# Final Project Report for IS 804 Advanced Quantitative Methods for IS Research: Statistical Learning

Zahid Hasan*

## Introduction

In this project, I apply statistical learning metods from ISLR book on the publicly available car price dataset kaggle website in the following link. My all works related to this project are also avaiable at my github link.

# Chapter 2

I start the experiment by adding the dataset in the working environment. The summary and name would provide the variables related to dataset. By attaching them it is convenient to call them using their name directly.

```
card <- read.csv("CarPrice_Assignment.csv")
attach(card) # attaching the variable names
coln <- colnames(card)
print(coln)
```

```
##  [1] "car_ID"           "symboling"        "CarName"          "fueltype"
##  [5] "aspiration"       "doornumber"       "carbody"          "drivewheel"
##  [9] "enginelocation"   "wheelbase"        "carlength"        "carwidth"
## [13] "carheight"        "curbweight"       "enginetype"       "cylindernumber"
## [17] "enginesize"       "fuelsystem"       "boreratio"        "stroke"
## [21] "compressionratio" "horsepower"       "peakrpm"          "citympg"
## [25] "highwaympg"       "price"
```

```
summary(card)
```

```
##      car_ID      symboling                  CarName      fueltype   aspiration
##  Min.   :  1   Min.   :-2.0000   peugeot 504   :  6   diesel: 20   std  :168
##  1st Qu.: 52   1st Qu.: 0.0000   toyota corolla:  6   gas   :185   turbo: 37
##  Median :103   Median : 1.0000   toyota corona :  6
##  Mean   :103   Mean   : 0.8341   subaru dl     :  4
##  3rd Qu.:154   3rd Qu.: 2.0000   honda civic   :  3
##  Max.   :205   Max.   : 3.0000   mazda 626     :  3
##                                  (Other)       :177
##  doornumber        carbody    drivewheel enginelocation   wheelbase
##  four:115   convertible: 6   4wd:  9    front:202      Min.   : 86.60
```

*Ph.D. Student, Information System, UMBC

```
##  two : 90   hardtop    : 8   fwd:120    rear :   3      1st Qu.: 94.50
##             hatchback  :70   rwd: 76                    Median : 97.00
##             sedan      :96                              Mean   : 98.76
##             wagon      :25                              3rd Qu.:102.40
##                                                         Max.   :120.90
##
##    carlength        carwidth       carheight       curbweight   enginetype
##  Min.   :141.1   Min.   :60.30   Min.   :47.80   Min.   :1488   dohc : 12
##  1st Qu.:166.3   1st Qu.:64.10   1st Qu.:52.00   1st Qu.:2145   dohcv:  1
##  Median :173.2   Median :65.50   Median :54.10   Median :2414   l    : 12
##  Mean   :174.0   Mean   :65.91   Mean   :53.72   Mean   :2556   ohc  :148
##  3rd Qu.:183.1   3rd Qu.:66.90   3rd Qu.:55.50   3rd Qu.:2935   ohcf : 15
##  Max.   :208.1   Max.   :72.30   Max.   :59.80   Max.   :4066   ohcv : 13
##                                                                 rotor:  4
##  cylindernumber   enginesize      fuelsystem   boreratio        stroke
##  eight :  5     Min.   : 61.0   mpfi   :94   Min.   :2.54   Min.   :2.070
##  five  : 11     1st Qu.: 97.0   2bbl   :66   1st Qu.:3.15   1st Qu.:3.110
##  four  :159     Median :120.0   idi    :20   Median :3.31   Median :3.290
##  six   : 24     Mean   :126.9   1bbl   :11   Mean   :3.33   Mean   :3.255
##  three :  1     3rd Qu.:141.0   spdi   : 9   3rd Qu.:3.58   3rd Qu.:3.410
##  twelve:  1     Max.   :326.0   4bbl   : 3   Max.   :3.94   Max.   :4.170
##  two   :  4                     (Other): 2
##  compressionratio   horsepower       peakrpm        citympg
##  Min.   : 7.00    Min.   : 48.0   Min.   :4150   Min.   :13.00
##  1st Qu.: 8.60    1st Qu.: 70.0   1st Qu.:4800   1st Qu.:19.00
##  Median : 9.00    Median : 95.0   Median :5200   Median :24.00
##  Mean   :10.14    Mean   :104.1   Mean   :5125   Mean   :25.22
##  3rd Qu.: 9.40    3rd Qu.:116.0   3rd Qu.:5500   3rd Qu.:30.00
##  Max.   :23.00    Max.   :288.0   Max.   :6600   Max.   :49.00
##
##    highwaympg        price
##  Min.   :16.00   Min.   : 5118
##  1st Qu.:25.00   1st Qu.: 7788
##  Median :30.00   Median :10295
##  Mean   :30.75   Mean   :13277
##  3rd Qu.:34.00   3rd Qu.:16503
##  Max.   :54.00   Max.   :45400
##
```

## Basic codes from the chapter 2

The chapter two introduces as some of the important introductory concept in R. In this chapter I have run and understand the basics from the book and showed it here. I will be using the codes to my dataset to implement the introductory codes and check the lengths and summary.

In this section, I will be also ploting the statictical parameters like mean, vaiances of the output variable and some input predictor variables.

```
library(ISLR)  # making all dataset available
```

```
## Warning: package 'ISLR' was built under R version 4.0.0
```

```r
x <- c(1,6,2)
y <- c(1,4,3)
x-y       #x+y, x*y
```

```
## [1]  0  2 -1
```

```r
length(card) # length(y)
```

```
## [1] 26
```

```r
ls() # check existing variables
```

```
## [1] "card" "coln" "x"    "y"
```

```r
rm(x,y) # remove variables
x <- matrix(data=c(1,2,3,4), nrow=2, ncol=2)
matrix(c(1,2,3,4),2,2, byrow=TRUE)
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    3    4
```

```r
x^2
```

```
##      [,1] [,2]
## [1,]    1    9
## [2,]    4   16
```

```r
sqrt(x)
```

```
##          [,1]     [,2]
## [1,] 1.000000 1.732051
## [2,] 1.414214 2.000000
```

```r
x <- rnorm(50)
y <- x + rnorm(50, mean=50, sd=1)
cor(x,y)
```

```
## [1] 0.7647336
```

```r
set.seed(1303)
rnorm(50)
```

```
##  [1] -1.1439763145  1.3421293656  2.1853904757  0.5363925179  0.0631929665
##  [6]  0.5022344825 -0.0004167247  0.5658198405 -0.5725226890 -1.1102250073
## [11] -0.0486871234 -0.6956562176  0.8289174803  0.2066528551 -0.2356745091
## [16] -0.5563104914 -0.3647543571  0.8623550343 -0.6307715354  0.3136021252
## [21] -0.9314953177  0.8238676185  0.5233707021  0.7069214120  0.4202043256
## [26] -0.2690521547 -1.5103172999 -0.6902124766 -0.1434719524 -1.0135274099
## [31]  1.5732737361  0.0127465055  0.8726470499  0.4220661905 -0.0188157917
## [36]  2.6157489689 -0.6931401748 -0.2663217810 -0.7206364412  1.3677342065
## [41]  0.2640073322  0.6321868074 -1.3306509858  0.0268888182  1.0406363208
## [46]  1.3120237985 -0.0300020767 -0.2500257125  0.0234144857  1.6598706557
```

```r
set.seed(3)
y <- rnorm(100)
mean(y)
```

```
## [1] 0.01103557
```

```r
var(y)
```

```
## [1] 0.7328675
```

```r
sqrt(var(y))
```

```
## [1] 0.8560768
```

```r
sd(y)
```

```
## [1] 0.8560768
```

Now implementing the commands to get the dataset description on output variables.

```r
mean(price)
```

```
## [1] 13276.71
```

```r
var(price)
```

```
## [1] 63821762
```

```r
sqrt(var(price))
```

```
## [1] 7988.852
```

```r
sd(price)
```

```
## [1] 7988.852
```

```r
mode(price)
```

```
## [1] "numeric"
```

We can also implementing the commands to get the statistical parameters for the input variables too.

```r
mean(enginesize)
```

```
## [1] 126.9073
```

```r
var(enginesize)
```

```
## [1] 1734.114
```

```r
sqrt(var(enginesize))
```

```
## [1] 41.64269
```

```r
sd(enginesize)
```

```
## [1] 41.64269
```

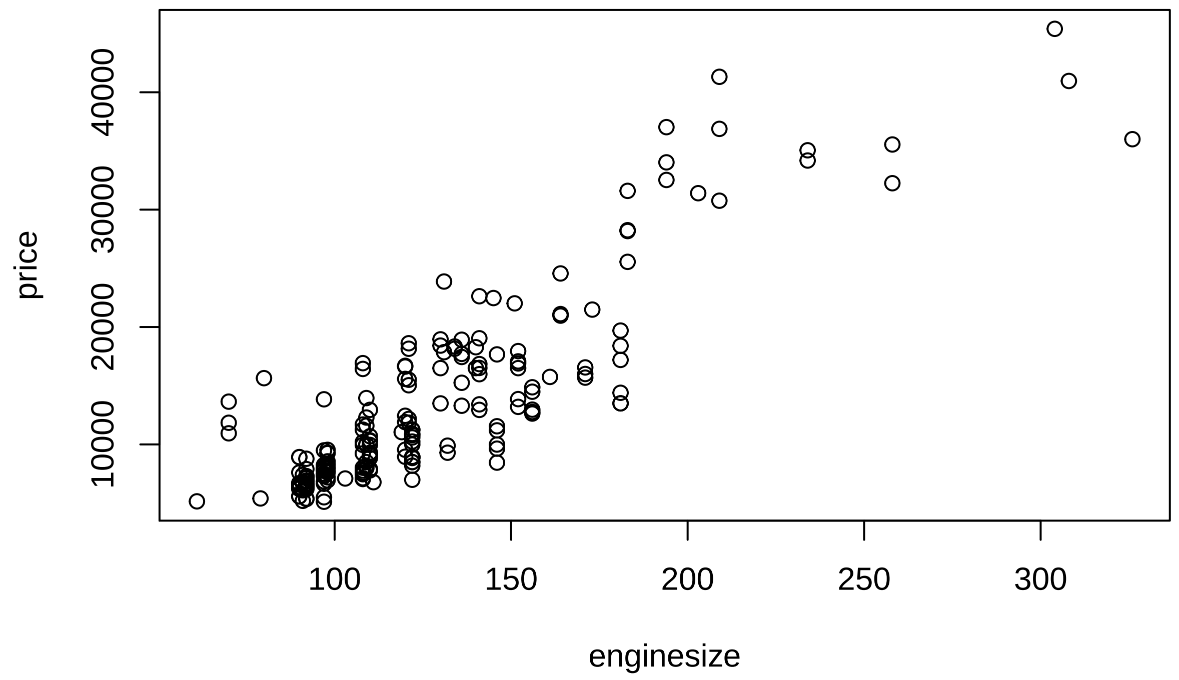
```r
mode(enginesize)
```

```
## [1] "numeric"
```

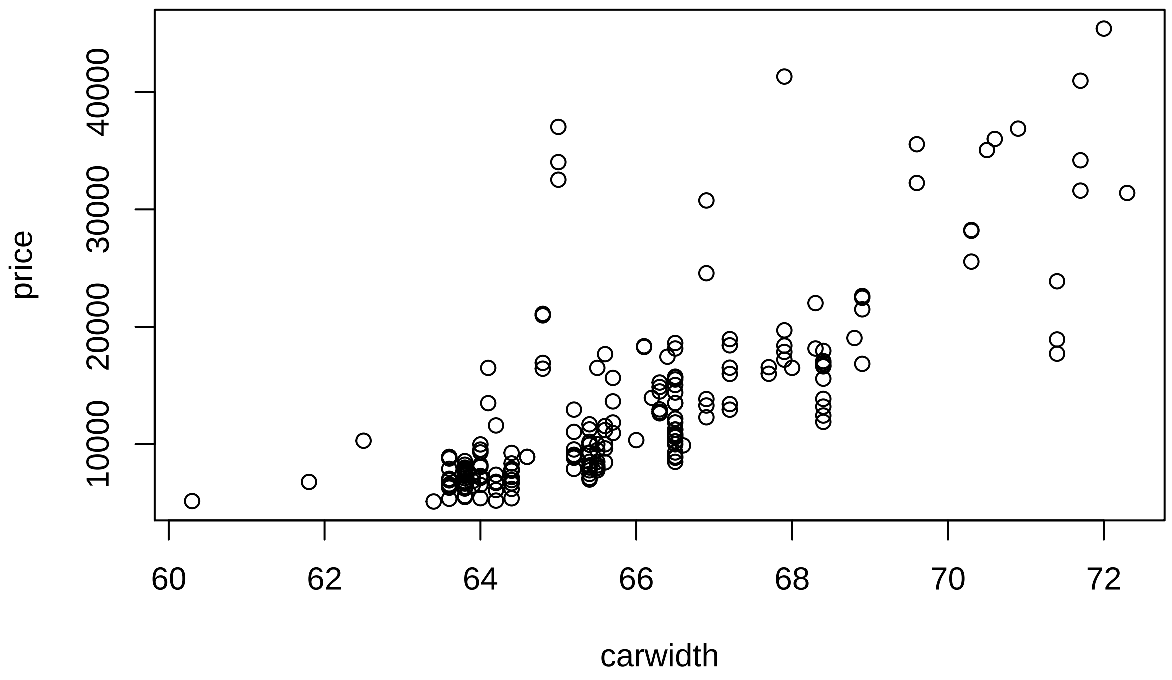
## Plot function from chapter 2

In this section using the plot option shown in chapter 2, I plot some the variables together to find the relationship between them and the output predictors. I used both categorical and continuous predictor to demonstrate the relationship between them.

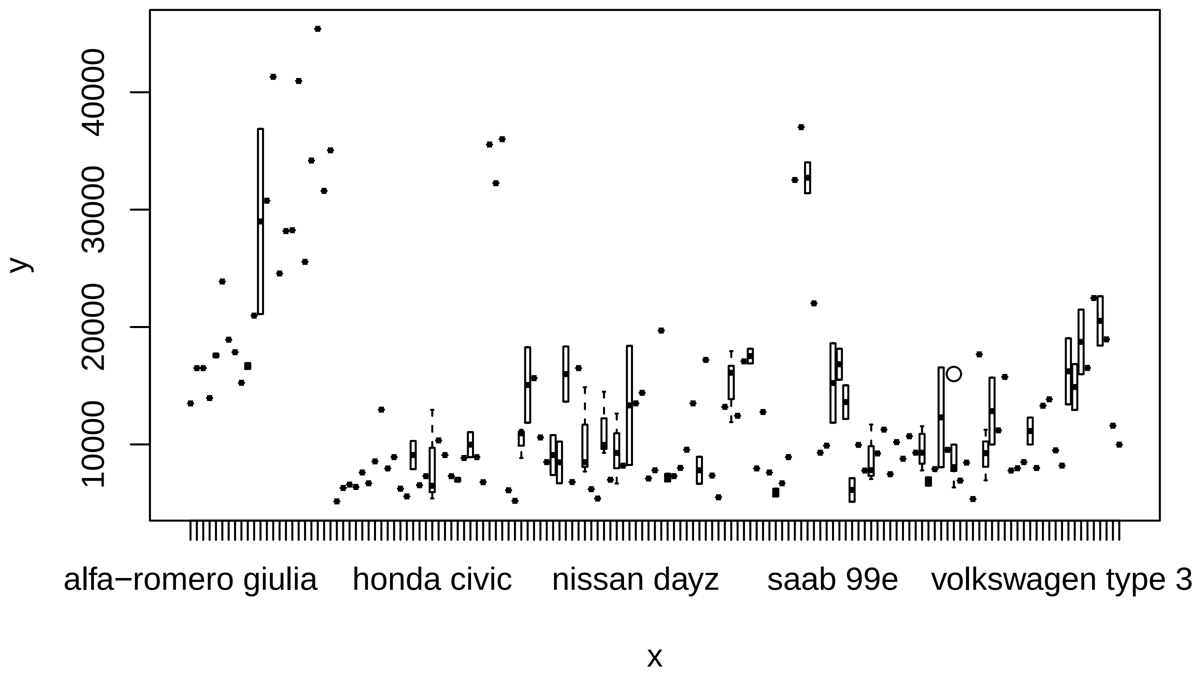
```r
plot(enginesize, price)
```



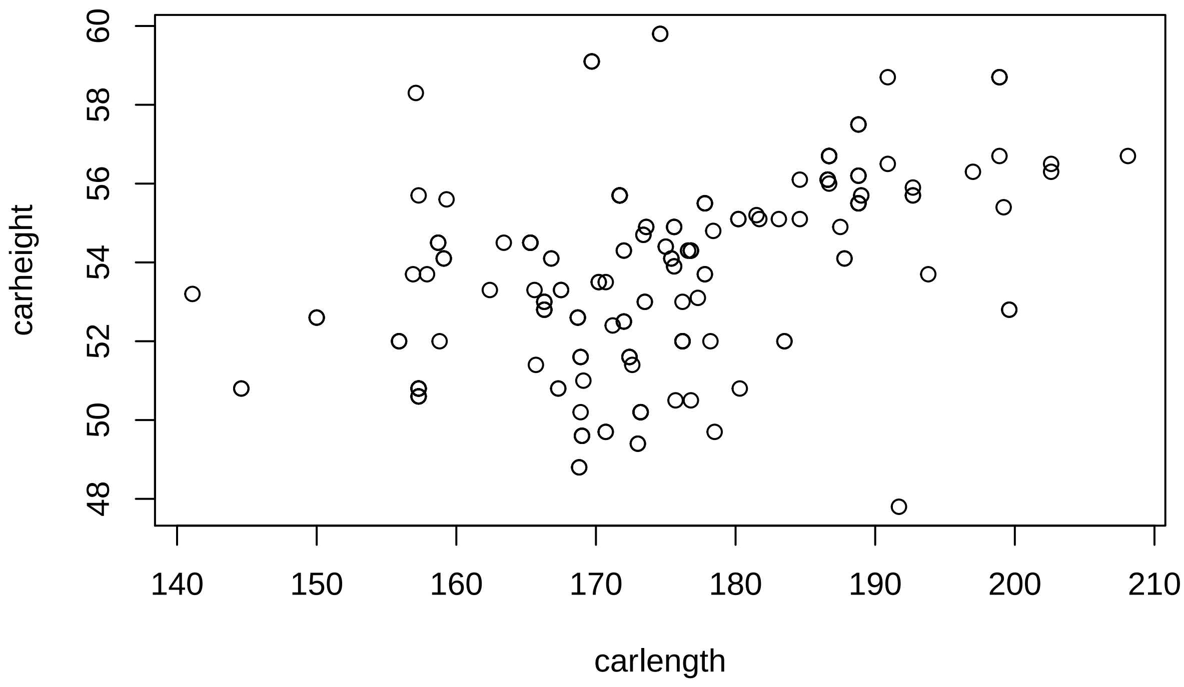
```r
plot(carwidth, price)
```



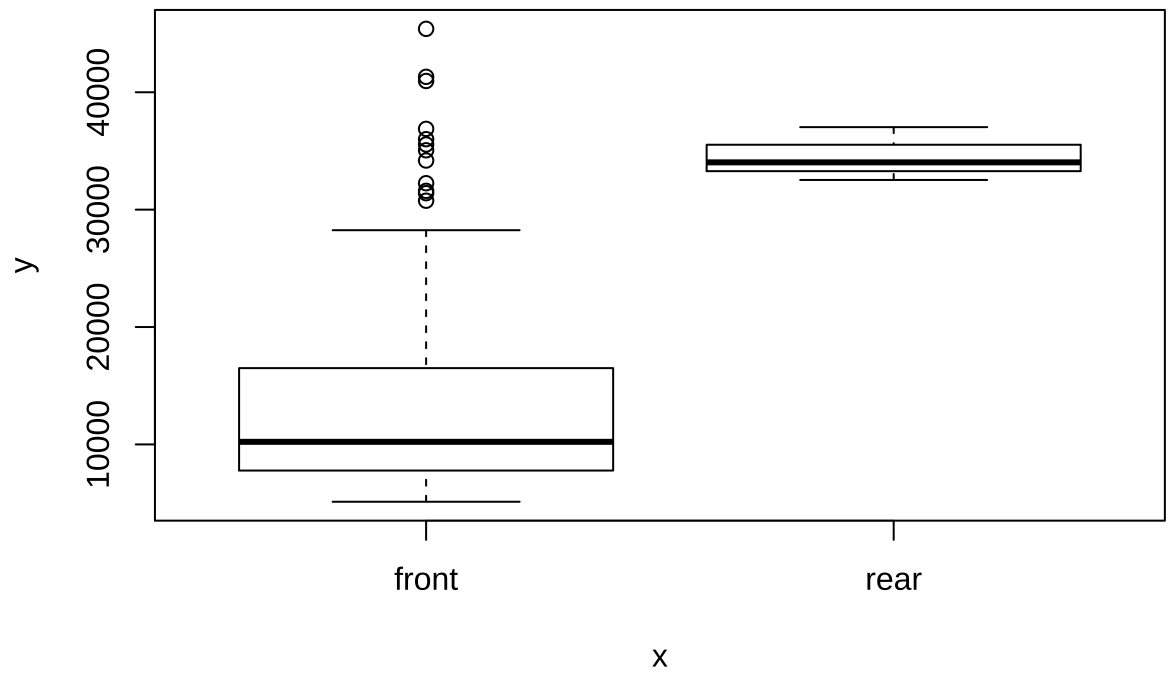
```r
plot(CarName, price)
```



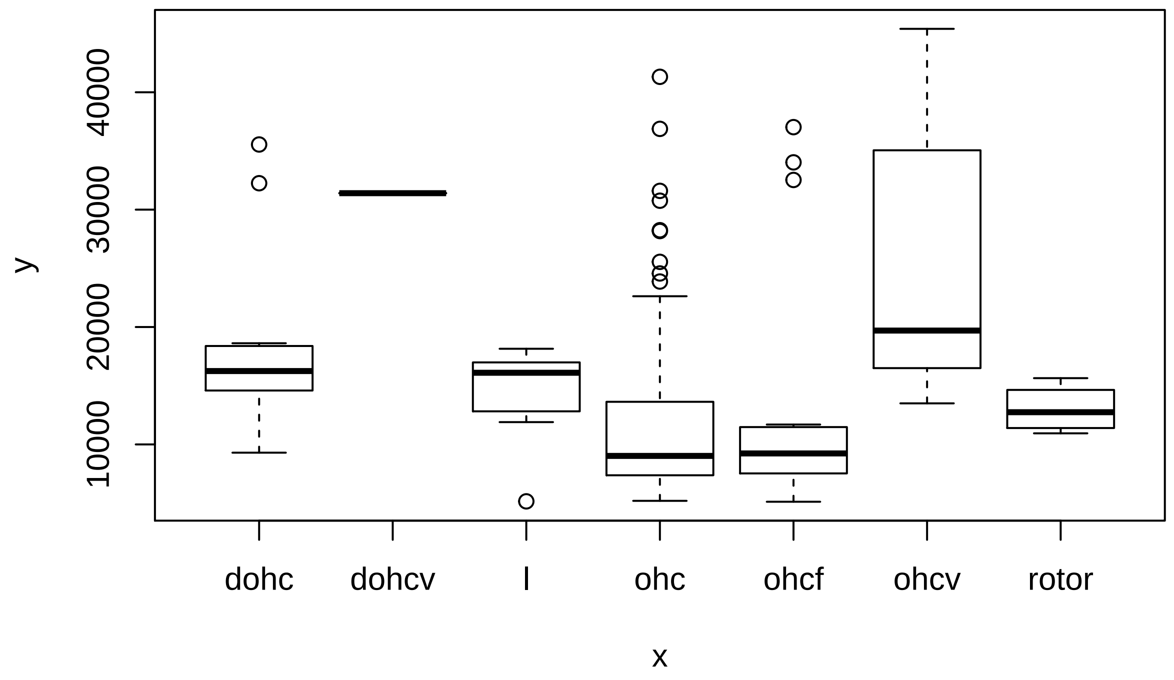
```r
plot(carlength, carheight)
```



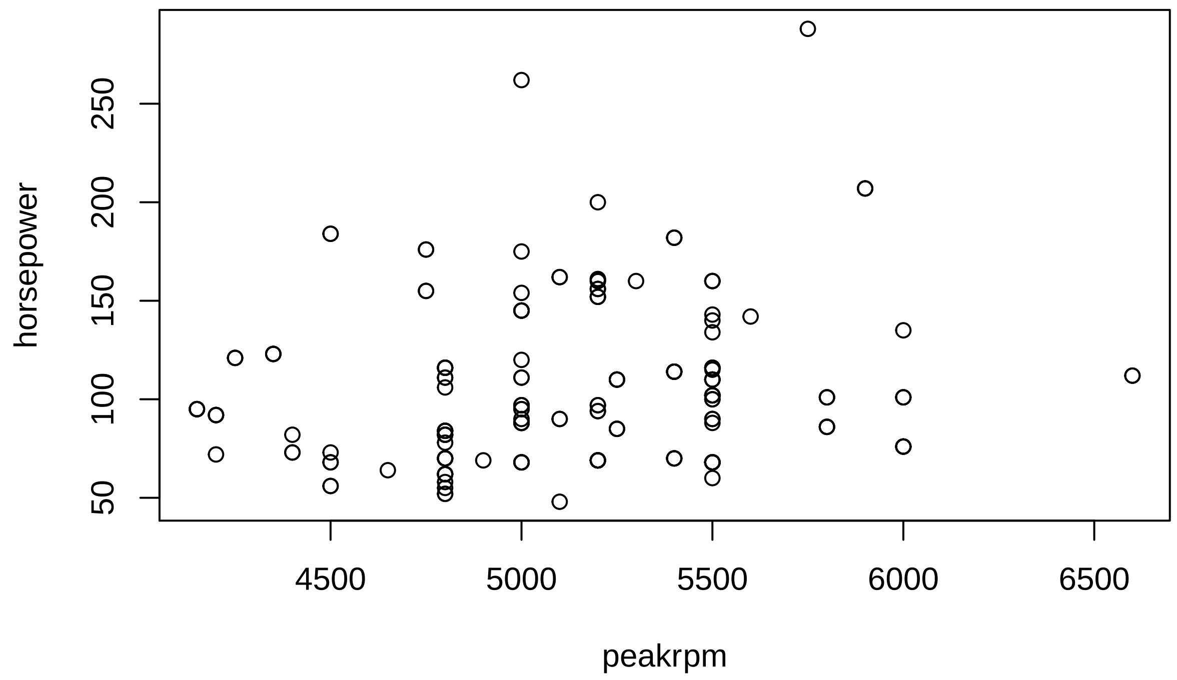
```r
plot(enginelocation, price)
```



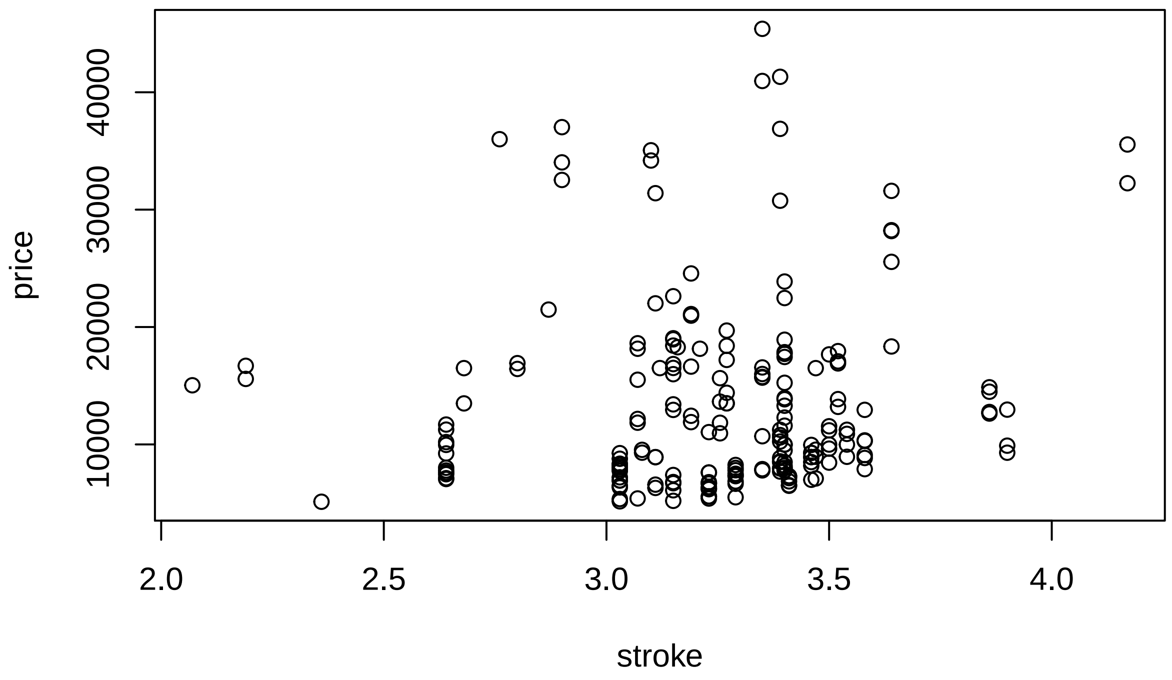
```r
plot(enginetype, price)
```



```r
plot(peakrpm, horsepower)
```



```r
plot(stroke, price)
```



In those we can observe that some variables like enginesize, carwidth are closely related to predictor variables (Price). Some relations are not obvious form the plots like stroke and price.

# Chapter 3

In this section, I will be implement codes of linear regression in my dataset. Firstly, I will apply linear regression using all the predictor features. Then I will narrow the features for better understand and explain the code and methods.

```
attach(card)
```

```
## The following objects are masked from card (pos = 4):
##
##     aspiration, boreratio, car_ID, carbody, carheight, carlength,
##     CarName, carwidth, citympg, compressionratio, curbweight,
##     cylindernumber, doornumber, drivewheel, enginelocation, enginesize,
##     enginetype, fuelsystem, fueltype, highwaympg, horsepower, peakrpm,
##     price, stroke, symboling, wheelbase
```

```
attach(card)
```

```
## The following objects are masked from card (pos = 3):
##
##     aspiration, boreratio, car_ID, carbody, carheight, carlength,
##     CarName, carwidth, citympg, compressionratio, curbweight,
##     cylindernumber, doornumber, drivewheel, enginelocation, enginesize,
##     enginetype, fuelsystem, fueltype, highwaympg, horsepower, peakrpm,
##     price, stroke, symboling, wheelbase

## The following objects are masked from card (pos = 5):
##
##     aspiration, boreratio, car_ID, carbody, carheight, carlength,
##     CarName, carwidth, citympg, compressionratio, curbweight,
##     cylindernumber, doornumber, drivewheel, enginelocation, enginesize,
##     enginetype, fuelsystem, fueltype, highwaympg, horsepower, peakrpm,
##     price, stroke, symboling, wheelbase
```

```
lm.fit = lm(price~., data =  card)
#lm.fit = lm(price~fuelsystem+peakrpm+citympg+CarName+enginesize+enginetype+carwidth+curbweight+carleng
summary(lm.fit)
```

```
##
## Call:
## lm(formula = price ~ ., data = card)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -1576      0      0      0   1421
##
## Coefficients: (12 not defined because of singularities)
##                                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)                              1464.060  24067.028   0.061 0.951958
## car_ID                                    153.076     64.684   2.367 0.025687
## symboling                                 725.668    323.690   2.242 0.033723
## CarNamealfa-romero Quadrifoglio          5339.290   4141.961   1.289 0.208724
## CarNamealfa-romero stelvio               2851.924   1556.345   1.832 0.078362
## CarNameaudi 100 ls                       3980.044   3341.799   1.191 0.244418
## CarNameaudi 100ls                        5440.965   3939.323   1.381 0.178976
## CarNameaudi 4000                         9453.414   4893.730   1.932 0.064359
## CarNameaudi 5000                         7315.676   4148.741   1.763 0.089593
## CarNameaudi 5000s (diesel)               5048.734   4806.719   1.050 0.303223
## CarNameaudi fox                          8004.584   3850.889   2.079 0.047659
## CarNamebmw 320i                          3665.224   2932.400   1.250 0.222468
## CarNamebmw x1                           16398.848   4064.364   4.035 0.000427
## CarNamebmw x3                           15669.006   3870.528   4.048 0.000412
## CarNamebmw x4                           16597.885   4245.075   3.910 0.000591
## CarNamebmw x5                           26105.388   4116.060   6.342 1.02e-06
## CarNamebmw z4                           15092.926   4104.480   3.677 0.001079
## CarNamebuick century                     3394.510   6114.139   0.555 0.583510
## CarNamebuick century luxus (sw)          2830.130   6144.487   0.461 0.648917
## CarNamebuick century special              315.888   5317.826   0.059 0.953086
## CarNamebuick electra 225 custom          -238.538   5517.392  -0.043 0.965845
## CarNamebuick opel isuzu deluxe          -2746.797   4648.634  -0.591 0.559701
## CarNamebuick regal sport coupe (turbo)   8247.321   5282.930   1.561 0.130585
## CarNamebuick skyhawk                     2498.152   5625.634   0.444 0.660669
## CarNamebuick skylark                    -2256.734   5472.035  -0.412 0.683418
## CarNamechevrolet impala                   745.695   4422.267   0.169 0.867398
## CarNamechevrolet monte carlo            -5791.923   3932.695  -1.473 0.152819
## CarNamechevrolet vega 2300              -6942.823   3806.475  -1.824 0.079675
## CarNamedodge challenger se              -5644.893   3638.855  -1.551 0.132922
## CarNamedodge colt (sw)                  -8616.639   3757.617  -2.293 0.030177
## CarNamedodge colt hardtop               -9380.564   3719.607  -2.522 0.018133
## CarNamedodge coronet custom             -8832.211   3661.983  -2.412 0.023229
## CarNamedodge coronet custom (sw)        -5385.044   4548.548  -1.184 0.247162
## CarNamedodge d200                       -7061.034   3456.794  -2.043 0.051348
## CarNamedodge dart custom                -3481.303   3974.749  -0.876 0.389129
## CarNamedodge monaco (sw)                -8467.722   3715.083  -2.279 0.031099
## CarNamedodge rampage                    -7760.845   3894.157  -1.993 0.056861
## CarNamehonda accord                     -3275.628   5032.000  -0.651 0.520785
## CarNamehonda accord cvcc                -6814.723   4655.065  -1.464 0.155198
## CarNamehonda accord lx                  -5594.750   4749.667  -1.178 0.249496
## CarNamehonda civic                      -4621.260   4100.458  -1.127 0.270032
## CarNamehonda civic (auto)               -8117.608   4535.662  -1.790 0.085153
## CarNamehonda civic 1300                 -3924.742   4903.031  -0.800 0.430691
## CarNamehonda civic 1500 gl              -3391.219   5103.479  -0.664 0.512227
## CarNamehonda civic cvcc                 -5930.849   4596.581  -1.290 0.208313
## CarNamehonda prelude                    -4625.080   4938.351  -0.937 0.357602
## CarNameisuzu D-Max                      -7981.442   4919.273  -1.622 0.116765
## CarNameisuzu D-Max V-Cross              -9077.711   4862.983  -1.867 0.073261
## CarNameisuzu MU-X                       -7911.400   3885.498  -2.036 0.052043
## CarNamejaguar xf                          160.068   3672.109   0.044 0.965564
## CarNamejaguar xj                        -2986.856   3661.432  -0.816 0.422047
## CarNamejaguar xk                         4389.084   6753.443   0.650 0.521456
## CarNamemaxda glc deluxe                 -7881.451   4806.507  -1.640 0.113103
## CarNamemaxda rx3                        -9080.132   4847.291  -1.873 0.072319
## CarNamemazda 626                        -8611.028   4624.289  -1.862 0.073924
## CarNamemazda glc                        -6522.541   4440.118  -1.469 0.153829
## CarNamemazda glc 4                      -6814.221   5650.857  -1.206 0.238726
## CarNamemazda glc custom                 -8221.173   4826.224  -1.703 0.100412
## CarNamemazda glc custom l              -11155.639   4696.154  -2.375 0.025183
## CarNamemazda glc deluxe                 -8189.991   4851.734  -1.688 0.103362
## CarNamemazda rx-4                       -9168.168   4685.529  -1.957 0.061203
## CarNamemazda rx-7 gs                    -5694.495   4599.115  -1.238 0.226716
## CarNamemazda rx2 coupe                  -7388.763   4860.722  -1.520 0.140556
## CarNamemercury cougar                   -8178.650   6009.231  -1.361 0.185190
## CarNamemitsubishi g4                   -15717.851   6169.643  -2.548 0.017103
## CarNamemitsubishi lancer               -16251.139   6357.561  -2.556 0.016772
## CarNamemitsubishi mirage               -18099.163   6498.938  -2.785 0.009854
## CarNamemitsubishi mirage g4            -16070.951   6325.950  -2.540 0.017384
## CarNamemitsubishi montero              -18648.094   6152.657  -3.031 0.005459
## CarNamemitsubishi outlander            -17041.307   6287.017  -2.711 0.011737
## CarNamemitsubishi pajero               -18035.063   6236.337  -2.892 0.007638
## CarNamenissan clipper                  -14461.607   7557.316  -1.914 0.066743
## CarNamenissan dayz                     -15537.918   7605.331  -2.043 0.051309
## CarNamenissan fuga                     -13801.149   8073.239  -1.709 0.099271
## CarNamenissan gt-r                     -15601.158   6975.502  -2.237 0.034110
## CarNamenissan juke                     -13042.895   7277.852  -1.792 0.084757
## CarNamenissan kicks                    -14465.254   8196.035  -1.765 0.089325
## CarNamenissan latio                    -14503.856   7078.671  -2.049 0.050685
## CarNamenissan leaf                     -13051.687   7260.070  -1.798 0.083843
## CarNamenissan note                     -12338.572   7463.771  -1.653 0.110331
## CarNamenissan nv200                   -16025.649   7182.645  -2.231 0.034509
## CarNamenissan otti                    -15060.861   7682.618  -1.960 0.060749
## CarNamenissan rogue                   -14305.794   7032.968  -2.034 0.052261
## CarNamenissan teana                   -15631.312   7879.622  -1.984 0.057934
## CarNamenissan titan                   -12235.253   7220.538  -1.695 0.102116
## CarNameNissan versa                   -13413.773   6942.545  -1.932 0.064311
## CarNamepeugeot 304                    -26389.554   6262.505  -4.214 0.000267
## CarNamepeugeot 504                    -23978.749   6402.633  -3.745 0.000906
## CarNamepeugeot 504 (sw)               -23709.629   6380.998  -3.716 0.000977
## CarNamepeugeot 505s turbo diesel      -22733.899   6714.965  -3.386 0.002266
## CarNamepeugeot 604sl                  -24539.992   6577.093  -3.731 0.000939
## CarNameplymouth cricket               -21289.464   8681.516  -2.452 0.021222
## CarNameplymouth duster                -23294.031   8326.902  -2.797 0.009567
## CarNameplymouth fury gran sedan       -23615.334   8797.386  -2.684 0.012478
## CarNameplymouth fury iii              -23123.441   8818.842  -2.622 0.014418
## CarNameplymouth satellite custom (sw)  -21743.841   8711.085  -2.496 0.019226
## CarNameplymouth valiant               -17988.655   8785.531  -2.048 0.050834
## CarNameporcshce panamera               -4303.206  10526.456  -0.409 0.686035
## CarNameporsche boxter                    393.465  11113.478   0.035 0.972028
## CarNameporsche cayenne                 -2956.282  10465.289  -0.282 0.779809
## CarNameporsche macan                  -12031.956   8866.189  -1.357 0.186427
## CarNamerenault 12tl                   -17786.564   8646.623  -2.057 0.049843
## CarNamerenault 5 gtl                  -20579.681   8556.385  -2.405 0.023577
## CarNamesaab 99e                       -16370.556   9256.390  -1.769 0.088699
## CarNamesaab 99gle                     -15269.026   9336.815  -1.635 0.114025
## CarNamesaab 99le                      -18228.042   9231.742  -1.974 0.059036
## CarNamesubaru                         -27076.490   9392.453  -2.883 0.007808
## CarNamesubaru baja                    -25469.712   9011.549  -2.826 0.008933
## CarNamesubaru brz                     -24260.657   9253.575  -2.622 0.014427
## CarNamesubaru dl                      -24848.991   9321.737  -2.666 0.013031
## CarNamesubaru r1                      -26140.178   9207.082  -2.839 0.008665
## CarNamesubaru r2                      -25641.614   8827.539  -2.905 0.007408
## CarNamesubaru trezia                  -25979.569   9701.227  -2.678 0.012665
## CarNamesubaru tribeca                 -24222.721   9501.112  -2.549 0.017031
## CarNametoyota carina                  -28409.459  11515.664  -2.467 0.020530
## CarNametoyota celica gt               -25805.686  11595.954  -2.225 0.034939
## CarNametoyota celica gt liftback      -31145.270  10832.939  -2.875 0.007953
## CarNametoyota corolla                 -24934.252  10863.751  -2.295 0.030042
## CarNametoyota corolla 1200            -25121.115  10594.321  -2.371 0.025424
## CarNametoyota corolla 1600 (sw)       -20241.570  10685.135  -1.894 0.069351
## CarNametoyota corolla liftback        -25071.958  10936.948  -2.292 0.030223
## CarNametoyota corolla tercel          -30453.878  10869.368  -2.802 0.009468
## CarNametoyota corona                  -25299.866  10843.154  -2.333 0.027640
## CarNametoyota corona hardtop          -21796.585  11027.174  -1.977 0.058781
## CarNametoyota corona liftback         -26119.358  10798.687  -2.419 0.022875
## CarNametoyota corona mark ii          -25795.352  10651.811  -2.422 0.022725
## CarNametoyota cressida                -20941.386  11637.854  -1.799 0.083570
## CarNametoyota mark ii                 -24785.364  10883.275  -2.277 0.031228
## CarNametoyota starlet                 -25091.653  10746.339  -2.335 0.027541
## CarNametoyota tercel                  -25336.364  11071.265  -2.288 0.030483
## CarNametoyouta tercel                 -22470.447  11167.292  -2.012 0.054670
## CarNamevokswagen rabbit               -28146.900  12718.048  -2.213 0.035870
## CarNamevolkswagen 1131 deluxe sedan   -28350.471  12222.975  -2.319 0.028491
## CarNamevolkswagen 411 (sw)            -30294.826  12296.250  -2.464 0.020682
## CarNamevolkswagen dasher            -27412.373  12723.706  -2.154 0.040649
## CarNamevolkswagen model 111         -29554.798  12696.486  -2.328 0.027974
## CarNamevolkswagen rabbit            -23080.543  12503.588  -1.846 0.076321
## CarNamevolkswagen rabbit custom     -25478.948  13041.257  -1.954 0.061573
## CarNamevolkswagen super beetle      -28172.032  13054.263  -2.158 0.040337
## CarNamevolkswagen type 3            -29758.368  12203.867  -2.438 0.021890
## CarNamevolvo 144ea                  -27427.953  12668.497  -2.165 0.039744
## CarNamevolvo 145e (sw)              -29127.275  12239.829  -2.380 0.024949
## CarNamevolvo 244dl                  -26297.009  12404.001  -2.120 0.043706
## CarNamevolvo 245                    -25251.993  12782.992  -1.975 0.058923
## CarNamevolvo 246                    -21108.975  13339.878  -1.582 0.125649
## CarNamevolvo 264gl                  -25037.924  12552.574  -1.995 0.056664
## CarNamevolvo diesel                 -23910.405  13115.100  -1.823 0.079804
## CarNamevw dasher                    -27150.785  13299.630  -2.041 0.051474
## CarNamevw rabbit                    -30000.371  12770.604  -2.349 0.026690
## fueltypegas                         -27422.306   9966.212  -2.752 0.010662
## aspirationturbo                        236.211    986.002   0.240 0.812547
## doornumbertwo                        -1289.204    726.157  -1.775 0.087544
## carbodyhardtop                         980.106   1811.658   0.541 0.593113
## carbodyhatchback                      1074.707   1218.575   0.882 0.385892
## carbodysedan                          2058.832   1031.011   1.997 0.056403
## carbodywagon                                NA         NA      NA       NA
## drivewheelfwd                          220.466   1260.739   0.175 0.862536
## drivewheelrwd                          697.180   1435.842   0.486 0.631350
## enginelocationrear                    9158.974   3987.839   2.297 0.029940
## wheelbase                              289.818    128.512   2.255 0.032766
## carlength                             -223.963     75.521  -2.966 0.006398
## carwidth                               707.880    306.717   2.308 0.029218
## carheight                             -461.834    272.256  -1.696 0.101767
## curbweight                              10.847      3.395   3.195 0.003650
## enginetypedohcv                             NA         NA      NA       NA
## enginetypel                                 NA         NA      NA       NA
## enginetypeohc                        -3823.290   1866.254  -2.049 0.050717
## enginetypeohcf                              NA         NA      NA       NA
## enginetypeohcv                         804.447   2282.580   0.352 0.727358
## enginetyperotor                       6623.579   3942.157   1.680 0.104898
## cylindernumberfive                          NA         NA      NA       NA
## cylindernumberfour                    7150.023   2131.208   3.355 0.002447
## cylindernumbersix                           NA         NA      NA       NA
## cylindernumberthree                         NA         NA      NA       NA
## cylindernumbertwelve                        NA         NA      NA       NA
## cylindernumbertwo                           NA         NA      NA       NA
## enginesize                              55.532     50.444   1.101 0.281043
## fuelsystem2bbl                        3954.307   2726.010   1.451 0.158854
## fuelsystem4bbl                              NA         NA      NA       NA
## fuelsystemidi                               NA         NA      NA       NA
## fuelsystemmfi                               NA         NA      NA       NA
## fuelsystemmpfi                        2993.555   2482.520   1.206 0.238733
## fuelsystemspdi                        2526.711   3442.636   0.734 0.469549
## fuelsystemspfi                        2068.375   4073.432   0.508 0.615894
## boreratio                            -3484.606   1792.683  -1.944 0.062818
## stroke                               -1314.657   1072.834  -1.225 0.231407
## compressionratio                     -1910.210    772.424  -2.473 0.020255
## horsepower                             -42.876     34.010  -1.261 0.218616
## peakrpm                                   3.286      1.008   3.261 0.003099
## citympg                                 309.298    223.210   1.386 0.177618
## highwaympg                             -124.218    164.066  -0.757 0.455783
##
## (Intercept)
## car_ID                                *
## symboling                             *
## CarNamealfa-romero Quadrifoglio
## CarNamealfa-romero stelvio            .
## CarNameaudi 100 ls
## CarNameaudi 100ls
## CarNameaudi 4000                      .
## CarNameaudi 5000                      .
## CarNameaudi 5000s (diesel)
## CarNameaudi fox                       *
## CarNamebmw 320i
## CarNamebmw x1                         ***
## CarNamebmw x3                         ***
## CarNamebmw x4                         ***
## CarNamebmw x5                         ***
## CarNamebmw z4                         **
## CarNamebuick century
## CarNamebuick century luxus (sw)
## CarNamebuick century special
## CarNamebuick electra 225 custom
## CarNamebuick opel isuzu deluxe
## CarNamebuick regal sport coupe (turbo)
## CarNamebuick skyhawk
## CarNamebuick skylark
## CarNamechevrolet impala
## CarNamechevrolet monte carlo
## CarNamechevrolet vega 2300            .
## CarNamedodge challenger se
## CarNamedodge colt (sw)                *
## CarNamedodge colt hardtop             *
## CarNamedodge coronet custom           *
## CarNamedodge coronet custom (sw)
## CarNamedodge d200                     .
## CarNamedodge dart custom
## CarNamedodge monaco (sw)              *
## CarNamedodge rampage                  .
## CarNamehonda accord
## CarNamehonda accord cvcc
## CarNamehonda accord lx
## CarNamehonda civic
## CarNamehonda civic (auto)             .
## CarNamehonda civic 1300
## CarNamehonda civic 1500 gl
## CarNamehonda civic cvcc
## CarNamehonda prelude
## CarNameisuzu D-Max
## CarNameisuzu D-Max V-Cross            .
## CarNameisuzu MU-X                     .
## CarNamejaguar xf
## CarNamejaguar xj
## CarNamejaguar xk
## CarNamemaxda glc deluxe
## CarNamemaxda rx3                        .
## CarNamemazda 626                        .
## CarNamemazda glc
## CarNamemazda glc 4
## CarNamemazda glc custom
## CarNamemazda glc custom l               *
## CarNamemazda glc deluxe
## CarNamemazda rx-4                       .
## CarNamemazda rx-7 gs
## CarNamemazda rx2 coupe
## CarNamemercury cougar
## CarNamemitsubishi g4                    *
## CarNamemitsubishi lancer                *
## CarNamemitsubishi mirage                **
## CarNamemitsubishi mirage g4             *
## CarNamemitsubishi montero               **
## CarNamemitsubishi outlander             *
## CarNamemitsubishi pajero                **
## CarNamenissan clipper                   .
## CarNamenissan dayz                      .
## CarNamenissan fuga                      .
## CarNamenissan gt-r                      *
## CarNamenissan juke                      .
## CarNamenissan kicks                     .
## CarNamenissan latio                     .
## CarNamenissan leaf                      .
## CarNamenissan note
## CarNamenissan nv200                     *
## CarNamenissan otti                      .
## CarNamenissan rogue                     .
## CarNamenissan teana                     .
## CarNamenissan titan
## CarNameNissan versa                     .
## CarNamepeugeot 304                      ***
## CarNamepeugeot 504                      ***
## CarNamepeugeot 504 (sw)                 ***
## CarNamepeugeot 505s turbo diesel        **
## CarNamepeugeot 604sl                    ***
## CarNameplymouth cricket                 *
## CarNameplymouth duster                  **
## CarNameplymouth fury gran sedan         *
## CarNameplymouth fury iii                *
## CarNameplymouth satellite custom (sw)   *
## CarNameplymouth valiant                 .
## CarNameporcshce panamera
## CarNameporsche boxter
## CarNameporsche cayenne
## CarNameporsche macan
## CarNamerenault 12tl                     *
## CarNamerenault 5 gtl                    *
## CarNamesaab 99e                         .
## CarNamesaab 99gle
## CarNamesaab 99le                       .
## CarNamesubaru                          **
## CarNamesubaru baja                     **
## CarNamesubaru brz                      *
## CarNamesubaru dl                       *
## CarNamesubaru r1                       **
## CarNamesubaru r2                       **
## CarNamesubaru trezia                   *
## CarNamesubaru tribeca                  *
## CarNametoyota carina                   *
## CarNametoyota celica gt                *
## CarNametoyota celica gt liftback       **
## CarNametoyota corolla                  *
## CarNametoyota corolla 1200             *
## CarNametoyota corolla 1600 (sw)        .
## CarNametoyota corolla liftback         *
## CarNametoyota corolla tercel           **
## CarNametoyota corona                   *
## CarNametoyota corona hardtop           .
## CarNametoyota corona liftback          *
## CarNametoyota corona mark ii           *
## CarNametoyota cressida                 .
## CarNametoyota mark ii                  *
## CarNametoyota starlet                  *
## CarNametoyota tercel                   *
## CarNametoyouta tercel                  .
## CarNamevokswagen rabbit                *
## CarNamevolkswagen 1131 deluxe sedan    *
## CarNamevolkswagen 411 (sw)             *
## CarNamevolkswagen dasher               *
## CarNamevolkswagen model 111            *
## CarNamevolkswagen rabbit               .
## CarNamevolkswagen rabbit custom        .
## CarNamevolkswagen super beetle         *
## CarNamevolkswagen type 3               *
## CarNamevolvo 144ea                     *
## CarNamevolvo 145e (sw)                 *
## CarNamevolvo 244dl                     *
## CarNamevolvo 245                       .
## CarNamevolvo 246
## CarNamevolvo 264gl                     .
## CarNamevolvo diesel                    .
## CarNamevw dasher                       .
## CarNamevw rabbit                       *
## fueltypegas                            *
## aspirationturbo
## doornumbertwo                          .
## carbodyhardtop
## carbodyhatchback
## carbodysedan                           .
## carbodywagon
## drivewheelfwd
## drivewheelrwd
## enginelocationrear                       *
## wheelbase                                *
## carlength                                **
## carwidth                                 *
## carheight
## curbweight                               **
## enginetypedohcv
## enginetypel
## enginetypeohc                            .
## enginetypeohcf
## enginetypeohcv
## enginetyperotor
## cylindernumberfive
## cylindernumberfour                       **
## cylindernumbersix
## cylindernumberthree
## cylindernumbertwelve
## cylindernumbertwo
## enginesize
## fuelsystem2bbl
## fuelsystem4bbl
## fuelsystemidi
## fuelsystemmfi
## fuelsystemmpfi
## fuelsystemspdi
## fuelsystemspfi
## boreratio                                .
## stroke
## compressionratio                         *
## horsepower
## peakrpm                                  **
## citympg
## highwaympg
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1100 on 26 degrees of freedom
## Multiple R-squared:  0.9976, Adjusted R-squared:  0.9811
## F-statistic: 60.35 on 178 and 26 DF,  p-value: < 2.2e-16
```

By focusing on carname feaure only

```
lm.fit1 = lm(price~CarName)
summary(lm(price~horsepower+CarName))
```

```
##
## Call:
## lm(formula = price ~ horsepower + CarName)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -5274      0      0      0   5274
##
```

```
## Coefficients:
##                                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)                             3983.02    2577.88   1.545 0.127862
## horsepower                                85.69      11.23   7.629 2.85e-10 ***
## CarNamealfa-romero Quadrifoglio         -679.82    3227.23  -0.211 0.833911
## CarNamealfa-romero stelvio              3005.00    3190.88   0.942 0.350297
## CarNameaudi 100 ls                      1226.24    3192.49   0.384 0.702332
## CarNameaudi 100ls                       3956.46    2763.44   1.432 0.157686
## CarNameaudi 4000                        7894.89    3207.47   2.461 0.016889 *
## CarNameaudi 5000                        5510.69    3190.90   1.727 0.089584 .
## CarNameaudi 5000s (diesel)               165.18    3238.01   0.051 0.959492
## CarNameaudi fox                         1840.69    3190.90   0.577 0.566308
## CarNamebmw 320i                         4039.44    2765.67   1.461 0.149626
## CarNamebmw x1                           6618.06    3192.86   2.073 0.042726 *
## CarNamebmw x3                          12026.91    2800.58   4.294 6.89e-05 ***
## CarNamebmw x4                          11180.76    3289.04   3.399 0.001239 **
## CarNamebmw x5                          21735.76    3289.04   6.609 1.42e-08 ***
## CarNamebmw z4                          10213.06    3192.86   3.199 0.002255 **
## CarNamebuick century                   13652.68    3193.73   4.275 7.36e-05 ***
## CarNamebuick century luxus (sw)        13724.68    3193.73   4.297 6.82e-05 ***
## CarNamebuick century special           21209.37    3294.56   6.438 2.73e-08 ***
## CarNamebuick electra 225 custom        11028.68    3193.73   3.453 0.001051 **
## CarNamebuick opel isuzu deluxe         16918.49    3228.93   5.240 2.43e-06 ***
## CarNamebuick regal sport coupe (turbo) 25649.37    3294.56   7.785 1.56e-10 ***
## CarNamebuick skyhawk                   17076.68    3193.73   5.347 1.64e-06 ***
## CarNamebuick skylark                   17790.49    3228.93   5.510 9.00e-07 ***
## CarNamechevrolet impala                -2945.31    3268.41  -0.901 0.371304
## CarNamechevrolet monte carlo           -3686.57    3223.95  -1.143 0.257613
## CarNamechevrolet vega 2300             -3406.57    3223.95  -1.057 0.295132
## CarNamedodge challenger se             -3433.18    3227.23  -1.064 0.291899
## CarNamedodge colt (sw)                 -2201.18    3227.23  -0.682 0.497961
## CarNamedodge colt hardtop              -3118.18    3227.23  -0.966 0.338022
## CarNamedodge coronet custom            -4165.76    3192.49  -1.305 0.197183
## CarNamedodge coronet custom (sw)       -3444.58    3213.66  -1.072 0.288302
## CarNamedodge d200                      -4766.76    3192.49  -1.493 0.140921
## CarNamedodge dart custom               -2603.05    3201.33  -0.813 0.419535
## CarNamedodge monaco (sw)               -3581.18    3227.23  -1.110 0.271799
## CarNamedodge rampage                   -4238.18    3227.23  -1.313 0.194359
## CarNamehonda accord                    -2257.66    2777.62  -0.813 0.419712
## CarNamehonda accord cvcc               -3966.73    3215.01  -1.234 0.222336
## CarNamehonda accord lx                 -3200.73    3215.01  -0.996 0.323673
## CarNamehonda civic                     -1964.31    2640.08  -0.744 0.459911
## CarNamehonda civic (auto)              -2207.37    3193.28  -0.691 0.492210
## CarNamehonda civic 1300                -2257.66    3203.22  -0.705 0.483798
## CarNamehonda civic 1500 gl             -3200.73    3215.01  -0.996 0.323673
## CarNamehonda civic cvcc                -3503.73    2791.21  -1.255 0.214503
## CarNamehonda prelude                   -2507.66    3203.22  -0.783 0.436951
## CarNameisuzu D-Max                      -856.25    2785.24  -0.307 0.759641
## CarNameisuzu D-Max V-Cross             -1065.07    3223.95  -0.330 0.742339
## CarNameisuzu MU-X                      -3882.11    3212.34  -1.208 0.231845
## CarNamejaguar xf                       16484.92    3273.35   5.036 5.09e-06 ***
## CarNamejaguar xj                       13184.92    3273.35   4.028 0.000168 ***
## CarNamejaguar xk                        9565.28    3613.67   2.647 0.010483 *
## CarNamemaxda glc deluxe                -3715.18    3227.23  -1.151 0.254458
## CarNamemaxda rx3                     -4615.18    3227.23  -1.430 0.158155
## CarNamemazda 626                     -1321.87    2616.34  -0.505 0.615341
## CarNamemazda glc                      1610.35    2763.39   0.583 0.562365
## CarNamemazda glc 4                      93.36    3202.25   0.029 0.976844
## CarNamemazda glc custom               -586.28    3205.26  -0.183 0.855518
## CarNamemazda glc custom l            -2686.28    3205.26  -0.838 0.405485
## CarNamemazda glc deluxe               -543.79    2809.24  -0.194 0.847199
## CarNamemazda rx-4                    -2025.73    2791.21  -0.726 0.470961
## CarNamemazda rx-7 gs                  4598.99    2777.06   1.656 0.103204
## CarNamemazda rx2 coupe               -3015.18    3227.23  -0.934 0.354094
## CarNamemercury cougar                -2476.39    3270.86  -0.757 0.452106
## CarNamemitsubishi g4                 -3199.80    2605.36  -1.228 0.224434
## CarNamemitsubishi lancer             -3621.18    3227.23  -1.122 0.266538
## CarNamemitsubishi mirage             -4421.18    3227.23  -1.370 0.176072
## CarNamemitsubishi mirage g4          -3509.50    2610.55  -1.344 0.184160
## CarNamemitsubishi montero            -4535.05    3201.33  -1.417 0.162038
## CarNamemitsubishi outlander          -3855.08    2605.39  -1.480 0.144472
## CarNamemitsubishi pajero             -3335.05    3201.33  -1.042 0.301916
## CarNamenissan clipper                 -470.93    2763.67  -0.170 0.865299
## CarNamenissan dayz                   -3509.43    3223.95  -1.089 0.280933
## CarNamenissan fuga                   -2609.43    3223.95  -0.809 0.421656
## CarNamenissan gt-r                   -1597.16    3252.30  -0.491 0.625250
## CarNamenissan juke                   -2096.87    3225.57  -0.650 0.518254
## CarNamenissan kicks                  -1422.72    3343.82  -0.425 0.672091
## CarNamenissan latio                  -2721.87    2803.37  -0.971 0.335686
## CarNamenissan leaf                   -2596.87    3225.57  -0.805 0.424114
## CarNamenissan note                   -1896.87    3225.57  -0.588 0.558806
## CarNamenissan nv200                  -2746.29    3194.76  -0.860 0.393597
## CarNamenissan otti                   -3509.43    3223.95  -1.089 0.280933
## CarNamenissan rogue                  -3296.58    2781.23  -1.185 0.240818
## CarNamenissan teana                   -494.98    3238.01  -0.153 0.879044
## CarNamenissan titan                  -2546.87    3225.57  -0.790 0.433041
## CarNameNissan versa                  -4396.87    3225.57  -1.363 0.178202
## CarNamepeugeot 304                    1076.10    3195.94   0.337 0.737574
## CarNamepeugeot 504                    3254.80    2443.16   1.332 0.188093
## CarNamepeugeot 504 (sw)                144.71    3194.76   0.045 0.964030
## CarNamepeugeot 505s turbo diesel      4951.10    3195.94   1.549 0.126873
## CarNamepeugeot 604sl                  3387.30    2764.67   1.225 0.225535
## CarNameplymouth cricket              -4766.76    3192.49  -1.493 0.140921
## CarNameplymouth duster               -3644.58    3213.66  -1.134 0.261504
## CarNameplymouth fury gran sedan      -2201.18    3227.23  -0.682 0.497961
## CarNameplymouth fury iii             -3909.68    2805.28  -1.394 0.168823
## CarNameplymouth satellite custom (sw)  -3118.18    3227.23  -0.966 0.338022
## CarNameplymouth valiant              -2603.05    3201.33  -0.813 0.419535
## CarNameporcshce panamera             10806.42    3368.17   3.208 0.002192 **
## CarNameporsche boxter                15306.42    3368.17   4.544 2.91e-05 ***
## CarNameporsche cayenne                7522.08    3160.25   2.380 0.020670 *
## CarNameporsche macan                  5780.81    3211.07   1.800 0.077109 .
## CarNamerenault 12tl                  -2400.44    3199.59  -0.750 0.456201
## CarNamerenault 5 gtl                 -1800.44    3199.59  -0.563 0.575840
## CarNamesaab 99e                       -316.64    2776.51  -0.114 0.909603
## CarNamesaab 99gle                     1278.36    2776.51   0.460 0.646967
## CarNamesaab 99le                       195.69    2763.41   0.071 0.943792
## CarNamesubaru                        -4330.88    2792.01  -1.551 0.126396
## CarNamesubaru baja                   -2078.21    3196.59  -0.650 0.518218
## CarNamesubaru brz                    -3234.89    3207.47  -1.009 0.317453
## CarNamesubaru dl                     -2654.80    2539.79  -1.045 0.300306
## CarNamesubaru r1                     -1776.89    3207.47  -0.554 0.581756
## CarNamesubaru r2                     -2236.00    3190.88  -0.701 0.486312
## CarNamesubaru trezia                 -3546.89    3207.47  -1.106 0.273449
## CarNamesubaru tribeca                -1840.21    3196.59  -0.576 0.567098
## CarNametoyota carina                  -518.02    3238.01  -0.160 0.873462
## CarNametoyota celica gt                459.35    3219.31   0.143 0.887040
## CarNametoyota celica gt liftback     -4282.69    3190.90  -1.342 0.184867
## CarNametoyota corolla                -2194.00    2448.11  -0.896 0.373914
## CarNametoyota corolla 1200           -2795.79    2809.24  -0.995 0.323838
## CarNametoyota corolla 1600 (sw)      -1398.02    3238.01  -0.432 0.667550
## CarNametoyota corolla liftback       -1572.62    2763.85  -0.569 0.571594
## CarNametoyota corolla tercel         -4042.69    3190.90  -1.267 0.210328
## CarNametoyota corona                 -1914.92    2454.29  -0.780 0.438482
## CarNametoyota corona hardtop         -2378.02    3238.01  -0.734 0.465711
## CarNametoyota corona liftback        -5474.47    3191.38  -1.715 0.091705 .
## CarNametoyota corona mark ii         -3948.02    3238.01  -1.219 0.227762
## CarNametoyota cressida                3745.53    3191.38   1.174 0.245420
## CarNametoyota mark ii                -1461.98    2632.65  -0.555 0.580843
## CarNametoyota starlet                -2797.84    2777.62  -1.007 0.318058
## CarNametoyota tercel                 -2724.47    3191.38  -0.854 0.396847
## CarNametoyouta tercel                -1601.21    3230.67  -0.496 0.622063
## CarNamevokswagen rabbit               -664.08    3258.98  -0.204 0.839259
## CarNamevolkswagen 1131 deluxe sedan  -3291.97    3204.22  -1.027 0.308579
## CarNamevolkswagen 411 (sw)           -2771.97    3204.22  -0.865 0.390611
## CarNamevolkswagen dasher              -895.71    2769.98  -0.323 0.747603
## CarNamevolkswagen model 111           -444.08    3258.98  -0.136 0.892093
## CarNamevolkswagen rabbit              -114.31    3190.90  -0.036 0.971549
## CarNamevolkswagen rabbit custom       4034.82    3227.23   1.250 0.216320
## CarNamevolkswagen super beetle        -315.18    3227.23  -0.098 0.922543
## CarNamevolkswagen type 3             -3071.97    3204.22  -0.959 0.341746
## CarNamevolvo 144ea                     506.97    2778.78   0.182 0.855882
## CarNamevolvo 145e (sw)                1140.42    2763.59   0.413 0.681405
## CarNamevolvo 244dl                    4125.98    2767.24   1.491 0.141473
## CarNamevolvo 245                      2762.92    3191.06   0.866 0.390213
## CarNamevolvo 246                      9403.47    3191.38   2.947 0.004648 **
## CarNamevolvo 264gl                    4713.78    2779.98   1.696 0.095415 .
## CarNamevolvo diesel                   1084.63    3241.90   0.335 0.739180
## CarNamevw dasher                      -100.44    3199.59  -0.031 0.975068
## CarNamevw rabbit                     -1715.44    3199.59  -0.536 0.593946
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2256 on 57 degrees of freedom
## Multiple R-squared:  0.9777, Adjusted R-squared:  0.9202
## F-statistic: 17.01 on 147 and 57 DF,  p-value: < 2.2e-16
```

## Simple Linear Regression

From above result we can say that car name provides too much information regarding the car price. In this project we are more focused on getting car price from car inbuilt features like size, engine quality, top-speed.

We will be avoiding the carnames from now on as feature list. IN this pae we apply simple linear regression on the the car price dataset. As we are selecting only one feaures.

```
lm.fit = lm(price~enginesize)
lm.fit
```

```
##
## Call:
## lm(formula = price ~ enginesize)
##
## Coefficients:
## (Intercept)   enginesize
##     -8005.4        167.7
```

After fitting the model by calling names and coefficient the r will return model parameters and coefficients for considered variables.

```
names(lm.fit)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

```
coef(lm.fit)
```

```
## (Intercept)  enginesize
##  -8005.4455    167.6984
```

```
confint(lm.fit)
```

```
##                 2.5 %     97.5 %
## (Intercept) -9727.1913 -6283.6997
## enginesize    154.8047   180.5922
```

After fitting the model we can see the the performance of model fit. By summarizing the different statistical parameters; F-score, significance for the model we can understand the model fitness. We can see the p-value low for the significance and reject the null-hypothesis

```
summary(lm.fit)
```

```
##
## Call:
## lm(formula = price ~ enginesize)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10664.2  -2225.0   -482.4   1588.0  14271.5
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8005.446    873.221  -9.168   <2e-16 ***
## enginesize    167.698      6.539  25.645   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3889 on 203 degrees of freedom
## Multiple R-squared:  0.7641, Adjusted R-squared:  0.763
## F-statistic: 657.6 on 1 and 203 DF,  p-value: < 2.2e-16
```

As we have a trained model we can use the model to predict the car price from car features using the trained linear regression model. We can also plot the output and variables. In R we do this by as follows

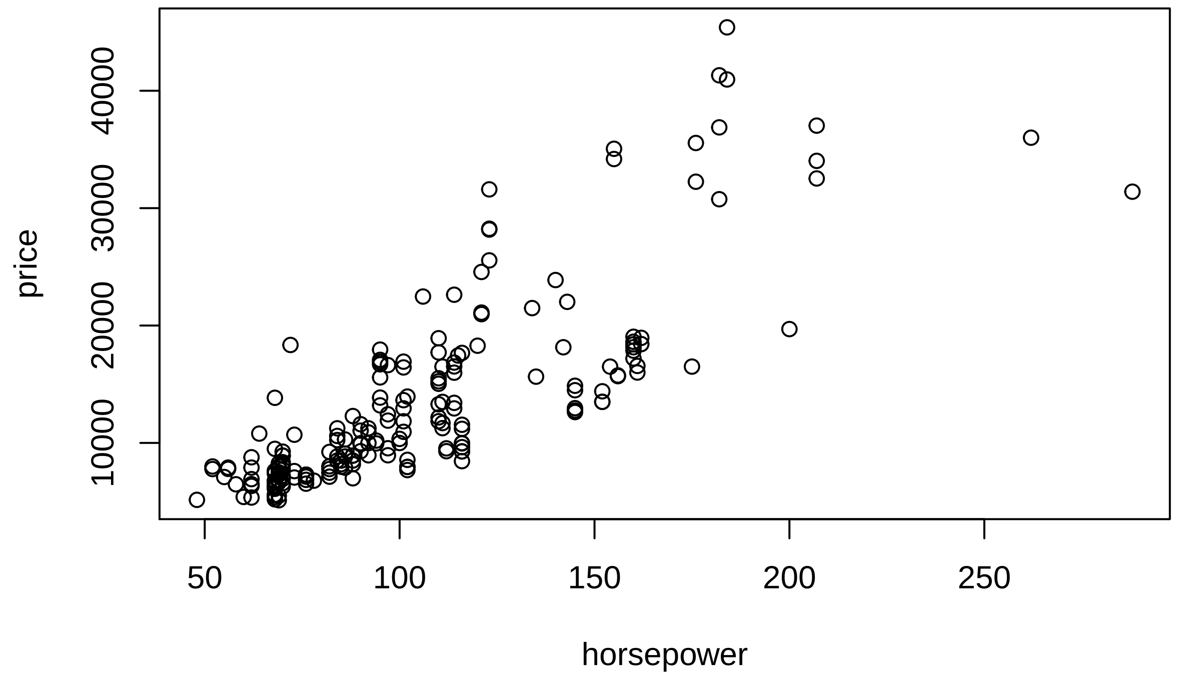
```
predict(lm.fit,data.frame(enginesize=(c(5,10,15))), interval="confidence")
```

```
##         fit       lwr       upr
## 1 -7166.953 -8827.550 -5506.356
## 2 -6328.461 -7928.170 -4728.752
## 3 -5489.969 -7029.082 -3950.857
```

```
predict(lm.fit,data.frame(enginesize=(c(5,10,15))), interval="prediction")
```

```
##         fit       lwr      upr
## 1 -7166.953 -15013.59  679.686
## 2 -6328.461 -14162.44 1505.518
## 3 -5489.969 -13311.80 2331.861
```

```
plot(horsepower, price)
```



Now considering adding more variables

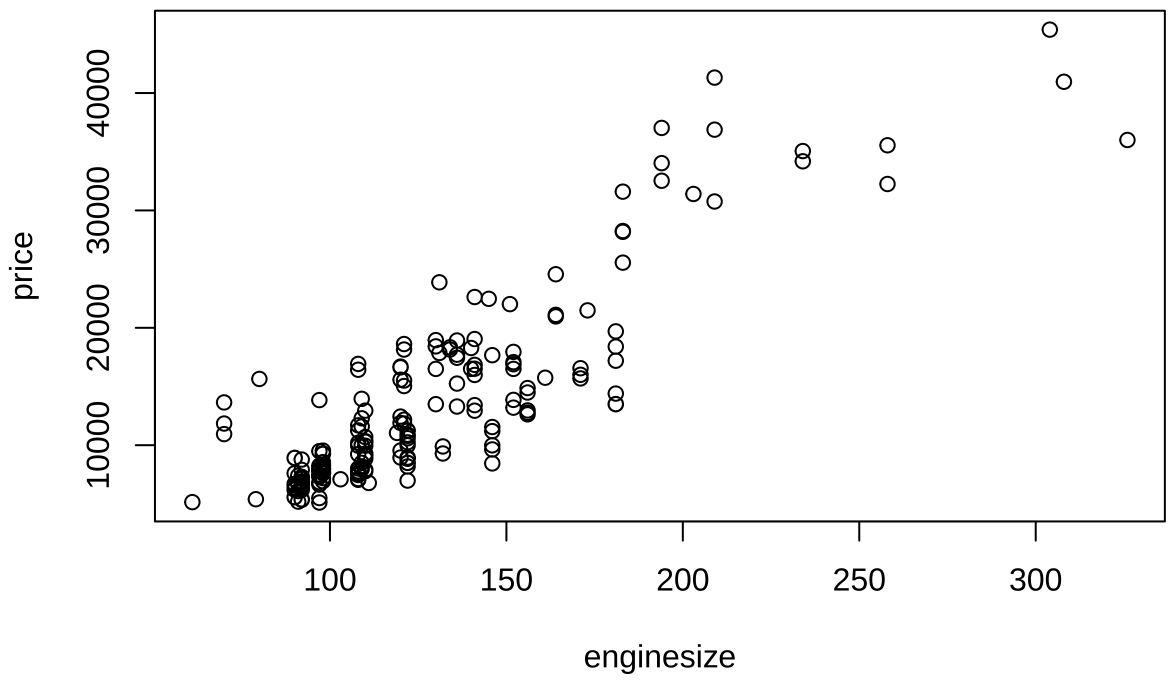
## Multiple Linear Regression

In the next section I will be using more than one variables and examine the summary of the linear regression model. For this we choose the car length and horsepower for discussing the results more clearly. We select the car length and horsepower by observing the summary of model from earlier analysis. The three stars shows their significance.

```
lm.fit = lm(price~carlength+horsepower)
lm.fit
```

```
##
## Call:
## lm(formula = price ~ carlength + horsepower)
##
## Coefficients:
## (Intercept)    carlength   horsepower
##    -38111.7        220.3        125.3
```

```
plot(enginesize, price)
```



In this experiment we conduct our analysis by incorporting more variables. We retrain the model using different features sets. We reevaluate the performance by considering the selected features set.

```r
lm.fit = lm(price~enginelocation+enginesize+carlength+aspiration
            +curbweight+drivewheel, data =  card)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = price ~ enginelocation + enginesize + carlength +
##     aspiration + curbweight + drivewheel, data = card)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7527.3 -1503.2  -100.9  1365.2 15175.9
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -14656.455   4933.580  -2.971  0.00334 **
## enginelocationrear  13029.009   2024.335   6.436 9.12e-10 ***
## enginesize             96.835     12.597   7.687 6.98e-13 ***
## carlength              19.853     42.969   0.462  0.64457
## aspirationturbo       438.735    694.900   0.631  0.52853
## curbweight              4.420      1.741   2.539  0.01189 *
## drivewheelfwd         -56.037   1208.333  -0.046  0.96306
## drivewheelrwd        1770.224   1196.704   1.479  0.14067
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3242 on 197 degrees of freedom
## Multiple R-squared:  0.841,  Adjusted R-squared:  0.8354
## F-statistic: 148.9 on 7 and 197 DF,  p-value: < 2.2e-16
```

In the following section, I will be implement the polymial regression by using polynomials of features and lm function in R.

```r
summary(lm(price~poly(peakrpm,4)+aspiration+carlength+carheight
           +curbweight+fuelsystem+doornumber+wheelbase
           +enginetype, data =card))
```

```
##
## Call:
## lm(formula = price ~ poly(peakrpm, 4) + aspiration + carlength +
##     carheight + curbweight + fuelsystem + doornumber + wheelbase +
##     enginetype, data = card)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10540.7  -1989.4   -518.2   1636.9  14403.2
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -20524.628   9427.083  -2.177 0.030760 *
## poly(peakrpm, 4)1  21768.804   5075.402   4.289 2.92e-05 ***
## poly(peakrpm, 4)2  18325.242   4574.880   4.006 9.02e-05 ***
## poly(peakrpm, 4)3 -15038.206   4186.121  -3.592 0.000422 ***
## poly(peakrpm, 4)4 -23200.508   5173.456  -4.485 1.29e-05 ***
## aspirationturbo    -1177.121    899.384  -1.309 0.192258
## carlength            -25.794     66.235  -0.389 0.697409
## carheight           -246.427    160.700  -1.533 0.126908
## curbweight            14.575      1.437  10.142  < 2e-16 ***
## fuelsystem2bbl      6464.969   1594.720   4.054 7.47e-05 ***
## fuelsystem4bbl      4375.865   4428.886   0.988 0.324458
## fuelsystemidi       5393.846   1865.420   2.891 0.004304 **
## fuelsystemmfi       2679.635   4111.384   0.652 0.515383
## fuelsystemmpfi      6151.123   1625.480   3.784 0.000209 ***
## fuelsystemspdi      3046.670   2149.272   1.418 0.158045
## fuelsystemspfi       981.934   4009.347   0.245 0.806803
## doornumbertwo       1773.610    654.866   2.708 0.007410 **
## wheelbase             71.127    107.366   0.662 0.508510
## enginetypedohcv     -133.867   3919.097  -0.034 0.972789
## enginetypel        -2586.584   1758.206  -1.471 0.142987
## enginetypeohc       1497.530   1334.909   1.122 0.263424
## enginetypeohcf      2270.551   1659.947   1.368 0.173056
## enginetypeohcv      1520.122   1546.661   0.983 0.326998
## enginetyperotor    -7374.158   4055.943  -1.818 0.070700 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3569 on 181 degrees of freedom
## Multiple R-squared:  0.8229, Adjusted R-squared:  0.8004
## F-statistic: 36.58 on 23 and 181 DF,  p-value: < 2.2e-16
```

From the F-staticstics value 134 which is much higer than 1, it is evident that at least one features are related to the output variable car price.

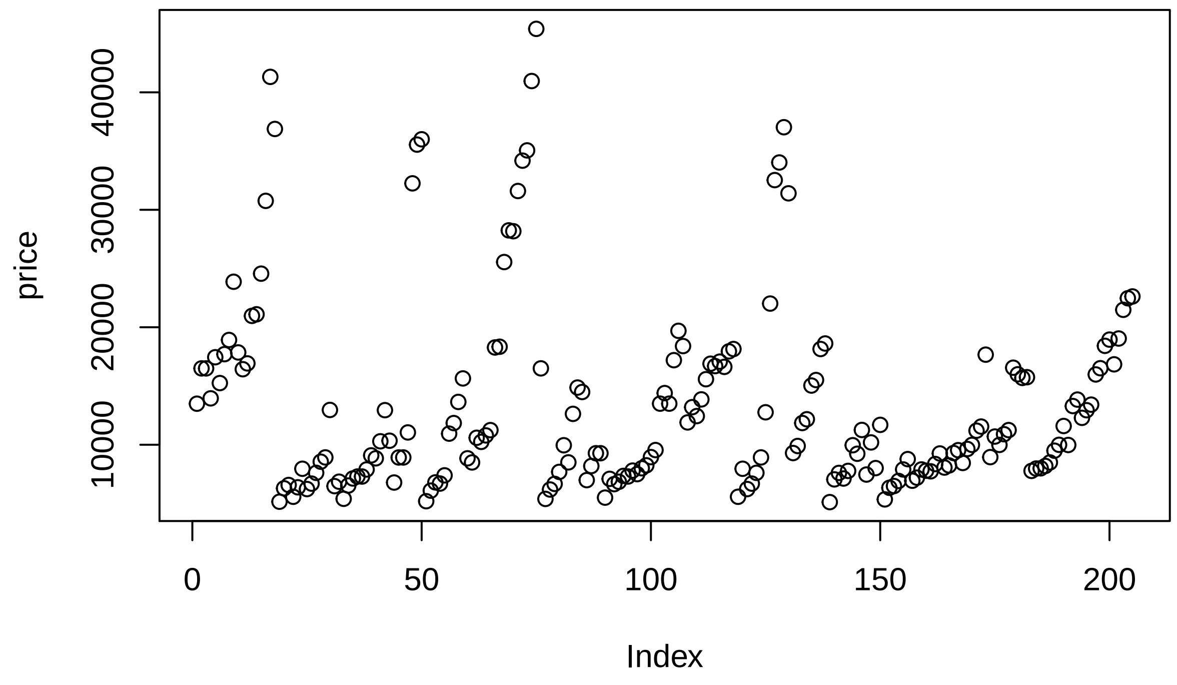
# Chapter 4

In the car dataset the output predictor is car price;a continuous varible. To apply classification techniques I reorganize my output predictor variable price as high and low based on median threshold. I have observe the price median as 10300. We label the price greater than threshold are high Yes and below No. We have created a classification problem of car price high or low in the car price dataset. We try to predict the car price label based on the data features. First we load the data.

```r
library(ISLR)


dim(card)
```

```
## [1] 205  26
```

```r
plot(price)
```



```r
high = as.factor(ifelse(price<=10300, "No", "Yes"))

card = data.frame(card, high)
```

## Logistic Regression

We fit the model as targetting the created class labels high.

```r
glm.fits=glm(as.numeric(high)~fuelsystem+peakrpm+citympg
             + enginesize+enginetype+carwidth+curbweight+carlength,
             data = card)
```

After fitting the model we look into the fitted model by summary function.

```r
summary(glm.fits)
```

```
##
## Call:
## glm(formula = as.numeric(high) ~ fuelsystem + peakrpm + citympg +
##     enginesize + enginetype + carwidth + curbweight + carlength,
##     data = card)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.65836  -0.14655   0.00762   0.13548   0.79154
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -4.709e-01  1.317e+00  -0.357  0.72119
## fuelsystem2bbl   3.976e-02  1.050e-01   0.379  0.70529
## fuelsystem4bbl   4.409e-01  3.507e-01   1.257  0.21025
## fuelsystemidi    3.301e-01  1.427e-01   2.313  0.02184 *
## fuelsystemmfi    5.407e-01  3.099e-01   1.745  0.08271 .
## fuelsystemmpfi   3.796e-01  1.112e-01   3.413  0.00079 ***
## fuelsystemspdi   1.173e-01  1.424e-01   0.824  0.41112
## fuelsystemspfi   6.309e-01  3.084e-01   2.046  0.04217 *
## peakrpm          9.905e-06  6.071e-05   0.163  0.87056
## citympg         -1.367e-02  8.064e-03  -1.695  0.09167 .
## enginesize      -1.174e-03  1.238e-03  -0.949  0.34396
## enginetypedohcv -2.158e-01  3.321e-01  -0.650  0.51658
## enginetypel      1.057e-01  1.324e-01   0.798  0.42579
## enginetypeohc    3.080e-02  9.804e-02   0.314  0.75374
## enginetypeohcf  -3.959e-02  1.222e-01  -0.324  0.74642
## enginetypeohcv  -4.262e-02  1.292e-01  -0.330  0.74195
## enginetyperotor  2.020e-01  3.153e-01   0.641  0.52258
## carwidth         2.712e-02  2.255e-02   1.202  0.23079
## curbweight       5.022e-04  1.622e-04   3.095  0.00227 **
## carlength       -5.292e-03  4.353e-03  -1.216  0.22557
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.08278914)
##
##     Null deviance: 51.249  on 204  degrees of freedom
## Residual deviance: 15.316  on 185  degrees of freedom
## AIC: 91.972
##
## Number of Fisher Scoring iterations: 2
```

```r
coef(glm.fits)
```

```
##     (Intercept)  fuelsystem2bbl  fuelsystem4bbl   fuelsystemidi   fuelsystemmfi
##   -4.708631e-01    3.975924e-02    4.409199e-01    3.301087e-01    5.406812e-01
##  fuelsystemmpfi  fuelsystemspdi  fuelsystemspfi         peakrpm         citympg
##    3.795650e-01    1.173331e-01    6.309112e-01    9.905254e-06   -1.367271e-02
##      enginesize enginetypedohcv     enginetypel   enginetypeohc  enginetypeohcf
##   -1.174460e-03   -2.158418e-01    1.056797e-01    3.080150e-02   -3.958632e-02
##  enginetypeohcv enginetyperotor        carwidth      curbweight       carlength
##   -4.261800e-02    2.019611e-01    2.711590e-02    5.021979e-04   -5.292212e-03
```

```r
summary(glm.fits)$coef
```

```
##                      Estimate   Std. Error    t value     Pr(>|t|)
## (Intercept)     -4.708631e-01 1.317430e+00 -0.3574102 0.7211921146
## fuelsystem2bbl   3.975924e-02 1.049685e-01  0.3787732 0.7052906342
## fuelsystem4bbl   4.409199e-01 3.507004e-01  1.2572554 0.2102460397
## fuelsystemidi    3.301087e-01 1.427342e-01  2.3127519 0.0218365932
## fuelsystemmfi    5.406812e-01 3.099105e-01  1.7446368 0.0827087800
## fuelsystemmpfi   3.795650e-01 1.112147e-01  3.4129028 0.0007895722
## fuelsystemspdi   1.173331e-01 1.424302e-01  0.8237940 0.4111173753
## fuelsystemspfi   6.309112e-01 3.083676e-01  2.0459709 0.0421744484
## peakrpm          9.905254e-06 6.070519e-05  0.1631698 0.8705628429
## citympg         -1.367271e-02 8.064297e-03 -1.6954627 0.0916699914
## enginesize      -1.174460e-03 1.237837e-03 -0.9487995 0.3439603704
## enginetypedohcv -2.158418e-01 3.321260e-01 -0.6498793 0.5165763097
## enginetypel      1.056797e-01 1.324017e-01  0.7981750 0.4257923143
## enginetypeohc    3.080150e-02 9.803978e-02  0.3141735 0.7537432676
## enginetypeohcf  -3.958632e-02 1.222364e-01 -0.3238506 0.7464169938
## enginetypeohcv  -4.261800e-02 1.292357e-01 -0.3297694 0.7419472714
## enginetyperotor  2.019611e-01 3.152725e-01  0.6405924 0.5225802287
## carwidth         2.711590e-02 2.255351e-02  1.2022916 0.2307871882
## curbweight       5.021979e-04 1.622369e-04  3.0954601 0.0022705322
## carlength       -5.292212e-03 4.352501e-03 -1.2159012 0.2255717778
```

```r
summary(glm.fits)$coef[,4]
```

```
##     (Intercept)  fuelsystem2bbl  fuelsystem4bbl   fuelsystemidi   fuelsystemmfi
##    0.7211921146    0.7052906342    0.2102460397    0.0218365932    0.0827087800
##  fuelsystemmpfi  fuelsystemspdi  fuelsystemspfi         peakrpm         citympg
##    0.0007895722    0.4111173753    0.0421744484    0.8705628429    0.0916699914
##      enginesize enginetypedohcv     enginetypel   enginetypeohc  enginetypeohcf
##    0.3439603704    0.5165763097    0.4257923143    0.7537432676    0.7464169938
##  enginetypeohcv enginetyperotor        carwidth      curbweight       carlength
##    0.7419472714    0.5225802287    0.2307871882    0.0022705322    0.2255717778
```

From the summary above we see the significance of the fuelsystem and curbweight are highest based on their smaller p-value.

Now we check the classifiers performance based on its decision on the dataset. For that we create the confusion matrix for the classifier.

```r
glm.probs=predict(glm.fits,type="response")
glm.probs[1:10]
```

```
##        1        2        3        4        5        6        7        8
## 1.642826 1.642826 1.765080 1.571926 1.872245 1.692959 1.918991 1.974233
##        9       10
## 2.073741 2.052672
```

```r
contrasts(high)
```

```
##     Yes
## No    0
## Yes   1
```

```r
glm.pred=rep("No", 205)
glm.pred[glm.probs>1.5]="Yes"
table(glm.pred, high)
```

```
##          high
## glm.pred No Yes
##      No  89   4
##      Yes 14  98
```

```
mean(glm.pred==high)
```

```
## [1] 0.9121951
```

From the result we can see that the model have correctly classified 89 no instances and 98 No instances. The model accuracy is 91.22% in the training instances.

Now in next case we only consider case where peakrpm is lower than 6000 we devide our dataset by taking the instances where peak values are smaller than 6000. We refit the model using the cropped dataset.

```
train=(peakrpm<6000)
ccard.6000=card[!train,]
dim(ccard.6000)
```

```
## [1] 11 27
```

```
high.6000=high[!train]


glm.fits=glm(as.numeric(high)~peakrpm+citympg
             + enginesize+carwidth+curbweight+carlength,
             data = card, subset = train)


glm.probs=predict(glm.fits,ccard.6000,type="response")
```

By selecting appropriate threshold we can get the prediction form the model for the 11 test dataset.

```
glm.pred=rep("No",11)
glm.pred[glm.probs>1.5]="Yes"
table(glm.pred,as.factor(high.6000))
```

```
##
## glm.pred No Yes
##      No   7   0
##      Yes  0   4
```

```
mean(glm.pred==high.6000)
```

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

```
mean(glm.pred!=high.6000)
```

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

Here the model sucessfully classifed all the instances from the features; 7 no and 4 yes classes

We can consider only 3 variables to check the synergy. For that we take 3 variables to fit the high class.

```
glm.fits=glm(as.factor(high)~carwidth+curbweight+enginesize,
             data=card,family=binomial,subset=train)
glm.probs=predict(glm.fits,ccard.6000,type="response")
glm.pred=rep("No",11)
glm.pred[glm.probs>.2]="Yes"
table(glm.pred,high.6000)
```

```
##         high.6000
## glm.pred No Yes
##      No   7   0
##      Yes  0   4
```

```
mean(glm.pred==high.6000)
```

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

After preparing the model, we can see that the model again perform well on the test dataset. It again classified all the test instance correctly. We can see the result in the above confusion matrix.

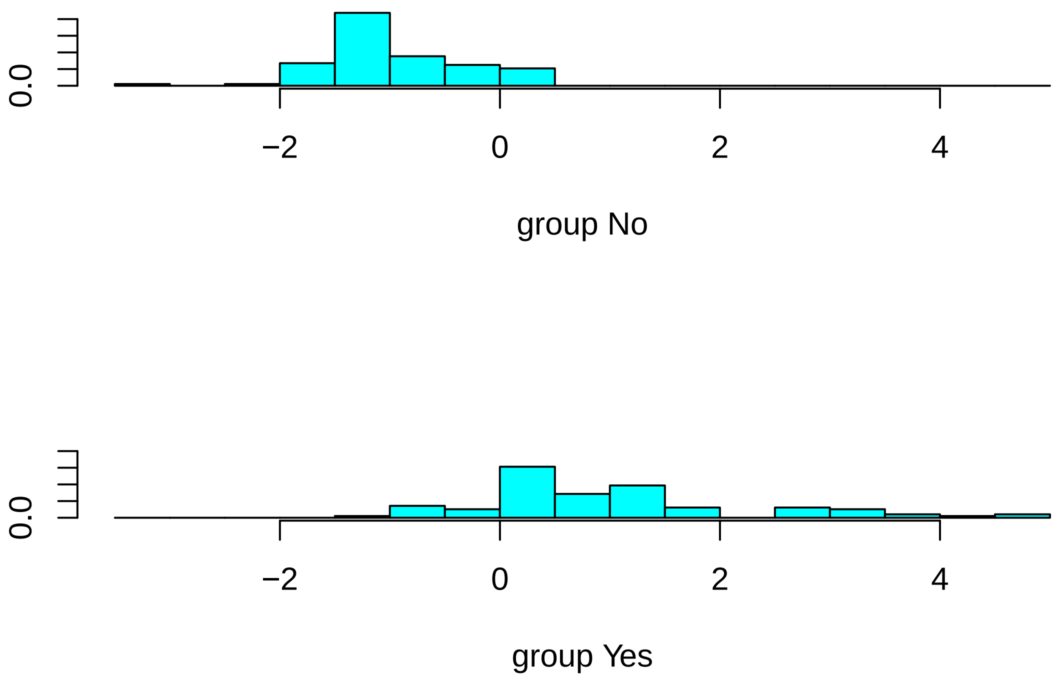
## Linear Discriminant Analysis

In this section we implement LDA on the high class for car data we created in earlier example.

```
library(MASS)

lda.fit=lda(as.factor(high)~carwidth+enginesize,
            data=card,subset=train) # fitting model

lda.fit # provides summary
```

```
## Call:
## lda(as.factor(high) ~ carwidth + enginesize, data = card, subset = train)
##
## Prior probabilities of groups:
##        No       Yes
## 0.4948454 0.5051546
##
## Group means:
##     carwidth enginesize
## No  64.47917   103.2812
## Yes 67.45408   154.6429
##
## Coefficients of linear discriminants:
##                   LD1
## carwidth   0.46165129
## enginesize 0.01191789
```

```
plot(lda.fit)
```



Frob above plot we see the difference in distribution between high and low class for the car price. We can use the previous model to predict new outcome

```
lda.pred=predict(lda.fit, ccard.6000)

names(lda.pred)
```

```
## [1] "class"     "posterior" "x"
```

```
lda.class=lda.pred$class


table(lda.class,high.6000)
```

```
##          high.6000
## lda.class No Yes
##       No   7   4
##       Yes  0   0
```

```
mean(lda.class==high.6000)
```

```
## [1] 0.6363636
```

The model failed to correctly classify any yes instance in the previous section. It predicted all as NO. We can change the threshold to check result across the threshold.

```
# changing default threshold
sum(lda.pred$posterior[,1]>=.9)
```

```
## [1] 7
```

```
sum(lda.pred$posterior[,1]<.9)
```

```
## [1] 4
```

```
lda.pred$posterior[1:11,1]
```

```
##        32        34        35        36        37        56        57        58
## 0.9398168 0.9344186 0.9344186 0.9344186 0.9398168 0.8346551 0.8346551 0.8346551
##        59       166       167
## 0.7993691 0.9251628 0.9251628
```

```
lda.class[1:11]
```

```
##  [1] No No No No No No No No No No No
## Levels: No Yes
```

```
sum(lda.pred$posterior[,1]>.9)
```

```
## [1] 7
```

In this case by changing the threshold we get better prediciton for the same model.

## Quadratic Discriminant Analysis

```
qda.fit=qda(as.factor(high)~carwidth+enginesize,data=Smarket,subset=train)
qda.fit
```

```
## Call:
## qda(as.factor(high) ~ carwidth + enginesize, data = Smarket,
##     subset = train)
##
## Prior probabilities of groups:
##        No       Yes
## 0.4948454 0.5051546
##
## Group means:
##     carwidth enginesize
## No  64.47917   103.2812
## Yes 67.45408   154.6429
```

```
qda.class=predict(qda.fit, ccard.6000)$class
table(qda.class, high.6000)
```

```
##          high.6000
## qda.class No Yes
##       No   7   0
##       Yes  0   4
```

```
mean(qda.class==high.6000)
```

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

From the output we can see that quadradic discrimant analysis provide accurate result on classifying test data. It predicted all 7 no and 4 yes class correctly.

## K-Nearest Neighbors

In this section, we implement k-nearest Neighbors for the car data to classify the car price as high or no.

```
# need class library
library(class)
train.X=cbind(carwidth,enginesize, fuelsystem, curbweight)[train,]

test.X=cbind(carwidth,enginesize, fuelsystem, curbweight)[!train,]

train.high=high[train]

set.seed(1) # reproducible result
# 4 arguments
knn.pred=knn(train.X,test.X,train.high,k=5)

table(knn.pred,high.6000)
```

```
##         high.6000
## knn.pred No Yes
##      No   7   3
##      Yes  0   1
```

```
# increasing the number of K (3 in this case)
knn.pred=knn(train.X,test.X,train.high,k=3)

table(knn.pred, high.6000)
```

```
##         high.6000
## knn.pred No Yes
##      No   7   1
##      Yes  0   3
```

```
mean(knn.pred==high.6000)
```

```
## [1] 0.9090909
```

The KNN failed to classify the car price high class. I have conducted experiment for different k values (5,10,15,25). In the previous result, the classifier correctly classified 10 intance out of 11 when KNN k-value is 3. By changing k to 5, the model loose it performance and detect 8 correctly out of 11 test instancs.

# Chapter 5

In this section, I implement some of the resampling method on the car data from the textbook. Firstly, the validation set approach has been discussed.

## Validation Set Approach

The car dataset contains 205 instances. For this experiment, I take 195 for training and rest 10 for validation set. The random seed is here to remove the selection bias.

```
# seed 1
set.seed(1)
#attach(card)
train =  sample(205, 195)
```

The train dataset consists of 195 instances randomly taken from the 205 instances. The model is trained using this 195 instances.

```
lm.fit = lm(price~ curbweight+carwidth + peakrpm+horsepower+
             +carlength+fueltype +carbody +enginesize+carheight,
            data = card, subset= train)

summary(lm.fit)
```

```
##
## Call:
## lm(formula = price ~ curbweight + carwidth + peakrpm + horsepower +
##     +carlength + fueltype + carbody + enginesize + carheight,
##     data = card, subset = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8837.4 -1973.8    58.9  1463.0 14134.5
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -6.719e+04  1.391e+04  -4.832 2.87e-06 ***
## curbweight        2.845e+00  1.665e+00   1.708 0.089283 .
## carwidth          6.619e+02  2.379e+02   2.782 0.005972 **
## peakrpm           2.501e+00  6.851e-01   3.651 0.000341 ***
## horsepower        4.003e+01  1.403e+01   2.853 0.004833 **
## carlength        -5.428e+01  5.369e+01  -1.011 0.313434
## fueltypegas      -2.149e+03  9.956e+02  -2.158 0.032211 *
## carbodyhardtop   -2.586e+03  1.790e+03  -1.445 0.150165
## carbodyhatchback -5.627e+03  1.417e+03  -3.972 0.000103 ***
## carbodysedan     -4.640e+03  1.464e+03  -3.170 0.001791 **
## carbodywagon     -6.559e+03  1.631e+03  -4.022 8.46e-05 ***
## enginesize        8.996e+01  1.374e+01   6.546 5.85e-10 ***
## carheight         3.263e+02  1.418e+02   2.301 0.022537 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3183 on 182 degrees of freedom
## Multiple R-squared:  0.8559, Adjusted R-squared:  0.8464
## F-statistic: 90.07 on 12 and 182 DF,  p-value: < 2.2e-16
```

```
mean((card$price-predict(lm.fit,card))[-train]^2)
```

```
## [1] 3626699
```

From above, we can see that the carbody, enginesize and incept has the smallest p-value. The F-statistic value is higher than 1. And R-squared value is close to one, meaning covering the data variance by the features.

We get the error of 3626699 for training a linear model. We will compare polynomial of two and three model for the same features on the same test dataset.

In the next section, I will use quadratic regression to experiment with the validation set approach. I will implement different model and check their performance based on the validation set result.

```
# preparing quadratic regression
lm.fit2 = lm(price~poly(curbweight,2)+poly(carwidth,2)+peakrpm+horsepower+
             +carlength+fueltype +carbody +enginesize+carheight,
             data = card, subset= train)

summary(lm.fit2)
```

```
##
## Call:
## lm(formula = price ~ poly(curbweight, 2) + poly(carwidth, 2) +
##     peakrpm + horsepower + +carlength + fueltype + carbody +
##     enginesize + carheight, data = card, subset = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7388.6 -1605.6  -146.8  1244.0 14254.8
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -2.562e+04  1.108e+04  -2.312 0.021926 *
## poly(curbweight, 2)1  1.834e+04  1.177e+04   1.558 0.120889
## poly(curbweight, 2)2  9.579e+03  4.704e+03   2.036 0.043190 *
## poly(carwidth, 2)1    1.181e+04  7.100e+03   1.663 0.098084 .
## poly(carwidth, 2)2    1.110e+04  4.043e+03   2.746 0.006640 **
## peakrpm               1.990e+00  6.554e-01   3.036 0.002750 **
## horsepower            6.421e+01  1.471e+01   4.365 2.14e-05 ***
## carlength             5.409e+01  5.571e+01   0.971 0.332938
## fueltypegas          -2.496e+03  9.424e+02  -2.649 0.008798 **
## carbodyhardtop       -2.833e+03  1.691e+03  -1.675 0.095634 .
## carbodyhatchback     -6.143e+03  1.370e+03  -4.484 1.30e-05 ***
## carbodysedan         -5.378e+03  1.419e+03  -3.790 0.000205 ***
## carbodywagon         -7.345e+03  1.575e+03  -4.663 6.04e-06 ***
## enginesize            5.748e+01  1.532e+01   3.753 0.000236 ***
## carheight             2.445e+02  1.355e+02   1.804 0.072903 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3004 on 180 degrees of freedom
## Multiple R-squared:  0.873,  Adjusted R-squared:  0.8631
## F-statistic: 88.39 on 14 and 180 DF,  p-value: < 2.2e-16
```

```
# Prediction with rest
mean((card$price-predict(lm.fit2,card))[-train]^2)
```

```
## [1] 4236347
```

After training order 2 polynomials the prediction error in the model are 4236347 in the test dataset.

```
# preparing cubic regression
lm.fit3 = lm(price~poly(curbweight,3)+poly(carwidth,3)+peakrpm+horsepower+
             +carlength+fueltype +carbody +enginesize+carheight,
             data = card, subset= train)

mean((card$price-predict(lm.fit3,card))[-train]^2)
```

```
## [1] 4489315
```

By using the, the third order polynomial on curbwidth, we see the new errors are 4489315, slightly higher than the quadratic polynomial (4236347) on the first two features and also smaller than the linear regression model whose error was 3626699 To summarize the result, the linear worked best on the linear regression model.

By changing seed and re-evaluating the same model we can expect a slight different result. The seed changes the 10 test data samples randomly.

```
set.seed(4)
#attach(card)

train =  sample(205, 195)

lm.fit = lm(price~ curbweight+carwidth+peakrpm+horsepower+
             +carlength+fueltype +carbody +enginesize+carheight,
            data = card, subset= train)

summary(lm.fit)
```

```
##
```

```
## Call:
## lm(formula = price ~ curbweight + carwidth + peakrpm + horsepower +
##     +carlength + fueltype + carbody + enginesize + carheight,
##     data = card, subset = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8932.5 -1892.7    -8.5  1435.4 14157.1
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -6.413e+04  1.399e+04  -4.584 8.46e-06 ***
## curbweight        3.052e+00  1.677e+00   1.819 0.070499 .
## carwidth          6.598e+02  2.393e+02   2.757 0.006419 **
## peakrpm           2.183e+00  6.559e-01   3.328 0.001059 **
## horsepower        4.318e+01  1.403e+01   3.077 0.002414 **
## carlength        -6.422e+01  5.328e+01  -1.205 0.229645
## fueltypegas      -2.006e+03  9.846e+02  -2.037 0.043115 *
## carbodyhardtop   -2.510e+03  1.790e+03  -1.402 0.162724
## carbodyhatchback -5.542e+03  1.421e+03  -3.901 0.000135 ***
## carbodysedan     -4.450e+03  1.462e+03  -3.043 0.002692 **
## carbodywagon     -6.446e+03  1.640e+03  -3.931 0.000120 ***
## enginesize        8.675e+01  1.386e+01   6.260 2.69e-09 ***
## carheight         3.221e+02  1.394e+02   2.310 0.021985 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3185 on 182 degrees of freedom
## Multiple R-squared:  0.8568, Adjusted R-squared:  0.8473
## F-statistic: 90.73 on 12 and 182 DF,  p-value: < 2.2e-16
```

```r
mean((card$price-predict(lm.fit,card))[-train]^2)
```

```
## [1] 3290316
```

From the newdata sampling, the new error on the linear regression model is 3290316, slightly smaller than earlier experiment with seed 1. Next we train the quadratic model.

```r
# preparing quadratic regression
lm.fit2 = lm(price~poly(curbweight,2)+poly(carwidth,2)+peakrpm+horsepower+
            +carlength+fueltype +carbody +enginesize+carheight,
            data = card, subset= train)


# Prediction with rest
mean((card$price-predict(lm.fit2,card))[-train]^2)
```

```
## [1] 2497317
```

In the quadratic model the new error on test set become the error is 2497317, almost half of the earlier example. In this case it seems quadradic model is better than linear model.

```r
# preparing cubic regression
lm.fit3 = lm(price~poly(curbweight,3)+poly(carwidth,3)+peakrpm+horsepower+
             +carlength+fueltype +carbody +enginesize+carheight,
             data = card, subset= train)
summary(lm.fit3)
```

```
##
## Call:
## lm(formula = price ~ poly(curbweight, 3) + poly(carwidth, 3) +
##     peakrpm + horsepower + +carlength + fueltype + carbody +
##     enginesize + carheight, data = card, subset = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7676.9 -1586.7   -92.4  1164.9 14391.7
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -2.234e+04  1.118e+04  -1.998 0.047231 *
## poly(curbweight, 3)1  2.475e+04  1.256e+04   1.971 0.050292 .
## poly(curbweight, 3)2  9.458e+03  4.793e+03   1.973 0.050002 .
## poly(curbweight, 3)3 -1.365e+03  3.953e+03  -0.345 0.730326
## poly(carwidth, 3)1    8.686e+03  7.848e+03   1.107 0.269878
## poly(carwidth, 3)2    1.129e+04  4.126e+03   2.735 0.006870 **
## poly(carwidth, 3)3    4.102e+03  3.926e+03   1.045 0.297496
## peakrpm               1.581e+00  6.375e-01   2.480 0.014055 *
## horsepower            6.601e+01  1.513e+01   4.364 2.16e-05 ***
## carlength             4.430e+01  5.711e+01   0.776 0.438974
## fueltypegas          -2.230e+03  9.371e+02  -2.379 0.018413 *
## carbodyhardtop       -2.705e+03  1.703e+03  -1.588 0.114105
## carbodyhatchback     -5.831e+03  1.388e+03  -4.200 4.20e-05 ***
## carbodysedan         -5.007e+03  1.430e+03  -3.501 0.000587 ***
## carbodywagon         -7.349e+03  1.598e+03  -4.598 8.05e-06 ***
## enginesize            5.214e+01  1.621e+01   3.216 0.001546 **
## carheight             2.544e+02  1.381e+02   1.842 0.067082 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3025 on 178 degrees of freedom
## Multiple R-squared:  0.8737, Adjusted R-squared:  0.8623
## F-statistic: 76.93 on 16 and 178 DF,  p-value: < 2.2e-16
```

```r
mean((card$price-predict(lm.fit3,card))[-train]^2)
```

```
## [1] 2714493
```

In this case, the quadratic model gets the error of 2714493, a little higher than quadratic model. In this test case, the quadratic model performed best.

### Leave one-out-cross validation

In this resampling method, we put one training instance as the test example to check the model performances.

```r
# used all continuous value predictor
glm.fit =  glm(price~ curbweight+carwidth+peakrpm+horsepower+
              +carlength+fueltype +carbody +enginesize+carheight ,
              data = card)
coef(glm.fit)
```

```
##      (Intercept)       curbweight         carwidth          peakrpm
##    -65287.278532         2.950661       660.153937         2.257717
##       horsepower        carlength      fueltypegas   carbodyhardtop
##        41.718643       -62.625543     -2012.332996     -2546.677919
## carbodyhatchback     carbodysedan     carbodywagon       enginesize
##     -5555.412413     -4573.041023     -6553.308168        88.441393
##        carheight
##       335.571161
```

```r
lm.fit =  lm(price~  curbweight+carwidth+peakrpm+horsepower+
              +carlength+fueltype +carbody +enginesize+carheight ,
              data = card)
coef(lm.fit)
```

```
##      (Intercept)       curbweight         carwidth          peakrpm
##    -65287.278532         2.950661       660.153937         2.257717
##       horsepower        carlength      fueltypegas   carbodyhardtop
##        41.718643       -62.625543     -2012.332996     -2546.677919
## carbodyhatchback     carbodysedan     carbodywagon       enginesize
##     -5555.412413     -4573.041023     -6553.308168        88.441393
##        carheight
##       335.571161
```

The glm and lm provided the same result, which is evident by previous result of th coefficient values

```r
#Library
library(boot)
set.seed(1)
glm.fit=  glm(price~ curbweight+carwidth+peakrpm+horsepower+
              +carlength+fueltype +carbody +enginesize+carheight ,
              data = card)

cv.err = cv.glm(card, glm.fit)
cv.err$delta
```

```
## [1] 11009074 11004207
```

We get the error value of 11M.

```r
  # Polynomial

cv.error = rep(0,5)

for (i in 1:5){
    glm.fit =  glm(price~ poly(curbweight, i)+carwidth+peakrpm+horsepower+
             +carlength+fueltype +carbody +poly(enginesize,i)
             +carheight , data = card)
    cv.error[i] = cv.glm(card, glm.fit)$delta[1]
}
cv.error
```

```
## [1] 11009074 11145974  8861496  8826774 12322018
```

Now, applying polynomials upto 5, we see that the average error decreses in the third and forth polymial about 8M on the test left data instance.

## k fold cross-validation

In k-fold we partition data in k sections and use k-1 as the training instance and test on the rest partition.

```
set.seed(17)

cv.error.10 = rep(0,10)

for (i in 1:10){
  glm.fit = glm(price~ poly(curbweight,i)+carwidth
                +peakrpm+horsepower+carlength+fueltype +carbody
             +poly(enginesize,i)+carheight , data = card)
    cv.error.10[i] = cv.glm(card, glm.fit, K = 10)$delta[1]
}
cv.error.10
```

```
##  [1]    10947388    11347044     9216694     9141835    12369026    18336440
##  [7]  1819380000  2173132692 17382271800  2972794081
```

In the previous result, we see error decreases initially with the model degree and again rises showing overfit and huge training error on test dataset. The high polynomial model suffers from high variance problem.

## Bootstrap

To implement boostrap we will use boot function.

```
alpha.fn=function(data,index){
  X=data$carlength[index]
  Y=data$price[index]
  return((var(Y)-cov(X,Y))/(var(X)+var(Y)-2*cov(X,Y)))
}

alpha.fn(card,1:100)
```

```
## [1] 1.001215
```

This provide the alpha value of 1.001215, now we select seed to recompute the alpha value for the car dataset. We take 100 samples in consideration.

```r
set.seed(1)
alpha.fn(card,sample(100,100,replace=T))
```

```
## [1] 1.001173
```

Recomputing we get the value of 1.001173, very similar to the earlier sampling dataset.

```r
boot(card,alpha.fn,R=1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = card, statistic = alpha.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original       bias     std. error
## t1* 1.001054 8.410011e-06 0.0001056349
```

The previous result shows the statistical distribution of alpha. The original value is about 1, with very low bias and standard deviation.

We will use the bootstrap model to analyse the performance of the linear model fit using the car dataset.

```r
boot.fn=function(data,index)
  return(coef(lm(price~ curbweight+carwidth+peakrpm+horsepower+
             +carlength+fueltype +carbody +enginesize+carheight
             ,data=card,subset=index)))

boot.fn(card, 1:203)
```

```
##      (Intercept)       curbweight         carwidth          peakrpm
##    -63530.972604         2.747903       638.131288         2.175832
##       horsepower        carlength      fueltypegas   carbodyhardtop
##        43.344359       -60.676505     -1972.137018     -2573.963804
## carbodyhatchback      carbodysedan      carbodywagon       enginesize
##     -5550.370868     -4605.758455     -6477.290774        89.381308
##        carheight
##       334.490382
```

```r
set.seed(1)

boot.fn(card,sample(205,205,replace=T))
```

```
##      (Intercept)       curbweight         carwidth          peakrpm
##    -5.437980e+04     1.323636e+00     5.404937e+02     9.925484e-01
##       horsepower        carlength      fueltypegas   carbodyhardtop
##     3.918690e+01    -3.556285e+01    -1.498810e+03    -3.917827e+03
## carbodyhatchback      carbodysedan      carbodywagon       enginesize
##    -9.299400e+03    -8.348023e+03    -1.171707e+04     1.031566e+02
##        carheight
##     4.167942e+02
```

```r
boot.fn(card,sample(205, 205,replace=T))
```

```
##      (Intercept)       curbweight         carwidth          peakrpm
##    -73593.750256         2.725820       978.588449         1.279732
##       horsepower        carlength      fueltypegas  carbodyhardtop
##        40.618748       -84.377984      -917.918748      -716.341183
## carbodyhatchback     carbodysedan     carbodywagon       enginesize
##     -5859.570439     -4577.366174     -6303.931150        81.165508
##        carheight
##       272.748726
```

```r
boot(Auto,boot.fn,100)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Auto, statistic = boot.fn, R = 100)
##
##
## Bootstrap Statistics :
##           original        bias     std. error
## t1*  -65287.278532 1218.37610669 1.307132e+04
## t2*       2.950661    0.08560465 1.340050e+00
## t3*     660.153937   14.77530690 2.554868e+02
## t4*       2.257717   -0.15537835 6.575903e-01
## t5*      41.718643   -1.07212993 1.271143e+01
## t6*     -62.625543   -1.60625643 4.727537e+01
## t7*   -2012.332996  195.83349433 9.875831e+02
## t8*   -2546.677919  -47.19188736 3.346254e+03
## t9*   -5555.412413   31.13727022 2.002547e+03
## t10*  -4573.041023  145.08996152 2.051623e+03
## t11*  -6553.308168   62.69901666 2.184819e+03
## t12*     88.441393   -2.60482640 1.930561e+01
## t13*    335.571161  -22.22436521 1.274739e+02
```

Here the above result gives the features value and their bias variances. We can see that carbody has the highest stardard deviation of 3346.

```r
summary(lm(price~carlength,data=card))$coef
```

```
##                Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) -63690.6716  5792.7934 -10.99481 2.313319e-22
## carlength      442.2161    33.1996  13.31992 1.678707e-29
```

```r
summary(lm(price~carlength+carwidth+peakrpm,data=card))$coef
```

```
##                  Estimate   Std. Error    t value     Pr(>|t|)
## (Intercept) -1.702890e+05 1.442212e+04 -11.807488 8.804821e-25
## carlength    1.219443e+02 5.487483e+01   2.222226 2.738293e-02
## carwidth     2.324874e+03 3.098843e+02   7.502395 1.980929e-12
## peakrpm      1.778318e+00 7.869361e-01   2.259799 2.490620e-02
```

```
boot.fn=function(data,index)
  coefficients(lm(price~carwidth+I(carwidth^2),data=card,subset=index))
set.seed(1)
boot(card,boot.fn,1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = card, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##        original        bias     std. error
## t1* 671322.7695 -15668.054071 284011.07836
## t2* -22459.6189    471.898286   8633.88946
## t3*    189.0844     -3.549942     65.57634
```

```
summary(lm(price~carwidth+I(carwidth^2),data=card))$coef
```

```
##                  Estimate   Std. Error   t value    Pr(>|t|)
## (Intercept)   671322.7695 254902.47186  2.633646 0.009101236
## carwidth      -22459.6189   7628.20546 -2.944286 0.003616364
## I(carwidth^2)    189.0844     57.02578  3.315771 0.001083457
```

```
boot.fn=function(data,index)
  coefficients(lm(price~carlength+I(carlength^2),data=card,subset=index))
set.seed(1)
boot(card,boot.fn,1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = card, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##          original        bias     std. error
## t1* 185969.293744 -1.952034e+03 60313.037599
## t2*  -2425.276416  2.422859e+01   719.556222
## t3*      8.192746 -7.496051e-02     2.136189
```

```
summary(lm(price~carlength+I(carlength^2),data=card))$coef
```

```
##                       Estimate  Std. Error   t value     Pr(>|t|)
## (Intercept)    185969.293744 57979.22038  3.207516 1.556900e-03
## carlength       -2425.276416   663.62587 -3.654584 3.283737e-04
## I(carlength^2)      8.192746     1.89387  4.325929 2.386447e-05
```

In ealier result we see the comparison of applying the feature and the square of the features. The carlength both the parameter and square have similar bootshraping error.

# Chapter 6

In this section, the model selection methods like best subset selection and dimentionality reduction techniques are applied on the car dataset.

## Best Subset Selection

Firstly we remove the missing data instances from the dataset.

```r
### Lab 1 best subset selection
dim(card)
```

```
## [1] 205  27
```

```r
card =  na.omit(card)

dim(card)
```

```
## [1] 205  27
```

```r
sum(is.na(card))
```

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

From the sum result of 0 we know that there are no missing data points in the datset instances.

```r
## Choosing the best feature set by BIC, Cp , AIC ...
library(leaps)
attach(card)
```

```
## The following object is masked _by_ .GlobalEnv:
##
##     high
```

```
## The following objects are masked from card (pos = 7):
##
##     aspiration, boreratio, car_ID, carbody, carheight, carlength,
##     CarName, carwidth, citympg, compressionratio, curbweight,
##     cylindernumber, doornumber, drivewheel, enginelocation, enginesize,
##     enginetype, fuelsystem, fueltype, highwaympg, horsepower, peakrpm,
##     price, stroke, symboling, wheelbase
```

```
## The following objects are masked from card (pos = 8):
##
##     aspiration, boreratio, car_ID, carbody, carheight, carlength,
##     CarName, carwidth, citympg, compressionratio, curbweight,
##     cylindernumber, doornumber, drivewheel, enginelocation, enginesize,
##     enginetype, fuelsystem, fueltype, highwaympg, horsepower, peakrpm,
##     price, stroke, symboling, wheelbase
## The following objects are masked from card (pos = 10):
##
##     aspiration, boreratio, car_ID, carbody, carheight, carlength,
##     CarName, carwidth, citympg, compressionratio, curbweight,
##     cylindernumber, doornumber, drivewheel, enginelocation, enginesize,
##     enginetype, fuelsystem, fueltype, highwaympg, horsepower, peakrpm,
##     price, stroke, symboling, wheelbase
```

```
regfit.full =  regsubsets(price~fuelsystem+peakrpm+citympg
                          + enginesize+enginetype+carwidth+curbweight+carlength
                          + highwaympg+ boreratio+ stroke + wheelbase + drivewheel
                          + enginelocation+ aspiration+ doornumber
                          + horsepower+ compressionratio,
                          data = card)

summary(regfit.full)
```

```
## Subset selection object
## Call: regsubsets.formula(price ~ fuelsystem + peakrpm + citympg + enginesize +
##     enginetype + carwidth + curbweight + carlength + highwaympg +
##     boreratio + stroke + wheelbase + drivewheel + enginelocation +
##     aspiration + doornumber + horsepower + compressionratio,
##     data = card)
## 30 Variables  (and intercept)
##                      Forced in Forced out
## fuelsystem2bbl           FALSE      FALSE
## fuelsystem4bbl           FALSE      FALSE
## fuelsystemidi            FALSE      FALSE
## fuelsystemmfi            FALSE      FALSE
## fuelsystemmpfi           FALSE      FALSE
## fuelsystemspdi           FALSE      FALSE
## fuelsystemspfi           FALSE      FALSE
## peakrpm                  FALSE      FALSE
## citympg                  FALSE      FALSE
## enginesize               FALSE      FALSE
## enginetypedohcv          FALSE      FALSE
## enginetypel              FALSE      FALSE
## enginetypeohc            FALSE      FALSE
## enginetypeohcf           FALSE      FALSE
## enginetypeohcv           FALSE      FALSE
## enginetyperotor          FALSE      FALSE
## carwidth                 FALSE      FALSE
## curbweight               FALSE      FALSE
## carlength                FALSE      FALSE
## highwaympg               FALSE      FALSE
## boreratio                FALSE      FALSE
## stroke                   FALSE      FALSE
## wheelbase                FALSE      FALSE
## drivewheelfwd            FALSE      FALSE
## drivewheelrwd            FALSE      FALSE
## enginelocationrear       FALSE      FALSE
## aspirationturbo          FALSE      FALSE
## doornumbertwo            FALSE      FALSE
## horsepower               FALSE      FALSE
## compressionratio       FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          fuelsystem2bbl fuelsystem4bbl fuelsystemidi fuelsystemmfi
## 1  ( 1 ) " "            " "            " "           " "
## 2  ( 1 ) " "            " "            " "           " "
## 3  ( 1 ) " "            " "            " "           " "
## 4  ( 1 ) " "            " "            " "           " "
## 5  ( 1 ) " "            " "            " "           " "
## 6  ( 1 ) " "            " "            " "           " "
## 7  ( 1 ) " "            " "            " "           " "
## 8  ( 1 ) " "            " "            " "           " "
##          fuelsystemmpfi fuelsystemspdi fuelsystemspfi peakrpm citympg
## 1  ( 1 ) " "            " "            " "            " "     " "
## 2  ( 1 ) " "            " "            " "            " "     " "
## 3  ( 1 ) " "            " "            " "            " "     " "
## 4  ( 1 ) " "            " "            " "            " "     " "
## 5  ( 1 ) " "            " "            " "            "*"     " "
## 6  ( 1 ) " "            " "            " "            " "     " "
## 7  ( 1 ) " "            " "            " "            " "     " "
## 8  ( 1 ) " "            " "            " "            " "     " "
##          enginesize enginetypedohcv enginetypel enginetypeohc enginetypeohcf
## 1  ( 1 ) "*"        " "             " "         " "           " "
## 2  ( 1 ) "*"        " "             " "         " "           " "
## 3  ( 1 ) "*"        " "             " "         " "           " "
## 4  ( 1 ) "*"        " "             " "         " "           " "
## 5  ( 1 ) "*"        " "             " "         " "           " "
## 6  ( 1 ) "*"        " "             " "         "*"           " "
## 7  ( 1 ) "*"        " "             " "         " "           "*"
## 8  ( 1 ) "*"        " "             " "         " "           " "
##          enginetypeohcv enginetyperotor carwidth curbweight carlength
## 1  ( 1 ) " "            " "             " "      " "        " "
## 2  ( 1 ) " "            " "             " "      " "        " "
## 3  ( 1 ) " "            " "             "*"      " "        " "
## 4  ( 1 ) " "            " "             "*"      " "        " "
## 5  ( 1 ) " "            " "             "*"      " "        " "
## 6  ( 1 ) " "            "*"             "*"      " "        " "
## 7  ( 1 ) "*"            "*"             "*"      " "        " "
## 8  ( 1 ) "*"            "*"             "*"      " "        " "
##          highwaympg boreratio stroke wheelbase drivewheelfwd drivewheelrwd
## 1  ( 1 ) " "        " "       " "    " "       " "           " "
## 2  ( 1 ) " "        " "       " "    " "       " "           "*"
## 3  ( 1 ) " "        " "       " "    " "       " "           " "
## 4  ( 1 ) " "        " "       " "    " "       "*"           " "
## 5  ( 1 ) " "        " "       " "    " "       "*"           " "
## 6  ( 1 ) " "        " "       "*"    " "       " "           " "
## 7  ( 1 ) " "        " "       "*"    " "       " "           " "
## 8  ( 1 ) " "        "*"       "*"    " "       "*"           " "
##          enginelocationrear aspirationturbo doornumbertwo horsepower
## 1  ( 1 ) " "                " "             " "           " "
## 2  ( 1 ) " "                " "             " "           " "
## 3  ( 1 ) "*"                " "             " "           " "
## 4  ( 1 ) "*"                " "             " "           " "
## 5  ( 1 ) "*"                " "             " "           " "
## 6  ( 1 ) "*"                " "             " "            " "
## 7  ( 1 ) "*"                " "             " "            " "
## 8  ( 1 ) "*"                " "             " "            " "
##          compressionratio
## 1  ( 1 ) " "
## 2  ( 1 ) " "
## 3  ( 1 ) " "
## 4  ( 1 ) " "
## 5  ( 1 ) " "
## 6  ( 1 ) " "
## 7  ( 1 ) " "
## 8  ( 1 ) " "
```

In above the model selected the best model based on the Residual sum of squared error. The * locations in the model shows that the best model takes the feaure of engine size and the second model considers the enginesize and drivewheel position. This experiment showed top 8 models we can extend that by providing nvmax parameters as follows.

```r
regfit.full = regsubsets(price~fuelsystem+peakrpm+citympg
                         + enginesize+enginetype+carwidth+curbweight+carlength
                         + highwaympg+ boreratio+ stroke + wheelbase + drivewheel
                         + enginelocation+ aspiration+ doornumber+ horsepower+ compressionratio,
                         data = card, nvmax = 19)

summary(regfit.full)
```

```
## Subset selection object
## Call: regsubsets.formula(price ~ fuelsystem + peakrpm + citympg + enginesize +
##     enginetype + carwidth + curbweight + carlength + highwaympg +
##     boreratio + stroke + wheelbase + drivewheel + enginelocation +
##     aspiration + doornumber + horsepower + compressionratio,
##     data = card, nvmax = 19)
## 30 Variables  (and intercept)
##                      Forced in Forced out
## fuelsystem2bbl           FALSE      FALSE
## fuelsystem4bbl           FALSE      FALSE
## fuelsystemidi            FALSE      FALSE
## fuelsystemmfi            FALSE      FALSE
## fuelsystemmpfi           FALSE      FALSE
## fuelsystemspdi           FALSE      FALSE
## fuelsystemspfi           FALSE      FALSE
## peakrpm                  FALSE      FALSE
## citympg                  FALSE      FALSE
## enginesize               FALSE      FALSE
## enginetypedohcv          FALSE      FALSE
## enginetypel              FALSE      FALSE
## enginetypeohc            FALSE      FALSE
## enginetypeohcf           FALSE      FALSE
## enginetypeohcv           FALSE      FALSE
## enginetyperotor          FALSE      FALSE
## carwidth                 FALSE      FALSE
## curbweight               FALSE      FALSE
## carlength                FALSE      FALSE
## highwaympg             FALSE      FALSE
## boreratio              FALSE      FALSE
## stroke                 FALSE      FALSE
## wheelbase              FALSE      FALSE
## drivewheelfwd          FALSE      FALSE
## drivewheelrwd          FALSE      FALSE
## enginelocationrear     FALSE      FALSE
## aspirationturbo        FALSE      FALSE
## doornumbertwo          FALSE      FALSE
## horsepower             FALSE      FALSE
## compressionratio       FALSE      FALSE
## 1 subsets of each size up to 19
## Selection Algorithm: exhaustive
##           fuelsystem2bbl fuelsystem4bbl fuelsystemidi fuelsystemmfi
## 1  ( 1 )  " "            " "            " "           " "
## 2  ( 1 )  " "            " "            " "           " "
## 3  ( 1 )  " "            " "            " "           " "
## 4  ( 1 )  " "            " "            " "           " "
## 5  ( 1 )  " "            " "            " "           " "
## 6  ( 1 )  " "            " "            " "           " "
## 7  ( 1 )  " "            " "            " "           " "
## 8  ( 1 )  " "            " "            " "           " "
## 9  ( 1 )  " "            " "            " "           " "
## 10  ( 1 ) " "            " "            " "           " "
## 11  ( 1 ) " "            " "            " "           " "
## 12  ( 1 ) " "            " "            "*"           " "
## 13  ( 1 ) " "            " "            "*"           " "
## 14  ( 1 ) " "            " "            "*"           " "
## 15  ( 1 ) " "            " "            "*"           "*"
## 16  ( 1 ) " "            " "            "*"           " "
## 17  ( 1 ) " "            " "            "*"           " "
## 18  ( 1 ) " "            " "            "*"           "*"
## 19  ( 1 ) " "            " "            "*"           "*"
##           fuelsystemmpfi fuelsystemspdi fuelsystemspfi peakrpm citympg
## 1  ( 1 )  " "            " "            " "            " "     " "
## 2  ( 1 )  " "            " "            " "            " "     " "
## 3  ( 1 )  " "            " "            " "            " "     " "
## 4  ( 1 )  " "            " "            " "            " "     " "
## 5  ( 1 )  " "            " "            " "            "*"     " "
## 6  ( 1 )  " "            " "            " "            " "     " "
## 7  ( 1 )  " "            " "            " "            " "     " "
## 8  ( 1 )  " "            " "            " "            " "     " "
## 9  ( 1 )  " "            " "            " "            " "     " "
## 10  ( 1 ) " "            " "            " "            "*"     " "
## 11  ( 1 ) " "            " "            " "            "*"     " "
## 12  ( 1 ) " "            " "            " "            "*"     " "
## 13  ( 1 ) " "            "*"            " "            "*"     " "
## 14  ( 1 ) " "            "*"            " "            "*"     " "
## 15  ( 1 ) " "            "*"            " "            "*"     " "
## 16  ( 1 ) " "            "*"            " "            "*"     " "
## 17  ( 1 ) " "            "*"            " "            "*"     " "
## 18  ( 1 ) " "            "*"            " "            "*"     " "
## 19  ( 1 ) "*"            "*"            " "            "*"     " "
##           enginesize enginetypedohcv enginetypel enginetypeohc enginetypeohcf
## 1  ( 1 )  "*"        " "             " "         " "           " "
## 2  ( 1 )  "*"        " "             " "         " "           " "
## 3  ( 1 )  "*"        " "             " "         " "           " "
## 4  ( 1 )  "*"        " "             " "         " "           " "
## 5  ( 1 )  "*"        " "             " "         " "           " "
## 6  ( 1 )  "*"        " "             " "         "*"           " "
## 7  ( 1 )  "*"        " "             " "         " "           "*"
## 8  ( 1 )  "*"        " "             " "         " "           " "
## 9  ( 1 )  "*"        " "             " "         "*"           " "
## 10  ( 1 ) "*"        " "             " "         "*"           " "
## 11  ( 1 ) "*"        " "             " "         "*"           " "
## 12  ( 1 ) "*"        " "             " "         "*"           " "
## 13  ( 1 ) "*"        " "             " "         "*"           " "
## 14  ( 1 ) "*"        " "             " "         "*"           " "
## 15  ( 1 ) "*"        " "             " "         "*"           " "
## 16  ( 1 ) "*"        " "             "*"         "*"           " "
## 17  ( 1 ) "*"        " "             "*"         "*"           " "
## 18  ( 1 ) "*"        " "             "*"         "*"           " "
## 19  ( 1 ) "*"        " "             "*"         "*"           " "
##           enginetypeohcv enginetyperotor carwidth curbweight carlength
## 1  ( 1 )  " "            " "             " "      " "        " "
## 2  ( 1 )  " "            " "             " "      " "        " "
## 3  ( 1 )  " "            " "             "*"      " "        " "
## 4  ( 1 )  " "            " "             "*"      " "        " "
## 5  ( 1 )  " "            " "             "*"      " "        " "
## 6  ( 1 )  " "            "*"             "*"      " "        " "
## 7  ( 1 )  "*"            "*"             "*"      " "        " "
## 8  ( 1 )  "*"            "*"             "*"      " "        " "
## 9  ( 1 )  "*"            "*"             "*"      " "        " "
## 10  ( 1 ) "*"            "*"             "*"      " "        " "
## 11  ( 1 ) "*"            "*"             "*"      " "        " "
## 12  ( 1 ) "*"            "*"             "*"      " "        " "
## 13  ( 1 ) "*"            "*"             "*"      " "        " "
## 14  ( 1 ) "*"            "*"             "*"      " "        " "
## 15  ( 1 ) "*"            "*"             "*"      " "        " "
## 16  ( 1 ) "*"            "*"             "*"      "*"        " "
## 17  ( 1 ) "*"            "*"             "*"      "*"        " "
## 18  ( 1 ) "*"            "*"             "*"      "*"        " "
## 19  ( 1 ) "*"            "*"             "*"      "*"        " "
##           highwaympg boreratio stroke wheelbase drivewheelfwd drivewheelrwd
## 1  ( 1 )  " "        " "       " "    " "       " "           " "
## 2  ( 1 )  " "        " "       " "    " "       " "           "*"
## 3  ( 1 )  " "        " "       " "    " "       " "           " "
## 4  ( 1 )  " "        " "       " "    " "       "*"           " "
## 5  ( 1 )  " "        " "       " "    " "       "*"           " "
## 6  ( 1 )  " "        " "       "*"    " "       " "           " "
## 7  ( 1 )  " "        " "       "*"    " "       " "           " "
## 8  ( 1 )  " "        "*"       "*"    " "       "*"           " "
## 9  ( 1 )  " "        "*"       "*"    " "       "*"           " "
## 10  ( 1 ) " "        "*"       "*"    " "       "*"           " "
## 11  ( 1 ) " "        "*"       "*"    " "       "*"           " "
## 12  ( 1 ) " "        "*"       "*"    " "       "*"           " "
## 13  ( 1 ) " "        "*"       "*"    " "       " "           "*"
## 14  ( 1 ) " "        "*"       "*"    " "       " "           "*"
## 15  ( 1 ) " "        "*"       "*"     " "       " "            "*"
## 16  ( 1 ) "*"        "*"       "*"     " "       " "            "*"
## 17  ( 1 ) "*"        "*"       "*"     " "       " "            "*"
## 18  ( 1 ) "*"        "*"       "*"     " "       " "            "*"
## 19  ( 1 ) "*"        "*"       "*"     " "       " "            "*"
##           enginelocationrear aspirationturbo doornumbertwo horsepower
## 1  ( 1 )  " "                " "             " "           " "
## 2  ( 1 )  " "                " "             " "           " "
## 3  ( 1 )  "*"                " "             " "           " "
## 4  ( 1 )  "*"                " "             " "           " "
## 5  ( 1 )  "*"                " "             " "           " "
## 6  ( 1 )  "*"                " "             " "           " "
## 7  ( 1 )  "*"                " "             " "           " "
## 8  ( 1 )  "*"                " "             " "           " "
## 9  ( 1 )  "*"                " "             " "           " "
## 10  ( 1 ) "*"                " "             " "           " "
## 11  ( 1 ) "*"                "*"             " "           " "
## 12  ( 1 ) "*"                " "             " "           " "
## 13  ( 1 ) "*"                " "             " "           " "
## 14  ( 1 ) "*"                "*"             " "           " "
## 15  ( 1 ) "*"                "*"             " "           " "
## 16  ( 1 ) "*"                " "             " "           " "
## 17  ( 1 ) "*"                "*"             " "           " "
## 18  ( 1 ) "*"                "*"             " "           " "
## 19  ( 1 ) "*"                "*"             " "           " "
##           compressionratio
## 1  ( 1 )  " "
## 2  ( 1 )  " "
## 3  ( 1 )  " "
## 4  ( 1 )  " "
## 5  ( 1 )  " "
## 6  ( 1 )  " "
## 7  ( 1 )  " "
## 8  ( 1 )  " "
## 9  ( 1 )  " "
## 10  ( 1 ) " "
## 11  ( 1 ) " "
## 12  ( 1 ) "*"
## 13  ( 1 ) "*"
## 14  ( 1 ) "*"
## 15  ( 1 ) "*"
## 16  ( 1 ) "*"
## 17  ( 1 ) "*"
## 18  ( 1 ) "*"
## 19  ( 1 ) "*"
```

```
reg.summary =  summary(regfit.full)
names(reg.summary)
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

```
reg.summary$rsq
```

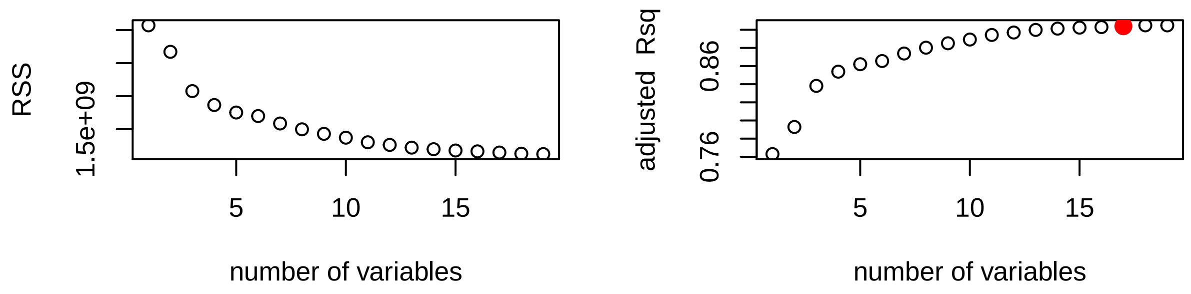
```
##  [1] 0.7641291 0.7948774 0.8405232 0.8567114 0.8654310 0.8695393 0.8781805
##  [8] 0.8849685 0.8902002 0.8946820 0.9000161 0.9030713 0.9062538 0.9080701
## [15] 0.9095691 0.9105845 0.9119274 0.9132864 0.9137280
```

Now the result shows top 17 models with the r squared values of the model with different top features.

```
par(mfrow = c(2,2))

plot(reg.summary$rss, xlab= "number of variables", ylab = "RSS")

plot(reg.summary$adjr2, xlab= "number of variables", ylab = "adjusted  Rsq")
which.max(reg.summary$adjr2) # return 17
```

```
## [1] 18
```

```
points(17, reg.summary$adjr2[17], col ="red", cex = 2, pch =20)
```



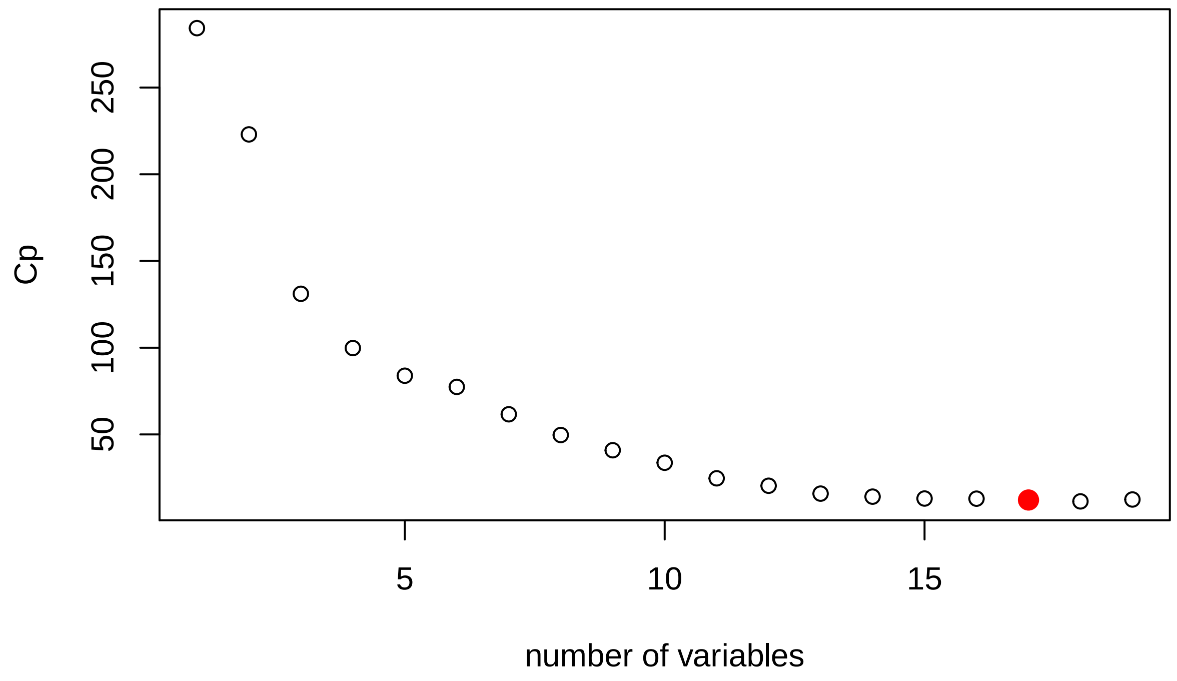
The plot shows the model error decrese with increasing variable numbers.

The previous value retuns 17. We will use this to plot the cp and BIC statistics.

```
plot(reg.summary$cp, xlab= "number of variables", ylab = "Cp")
which.min(reg.summary$cp) #18
```

```
## [1] 18
```

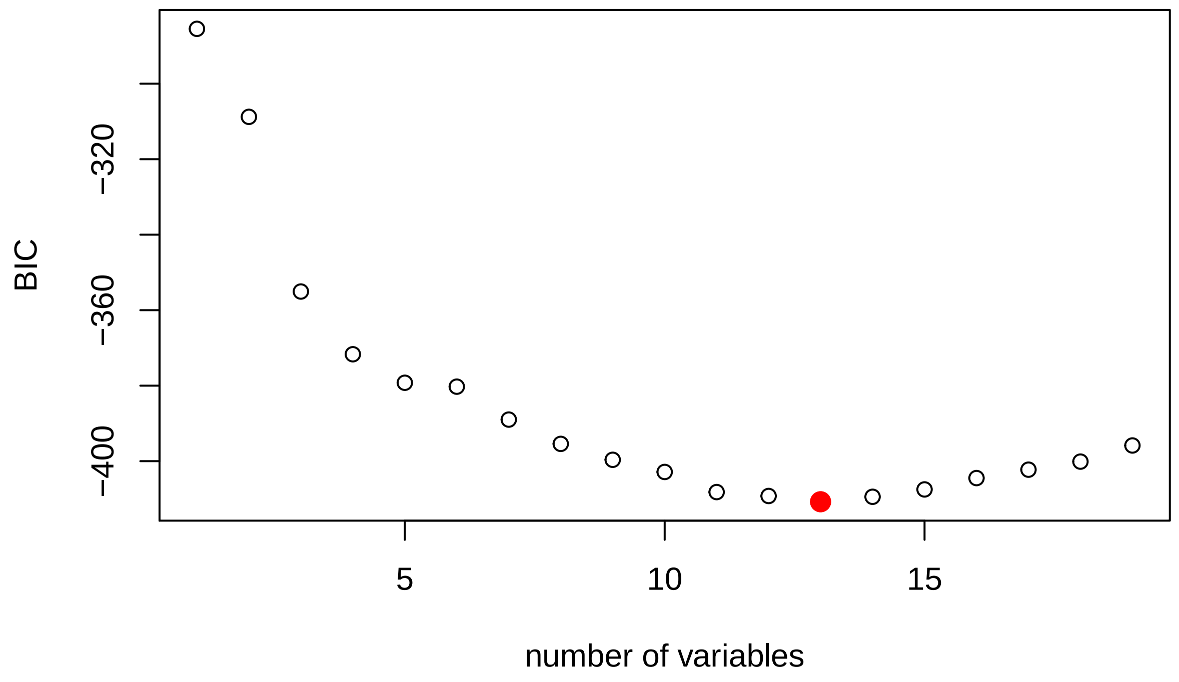
```r
points(17, reg.summary$cp[17], col ="red", cex = 2, pch =20)
```



```r
which.min(reg.summary$bic) #13
```

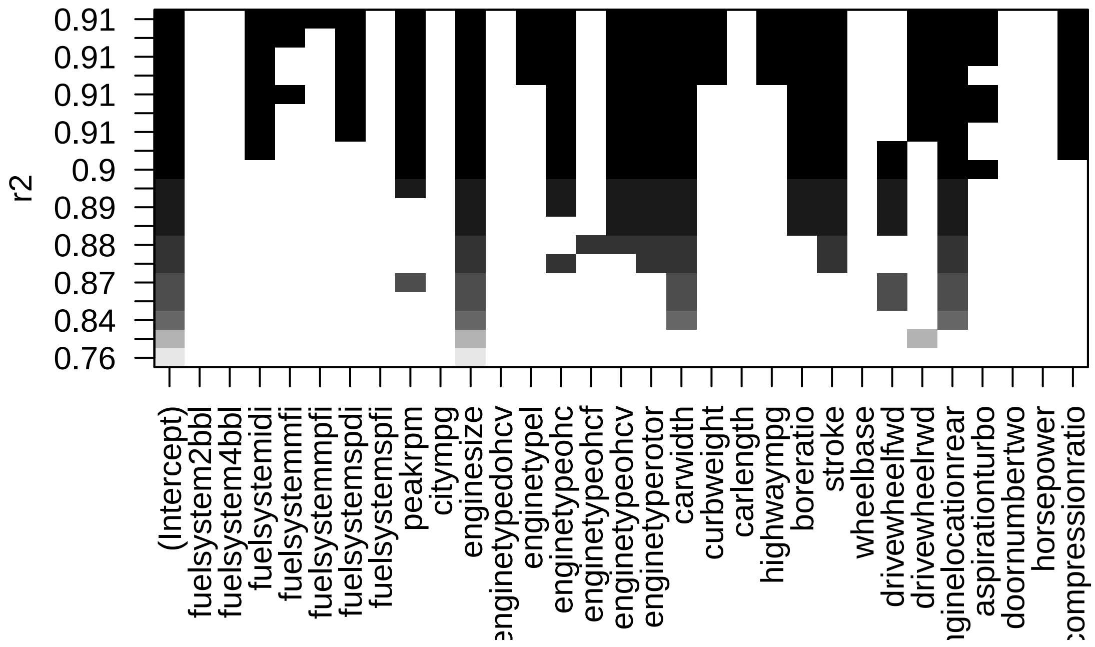
```
## [1] 13
```

```r
plot(reg.summary$bic, xlab= "number of variables", ylab = "BIC")
points(13, reg.summary$bic[13], col ="red", cex = 2, pch =20)
```

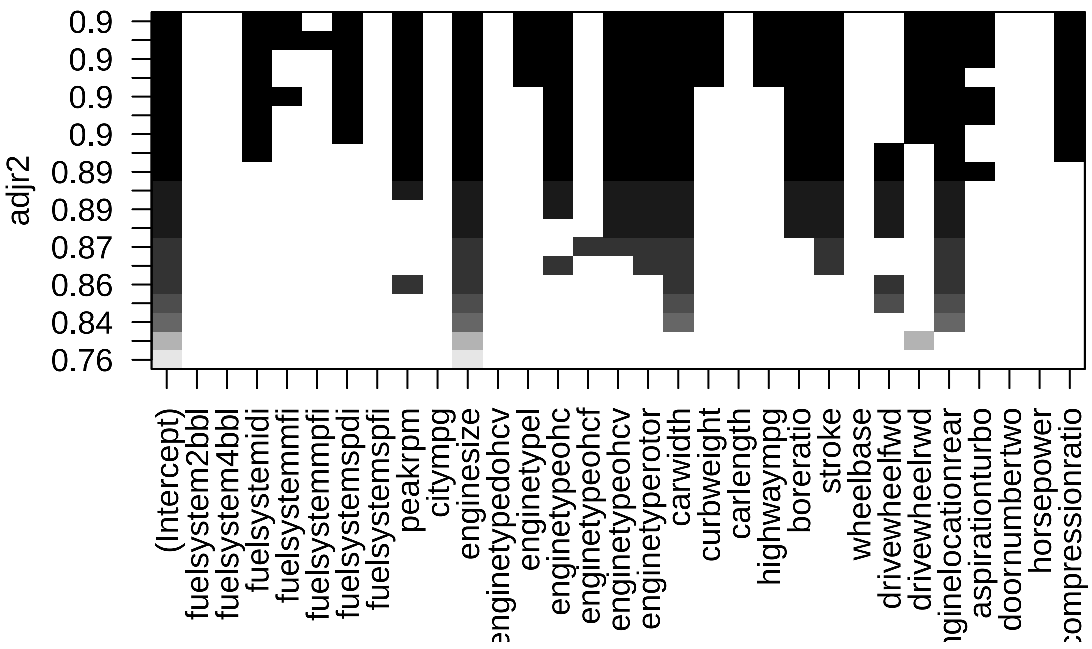


The Cp value decrease with new features, now we do the same for the BIC criterion for too in the car dataset. We observe from the above model that BIC selected the 13 feature model as the best candidate while Cp selected 17 feature/predictor variables models.

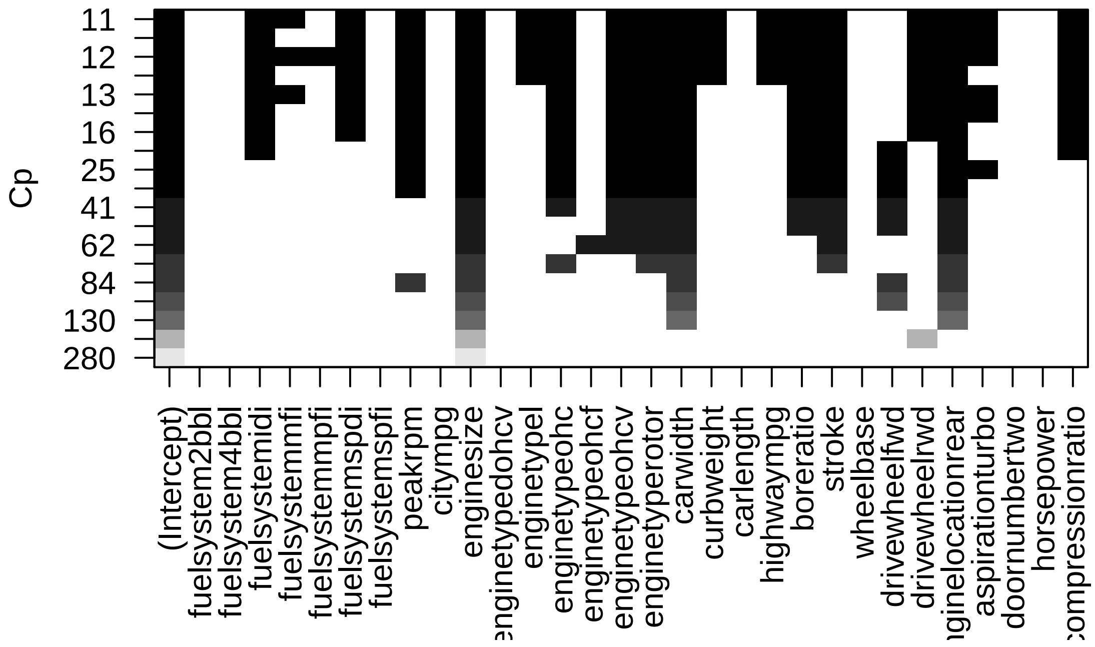
```
plot(regfit.full, scale = "r2")
```



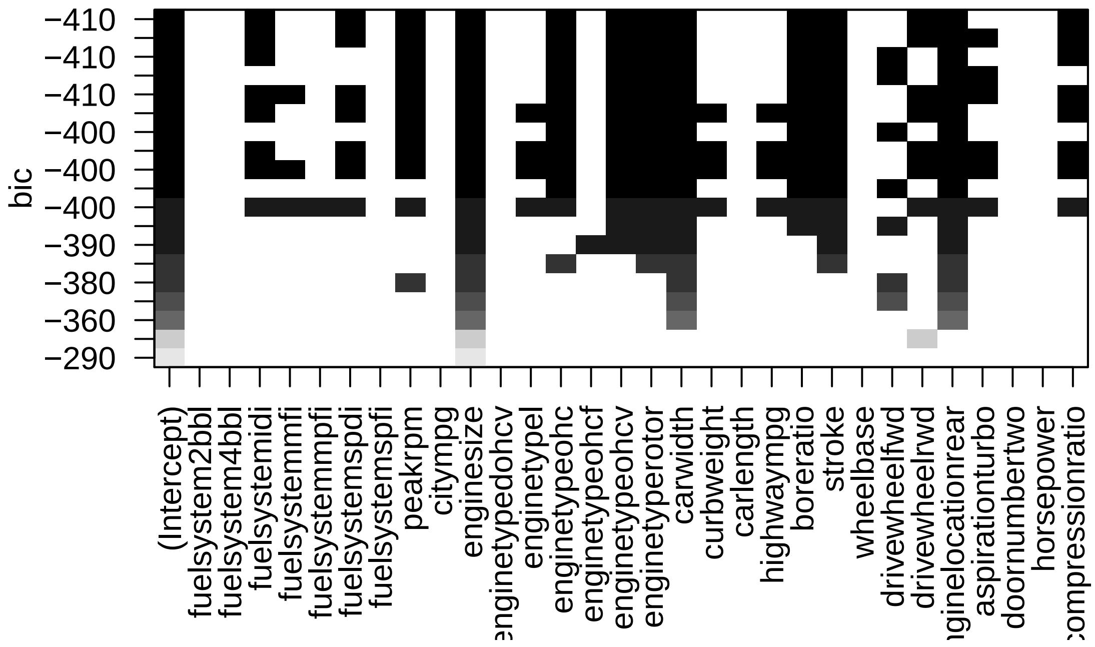
```r
plot(regfit.full, scale = "adjr2")
```



```r
plot(regfit.full, scale = "Cp")
```



```r
plot(regfit.full, scale = "bic")
```



```r
coef(regfit.full, 7)
```

```
##        (Intercept)          enginesize      enginetypeohcf      enginetypeohcv
##        -57712.5552            148.4221          -3804.7916          -4852.3248
##    enginetyperotor            carwidth              stroke enginelocationrear
##          7800.8444           1007.4850          -4307.4211          14256.4615
```

Here the above plot show different model with different criterion BIC, Cp as they select different feature set.

## Foward and Backward stepwise selection

We use the parameter method to select backward or forward selection

In this method, the model start with smallest variables and add new variable in the next iteration.

```r
regfit.fwd  = regsubsets(price~fuelsystem+peakrpm+citympg
                         + enginesize+enginetype+carwidth+curbweight+carlength
                         + highwaympg+ boreratio+ stroke + wheelbase + drivewheel
                         + enginelocation+ aspiration+ doornumber+ horsepower+ compressionratio,
                         data= card, nvmax =8, method = "forward")

summary(regfit.fwd)
```

```
## Subset selection object
## Call: regsubsets.formula(price ~ fuelsystem + peakrpm + citympg + enginesize +
##     enginetype + carwidth + curbweight + carlength + highwaympg +
##     boreratio + stroke + wheelbase + drivewheel + enginelocation +
##     aspiration + doornumber + horsepower + compressionratio,
##     data = card, nvmax = 8, method = "forward")
## 30 Variables  (and intercept)
##                    Forced in Forced out
## fuelsystem2bbl         FALSE      FALSE
## fuelsystem4bbl         FALSE      FALSE
## fuelsystemidi          FALSE      FALSE
## fuelsystemmfi          FALSE      FALSE
## fuelsystemmpfi         FALSE      FALSE
## fuelsystemspdi         FALSE      FALSE
## fuelsystemspfi         FALSE      FALSE
## peakrpm                FALSE      FALSE
## citympg                FALSE      FALSE
## enginesize             FALSE      FALSE
## enginetypedohcv        FALSE      FALSE
## enginetypel            FALSE      FALSE
## enginetypeohc          FALSE      FALSE
## enginetypeohcf         FALSE      FALSE
## enginetypeohcv         FALSE      FALSE
## enginetyperotor        FALSE      FALSE
## carwidth               FALSE      FALSE
## curbweight             FALSE      FALSE
## carlength              FALSE      FALSE
## highwaympg             FALSE      FALSE
## boreratio              FALSE      FALSE
## stroke                 FALSE      FALSE
## wheelbase              FALSE      FALSE
## drivewheelfwd          FALSE      FALSE
## drivewheelrwd          FALSE      FALSE
## enginelocationrear     FALSE      FALSE
## aspirationturbo        FALSE      FALSE
## doornumbertwo          FALSE      FALSE
## horsepower             FALSE      FALSE
## compressionratio       FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: forward
##          fuelsystem2bbl fuelsystem4bbl fuelsystemidi fuelsystemmfi
## 1  ( 1 ) " "            " "            " "           " "
## 2  ( 1 ) " "            " "            " "           " "
## 3  ( 1 ) " "            " "            " "           " "
## 4  ( 1 ) " "            " "            " "           " "
## 5  ( 1 ) " "            " "            " "           " "
## 6  ( 1 ) " "            " "            " "           " "
## 7  ( 1 ) " "            " "            " "           " "
## 8  ( 1 ) " "            " "            " "           " "
##          fuelsystemmpfi fuelsystemspdi fuelsystemspfi peakrpm citympg
## 1  ( 1 ) " "            " "            " "            " "     " "
## 2  ( 1 ) " "            " "            " "            " "     " "
## 3  ( 1 ) " "            " "            " "            " "     " "
## 4  ( 1 ) " "            " "            " "            " "     " "
## 5  ( 1 ) " "            " "            " "            "*"     " "
## 6  ( 1 ) " "            " "            " "            "*"      " "
## 7  ( 1 ) " "            " "            " "            "*"      " "
## 8  ( 1 ) " "            " "            " "            "*"      " "
##          enginesize enginetypedohcv enginetypel enginetypeohc enginetypeohcf
## 1  ( 1 ) "*"        " "             " "         " "           " "
## 2  ( 1 ) "*"        " "             " "         " "           " "
## 3  ( 1 ) "*"        " "             " "         " "           " "
## 4  ( 1 ) "*"        " "             " "         " "           " "
## 5  ( 1 ) "*"        " "             " "         " "           " "
## 6  ( 1 ) "*"        " "             " "         " "           " "
## 7  ( 1 ) "*"        " "             " "         " "           " "
## 8  ( 1 ) "*"        " "             " "         " "           " "
##          enginetypeohcv enginetyperotor carwidth curbweight carlength
## 1  ( 1 ) " "            " "             " "      " "        " "
## 2  ( 1 ) " "            " "             " "      " "        " "
## 3  ( 1 ) " "            " "             " "      " "        " "
## 4  ( 1 ) " "            " "             "*"      " "        " "
## 5  ( 1 ) " "            " "             "*"      " "        " "
## 6  ( 1 ) "*"            " "             "*"      " "        " "
## 7  ( 1 ) "*"            " "             "*"      " "        " "
## 8  ( 1 ) "*"            " "             "*"      " "        " "
##          highwaympg boreratio stroke wheelbase drivewheelfwd drivewheelrwd
## 1  ( 1 ) " "        " "       " "    " "       " "           " "
## 2  ( 1 ) " "        " "       " "    " "       " "           "*"
## 3  ( 1 ) " "        " "       " "    " "       " "           "*"
## 4  ( 1 ) " "        " "       " "    " "       " "           "*"
## 5  ( 1 ) " "        " "       " "    " "       " "           "*"
## 6  ( 1 ) " "        " "       " "    " "       " "           "*"
## 7  ( 1 ) " "        " "       "*"    " "       " "           "*"
## 8  ( 1 ) " "        "*"       "*"    " "       " "           "*"
##          enginelocationrear aspirationturbo doornumbertwo horsepower
## 1  ( 1 ) " "                " "             " "           " "
## 2  ( 1 ) " "                " "             " "           " "
## 3  ( 1 ) "*"                " "             " "           " "
## 4  ( 1 ) "*"                " "             " "           " "
## 5  ( 1 ) "*"                " "             " "           " "
## 6  ( 1 ) "*"                " "             " "           " "
## 7  ( 1 ) "*"                " "             " "           " "
## 8  ( 1 ) "*"                " "             " "           " "
##          compressionratio
## 1  ( 1 ) " "
## 2  ( 1 ) " "
## 3  ( 1 ) " "
## 4  ( 1 ) " "
## 5  ( 1 ) " "
## 6  ( 1 ) " "
## 7  ( 1 ) " "
## 8  ( 1 ) " "
```

The forward selection method selects enginesize at first and then drivewheel and never drops the fearures. While in earlier method, in subset selection based on BIC/Cp some features were dropped later on. The backward traces in the reverse way it started with all the variables. In backward selection once the feature is dropped it is not recovered later.

We see the following result of backward in the following results

```r
regfit.bwd  = regsubsets(price~fuelsystem+peakrpm+citympg
                         + enginesize+enginetype+carwidth+curbweight+carlength
                         + highwaympg+ boreratio+ stroke + wheelbase + drivewheel
                         + enginelocation+ aspiration+ doornumber+ horsepower+ compressionratio,
                         data= card, nvmax =8, method = "backward")

summary(regfit.bwd)
```

```
## Subset selection object
## Call: regsubsets.formula(price ~ fuelsystem + peakrpm + citympg + enginesize +
##     enginetype + carwidth + curbweight + carlength + highwaympg +
##     boreratio + stroke + wheelbase + drivewheel + enginelocation +
##     aspiration + doornumber + horsepower + compressionratio,
##     data = card, nvmax = 8, method = "backward")
## 30 Variables  (and intercept)
##                      Forced in Forced out
## fuelsystem2bbl           FALSE      FALSE
## fuelsystem4bbl           FALSE      FALSE
## fuelsystemidi            FALSE      FALSE
## fuelsystemmfi            FALSE      FALSE
## fuelsystemmpfi           FALSE      FALSE
## fuelsystemspdi           FALSE      FALSE
## fuelsystemspfi           FALSE      FALSE
## peakrpm                  FALSE      FALSE
## citympg                  FALSE      FALSE
## enginesize               FALSE      FALSE
## enginetypedohcv          FALSE      FALSE
## enginetypel              FALSE      FALSE
## enginetypeohc            FALSE      FALSE
## enginetypeohcf           FALSE      FALSE
## enginetypeohcv           FALSE      FALSE
## enginetyperotor          FALSE      FALSE
## carwidth                 FALSE      FALSE
## curbweight               FALSE      FALSE
## carlength                FALSE      FALSE
## highwaympg               FALSE      FALSE
## boreratio                FALSE      FALSE
## stroke                   FALSE      FALSE
## wheelbase                FALSE      FALSE
## drivewheelfwd            FALSE      FALSE
## drivewheelrwd            FALSE      FALSE
## enginelocationrear       FALSE      FALSE
## aspirationturbo          FALSE      FALSE
## doornumbertwo            FALSE      FALSE
## horsepower               FALSE      FALSE
## compressionratio         FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: backward
##          fuelsystem2bbl fuelsystem4bbl fuelsystemidi fuelsystemmfi
## 1  ( 1 ) " "            " "            " "           " "
## 2  ( 1 ) " "            " "            " "           " "
## 3  ( 1 ) " "            " "            " "           " "
## 4  ( 1 ) " "            " "            " "           " "
## 5  ( 1 ) " "            " "            " "           " "
## 6  ( 1 ) " "            " "            " "           " "
## 7  ( 1 ) " "            " "            " "           " "
## 8  ( 1 ) " "            " "            " "           " "
##          fuelsystemmpfi fuelsystemspdi fuelsystemspfi peakrpm citympg
## 1  ( 1 ) " "            " "            " "            " "     " "
## 2  ( 1 ) " "            " "            " "            " "     " "
## 3  ( 1 ) " "            " "            " "            " "     " "
## 4  ( 1 ) " "            " "            " "            " "     " "
## 5  ( 1 ) " "            " "            " "            " "     " "
## 6  ( 1 ) " "            " "            " "            " "     " "
## 7  ( 1 ) " "            " "            " "            "*"     " "
## 8  ( 1 ) " "            " "            " "            "*"     " "
##          enginesize enginetypedohcv enginetypel enginetypeohc enginetypeohcf
## 1  ( 1 ) "*"        " "             " "         " "           " "
## 2  ( 1 ) "*"        " "             " "         " "           " "
## 3  ( 1 ) "*"        " "             " "         " "           " "
## 4  ( 1 ) "*"        " "             " "         " "           " "
## 5  ( 1 ) "*"        " "             " "         " "           " "
## 6  ( 1 ) "*"        " "             " "         " "           " "
## 7  ( 1 ) "*"        " "             " "         " "           " "
## 8  ( 1 ) "*"        " "             " "         "*"           " "
##          enginetypeohcv enginetyperotor carwidth curbweight carlength
## 1  ( 1 ) " "            " "             " "      " "        " "
## 2  ( 1 ) " "            " "             "*"      " "        " "
## 3  ( 1 ) " "            " "             "*"      " "        " "
## 4  ( 1 ) " "            "*"             "*"      " "        " "
## 5  ( 1 ) "*"            "*"             "*"      " "        " "
## 6  ( 1 ) "*"            "*"             "*"      " "        " "
## 7  ( 1 ) "*"            "*"             "*"      " "        " "
## 8  ( 1 ) "*"            "*"             "*"      " "        " "
##          highwaympg boreratio stroke wheelbase drivewheelfwd drivewheelrwd
## 1  ( 1 ) " "        " "       " "    " "       " "           " "
## 2  ( 1 ) " "        " "       " "    " "       " "           " "
## 3  ( 1 ) " "        " "       " "    " "       " "           " "
## 4  ( 1 ) " "        " "       " "    " "       " "           " "
## 5  ( 1 ) " "        " "       " "    " "       " "           " "
## 6  ( 1 ) " "        " "       "*"    " "       " "           " "
## 7  ( 1 ) " "        " "       "*"    " "       " "           " "
## 8  ( 1 ) " "        " "       "*"    " "       " "           " "
##          enginelocationrear aspirationturbo doornumbertwo horsepower
## 1  ( 1 ) " "                " "             " "           " "
## 2  ( 1 ) " "                " "             " "           " "
## 3  ( 1 ) "*"                " "             " "           " "
## 4  ( 1 ) "*"                " "             " "           " "
## 5  ( 1 ) "*"                " "             " "           " "
## 6  ( 1 ) "*"                " "             " "           " "
## 7  ( 1 ) "*"                " "             " "           " "
## 8  ( 1 ) "*"                " "             " "           " "
##          compressionratio
## 1  ( 1 ) " "
## 2  ( 1 ) " "
## 3  ( 1 ) " "
## 4  ( 1 ) " "
## 5  ( 1 ) " "
## 6  ( 1 ) " "
## 7  ( 1 ) " "
## 8  ( 1 ) " "
```

Here we see the difference in features for the backward model compared to forward model. For example the second best model for backward and forward are different as shown in above figure.

```r
coef(regfit.full, 7)
```

```
##       (Intercept)         enginesize     enginetypeohcf     enginetypeohcv
##       -57712.5552           148.4221         -3804.7916         -4852.3248
##    enginetyperotor           carwidth             stroke enginelocationrear
##         7800.8444          1007.4850         -4307.4211         14256.4615
```

```r
coef(regfit.fwd, 7)
```

```
##       (Intercept)            peakrpm         enginesize     enginetypeohcv
##     -78543.314337           1.837289         127.381826       -3538.595514
##          carwidth             stroke      drivewheelrwd enginelocationrear
##       1099.363038       -2110.755789        1962.714917       10219.114749
```

```r
coef(regfit.bwd, 7)
```

```
##       (Intercept)            peakrpm         enginesize     enginetypeohcv
##     -73678.385495           1.555476         150.900289       -4741.677505
##    enginetyperotor           carwidth             stroke enginelocationrear
##       6784.909984        1048.461782       -2839.813241        9839.862024
```

From above result we see that the coefficient and features are different three approaches of subset selection, forward and backward selection.

## Validation Approach

Firstly, we divide the car data instances in test and train set.

```r
set.seed(1)

train = sample(c(TRUE, FALSE), nrow(card), rep= TRUE)

test = (!train)
```

Now we apply the subset selection method.

```r
regfit.best  = regsubsets(price~peakrpm+citympg
                          + enginesize+enginetype+carwidth+ curbweight+carlength
                          + highwaympg+ boreratio+ stroke + wheelbase + drivewheel
                          + enginelocation+ aspiration+ doornumber+ horsepower+ compressionratio,
                         data= card[train,], nvmax =19)
```

Now we test it on the separated set as we trained only using the training examples.

```r
test.mat =  model.matrix(price~peakrpm+citympg
                          + enginesize+enginetype+carwidth+ curbweight+carlength
                          + highwaympg+ boreratio+ stroke + wheelbase + drivewheel
                          + enginelocation+ aspiration+ doornumber+ horsepower+ compressionratio, data =

val.errors = rep(NA, 19)

for (i in 1:19){
  coefi = coef(regfit.best, id = i)
  pred =  test.mat[, names(coefi)]%*%coefi
  val.errors[i] =  mean((price[test]-pred)^2)
}
```

In previous we created test and cross validation set. I will use it to check model performance.

```r
val.errors
```

```
##  [1] 19984230 17389803 12842020 11551530 11345284 11966102 10394542 11589571
##  [9] 11851994 11101306 10412482 13537134 12214163 13447630 12119045 11461321
## [17] 11548419 11466866 11931765
```

```r
which.min(val.errors) # output 7
```

```
## [1] 7
```

```r
coef(regfit.best, 7)
```

```
##       (Intercept)        enginesize      enginetypeohc    enginetyperotor
##        -63677.139           123.365           3133.558           6514.236
##          carwidth            stroke      drivewheelrwd enginelocationrear
##          1074.321         -4063.746           2116.979          12109.372
```

In the output the model shows the 7 varibles for the best model. Then we carry out our analysis by taking the best model.

```r
predict.regsubsets = function(object, newdata, id, ...){
  form  =  as.formula(object$call[[2]])
  mat = model.matrix(form, newdata)
  coefi =  coef(object, id  =id)
  xvars = names(coefi)
  mat[, xvars]%*%coefi
}
```

The above function is an user defined prediction method.

```r
regfit.best =  regsubsets(price~peakrpm+citympg
                          + enginesize+enginetype+carwidth+ curbweight+carlength
                          + highwaympg+ boreratio+ stroke + wheelbase + drivewheel
                          + enginelocation+ aspiration+ doornumber+ horsepower+ compressionratio,
                          data= card, nvmax = 19)

coef(regfit.best, 7)
```

```
##        (Intercept)         enginesize     enginetypeohcf      enginetypeohcv
##        -57712.5552           148.4221         -3804.7916          -4852.3248
##    enginetyperotor           carwidth             stroke enginelocationrear
##          7800.8444          1007.4850         -4307.4211          14256.4615
```

Now we take 10 training set.

```
k = 10
set.seed(1)
folds = sample(1:k, nrow(card), replace =TRUE)

cv.errors = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
```

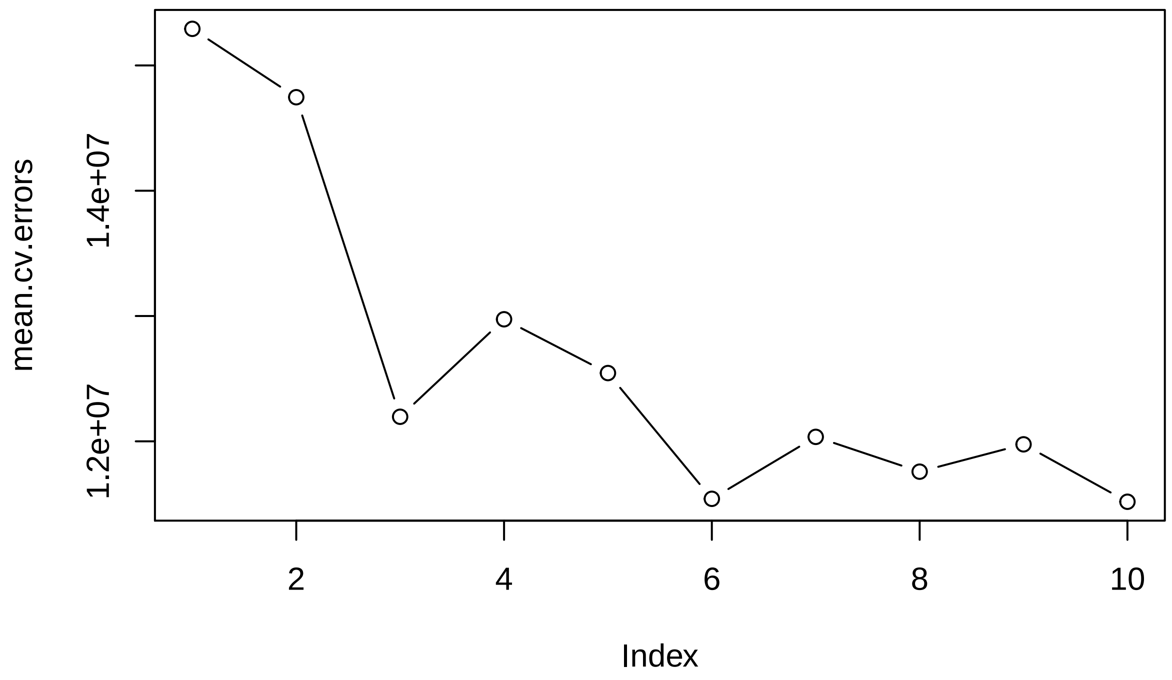
```
for (j in 1:k){
  best.fit =  regsubsets(price~peakrpm+citympg+ enginesize+fueltype
                        +carwidth+curbweight+carlength
                        + highwaympg+ boreratio+ stroke + wheelbase,
                        data= card[folds!=j,], nvmax = 10)
  for (i in 1:10){
    pred = predict(best.fit, card[folds==j,], id = i)
    cv.errors[j, i]= mean((price[folds==j]-pred)^2)
  }
}
```

Now we have 10x10 matrix which (i,j) corresponds to the MSE of ith validation for the best j-variable model

```
mean.cv.errors = apply(cv.errors, 2, mean)

mean.cv.errors
```

```
##        1        2        3        4        5        6        7        8
## 15291913 14746398 12195925 12974205 12544768 11541076 12035643 11757833
##        9       10
## 11975901 11517819
```

```
par(mfrow = c(1,1))

plot(mean.cv.errors, type ="b")
```



The plot shows mean error with different feaures using the cross validation approach.

```
reg.best  =  regsubsets(price~fuelsystem+peakrpm+citympg
                       + enginesize+enginetype+carwidth+curbweight+carlength
                       + highwaympg+ boreratio+ stroke + wheelbase + drivewheel
                       + enginelocation+ aspiration+ doornumber+ horsepower+ compressionratio,
                       data= card, nvmax = 19)

coef(reg.best, 11)
```

```
##      (Intercept)          peakrpm         enginesize      enginetypeohc
##    -53108.326745         1.416018         163.509438        1951.327128
##    enginetypeohcv  enginetyperotor           carwidth          boreratio
##     -4329.980717      8254.922067         993.205807       -4051.216381
##           stroke     drivewheelfwd enginelocationrear     aspirationturbo
##     -4374.872812     -1906.461545       10841.290144        1722.404716
```

## Ridge Regression and Lasso

### Ridge Regression

In this experiment the glmnet r package will be used for carry out lasso and ridge regression. In the function we select alpha as 0 for ridgre regression.

```r
x =  model.matrix(price~fuelsystem+peakrpm+citympg
                  + enginesize+enginetype+carwidth+curbweight+carlength
                  + highwaympg+ boreratio+ stroke + wheelbase + drivewheel
                  + enginelocation+ aspiration+ doornumber+ horsepower+ compressionratio,
                       data= card)[,-1]


y = price
```

We have renamed our variables to conduct the next experiments.

```r
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 3.0-2
```

```r
grid  = 10^seq(10,-2,length=100)
ridge.mod = glmnet(x,y, alpha = 0, lambda = grid)
```

```r
dim(coef(ridge.mod))
```

```
## [1]  31 100
```

We get the size of 31x100, where 31 are for 9 predictors and intercept. We have total 30 variables under consideration. The 100 is for 100 different value of lambda.

```r
ridge.mod$lambda[40]
```

```
## [1] 187381.7
```

```r
coef(ridge.mod)[,40]
```

```
##       (Intercept)     fuelsystem2bbl     fuelsystem4bbl      fuelsystemidi
##      317.91583110      -255.71343575       -54.41856690        96.51008402
##      fuelsystemmfi     fuelsystemmpfi     fuelsystemspdi     fuelsystemspfi
##      -38.63725175       247.69181319       -92.64081048       -97.98608462
##           peakrpm            citympg         enginesize    enginetypedohcv
##       -0.03673657       -25.46071261         5.48887688       600.07601062
##        enginetypel      enginetypeohc     enginetypeohcf     enginetypeohcv
##       -5.49532973      -174.39748241        18.46413682       397.08019754
##    enginetyperotor           carwidth         curbweight          carlength
##      -23.33075986        88.97961050         0.40035969        13.42014128
##        highwaympg          boreratio             stroke          wheelbase
##      -24.52574014       482.90216899        64.03113340        23.25517338
##      drivewheelfwd      drivewheelrwd enginelocationrear    aspirationturbo
##     -297.38476312       326.17609539       804.93710933       102.08537890
##       doornumbertwo         horsepower   compressionratio
##       -5.15241216         5.24228034         5.12026915
```

```
sqrt(sum(coef(ridge.mod)[-1, 40]^2))
```

```
## [1] 1344.713
```

In the above result, we find the l2 norm of 1344 for lambda of 187381

```
ridge.mod$lambda[60]
```

```
## [1] 705.4802
```

```
coef(ridge.mod)[,60]
```

```
##       (Intercept)     fuelsystem2bbl     fuelsystem4bbl      fuelsystemidi
##     -4.867311e+04      -1.254438e+02      -6.842027e+01       1.490822e+03
##     fuelsystemmfi     fuelsystemmpfi     fuelsystemspdi     fuelsystemspfi
##     -2.549469e+03      -1.007556e+02      -1.918467e+03      -1.923453e+03
##           peakrpm            citympg         enginesize    enginetypedohcv
##      7.665295e-01      -7.194260e+00       8.228998e+01       4.227470e+02
##       enginetypel      enginetypeohc     enginetypeohcf     enginetypeohcv
##     -1.923172e+03       1.352813e+03       1.706284e+02      -1.244039e+03
##    enginetyperotor           carwidth          curbweight          carlength
##      3.791835e+03       6.409885e+02       2.703191e+00      -1.099005e+01
##        highwaympg          boreratio             stroke          wheelbase
##     -5.437710e+00      -2.206767e+03      -2.274208e+03       1.151903e+02
##      drivewheelfwd      drivewheelrwd enginelocationrear    aspirationturbo
##     -6.192992e+02       1.112299e+03       1.179112e+04       4.576378e+02
##      doornumbertwo         horsepower   compressionratio
##      1.734074e+02       3.074386e+01      -3.418842e+01
```

```
sqrt(sum(coef(ridge.mod)[-1, 60]^2))
```

```
## [1] 13753.58
```

In the above result, we find the l2 norm of 13753 for lambda of 705. So, we conclude in the car data that for smaller lambda we get smaller l2 error.

```
predict(ridge.mod, s = 50, type= "coefficients")[1:20,]
```

```
##     (Intercept)  fuelsystem2bbl  fuelsystem4bbl   fuelsystemidi   fuelsystemmfi
##   -38060.213941      111.114417     -829.752265     9315.356962    -3040.814172
##  fuelsystemmpfi  fuelsystemspdi  fuelsystemspfi         peakrpm         citympg
##      678.706065    -2460.961617     -892.175511        1.931968      -20.656992
##      enginesize enginetypedohcv     enginetypel   enginetypeohc  enginetypeohcf
##      151.111619     2598.204782    -1630.423702     1945.435424      333.657362
##  enginetypeohcv enginetyperotor        carwidth      curbweight       carlength
##    -3760.346346     9501.808424      669.201875        3.265289      -50.702701
```

In above we get new prediction for a new lambda of 50.

We split the training instances for estimating test error in rigde and lasso.

```
set.seed(1)
train = sample(1:nrow(x), nrow(x)/2)
test = (-train)
y.test = y[test]
```

Now we conduct experiment on the segmented data using glm.

```
ridge.mod =  glmnet(x[train,], y[train], alpha = 0, lambda = grid, thresh = 1e-12)
ridge.pred =  predict(ridge.mod, s =4, newx = x[test,])
mean((ridge.pred - y.test)^2)
```

```
## [1] 9866127
```

Using features we find the error of 9866127

```
mean((mean(y[train])-y.test)^2)
```

```
## [1] 65445936
```

If we use only the mean to predict the result we get higher error of 65445936.

We can aslo check same performance using very high lambda.

```
ridge.pred =  predict(ridge.mod, s = 1e10, newx = x[test,])
mean((ridge.pred-y.test)^2)
```

```
## [1] 65445316
```

In above we get very similar value of 65M of the earlier mean only model.

```
ridge.pred = predict(ridge.mod, s =0, newx = x[test,])
mean((ridge.pred-y.test)^2)
```

```
## [1] 9903438
```

```
lm(y~x, subset =  train)
```

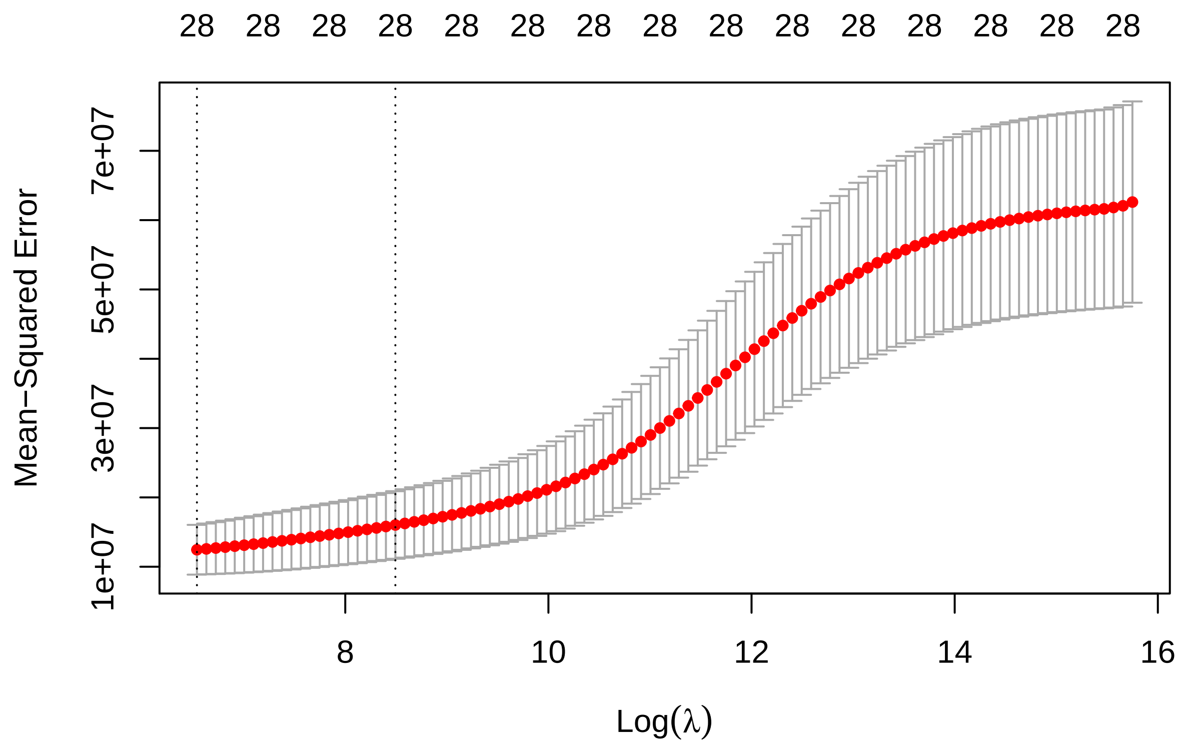
```
##
## Call:
## lm(formula = y ~ x, subset = train)
##
## Coefficients:
##         (Intercept)      xfuelsystem2bbl      xfuelsystem4bbl
##          -46019.584             -122.079            -1115.572
##      xfuelsystemidi       xfuelsystemmfi      xfuelsystemmpfi
##             583.330                   NA              400.914
##     xfuelsystemspdi      xfuelsystemspfi             xpeakrpm
##           -2668.129                   NA                1.754
##           xcitympg          xenginesize     xenginetypedohcv
##               4.886              200.276             4519.275
##        xenginetypel       xenginetypeohc      xenginetypeohcf
##           -2204.640              808.099            -2616.709
##     xenginetypeohcv     xenginetyperotor            xcarwidth
##           -5835.691            12691.241             1070.224
##         xcurbweight           xcarlength          xhighwaympg
##               1.886              -40.143               34.513
##          xboreratio              xstroke           xwheelbase
##           -2282.749            -5743.011             -134.104
##      xdrivewheelfwd       xdrivewheelrwd  xenginelocationrear
##             583.449             1486.237            15943.692
##    xaspirationturbo      xdoornumbertwo          xhorsepower
##            4833.875             -947.648              -50.918
##   xcompressionratio
##            -178.472
```

```
predict(ridge.mod, s=0,  newx = x[test,], type="coefficients")[1:20,]
```

```
##     (Intercept)  fuelsystem2bbl  fuelsystem4bbl   fuelsystemidi   fuelsystemmfi
##   -46019.459092     -122.097895    -1115.355871      583.284935        0.000000
##  fuelsystemmpfi  fuelsystemspdi  fuelsystemspfi         peakrpm         citympg
##      400.769613    -2668.197866        0.000000        1.753604        4.893521
##      enginesize enginetypedohcv     enginetypel   enginetypeohc  enginetypeohcf
##      200.267096     4518.078384    -2204.513145      808.249332    -2616.449885
##  enginetypeohcv enginetyperotor        carwidth      curbweight       carlength
##    -5835.474487    12690.751953     1070.208519        1.885694      -40.150598
```

The mean for lambda 0 is similar to 9M of the prediction error of lambda 4.

```
set.seed(1)
cv.out = cv.glmnet(x[train, ], y[train], alpha = 0)
plot(cv.out)
```



```
bestlam = cv.out$lambda.min
bestlam
```

```
## [1] 691.9598
```

We find the value of best lambda is around 692 using the validation set approach. Now we predict using the best lambda and check the mse value on test dataset.

```
ridge.pred = predict(ridge.mod, s =bestlam, newx = x[test,])
mean((ridge.pred - y.test)^2)
```

```
## [1] 9602655
```

We also find the best model coefficient using the bestlambda.

```
out =  glmnet(x,y, alpha = 0)
predict(out, type = "coefficients", s = bestlam)[1:20,]
```

```
##     (Intercept)  fuelsystem2bbl  fuelsystem4bbl   fuelsystemidi   fuelsystemmfi
##   -4.846268e+04   -1.307088e+02   -5.707612e+01    1.505479e+03   -2.550307e+03
##  fuelsystemmpfi  fuelsystemspdi  fuelsystemspfi         peakrpm         citympg
##   -9.830366e+01   -1.922857e+03   -1.907042e+03    7.611355e-01   -7.971368e+00
##      enginesize enginetypedohcv     enginetypel   enginetypeohc  enginetypeohcf
##    8.259412e+01    3.796631e+02   -1.906869e+03    1.364857e+03    1.739892e+02
##  enginetypeohcv enginetyperotor        carwidth      curbweight       carlength
##   -1.258224e+03    3.810482e+03    6.398897e+02    2.670809e+00   -1.220085e+01
```

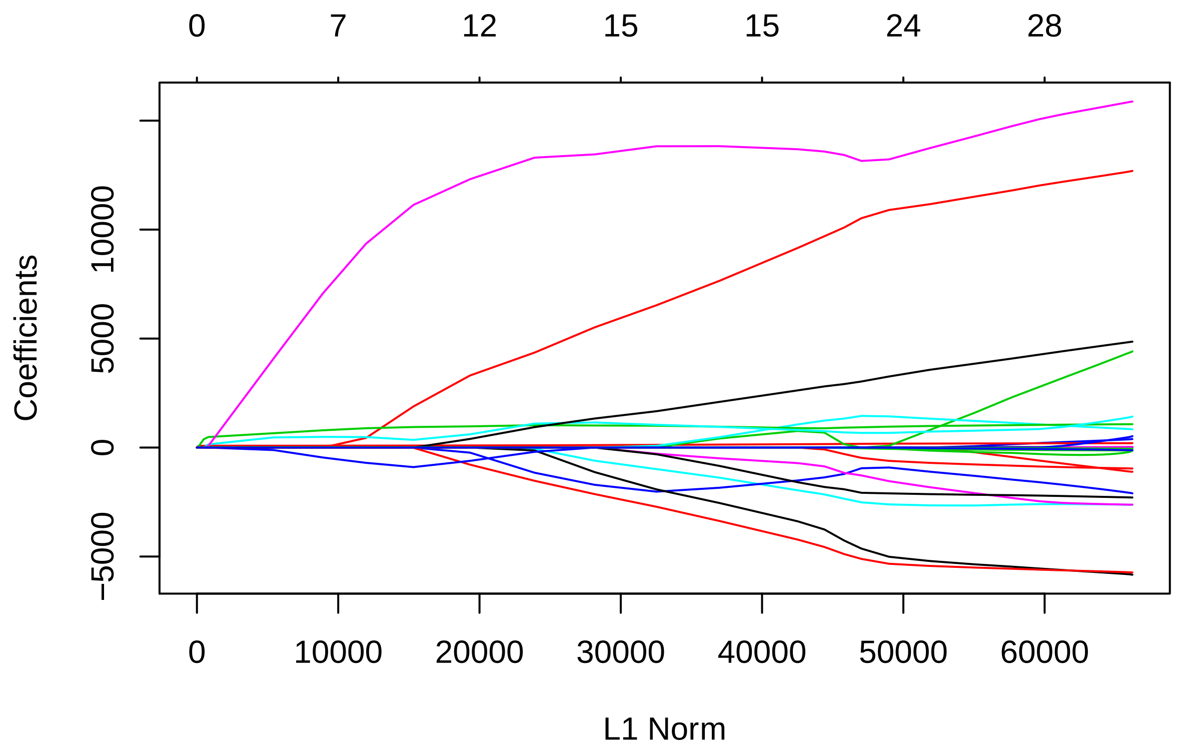
We see the coefficients are not zero that often.

**Lasso**

We use the alpha of 1 to implement lasso using the similar method used for rigde regression. We also plot the model for visualization.

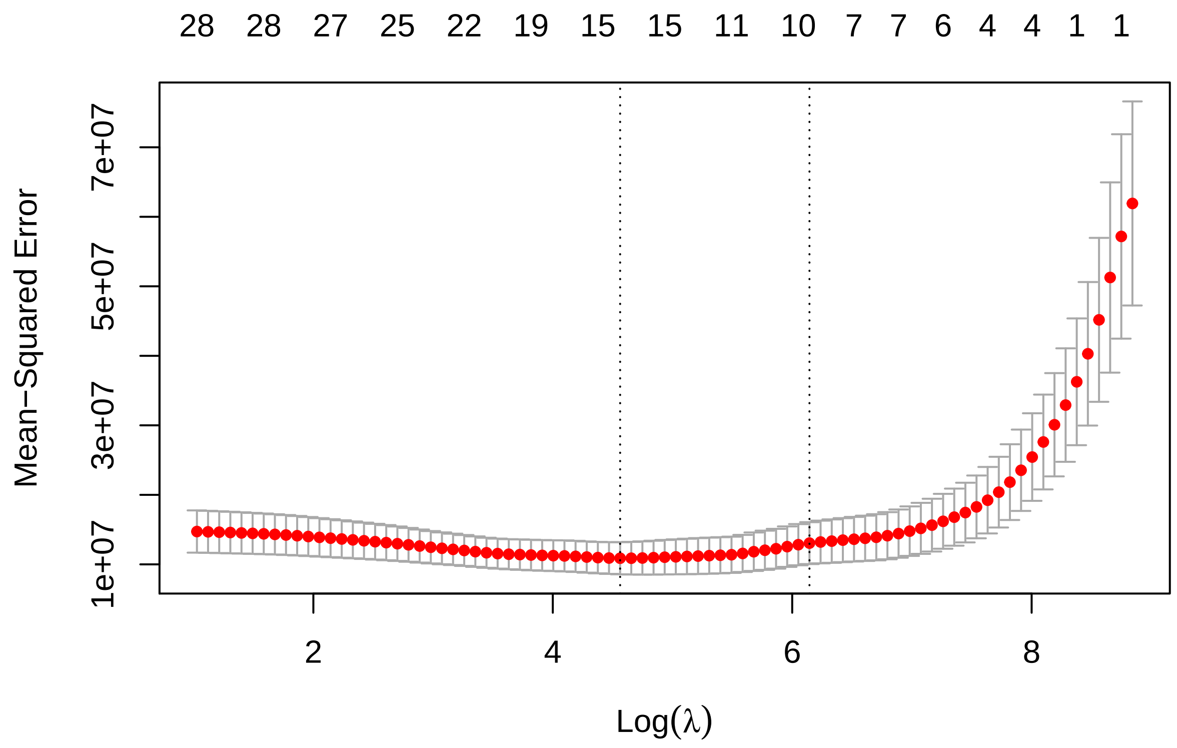
```r
lasso.mod = glmnet(x[train,], y[train], alpha = 1, lambda = grid)
plot(lasso.mod)
```

```
## Warning in regularize.values(x, y, ties, missing(ties)): collapsing to unique
## 'x' values
```



I use the random train test split for the lass and find the best lambda value.

```r
set.seed(1)
cv.out =  cv.glmnet(x[train,], y[train], alpha=1)
plot(cv.out)
```



```
bestlam = cv.out$lambda.min
lasso.pred = predict(lasso.mod, s =bestlam, newx = x[test,])
mean((lasso.pred-y.test)^2)
```

```
## [1] 9456810
```

For best lambda the lasso also provides the error of about 9M like the ridge model.

```
out = glmnet(x,y, alpha =1, lambda =grid)
lasso.coef = predict(out, type ="coefficients", s= bestlam)[1:20,]
lasso.coef
```

```
##     (Intercept)  fuelsystem2bbl  fuelsystem4bbl   fuelsystemidi   fuelsystemmfi
##   -48252.335050       -2.068202        0.000000      479.481926    -1806.104148
##  fuelsystemmpfi  fuelsystemspdi  fuelsystemspfi         peakrpm         citympg
##       56.647436    -1604.347645     -102.546469        1.332641        0.000000
##      enginesize enginetypedohcv     enginetypel   enginetypeohc  enginetypeohcf
##      132.917423      331.801144     -543.830839     1396.374341        0.000000
##  enginetypeohcv enginetyperotor        carwidth     curbweight       carlength
##    -2633.789111     6013.736600      705.197259        1.674569        0.000000
```

```
lasso.coef[lasso.coef != 0]
```

```
##     (Intercept)  fuelsystem2bbl   fuelsystemidi   fuelsystemmfi  fuelsystemmpfi
##   -48252.335050       -2.068202      479.481926    -1806.104148       56.647436
##  fuelsystemspdi  fuelsystemspfi         peakrpm      enginesize enginetypedohcv
##    -1604.347645     -102.546469        1.332641      132.917423      331.801144
##     enginetypel   enginetypeohc  enginetypeohcv enginetyperotor        carwidth
##     -543.830839     1396.374341    -2633.789111     6013.736600      705.197259
##      curbweight
##        1.674569
```

In lasso coefficient we find many values are zeros unlike the ridge regression.

## PCR and PLS

### Principle Components Regression (PCR)

For the experiment the pls library of r will be used.

```r
library(pls)
```

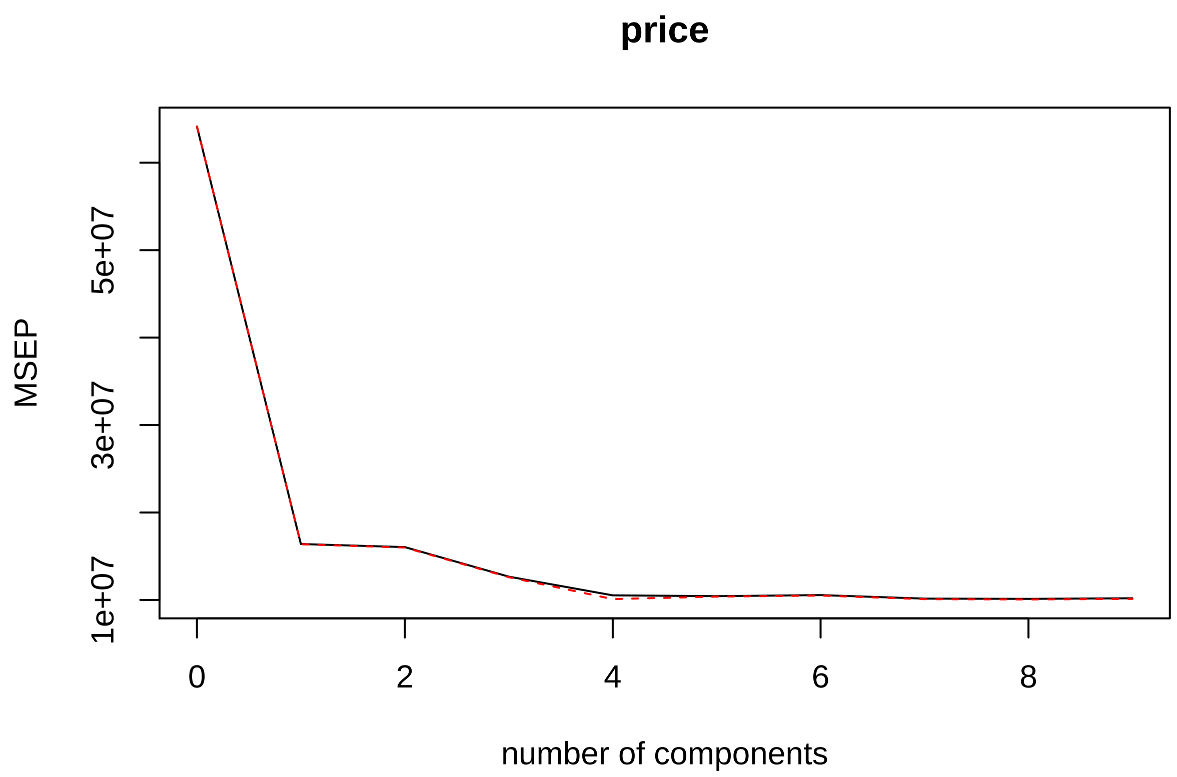
```
##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings
```

```r
set.seed(2)
pcr.fit = pcr(price~peakrpm+citympg+ enginesize
              +carwidth+curbweight+carlength
              + highwaympg+  horsepower+enginelocation,
              data = card, scale = TRUE, validation = "CV")



summary(pcr.fit)
```

```
## Data:    X dimension: 205 9
##  Y dimension: 205 1
## Fit method: svdpc
## Number of components considered: 9
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            8008     4049     4006     3559     3245     3230     3249
## adjCV         8008     4046     4000     3550     3178     3222     3241
##        7 comps  8 comps  9 comps
## CV        3187     3182     3193
## adjCV     3177     3172     3182
##
## TRAINING: % variance explained
##        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X         61.48    78.30    88.21    92.44    96.46       98    99.05    99.77
## price     74.98    76.18    81.35    84.88    84.99       85    85.78    85.90
##        9 comps
## X       100.00
## price    85.92
```

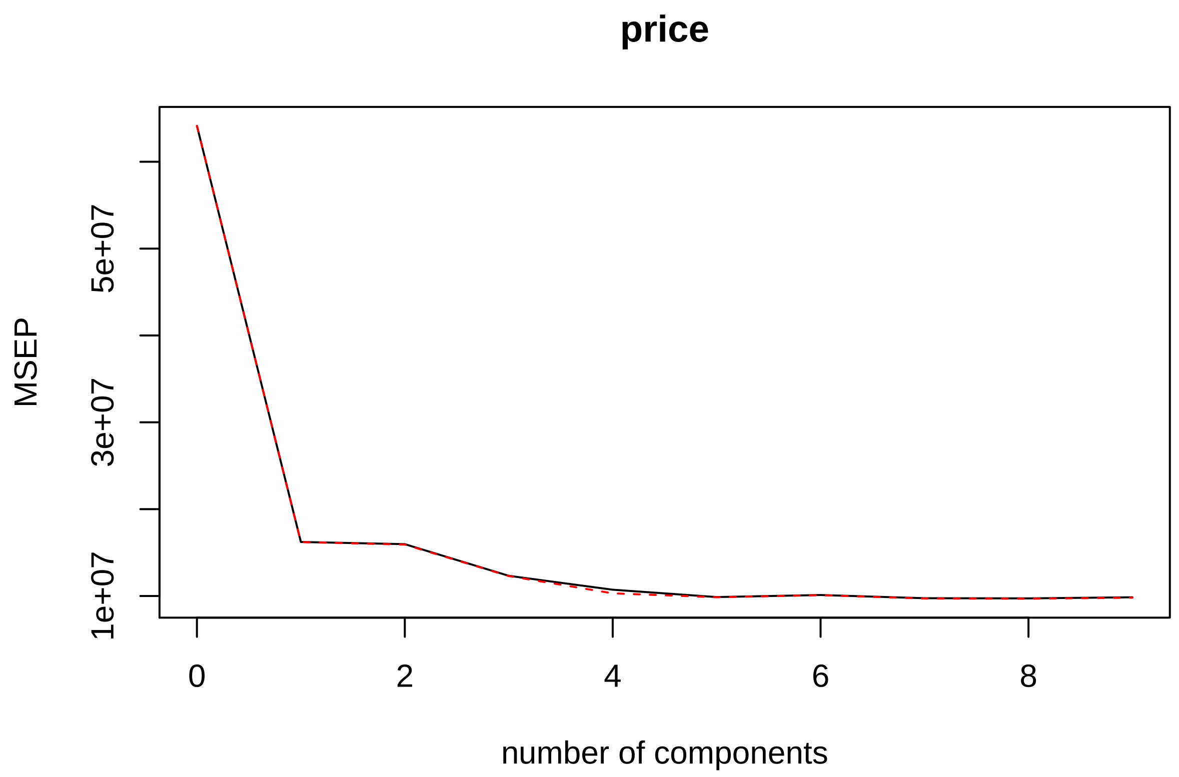
We find the the PCR model performed bes in case of 8 components when error is 3182 the lower than anyother. The 5 componest also covered 97% of the total variance.

```r
validationplot(pcr.fit, val.type = "MSEP")
```



We see the rsult in the plot with the error and components added.

```r
set.seed(1)

pcr.fit = pcr(price~peakrpm+citympg+ enginesize
              +carwidth+curbweight+carlength
              + highwaympg+  horsepower+enginelocation,
              data = card, scale = TRUE, validation = "CV")

validationplot(pcr.fit, val.type = "MSEP")
```



The above plot shows the model preformance on the test data. We also observe the lowest error around the 8th components. At next we use the 8 components to predict the error on test instances.

```
x =  model.matrix(price~peakrpm+citympg+ enginesize
                  +carwidth+curbweight+carlength
                  + highwaympg+  horsepower+enginelocation,
                  data= card)[,-1]


y = price
pcr.pred = predict(pcr.fit, x[test,], ncomp =8)
mean((pcr.pred - y.test)^2)
```

```
## [1] 9278896
```

The test error si 9278896 using the best components. We then retrain the model using the 8 component as found for smallest error.

```
pcr.fit =  pcr(y~x, scale= TRUE, ncomp = 8)

summary(pcr.fit)
```

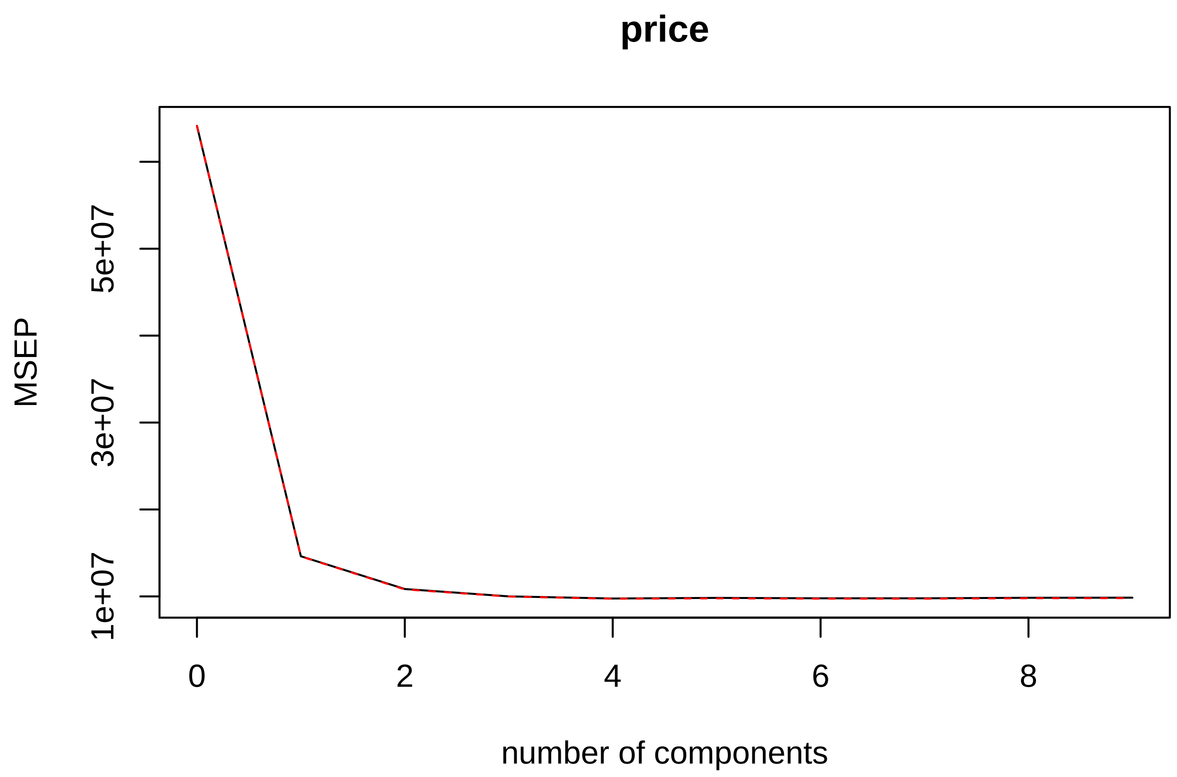
```
## Data:    X dimension: 205 9
##  Y dimension: 205 1
## Fit method: svdpc
## Number of components considered: 8
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X    61.48    78.30    88.21    92.44    96.46       98    99.05    99.77
## y    74.98    76.18    81.35    84.88    84.99       85    85.78    85.90
```

### Partial Least Squares

```
set.seed(1)

#partial least square

pls.fit = plsr(price~peakrpm+citympg+ enginesize
               +carwidth+curbweight+carlength
               + highwaympg+  horsepower+enginelocation,
               data = card,  scale = TRUE, validation = "CV")

summary(pls.fit)
```

```
## Data:    X dimension: 205 9
##  Y dimension: 205 1
## Fit method: kernelpls
## Number of components considered: 9
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            8008     3823     3294     3162     3124     3135     3128
## adjCV         8008     3821     3287     3160     3119     3126     3121
##        7 comps  8 comps  9 comps
## CV        3127     3137     3139
## adjCV     3120     3129     3131
##
## TRAINING: % variance explained
##        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X        61.32    72.93    84.85    92.20    93.46    96.13    98.30    99.48
## price    77.61    83.84    85.03    85.56    85.90    85.91    85.91    85.91
##        9 comps
## X       100.00
## price    85.92
```

```
validationplot(pls.fit, val.type = "MSEP")
```



From above result we see the lowest error occurs for the 4th component the value of 3124. In test case we use 4 components as found here.

```
pls.pred = predict(pls.fit, x[test,], ncomp = 4)

mean((pls.pred - y.test)^2)
```

```
## [1] 9449577
```

```
pls.fit =  plsr(price~peakrpm+citympg+ enginesize
                +carwidth+curbweight+carlength
                + highwaympg+  horsepower+enginelocation,
                data = card,  scale = TRUE, ncomp =4)

summary(pls.fit)
```

```
## Data:    X dimension: 205 9
##  Y dimension: 205 1
## Fit method: kernelpls
## Number of components considered: 4
## TRAINING: % variance explained
##        1 comps  2 comps  3 comps  4 comps
## X        61.32    72.93    84.85    92.20
## price    77.61    83.84    85.03    85.56
```

Using 4 components we find error of 9446577, very similar to PCR but with smaller number of components. PCR chose more components.

# Chapter 7

## Polynomial Regression and Step function

In this section I will implement different nonlinear estimation method to experiment on car dataset. Firsly we will use poly function with lm function to implement the polynomial regression. In this experiment, I will use only one feature of enginesize for better explanation. We have seen the significance of enginesize feature in last chapter.

```
library(ISLR)
attach(card)
```

```
## The following object is masked _by_ .GlobalEnv:
##
##     high

## The following objects are masked from card (pos = 6):
##
##     aspiration, boreratio, car_ID, carbody, carheight, carlength,
##     CarName, carwidth, citympg, compressionratio, curbweight,
##     cylindernumber, doornumber, drivewheel, enginelocation, enginesize,
##     enginetype, fuelsystem, fueltype, high, highwaympg, horsepower,
##     peakrpm, price, stroke, symboling, wheelbase

## The following objects are masked from card (pos = 11):
##
##     aspiration, boreratio, car_ID, carbody, carheight, carlength,
##     CarName, carwidth, citympg, compressionratio, curbweight,
##     cylindernumber, doornumber, drivewheel, enginelocation, enginesize,
##     enginetype, fuelsystem, fueltype, highwaympg, horsepower, peakrpm,
##     price, stroke, symboling, wheelbase

## The following objects are masked from card (pos = 12):
##
##     aspiration, boreratio, car_ID, carbody, carheight, carlength,
##     CarName, carwidth, citympg, compressionratio, curbweight,
##     cylindernumber, doornumber, drivewheel, enginelocation, enginesize,
##     enginetype, fuelsystem, fueltype, highwaympg, horsepower, peakrpm,
##     price, stroke, symboling, wheelbase

## The following objects are masked from card (pos = 14):
##
##     aspiration, boreratio, car_ID, carbody, carheight, carlength,
##     CarName, carwidth, citympg, compressionratio, curbweight,
##     cylindernumber, doornumber, drivewheel, enginelocation, enginesize,
##     enginetype, fuelsystem, fueltype, highwaympg, horsepower, peakrpm,
##     price, stroke, symboling, wheelbase
```

```
fit = lm(price~poly(enginesize, 4), data = card)
coef(summary(fit))
```

```
##                          Estimate Std. Error     t value      Pr(>|t|)
## (Intercept)            13276.7106   255.8457 51.89341988 5.439147e-118
## poly(enginesize, 4)1  99743.0989  3663.1536 27.22875167  3.369131e-69
## poly(enginesize, 4)2  -1165.2811  3663.1536 -0.31810873  7.507344e-01
## poly(enginesize, 4)3 -19642.6343  3663.1536 -5.36221972  2.257410e-07
## poly(enginesize, 4)4   -154.3066  3663.1536 -0.04212397  9.664419e-01
```

There are also alternative ways to implement the polynomial regression over the dataset. We can use I or use the cbind function for conci

```
fit2 = lm(price~poly(enginesize, 4, raw=T), data = card)
coef(summary(fit))
```

```
##                          Estimate Std. Error     t value      Pr(>|t|)
## (Intercept)            13276.7106   255.8457 51.89341988 5.439147e-118
## poly(enginesize, 4)1  99743.0989  3663.1536 27.22875167  3.369131e-69
## poly(enginesize, 4)2  -1165.2811  3663.1536 -0.31810873  7.507344e-01
## poly(enginesize, 4)3 -19642.6343  3663.1536 -5.36221972  2.257410e-07
## poly(enginesize, 4)4   -154.3066  3663.1536 -0.04212397  9.664419e-01
```

```
fit2a = lm(price~enginesize+I(enginesize^2)+I(enginesize^3) + I(enginesize^4), data=card)

coef(fit2a)
```

```
##     (Intercept)      enginesize I(enginesize^2) I(enginesize^3) I(enginesize^4)
##    2.171021e+04   -4.221207e+02    3.530566e+00   -6.129592e-03   -7.917883e-07
```

```
fit2b =  lm(price~cbind(enginesize, enginesize^2, enginesize^3,enginesize^4), data=card)

coef(fit2b)
```

```
##                                                              (Intercept)
##                                                             2.171021e+04
## cbind(enginesize, enginesize^2, enginesize^3, enginesize^4)enginesize
##                                                            -4.221207e+02
##           cbind(enginesize, enginesize^2, enginesize^3, enginesize^4)
##                                                             3.530566e+00
##           cbind(enginesize, enginesize^2, enginesize^3, enginesize^4)
##                                                            -6.129592e-03
##           cbind(enginesize, enginesize^2, enginesize^3, enginesize^4)
##                                                            -7.917883e-07
```

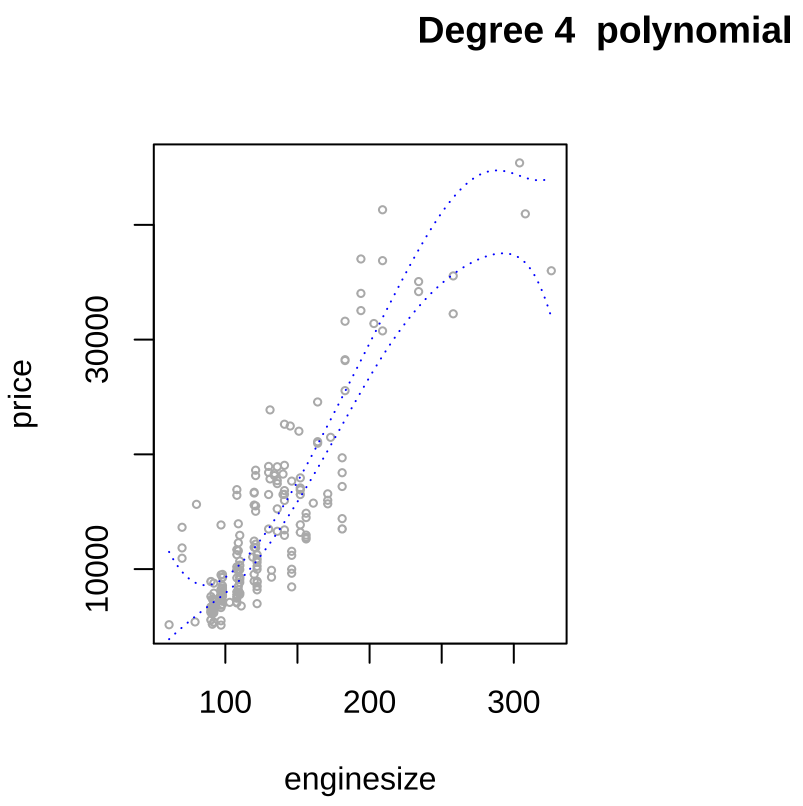
In the previous two section, we find similar coefficients for all the poly, I and cbind function methods. This shows the equivalance of the implementaions.

Here we specify the range of enginesize for prediction.

```
engsrange =  range(enginesize)

engs.grid =  seq(from=engsrange[1], to = engsrange[2])

preds= predict(fit, newdata = list(enginesize=engs.grid), se=TRUE)

se.bands =  cbind(preds$fit+2*preds$se.fit, preds$fit-2*preds$se)
```

Now we can plot the result of previous sectoin.

```
#plot
par(mfrow=c(1,2), mar = c(4.5, 4.5,1,1), oma=c(0,0,4,0))
plot(enginesize, price, xlim= engsrange, cex=0.5, col="darkgrey")
title("Degree 4  polynomial", outer=T)
matlines(engs.grid, se.bands, lwd=1, col="blue", lty=3)
```



Next, we re-evaluate the equivalence between the poly() and I() method by check the prediciton differences.

```
preds2= predict(fit2, newdata=list(enginesize=engs.grid), se=TRUE)
max(abs(preds$fit- preds2$fit))
```

```
## [1] 2.582965e-10
```

The prediction are almost same.

```
fit.1 = lm(price~enginesize, data=card)
fit.2 = lm(price~poly(enginesize,2), data=card)
fit.3 = lm(price~poly(enginesize,3), data=card)
fit.4 = lm(price~poly(enginesize,4), data=card)
fit.5 = lm(price~poly(enginesize,5), data=card)
anova(fit.1, fit.2, fit.3, fit.4, fit.5)
```

```
## Analysis of Variance Table
```

```
##
## Model 1: price ~ enginesize
## Model 2: price ~ poly(enginesize, 2)
## Model 3: price ~ poly(enginesize, 3)
## Model 4: price ~ poly(enginesize, 4)
## Model 5: price ~ poly(enginesize, 5)
##   Res.Df        RSS Df Sum of Sq       F    Pr(>F)
## 1    203 3070953588
## 2    202 3069595708  1   1357880  0.1009    0.7511
## 3    201 2683762625  1 385833083 28.6606 2.365e-07 ***
## 4    200 2683738814  1     23811  0.0018    0.9665
## 5    199 2678969866  1   4768948  0.3542    0.5524
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Here, the linear polynomial seems fit. But changing from 1 to 2 is not significant.

```
coef(summary(fit.5))
```

```
##                          Estimate Std. Error     t value      Pr(>|t|)
## (Intercept)            13276.7106   256.2598 51.80957650 1.758719e-117
## poly(enginesize, 5)1   99743.0989  3669.0816 27.18475861  6.373145e-69
## poly(enginesize, 5)2   -1165.2811  3669.0816 -0.31759477  7.511253e-01
## poly(enginesize, 5)3 -19642.6343  3669.0816 -5.35355607  2.364929e-07
## poly(enginesize, 5)4    -154.3066  3669.0816 -0.04205591  9.664963e-01
## poly(enginesize, 5)5   -2183.7922  3669.0816 -0.59518767  5.523942e-01
```

The result is also evident from the anova test above. We see the relation of t value and p value from the above result

```
(-5.3535)^2
```

```
## [1] 28.65996
```

The same as 28.6606 of the avona result earlier.

In next section, we add another feaure carwidth for the analysis

```
fit.1 = lm(price~carwidth+enginesize, data=card)
fit.2 = lm(price~poly(enginesize,2)+carwidth, data=card)
fit.3 = lm(price~poly(enginesize,3)+carwidth, data=card)
anova(fit.1, fit.2, fit.3)
```

```
## Analysis of Variance Table
##
## Model 1: price ~ carwidth + enginesize
## Model 2: price ~ poly(enginesize, 2) + carwidth
## Model 3: price ~ poly(enginesize, 3) + carwidth
##   Res.Df        RSS Df Sum of Sq       F    Pr(>F)
## 1    202 2686419306
## 2    201 2671194631  1  15224675  1.3799    0.2415
## 3    200 2206634677  1 464559954 42.1057 6.661e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We find the model 2 to model 3 is insignificant.

In next section we create class label for the price in car dataset by selecting the modality.

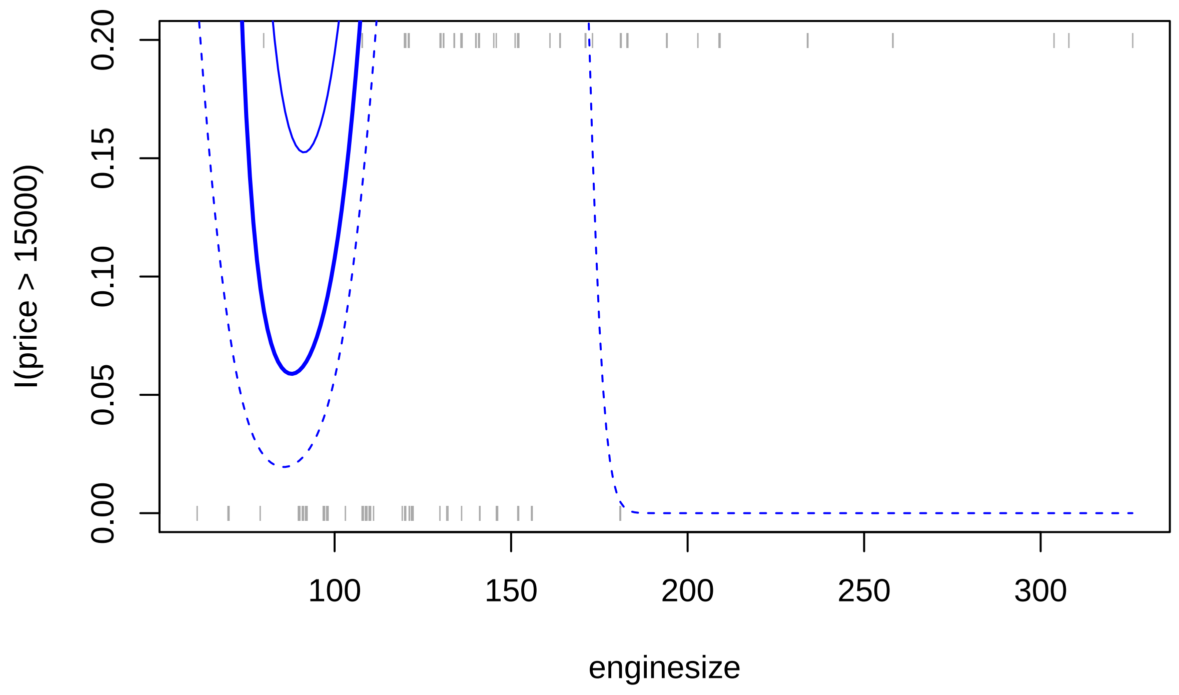
```
fit = glm(I(price>10300)~poly(enginesize, 4), data=card, family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
preds = predict(fit, newdata = list(enginesize=engs.grid), se=T)
```

```
pfit = exp(preds$fit)/(1+exp(preds$fit))
se.bands.logit = cbind(preds$fit+2*preds$se.fit, preds$fit-2*preds$se.fit)
se.bands = exp(se.bands.logit)/(1+exp(se.bands.logit))

preds = predict(fit, newdata = list(enginesize=engs.grid), type="response", se=T)
```

```
plot(enginesize, I(price>15000), xlim=engsrange, type ="n", ylim= c(0,0.2))
points(jitter(enginesize), I((price>15000)/5), cex=0.5, pch="|", col="darkgrey")
lines(engs.grid, pfit, lwd=2, col="blue")
matlines(engs.grid, se.bands, lwd = 1, col="blue", lyt=3)
```



```
table(cut(enginesize, 4))
```

```
##
## (60.7,127]  (127,194]  (194,260]  (260,326]
##        130         61         11          3
```

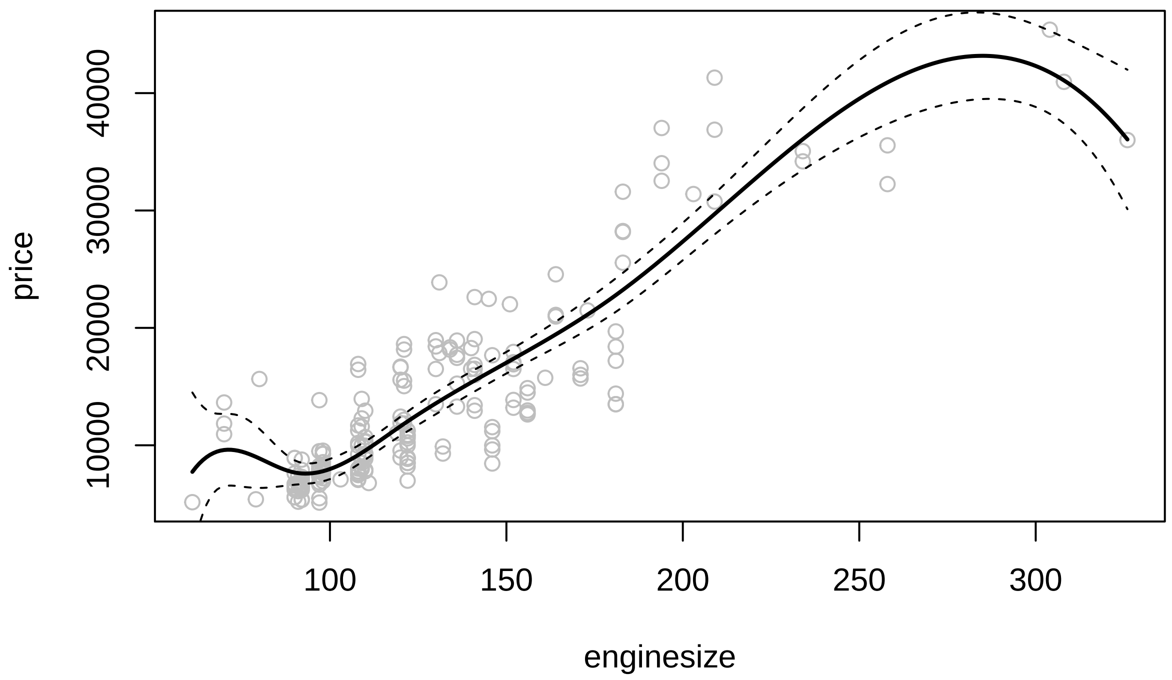
```r
fit =lm(price~cut(enginesize, 4), data=card)
coef(summary(fit))
```

```
##                              Estimate Std. Error  t value     Pr(>|t|)
## (Intercept)                  9078.862   314.6253 28.85611 2.098258e-73
## cut(enginesize, 4)(127,194]  7939.731   556.7310 14.26134 2.427508e-32
## cut(enginesize, 4)(194,260] 25555.638  1126.4369 22.68715 2.459773e-57
## cut(enginesize, 4)(260,326] 31707.805  2094.8786 15.13587 4.816748e-35
```

The cut method selected the point 127, 194 and 260 enginesize.

### spline

```r
library(splines)

fit = lm(price~bs(enginesize, knots = c(90,120,180)), data=card)
pred = predict(fit, newdata = list(enginesize=engs.grid), se=T)
plot(enginesize, price, col="grey")
lines(engs.grid, pred$fit, lwd=2)
lines(engs.grid, pred$fit+2*pred$se, lty="dashed")
lines(engs.grid, pred$fit-2*pred$se, lty="dashed")
```



In the implementation we specified 90, 120 and 180 as knots to create spline of 6 basis functions.

```r
dim(bs(enginesize, knots=c(90,120,180)))
```

```
## [1] 205   6
```

```r
dim(bs(enginesize, df=6))
```

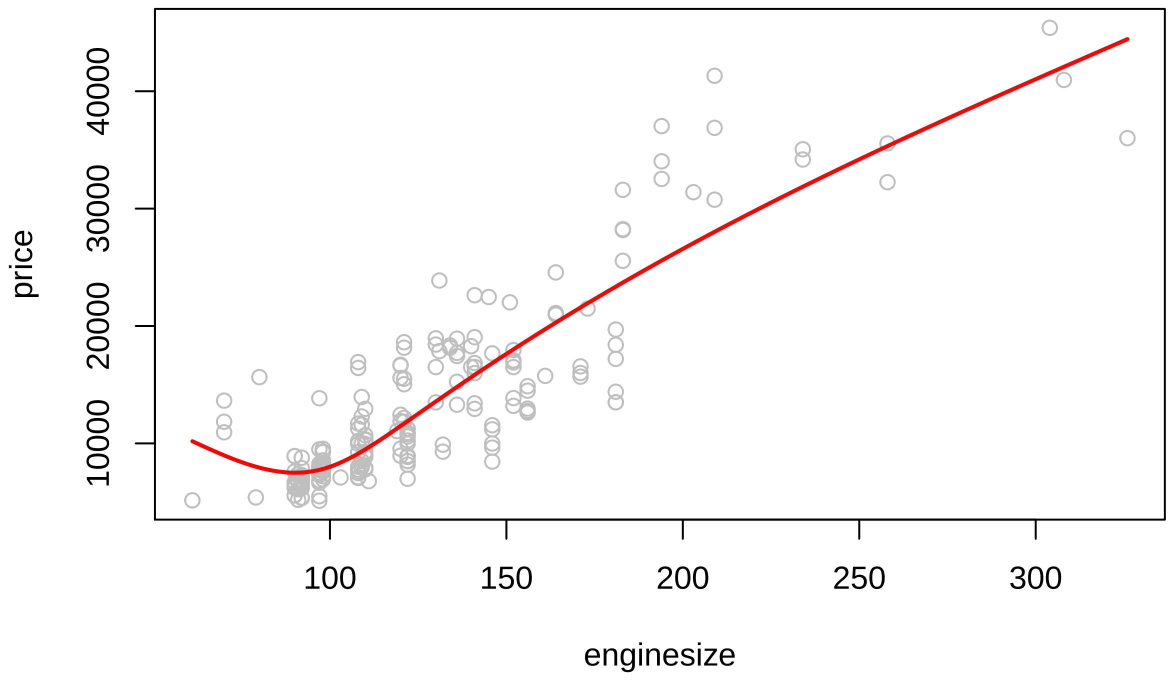
```
## [1] 205   6
```

```r
attr(bs(enginesize, df=6), "knots")
```

```
## 25% 50% 75%
##  97 120 141
```

We see that the r select knots in 97, 120 and 141, near the points we selected in earlier methods.

```r
fit2 = lm(price~ns(enginesize, df=4), data=card)

pred2= predict(fit2, newdata = list(enginesize=engs.grid), se=T)
plot(enginesize, price, col="grey")
lines(engs.grid, pred2$fit, col="red", lwd=2)
```



We fit the previous model using 4 degree of freeedom.

In next experiment we use the smooth spline method.

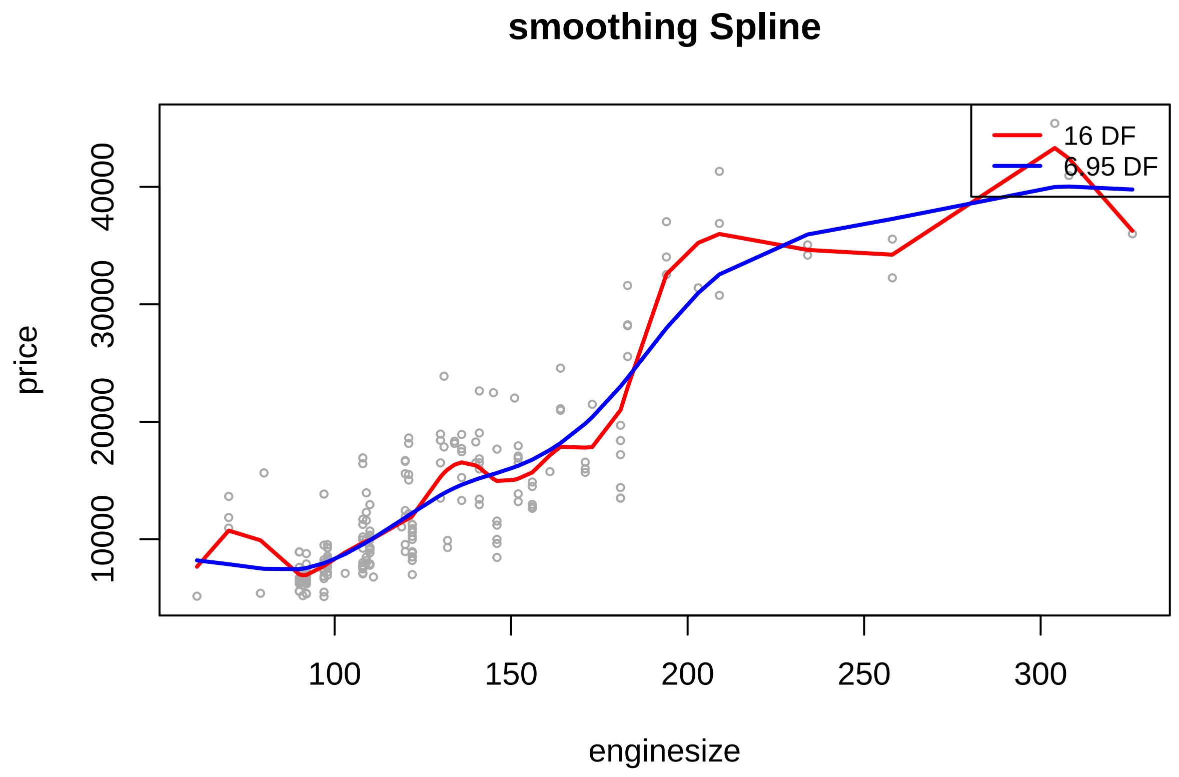
```
plot(enginesize, price, xlim= engsrange, cex=0.5, col="darkgrey")
title("smoothing Spline")
fit = smooth.spline(enginesize, price, df=16)
fit2 = smooth.spline(enginesize, price, cv=TRUE)
```

```
## Warning in smooth.spline(enginesize, price, cv = TRUE): cross-validation with
## non-unique 'x' values seems doubtful
```

```
fit2$df
```

```
## [1] 6.953457
```

```
lines(fit,col="red", lwd=2)
lines(fit2,col="blue", lwd=2)
legend("topright", legend = c("16 DF", "6.95 DF"), col=c("red", "blue"), lty=1, lwd=2, cex=0.8)
```



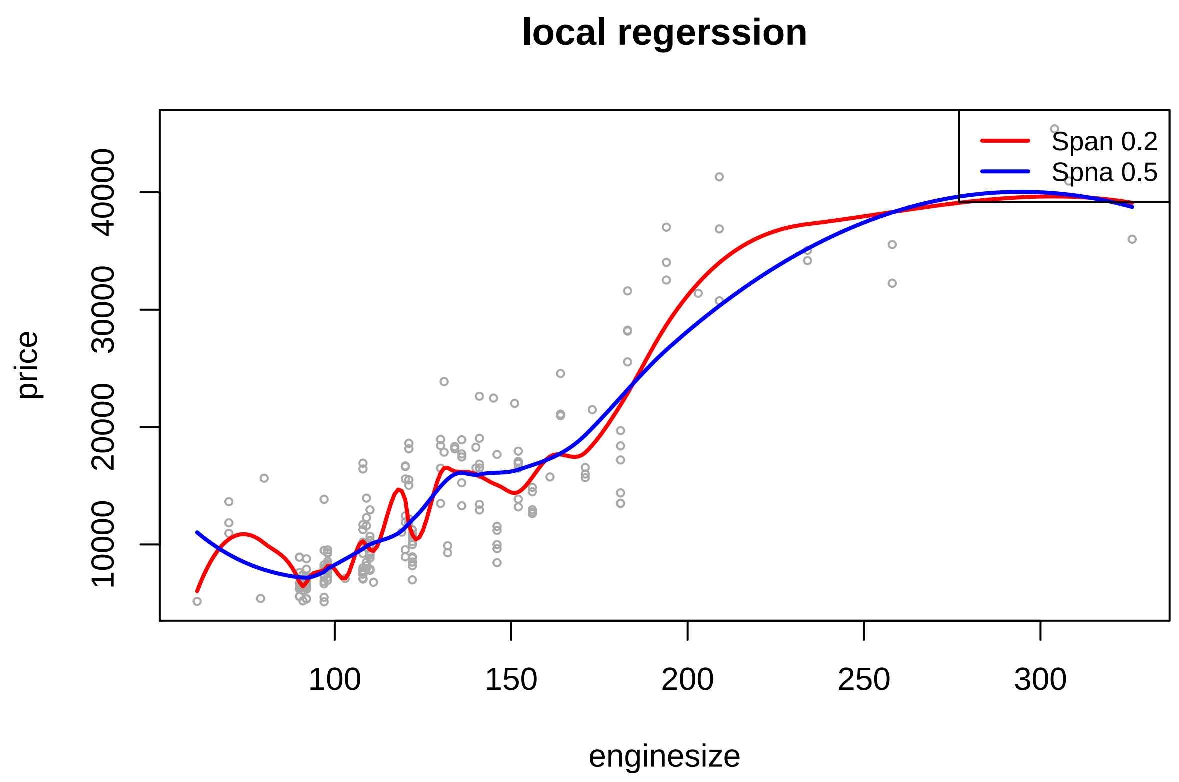
We see the comparison between 16 and 6.5 degree of freeedom. 16 DF model fit the data with high accuracy by taking more wibble form.

```
plot(enginesize, price, xlim = engsrange, cex=.5, col="darkgrey")
title("local regerssion")
fit = loess(price~enginesize, span=.2, data=card)
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at 97
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 5
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 5.0063e-017
```

```
fit2 = loess(price~enginesize, span=.5, data=card)
lines(engs.grid, predict(fit, data.frame(enginesize=engs.grid)), col="red", lwd=2)
lines(engs.grid, predict(fit2, data.frame(enginesize=engs.grid)), col="blue", lwd=2)
legend("topright", legend = c("Span 0.2", "Spna 0.5"), col=c("red", "blue"), lty=1, lwd=2, cex=0.8)
```



We can also select the span parameters to control the model fitness over the training instances

## General additive model (GAM)

Using general additive model we cab combine different methods together.

```
gam1 = lm(price~ns(enginesize, 4)+ns(carwidth,3), data=card)
library(gam)
```

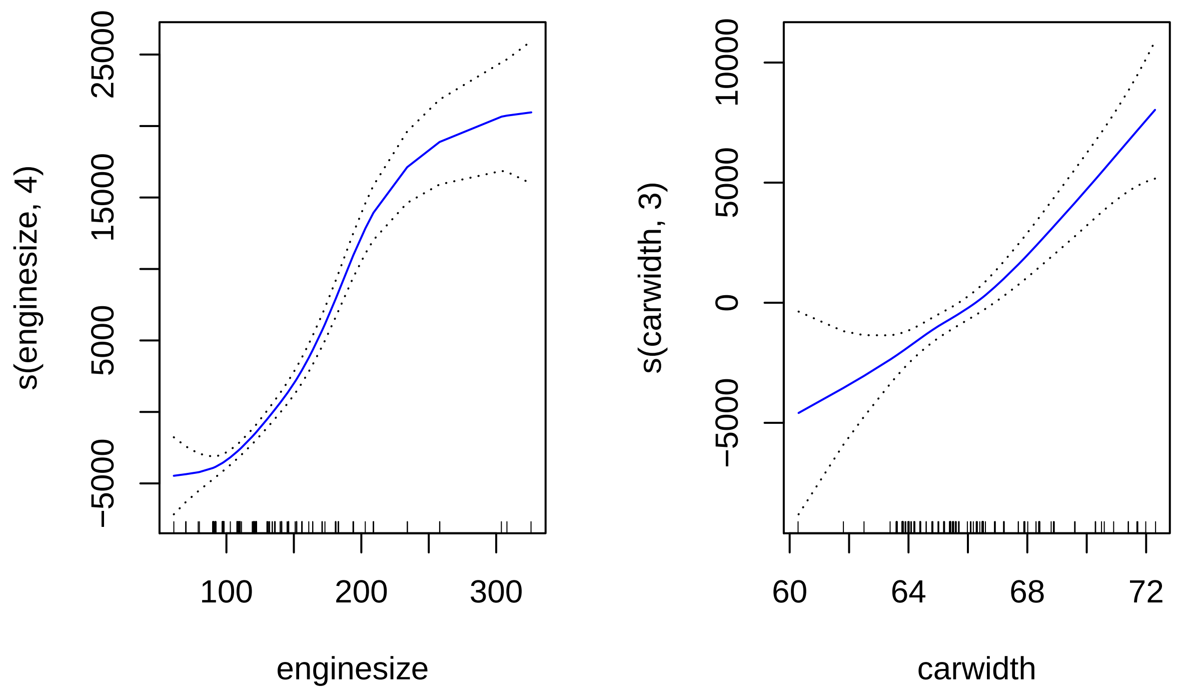
```
## Loading required package: foreach
```

```
## Loaded gam 1.16.1
```

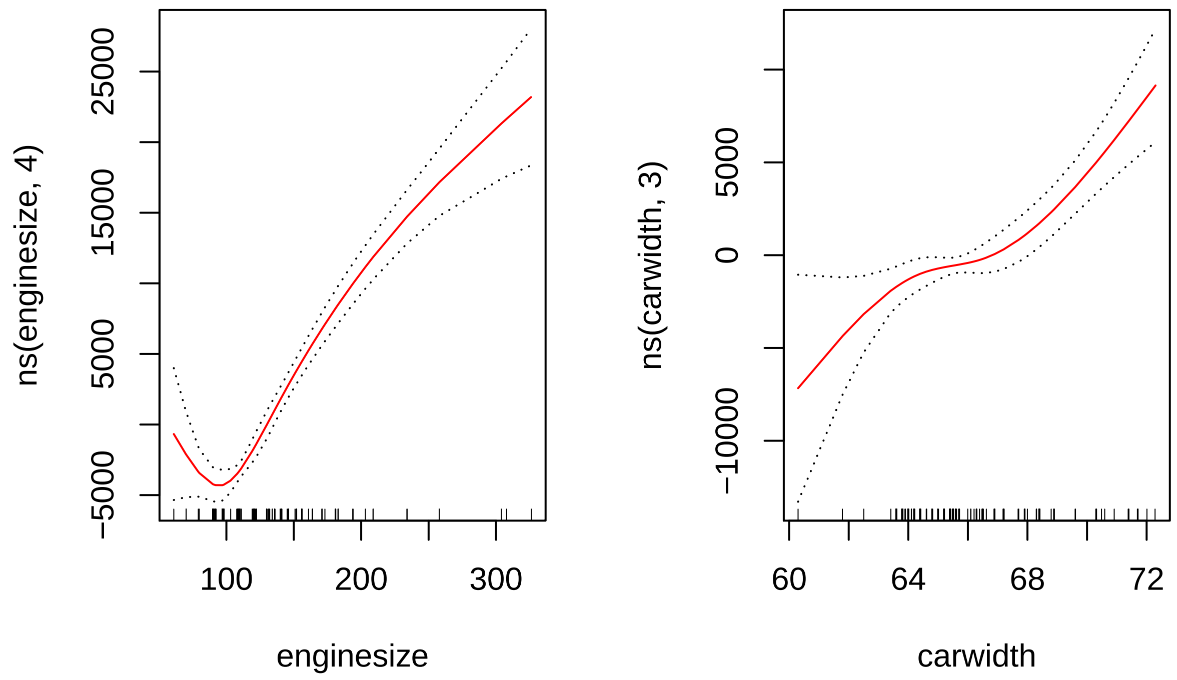
```r
gam.m3 =gam(price~s(enginesize,4)+s(carwidth,3), data=card)
```

```
## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument
## ignored
```

```r
par(mfrow = c(1,2))
plot(gam.m3, se=TRUE, col='blue')
```



```r
par(mfrow = c(1,2))
plot.Gam(gam1, se=TRUE, col='red')
```



The previous two section we implement spline and smooth spline as additive model. We find the fitness and difference in the plot in the boundary regions.

```
gam.m1= gam(price~s(enginesize,4), data=card)
```

```
## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument
## ignored
```

```
gam.m2= gam(price~s(enginesize,4)+carwidth, data=card)
```

```
## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument
## ignored
```

```
anova(gam.m1, gam.m2, gam.m3, test="F")
```

```
## Analysis of Deviance Table
##
## Model 1: price ~ s(enginesize, 4)
## Model 2: price ~ s(enginesize, 4) + carwidth
## Model 3: price ~ s(enginesize, 4) + s(carwidth, 3)
##   Resid. Df Resid. Dev     Df  Deviance       F    Pr(>F)
## 1       200 2576573634
## 2       199 2132406763 1.0000 444166871 41.4787 8.919e-10 ***
## 3       197 2109535663 2.0002  22871101  1.0678    0.3457
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In previous section we define three models (1. Linear and smooth spline 2. add features, linear and smooth spline 3. Smooth spline for two features and linear for the other) and compare the significance of going from one model to another.

The model summary of model are are given below

```r
summary(gam.m3)
```

```
##
## Call: gam(formula = price ~ s(enginesize, 4) + s(carwidth, 3), data = card)
## Deviance Residuals:
##     Min      1Q  Median      3Q     Max
## -7913.5 -1767.3  -656.6  1447.1 13783.1
##
## (Dispersion Parameter for gaussian family taken to be 10708319)
##
##     Null Deviance: 13019639362 on 204 degrees of freedom
## Residual Deviance: 2109535663 on 196.9997 degrees of freedom
## AIC: 3909.844
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##                   Df     Sum Sq    Mean Sq F value    Pr(>F)
## s(enginesize, 4)   1 9870407416 9870407416  921.75 < 2.2e-16 ***
## s(carwidth, 3)     1  492363471  492363471   45.98 1.359e-10 ***
## Residuals        197 2109535663   10708319
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                  Npar Df  Npar F     Pr(F)
## (Intercept)
## s(enginesize, 4)       3 15.1189 6.791e-09 ***
## s(carwidth, 3)         2  1.8634    0.1579
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

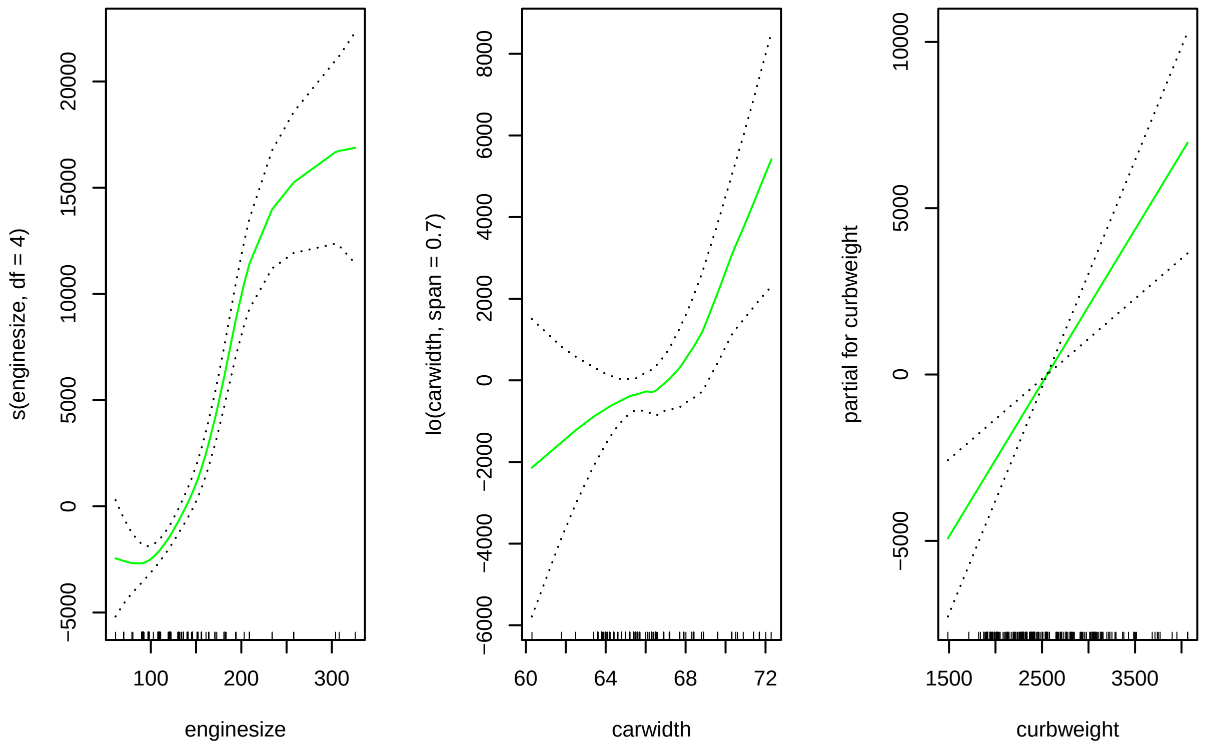
We find the significance of the polynomial features and spline models.

We use the predict and gam library to plot the prediction of the models in the next section with different additive models.

```r
preds =predict(gam.m2, newdata=card)

par(mfrow = c(1,3))
gam.lo= gam(price~s(enginesize, df=4)+lo(carwidth, span = 0.7)
            +curbweight, data=card)
```

```
## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument
## ignored
```

```
plot.Gam(gam.lo, se=TRUE, col="green")
```

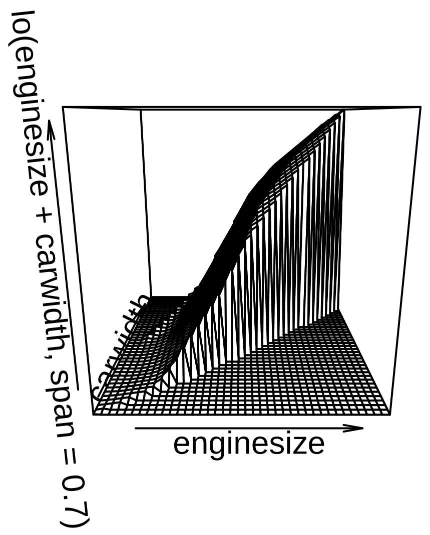


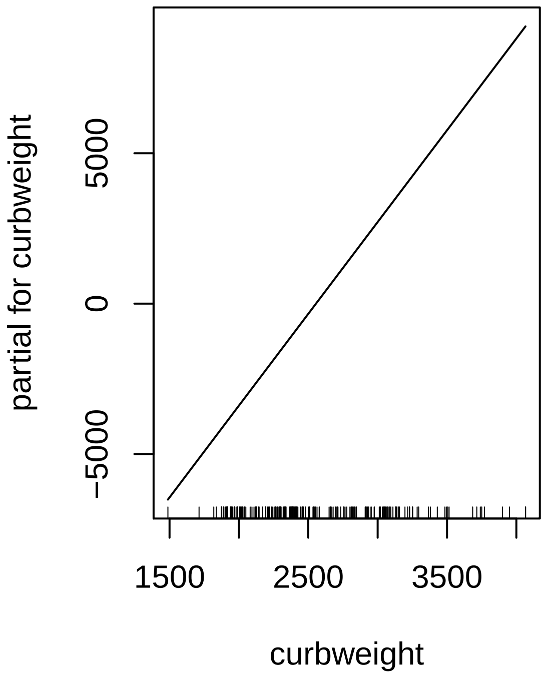
```
gam.lo.i= gam(price~lo(enginesize+carwidth, span = 0.7)
              +curbweight, data=card)
```

```
## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument
## ignored
```

We can use akima two see the two dimension of plot for two variable for the car price dataset against the carwith and enginesize variable.

```
library(akima)
par(mfrow = c(1,2))
plot(gam.lo.i)
```

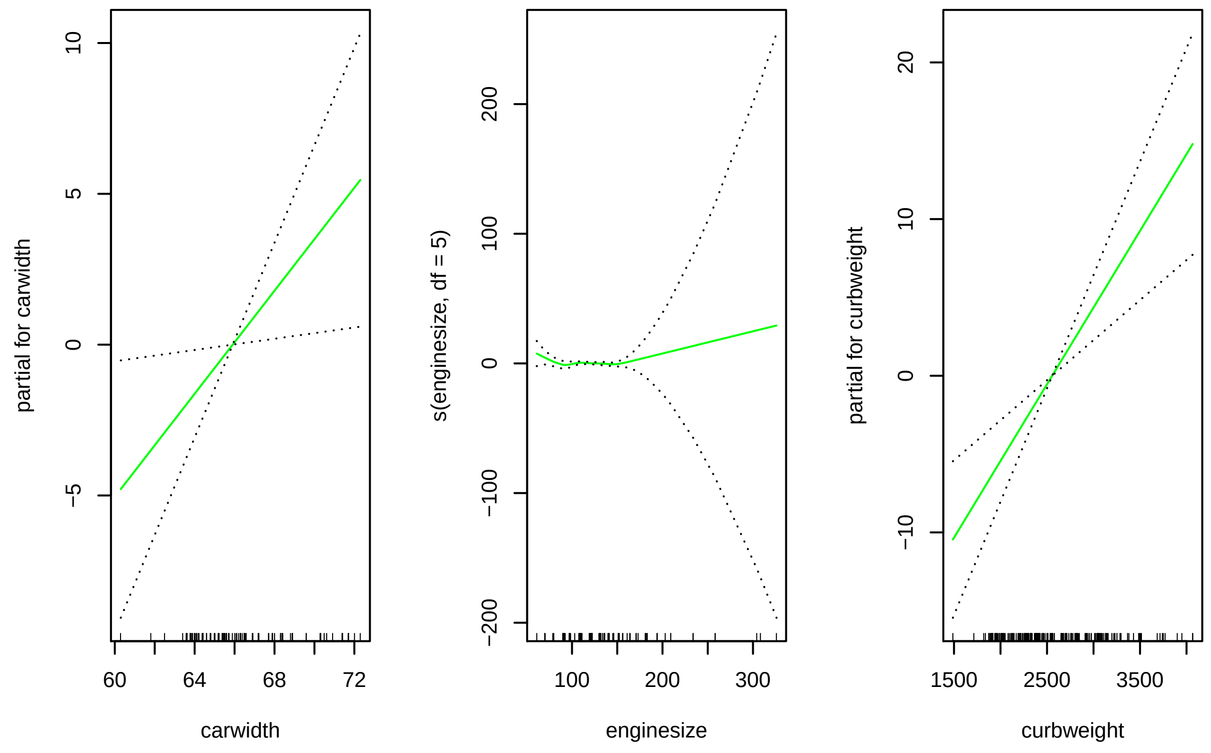




```
gam.lr = gam(I(price>10400)~carwidth+s(enginesize, df=5)
             +curbweight, family = binomial, data=card)
```

```
## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument
## ignored
```

```
par(mfrow = c(1,3))

plot(gam.lr, se=T, col="green")
```



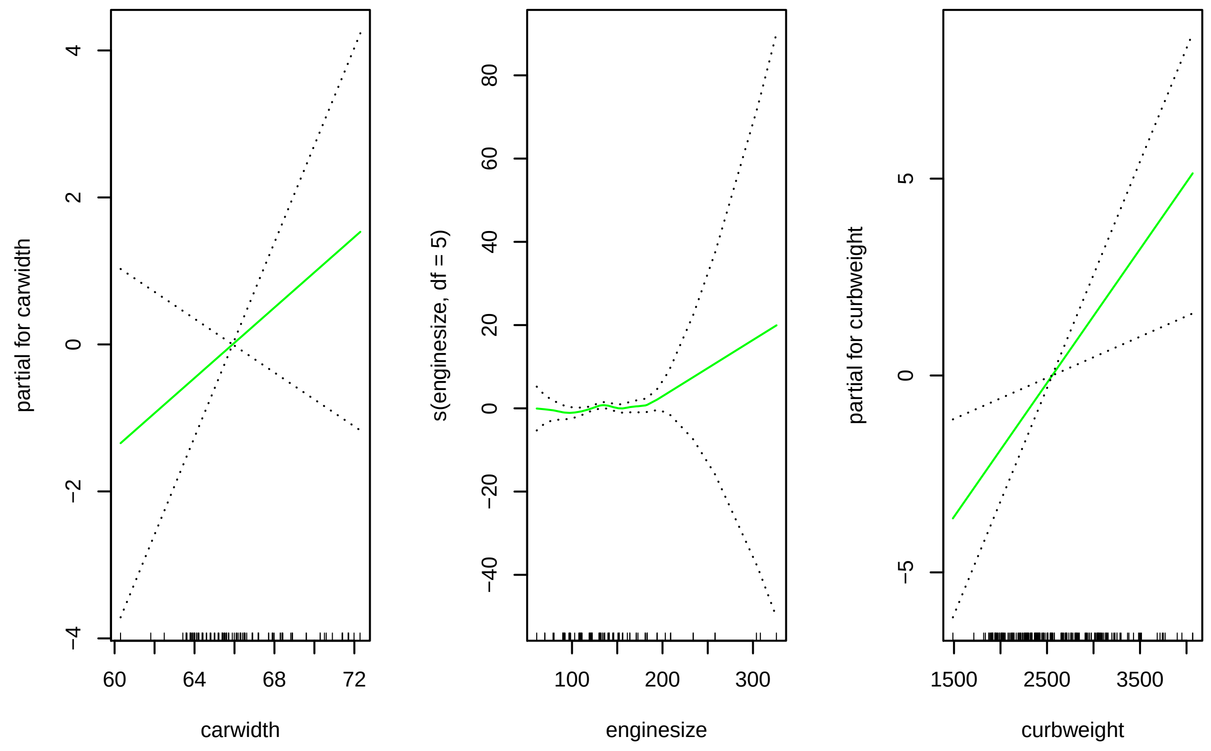
```r
table(curbweight, I(price>15000))
```

```
##
## curbweight FALSE TRUE
##       1488     1    0
##       1713     1    0
##       1819     1    0
##       1837     1    0
##       1874     2    0
##       1876     2    0
##       1889     1    0
##       1890     1    0
##       1900     1    0
##       1905     1    0
##       1909     2    0
##       1918     3    0
##       1938     1    0
##       1940     1    0
##       1944     1    0
##       1945     1    0
##       1950     1    0
##       1951     1    0
##       1956     1    0
##       1967     2    0
##       1971     1    0
##          1985      1      0
##          1989      3      0
##          2004      1      0
##          2008      1      0
##          2010      1      0
##          2015      1      0
##          2017      1      0
##          2024      2      0
##          2028      1      0
##          2037      1      0
##          2040      1      0
##          2050      1      0
##          2081      1      0
##          2094      1      0
##          2109      1      0
##          2120      1      0
##          2122      1      0
##          2128      2      0
##          2140      1      0
##          2145      2      0
##          2169      1      0
##          2190      1      0
##          2191      2      0
##          2204      1      0
##          2209      1      0
##          2212      1      0
##          2221      1      0
##          2236      1      0
##          2240      1      0
##          2254      1      0
##          2261      1      0
##          2264      1      0
##          2265      1      0
##          2275      3      0
##          2280      1      0
##          2289      1      0
##          2290      2      0
##          2293      1      0
##          2300      2      0
##          2302      1      0
##          2304      1      0
##          2319      1      0
##          2324      1      0
##          2326      1      0
##          2328      1      0
##          2337      2      0
##          2340      1      0
##          2365      1      0
##          2370      1      0
##          2372      1      0
##          2380      2      0
##          2385      4      0
##          2395      0      2
##          2403      2      0
##          2405     1     0
##          2410     2     0
##          2414     2     0
##          2420     1     0
##          2425     1     0
##          2443     1     0
##          2455     1     0
##          2458     1     0
##          2460     1     0
##          2465     1     0
##          2480     1     0
##          2500     0     1
##          2507     0     1
##          2510     1     0
##          2535     2     0
##          2536     1     0
##          2540     1     0
##          2548     1     1
##          2551     1     0
##          2563     1     0
##          2579     2     0
##          2650     1     0
##          2658     1     0
##          2661     1     0
##          2670     0     1
##          2679     1     0
##          2695     1     0
##          2700     0     1
##          2707     0     1
##          2710     0     1
##          2714     1     0
##          2734     1     0
##          2756     0     2
##          2758     0     1
##          2765     0     1
##          2778     0     1
##          2800     0     1
##          2808     0     1
##          2811     1     0
##          2818     1     0
##          2823     0     1
##          2824     0     1
##          2833     1     0
##          2844     0     1
##          2847     0     1
##          2910     0     1
##          2912     1     0
##          2921     1     0
##          2926     1     0
##          2935     0     1
##          2952     0     1
##          2954     0     1
##          2975     0     1
##          2976     0     1
##        3012     0    1
##        3016     0    1
##        3020     1    0
##        3034     1    0
##        3042     0    1
##        3045     0    1
##        3049     0    1
##        3053     0    1
##        3055     0    1
##        3060     1    0
##        3062     0    1
##        3071     0    1
##        3075     0    2
##        3086     0    1
##        3095     1    0
##        3110     1    0
##        3130     0    1
##        3131     0    1
##        3139     0    2
##        3151     0    1
##        3157     0    1
##        3197     1    0
##        3217     0    1
##        3230     1    1
##        3252     0    2
##        3285     0    1
##        3296     1    0
##        3366     0    1
##        3380     0    1
##        3430     1    0
##        3485     0    1
##        3495     0    1
##        3505     0    1
##        3515     0    1
##        3685     0    1
##        3715     0    1
##        3740     0    1
##        3750     0    1
##        3770     0    1
##        3900     0    1
##        3950     0    1
##        4066     0    2
```

```
gam.lr.s = gam(I(price>15000)~carwidth+s(enginesize, df=5)
               +curbweight, family = binomial, data=card)
```

```
## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument
## ignored
```

```
plot(gam.lr.s, se =T, col ="green")
```



# Chapter 8

In this section, I will implement different tree based methods on the car dataset.

## Classification Tree

I create class label for the car price by selecting the median as threshold value. I use the tree library to implement the classification tree.

```
library(tree)
high = ifelse(price<=10300, "No", "Yes")

card = data.frame(card, high)

attach(card)
```

```
## The following object is masked _by_ .GlobalEnv:
##
##     high
```

```
## The following objects are masked from card (pos = 8):
##
##     aspiration, boreratio, car_ID, carbody, carheight, carlength,
##     CarName, carwidth, citympg, compressionratio, curbweight,
##     cylindernumber, doornumber, drivewheel, enginelocation, enginesize,
##     enginetype, fuelsystem, fueltype, high, highwaympg, horsepower,
##     peakrpm, price, stroke, symboling, wheelbase

## The following objects are masked from card (pos = 12):
##
##     aspiration, boreratio, car_ID, carbody, carheight, carlength,
##     CarName, carwidth, citympg, compressionratio, curbweight,
##     cylindernumber, doornumber, drivewheel, enginelocation, enginesize,
##     enginetype, fuelsystem, fueltype, high, highwaympg, horsepower,
##     peakrpm, price, stroke, symboling, wheelbase

## The following objects are masked from card (pos = 17):
##
##     aspiration, boreratio, car_ID, carbody, carheight, carlength,
##     CarName, carwidth, citympg, compressionratio, curbweight,
##     cylindernumber, doornumber, drivewheel, enginelocation, enginesize,
##     enginetype, fuelsystem, fueltype, highwaympg, horsepower, peakrpm,
##     price, stroke, symboling, wheelbase

## The following objects are masked from card (pos = 18):
##
##     aspiration, boreratio, car_ID, carbody, carheight, carlength,
##     CarName, carwidth, citympg, compressionratio, curbweight,
##     cylindernumber, doornumber, drivewheel, enginelocation, enginesize,
##     enginetype, fuelsystem, fueltype, highwaympg, horsepower, peakrpm,
##     price, stroke, symboling, wheelbase

## The following objects are masked from card (pos = 20):
##
##     aspiration, boreratio, car_ID, carbody, carheight, carlength,
##     CarName, carwidth, citympg, compressionratio, curbweight,
##     cylindernumber, doornumber, drivewheel, enginelocation, enginesize,
##     enginetype, fuelsystem, fueltype, highwaympg, horsepower, peakrpm,
##     price, stroke, symboling, wheelbase
```

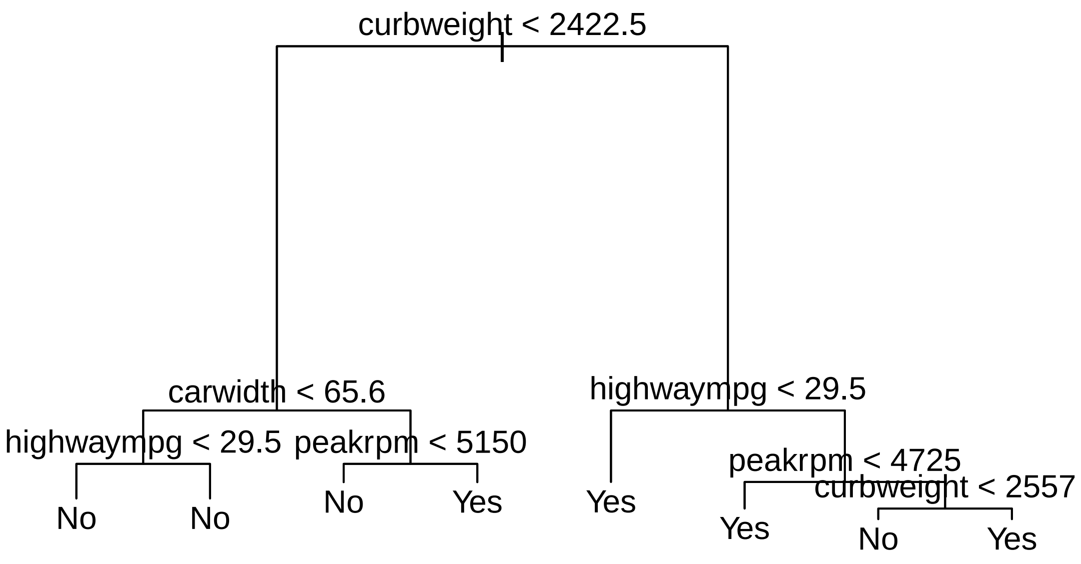
```r
tree.card =  tree(high~fuelsystem+peakrpm+citympg
                  + enginesize+enginetype+carwidth+curbweight+carlength
                  + highwaympg+ boreratio+ stroke + wheelbase + drivewheel
                  + enginelocation+ aspiration+ doornumber+ horsepower+ compressionratio,
                  data = card)


summary(tree.card)
```

```
##
## Classification tree:
## tree(formula = high ~ fuelsystem + peakrpm + citympg + enginesize +
##     enginetype + carwidth + curbweight + carlength + highwaympg +
##     boreratio + stroke + wheelbase + drivewheel + enginelocation +
##     aspiration + doornumber + horsepower + compressionratio,
##     data = card)
## Variables actually used in tree construction:
## [1] "curbweight" "carwidth"   "highwaympg" "peakrpm"
## Number of terminal nodes:  8
## Residual mean deviance:  0.17 = 33.5 / 197
## Misclassification error rate: 0.03902 = 8 / 205
```

From the summary, we see the performance of the tree, it misclassified 8 instances in the training examples.

We observe the graphical representation of the tree in the following section.

```r
plot(tree.card)
text(tree.card, pretty= 0)
```



We see the classified tree via the classifier method. It selected curbweight as 1st label feaures. We can also get the description by following code

```r
tree.card
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 205 284.200 No ( 0.50244 0.49756 )
##    2) curbweight < 2422.5 104  65.840 No ( 0.90385 0.09615 )
##      4) carwidth < 65.6 92  26.440 No ( 0.96739 0.03261 )
##        8) highwaympg < 29.5 9  11.460 No ( 0.66667 0.33333 ) *
##        9) highwaympg > 29.5 83   0.000 No ( 1.00000 0.00000 ) *
##      5) carwidth > 65.6 12  16.300 Yes ( 0.41667 0.58333 )
##       10) peakrpm < 5150 7   8.376 No ( 0.71429 0.28571 ) *
##       11) peakrpm > 5150 5   0.000 Yes ( 0.00000 1.00000 ) *
##    3) curbweight > 2422.5 101  60.700 Yes ( 0.08911 0.91089 )
##      6) highwaympg < 29.5 79   0.000 Yes ( 0.00000 1.00000 ) *
##      7) highwaympg > 29.5 22  29.770 Yes ( 0.40909 0.59091 )
##       14) peakrpm < 4725 8   0.000 Yes ( 0.00000 1.00000 ) *
##       15) peakrpm > 4725 14  18.250 No ( 0.64286 0.35714 )
##         30) curbweight < 2557 8   6.028 No ( 0.87500 0.12500 ) *
##         31) curbweight > 2557 6   7.638 Yes ( 0.33333 0.66667 ) *
```

The above result shows the rule of tree classifier for classifying car price as high or low.

Now I use the validaton set to test the model performance. The model is trained only on the training instances.

```
set.seed(2)

train =  sample(1:nrow(card), 150)
card.test  =  card[-train,]
high.test =  high[-train]
tree.card = tree(high~fuelsystem+peakrpm+citympg
                 + enginesize+enginetype+carwidth+curbweight+carlength
                 + highwaympg+ boreratio+ stroke + wheelbase + drivewheel
                 + enginelocation+ aspiration+ doornumber+ horsepower+ compressionratio,
                 data = card, subset=train)

tree.pred = predict(tree.card, card.test, type = "class")

table(tree.pred, high.test)
```

```
##          high.test
## tree.pred No Yes
##       No  25   4
##       Yes  1  25
```

From the table above we find that the classifier was able to correctly classify 50 instances out of 54 test instances. It was trained on 150 training examples.

```
set.seed(3)

cv.card = cv.tree(tree.card, FUN=prune.misclass)
names(cv.card)
```

```
## [1] "size"   "dev"    "k"      "method"
```

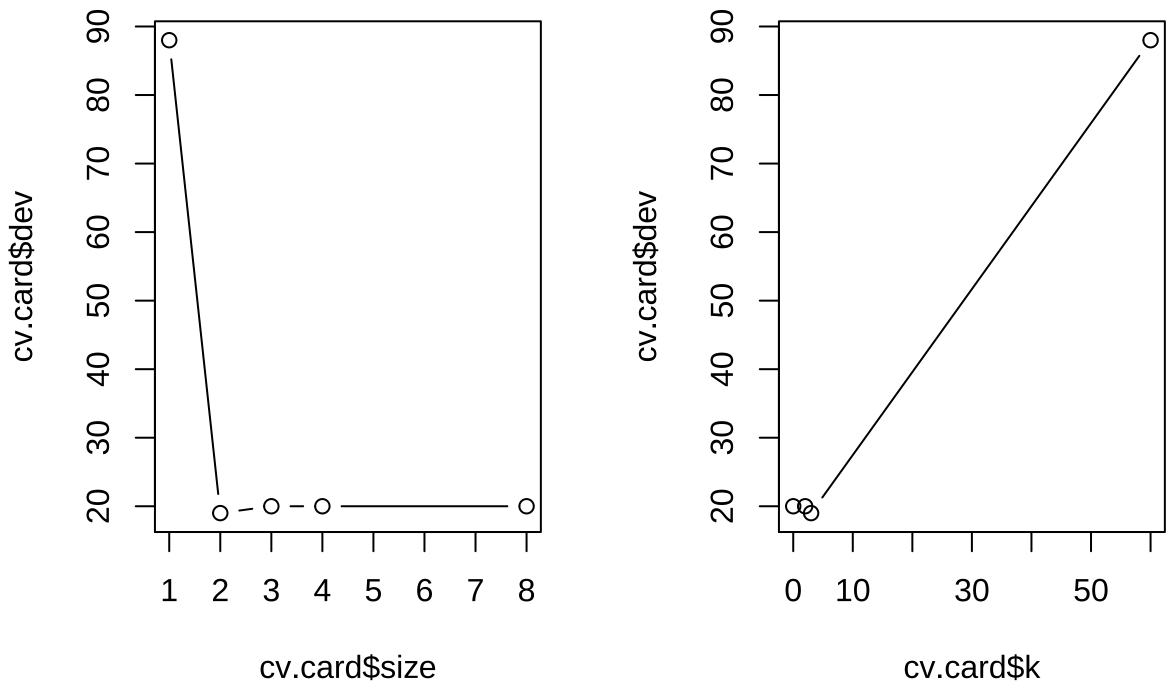
```
cv.card
```

```
## $size
## [1] 8 4 3 2 1
##
```

```
## $dev
## [1] 20 20 20 19 88
##
## $k
## [1] -Inf    0    2    3   60
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

The dev corresponds to misclassification. For 2 the misclassification rate is minimum for the tree method.

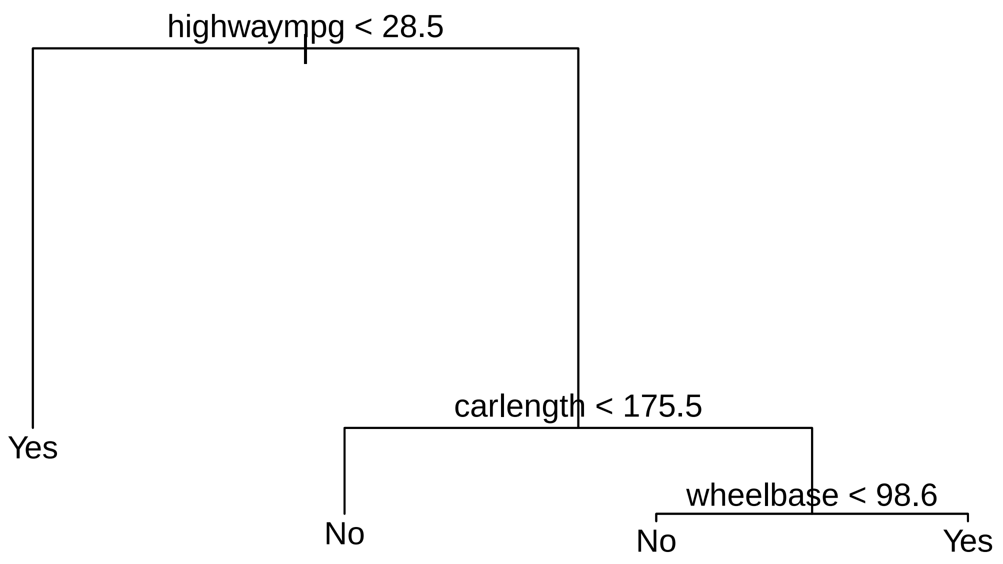
```
par(mfrow= c(1,2))
plot(cv.card$size, cv.card$dev, type="b")
plot(cv.card$k, cv.card$dev, type="b")
```



The result shows that with tree size the misclassification decreases.

We now create a 4 node tree using prune missclassification.

```
prune.card = prune.misclass(tree.card, best=4)
plot(prune.card)
text(prune.card, pretty = 0)
```



```r
tree.pred = predict(prune.card, card.test, type = "class")
table(tree.pred, high.test)
```

```
##          high.test
## tree.pred No Yes
##       No  24   4
##       Yes  2  25
```

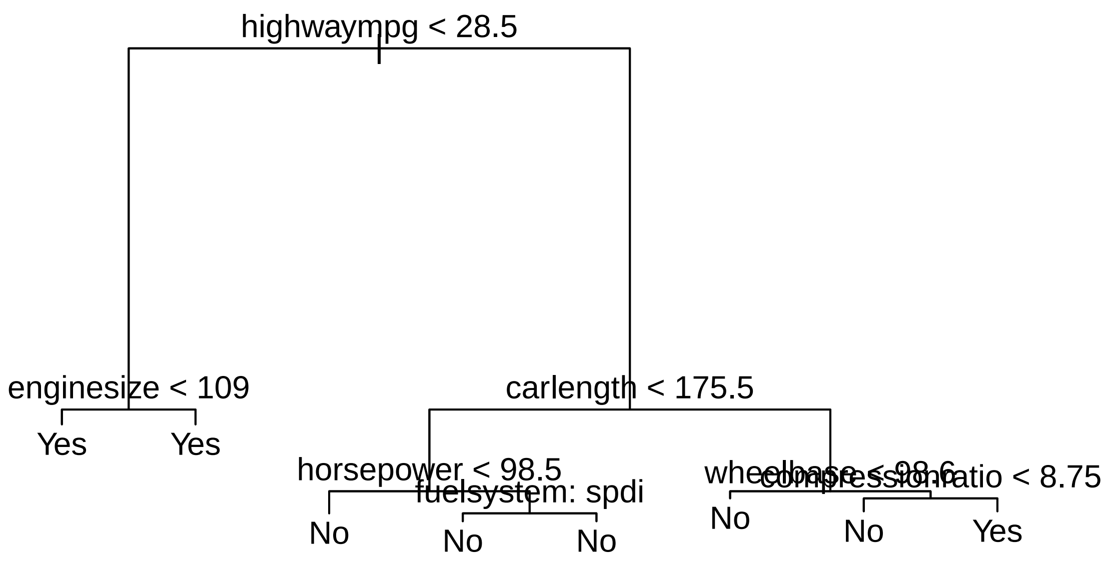
The previous result shows 6 misclassification result on the validation set. The accuracy is 89%.

```r
(24+25)/(26+29)
```

```
## [1] 0.8909091
```

Now we fit the tree model for different tree size and check model performance. I use more tree label than earlier example.

```r
prune.card = prune.misclass(tree.card, best=6)
plot(prune.card)
text(prune.card, pretty = 0)
```



```r
tree.pred = predict(prune.card, card.test, type = "class")
table(tree.pred, high.test)
```

```
##          high.test
## tree.pred No Yes
##       No  26   5
##       Yes  0  24
```

We see than the error has decrease in the result as we have used bigger tree.

### Fitting regression trees

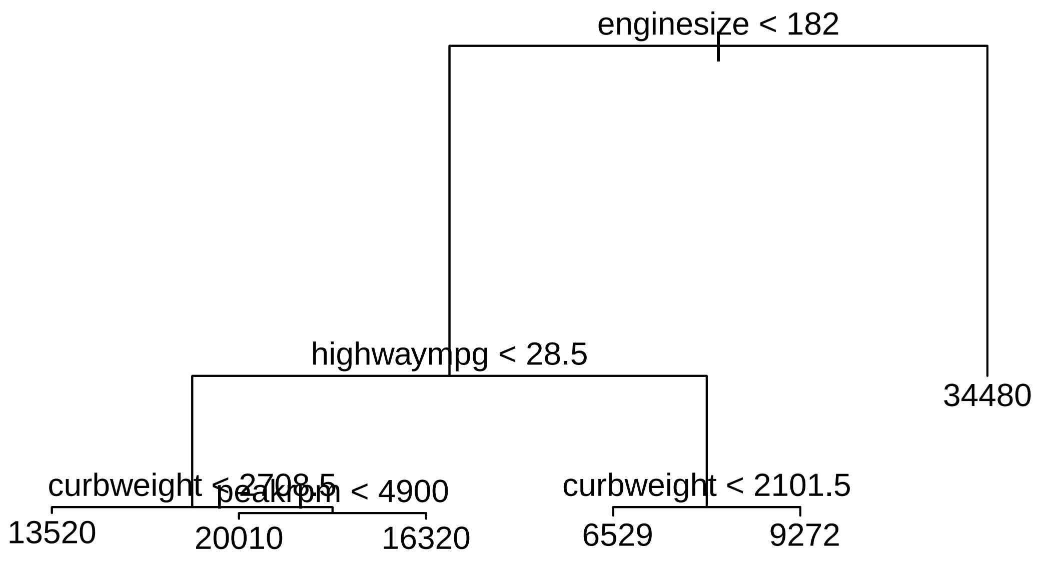
Similar to classifier, we first fit the model,

```r
set.seed(1)

train = sample(1:nrow(card), nrow(card)/2)
tree.card = tree(price~fuelsystem+peakrpm+citympg
                 + enginesize+enginetype+carwidth+curbweight+carlength
                 + highwaympg+ boreratio+ stroke + wheelbase + drivewheel
                 + enginelocation+ aspiration+ doornumber+ horsepower+ compressionratio,
                 data = card, subset=train)

summary(tree.card)
```

```
##
## Regression tree:
## tree(formula = price ~ fuelsystem + peakrpm + citympg + enginesize +
##     enginetype + carwidth + curbweight + carlength + highwaympg +
##     boreratio + stroke + wheelbase + drivewheel + enginelocation +
##     aspiration + doornumber + horsepower + compressionratio,
##     data = card, subset = train)
## Variables actually used in tree construction:
## [1] "enginesize" "highwaympg" "curbweight" "peakrpm"
## Number of terminal nodes:  6
## Residual mean deviance:  6695000 = 642700000 / 96
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -8926.00 -1193.00   -19.28     0.00  1227.00 10920.00
```

In summary the model finds four veriables to decide the car price. It created 6 nodes tree. The graphical structure is as follow;

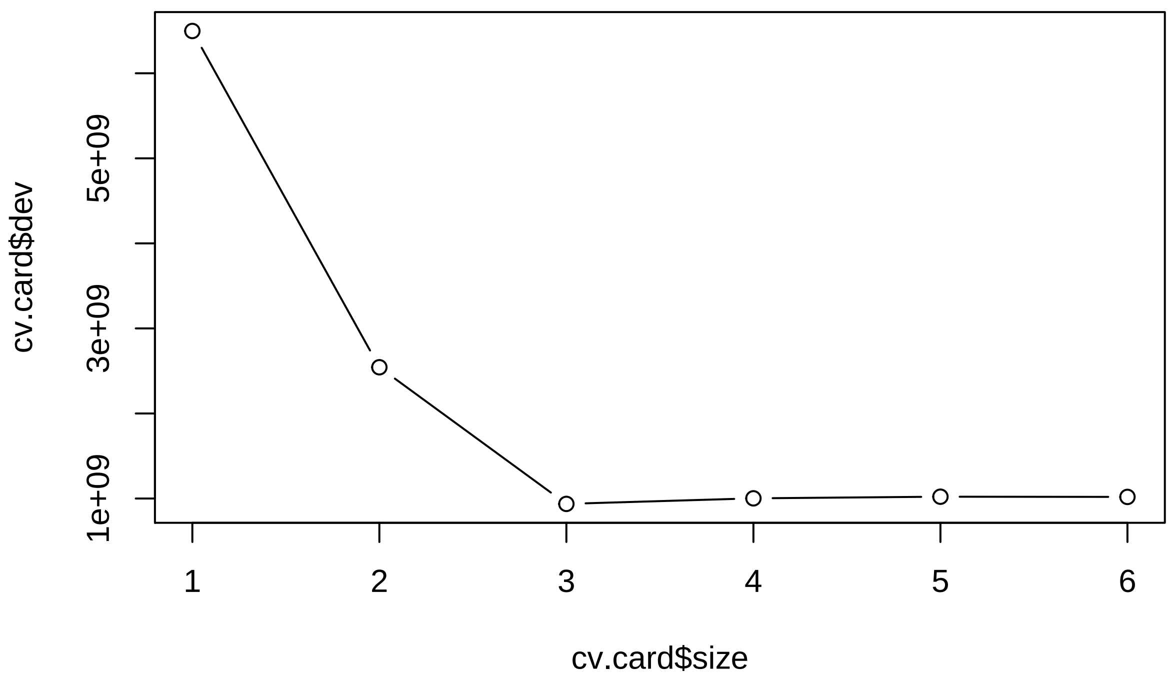
```r
plot(tree.card)
text(tree.card, pretty = 0)
```



In above figure, we see the 6 node tree found by the model.

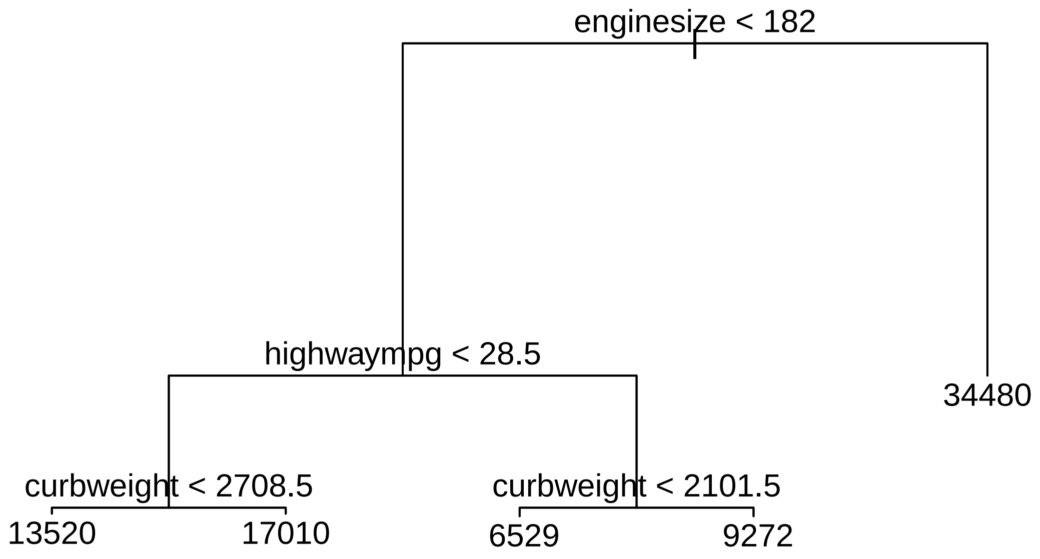
We apply pruning to check model performance across the tree size.

```r
cv.card = cv.tree(tree.card)
plot(cv.card$size, cv.card$dev, type ="b")
```



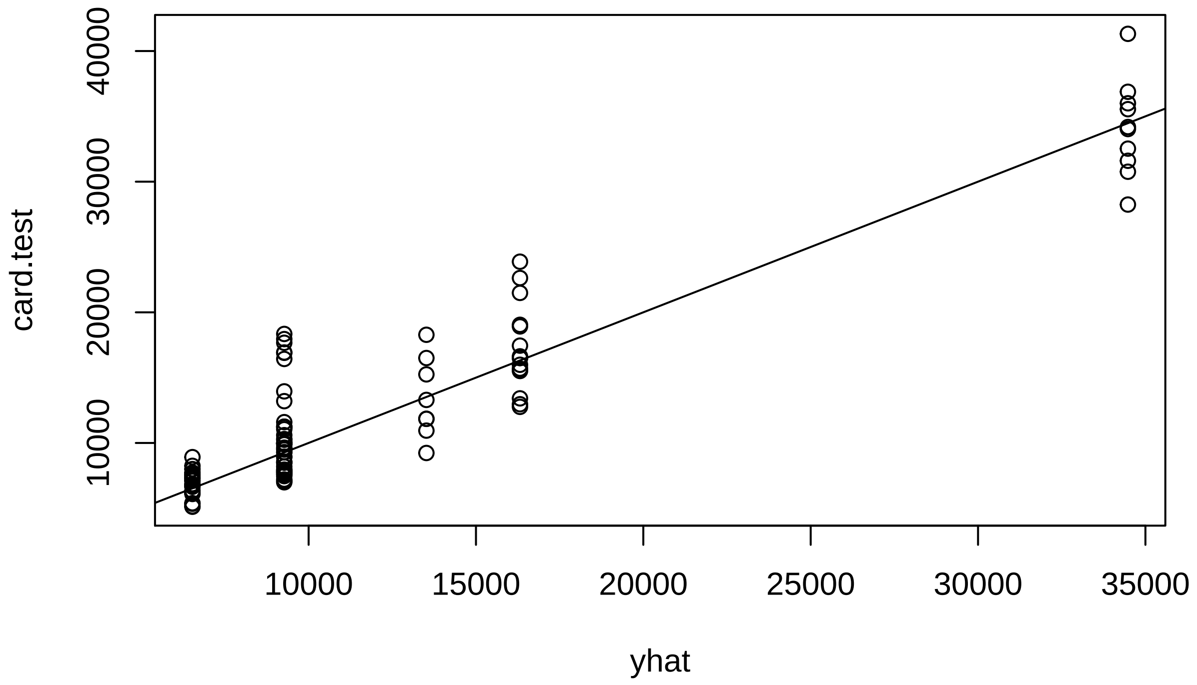
The above graph shows tree size vs error graph for the car price using regression tree.

```r
prune.card = prune.tree(tree.card, best=5)
plot(prune.card)
text(prune.card, pretty = 0)
```



We controlled the tree size by specifing 5. The plot shows the estimated 5 node tree for the car price dataset.

```
yhat = predict(tree.card, newdata= card[-train,])
card.test = card[-train, "price"]
plot(yhat, card.test)
abline(0,1)
```



```r
mean((yhat-card.test)^2)
```

```
## [1] 8637294
```

In last example, we see the tree performance using the cross-validation approach. We find the final error on test set is 8637294.

In next we will see the bagging and random forest and compare the error result.

## Bagging and Random forest

In this part we will apply bagging and random forest by randomforest function.

### Bagging

The bagging is special case of random forest with considering all the features at a time.

```r
library(randomForest)
```

```
## randomForest 4.6-14
```

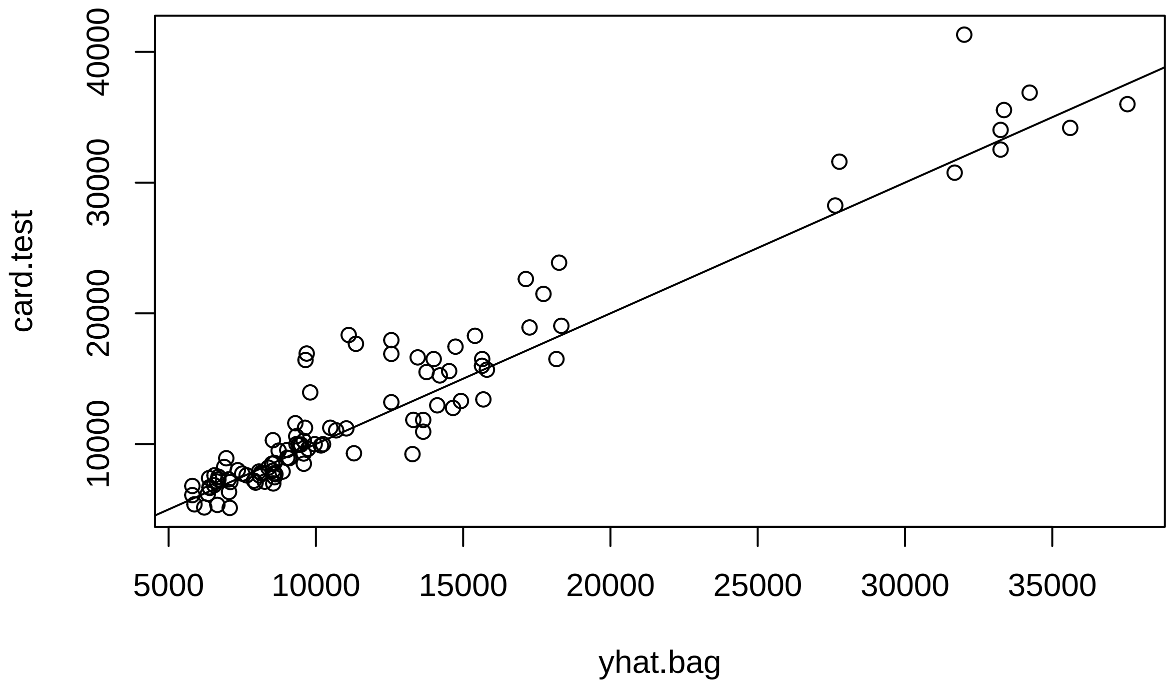
```
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(1)
bag.card = randomForest(price~fuelsystem+peakrpm+citympg
                        + enginesize+enginetype+carwidth+curbweight+carlength
                        + highwaympg+ boreratio+ stroke + wheelbase + drivewheel
                        + enginelocation+ aspiration+ doornumber+ horsepower+ compressionratio,
                        data = card, subset=train, mtry =18, importance =TRUE)
bag.card
```

```
##
## Call:
##  randomForest(formula = price ~ fuelsystem + peakrpm + citympg +      enginesize + enginetype + carw
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 18
##
##           Mean of squared residuals: 6074559
##                     % Var explained: 90.13
```

We specified mtry = 18, as we have 18 features for the car price dataset. The random forest follows bagging approach for estimation.

```
yhat.bag = predict(bag.card, newdata = card[-train,])
plot(yhat.bag, card.test)
abline(0,1)
```



```r
mean((yhat.bag - card.test)^2)
```

```
## [1] 5694545
```

In the cross-validation approach we observe the error of 5694545. This is smaller than tree regression method since bagging combines result different regressor.

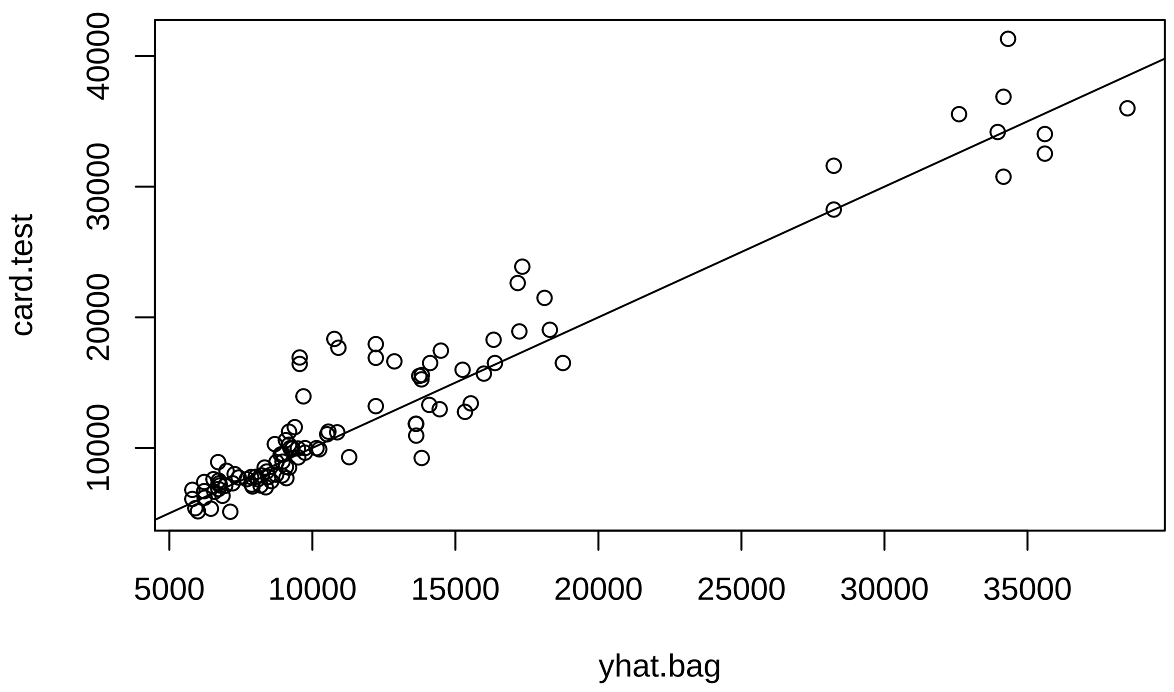
We can also control the number of tree in bagging and and check its performance on the test result.

```r
set.seed(1)
bag.card = randomForest(price~fuelsystem+peakrpm+citympg
                        + enginesize+enginetype+carwidth+curbweight+carlength
                        + highwaympg+ boreratio+ stroke + wheelbase + drivewheel
                        + enginelocation+ aspiration+ doornumber+ horsepower+ compressionratio,
                        data = card, subset=train, mtry =18, ntree = 25)
bag.card
```

```
##
## Call:
##  randomForest(formula = price ~ fuelsystem + peakrpm + citympg +      enginesize + enginetype + carw
##                Type of random forest: regression
##                      Number of trees: 25
## No. of variables tried at each split: 18
##
##           Mean of squared residuals: 5560265
##                     % Var explained: 90.97
```

We evaluate the bagging model on the validation dataset.

```r
yhat.bag = predict(bag.card, newdata = card[-train,])
plot(yhat.bag, card.test)
abline(0,1)
```



```r
mean((yhat.bag - card.test)^2)
```

```
## [1] 6000532
```

We find the final error is 6000532, similar to earlier bagging method but smaller than decision tree regression method.

**Random forest**

By controlling mtry parameters we implement random forest over the car dataset. I used 18/3 = 6 features for the car data.

```r
set.seed(1)
rf.card = randomForest(price~fuelsystem+peakrpm+citympg
                         + enginesize+enginetype+carwidth+curbweight+carlength
                         + highwaympg+ boreratio+ stroke + wheelbase + drivewheel
                         + enginelocation+ aspiration+ doornumber+ horsepower+ compressionratio,
                         data = card, subset=train, mtry = 6, importance=TRUE)
yhat.rf = predict(rf.card, newdata = card[-train, ])
mean((yhat.rf - card.test)^2)
```

```
## [1] 5559902
```

Using random forest we find the error of 5559902 comparable to the bagging method by using 6 features at maximum each time. The random forest also better estimates than the decision tree method.

We can also observe the variable importance in the random forest in r.

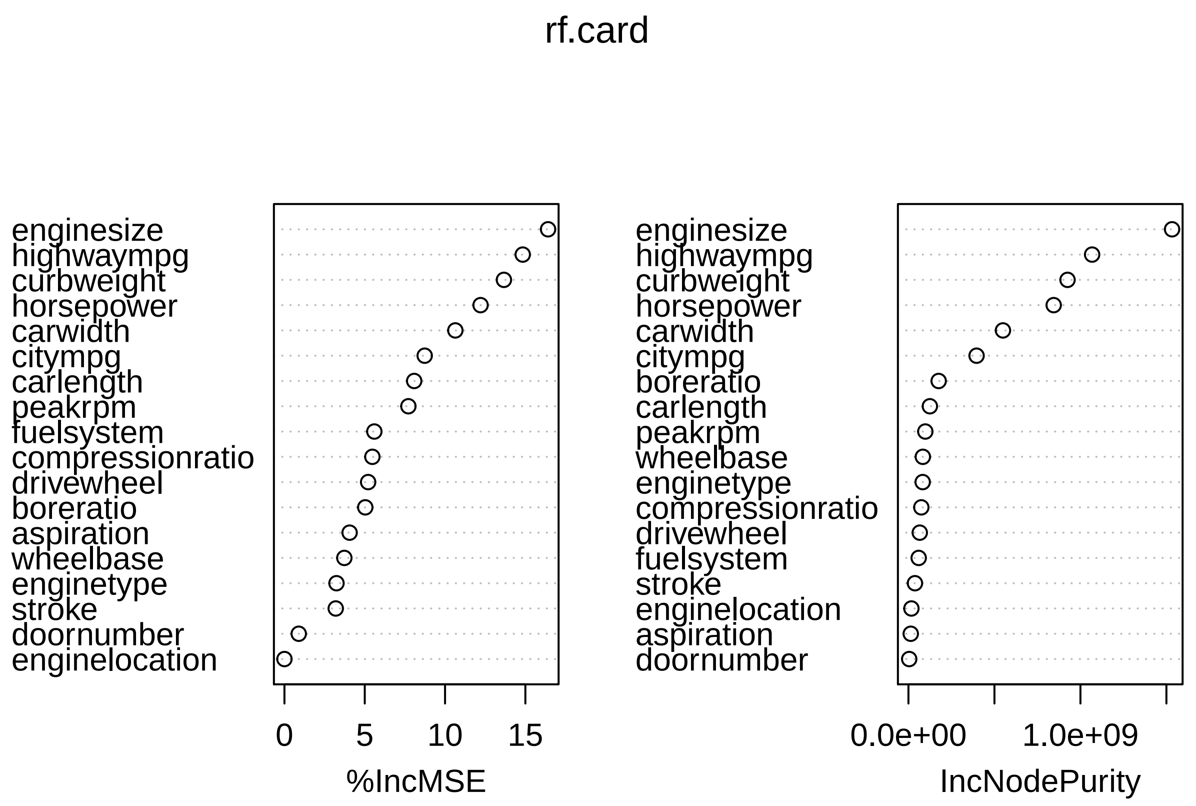
```r
importance(rf.card)
```

```
##                     %IncMSE IncNodePurity
## fuelsystem        5.5964884      59649716
## peakrpm           7.7166068      98176631
## citympg           8.7319301     396102906
## enginesize       16.4120100    1532751906
## enginetype        3.2414265      82863997
## carwidth         10.6398536     549513349
## curbweight       13.6639477     925056694
## carlength         8.0778921     124732748
## highwaympg       14.8362882    1067921018
## boreratio         5.0328482     175789746
## stroke            3.1947744      37816082
## wheelbase         3.7281764      83666728
## drivewheel        5.2116266      65821154
## enginelocation    0.0000000      17265212
## aspiration        4.0570564      13486013
## doornumber        0.8913836       4503901
## horsepower       12.2105869     844513965
## compressionratio  5.4739152      74924174
```

From the importance we see that, enginesize is the most important feature for the random forest. Since enginesize gets maximum value in the random forest approach.

We can also plot their respective importance.

```r
varImpPlot(rf.card)
```



We can see the previous result in the plot in this section. The enginesize gets chosen as the best important feature.

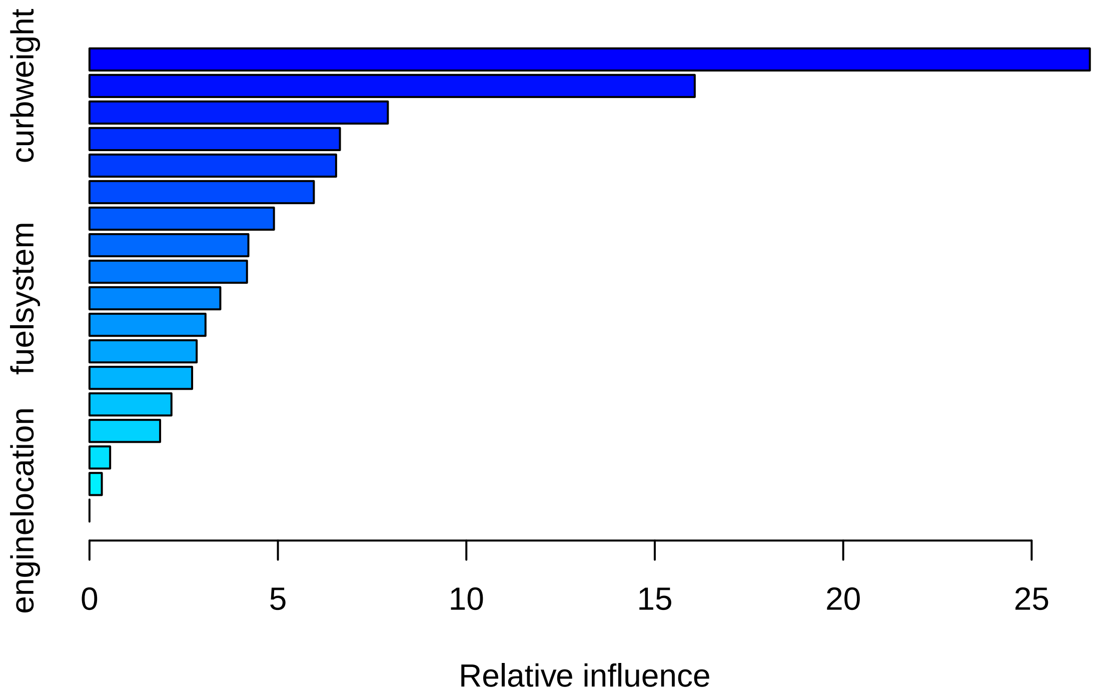
## boosting

We will use gbm package for applying boosting over the car price dataset. In the r we can selection the interaction option in boosting method. In my experiment, I have chosen 6.

```
library(gbm)
```

```
## Loaded gbm 2.1.5
```

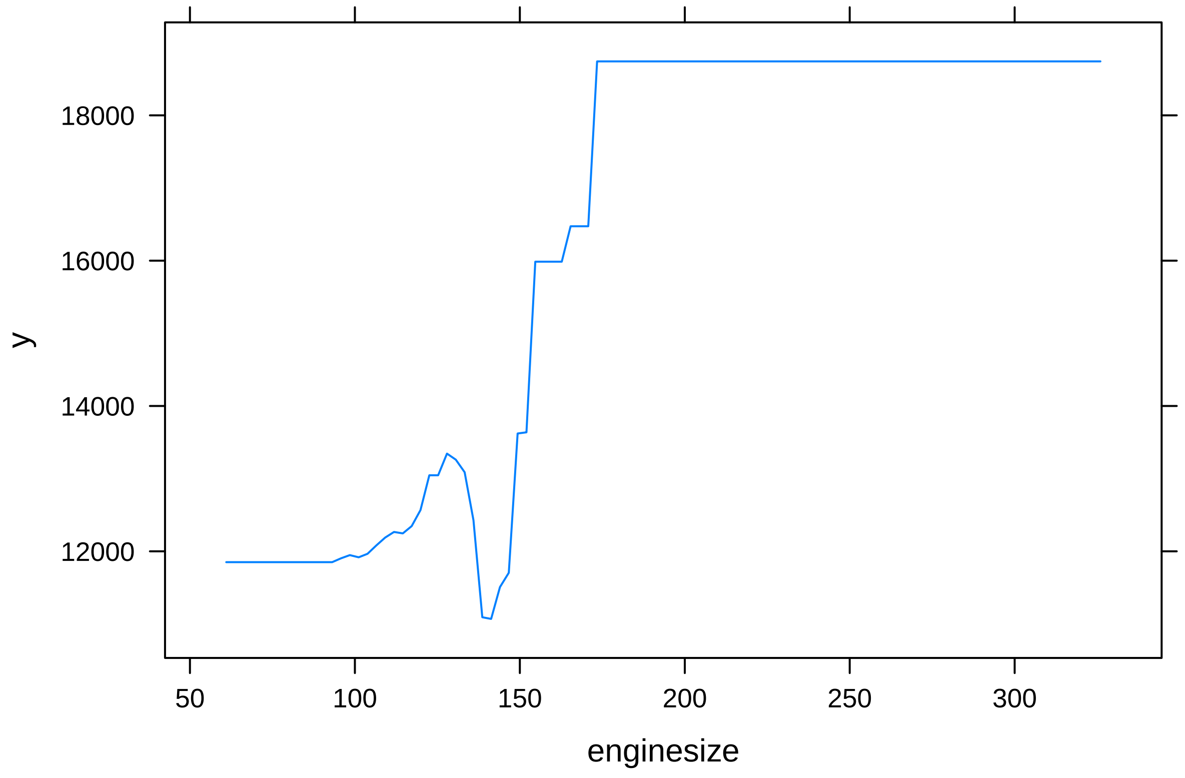
```
set.seed(1)
boost.card = gbm(price~fuelsystem+peakrpm+citympg
                 + enginesize+enginetype+carwidth+curbweight+carlength
                 + highwaympg+ boreratio+ stroke + wheelbase + drivewheel
                 + enginelocation+ aspiration+ doornumber+ horsepower+ compressionratio,
                 data = card[-train,], distribution = "gaussian", n.trees = 5000,
                 interaction.depth = 6)
summary(boost.card)
```



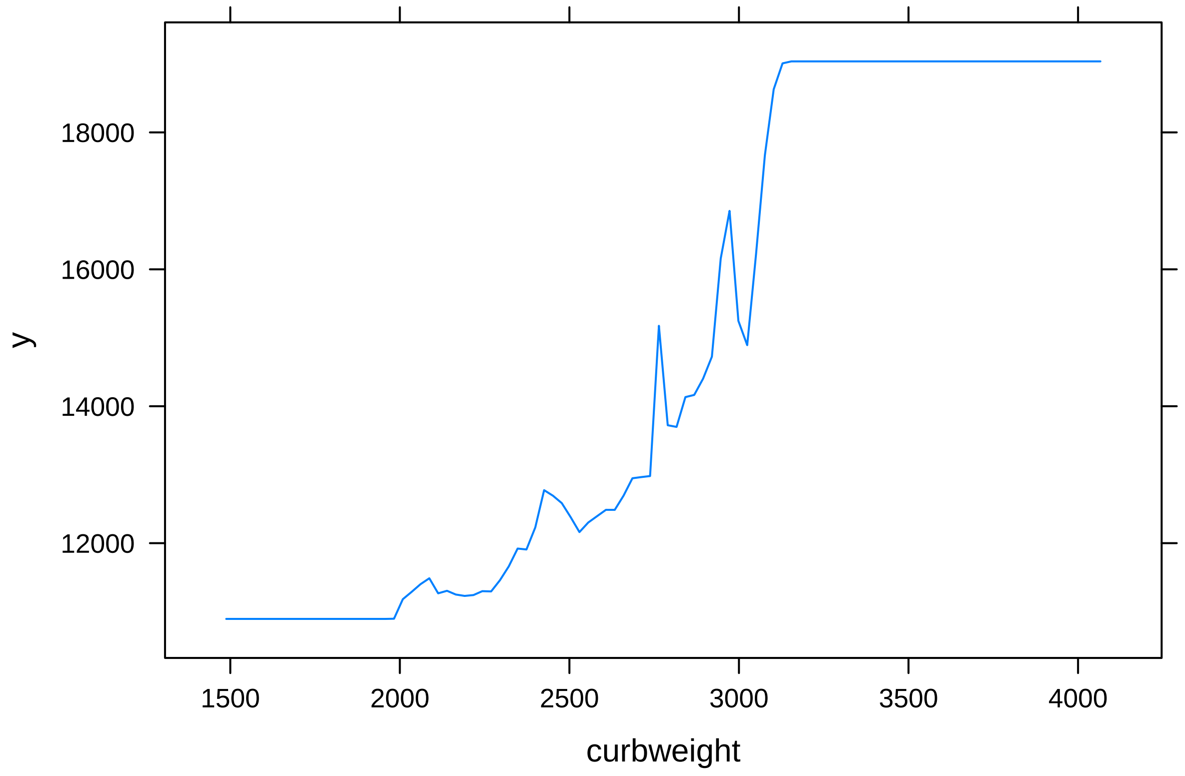
```
##                                 var    rel.inf
## enginesize             enginesize 26.5446690
## curbweight             curbweight 16.0603711
## highwaympg             highwaympg  7.9170659
## carwidth                 carwidth  6.6460265
## carlength               carlength  6.5441080
## horsepower             horsepower  5.9538873
## wheelbase               wheelbase  4.8950393
## boreratio               boreratio  4.2160147
## citympg                   citympg  4.1796560
## fuelsystem             fuelsystem  3.4713013
## enginetype             enginetype  3.0789409
## compressionratio compressionratio  2.8443091
## stroke                     stroke  2.7229956
## peakrpm                   peakrpm  2.1756799
## drivewheel             drivewheel  1.8741826
## doornumber             doornumber  0.5477344
## aspiration             aspiration  0.3280184
## enginelocation     enginelocation  0.0000000
```

We again see that the most important feature in boosting is enginesize as it gets most rel.inf parameter of 26.5447

```
par(mfrow = c(1,2))
plot(boost.card, i ="enginesize")
```



```
plot(boost.card, i= "curbweight")
```



In above plots we see the car price estimation based on the enginesize and curbweight predictors.

```
yhat.boost =  predict(boost.card, newdata = card[-train,], n.trees = 5000)
mean((yhat.boost - card.test)^2)
```

```
## [1] 215913.8
```

We estimate the cross validation error for boosting method and found the value of 215913, smaller than both the random forest and decision trees. The tree performed better in the car price than the linear methods.

Finally we experiment with the shrinkage parameter lambda.

```
boost.card = gbm(price~fuelsystem+peakrpm+citympg
                 + enginesize+enginetype+carwidth+curbweight+carlength
                 + highwaympg+ boreratio+ stroke + wheelbase + drivewheel
                 + enginelocation+ aspiration+ doornumber+ horsepower+ compressionratio,
                 data = card[-train,], distribution = "gaussian", n.trees = 5000,
                 interaction.depth = 4, shrinkage = 0.2, verbose = F)
yhat.boost =  predict(boost.card, newdata = card[-train,], n.trees = 5000)
mean((yhat.boost - card.test)^2)
```

```
## [1] 209760
```

By tuning lambda, we get a little better result (209460) in the cross validation data compared to earlier boosting cross-validation result (215913.8).

# Chapter 9

To apply the support vector classifier and sVM we will use e1071 library in R.

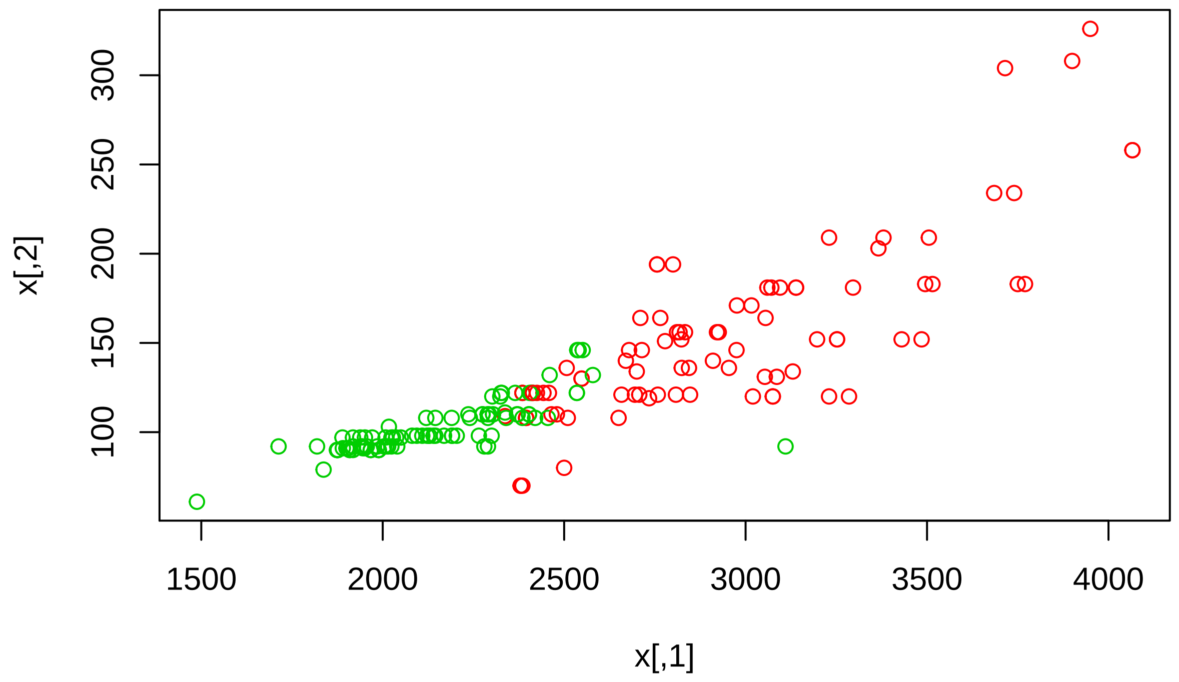
## Support Vector Classifier

First I format the car dataset for training the support vector classifier. I have used engineize and curbweight. We also created class label for the target using the median value for the car price.

```
set.seed(1)

high = ifelse(price<=10400, 0, 1)
y = high
cutlen = 180 # upto 205


x =  matrix( c(curbweight[1:cutlen], enginesize[1:cutlen]),ncol = 2, nrow  = cutlen) #very important

y = high[1:cutlen]

cardshort = data.frame(x = x, y = as.factor(y))
attach(cardshort)
```

```
## The following object is masked _by_ .GlobalEnv:
##
##     y
```

```
plot(x, col = (3-y))
```



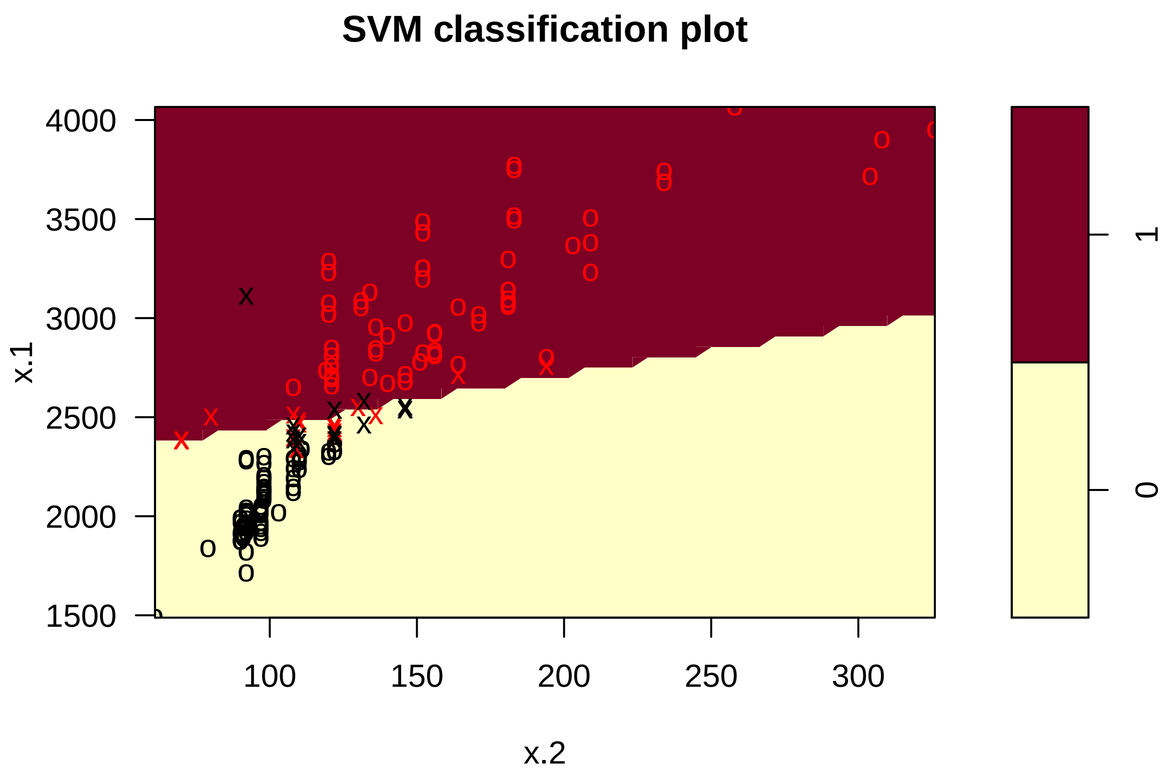
We see the distribution of the car price for the two variables. The car price red denotes high price.

I use the svm function from the library to implement car price classifier based on the two features I selected earlier.

```
library(e1071)
svmfit = svm(y~., data = cardshort, kernel = "linear", cost = 20, scale = FALSE)
```

As we have already trained the SVM classifier we can plot the classifier by as follow

```
plot(svmfit, cardshort)
```



We see that, the two region seperated the car prices. The red region return 1 and the greyish area return 0 in the feature space.

```
svmfit$index
```

```
##  [1]   1   2   4   6  11  12  13  42  56  57  58  59  62  64  65 127 128 146 175
## [20] 177 178  29  41  60  61  63  81  87  88  89 124 131 132 144 145 148 149 156
## [39] 168 169 170 176
```

```
summary(svmfit)
```

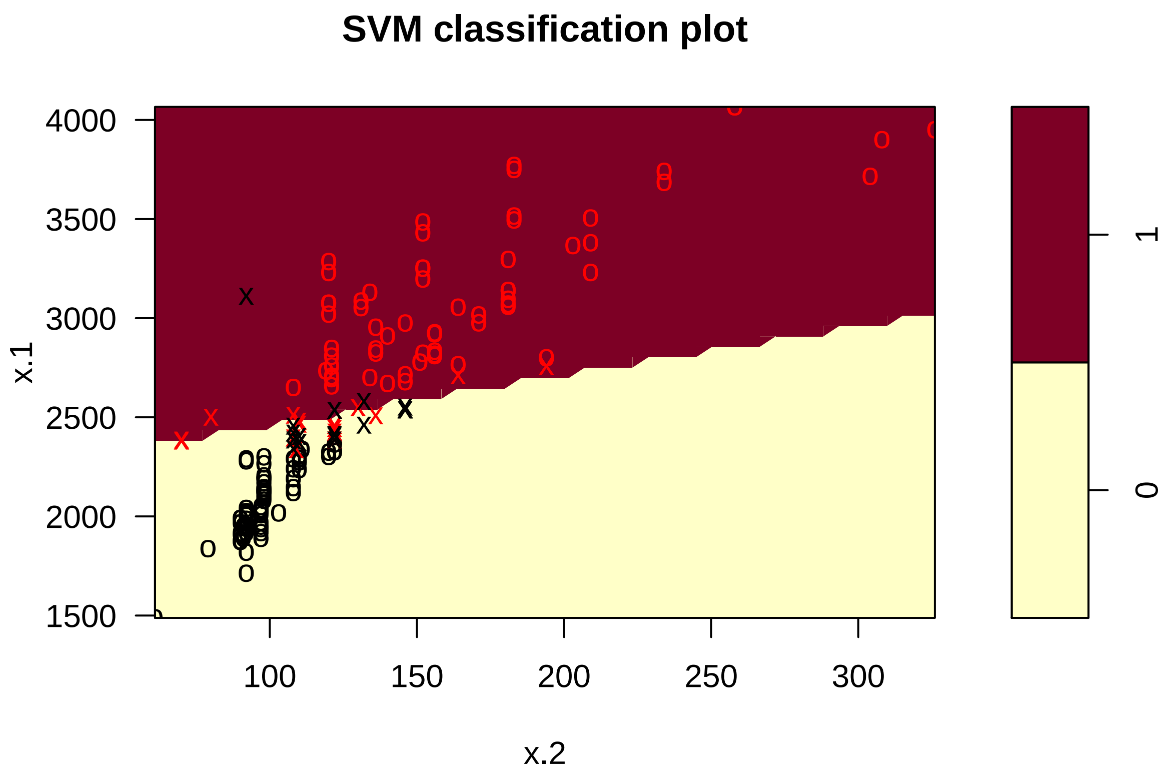
```
##
## Call:
## svm(formula = y ~ ., data = cardshort, kernel = "linear", cost = 20,
##     scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  20
##
## Number of Support Vectors:  42
##
##  ( 21 21 )
##
```

```
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

Here we see that linear kernel was used with cost 20.

We can change the cost parameters to smaller value and conduct experiment on the car price dataset.

```
svmfit = svm(y~., data = cardshort, kernel = "linear", cost = 0.01, scale = FALSE)
plot(svmfit, cardshort)
```



```
svmfit$index
```

```
##  [1]   1   2   4   6  11  12  13  42  56  57  58  59  62  64  65 127 128 146 175
## [20] 177 178  29  41  60  61  63  81  87  88  89 124 131 132 144 145 148 149 156
## [39] 168 169 170 176
```

With the cost we can control the number of support vector. With smaller cost parameter we find higher number of support vectors.

Now we sweep the value of cost in implement svm for different cost values

```r
set.seed(1)


tune.out = tune(svm,y~.,
                data = cardshort, ranges = list(cost =c(0.001, 0.01, 0.1, 1 ,5 ,10, 100)) )
```

We can observe the summary of the model as follows.

```r
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   100
##
## - best performance: 0.08333333
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-03 0.46666667 0.10861391
## 2 1e-02 0.46666667 0.10861391
## 3 1e-01 0.11111111 0.05237828
## 4 1e+00 0.11111111 0.06415003
## 5 5e+00 0.08888889 0.05367177
## 6 1e+01 0.08888889 0.05367177
## 7 1e+02 0.08333333 0.05399030
```

From summary value we can see the best cost for linear kernel is 0.0833.

We can also see the summary of the best model using R for the car data for the linear kernel.

```r
bestmod = tune.out$best.model
summary(bestmod)
```

```
##
## Call:
## best.tune(method = svm, train.x = y ~ ., data = cardshort, ranges = list(cost = c(0.001,
##     0.01, 0.1, 1, 5, 10, 100)))
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  100
##
## Number of Support Vectors:  39
##
##  ( 22 17 )
```

```
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

In previous section, we see the summary for the best model. It has 39 support vectors.

We can also predict the model performance on the test data.

```
xtest =  matrix(c(curbweight[(cutlen+1):205], enginesize[(cutlen+1):205] ),
                ncol = 2, nrow  =205- cutlen)
ytest = high[(cutlen+1):205]

cardshorttest = data.frame(x = xtest, y = as.factor(ytest))

ypred = predict(bestmod, cardshorttest)

table(predict = ypred, truth = cardshorttest$y)
```

```
##        truth
## predict  0  1
##       0  8  1
##       1  0 16
```

From previous result, we see that the model performance on the validation set. The model correctly classified 24 instances and failed in estimating 1 instance.

```
svmfit = svm(y~., data = cardshort, kernel = "linear", cost = 1, scale = FALSE)
ypred = predict(svmfit, cardshorttest)

table(predict = ypred, truth = cardshorttest$y)
```

```
##        truth
## predict  0  1
##       0  8  1
##       1  0 16
```

In the car data the cost from 0.1 to 1 didn't impact the test performances.

Now we change the cost to a high values.

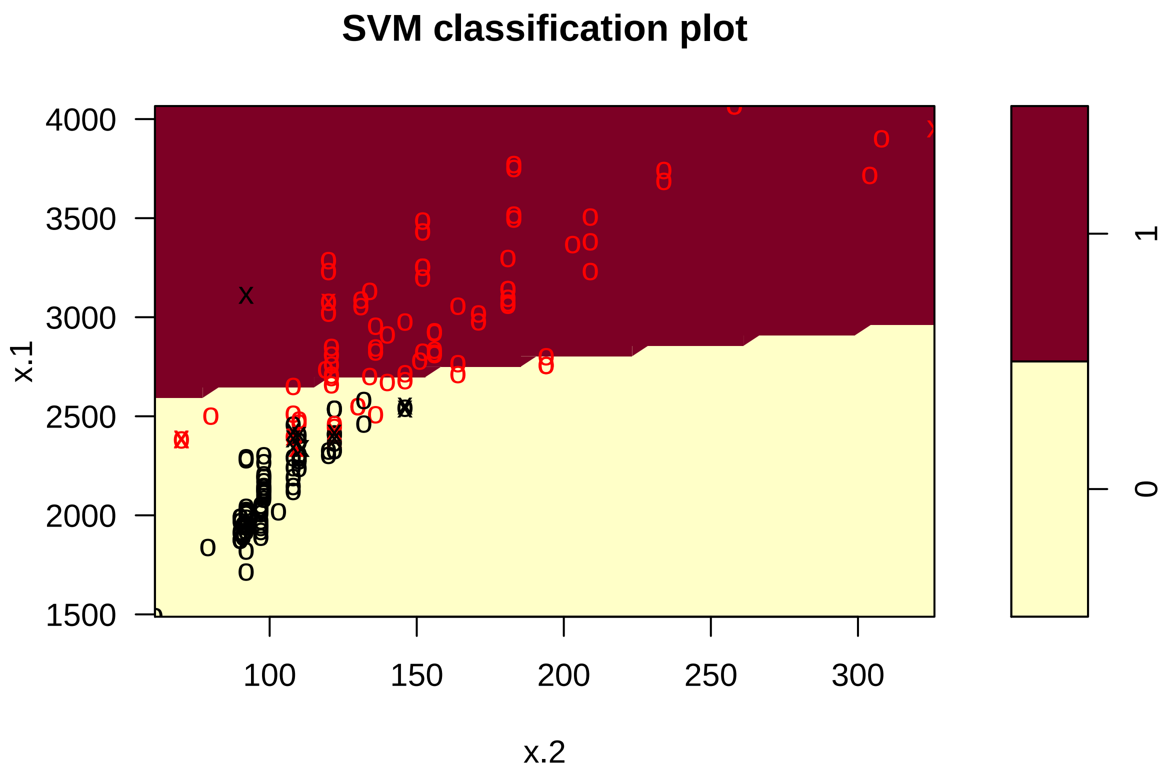
```
svmfit = svm(y~., data = cardshort, kernel = "linear", cost = 1e05, scale = FALSE)
summary(svmfit)
```

```
##
## Call:
## svm(formula = y ~ ., data = cardshort, kernel = "linear", cost = 1e+05,
##     scale = FALSE)
##
##
```

```
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1e+05
##
## Number of Support Vectors:  19
##
##  ( 9 10 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

Now we see teh support vectors number is 19, smaller than earlier cost = .1 (39).

```
plot(svmfit, cardshort)
```

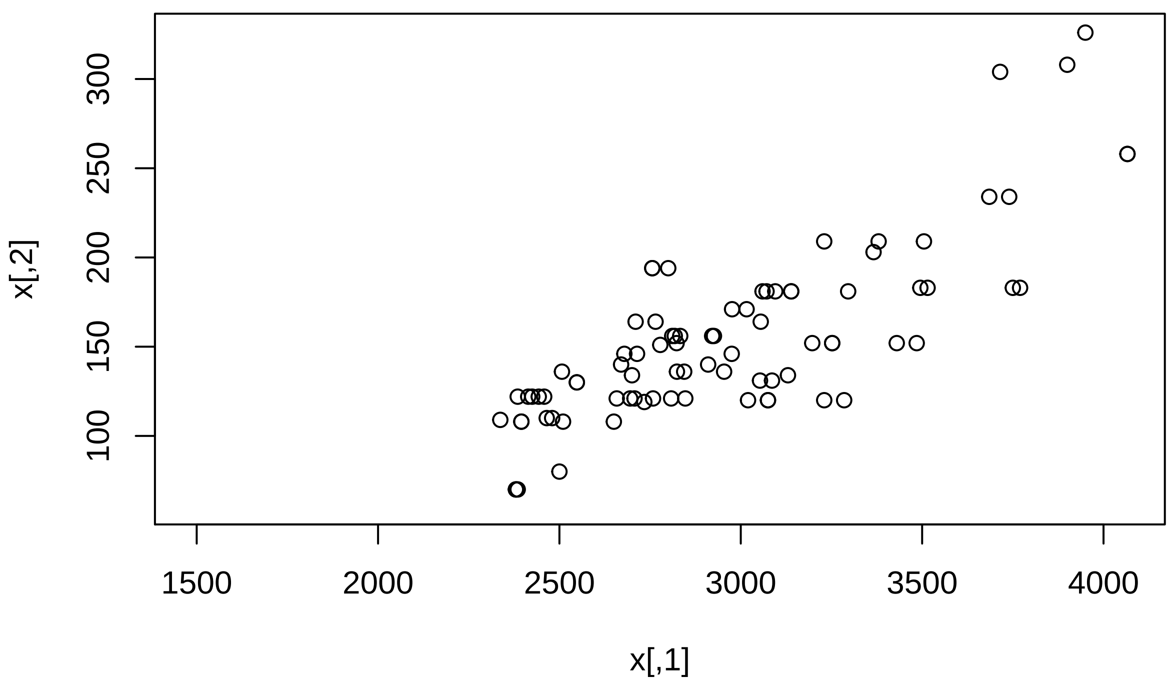


We plot the model and see the boundary lines are smooth compared to the earlier small cost value.

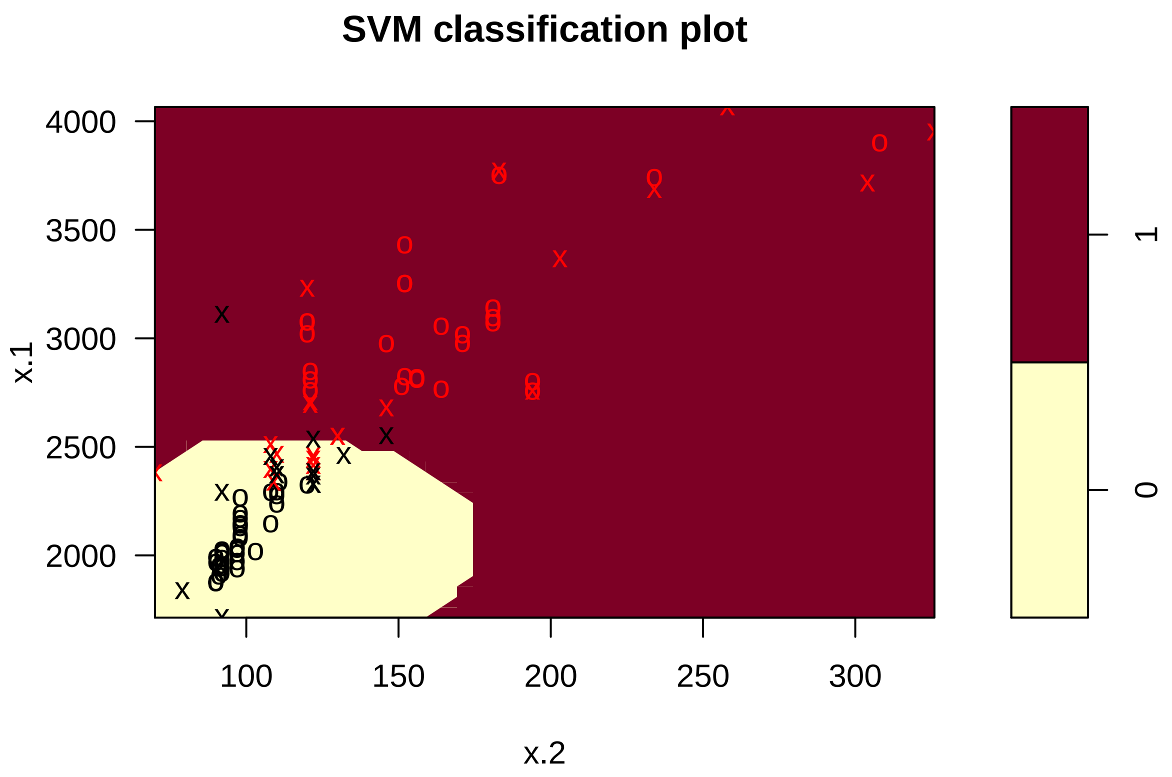
## SVM

In this section I experiment with the non-linear kernel for the car price dataset. We use previously defined class labels for this experiments.

```
plot(x, col =y)
```



```
train = sample(180,100)


svmfit = svm(y~., data = cardshort[train,], kernel = "radial", gamma = 1, cost = 1)
plot(svmfit, cardshort[train,])
```



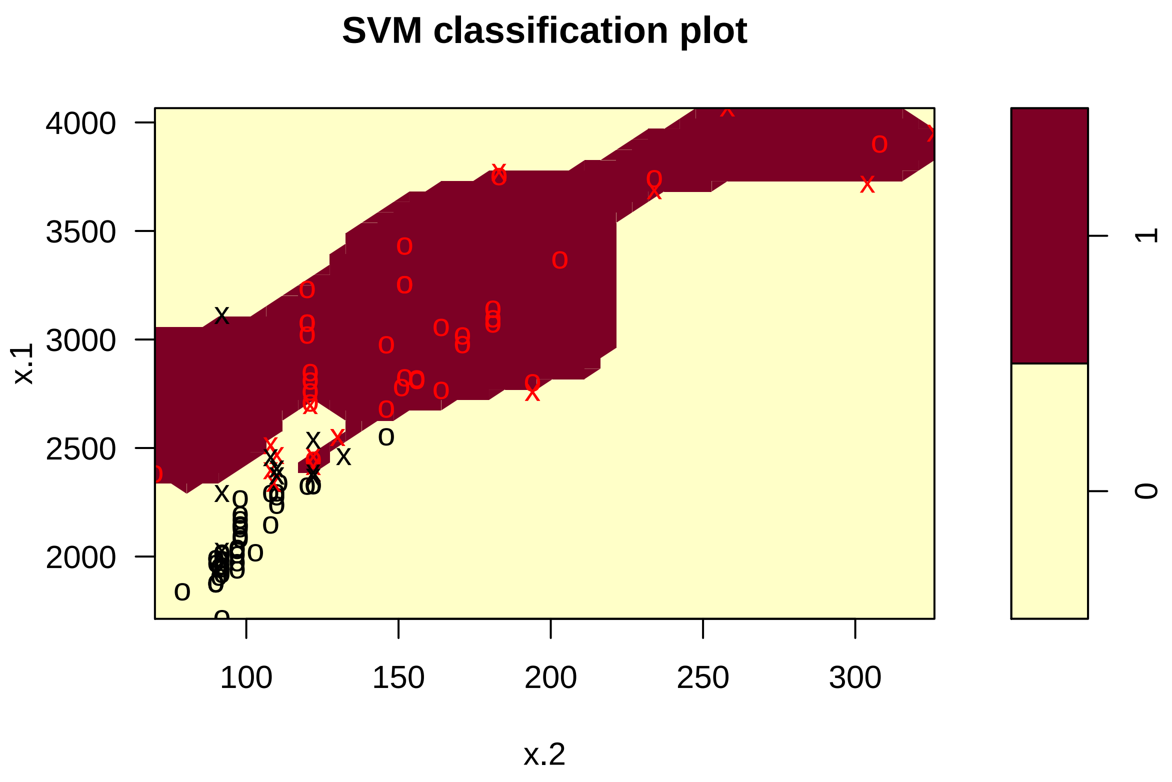
We see curved boundary for the decision classifier generated by radial basis classifiers.

```r
summary(svmfit)
```

```
##
## Call:
## svm(formula = y ~ ., data = cardshort[train, ], kernel = "radial",
##     gamma = 1, cost = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  38
##
##  ( 22 16 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

The radial kernel selected 38 support vectors to draw the bounday between the class labels.

```r
svmfit = svm(y~., data = cardshort[train,], kernel = "radial", gamma = 1, cost = 1e5)
plot(svmfit, cardshort[train,])
```



```r
summary(svmfit)
```

```
##
## Call:
## svm(formula = y ~ ., data = cardshort[train, ], kernel = "radial",
##     gamma = 1, cost = 1e+05)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1e+05
##
## Number of Support Vectors:  29
##
##  ( 17 12 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

We see complex boundary generated for the the decision by the svm with a high cost function with 29 support vectors.

```
set.seed(1)
tune.out = tune(svm,y~., data = cardshort[train,], kernel = "radial",
                ranges = list(cost =c(0.1, 1, 10, 100, 1000)) )
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   100
##
## - best performance: 0.1
##
## - Detailed performance results:
##    cost error dispersion
## 1 1e-01  0.13  0.1159502
## 2 1e+00  0.13  0.1159502
## 3 1e+01  0.13  0.1159502
## 4 1e+02  0.10  0.1247219
## 5 1e+03  0.11  0.1286684
```

We use the tune to find the best classifier by sweeping the cost value. We find that the model gets cost 0.1 as the best model. We evaluated the model on the test cases.

```
table(truc = cardshort[-train, "y"], pred = predict(tune.out$best.model,
                                                    newdata = cardshort[-train,]))
```

```
##     pred
## truc  0  1
##    0 41  3
##    1  2 34
```

Now we see the model performed with high accuracy on large number of test case and trained upon small number of instance. The model accuracy is

```
(41+34)/(43+37)
```

```
## [1] 0.9375
```

## ROC curve

Firstly, we define a function for plotting the ROC curves

```r
library(ROCR)
```

```
## Loading required package: gplots

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess
```

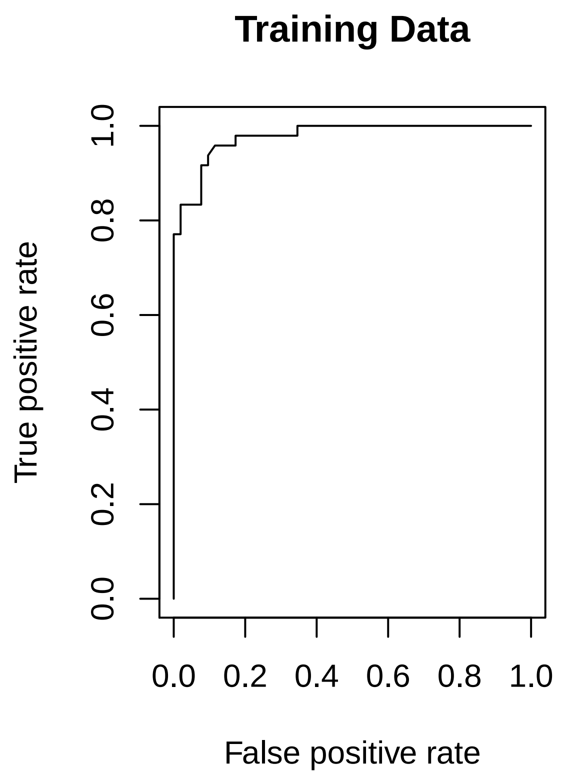
```r
rocplot = function(pred, truth, ...){
  predob = prediction(pred, truth)
  perf = performance(predob, "tpr", "fpr")
  plot(perf,...)
}
```

Now we implement the actual svm fit using the train car data portion.

```r
svmfit.opt = svm(y~., data = cardshort[train,],
                 kernel = "radial", gamma = 2, cost = 1, decision.values =T)


fitted = attributes(predict(svmfit.opt, cardshort[train,], decision.value = T))$decision.values
```

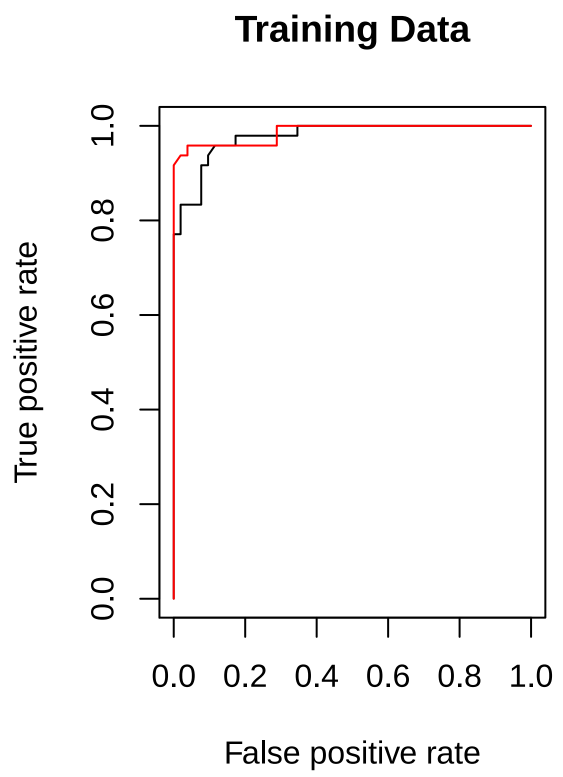
Next, we use the fitted model to plot the ROC curve.

```r
par(mfrow =c(1,2))

rocplot(fitted, cardshort[train, "y"], main = "Training Data")
```



The previous plot shows the ROC curves. The classifier seems working great on training instances as we see the area under the ROC curve is close to 0.

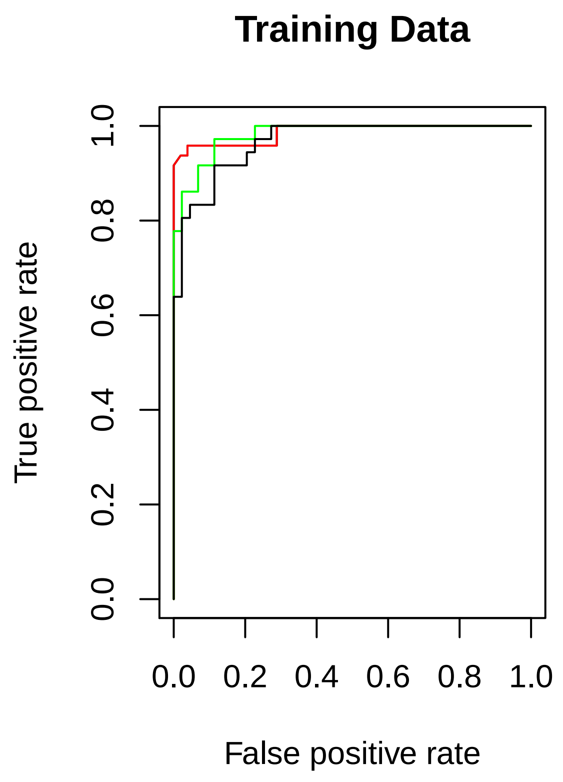
```
par(mfrow =c(1,2))
rocplot(fitted, cardshort[train, "y"], main = "Training Data")

svmfit.flex = svm(y~., data = cardshort[train,],
                  kernel = "radial", gamma = 2, cost = 10000, decision.values =T)


fitted = attributes(predict(svmfit.flex, cardshort[train,], decision.value = T))$decision.values

rocplot(fitted, cardshort[train, "y"], add =T, col ="red")
```



We overlap the two roc curve for two different cost function. In the lower FPR the red model seems a little better and at the FPR 0.4, the black ROC curve performed better.

Now we finally plot the the test models ROC curves in the same plot for comparison.

```
par(mfrow =c(1,2))
rocplot(fitted, cardshort[train, "y"], main = "Training Data")

svmfit.flex = svm(y~., data = cardshort[train,],
                  kernel = "radial", gamma = 2, cost = 10000, decision.values =T)


fitted = attributes(predict(svmfit.flex, cardshort[train,], decision.value = T))$decision.values

rocplot(fitted, cardshort[train, "y"], add =T, col ="red")



fitted = attributes(predict(svmfit.opt, cardshort[-train,], decision.value = T))$decision.values

rocplot(fitted, cardshort[-train, "y"], add =T, col ="green")


fitted = attributes(predict(svmfit.flex, cardshort[-train,], decision.value = T))$decision.values

rocplot(fitted, cardshort[-train, "y"], add =T, col ="black")
```



From the above figure, we see that the RED and green (optimal) model perfromed better than the black curve (gamma parameter 2) in terms of area under the curves.