175  1        3.246564
## peakrpm          2.785135  1        1.668873
## highwaympg       6.665225  1        2.581710
```

```r
# Drop curbweight
data.train.5 <- subset(data.train.4, select = -c(curbweight))
```

```r
mod.dropcurbweight <- lm(price ~., data = data.train.5)
summary(mod.dropcurbweight)
```

```
##
## Call:
## lm(formula = price ~ ., data = data.train.5)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.31743 -0.08355 -0.00649  0.08623  0.44403
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         9.5768979  0.1541953  62.109  < 2e-16 ***
## symboling-1         0.1247604  0.1079067   1.156  0.24960
## symboling0          0.1589973  0.1017581   1.563  0.12046
## symboling1          0.0460415  0.1088252   0.423  0.67290
## symboling2          0.1004957  0.1086745   0.925  0.35672
## symboling3          0.1915012  0.1157591   1.654  0.10034
## aspirationturbo     0.0159498  0.0464394   0.343  0.73178
## doornumbertwo      -0.0008726  0.0467502  -0.019  0.98514
## carbodyhardtop     -0.2325292  0.1015100  -2.291  0.02350 *
## carbodyhatchback   -0.3305513  0.0830027  -3.982  0.00011 ***
## carbodysedan       -0.2522146  0.0899194  -2.805  0.00576 **
## carbodywagon       -0.4304257  0.1020566  -4.218 4.45e-05 ***
## drivewheelfwd      -0.1384075  0.0710349  -1.948  0.05339 .
## drivewheelrwd       0.0417852  0.0775678   0.539  0.59097
## enginelocationrear  0.4002555  0.1499167   2.670  0.00850 **
## wheelbase           0.0399813  0.0426336   0.938  0.34999
## carlength           0.0388523  0.0395287   0.983  0.32738
## carwidth            0.1026776  0.0334502   3.070  0.00258 **
## carheight           0.0296653  0.0223365   1.328  0.18634
## enginesize          0.0668802  0.0380184   1.759  0.08077 .
## boreratio          -0.0278818  0.0212465  -1.312  0.19160
## stroke              0.0079909  0.0159950   0.500  0.61816
## compressionratio    0.0342304  0.0210822   1.624  0.10673
## horsepower          0.1624319  0.0397628   4.085 7.44e-05 ***
## peakrpm             0.0237891  0.0208792   1.139  0.25652
## highwaympg         -0.0409600  0.0315134  -1.300  0.19585
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1599 on 138 degrees of freedom
## Multiple R-squared:  0.9149, Adjusted R-squared:  0.8995
## F-statistic: 59.34 on 25 and 138 DF,  p-value: < 2.2e-16
```

```r
anova(mod.dropcurbweight)
```

```
## Analysis of Variance Table
##
## Response: price
##                   Df Sum Sq Mean Sq  F value    Pr(>F)
```

```
## symboling          5 9.7299  1.9460  76.1316 < 2.2e-16 ***
## aspiration         1 0.8879  0.8879  34.7369 2.754e-08 ***
## doornumber         1 0.5375  0.5375  21.0273 1.005e-05 ***
## carbody            4 5.6421  1.4105  55.1826 < 2.2e-16 ***
## drivewheel         2 9.5254  4.7627 186.3274 < 2.2e-16 ***
## enginelocation     1 0.4403  0.4403  17.2260 5.776e-05 ***
## wheelbase          1 4.6182  4.6182 180.6749 < 2.2e-16 ***
## carlength          1 2.0501  2.0501  80.2050 2.058e-15 ***
## carwidth           1 2.1718  2.1718  84.9650 4.562e-16 ***
## carheight          1 0.0219  0.0219   0.8572  0.356132
## enginesize         1 1.0976  1.0976  42.9395 1.037e-09 ***
## boreratio          1 0.0846  0.0846   3.3112  0.070978 .
## stroke             1 0.0001  0.0001   0.0024  0.961079
## compressionratio   1 0.1790  0.1790   7.0011  0.009092 **
## horsepower         1 0.8363  0.8363  32.7164 6.344e-08 ***
## peakrpm            1 0.0538  0.0538   2.1055  0.149039
## highwaympg         1 0.0432  0.0432   1.6894  0.195848
## Residuals        138 3.5274  0.0256
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
car::vif(mod.dropcurbweight)
```

```
##                      GVIF Df GVIF^(1/(2*Df))
## symboling         9.101685  5        1.247131
## aspiration        2.014124  1        1.419198
## doornumber        3.486942  1        1.867335
## carbody           7.860868  4        1.293999
## drivewheel        4.493323  2        1.455935
## enginelocation    1.737108  1        1.317994
## wheelbase        11.590897  1        3.404541
## carlength         9.964079  1        3.156593
## carwidth          7.135251  1        2.671189
## carheight         3.181588  1        1.783701
## enginesize        9.217215  1        3.035987
## boreratio         2.878641  1        1.696656
## stroke            1.631471  1        1.277290
## compressionratio  2.834292  1        1.683536
## horsepower       10.082446  1        3.175287
## peakrpm           2.779978  1        1.667327
## highwaympg        6.332905  1        2.516526
```

```
# Drop wheelbase
data.train.6 <- subset(data.train.5, select = -c(wheelbase))
mod.dropwheelbase <- lm(price ~., data = data.train.6)
summary(mod.dropwheelbase)
```

```
##
## Call:
## lm(formula = price ~ ., data = data.train.6)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -0.32694 -0.08064 -0.00539  0.08727  0.43425
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         9.539522   0.148891  64.071  < 2e-16 ***
## symboling-1         0.141037   0.106455   1.325 0.187396
## symboling0          0.167604   0.101299   1.655 0.100274
## symboling1          0.049669   0.108709   0.457 0.648458
## symboling2          0.102694   0.108602   0.946 0.345993
## symboling3          0.183077   0.115360   1.587 0.114783
## aspirationturbo     0.025413   0.045310   0.561 0.575783
## doornumbertwo      -0.003059   0.046672  -0.066 0.947844
## carbodyhardtop     -0.218994   0.100435  -2.180 0.030907 *
## carbodyhatchback   -0.310629   0.080203  -3.873 0.000165 ***
## carbodysedan       -0.235495   0.088096  -2.673 0.008413 **
## carbodywagon       -0.417695   0.101106  -4.131 6.20e-05 ***
## drivewheelfwd      -0.128171   0.070161  -1.827 0.069873 .
## drivewheelrwd       0.066719   0.072837   0.916 0.361252
## enginelocationrear  0.393762   0.149692   2.630 0.009488 **
## carlength           0.053581   0.036259   1.478 0.141739
## carwidth            0.115598   0.030468   3.794 0.000220 ***
## carheight           0.037990   0.020488   1.854 0.065818 .
## enginesize          0.075586   0.036852   2.051 0.042137 *
## boreratio          -0.028427   0.021229  -1.339 0.182743
## stroke              0.009059   0.015947   0.568 0.570902
## compressionratio    0.031978   0.020936   1.527 0.128923
## horsepower          0.147151   0.036255   4.059 8.19e-05 ***
## peakrpm             0.025744   0.020766   1.240 0.217161
## highwaympg         -0.044790   0.031234  -1.434 0.153820
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1598 on 139 degrees of freedom
## Multiple R-squared:  0.9144, Adjusted R-squared:  0.8996
## F-statistic: 61.83 on 24 and 139 DF,  p-value: < 2.2e-16
```

```
anova(mod.dropwheelbase)
```

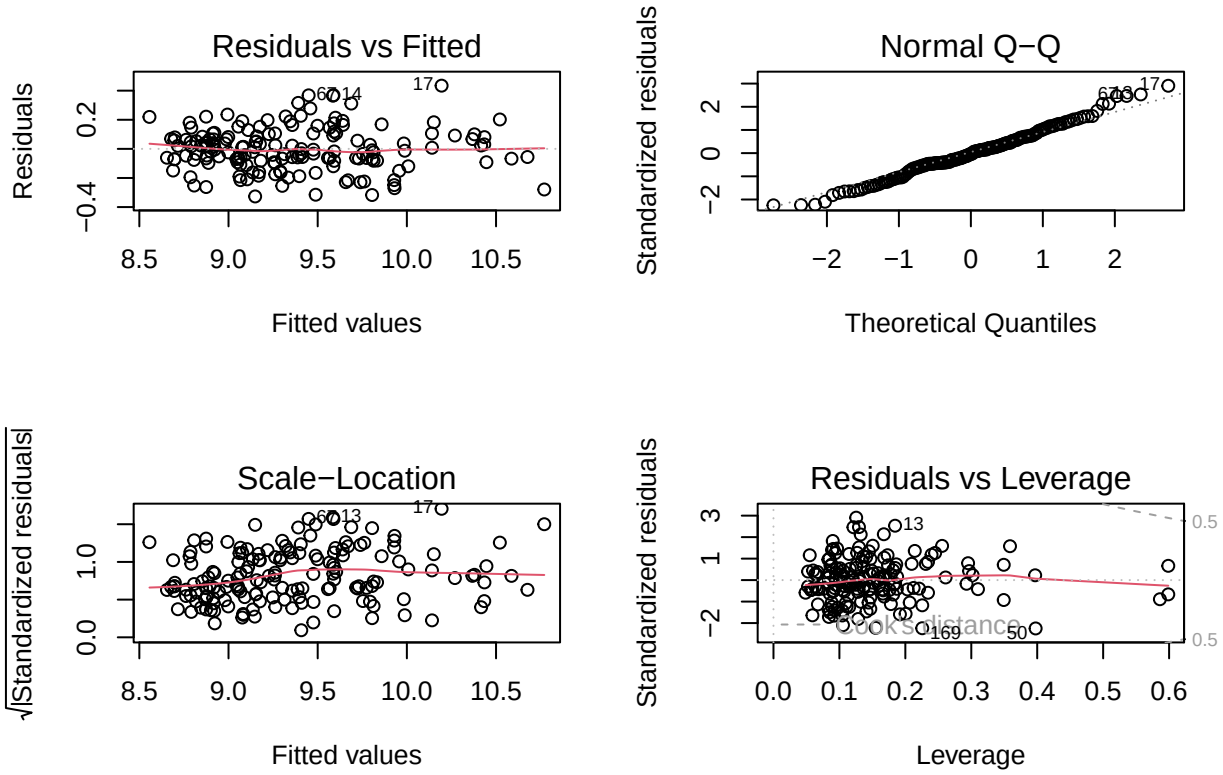
```
## Analysis of Variance Table
##
## Response: price
##                   Df Sum Sq Mean Sq  F value    Pr(>F)
## symboling          5 9.7299  1.9460  76.1977 < 2.2e-16 ***
## aspiration         1 0.8879  0.8879  34.7671 2.687e-08 ***
## doornumber         1 0.5375  0.5375  21.0456 9.917e-06 ***
## carbody            4 5.6421  1.4105  55.2305 < 2.2e-16 ***
## drivewheel         2 9.5254  4.7627 186.4892 < 2.2e-16 ***
## enginelocation     1 0.4403  0.4403  17.2409 5.715e-05 ***
## carlength          1 6.6008  6.6008 258.4641 < 2.2e-16 ***
## carwidth           1 2.2091  2.2091  86.5016 2.682e-16 ***
## carheight          1 0.0440  0.0440   1.7233   0.19143
## enginesize         1 1.0926  1.0926  42.7828 1.084e-09 ***
## boreratio          1 0.0788  0.0788   3.0840   0.08127 .
## stroke             1 0.0001  0.0001   0.0021   0.96394
```

```
## compressionratio   1 0.1736  0.1736   6.7982   0.01012 *
## horsepower         1 0.8160  0.8160  31.9529 8.629e-08 ***
## peakrpm            1 0.0664  0.0664   2.6009   0.10907
## highwaympg         1 0.0525  0.0525   2.0563   0.15382
## Residuals        139 3.5499  0.0255
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
car::vif(mod.dropwheelbase)
```

```
##                      GVIF Df GVIF^(1/(2*Df))
## symboling        7.973407  5        1.230735
## aspiration       1.919015  1        1.385285
## doornumber       3.478273  1        1.865013
## carbody          7.269161  4        1.281402
## drivewheel       3.704297  2        1.387319
## enginelocation   1.733403  1        1.316588
## carlength        8.391046  1        2.896730
## carwidth         5.924826  1        2.434096
## carheight        2.679055  1        1.636782
## enginesize       8.667684  1        2.944093
## boreratio        2.876487  1        1.696021
## stroke           1.623193  1        1.274046
## compressionratio 2.797518  1        1.672578
## horsepower       8.389342  1        2.896436
## peakrpm          2.752259  1        1.658994
## highwaympg       6.226565  1        2.495309
```

```
# Running diagnostics for mod.dropwheelbase
par(mfrow=c(2,2))
plot(mod.dropwheelbase)
```



Solution # 2 Ridge Regression:

Ridge Regression is used to fit a regression model when multicollinearity is present in the data. A centered and scaled MLR model minimizes the sum of squared residuals. Meanwhile, ridge regression seeks to minimize error sum of squares subject to a penalty function denoted by

$$\lambda \parallel \beta \parallel^2]. \tag{14}$$

Looking at the output of ridge regression model, we can see that the coefficients are shrunk towards zero but no sparsity has been achieved because none of the coefficients becomes exactly zero. It has a r-squared of 89%, which means that 89% of the variation in price is explained by the predictors.

```
# Solution # 2 for multicollinearity
# Ridge regression
pred.df <- data.train[,-1]
pred.mat <- data.matrix(pred.df)
resp <- data.train$price
mod.ridge.1 <- glmnet(pred.mat, resp, alpha=0, standardize=FALSE)
summary(mod.ridge.1)
```

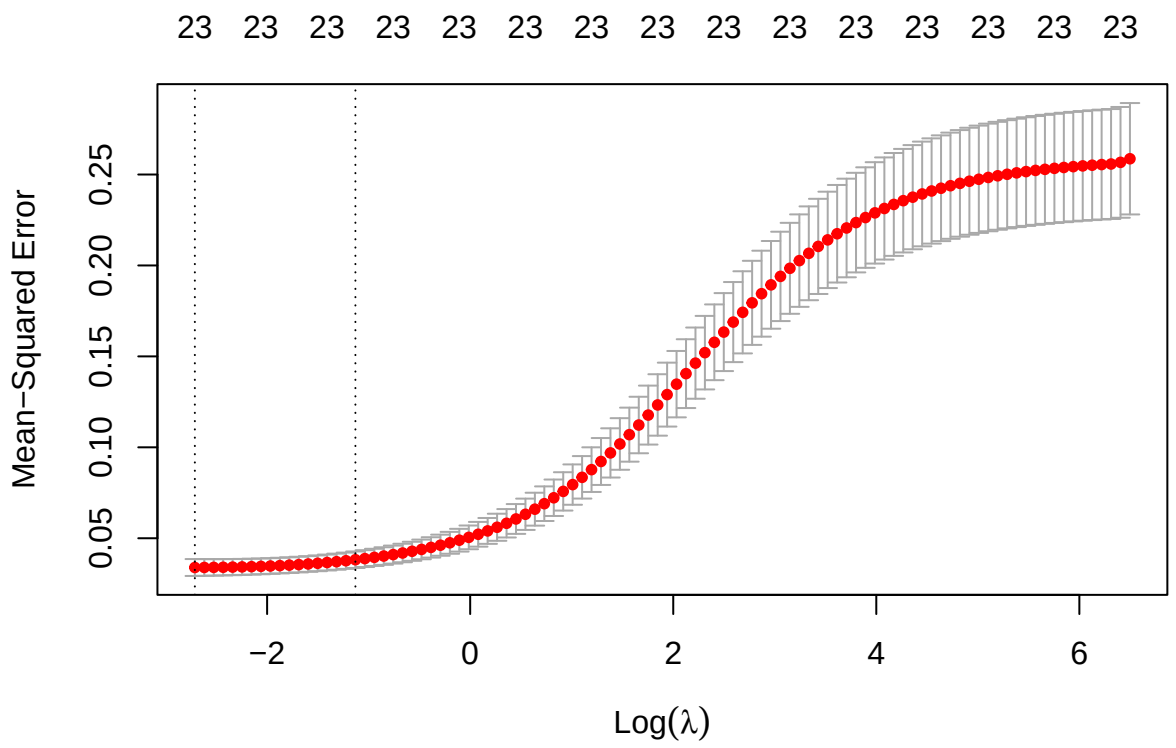
```
##           Length Class     Mode
## a0         100   -none-    numeric
## beta      2300   dgCMatrix S4
## df         100   -none-    numeric
## dim          2   -none-    numeric
```

```
## lambda     100   -none-    numeric
## dev.ratio  100   -none-    numeric
## nulldev      1   -none-    numeric
## npasses      1   -none-    numeric
## jerr         1   -none-    numeric
## offset       1   -none-    logical
## call         5   -none-    call
## nobs         1   -none-    numeric
```

```
cvfit.ridge <- cv.glmnet(pred.mat,resp,alpha=0,
                         standardize=FALSE,
                         type.measure = "mse", nfolds = 10)
best_lambda <- cvfit.ridge$lambda.min
best_lambda
```

```
## [1] 0.06640477
```

```
plot(cvfit.ridge)
```




```
best.mod.ridge <- glmnet(pred.mat, resp, alpha=0, standardize=FALSE, lambda=best_lambda)
coef(best.mod.ridge)
```

```
## 24 x 1 sparse Matrix of class "dgCMatrix"
##                               s0
```

```
## (Intercept)       9.3024311025
## symboling        -0.0004374836
## fueltype         -0.0190414627
## aspiration       -0.0010273227
## doornumber       -0.0324620726
## carbody          -0.0575184890
## drivewheel        0.0544210975
## enginelocation    0.0352042113
## enginetype        0.0142498635
## cylindernumber    0.0051825232
## fuelsystem        0.0264611714
## wheelbase         0.0191056671
## carlength         0.0320316982
## carwidth          0.0493353282
## carheight         0.0220263683
## curbweight        0.0901735043
## enginesize        0.0918222759
## boreratio         0.0022335544
## stroke           -0.0176247571
## compressionratio  0.0589199550
## horsepower        0.0943496037
## peakrpm           0.0346474180
## citympg          -0.0627807425
## highwaympg       -0.0185413692
```

```
y_predicted.ridge <- predict(best.mod.ridge, s=best_lambda, newx=pred.mat)
sst.ridge <- sum((resp - mean(resp))^2)
sse.ridge <- sum((y_predicted.ridge - resp)^2)
rsq.ridge <- 1-sse.ridge/sst.ridge
rsq.ridge
```

```
## [1] 0.8908044
```

Variable selection:

Stepwise Regression:

Stepwise Regression prepares a regression model by entering and removing predictor variables in a stepwise manner until there is no statistical valid reason to enter or remove anymore. It includes the predictor variables that are signficantly related to the target. The selections can be made forward, backward or in both directions. We have implemented the third approach below. We see that this method selects a model with 14 out of 23 predictors. It means that 14 out of 23 predictors are significant in explaining the response variable that is price.

```
# Conducting Variable Selection
# Stepwise Regression
fit.step <- lm(price ~., data = data.train)
mod.step <- step(fit.step, direction = "both", trace = 0)
summary(mod.step) # Selects 14 out of 23
```

```
##
## Call:
## lm(formula = price ~ symboling + carbody + drivewheel + enginelocation +
##     enginetype + cylindernumber + fuelsystem + wheelbase + carwidth +
```

```
##     curbweight + stroke + horsepower + citympg + highwaympg,
##     data = data.train)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.292232 -0.084779 -0.007079  0.092346  0.266832
##
## Coefficients: (1 not defined because of singularities)
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)            9.648963   0.201591  47.864  < 2e-16 ***
## symboling-1            0.145147   0.092293   1.573 0.118264
## symboling0             0.234154   0.089795   2.608 0.010200 *
## symboling1             0.171644   0.094574   1.815 0.071876 .
## symboling2             0.149882   0.094477   1.586 0.115109
## symboling3             0.234208   0.104520   2.241 0.026763 *
## carbodyhardtop        -0.188365   0.091505  -2.059 0.041569 *
## carbodyhatchback      -0.264296   0.078657  -3.360 0.001027 **
## carbodysedan          -0.183822   0.081500  -2.255 0.025800 *
## carbodywagon          -0.305103   0.088331  -3.454 0.000749 ***
## drivewheelfwd         -0.080210   0.064923  -1.235 0.218920
## drivewheelrwd          0.024635   0.068787   0.358 0.720831
## enginelocationrear     0.422296   0.159238   2.652 0.009014 **
## enginetypedohcv       -0.595838   0.243865  -2.443 0.015916 *
## enginetypel           -0.134154   0.092841  -1.445 0.150906
## enginetypeohc          0.154533   0.056992   2.711 0.007619 **
## enginetypeohcf        -0.035378   0.083439  -0.424 0.672274
## enginetypeohcv        -0.106525   0.081258  -1.311 0.192221
## enginetyperotor       -0.134681   0.206803  -0.651 0.516050
## cylindernumberfive    -0.272229   0.134070  -2.030 0.044378 *
## cylindernumberfour    -0.336106   0.134588  -2.497 0.013783 *
## cylindernumbersix     -0.215131   0.115416  -1.864 0.064618 .
## cylindernumbertwelve -0.473609   0.189816  -2.495 0.013865 *
## cylindernumbertwo            NA         NA      NA       NA
## fuelsystem2bbl        -0.101850   0.051318  -1.985 0.049317 *
## fuelsystem4bbl        -0.061693   0.166364  -0.371 0.711377
## fuelsystemidi          0.107776   0.073121   1.474 0.142955
## fuelsystemmfi         -0.033819   0.156150  -0.217 0.828882
## fuelsystemmpfi         0.009514   0.060040   0.158 0.874342
## fuelsystemspdi        -0.094519   0.085540  -1.105 0.271245
## wheelbase              0.051002   0.033579   1.519 0.131255
## carwidth               0.086699   0.033550   2.584 0.010884 *
## curbweight             0.138507   0.050907   2.721 0.007419 **
## stroke                -0.045691   0.018846  -2.424 0.016726 *
## horsepower             0.157715   0.043716   3.608 0.000441 ***
## citympg               -0.138924   0.061232  -2.269 0.024955 *
## highwaympg             0.110875   0.060631   1.829 0.069777 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1339 on 128 degrees of freedom
## Multiple R-squared:  0.9446, Adjusted R-squared:  0.9295
## F-statistic: 62.39 on 35 and 128 DF,  p-value: < 2.2e-16
```

Regularized Regression:

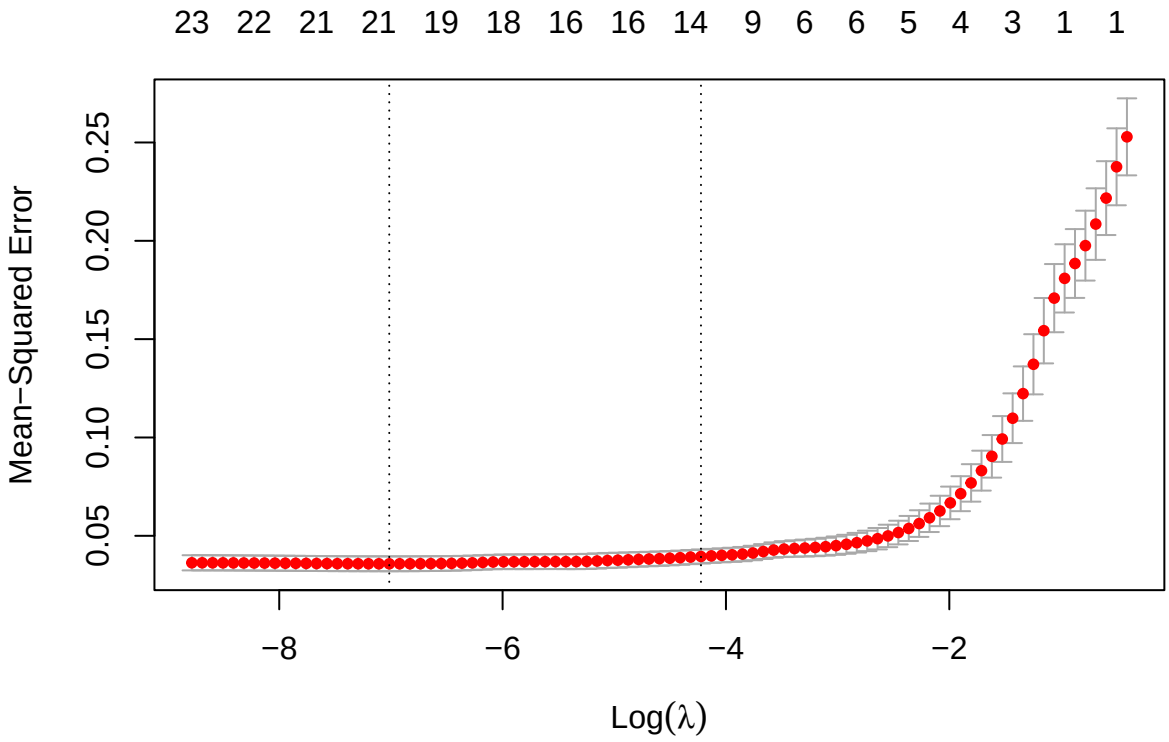
Lasso Regression:

Lasso Regression is used to fit a regression model when multicollinearity is present in the data. It is an acronym for least, absolute shrinkage and selection operator, and L1 norm. With the use L1 norm constraint, we force some of the regression coefficients to zero inducing sparsity by removing the less important predictors from the fitted model. From the output of Lasso Regression, We can see that the important variables are non-zero. Also, it has a r-squared of 90.3%, which means that 90.3% of the variation in price is explained by the predictors.

```
# Regularized Regression
# Lasso Regression
cvfit.lasso <- cv.glmnet(pred.mat, resp,alpha=1,
                         standardize=FALSE, type.measure = "mse", nfolds = 10)
coef(cvfit.lasso)
```

```
## 24 x 1 sparse Matrix of class "dgCMatrix"
##                              s1
## (Intercept)       9.201128460
## symboling         .
## fueltype          .
## aspiration        .
## doornumber        .
## carbody          -0.015166116
## drivewheel        0.044888690
## enginelocation    .
## enginetype        0.001563192
## cylindernumber    .
## fuelsystem        0.023938535
## wheelbase         .
## carlength         0.004506144
## carwidth          0.045658052
## carheight         0.003394150
## curbweight        0.158852526
## enginesize        0.075505578
## boreratio         .
## stroke           -0.004103181
## compressionratio  0.041886307
## horsepower        0.094140824
## peakrpm           0.007802121
## citympg          -0.068477126
## highwaympg        .
```

```
plot(cvfit.lasso)
```




```
best_lambda2 <- cvfit.lasso$lambda.min
best_lambda2
```

```
## [1] 0.0008984882
```

```
best.mod.lasso <- glmnet(pred.mat, resp,alpha=1, lambda=best_lambda2,
                         standardize=FALSE)
coef(best.mod.lasso)
```

```
## 24 x 1 sparse Matrix of class "dgCMatrix"
##                              s0
## (Intercept)       9.053810874
## symboling         .
## fueltype          .
## aspiration        0.000910898
## doornumber       -0.063657414
## carbody          -0.067388945
## drivewheel        0.065438532
## enginelocation    0.340295125
## enginetype        0.013556620
## cylindernumber   -0.001372759
## fuelsystem        0.021202782
## wheelbase         0.027546315
## carlength         0.006569545
## carwidth          0.047603174
```

```
## carheight          0.021107326
## curbweight         0.146291979
## enginesize         0.091046376
## boreratio         -0.017616655
## stroke            -0.024347651
## compressionratio   0.069321945
## horsepower         0.077935030
## peakrpm            0.040956400
## citympg           -0.151908081
## highwaympg         0.066365926
```

```
y_predicted.lasso <- predict(best.mod.lasso, s=best_lambda2, newx=pred.mat)
sst.lasso <- sum((resp-mean(resp))^2)
sse.lasso <- sum((y_predicted.lasso - resp)^2)
rsq.lasso <- 1-sse.lasso/sst.lasso
rsq.lasso
```

```
## [1] 0.9032841
```

Elastic net:

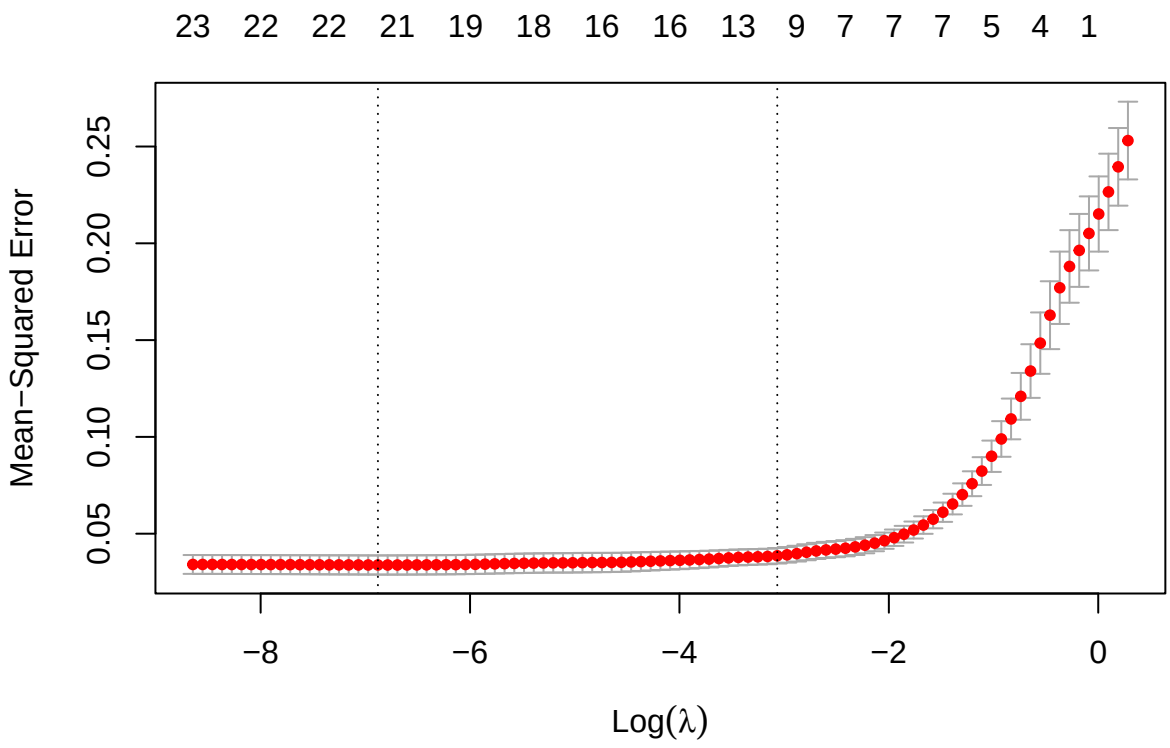
Elastic net is a regularized regression approach which is a combination of both lasso and ridge regression. The elastic net penalty is a convex combination of both penalizations (L2 and L1). Both ridge and lasso regression contain convex optimization problem. However, lasso is not always strictly convex like ridge regression. Meanwhile, elastic net is always strictly convex and combines the predictive properties of ridge regression with the sparsity properties of lasso. From the output of elastic net, We can see that the important variables are non-zero. Also, it has a r-squared of 90.2% (slightly lower than lasso), which means that 90.2% of the variation in price is explained by the predictors.

```
# Elastic net
cvfit.enet <- cv.glmnet(pred.mat, resp, alpha = 0.5,
                  standardize=FALSE, type.measure = "mse", nfolds = 10)
coef(cvfit.enet)
```

```
## 24 x 1 sparse Matrix of class "dgCMatrix"
##                              s1
## (Intercept)       9.1627867815
## symboling         .
## fueltype          .
## aspiration        .
## doornumber        .
## carbody          -0.0005569396
## drivewheel        0.0297342510
## enginelocation    .
## enginetype        .
## cylindernumber    .
## fuelsystem        0.0307032083
## wheelbase         .
## carlength         0.0089861863
## carwidth          0.0580474029
## carheight         .
## curbweight        0.1343871338
## enginesize        0.0787425188
```

```
## boreratio          .
## stroke             .
## compressionratio   0.0233150566
## horsepower         0.0895340532
## peakrpm            .
## citympg           -0.0564968405
## highwaympg         .
```

```
plot(cvfit.enet)
```




```
best_lambda3 <- cvfit.enet$lambda.min
best_lambda3
```

```
## [1] 0.001028296
```

```
best.mod.enet <- glmnet(pred.mat, resp,alpha=1, lambda=best_lambda3,
                        standardize=FALSE)
coef(best.mod.enet)
```

```
## 24 x 1 sparse Matrix of class "dgCMatrix"
##                               s0
## (Intercept)       9.0712938002
## symboling         .
## fueltype          .
```

```
## aspiration        .
## doornumber       -0.0634550298
## carbody          -0.0673956138
## drivewheel        0.0645696299
## enginelocation    0.3248538290
## enginetype        0.0134904069
## cylindernumber   -0.0009536492
## fuelsystem        0.0211861318
## wheelbase         0.0279394489
## carlength         0.0055136524
## carwidth          0.0458739370
## carheight         0.0212211811
## curbweight        0.1458866980
## enginesize        0.0909214428
## boreratio        -0.0159890245
## stroke           -0.0237723180
## compressionratio  0.0698593153
## horsepower        0.0799841162
## peakrpm           0.0408698193
## citympg          -0.1463860129
## highwaympg        0.0607995599
```

```
y_predicted.enet <- predict(best.mod.enet, s=best_lambda3, newx=pred.mat)
sst.enet <- sum((resp-mean(resp))^2)
sse.enet <- sum((y_predicted.enet - resp)^2)
rsq.enet <- 1-sse.enet/sst.enet
rsq.enet
```

```
## [1] 0.9030446
```

```
(all.coef <- cbind(coef(best.mod.ridge), coef(best.mod.lasso),
                   coef(best.mod.enet)))
```

```
## 24 x 3 sparse Matrix of class "dgCMatrix"
##                              s0           s0            s0
## (Intercept)       9.3024311025  9.053810874  9.0712938002
## symboling        -0.0004374836  .            .
## fueltype         -0.0190414627  .            .
## aspiration       -0.0010273227  0.000910898  .
## doornumber       -0.0324620726 -0.063657414 -0.0634550298
## carbody          -0.0575184890 -0.067388945 -0.0673956138
## drivewheel        0.0544210975  0.065438532  0.0645696299
## enginelocation    0.0352042113  0.340295125  0.3248538290
## enginetype        0.0142498635  0.013556620  0.0134904069
## cylindernumber    0.0051825232 -0.001372759 -0.0009536492
## fuelsystem        0.0264611714  0.021202782  0.0211861318
## wheelbase         0.0191056671  0.027546315  0.0279394489
## carlength         0.0320316982  0.006569545  0.0055136524
## carwidth          0.0493353282  0.047603174  0.0458739370
## carheight         0.0220263683  0.021107326  0.0212211811
## curbweight        0.0901735043  0.146291979  0.1458866980
## enginesize        0.0918222759  0.091046376  0.0909214428
## boreratio         0.0022335544 -0.017616655 -0.0159890245
```

```
## stroke           -0.0176247571 -0.024347651 -0.0237723180
## compressionratio  0.0589199550  0.069321945  0.0698593153
## horsepower        0.0943496037  0.077935030  0.0799841162
## peakrpm           0.0346474180  0.040956400  0.0408698193
## citympg          -0.0627807425 -0.151908081 -0.1463860129
## highwaympg       -0.0185413692  0.066365926  0.0607995599
```