FE590. Assignment #4.

Yifu He

2019-10-04

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.

```
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
change
#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.

```
Data=read.csv("Data_Set.csv")
Data=Data[,-1]
attach(Data)
library(leaps)
library(gam)

## Loading required package: splines

## Loading required package: foreach
```

```
## Loaded gam 1.16

library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 2.0-16

library(randomForest)

## randomForest 4.6-14

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

library(tree)
library(boot)
library(class)
library(MASS)
library(e1071)

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

The data are collected through various sources and complied by me. The soybean, corn, and oats price are obtained through macrotrends.net and validated through bloomberg.com. The weather data are obtained through the NCDC. The data set contains the monthly value of different variables representing factors that may affect the price of soybean future contract. The variables are:

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.

Solution

The six chosen models for the testing of the data would be: Linear Regression, Polynomial Linear regression, Lasso, GAMs with Polynomial, GAMs with Natural Spline, and Tree method.

The confidence level used in this test is $\alpha = 0.05$.

The validation method used will be validation set.

```
set.seed(personal)
quant=Data[,-2]
n=length(Soy)
```

```
sampling=sample(n,n/2)
quant.train=quant[sampling,]
quant.test=quant[-sampling,]
```

Linear Regression

First off, I use regsubset to determine the best possible combination of parameters based on these 9 variables.

```
reg.fit=regsubsets(Soy~.,data=quant.train,nvmax=9)
result=data.frame(rank(summary(reg.fit)$cp),
                  rank(summary(reg.fit)$bic),
                  rank(-summary(reg.fit)$adjr))
colnames(result)=c("Mallow's Cp", "BIC","Adj R^2")
result
```

```
##   Mallow's Cp BIC Adj R^2
## 1           9   9       9
## 2           8   6       8
## 3           5   2       7
## 4           1   1       5
## 5           2   3       3
## 6           3   4       1
## 7           4   5       2
## 8           6   7       4
## 9           7   8       6
```

With the result in hands, we can see that the model with 5 parameters are the best choice with overall good result in the three criteria. The five chosen parameters are:

```
coefmodel=coef(reg.fit,id=5)
names(coefmodel)
```

```
## [1] "(Intercept)" "SouthTemp"   "SouthRain"   "Oats"        "Corn"
## [6] "USD"
```

```
lm.fit=lm(Soy~UpMWTemp+SouthRain+Corn+Oats+USD,data=quant.train)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = Soy ~ UpMWTemp + SouthRain + Corn + Oats + USD,
##     data = quant.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.46930 -0.52825 -0.06889  0.42774  3.01514
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.290922   0.827578   5.185 4.25e-07 ***
```

```
## UpMWTemp     0.005557   0.003161   1.758   0.0799 .
## SouthRain    0.063064   0.048648   1.296   0.1960
## Corn         1.280781   0.109839  11.661  < 2e-16 ***
## Oats         1.051339   0.175367   5.995 6.52e-09 ***
## USD         -0.033170   0.007380  -4.494 1.04e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9809 on 269 degrees of freedom
## Multiple R-squared:  0.858,  Adjusted R-squared:  0.8553
## F-statistic:   325 on 5 and 269 DF,  p-value: < 2.2e-16
```

Looking at the summary, I remove the UpMWTemp as it is not statistically significant

```
lm.fit=lm(Soy~SouthRain+Corn+Oats+USD,data=quant.train)
summary(lm.fit)

##
## Call:
## lm(formula = Soy ~ SouthRain + Corn + Oats + USD, data = quant.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.5787 -0.5648 -0.0484  0.4605  3.1231
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.462971   0.824944   5.410 1.39e-07 ***
## SouthRain    0.088149   0.046687   1.888   0.0601 .
## Corn         1.269421   0.110072  11.533  < 2e-16 ***
## Oats         1.073700   0.175580   6.115 3.37e-09 ***
## USD         -0.033355   0.007408  -4.503 9.99e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9847 on 270 degrees of freedom
## Multiple R-squared:  0.8564, Adjusted R-squared:  0.8542
## F-statistic: 402.4 on 4 and 270 DF,  p-value: < 2.2e-16
```

The new linear regression shows great promising as all of the variable is statistically significance with $\alpha = 5\%$.

Then, we put calculate the testing and the training MSE for the regression:

```
lm.pred=predict(lm.fit,quant.train)
mean((Soy-lm.pred)^2)

## [1] 12.67291

lm.pred=predict(lm.fit,quant.test)
mean((Soy-lm.pred)^2)
```

```
## [1] 11.80402
```

With the above result, we can see that the model actually improve in testing data set with the MSE reduce to 10.7786.

Polynomial Regression

I apply the same method from the Linear Regression into the Polynomial Regression. First, I would use the exhaustive method to find the best combination

```
reg.fit=regsubsets(Soy~poly(OHRain,4)
                   +poly(OHTemp,4)
                   +poly(UpMWRain,4)
                   +poly(UpMWTemp,4)
                   +poly(SouthRain,4)
                   +poly(SouthTemp,4)
                   +poly(Corn,4)
                   +poly(Oats,4)
                   +poly(USD,4),
                data = quant.train,nvmax=36)
result=data.frame(rank(summary(reg.fit)$cp),
                  rank(summary(reg.fit)$bic),
                  rank(-summary(reg.fit)$adjr))
colnames(result)=c("Mallow's Cp", "BIC","Adj R^2")
result
```

```
##    Mallow's Cp BIC Adj R^2
## 1           36  17      36
## 2           30   9      35
## 3           19   4      34
## 4           13   1      29
## 5            9   2      25
## 6            6   3      21
## 7            1   5      18
## 8            2   6      16
## 9            4   7      14
## 10           3   8      12
## 11           5  10      10
## 12           7  11       8
## 13           8  12       7
## 14          10  13       4
## 15          11  14       1
## 16          12  15       2
## 17          14  16       3
## 18          15  18       5
## 19          16  19       6
## 20          17  20       9
## 21          18  21      11
## 22          20  22      13
## 23          21  23      15
```

```
## 24          22  24      17
## 25          23  25      19
## 26          24  26      20
## 27          25  27      22
## 28          26  28      23
## 29          27  29      24
## 30          28  30      26
## 31          29  31      27
## 32          31  32      28
## 33          32  33      30
## 34          33  34      31
## 35          34  35      32
## 36          35  36      33
```

The best method to choose is the 12-parameters one.

```
coefmodel=coef(reg.fit,id=12)
names(coefmodel)
```

```
##  [1] "(Intercept)"         "poly(OHRain, 4)4"    "poly(OHTemp, 4)3"
##  [4] "poly(SouthRain, 4)3" "poly(SouthTemp, 4)1" "poly(SouthTemp, 4)4"
##  [7] "poly(Corn, 4)1"      "poly(Corn, 4)2"      "poly(Corn, 4)4"
## [10] "poly(Oats, 4)1"      "poly(Oats, 4)2"      "poly(USD, 4)1"
## [13] "poly(USD, 4)4"
```

```
poly.fit=lm(Soy~I(OHRain)^3
                +OHTemp
                +I(UpMWRain)^3
                +UpMWTemp
                +SouthRain
                +I(SouthRain)^3
                +I(SouthTemp)^2
                +poly(Corn,2)
                +poly(Oats,2)
                +USD,
                data=quant.train)
summary(poly.fit)
```

```
##
## Call:
## lm(formula = Soy ~ I(OHRain)^3 + OHTemp + I(UpMWRain)^3 + UpMWTemp +
##     SouthRain + I(SouthRain)^3 + I(SouthTemp)^2 + poly(Corn,
##     2) + poly(Oats, 2) + USD, data = quant.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.34949 -0.55846 -0.02194  0.42563  3.12989
##
## Coefficients: (1 not defined because of singularities)
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     8.830981   1.083340   8.152 1.47e-14 ***
```

```
## I(OHRain)      -0.005360   0.061360  -0.087   0.9305
## OHTemp         -0.011745   0.044626  -0.263   0.7926
## I(UpMWRain)    -0.084940   0.080293  -1.058   0.2911
## UpMWTemp       -0.013380   0.027745  -0.482   0.6300
## SouthRain       0.096657   0.058134   1.663   0.0976 .
## I(SouthRain)          NA         NA      NA       NA
## I(SouthTemp)    0.045874   0.032119   1.428   0.1544
## poly(Corn, 2)1 24.340650   2.369576  10.272  < 2e-16 ***
## poly(Corn, 2)2 -2.328029   1.442344  -1.614   0.1077
## poly(Oats, 2)1 13.712989   2.282369   6.008 6.22e-09 ***
## poly(Oats, 2)2  2.044049   1.549935   1.319   0.1884
## USD            -0.031286   0.007503  -4.170 4.14e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.981 on 263 degrees of freedom
## Multiple R-squared:  0.8611, Adjusted R-squared:  0.8553
## F-statistic: 148.2 on 11 and 263 DF,  p-value: < 2.2e-16
```

With this result, I only pick Corn and Oats with the degree to 2 and USD for the polynomial model

```
poly.fit=lm(Soy~poly(Corn,2)+poly(Oats,2)+USD,data=quant.train)
summary(poly.fit)

##
## Call:
## lm(formula = Soy ~ poly(Corn, 2) + poly(Oats, 2) + USD, data =
quant.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.5684 -0.5636 -0.0451  0.4688  3.1886
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    10.563157   0.721236  14.646  < 2e-16 ***
## poly(Corn, 2)1 24.204557   2.342207  10.334  < 2e-16 ***
## poly(Corn, 2)2 -2.471439   1.437976  -1.719   0.0868 .
## poly(Oats, 2)1 13.558368   2.256230   6.009 6.04e-09 ***
## poly(Oats, 2)2  2.074362   1.517399   1.367   0.1728
## USD            -0.031837   0.007518  -4.235 3.15e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9875 on 269 degrees of freedom
## Multiple R-squared:  0.8561, Adjusted R-squared:  0.8534
## F-statistic:   320 on 5 and 269 DF,  p-value: < 2.2e-16
```

And then to the cross-validation test:

```
poly.pred=predict(poly.fit,quant.train)
mean((Soy-poly.pred)^2)

## [1] 12.67891

poly.pred=predict(poly.fit,quant.test)
mean((Soy-poly.pred)^2)

## [1] 11.74958
```

The polynomial model comes with a good improvement through the testing phase with the testing MSE of 10.5840

Lasso

First off, I build a grid of different lambda to choose which lambda yield the smallest cross-validation error.

```
x=model.matrix(Soy~.,quant.train)[,-1]
y=quant.train$Soy
grid=10^seq(10,-2,length=100)
lasso.fit=glmnet(x,y,alpha = 1,lambda = grid)
cv.lasso=cv.glmnet(x,y,alpha=1)
bestlasso=cv.lasso$lambda.min
bestlasso

## [1] 0.02685221
```

The answer is the lasso with $\lambda = 0.01775847$.

Apply this λ into the lasso model and calculate the testing MSE

```
lasso.fit.b=glmnet(x,y,alpha=1,lambda = bestlasso)
coef(lasso.fit.b)

## 10 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept)  4.031983947
## OHTemp       .
## UpMWTemp     .
## SouthTemp    0.007127496
## OHRain       .
## UpMWRain     .
## SouthRain    0.044319863
## Oats         1.046313666
## Corn         1.263569431
## USD         -0.031334630

x.test=model.matrix(Soy~.,quant.test)[,-1]
lasso.pred=predict(lasso.fit,s=bestlasso,x.test)
mean((quant.test$Soy-lasso.pred)^2)
```

```
## [1] 1.111262
```

The testing MSE is extraordinary good compared to the usual Linear Regression method.

GAM w/ polynomial

```
rank=c(1:15)
aic=rep(0,15)
for (i in rank){
gamp.fit=gam(Soy~poly(OHRain,i)
                +poly(OHTemp,i)
                +poly(UpMWRain,i)
                +poly(UpMWTemp,i)
                +poly(SouthRain,i)
                +poly(SouthTemp,i)
                +poly(Corn,i)
                +poly(Oats,i)
                +poly(USD,i),
             data = quant.train)
aic[i]=gamp.fit$aic}
aic
```

```
##  [1] 782.3627 793.9689 804.0434 805.0898 819.6984 824.2860 817.9516
##  [8] 830.5529 834.0124 837.0546 834.3828 832.3421 839.2749 832.7021
## [15] 841.5671
```

Based on the AIC ranking, it is the best to choose the 12th degree model

```
gamp.fit=gam(Soy~poly(OHRain,12)
                +poly(OHTemp,12)
                +poly(UpMWRain,12)
                +poly(UpMWTemp,12)
                +poly(SouthRain,12)
                +poly(SouthTemp,12)
                +poly(Corn,12)
                +poly(Oats,12)
                +poly(USD,12),
             data = quant.train)
summary(gamp.fit)
```

```
##
## Call: gam(formula = Soy ~ poly(OHRain, 12) + poly(OHTemp, 12) +
poly(UpMWRain,
##     12) + poly(UpMWTemp, 12) + poly(SouthRain, 12) + poly(SouthTemp,
##     12) + poly(Corn, 12) + poly(Oats, 12) + poly(USD, 12), data =
quant.train)
## Deviance Residuals:
##       Min        1Q    Median        3Q       Max
## -2.035698 -0.450334 -0.006277  0.423627  2.747934
##
## (Dispersion Parameter for gaussian family taken to be 0.8991)
##
```

```
##     Null Deviance: 1822.399 on 274 degrees of freedom
## Residual Deviance: 149.2455 on 166 degrees of freedom
## AIC: 832.3421
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##                      Df  Sum Sq Mean Sq  F value    Pr(>F)
## poly(OHRain, 12)     12   48.54   4.045   4.4991 2.991e-06 ***
## poly(OHTemp, 12)     12   37.86   3.155   3.5089 0.0001211 ***
## poly(UpMWRain, 12)   12  113.61   9.468  10.5306 2.530e-15 ***
## poly(UpMWTemp, 12)   12   49.87   4.156   4.6226 1.887e-06 ***
## poly(SouthRain, 12)  12   64.47   5.372   5.9755 1.290e-08 ***
## poly(SouthTemp, 12)  12   32.14   2.678   2.9786 0.0008614 ***
## poly(Corn, 12)       12 1267.93 105.661 117.5225 < 2.2e-16 ***
## poly(Oats, 12)       12   34.13   2.845   3.1638 0.0004359 ***
## poly(USD, 12)        12   24.60   2.050   2.2803 0.0104672 *
## Residuals           166  149.25   0.899
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
gamp.pred=predict(gamp.fit,quant.train)
mean((Soy-gamp.pred)^2)
```

```
## [1] 12.94656
```

```
gamp.pred=predict(gamp.fit,quant.test)
mean((Soy-gamp.pred)^2)
```

```
## [1] 917.9026
```

With the result of the testing MSE, it is the best that I should not build the model in the first place.

GAM w/ natural splines

```
gam.fit=gam(Soy~s(OHRain,6)
                +s(OHTemp,6)
                +s(UpMWRain,6)
                +s(UpMWTemp,6)
                +s(SouthRain,6)
                +s(SouthTemp,6)
                +s(Corn,6)
                +s(Oats,6)
                +s(USD,6),
            data = quant.train)
summary(gam.fit)
```

```
##
## Call: gam(formula = Soy ~ s(OHRain, 6) + s(OHTemp, 6) + s(UpMWRain,
##     6) + s(UpMWTemp, 6) + s(SouthRain, 6) + s(SouthTemp, 6) +
##     s(Corn, 6) + s(Oats, 6) + s(USD, 6), data = quant.train)
```

```
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.12329 -0.51476 -0.05631  0.48084  3.24024
##
## (Dispersion Parameter for gaussian family taken to be 0.8804)
##
##     Null Deviance: 1822.399 on 274 degrees of freedom
## Residual Deviance: 193.6965 on 219.9998 degrees of freedom
## AIC: 796.035
##
## Number of Local Scoring Iterations: 7
##
## Anova for Parametric Effects
##                  Df  Sum Sq Mean Sq   F value    Pr(>F)
## s(OHRain, 6)      1    0.06    0.06    0.0714   0.78962
## s(OHTemp, 6)      1    2.20    2.20    2.5001   0.11528
## s(UpMWRain, 6)    1    1.70    1.70    1.9289   0.16629
## s(UpMWTemp, 6)    1    0.63    0.63    0.7113   0.39992
## s(SouthRain, 6)   1   14.80   14.80   16.8090 5.817e-05 ***
## s(SouthTemp, 6)   1    4.11    4.11    4.6649   0.03187 *
## s(Corn, 6)        1 1468.68 1468.68 1668.1237 < 2.2e-16 ***
## s(Oats, 6)        1   40.35   40.35   45.8300 1.156e-10 ***
## s(USD, 6)         1   18.06   18.06   20.5077 9.733e-06 ***
## Residuals       220  193.70    0.88
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                 Npar Df Npar F    Pr(F)
## (Intercept)
## s(OHRain, 6)          5 1.4861 0.195399
## s(OHTemp, 6)          5 2.7576 0.019399 *
## s(UpMWRain, 6)        5 0.8100 0.543603
## s(UpMWTemp, 6)        5 2.8430 0.016474 *
## s(SouthRain, 6)       5 2.8659 0.015763 *
## s(SouthTemp, 6)       5 1.1634 0.328191
## s(Corn, 6)            5 4.1168 0.001357 **
## s(Oats, 6)            5 2.2669 0.048917 *
## s(USD, 6)             5 2.8525 0.016171 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

With the result, we see that under the confidence level of 5%, we pick 6 parameters: OHTemp, SouthRain, SouthTemp, Corn, Oats, and USD for our final model

```
gam.fit=gam(Soy~s(OHTemp,6)
               +s(SouthRain,6)
               +s(SouthTemp,6)
               +s(Corn,6)
               +s(Oats,6)
```

```
               +s(USD,6),
            data = quant.train)
summary(gam.fit)

##
## Call: gam(formula = Soy ~ s(OHTemp, 6) + s(SouthRain, 6) + s(SouthTemp,
##     6) + s(Corn, 6) + s(Oats, 6) + s(USD, 6), data = quant.train)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.22105 -0.48377 -0.06378  0.48972  3.43101
##
## (Dispersion Parameter for gaussian family taken to be 0.8712)
##
##     Null Deviance: 1822.399 on 274 degrees of freedom
## Residual Deviance: 207.3489 on 237.9996 degrees of freedom
## AIC: 778.7658
##
## Number of Local Scoring Iterations: 5
##
## Anova for Parametric Effects
##                   Df  Sum Sq Mean Sq   F value    Pr(>F)
## s(OHTemp, 6)       1    2.05    2.05    2.3585   0.12593
## s(SouthRain, 6)    1   14.24   14.24   16.3403 7.150e-05 ***
## s(SouthTemp, 6)    1    5.75    5.75    6.5972   0.01083 *
## s(Corn, 6)         1 1471.01 1471.01 1688.4590 < 2.2e-16 ***
## s(Oats, 6)         1   36.78   36.78   42.2167 4.736e-10 ***
## s(USD, 6)          1   17.34   17.34   19.8991 1.259e-05 ***
## Residuals        238  207.35    0.87
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                  Npar Df Npar F     Pr(F)
## (Intercept)
## s(OHTemp, 6)           5 1.0237 0.4042483
## s(SouthRain, 6)        5 2.4711 0.0331677 *
## s(SouthTemp, 6)        5 1.3970 0.2260874
## s(Corn, 6)             5 4.8855 0.0002833 ***
## s(Oats, 6)             5 2.6009 0.0259299 *
## s(USD, 6)              5 2.9868 0.0123392 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The new model has all of its parameters statistically significant under the 5% confidence level. Next step, we put the new GAM model to the cross-validation test:

```
gam.pred=predict(gam.fit,quant.train)
mean((Soy-gam.pred)^2)

## [1] 12.71863
```

```
gam.pred=predict(gam.fit,quant.test)
mean((Soy-gam.pred)^2)
```

```
## [1] 12.03297
```

The GAM model is also improving with the new test data set and even has lower testing MSE of 10.7442

Tree Model

Finally, we use the tree model to derive the relationship between Soy price and other parameters.

```
tree.fit=tree(Soy~.,data=quant.train)
tree.fit
```

```
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 275 1822.00  7.519
##    2) Corn < 3.16375 177  185.30  6.049
##      4) Corn < 2.24225 55   26.14  5.282 *
##      5) Corn > 2.24225 122  112.20  6.394 *
##    3) Corn > 3.16375 98  562.70 10.180
##      6) USD < 85.5445 23   49.42 13.780
##       12) Corn < 6.5325 13   15.10 12.940 *
##       13) Corn > 6.5325 10   13.05 14.880 *
##      7) USD > 85.5445 75  123.20  9.071
##       14) Oats < 2.7675 64   90.73  8.863 *
##       15) Oats > 2.7675 11   13.51 10.280 *
```

```
summary(tree.fit)
```

```
##
## Regression tree:
## tree(formula = Soy ~ ., data = quant.train)
## Variables actually used in tree construction:
## [1] "Corn" "USD"  "Oats"
## Number of terminal nodes:  6
## Residual mean deviance:  1.006 = 270.7 / 269
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -2.2200 -0.6793 -0.1573  0.0000  0.6123  3.2060
```

With the tree model, we can see that the model only makes use of the two parameters: Oats and Corn. Thus I decided not to trim it any further down as the trimmed model may lose its flexibility.

The tree model is put into testing:

```
tree.pred=predict(tree.fit,quant.train)
mean((Soy-tree.pred)^2)
```

```
## [1] 12.46678
```

```
tree.pred=predict(tree.fit,quant.test)
mean((Soy-tree.pred)^2)
```

```
## [1] 12.21687
```

Tree method also improves when put under the testing data set with the testing MSE of 10.7890

Result

As the result, when comparing both the training and testing MSE and the parameters used, I come to the conclusion that the Lasso model help to build the lowest testing MSE with great result compared to the other without making the intepretability obscured.

Question 3:

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

Solution

The four chosen models for this question would be: LDA, QDA, KNN, and Support Vector Machine. The dependent variable would be Direction, the change in direction of the Soybean price

The confidence level used and the method of selection is the same as of the one for quantitative variable

```
qual=Data[,-1]
qual.train=qual[sampling,]
qual.test=qual[-sampling,]
```

LDA

```
lda.fit=lda(Direction~.,data=qual.train)
lda.fit
```

```
## Call:
## lda(Direction ~ ., data = qual.train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.1272727 0.8727273
##
## Group means:
##        OHTemp UpMWTemp SouthTemp   OHRain UpMWRain SouthRain     Oats
## Down 53.46571 44.04571  62.20857 3.594857 2.429714  2.963714 1.633080
## Up   52.24417 42.15542  61.18167 3.529542 2.548667  2.931500 1.955391
##          Corn      USD
```

```
## Down 2.810020 94.67826
## Up   3.138584 95.73703
##
## Coefficients of linear discriminants:
##                   LD1
## OHTemp     0.07045927
## UpMWTemp  -0.12025261
## SouthTemp  0.05183778
## OHRain    -0.26629760
## UpMWRain   0.35597998
## SouthRain  0.11354605
## Oats       2.08100319
## Corn      -0.60109490
## USD        0.05059834
```

The data is fitted into the LDA model and the model then is put into testing with the testing data set:

```
lda.preds=predict(lda.fit,qual.test)
mean(lda.preds$class==qual.test$Direction)

## [1] 0.8836364
```

The LDA model yields good result with 88% correct prediction rate.

QDA

```
qda.fit=qda(Direction~.,data=qual.train)
qda.fit

## Call:
## qda(Direction ~ ., data = qual.train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.1272727 0.8727273
##
## Group means:
##        OHTemp UpMWTemp SouthTemp   OHRain UpMWRain SouthRain     Oats
## Down 53.46571 44.04571  62.20857 3.594857 2.429714  2.963714 1.633080
## Up   52.24417 42.15542  61.18167 3.529542 2.548667  2.931500 1.955391
##          Corn      USD
## Down 2.810020 94.67826
## Up   3.138584 95.73703
```

The data is fitted into the QDA model and the model then is put into testing with the testing data set:

```
qda.preds=predict(qda.fit,qual.test)
mean(qda.preds$class==qual.test$Direction)

## [1] 0.8036364
```

The QDA model yields decent result with 78.9091% correct prediction rate.

KNN

I put the testing and training data sets into the KNN models with the range of $k = \{1,2,3,4,5,6,7,8,9,10\}$

```
knn.test=qual.test[,-1]
knn.train=qual.train[,-1]
knnmean=rep(0,10)
for(i in c(1:10)){
  knn.pred=knn(knn.train,knn.test,qual.train$Direction,k=i)
  knnmean[i]=mean(knn.pred==qual.test$Direction)
}
knnmean
```

```
##  [1] 0.7854545 0.7781818 0.8436364 0.8618182 0.8763636 0.8727273 0.8836364
##  [8] 0.8836364 0.8836364 0.8836364
```

The best model with lowest possible k come with $k = 7$ and the prediction rate of 88%

SVM

For the SVM model type, I choose all three main type of SVM: Linear, Radial, and Polynomial to see which hold the best result.

Linear

```
tune.lm=tune(svm,Direction~.,
             data=qual.train,
             kernel="linear",
             ranges=list(cost=c(0.01,0.05,0.1,0.5,1,5,10,50,100)))
tune.lm$best.model
```

```
##
## Call:
## best.tune(method = svm, train.x = Direction ~ ., data = qual.train,
##     ranges = list(cost = c(0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50,
##         100)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##       gamma:  0.1111111
##
## Number of Support Vectors:  94
```

I choose cost of 0.01

Radial

```
tune.rad=tune(svm,Direction~.,
              data=qual.train,kernel="radial",
              ranges=list(cost=c(0.01,0.05,0.1,0.5,1,5,10,50,100),
                          gamma=c(0.01,0.05,0.1,0.5,1,5,10)))
tune.rad$best.model
```

```
##
## Call:
## best.tune(method = svm, train.x = Direction ~ ., data = qual.train,
##     ranges = list(cost = c(0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50,
##         100), gamma = c(0.01, 0.05, 0.1, 0.5, 1, 5, 10)), kernel =
"radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.01
##       gamma:  0.01
##
## Number of Support Vectors:  76
```

I choose cost of 0.01 and gamma of 0.01

Polynomial

```
tune.poly=tune(svm,Direction~.,
               data=qual.train,
               kernel="polynomial",
               ranges=list(cost=c(0.01,0.05,0.1,0.5,1,5,10,50,100),
                           degree=c(1,2,3,4,5)))
tune.poly$best.model
```

```
##
## Call:
## best.tune(method = svm, train.x = Direction ~ ., data = qual.train,
##     ranges = list(cost = c(0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50,
##         100), degree = c(1, 2, 3, 4, 5)), kernel = "polynomial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  4
##       gamma:  0.1111111
##      coef.0:  0
##
## Number of Support Vectors:  98
```

I choose cost of 0.01 and degree of 1

Testing

After choosing the best parameters for each type of model, I put them into testing

```
svm.lm=svm(Direction~.,data=qual.train,kernel="linear",cost=0.01)
svm.poly=svm(Direction~.,data=qual.train,kernel="polynomial",cost=0.01,degree
=1)
svm.rad=svm(Direction~.,data=qual.train,kernel="radial",cost=0.01,gamma=0.01)
svm.lm.pred=predict(svm.lm,qual.test)
mean(svm.lm.pred==qual.test$Direction)

## [1] 0.8836364

svm.poly.pred=predict(svm.poly,qual.test)
mean(svm.poly.pred==qual.test$Direction)

## [1] 0.8836364

svm.rad.pred=predict(svm.rad,qual.test)
mean(svm.rad.pred==qual.test$Direction)

## [1] 0.8836364
```

The result comes as a surprise as all three models have the same prediction rate of 88%. Thus, I would not have a predilection for any type of model for the SVM.

Result

Based on the above results, I woud choose the SVM for it holds the highest possible prediction rate, and also for its flexibility as all of its sub-models agree on the result, which might help in predicting future result.

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.

```
library(ISLR)
attach(Auto)
```

(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.

```
n=length(mpg)
var=rep(0,n)
```

```
var[mpg>median(mpg)]=1
Auto$mpgtype=as.factor(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.

```
tune.out=tune(svm,mpgtype~.,data=Auto,kernel="linear",ranges=list(cost=c(0.01
,0.05,0.1,0.5,1,5,10,50,100)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.01012821
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-02 0.07397436 0.04680629
## 2 5e-02 0.06115385 0.05222792
## 3 1e-01 0.05089744 0.04896492
## 4 5e-01 0.01775641 0.02670179
## 5 1e+00 0.01012821 0.01760404
## 6 5e+00 0.01782051 0.02088734
## 7 1e+01 0.02294872 0.01871043
## 8 5e+01 0.03570513 0.03005165
## 9 1e+02 0.03570513 0.03005165
```

```
tune.out$best.model
```

```
##
## Call:
## best.tune(method = svm, train.x = mpgtype ~ ., data = Auto, ranges =
list(cost = c(0.01,
##     0.05, 0.1, 0.5, 1, 5, 10, 50, 100)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##       gamma:  0.003205128
```

```
##
## Number of Support Vectors:  56
```

The SVM version with cost=1 has the lowest cross validation value

(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.

```
rad.out=tune(svm,mpgtype~.,data=Auto,kernel="radial",ranges=list(cost=c(0.01,
0.05,0.1,0.5,1,5,10,50,100),gamma=c(0.01,0.05,0.1,0.5,1,5,10)))
rad.out$best.model
```

```
##
## Call:
## best.tune(method = svm, train.x = mpgtype ~ ., data = Auto, ranges =
list(cost = c(0.01,
##     0.05, 0.1, 0.5, 1, 5, 10, 50, 100), gamma = c(0.01, 0.05,
##     0.1, 0.5, 1, 5, 10)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  50
##       gamma:  0.01
##
## Number of Support Vectors:  59
```

```
poly.out=tune(svm,mpgtype~.,data=Auto,kernel="polynomial",ranges=list(cost=c(
0.01,0.05,0.1,0.5,1,5,10,50,100),degree=c(1,2,3,4,5)))
poly.out$best.model
```

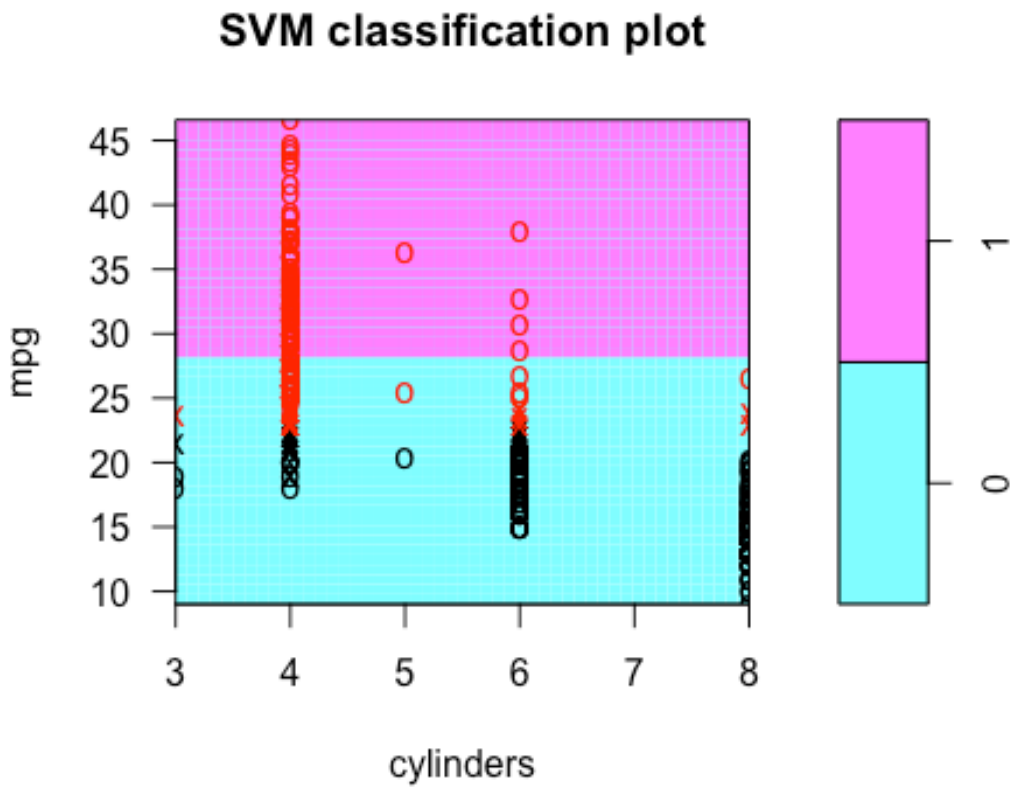
```
##
## Call:
## best.tune(method = svm, train.x = mpgtype ~ ., data = Auto, ranges =
list(cost = c(0.01,
##     0.05, 0.1, 0.5, 1, 5, 10, 50, 100), degree = c(1, 2, 3, 4,
##     5)), kernel = "polynomial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  100
##      degree:  1
##       gamma:  0.003205128
##      coef.0:  0
##
## Number of Support Vectors:  71
```

For the radial approach, the model with cost=100 and gamma=0.01 has the lowest error
For the polynomial SVM, the model with lowest error is no longer the one with the cost of 1 anymore. Instead, model with cost=100 and degree=1 has the lowest error

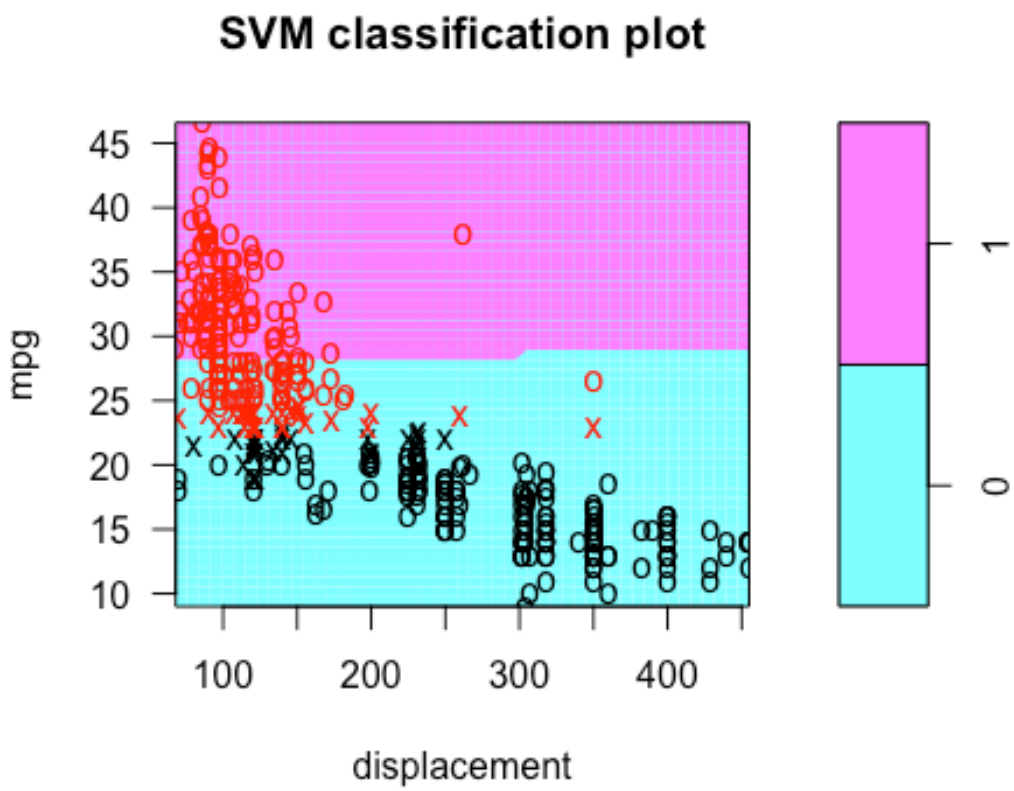
(d)

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.

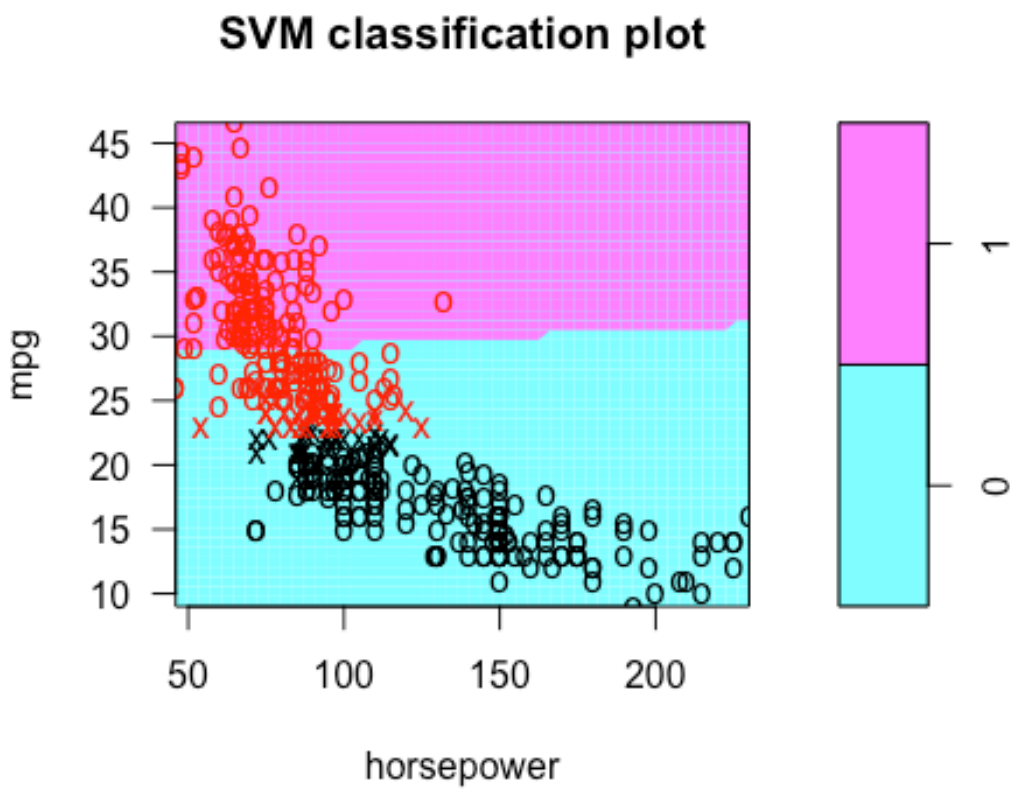
```
svm.lm=svm(mpgtype~.,data=Auto,kernel="linear",cost=1)
svm.poly=svm(mpgtype~.,data=Auto,kernel="polynomial",cost=100,degree=1)
svm.rad=svm(mpgtype~.,data=Auto,kernel="radial",cost=100,gamma=0.01)
##Plots for Linear SVM
plot(svm.lm,Auto,mpg~cylinders)
```



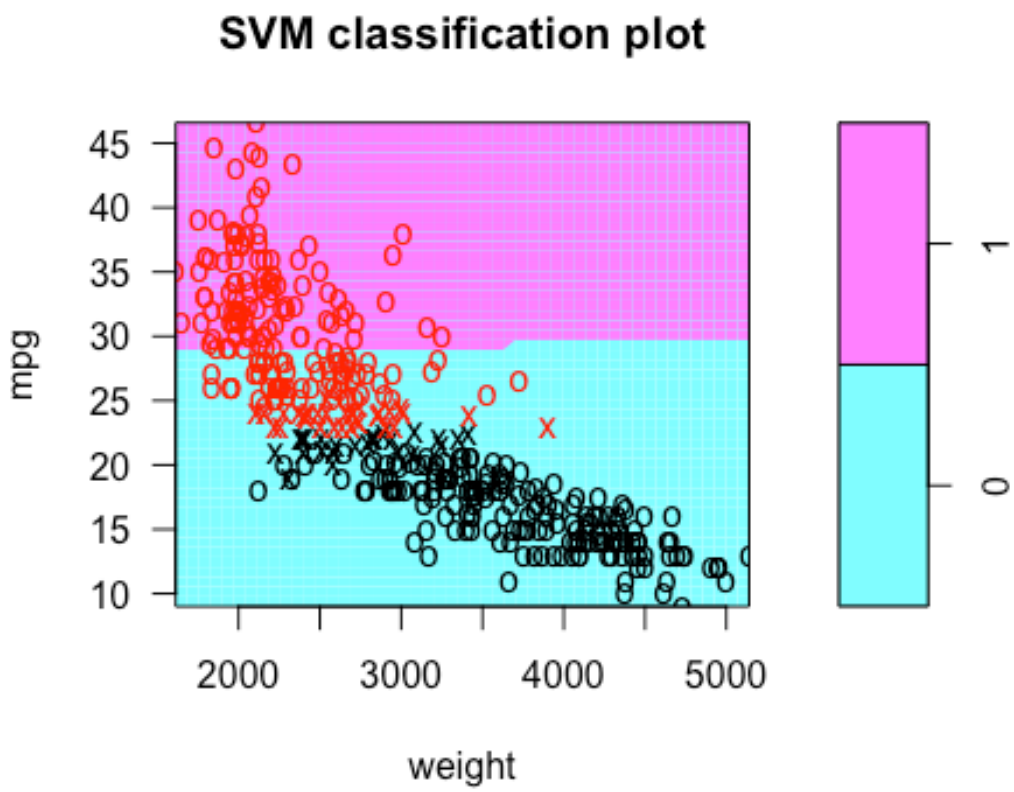

```
plot(svm.lm,Auto,mpg~displacement)
```



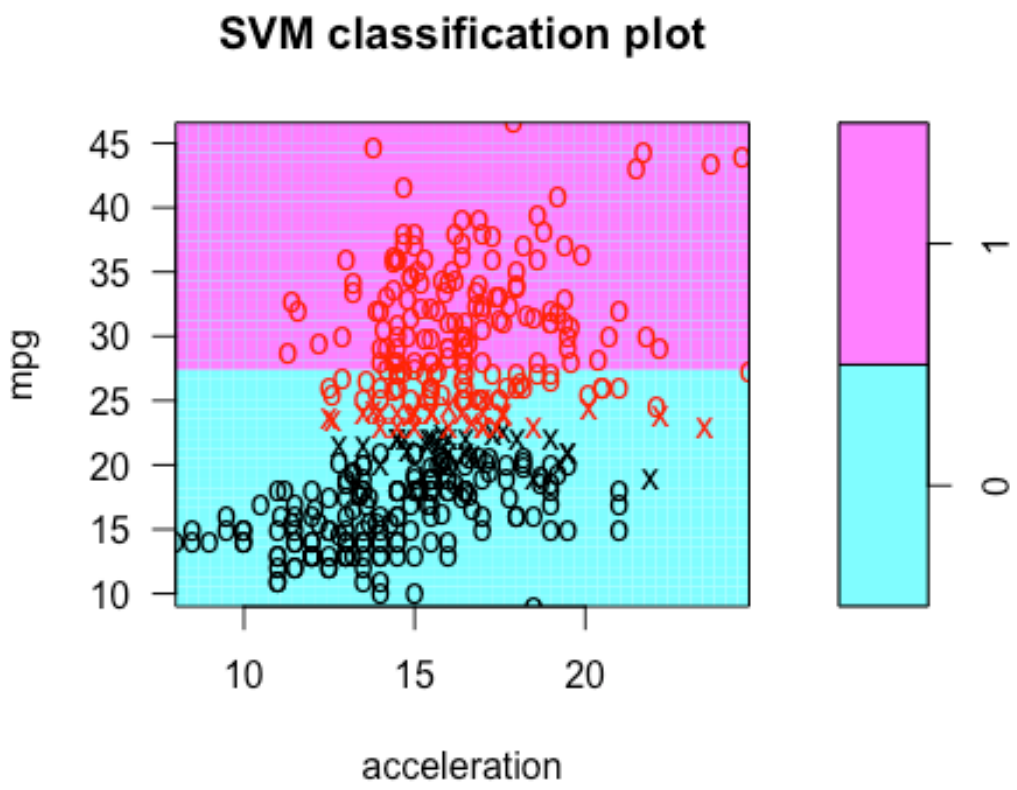

```
plot(svm.lm,Auto,mpg~horsepower)
```



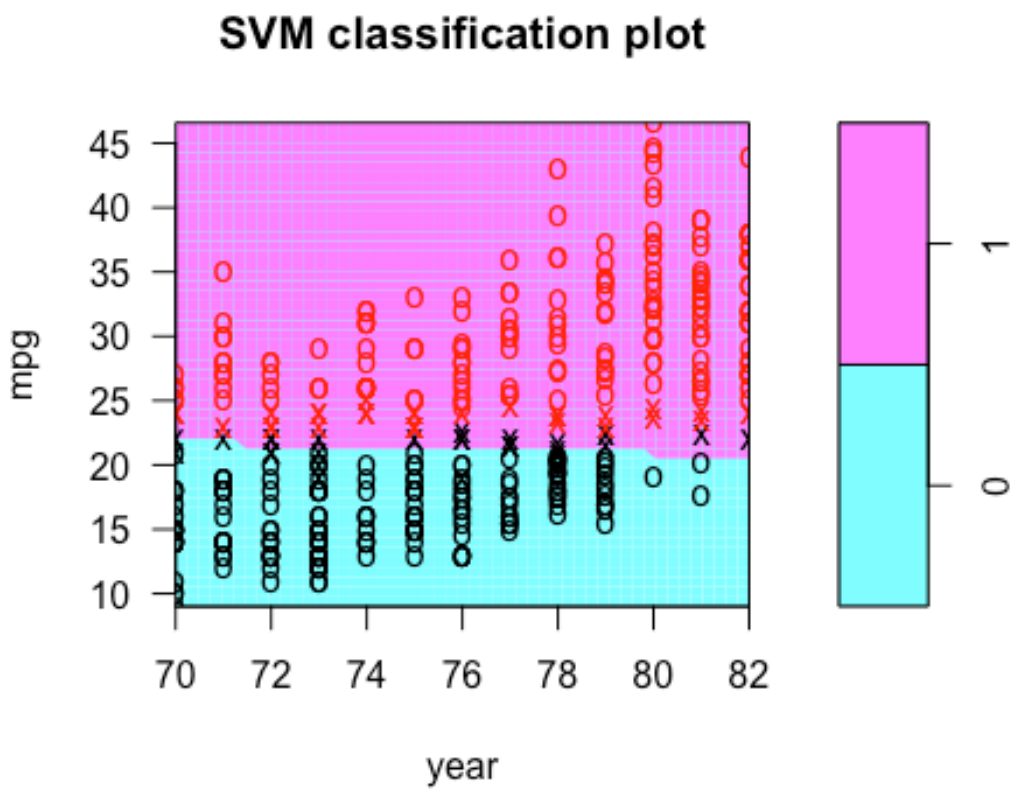

```
plot(svm.lm,Auto,mpg~weight)
```



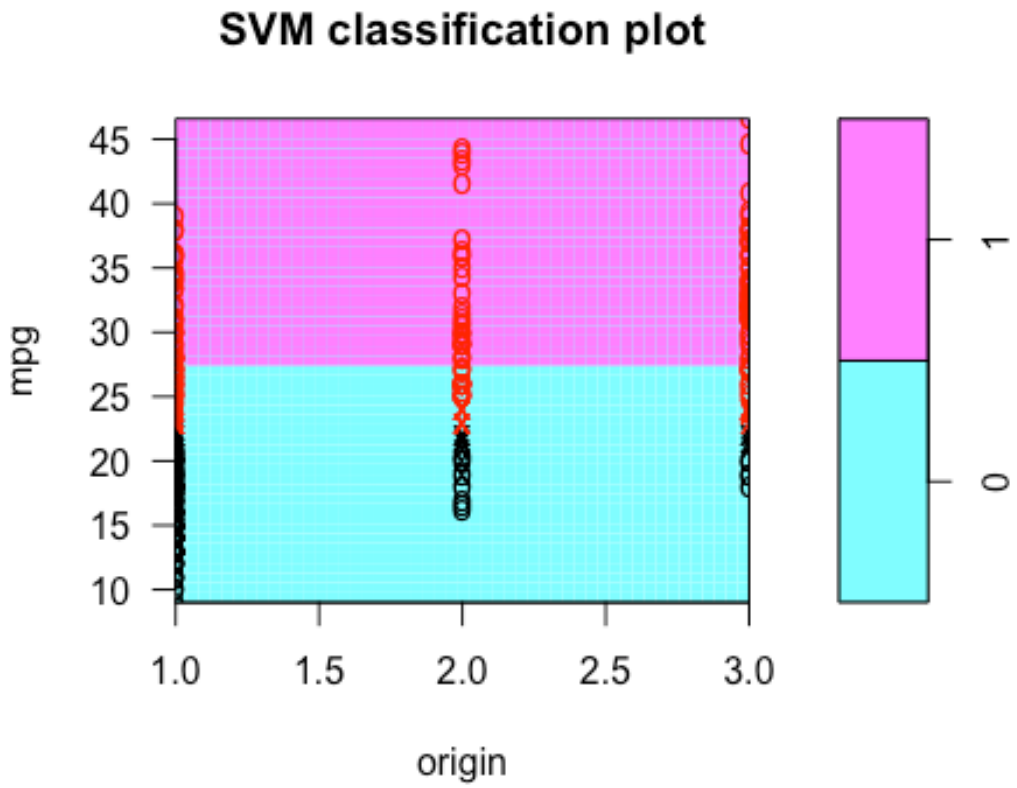

```
plot(svm.lm,Auto,mpg~acceleration)
```



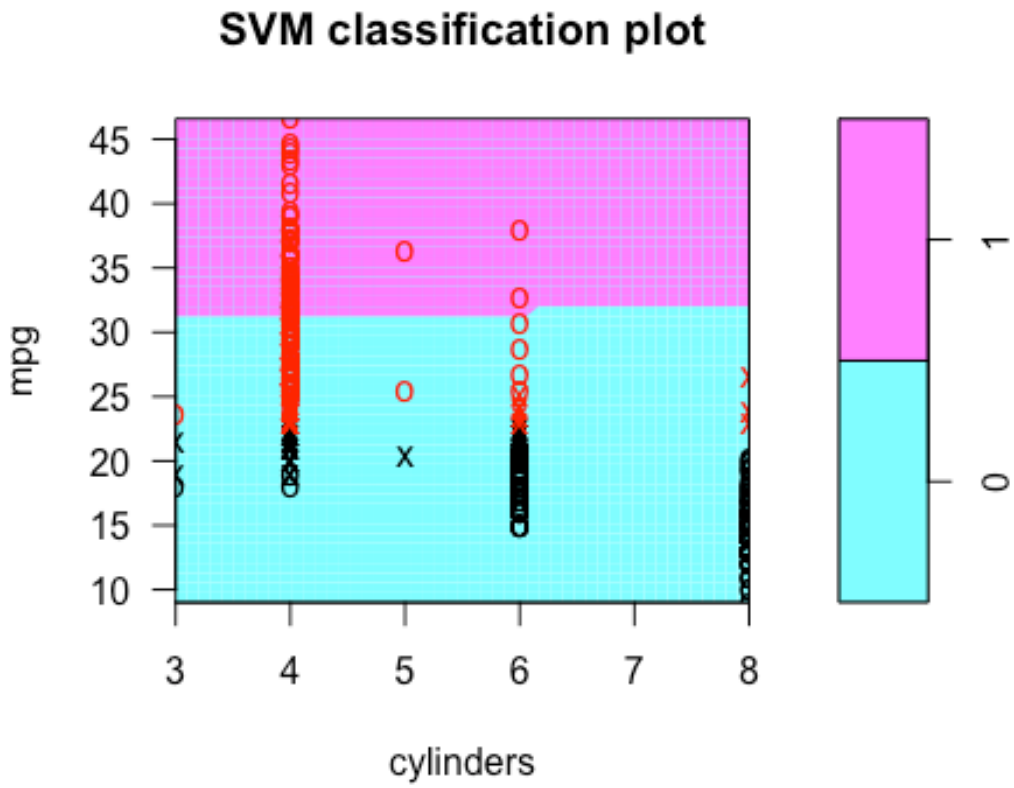

```
plot(svm.lm,Auto,mpg~year)
```



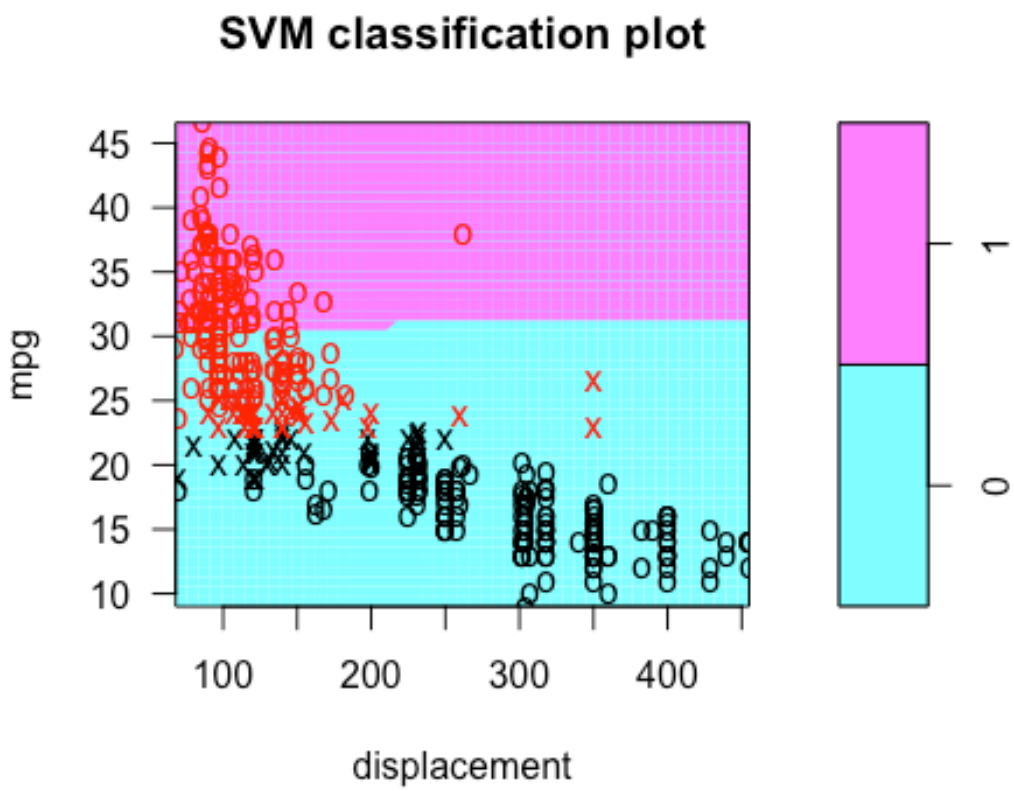

```
plot(svm.lm,Auto,mpg~origin)
```



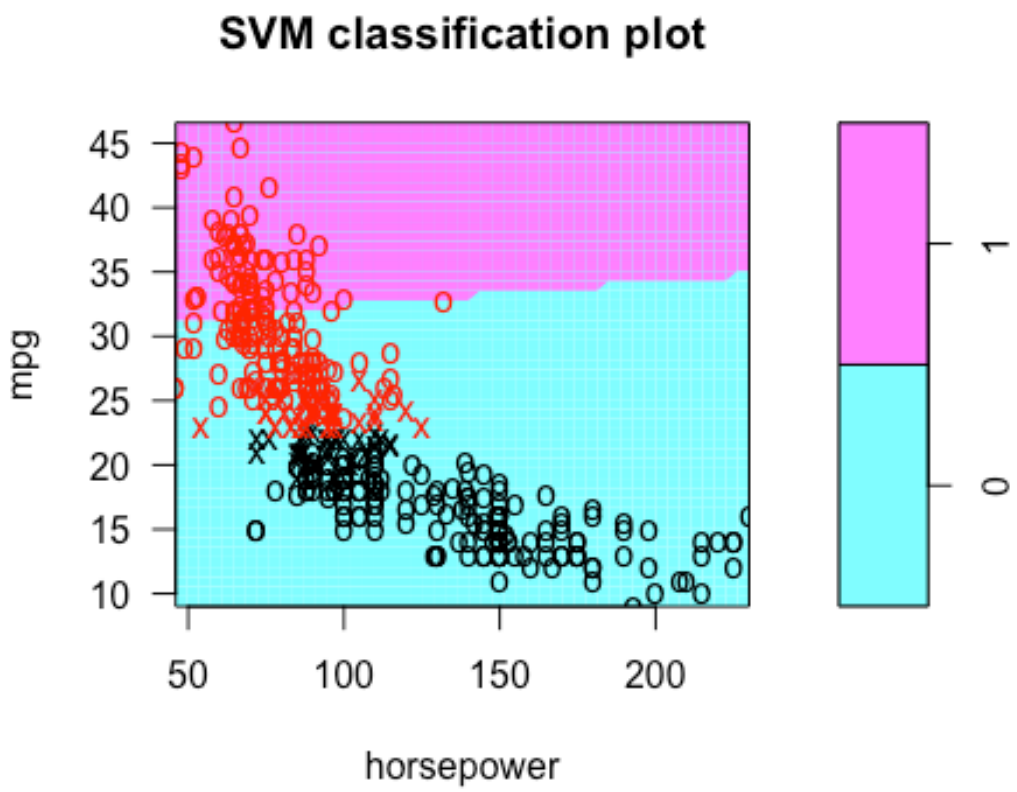

```
##Plots for Plolynomial SVM
plot(svm.poly,Auto,mpg~cylinders)
```



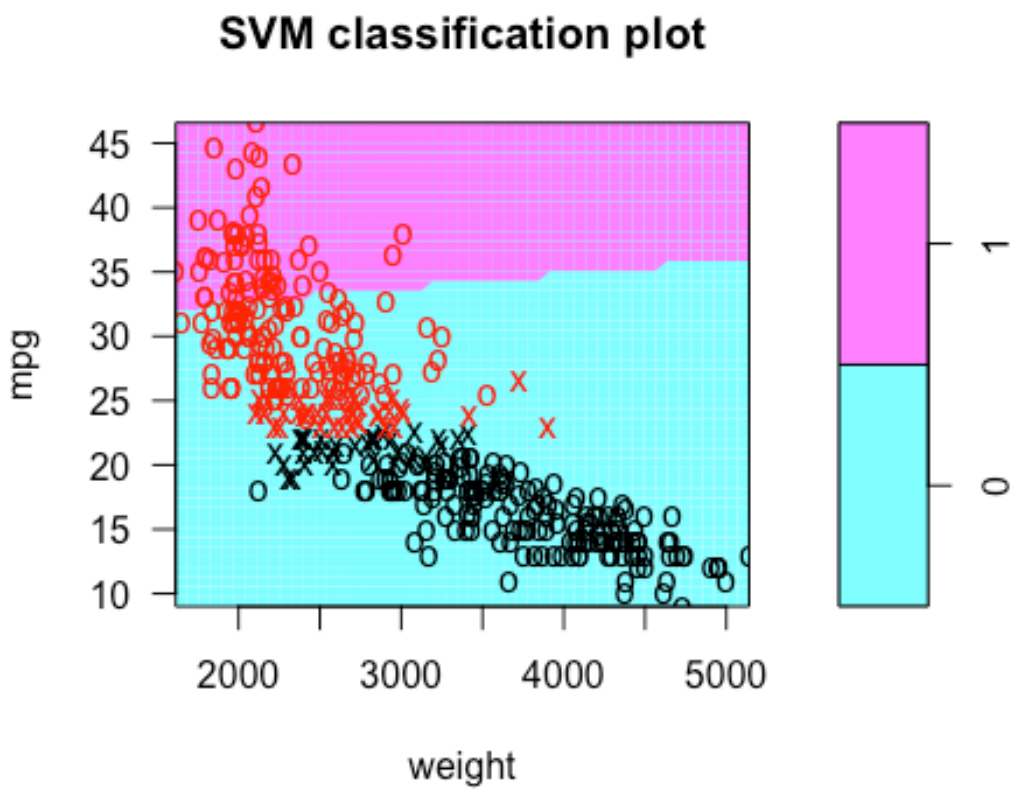

```
plot(svm.poly,Auto,mpg~displacement)
```



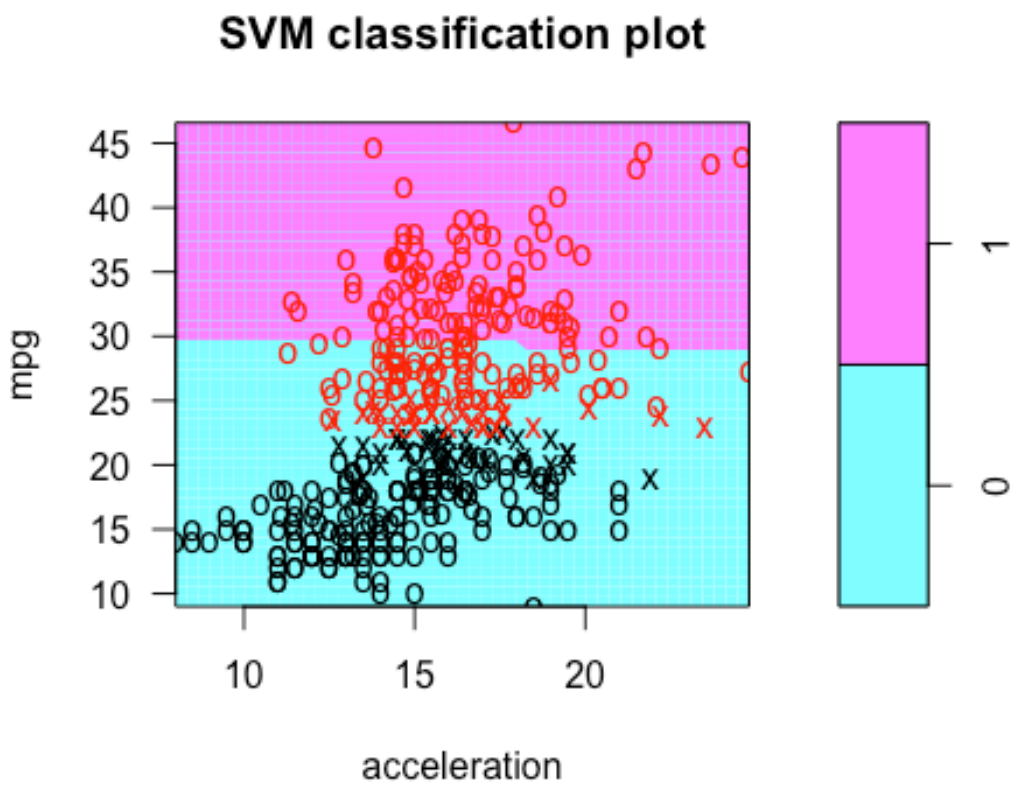

```
plot(svm.poly,Auto,mpg~horsepower)
```



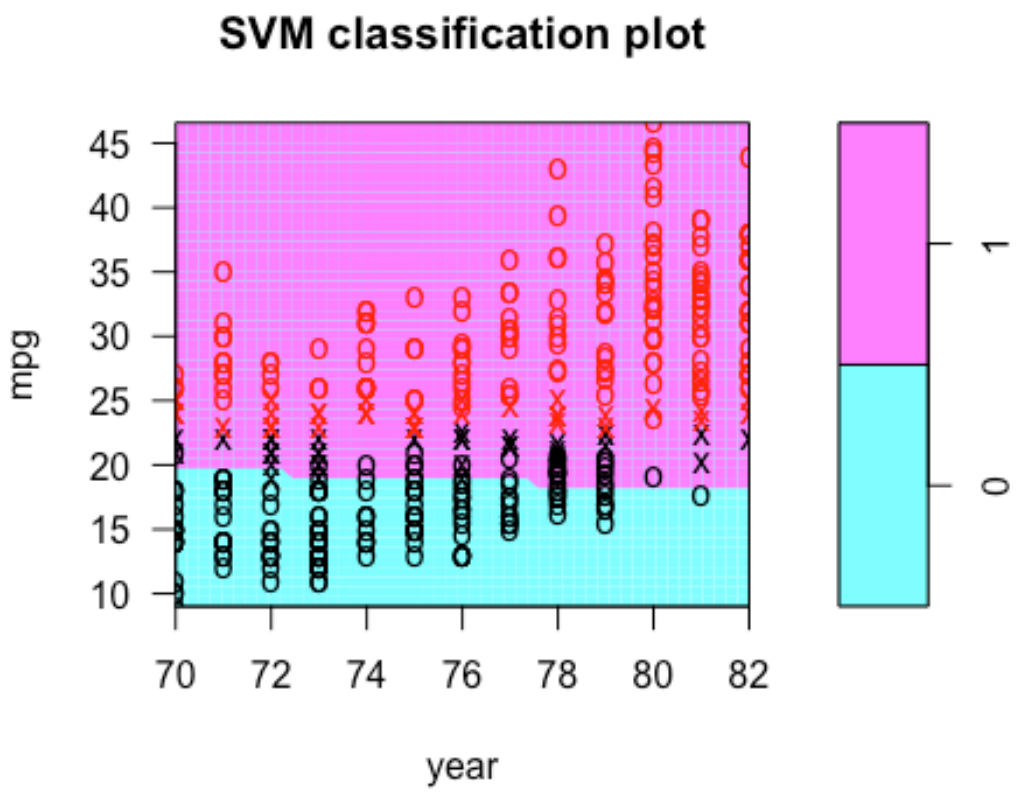

```
plot(svm.poly,Auto,mpg~weight)
```



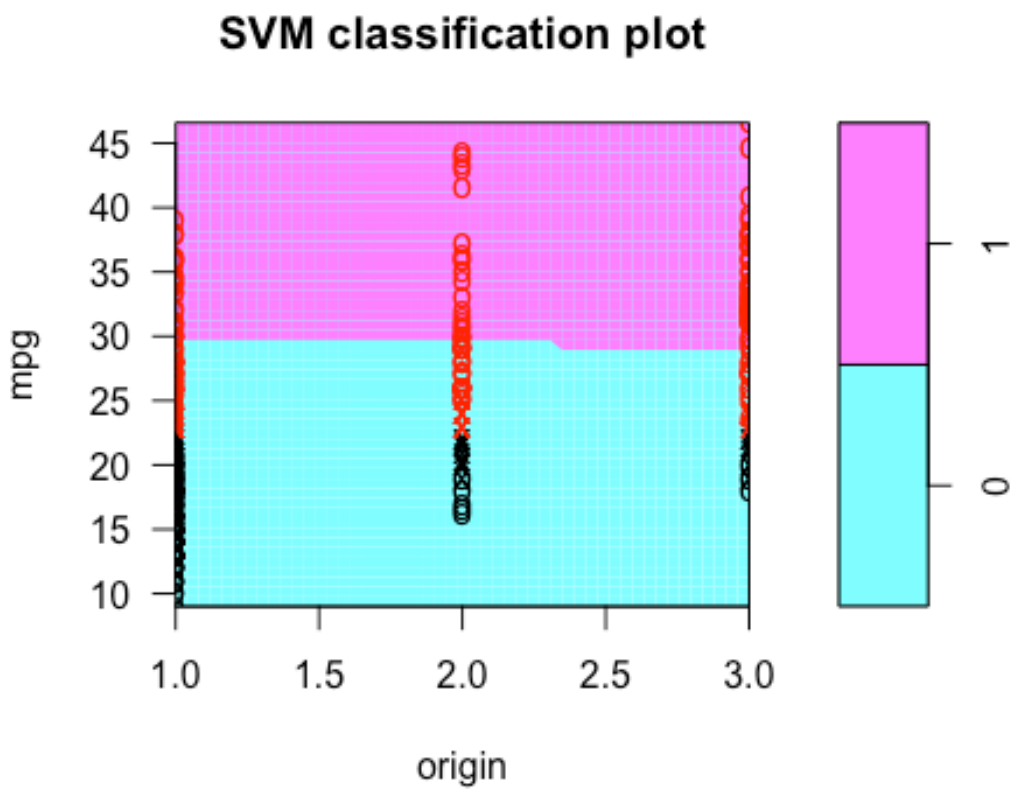

```
plot(svm.poly,Auto,mpg~acceleration)
```



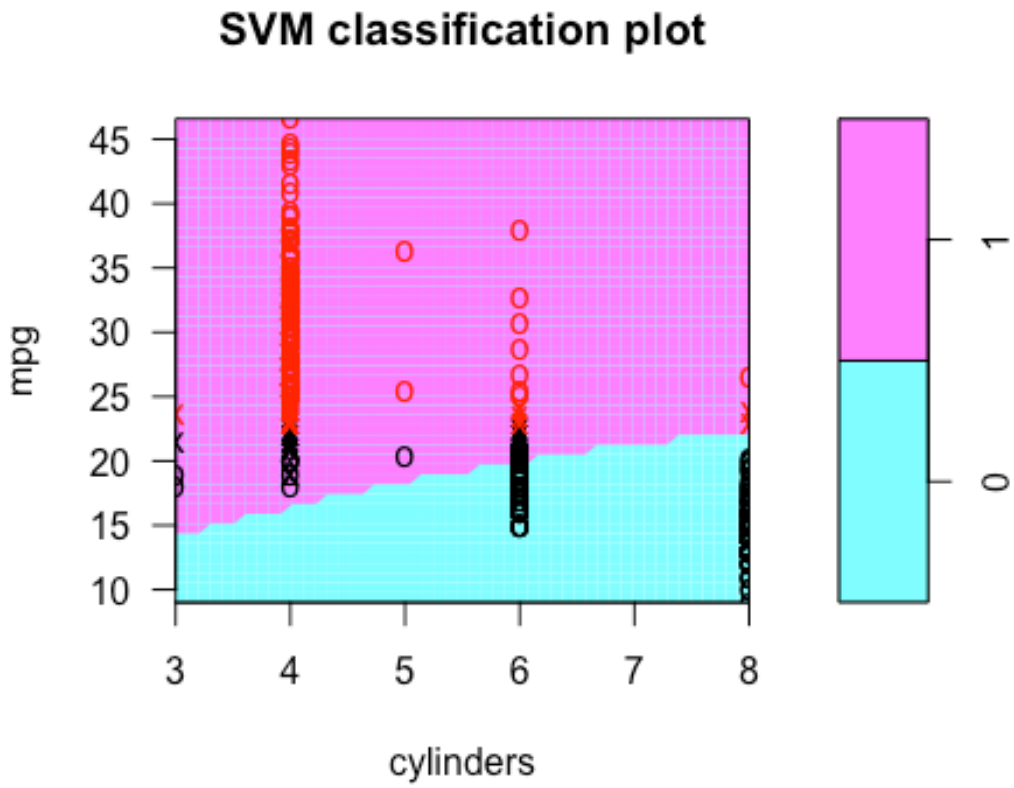

```
plot(svm.poly,Auto,mpg~year)
```



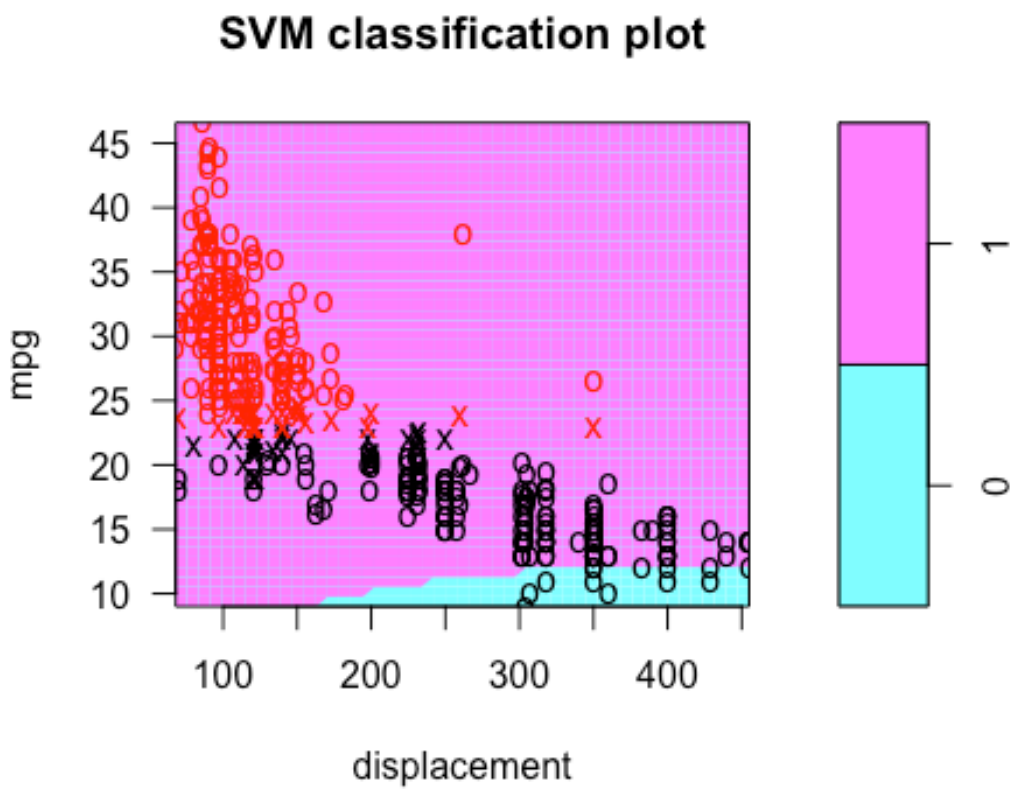

```
plot(svm.poly,Auto,mpg~origin)
```



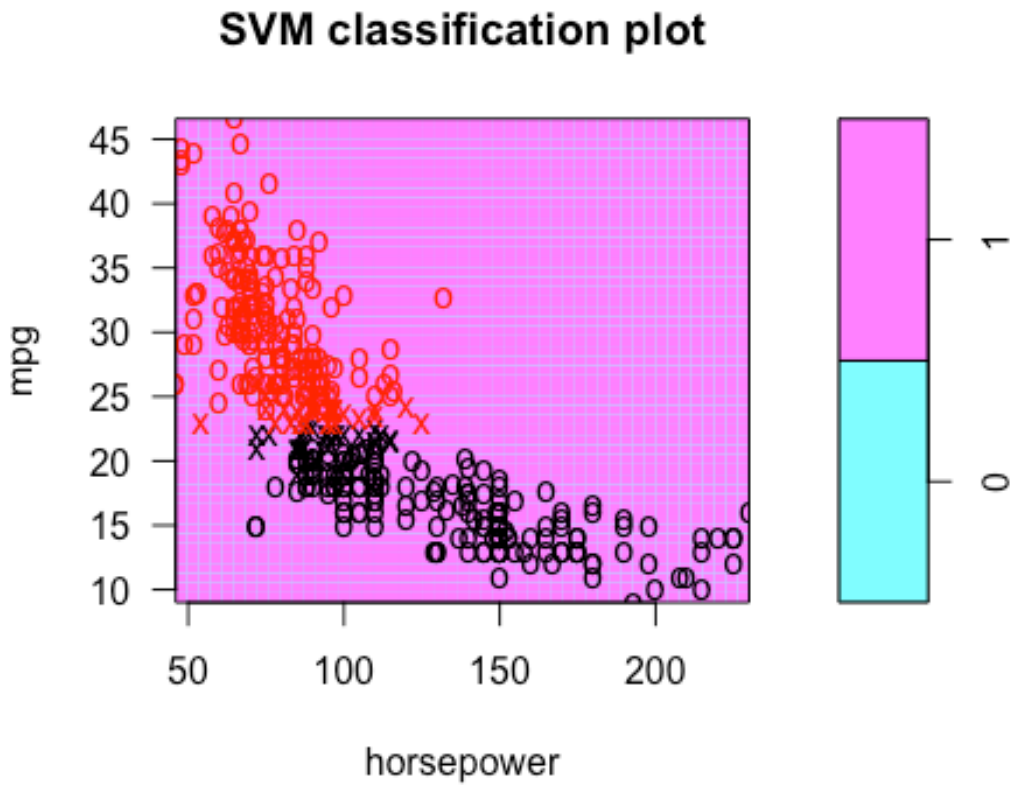

```
##Plots for Radial SVM
plot(svm.rad,Auto,mpg~cylinders)
```



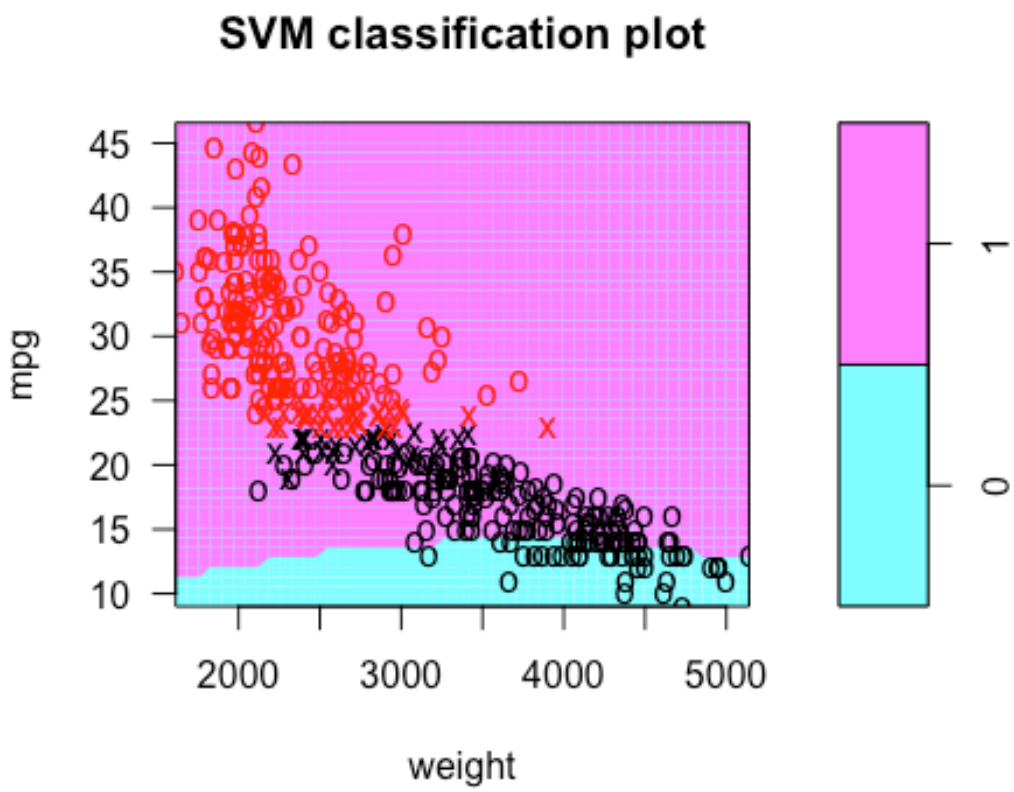

```
plot(svm.rad,Auto,mpg~displacement)
```



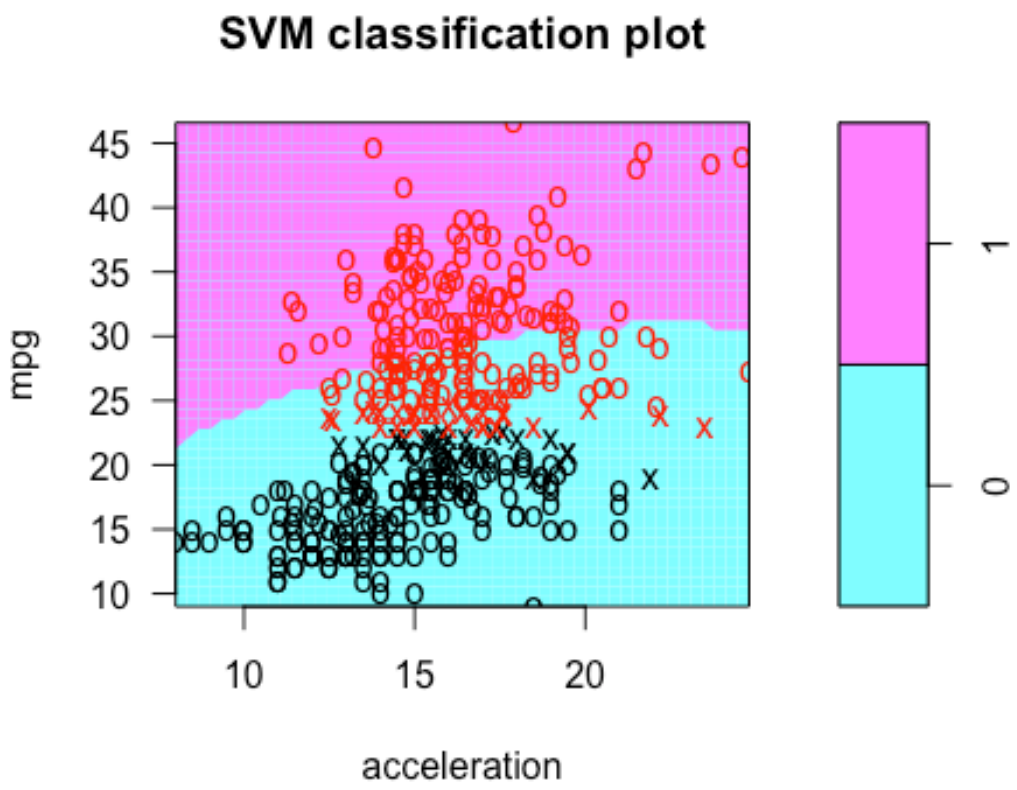

```
plot(svm.rad,Auto,mpg~horsepower)
```



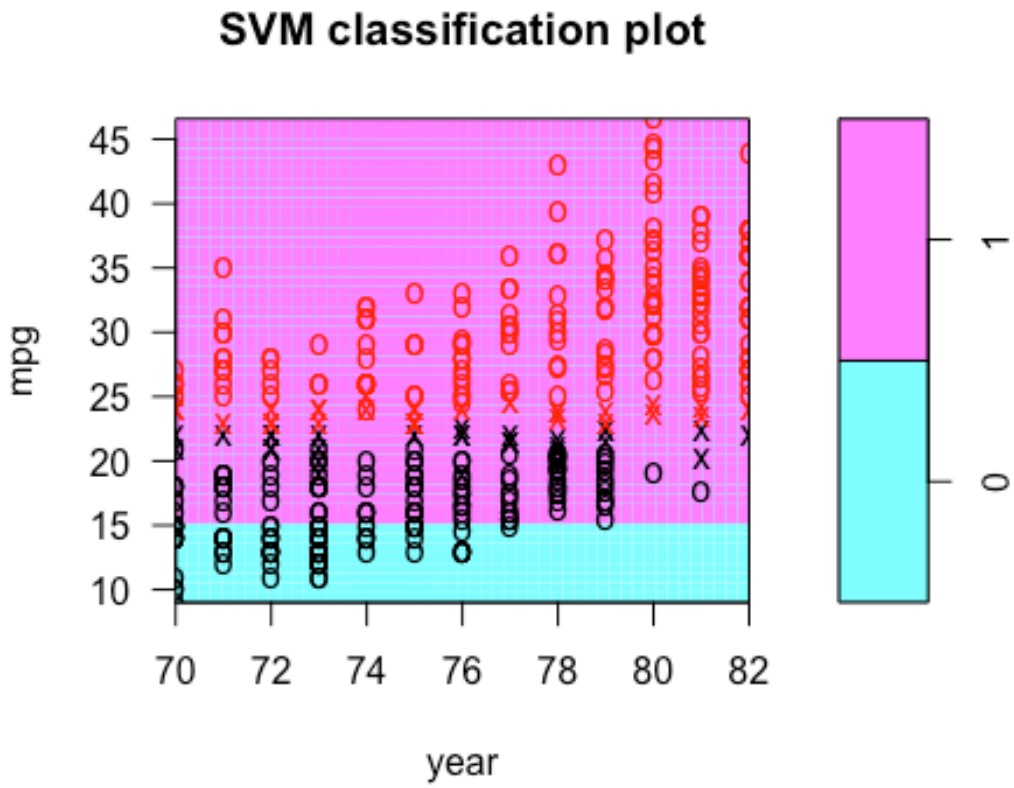

```
plot(svm.rad,Auto,mpg~weight)
```



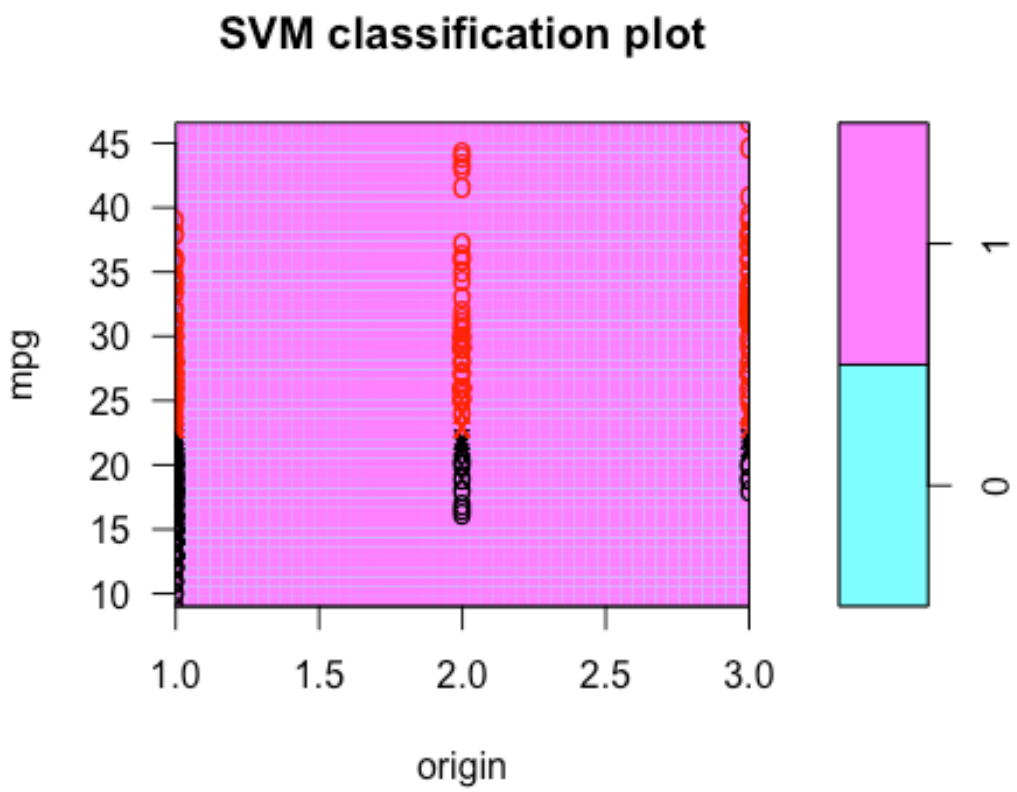

```
plot(svm.rad,Auto,mpg~acceleration)
```




```
plot(svm.rad,Auto,mpg~year)
```




```
plot(svm.rad,Auto,mpg~origin)
```


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