FE590. Assignment #4.

Yifu He

2019-05-08

Instructions

When you have completed the assignment, knit the document into a PDF file, and upload *both* the .pdf and .Rmd files to Canvas.

Note that you must have LaTeX installed in order to knit the equations below. If you do not have it installed, simply delete the questions below.

```
library(knitr)
library(stringi)
library(devtools)
```

```
## Warning: package 'devtools' was built under R version 3.5.2
```

```
## Warning: package 'usethis' was built under R version 3.5.2
```

```
CWID = 10442277 #Place here your Campus wide ID number, this will personalize
#your results, but still maintain the reproduceable nature of using seeds.
#If you ever need to reset the seed in this assignment, use this as your seed
#Papers that use -1 as this CWID variable will earn 0's so make sure you chan
ge
#this value before you submit your work.
personal = CWID %% 10000
set.seed(personal)
```

Question 1:

In this assignment, you will be required to find a set of data to run regression on. This data set should be financial in nature, and of a type that will work with the models we have discussed this semester (hint: we didn't look at time series) You may not use any of the data sets in the ISLR package that we have been looking at all semester. Your data set that you choose should have both qualitative and quantitative variables. (or has variables that you can transform)Provide a description of the data below, where you obtained it, what the variable names are and what it is describing.

Answer1 :

(i) Description of Data : The data is based on the daily historical bitcoin market prices and other factors related to it from 23th Feb 2010 to 20th Feb 2018. This data set was obtained from www.kaggle.com. The dataset includes 24 predictors and 2899 entries.

```
set.seed(personal)
setwd("/Users/yifuhe/Desktop")
data1 = read.csv("bitcoin_dataset.csv")
data = na.omit(data1)
nr = nrow(data)
nc = ncol(data)
print(paste0("The number of rows  in the data set are ", nrow(data)))

## [1] "The number of rows  in the data set are 2899"

print(paste0("The number of columns  in the data set are ", ncol(data)))

## [1] "The number of columns  in the data set are 24"

head(data)

##                  Date btc_market_price btc_total_bitcoins btc_market_cap
## 1 2010-02-23 00:00:00                0            2110700              0
## 2 2010-02-24 00:00:00                0            2120200              0
## 3 2010-02-25 00:00:00                0            2127600              0
## 4 2010-02-26 00:00:00                0            2136100              0
## 5 2010-02-27 00:00:00                0            2144750              0
## 6 2010-02-28 00:00:00                0            2152850              0
##   btc_trade_volume btc_blocks_size btc_avg_block_size
## 1                0               0       0.0002163347
## 2                0               0       0.0002817211
## 3                0               0       0.0002269054
## 4                0               0       0.0003186765
## 5                0               0       0.0002234162
## 6                0               0       0.0002914506
##   btc_n_orphaned_blocks btc_n_transactions_per_block
## 1                     0                            1
## 2                     0                            1
## 3                     0                            1
## 4                     0                            1
## 5                     0                            1
## 6                     0                            1
##   btc_median_confirmation_time btc_hash_rate btc_difficulty
## 1                            0  3.153929e-05       2.527738
## 2                            0  3.571305e-05       3.781179
## 3                            0  2.781859e-05       3.781179
## 4                            0  3.195378e-05       3.781179
## 5                            0  3.251768e-05       3.781179
## 6                            0  3.045008e-05       3.781179
##   btc_miners_revenue btc_transaction_fees btc_cost_per_transaction_percent
## 1                  0                    0                      25100.00000
```

```
## 2                 0                    0                        179.24528
## 3                 0                    0                       1057.14286
## 4                 0                    0                         64.58206
## 5                 0                    0                       1922.22222
## 6                 0                    0                        154.28571
##   btc_cost_per_transaction btc_n_unique_addresses btc_n_transactions
## 1                        0                    252                252
## 2                        0                    195                196
## 3                        0                    150                150
## 4                        0                    176                176
## 5                        0                    176                176
## 6                        0                    165                165
##   btc_n_transactions_total btc_n_transactions_excluding_popular
## 1                    42613                                  252
## 2                    42809                                  196
## 3                    42959                                  150
## 4                    43135                                  176
## 5                    43311                                  176
## 6                    43476                                  165
##   btc_n_transactions_excluding_chains_longer_than_100 btc_output_volume
## 1                                                 252             12600
## 2                                                 196             14800
## 3                                                 150              8100
## 4                                                 176             29349
## 5                                                 176              9101
## 6                                                 165             13399
##   btc_estimated_transaction_volume btc_estimated_transaction_volume_usd
## 1                               50                                    0
## 2                             5300                                    0
## 3                              700                                    0
## 4                            13162                                    0
## 5                              450                                    0
## 6                             5250                                    0
```

```
data$Date = as.Date(data$Date)
head(data) # after cleaning date column
```

```
##         Date btc_market_price btc_total_bitcoins btc_market_cap
## 1 2010-02-23                0            2110700              0
## 2 2010-02-24                0            2120200              0
## 3 2010-02-25                0            2127600              0
## 4 2010-02-26                0            2136100              0
## 5 2010-02-27                0            2144750              0
## 6 2010-02-28                0            2152850              0
##   btc_trade_volume btc_blocks_size btc_avg_block_size
## 1                0               0       0.0002163347
## 2                0               0       0.0002817211
## 3                0               0       0.0002269054
## 4                0               0       0.0003186765
## 5                0               0       0.0002234162
```

```
## 6                0              0      0.0002914506
##   btc_n_orphaned_blocks btc_n_transactions_per_block
## 1                     0                            1
## 2                     0                            1
## 3                     0                            1
## 4                     0                            1
## 5                     0                            1
## 6                     0                            1
##   btc_median_confirmation_time btc_hash_rate btc_difficulty
## 1                            0  3.153929e-05       2.527738
## 2                            0  3.571305e-05       3.781179
## 3                            0  2.781859e-05       3.781179
## 4                            0  3.195378e-05       3.781179
## 5                            0  3.251768e-05       3.781179
## 6                            0  3.045008e-05       3.781179
##   btc_miners_revenue btc_transaction_fees btc_cost_per_transaction_percent
## 1                  0                    0                      25100.00000
## 2                  0                    0                        179.24528
## 3                  0                    0                       1057.14286
## 4                  0                    0                         64.58206
## 5                  0                    0                       1922.22222
## 6                  0                    0                        154.28571
##   btc_cost_per_transaction btc_n_unique_addresses btc_n_transactions
## 1                        0                    252                252
## 2                        0                    195                196
## 3                        0                    150                150
## 4                        0                    176                176
## 5                        0                    176                176
## 6                        0                    165                165
##   btc_n_transactions_total btc_n_transactions_excluding_popular
## 1                    42613                                  252
## 2                    42809                                  196
## 3                    42959                                  150
## 4                    43135                                  176
## 5                    43311                                  176
## 6                    43476                                  165
##   btc_n_transactions_excluding_chains_longer_than_100 btc_output_volume
## 1                                                 252             12600
## 2                                                 196             14800
## 3                                                 150              8100
## 4                                                 176             29349
## 5                                                 176              9101
## 6                                                 165             13399
##   btc_estimated_transaction_volume btc_estimated_transaction_volume_usd
## 1                               50                                    0
## 2                             5300                                    0
## 3                              700                                    0
## 4                            13162                                    0
## 5                              450                                    0
## 6                             5250                                    0
```

(ii) Aim for the project : TO PREDICT BITCOIN MARKET PRICES FROM THE PREDICTORS GIVEN BELOW. ALSO TO FIND WHICH OF THE PREDICTORS ARE CORRELATED TO MARKET PRICES..

(iii) Response Variable is btc_market_price This dataset has the following features.

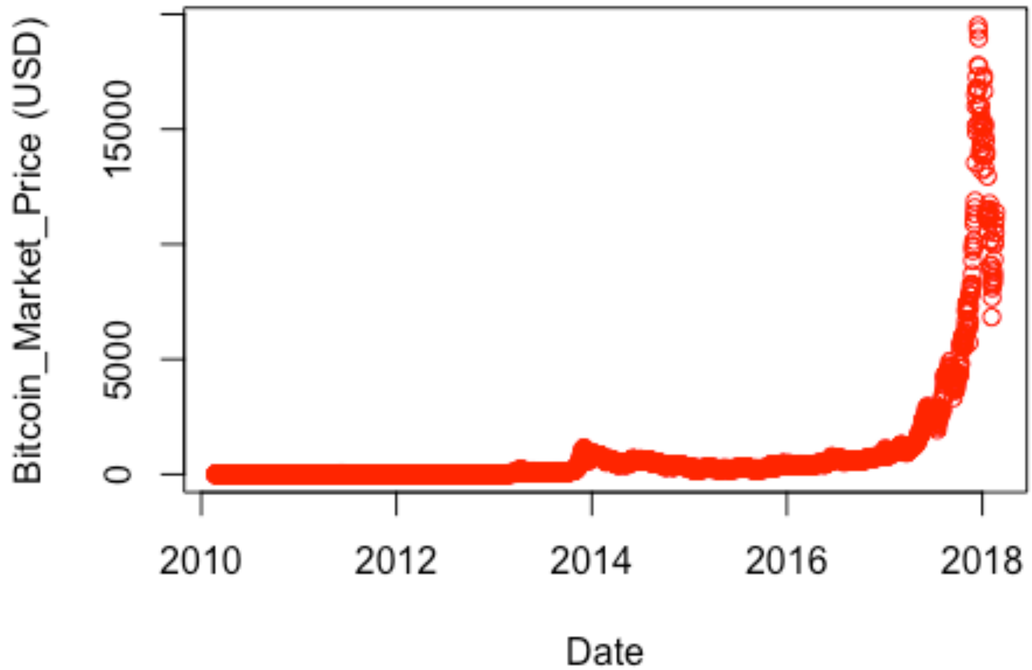
Date : Date of observation btc_market_price : Average USD market price across major bitcoin exchanges. btc_total_bitcoins : The total number of bitcoins that have already been mined. btc_market_cap : The total USD value of bitcoin supply in circulation. btc_trade_volume : The total USD value of trading volume on major bitcoin exchanges. btc_blocks_size : The total size of all block headers and transactions. btc_avg_block_size : The average block size in MB. btc_n_orphaned_blocks : The total number of blocks mined but ultimately not attached to the main Bitcoin blockchain. btc_n_transactions_per_block : The average number of transactions per block. btc_median_confirmation_time : The median time for a transaction to be accepted into a mined block. btc_hash_rate : The estimated number of tera hashes per second the Bitcoin network is performing. btc_difficulty : A relative measure of how difficult it is to find a new block. btc_miners_revenue : Total value of coinbase block rewards and transaction fees paid to miners. btc_transaction_fees : The total value of all transaction fees paid to miners. btc_cost_per_transaction_percent : miners revenue as percentage of the transaction volume. btc_cost_per_transaction : miners revenue divided by the number of transactions. btc_n_unique_addresses : The total number of unique addresses used on the Bitcoin blockchain. btc_n_transactions : The number of daily confirmed Bitcoin transactions. btc_n_transactions_total : Total number of transactions. btc_n_transactions_excluding_popular : The total number of Bitcoin transactions, excluding the 100 most popular addresses. btc_n_transactions_excluding_chains_longer_than_100 : The total number of Bitcoin transactions per day excluding long transaction chains. btc_output_volume : The total value of all transaction outputs per day. btc_estimated_transaction_volume : The total estimated value of transactions on the Bitcoin blockchain. btc_estimated_transaction_volume_usd : The estimated transaction value in USD value.

```r
# Looking at the data....
set.seed(personal)
summary(data)
```

```
##       Date            btc_market_price    btc_total_bitcoins
##  Min.   :2010-02-23   Min.   :    0.000   Min.   : 2110700  
##  1st Qu.:2012-02-23   1st Qu.:    6.714   1st Qu.: 8410825  
##  Median :2014-02-19   Median :  236.000   Median :12418575  
##  Mean   :2014-02-22   Mean   :  901.824   Mean   :11522310  
##  3rd Qu.:2016-02-26   3rd Qu.:  604.460   3rd Qu.:15255538  
##  Max.   :2018-02-20   Max.   :19498.683   Max.   :16876825  
##  btc_market_cap      btc_trade_volume    btc_blocks_size   
##  Min.   :0.000e+00   Min.   :0.000e+00   Min.   :     0.0  
##  1st Qu.:5.488e+07   1st Qu.:2.994e+05   1st Qu.:   779.5  
##  Median :3.364e+09   Median :1.024e+07   Median : 15035.0  
##  Mean   :1.451e+10   Mean   :8.231e+07   Mean   : 36202.8  
##  3rd Qu.:8.229e+09   3rd Qu.:2.935e+07   3rd Qu.: 59897.5  
```

```
##  Max.   :3.265e+11   Max.   :5.352e+09   Max.   :157665.0
##  btc_avg_block_size  btc_n_orphaned_blocks btc_n_transactions_per_block
##  Min.   :0.0002163   Min.   :0.0000        Min.   :   1.0
##  1st Qu.:0.0245796   1st Qu.:0.0000        1st Qu.:  54.5
##  Median :0.1996229   Median :0.0000        Median : 379.0
##  Mean   :0.3567457   Mean   :0.3581        Mean   : 679.3
##  3rd Qu.:0.6933442   3rd Qu.:0.0000        3rd Qu.:1245.7
##  Max.   :1.1103268   Max.   :7.0000        Max.   :2722.6
##  btc_median_confirmation_time btc_hash_rate      btc_difficulty
##  Min.   : 0.000               Min.   :       0   Min.   :3.000e+00
##  1st Qu.: 6.133               1st Qu.:      12   1st Qu.:1.627e+06
##  Median : 7.933               Median :   25981   Median :3.130e+09
##  Mean   : 7.561               Mean   : 1396897   Mean   :1.820e+11
##  3rd Qu.:10.271               3rd Qu.: 1132497   3rd Qu.:1.584e+11
##  Max.   :47.733               Max.   :25579249   Max.   :2.968e+12
##  btc_miners_revenue btc_transaction_fees btc_cost_per_transaction_percent
##  Min.   :       0   Min.   :   0.000     Min.   :    0.14
##  1st Qu.:   47011   1st Qu.:   9.624     1st Qu.:    1.18
##  Median :  888738   Median :  21.405     Median :    2.46
##  Mean   : 2306187   Mean   :  61.201     Mean   :   58.47
##  3rd Qu.: 1862391   3rd Qu.:  51.014     3rd Qu.:    5.84
##  Max.   :53191582   Max.   :1495.947     Max.   :88571.43
##  btc_cost_per_transaction btc_n_unique_addresses btc_n_transactions
##  Min.   :  0.000          Min.   :    110        Min.   :   118
##  1st Qu.:  4.172          1st Qu.:  17008        1st Qu.:  8056
##  Median :  7.839          Median : 131955        Median : 62960
##  Mean   : 15.192          Mean   : 196512        Mean   :103257
##  3rd Qu.: 14.976          3rd Qu.: 367857        3rd Qu.:191969
##  Max.   :161.686          Max.   :1072861        Max.   :490644
##  btc_n_transactions_total btc_n_transactions_excluding_popular
##  Min.   :    42613        Min.   :   118
##  1st Qu.:  2490264        1st Qu.:  6878
##  Median : 33231891        Median : 54894
##  Mean   : 70431859        Mean   : 95502
##  3rd Qu.:112793186        3rd Qu.:187552
##  Max.   :300576632        Max.   :470650
##  btc_n_transactions_excluding_chains_longer_than_100 btc_output_volume
##  Min.   :   118                                      Min.   :    6150
##  1st Qu.:  6836                                      1st Qu.:  496080
##  Median : 35658                                      Median : 1116561
##  Mean   : 64000                                      Mean   : 1567758
##  3rd Qu.:115688                                      3rd Qu.: 2029856
##  Max.   :318896                                      Max.   :45992223
##  btc_estimated_transaction_volume btc_estimated_transaction_volume_usd
##  Min.   :      7                  Min.   :0.000e+00
##  1st Qu.:  96478                  1st Qu.:9.700e+05
##  Median : 179252                  Median :3.902e+07
##  Mean   : 203961                  Mean   :2.131e+08
##  3rd Qu.: 258903                  3rd Qu.:1.386e+08
##  Max.   :5825066                  Max.   :5.760e+09
```

```r
#Analysing the given data set:---
set.seed(personal)
plot(data$Date, data$btc_market_price, xlab = "Date", ylab = "Bitcoin_Market_
Price (USD)", col = "red")
```



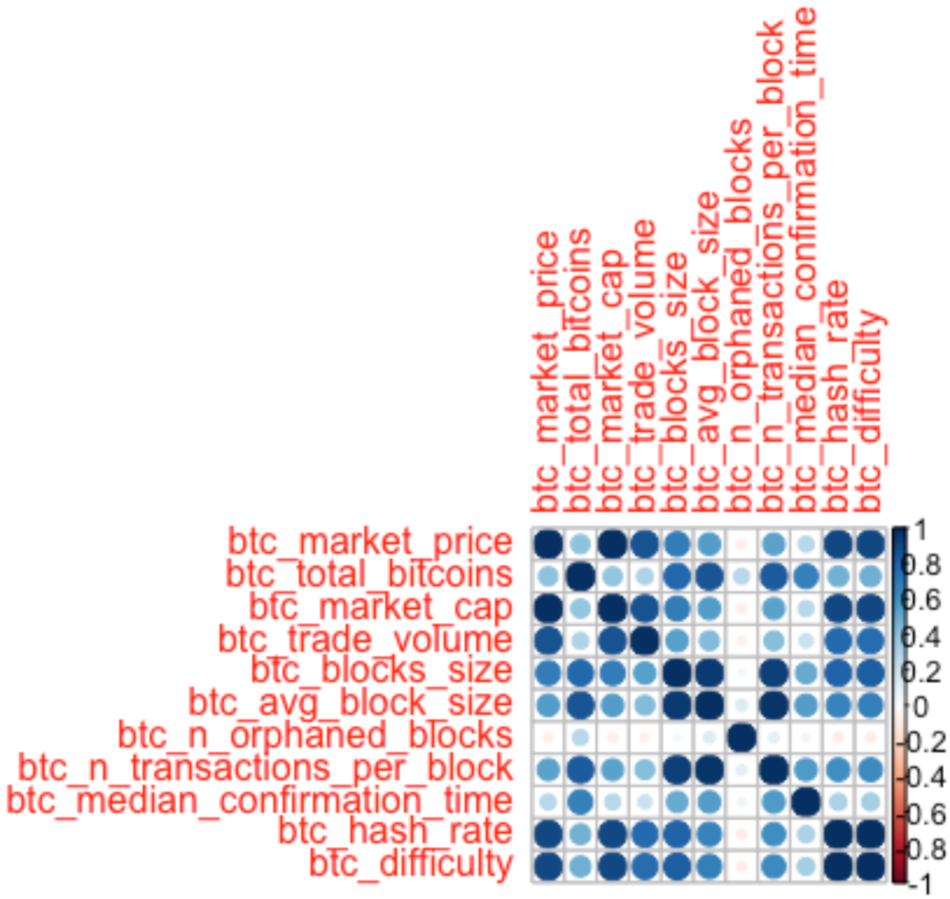

```r
#training and testing data
t_ind = sample(1:nr, 0.75 * nr, replace = F)
train = data[t_ind,]
test = data[-t_ind,]
cor(train[, -c(1,24)])[1,]

##                                  btc_market_price
##                                        1.00000000
##                                 btc_total_bitcoins
##                                        0.40648769
##                                     btc_market_cap
##                                        0.99980760
##                                   btc_trade_volume
##                                        0.86369765
##                                    btc_blocks_size
##                                        0.68955479
##                                 btc_avg_block_size
##                                        0.56025478
```

```
##                                        btc_n_orphaned_blocks 
##                                                  -0.08222715 
##                                btc_n_transactions_per_block 
##                                                   0.54527049 
##                                btc_median_confirmation_time 
##                                                   0.28260736 
##                                                btc_hash_rate 
##                                                   0.90001108 
##                                               btc_difficulty 
##                                                   0.89533355 
##                                           btc_miners_revenue 
##                                                   0.98480128 
##                                         btc_transaction_fees 
##                                                   0.78157994 
##                           btc_cost_per_transaction_percent 
##                                                  -0.01246046 
##                                     btc_cost_per_transaction 
##                                                   0.83623012 
##                                       btc_n_unique_addresses 
##                                                   0.66240921 
##                                           btc_n_transactions 
##                                                   0.56562461 
##                                     btc_n_transactions_total 
##                                                   0.69413628 
##                      btc_n_transactions_excluding_popular 
##                                                   0.55462051 
## btc_n_transactions_excluding_chains_longer_than_100 
##                                                   0.56676867 
##                                            btc_output_volume 
##                                                   0.10330967 
##                          btc_estimated_transaction_volume 
##                                                   0.04587268
```

```r
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```r
corrplot(cor(data[,2:12]), method = "circle")
```



(iv) After analysing the data before selecting the best predictors, some of the conclusions made are :—

- Bitcoin market prices increases exponentially wrt date but falls in between 2016 and 2018 drastically.
- From the above correlation diagram and the values calculated, predictors which seem to be correlated to the market price are : btc_market_cap, btc_trade_volume, btc_hash_rate, btc_miners_revenue,btc_difficulty, btc_cost_per_transaction

———————————————————————————————————————-

Question 2:

Pick a quantitative variable and fit at least four different models in order to predict that variable using the other predictors. Determine which of the models is the best fit. You will need to provide strong reasons as to why the particular model you chose is the best one. You will need to confirm the model you have selected provides the best fit and that you have obtained the best version of that particular model (i.e. subset selection or validation for example). You need to convince the grader that you have chosen the best model.

Answer 2 :

(MOdel 1) SIMPLE LINEAR REGRESSION : Since this is a simple regression model .. let us select the best predictor instead of applying to all the predictors. Also after finding correlation, let us see what are the results after the best subset selection, forward subset selection and backward subset selection.

```
set.seed(personal)
library(leaps)
p = regsubsets(btc_market_price ~ btc_market_cap + btc_trade_volume + btc_has
h_rate + btc_miners_revenue + btc_difficulty + btc_cost_per_transaction, data
= train)
q = regsubsets(btc_market_price ~ btc_market_cap + btc_trade_volume + btc_has
h_rate + btc_miners_revenue + btc_difficulty + btc_cost_per_transaction, data
= train, method = "forward")
r = regsubsets(btc_market_price ~ btc_market_cap + btc_trade_volume + btc_has
h_rate + btc_miners_revenue + btc_difficulty + btc_cost_per_transaction, data
= train, method = "backward")
summary(p)[7]

## $outmat
##          btc_market_cap btc_trade_volume btc_hash_rate btc_miners_revenue
## 1  ( 1 ) "*"            " "              " "           " "
## 2  ( 1 ) "*"            " "              " "           " "
## 3  ( 1 ) "*"            " "              " "           "*"
## 4  ( 1 ) "*"            "*"              " "           "*"
## 5  ( 1 ) "*"            " "              "*"           "*"
## 6  ( 1 ) "*"            "*"              "*"           "*"
##          btc_difficulty btc_cost_per_transaction
## 1  ( 1 ) " "            " "
## 2  ( 1 ) " "            "*"
## 3  ( 1 ) " "            "*"
## 4  ( 1 ) " "            "*"
## 5  ( 1 ) "*"            "*"
## 6  ( 1 ) "*"            "*"

summary(q)[7]

## $outmat
##          btc_market_cap btc_trade_volume btc_hash_rate btc_miners_revenue
## 1  ( 1 ) "*"            " "              " "           " "
## 2  ( 1 ) "*"            " "              " "           " "
## 3  ( 1 ) "*"            " "              " "           "*"
## 4  ( 1 ) "*"            "*"              " "           "*"
## 5  ( 1 ) "*"            "*"              "*"           "*"
## 6  ( 1 ) "*"            "*"              "*"           "*"
##          btc_difficulty btc_cost_per_transaction
## 1  ( 1 ) " "            " "
## 2  ( 1 ) " "            "*"
## 3  ( 1 ) " "            "*"
```

```
## 4  ( 1 ) " "           "*"
## 5  ( 1 ) " "           "*"
## 6  ( 1 ) "*"           "*"
```

```r
summary(r)[7]
```

```
## $outmat
##          btc_market_cap btc_trade_volume btc_hash_rate btc_miners_revenue
## 1  ( 1 ) "*"            " "              " "           " "
## 2  ( 1 ) "*"            " "              " "           " "
## 3  ( 1 ) "*"            " "              " "           "*"
## 4  ( 1 ) "*"            " "              "*"           "*"
## 5  ( 1 ) "*"            " "              "*"           "*"
## 6  ( 1 ) "*"            "*"              "*"           "*"
##          btc_difficulty btc_cost_per_transaction
## 1  ( 1 ) " "            " "
## 2  ( 1 ) " "            "*"
## 3  ( 1 ) " "            "*"
## 4  ( 1 ) " "            "*"
## 5  ( 1 ) "*"            "*"
## 6  ( 1 ) "*"            "*"
```

After seeing the best, forward and the backward subset selection, the btc_market_cap i.e the total USD of bitcoin in circulation is the best predictor. So applying simple linear regression using this predictor.

```r
set.seed(personal)
model1 = lm(btc_market_price ~ btc_market_cap, data = train)
model11 = lm(btc_market_price ~ btc_cost_per_transaction, data = train)
summary(model1)
```

```
##
## Call:
## lm(formula = btc_market_price ~ btc_market_cap, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -96.264 -34.972  -4.253   7.860 285.249
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    3.791e+01  1.093e+00    34.7   <2e-16 ***
## btc_market_cap 5.956e-08  2.507e-11  2375.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 47.97 on 2172 degrees of freedom
## Multiple R-squared:  0.9996, Adjusted R-squared:  0.9996
## F-statistic: 5.643e+06 on 1 and 2172 DF,  p-value: < 2.2e-16
```

```r
summary(model11)
```

```
##
## Call:
## lm(formula = btc_market_price ~ btc_cost_per_transaction, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7196.7  -254.7   153.0   527.6  7766.0
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              -527.598     35.182  -15.00   <2e-16 ***
## btc_cost_per_transaction   93.914      1.321   71.07   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1341 on 2172 degrees of freedom
## Multiple R-squared:  0.6993, Adjusted R-squared:  0.6991
## F-statistic:  5051 on 1 and 2172 DF,  p-value: < 2.2e-16
```

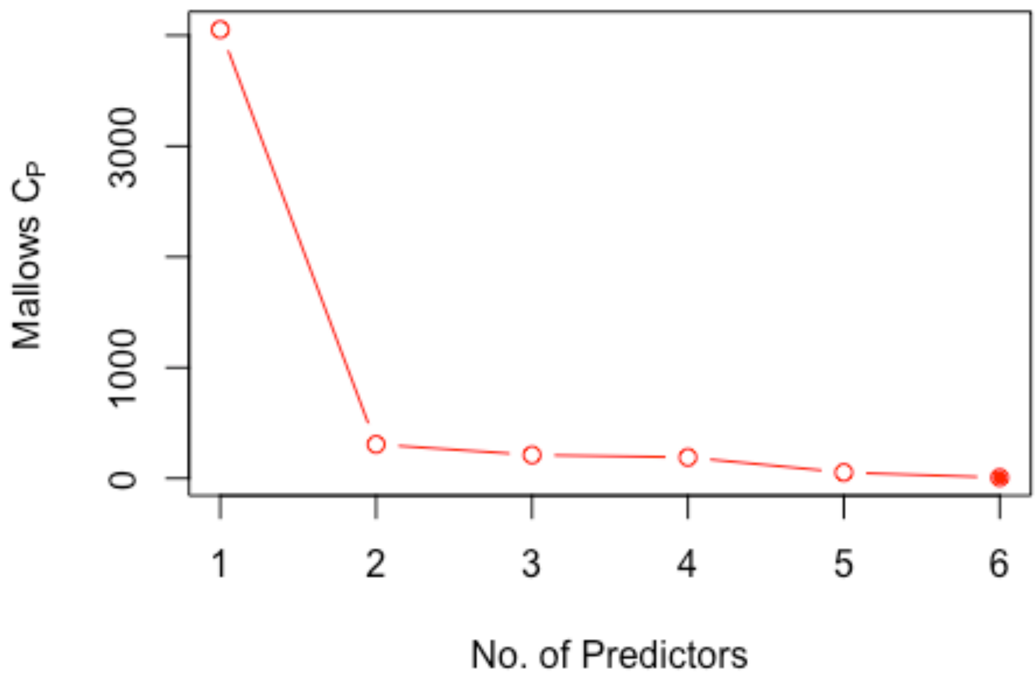
We can see how strongly btc_market_cap affects the btc_market price as adj R^2 is nearly equal to 1. Also the second best predictor btc_cost_per_transaction has a adjusted R-squared = 0.7012. Both are satistically significant as seen by the p-values. Now we test this and predict values using testing dataset.

```
set.seed(personal)
pred1 = predict(model1, newdata = test)
pred11 = predict(model11, newdata = test)
mss1 = mean((test$btc_market_price-pred1)^2)
mss11 = mean((test$btc_market_price-pred11)^2)
mss1

## [1] 2366.149

mss11

## [1] 1730603
```

MSS is too high so this is not a good model. Simple Linear regression is not good to fit. So we procced to multiple regression.

(Model ii) Multiple Linear Regression :

```
c1 <- summary(p)$cp
plot(c1,type='b',xlab="No. of Predictors",ylab=expression("Mallows C"[P]), co
l="red")

points(which.min(c1), c1[which.min(c1)], pch=20, col="red")
```



All the 6 predictors we selected have the minimum mallows Cp. So applying multiple regression model. But since btc_market_cap has adj R^2 = 0.996, we perform multiple linear regression with it and one multiple linear regression without it to exclude other variables as it is overshadowing others.

```
model12 = lm(btc_market_price ~ btc_market_cap + btc_trade_volume + btc_hash_
rate + btc_miners_revenue + btc_difficulty + btc_cost_per_transaction, data =
train)
summary(model12)

##
## Call:
## lm(formula = btc_market_price ~ btc_market_cap + btc_trade_volume +
##     btc_hash_rate + btc_miners_revenue + btc_difficulty + btc_cost_per_tra
nsaction,
##     data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -173.439  -13.540   -2.434   15.228  178.729
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 9.427e+00  8.232e-01  11.452  < 2e-16 ***
```

```
## btc_market_cap           5.622e-08  1.727e-10 325.458  < 2e-16 ***
## btc_trade_volume        -2.816e-08  4.062e-09  -6.931 5.49e-12 ***
## btc_hash_rate           -2.151e-05  1.576e-06 -13.649  < 2e-16 ***
## btc_miners_revenue       1.549e-05  1.093e-06  14.172  < 2e-16 ***
## btc_difficulty           1.880e-10  1.431e-11  13.137  < 2e-16 ***
## btc_cost_per_transaction  2.572e+00  5.970e-02  43.088  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 28.34 on 2167 degrees of freedom
## Multiple R-squared:  0.9999, Adjusted R-squared:  0.9999
## F-statistic: 2.695e+06 on 6 and 2167 DF,  p-value: < 2.2e-16
```

```
model13 = lm(btc_market_price ~ btc_trade_volume + btc_hash_rate + btc_miners
_revenue + btc_difficulty + btc_cost_per_transaction, data = train)
summary(model13)
```

```
##
## Call:
## lm(formula = btc_market_price ~ btc_trade_volume + btc_hash_rate +
##     btc_miners_revenue + btc_difficulty + btc_cost_per_transaction,
##     data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3432.5   -54.4    47.2    73.8  1636.1
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)              -6.371e+01  5.592e+00 -11.394  < 2e-16 ***
## btc_trade_volume         -2.327e-07  2.834e-08  -8.213 3.68e-16 ***
## btc_hash_rate            -2.781e-04  9.633e-06 -28.873  < 2e-16 ***
## btc_miners_revenue        3.558e-04  2.256e-06 157.734  < 2e-16 ***
## btc_difficulty            3.464e-09  7.186e-11  48.199  < 2e-16 ***
## btc_cost_per_transaction -5.133e+00  3.869e-01 -13.264  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 200.1 on 2168 degrees of freedom
## Multiple R-squared:  0.9933, Adjusted R-squared:  0.9933
## F-statistic: 6.444e+04 on 5 and 2168 DF,  p-value: < 2.2e-16
```

So after applying multiple linear regression without the btc_market_cap,the conclusion is that all the other variables too are significant.

```
set.seed(personal)
pred2 = predict(model12, newdata = test)
mss2 = mean((test$btc_market_price - pred2)^2)
mss2
```

```
## [1] 887.268
```

Still the error is significantly high but lower than simple linear. So we proceed to splines

(Model 3) : Polynomial regression : By the subset selections now select only and the above results drop btc_trade_volume

```
set.seed(personal)
fit1 = lm(btc_market_price ~ poly(btc_market_cap, 2), data = train)
fit2 = lm(btc_market_price ~ poly(btc_market_cap, 3), data = train)
summary(fit1)

##
## Call:
## lm(formula = btc_market_price ~ poly(btc_market_cap, 2), data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -106.601  -31.645   -6.336    4.144  281.421
##
## Coefficients:
##                            Estimate Std. Error  t value Pr(>|t|)
## (Intercept)                 912.059      1.014  899.600  < 2e-16 ***
## poly(btc_market_cap, 2)1 113959.971     47.272 2410.731  < 2e-16 ***
## poly(btc_market_cap, 2)2   -383.997     47.272   -8.123 7.54e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 47.27 on 2171 degrees of freedom
## Multiple R-squared:  0.9996, Adjusted R-squared:  0.9996
## F-statistic: 2.906e+06 on 2 and 2171 DF,  p-value: < 2.2e-16

summary(fit2)

##
## Call:
## lm(formula = btc_market_price ~ poly(btc_market_cap, 3), data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -135.140  -25.472  -17.719    4.841  268.138
##
## Coefficients:
##                            Estimate Std. Error  t value Pr(>|t|)
## (Intercept)               9.121e+02  9.546e-01  955.389   <2e-16 ***
## poly(btc_market_cap, 3)1  1.140e+05  4.451e+01 2560.235   <2e-16 ***
## poly(btc_market_cap, 3)2 -3.840e+02  4.451e+01   -8.627   <2e-16 ***
## poly(btc_market_cap, 3)3  7.430e+02  4.451e+01   16.692   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 44.51 on 2170 degrees of freedom
```

```
## Multiple R-squared:  0.9997, Adjusted R-squared:  0.9997
## F-statistic: 2.185e+06 on 3 and 2170 DF,  p-value: < 2.2e-16
```

```
fita = lm(btc_market_price ~ btc_trade_volume + btc_hash_rate + btc_miners_re
venue + btc_difficulty + btc_cost_per_transaction + poly(btc_market_cap, 2),
data = train)
fitb = lm(btc_market_price ~ btc_trade_volume + btc_hash_rate + btc_miners_re
venue + btc_difficulty + btc_cost_per_transaction + poly(btc_market_cap, 3),
data = train)
fitc = lm(btc_market_price ~ btc_trade_volume + btc_hash_rate + btc_miners_re
venue + btc_difficulty + btc_cost_per_transaction + poly(btc_market_cap, 4),
data = train)
fitd = lm(btc_market_price ~ btc_trade_volume + btc_hash_rate + btc_miners_re
venue + btc_difficulty + btc_cost_per_transaction + poly(btc_market_cap, 5),
data = train)
anova(fita,fitb,fitc,fitd)
```

```
## Analysis of Variance Table
##
## Model 1: btc_market_price ~ btc_trade_volume + btc_hash_rate + btc_miners_
revenue +
##     btc_difficulty + btc_cost_per_transaction + poly(btc_market_cap,
##     2)
## Model 2: btc_market_price ~ btc_trade_volume + btc_hash_rate + btc_miners_
revenue +
##     btc_difficulty + btc_cost_per_transaction + poly(btc_market_cap,
##     3)
## Model 3: btc_market_price ~ btc_trade_volume + btc_hash_rate + btc_miners_
revenue +
##     btc_difficulty + btc_cost_per_transaction + poly(btc_market_cap,
##     4)
## Model 4: btc_market_price ~ btc_trade_volume + btc_hash_rate + btc_miners_
revenue +
##     btc_difficulty + btc_cost_per_transaction + poly(btc_market_cap,
##     5)
##   Res.Df     RSS Df Sum of Sq       F    Pr(>F)    
## 1   2166 1024462                                   
## 2   2165  690414  1    334048 1318.18 < 2.2e-16 ***
## 3   2164  650834  1     39580  156.18 < 2.2e-16 ***
## 4   2163  548141  1    102694  405.24 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
pred3 = predict(fit1, newdata = test)
mss3 = mean((test$btc_market_cap - pred3)^2)
mss3
```

```
## [1] 1.67828e+21
```

The mse for this is higher than simple and multiple linear regression. So we discard this model.

(Model iv) So this time I choose random forrest over GAM and other techniques such as splines because they all have polynomials involved and by intution I thought those would give somewhat the same result. The reason I chose the random forest dataset size is medium not large

```
set.seed(personal)
library(randomForest)

## randomForest 4.6-14

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

?randomForest
model4 <- randomForest(x = train[,c(4, 10, 11, 12, 15)], y = train$btc_market
_price, ntree = 501)
summary(model4)

##                 Length Class  Mode
## call               4   -none- call
## type               1   -none- character
## predicted       2174   -none- numeric
## mse              501   -none- numeric
## rsq              501   -none- numeric
## oob.times       2174   -none- numeric
## importance         5   -none- numeric
## importanceSD       0   -none- NULL
## localImportance    0   -none- NULL
## proximity          0   -none- NULL
## ntree              1   -none- numeric
## mtry               1   -none- numeric
## forest            11   -none- list
## coefs              0   -none- NULL
## y               2174   -none- numeric
## test               0   -none- NULL
## inbag              0   -none- NULL

pred4 = predict(model4, newdata = test)
mss4 = mean((test$btc_market_price - pred4)^2)
mss4

## [1] 35380.11
```

SO my intution that the random forest will produce better results on test is false. Hence i Will choose multiple linear regression over the other models as it has the less test error. Also this is because of overfitting. All the models fit into the training set but do poorly on test data. This can be expected as this is a to predict future bitcoin market prices.

Question 3:

Do the same approach as in question 2, but this time for a qualitative variable.

Answer 3 : As there is no binary variable for the above dataset lets create one :——

```
set.seed(personal)
mprice = mean(data$btc_market_price)
mprice

## [1] 901.8236

market_price <- rep("Greaterthanmean", nr)
market_price[data$btc_market_price < mprice] <- "Not_greater_than_mean"
data$market_price <- market_price
data$market_price <- as.factor(data$market_price)
```

(Model i) : Logistic Regression Model : Now Im changing the test set and then taking the 6 predictors for which my correlation was high.

```
set.seed(personal)
t_ind2 = sample(1:nr, 0.70 * nr, replace = FALSE)
train2 = data[t_ind2,]
test2 = data[-t_ind2,]
direction = test2$market_price
glm.fit = glm(market_price ~ btc_market_cap + btc_trade_volume + btc_hash_rat
e + btc_miners_revenue + btc_difficulty + btc_cost_per_transaction, data = tr
ain2, family = binomial)

## Warning: glm.fit: algorithm did not converge

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

prob = predict(glm.fit, test2, type = "response")
nrtest2 = round(0.30 * nr)
p11 = rep("Not_greater_than_mean", nrtest2)
p11[prob > 0.5] = "Greaterthanmean"
table(p11, direction)

##                          direction
## p11                        Greaterthanmean Not_greater_than_mean
##   Greaterthanmean                        2                   760
##   Not_greater_than_mean                108                     0
```

No need to calculate further as Logistic Regression performed very badly on this dataset.

(Model ii) LDA Model:—-

```
library(MASS)
set.seed(personal)
fit33 = lda(market_price ~ btc_market_cap + btc_trade_volume + btc_hash_rate
+ btc_miners_revenue + btc_difficulty + btc_cost_per_transaction, data = trai
n2)
p2 = predict(fit33, test2)$class
table(p2, direction)
```

```
##                           direction
## p2                         Greaterthanmean Not_greater_than_mean
##   Greaterthanmean                       86                     0
##   Not_greater_than_mean                 24                   760
```

```
#Misclassification Error rate
miscl = mean(p2 != direction)
miscl
```

```
## [1] 0.02758621
```

We see only 35 datapoints were misclassified. LDA gave a very low missclassification error.

(Model iii)

```
set.seed(personal)
fit44 = qda(market_price ~  btc_market_cap + btc_trade_volume + btc_hash_rate
+ btc_miners_revenue + btc_difficulty + btc_cost_per_transaction, data = trai
n2)
p3 = predict(fit44, test2)$class
table(p3, direction)
```

```
##                           direction
## p3                         Greaterthanmean Not_greater_than_mean
##   Greaterthanmean                      107                    17
##   Not_greater_than_mean                  3                   743
```

```
#Misclassification Error rate
miscl1 = mean(p3 != direction)
miscl1
```

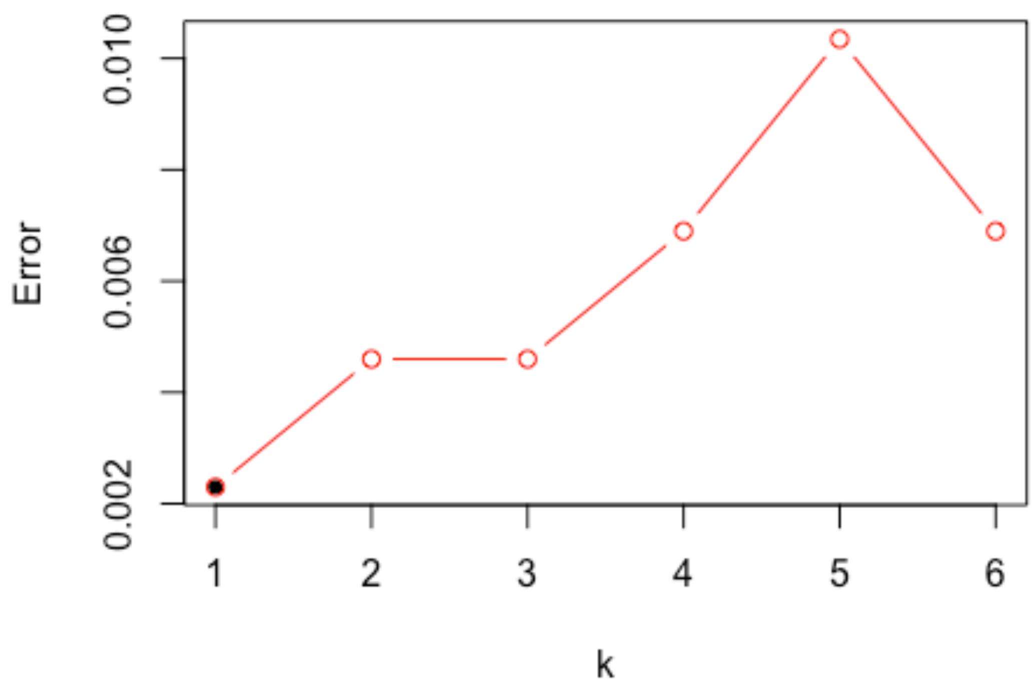
```
## [1] 0.02298851
```

QDA performs better than LDA and gives an error rate of 0.0206 which is considered to be very low.

(Model iv) KNN :

```
set.seed(personal)
library(class)
trainKNN = as.matrix(data.frame(train2$btc_market_cap, train2$btc_trade_volum
e, train2$btc_miners_revenue, train2$btc_difficulty, train2$btc_cost_per_tran
saction))
testKNN = as.matrix(data.frame(test2$btc_market_cap, test2$btc_trade_volume,
test2$btc_miners_revenue, test2$btc_difficulty, test2$btc_cost_per_transactio
n))
direction2 = train2$market_price
pKNN1 = knn(trainKNN, testKNN, direction2,k=1)
pKNN2 = knn(trainKNN, testKNN, direction2,k=3)
pKNN3 = knn(trainKNN, testKNN, direction2,k=5)
pKNN4 = knn(trainKNN, testKNN, direction2,k=10)
pKNN5 = knn(trainKNN, testKNN, direction2,k=25)
```

```
pKNN6 = knn(trainKNN, testKNN, direction2,k=67)

error1 = mean(pKNN1 != direction)
error2 = mean(pKNN2 != direction)
error3 = mean(pKNN3 != direction)
error4 = mean(pKNN4 != direction)
error5 = mean(pKNN5 != direction)
error6 = mean(pKNN6 != direction)
error = c(error1, error2, error3, error4, error5, error6)
plot(error, type='b', xlab="k", ylab="Error", col="red")
points(which.min(error), error[which.min(error)], pch=20, col="black")
```




```
error1

## [1] 0.002298851
```

Out of all the models KNN gives the lowest missclassification error on the test set . Hence KNN will be selectedf to give the best prediction for the data and the created binary variable #Question 4:

(Based on ISLR Chapter 9 #7) In this problem, you will use support vector approaches in order to predict whether a given car gets high or low gas mileage based on the Auto data set.

(a)

Create a binary variable that takes on a 1 for cars with gas mileage above the median, and a 0 for cars with gas mileage below the median.

```r
set.seed(personal)
library(ISLR)
attach(Auto)
#To make a binary variable use ifelse
median_mileage = median(Auto$mpg)
bin.var = ifelse(Auto$mpg > median_mileage, 1, 0)
Auto$binary.mpg = as.factor(bin.var)
```

(b)

Fit a support vector classifier to the data with various values of cost, in order to predict whether a car gets high or low gas mileage. Report the cross-validation errors associated with different values of this parameter. Comment on your results.

```r
set.seed(personal)
library(e1071)

## Warning: package 'e1071' was built under R version 3.5.2

x = tune(svm, binary.mpg~., data = Auto, kernel = "linear", ranges = list(cos
t = c(0.01, 0.1, 1, 5, 10, 100)))
summary(x)

##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.01275641
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-02 0.07391026 0.04245856
## 2 1e-01 0.05108974 0.04191745
## 3 1e+00 0.01275641 0.01344780
## 4 5e+00 0.01782051 0.01229997
## 5 1e+01 0.01782051 0.01229997
## 6 1e+02 0.03051282 0.01976051

print("The cross-validation error is minimized for cost = 1")

## [1] "The cross-validation error is minimized for cost = 1"
```

(c)

Now repeat for (b), this time using SVMs with radial and polynomial basis kernels, with different values of gamma and degree and cost. Comment on your results.

```
set.seed(personal)
y = tune(svm, binary.mpg ~., data = Auto, kernel = "polynomial", ranges = lis
t(cost = c(0.01, 0.1, 1, 5, 10, 100), degree = c(2, 3, 4)))
z = tune(svm, binary.mpg ~., data = Auto, kernel = "radial", ranges = list(co
st = c(0.01, 0.1, 1, 5, 10, 100), gamma = c(0.01, 0.1, 1, 5, 10, 100, 1000)))
summary(y)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##   100      2
##
## - best performance: 0.3060897
##
## - Detailed performance results:
##     cost degree     error dispersion
## 1  1e-02      2 0.5535897 0.04171890
## 2  1e-01      2 0.5535897 0.04171890
## 3  1e+00      2 0.5535897 0.04171890
## 4  5e+00      2 0.5535897 0.04171890
## 5  1e+01      2 0.4844231 0.11172253
## 6  1e+02      2 0.3060897 0.05506460
## 7  1e-02      3 0.5535897 0.04171890
## 8  1e-01      3 0.5535897 0.04171890
## 9  1e+00      3 0.5535897 0.04171890
## 10 5e+00      3 0.5535897 0.04171890
## 11 1e+01      3 0.5535897 0.04171890
## 12 1e+02      3 0.3445513 0.06156313
## 13 1e-02      4 0.5535897 0.04171890
## 14 1e-01      4 0.5535897 0.04171890
## 15 1e+00      4 0.5535897 0.04171890
## 16 5e+00      4 0.5535897 0.04171890
## 17 1e+01      4 0.5535897 0.04171890
## 18 1e+02      4 0.5535897 0.04171890
```

```
summary(z)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
```

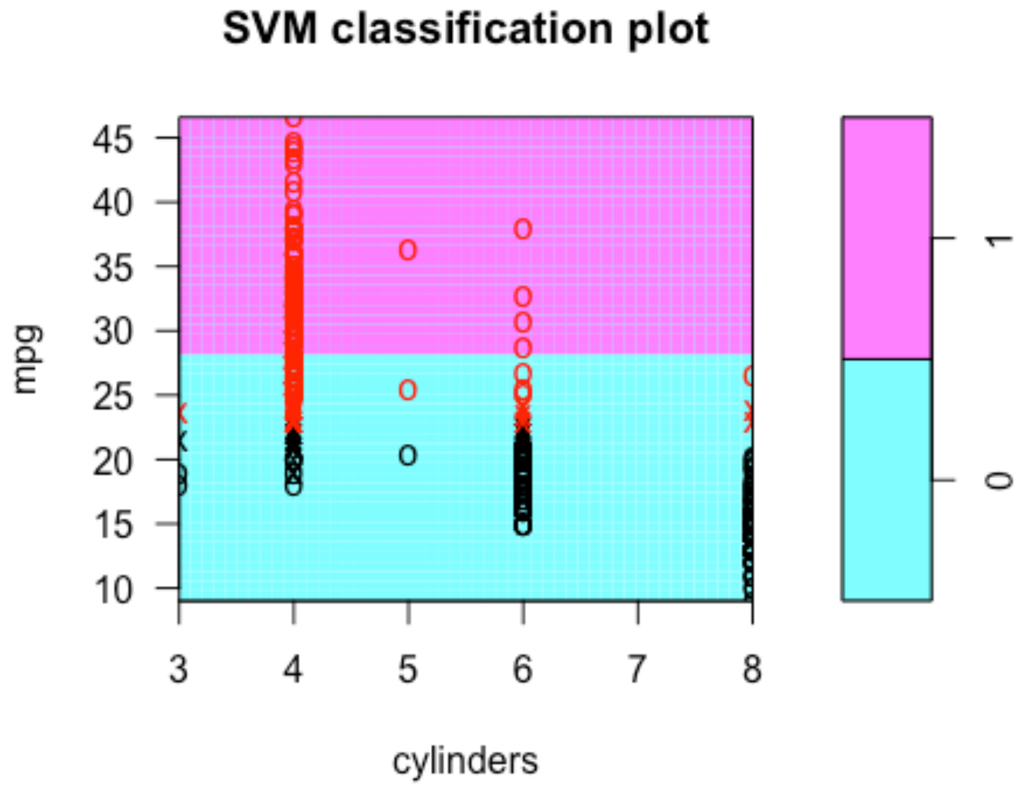
```
##
## - best parameters:
##  cost gamma
##   100  0.01
##
## - best performance: 0.01532051
##
## - Detailed performance results:
##      cost gamma      error dispersion
## 1  1e-02 1e-02 0.57647436 0.03687622
## 2  1e-01 1e-02 0.08923077 0.06276147
## 3  1e+00 1e-02 0.07403846 0.04271928
## 4  5e+00 1e-02 0.05115385 0.04018094
## 5  1e+01 1e-02 0.02557692 0.01709522
## 6  1e+02 1e-02 0.01532051 0.01788871
## 7  1e-02 1e-01 0.19852564 0.08608860
## 8  1e-01 1e-01 0.08166667 0.05510683
## 9  1e+00 1e-01 0.05628205 0.03983401
## 10 5e+00 1e-01 0.02814103 0.01893035
## 11 1e+01 1e-01 0.02044872 0.02020886
## 12 1e+02 1e-01 0.02301282 0.02244393
## 13 1e-02 1e+00 0.57647436 0.03687622
## 14 1e-01 1e+00 0.57647436 0.03687622
## 15 1e+00 1e+00 0.06378205 0.03674375
## 16 5e+00 1e+00 0.06641026 0.03678591
## 17 1e+01 1e+00 0.06641026 0.03678591
## 18 1e+02 1e+00 0.06641026 0.03678591
## 19 1e-02 5e+00 0.57647436 0.03687622
## 20 1e-01 5e+00 0.57647436 0.03687622
## 21 1e+00 5e+00 0.51762821 0.05340278
## 22 5e+00 5e+00 0.51256410 0.06327615
## 23 1e+01 5e+00 0.51256410 0.06327615
## 24 1e+02 5e+00 0.51256410 0.06327615
## 25 1e-02 1e+01 0.57647436 0.03687622
## 26 1e-01 1e+01 0.57647436 0.03687622
## 27 1e+00 1e+01 0.54839744 0.05805619
## 28 5e+00 1e+01 0.53814103 0.05381159
## 29 1e+01 1e+01 0.53814103 0.05381159
## 30 1e+02 1e+01 0.53814103 0.05381159
## 31 1e-02 1e+02 0.57647436 0.03687622
## 32 1e-01 1e+02 0.57647436 0.03687622
## 33 1e+00 1e+02 0.57647436 0.03687622
## 34 5e+00 1e+02 0.57647436 0.03687622
## 35 1e+01 1e+02 0.57647436 0.03687622
## 36 1e+02 1e+02 0.57647436 0.03687622
## 37 1e-02 1e+03 0.57647436 0.03687622
## 38 1e-01 1e+03 0.57647436 0.03687622
## 39 1e+00 1e+03 0.57647436 0.03687622
## 40 5e+00 1e+03 0.57647436 0.03687622
```

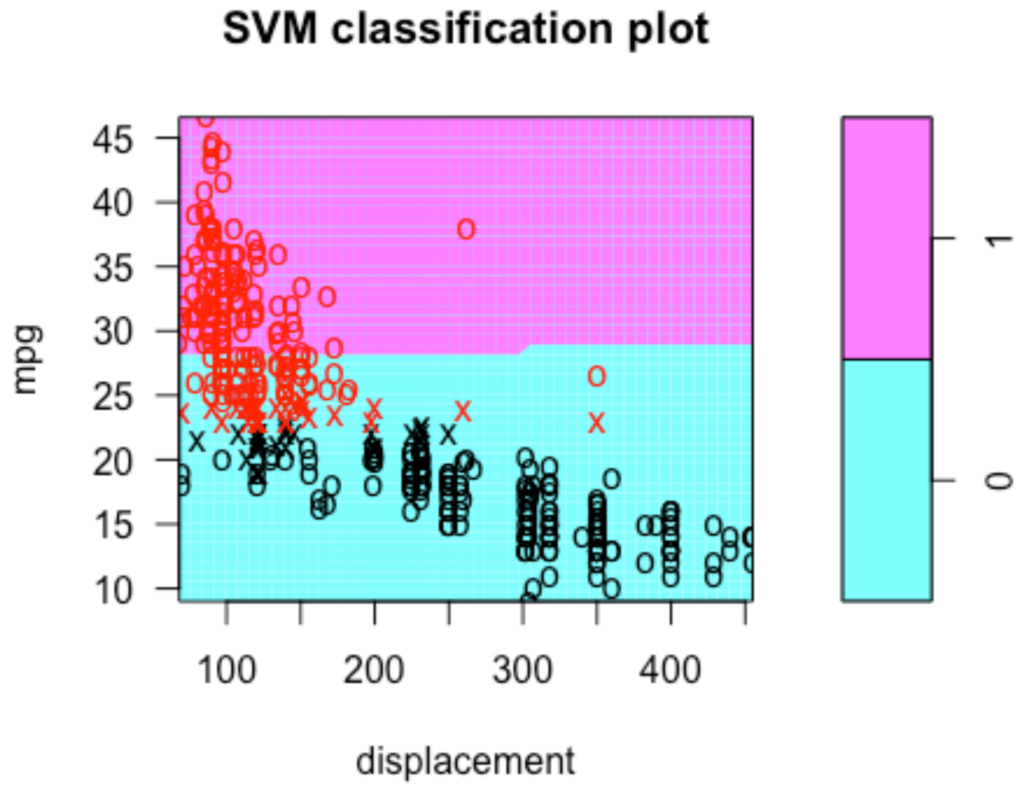
```
## 41 1e+01 1e+03 0.57647436 0.03687622
## 42 1e+02 1e+03 0.57647436 0.03687622

print("The lowest cross-validation error for a polynomial kernel, is obtained
for a degree of 2 and a cost of 100.")

## [1] "The lowest cross-validation error for a polynomial kernel, is obtaine
d for a degree of 2 and a cost of 100."

print("The lowest cross-validation error for a radial kernel, is obtained for
a gamma for 0.01 and a cost of 100.")

## [1] "The lowest cross-validation error for a radial kernel, is obtained fo
r a gamma for 0.01 and a cost of 100."
```

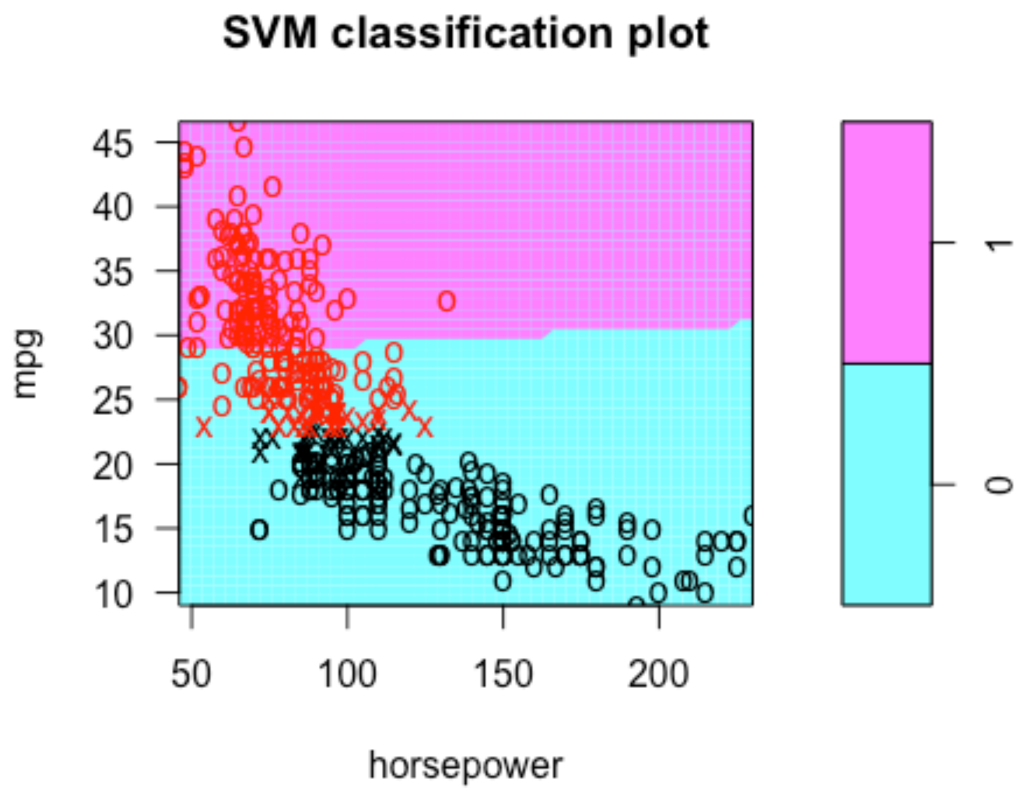
(d)

Make some plots to back up your assertions in (b) and (c). Hint: In the lab, we used the plot() function for svm objects only in cases with p=2 When p>2,you can use the plot() function to create plots displaying pairs of variables at a time. Essentially, instead of typing plot(svmfit , dat) where svmfit contains your fitted model and dat is a data frame containing your data, you can type plot(svmfit , dat, x1~x4) in order to plot just the first and fourth variables. However, you must replace x1 and x4 with the correct variable names. To find out more, type ?plot.svm.

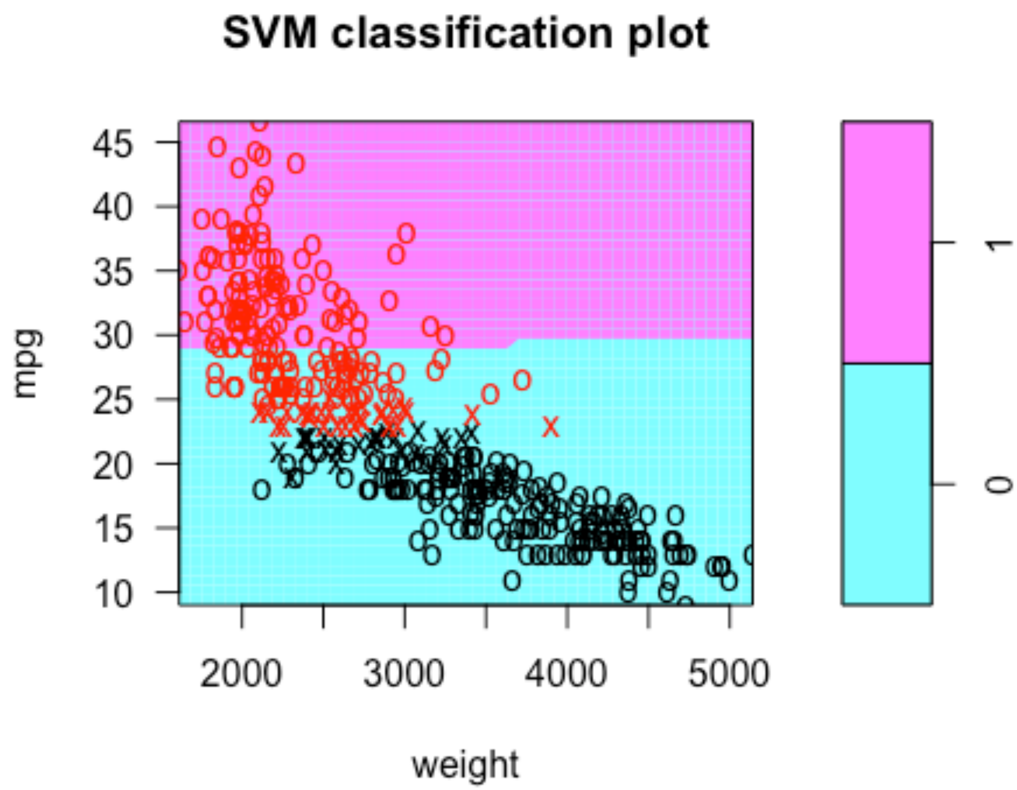
```
set.seed(personal)
svm_linear = svm(binary.mpg~., data = Auto, kernel = "linear", cost = 1)
svm_polynomial = svm(binary.mpg~., data = Auto, kernel = "polynomial", cost =
100, degree = 2)
svm_radial = svm(binary.mpg~., data = Auto, kernel = "radial", cost = 100, ga
mma = 0.01)

plottings = function(fitting) {
for (i in names(Auto)[!(names(Auto) %in% c("mpg", "binary.mpg", "name"))]) {
        plot(fitting, Auto, as.formula(paste("mpg~", i, sep = "")))
 }
}
plottings(svm_linear)
```

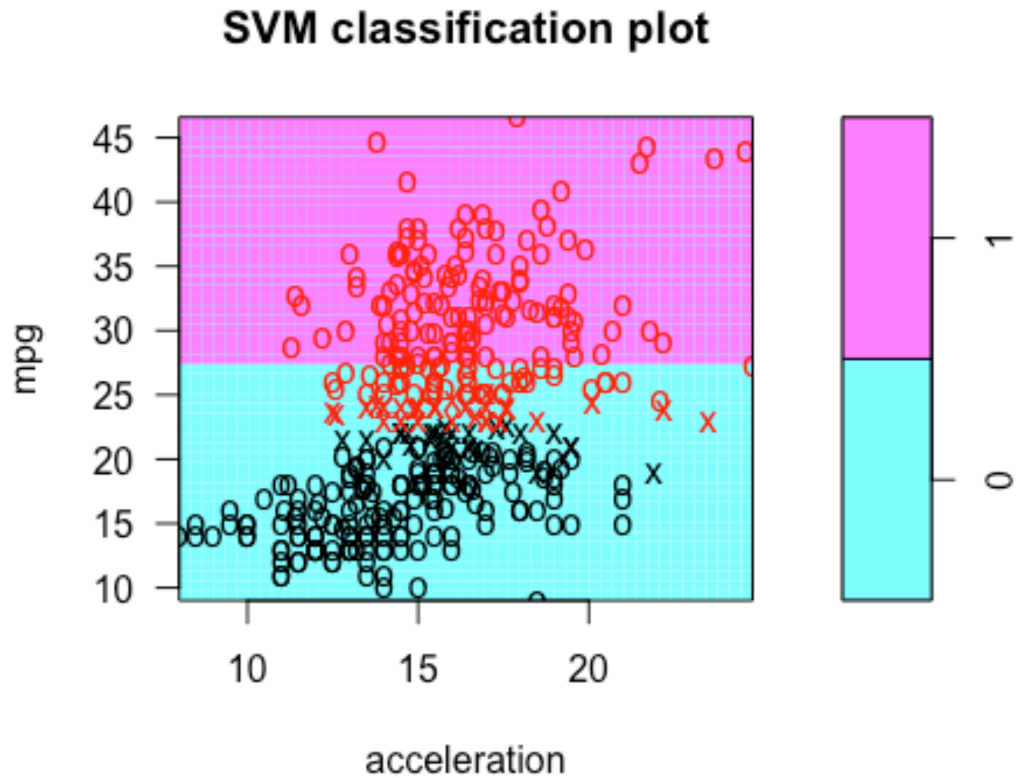

SVM classification plot
mpg
cylinders
45
40
35
30
25
20
15
10
3
4
5
6
7
8
1
0


SVM classification plot
mpg
displacement
45
40
35
30
25
20
15
10
100
200
300
400
1
0

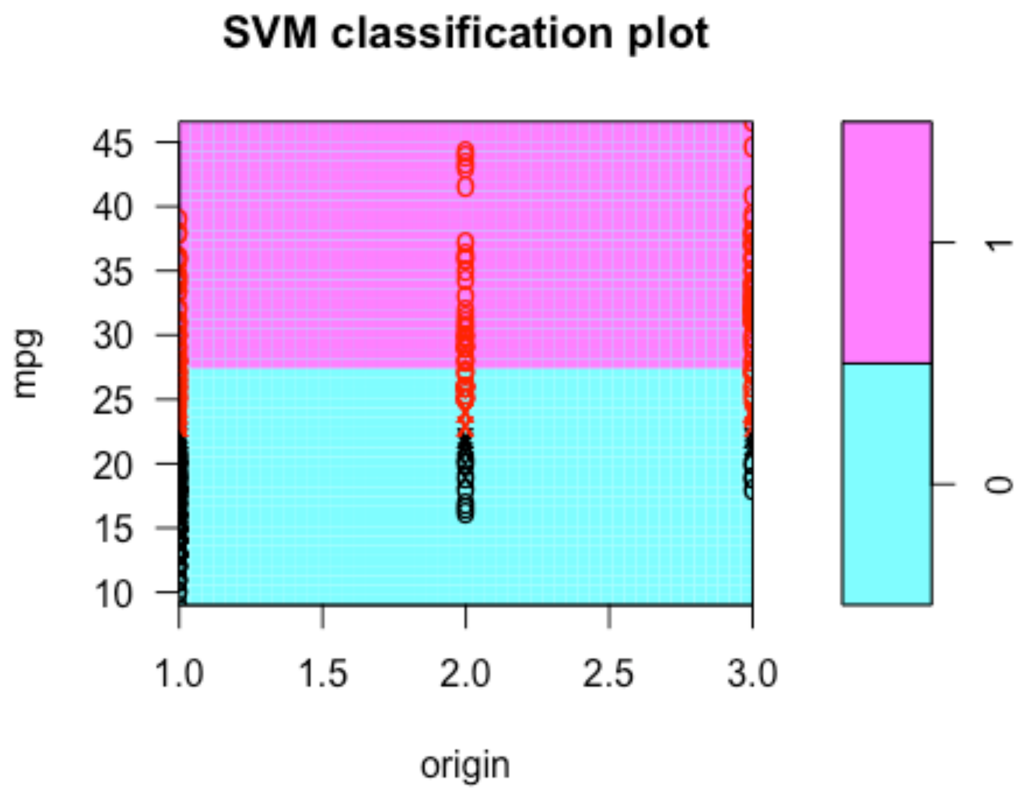


SVM classification plot
mpg
45
40
35
30
25
20
15
10
50
100
150
200
horsepower
1
0

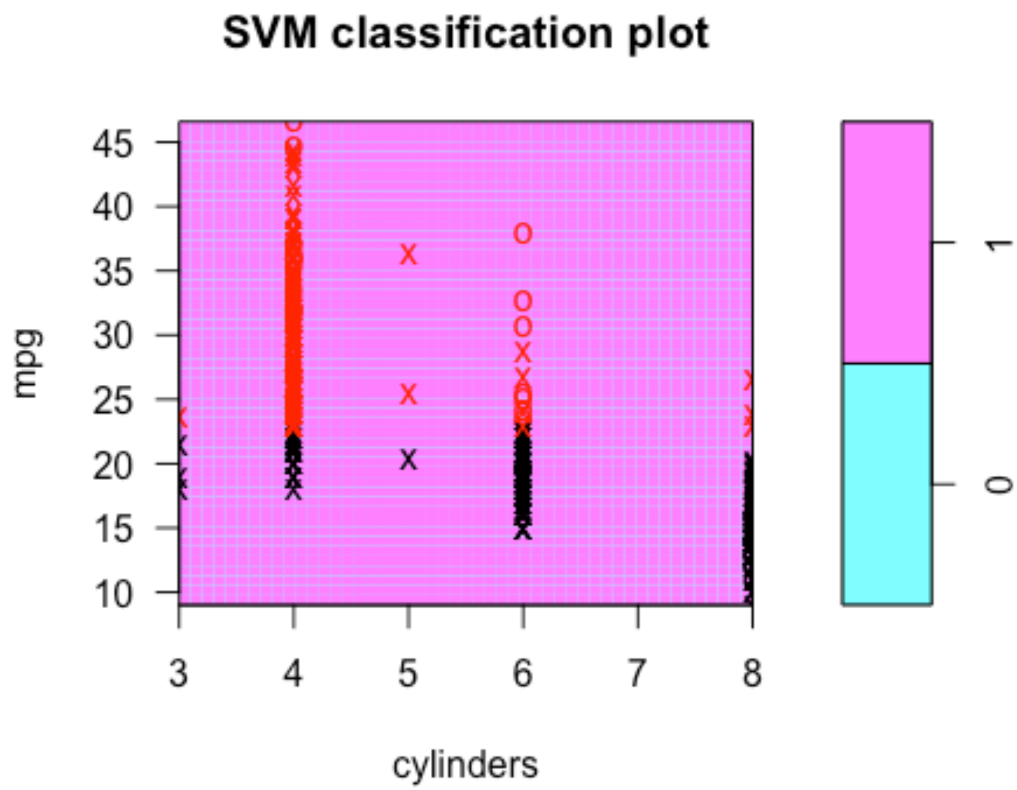

SVM classification plot
mpg
45
40
35
30
25
20
15
10
2000
3000
4000
5000
weight
1
0


SVM classification plot
mpg
acceleration
45
40
35
30
25
20
15
10
10
15
20
1
0

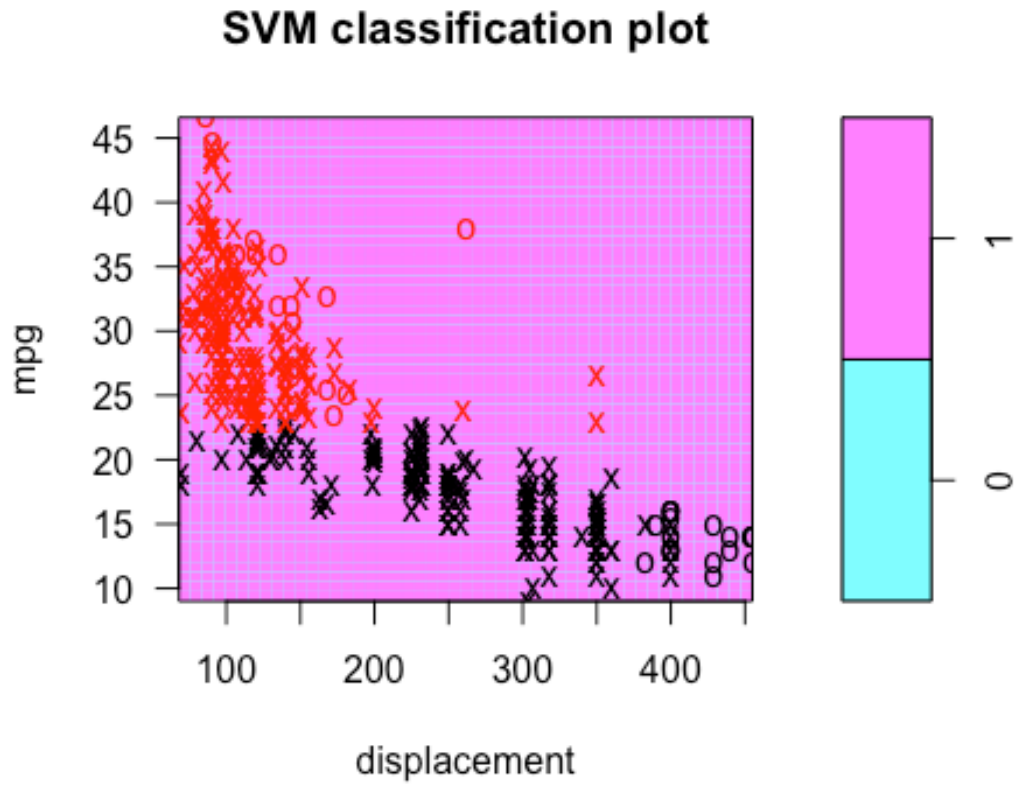
SVM classification plot
mpg
year
45
40
35
30
25
20
15
10
70
72
74
76
78
80
82
1
0

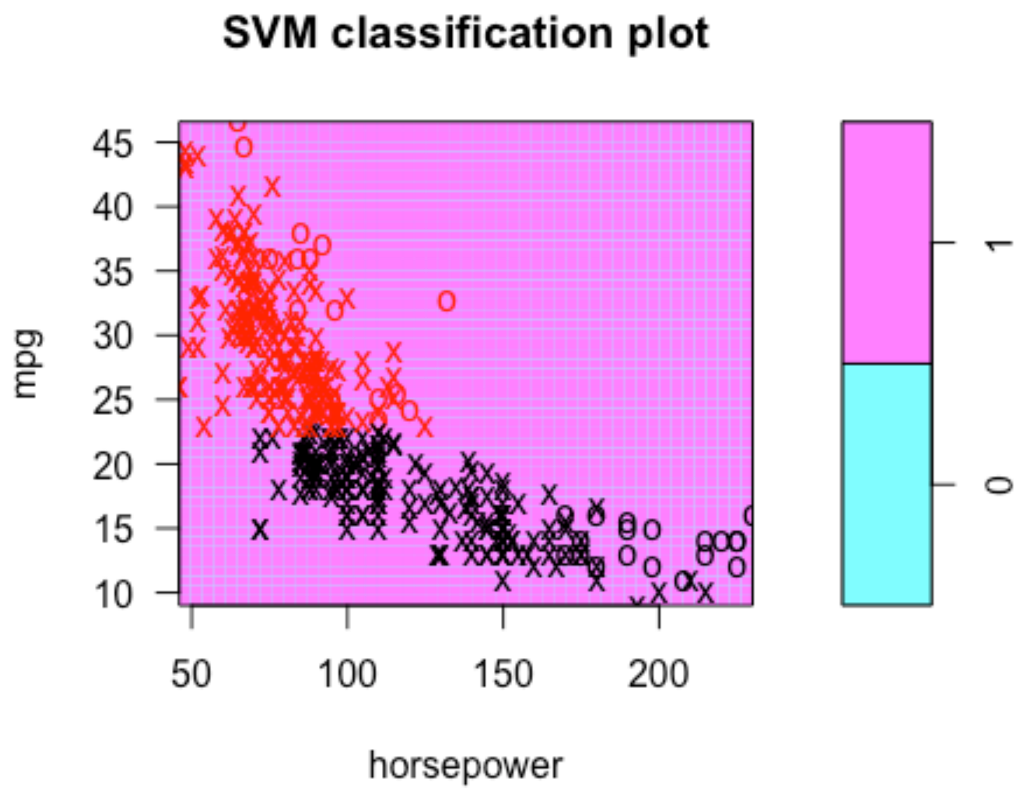


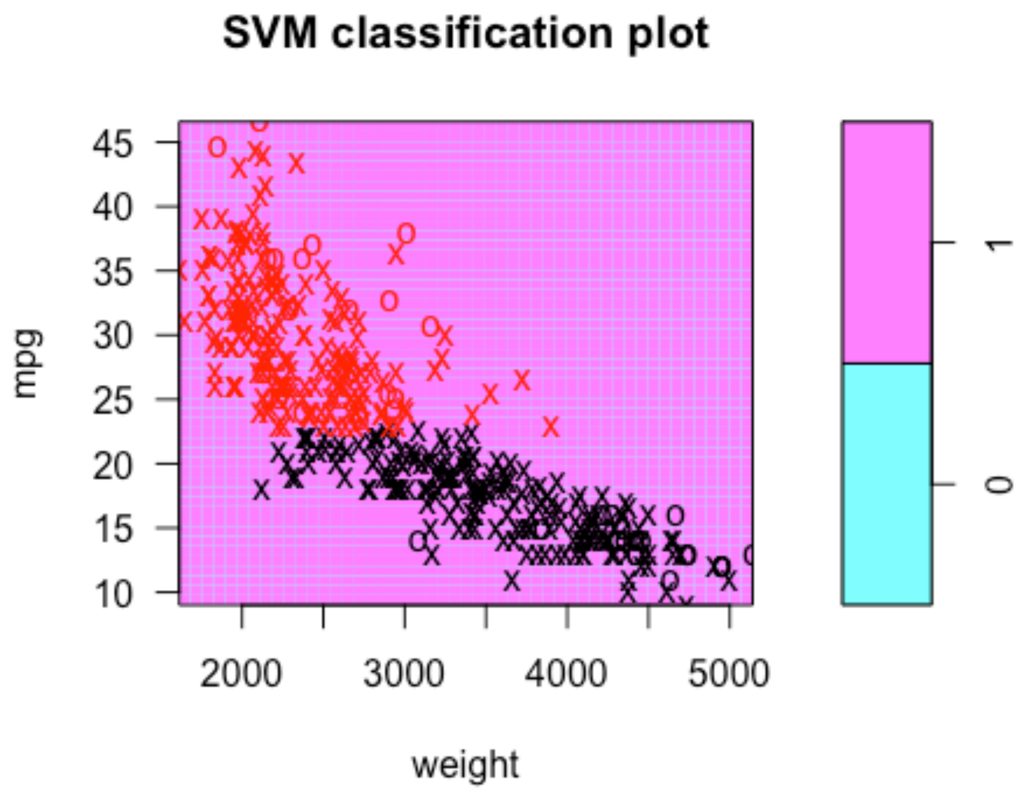

```
plottings(svm_polynomial)
```

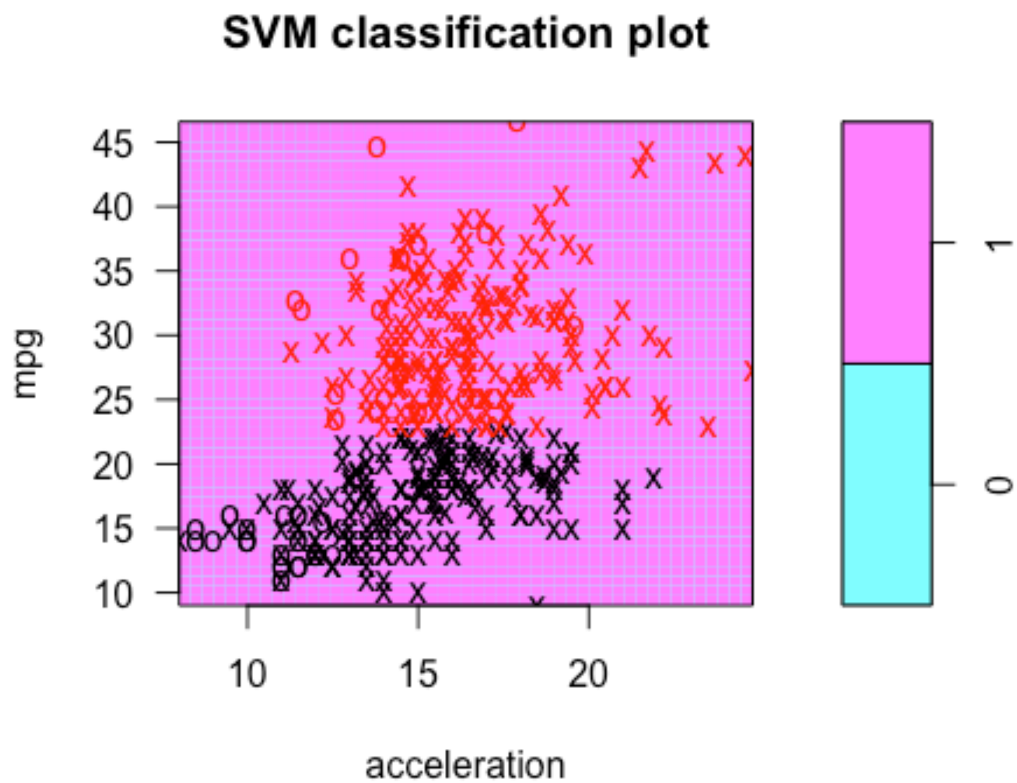


SVM classification plot
mpg
cylinders
45
40
35
30
25
20
15
10
3
4
5
6
7
8
1
0

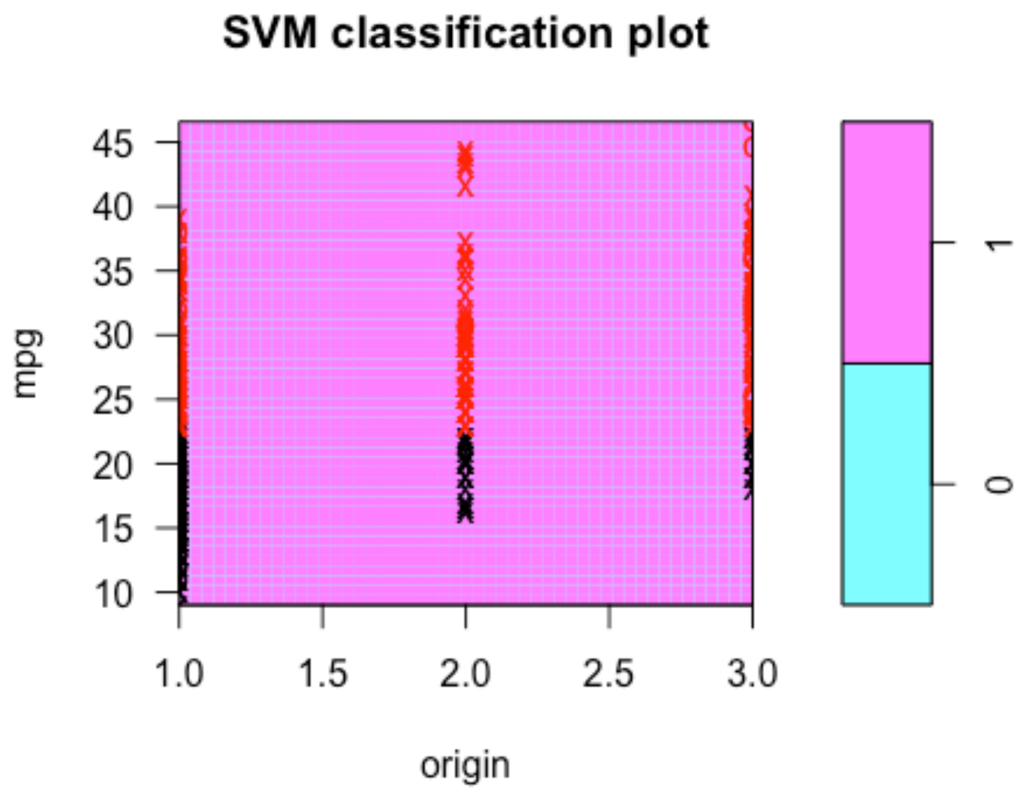

SVM classification plot
mpg
45
40
35
30
25
20
15
10
100
200
300
400
displacement
1
0


SVM classification plot
mpg
horsepower
45
40
35
30
25
20
15
10
50
100
150
200
1
0

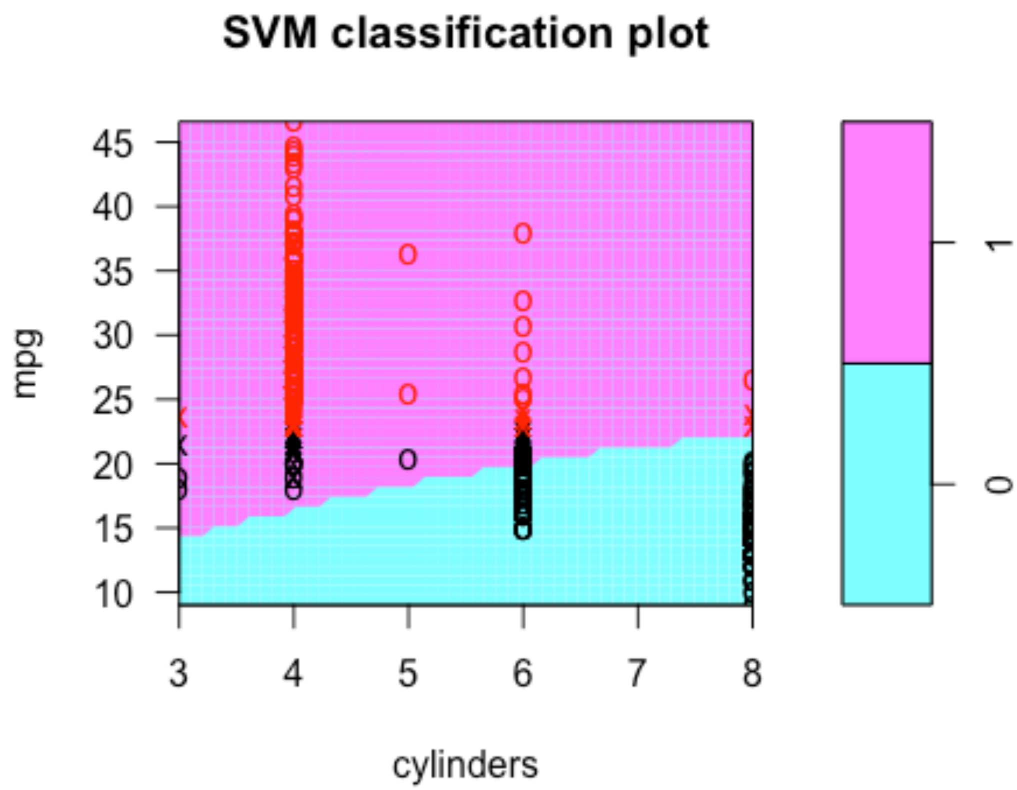

SVM classification plot
mpg
weight
45
40
35
30
25
20
15
10
2000
3000
4000
5000
1
0

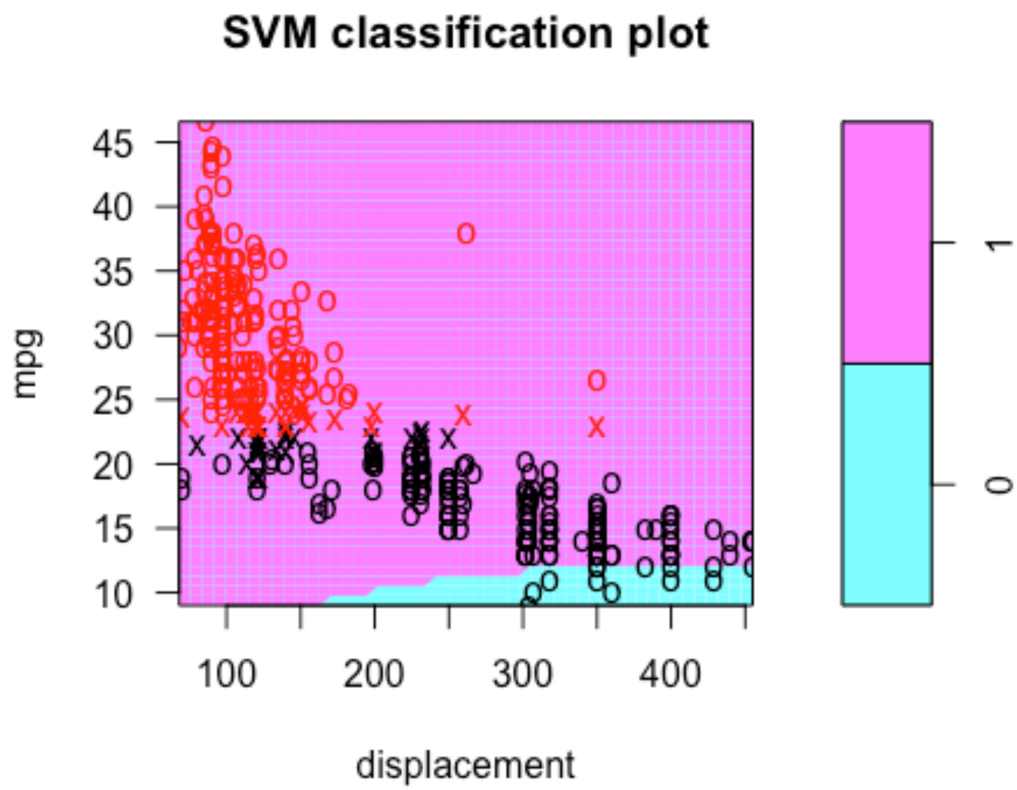

SVM classification plot
mpg
acceleration
45
40
35
30
25
20
15
10
10
15
20
1
0

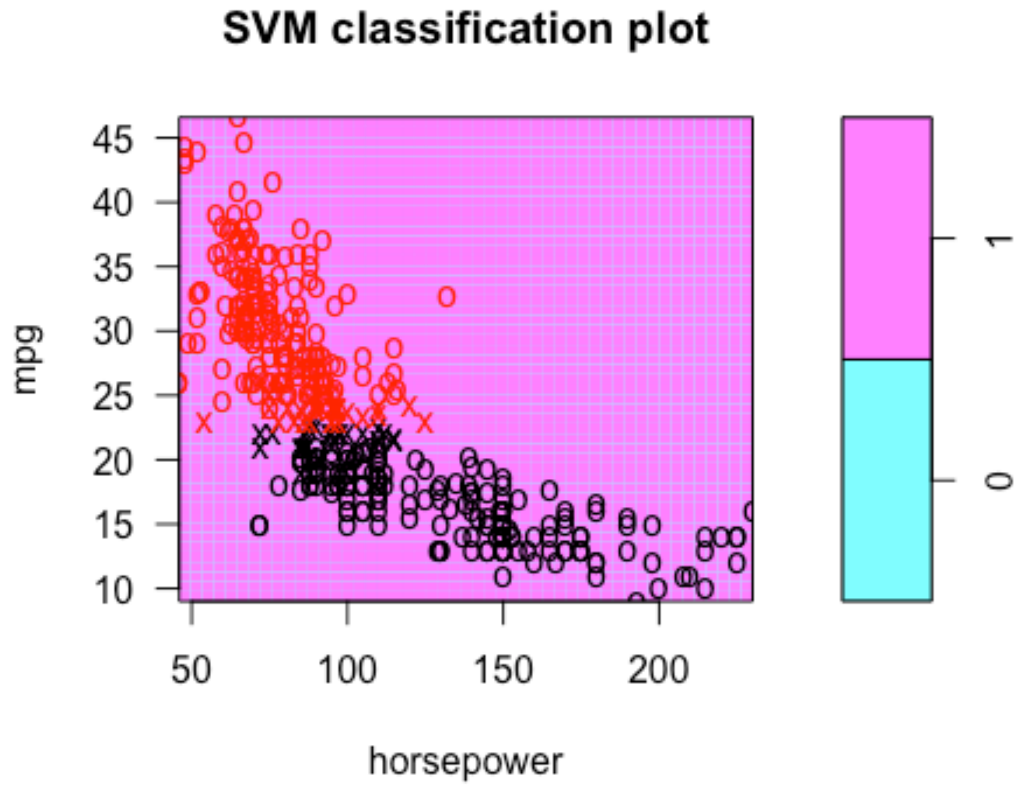

SVM classification plot
mpg
year
45
40
35
30
25
20
15
10
70
72
74
76
78
80
82
1
0



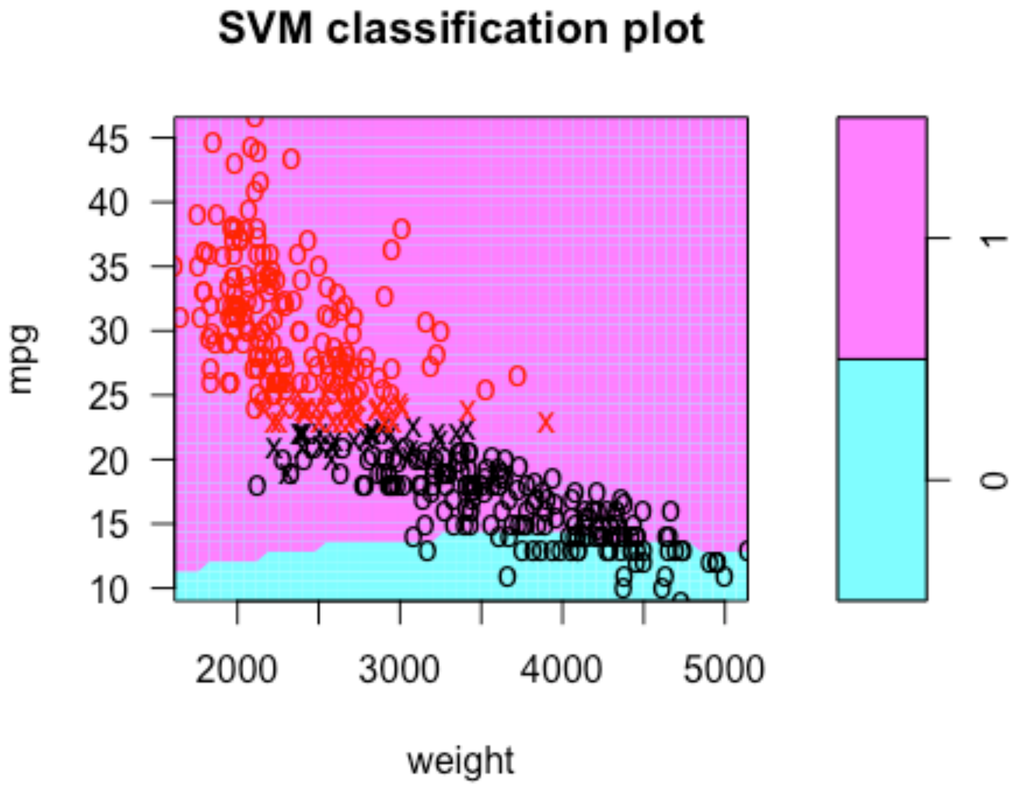

```
plottings(svm_radial)
```

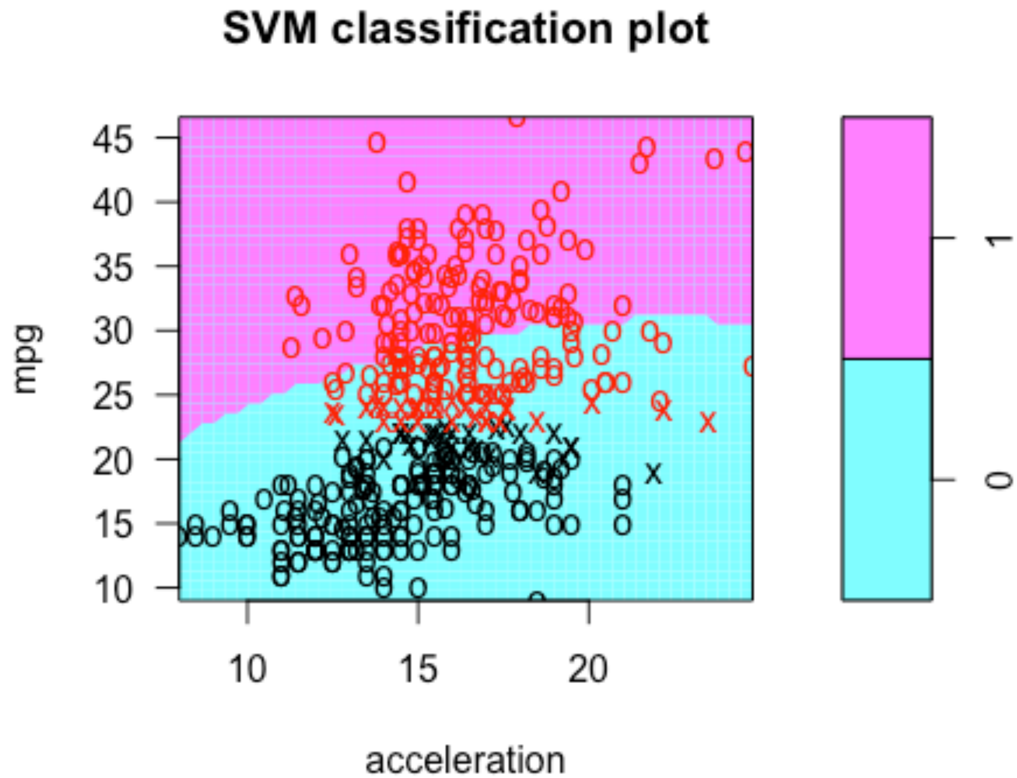

SVM classification plot
mpg
45
40
35
30
25
20
15
10
3
4
5
6
7
8
cylinders
1
0


SVM classification plot
mpg
45
40
35
30
25
20
15
10
100
200
300
400
displacement
1
0



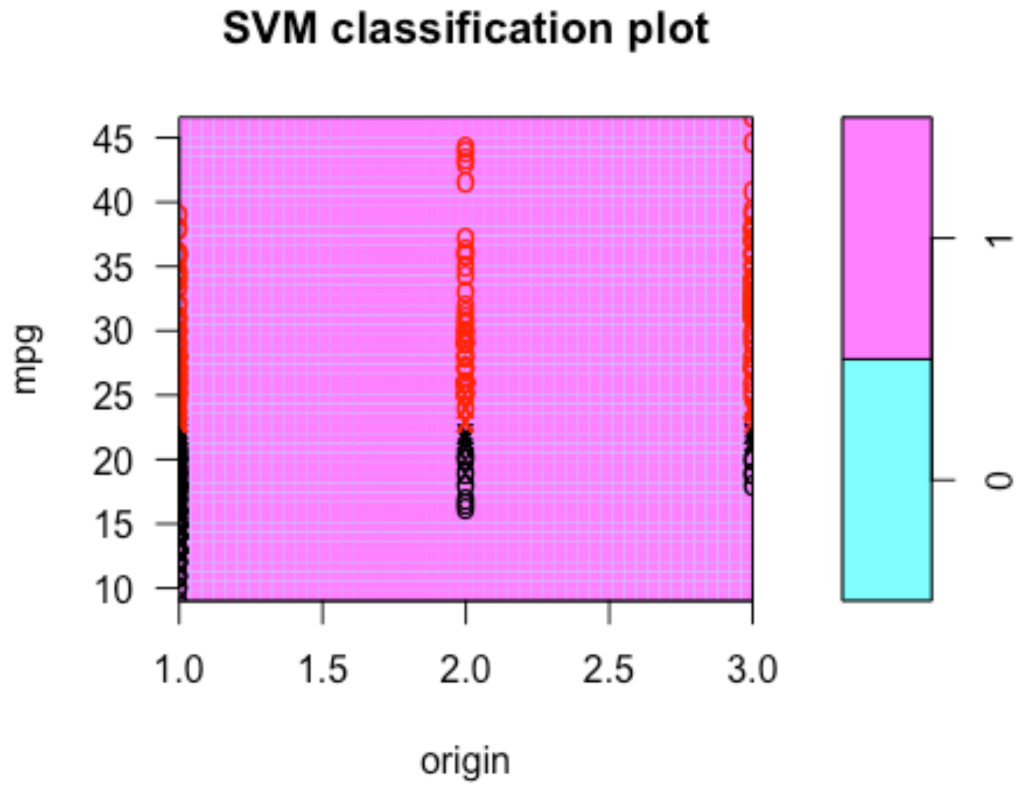
SVM classification plot
mpg
horsepower
45
40
35
30
25
20
15
10
50
100
150
200
1
0


SVM classification plot
mpg
weight
45
40
35
30
25
20
15
10
2000
3000
4000
5000
1
0



SVM classification plot
mpg
acceleration
45
40
35
30
25
20
15
10
10
15
20
1
0


SVM classification plot
mpg
45
40
35
30
25
20
15
10
70
72
74
76
78
80
82
year
1
0


SVM classification plot
mpg
45
40
35
30
25
20
15
10
1.0
1.5
2.0
2.5
3.0
origin
1
0