FE590. Assignment #4.

Anonymous

2018-05-01

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 = 10430108 #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 = 10430108 %% 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.

Our data is SPY ETF with 8 stocks which are AAPL, MSFT, AMZN, GOOG, JPM, JNJ, XOM and BAC. We transform all data into 3 sets which are close price data, return data and direction data.

Our purpuse is that we are going to build and select 2 statistical learning models to create a strategy for SPY ETF trading.

```
library(quantmod)

Stock <- c("SPY","AAPL", "MSFT", "AMZN", "GOOG", "JPM", "JNJ", "XOM", "BAC")
Start = "2010-01-01"
End = "2018-04-25"

sp500.close <- NULL
sp500.return <- NULL

for (i in Stock){
        temp.stock <- getSymbols(Symbols = i, from = Start, to = End, auto.assign = F)
        sp500.close <- cbind(sp500.close, temp.stock[,4])
```

```r
        temp.return <- dailyReturn(temp.stock, type = "arithmetic")

        sp500.return <- cbind(sp500.return, temp.return)
}
colnames(sp500.close) <- Stock
colnames(sp500.return) <- paste("r.",Stock, sep = "")
sp500.close <- data.frame(sp500.close)
sp500.return <- data.frame(sp500.return)
```

The first set of data is close price.

```r
head(sp500.close, 4)
```

```
##               SPY     AAPL  MSFT   AMZN     GOOG   JPM   JNJ   XOM   BAC
## 2010-01-04 113.33 30.57286 30.95 133.90 311.3500 42.85 64.68 69.15 15.69
## 2010-01-05 113.63 30.62571 30.96 134.69 309.9789 43.68 63.93 69.42 16.20
## 2010-01-06 113.71 30.13857 30.77 132.25 302.1647 43.92 64.45 70.02 16.39
## 2010-01-07 114.19 30.08286 30.45 130.00 295.1305 44.79 63.99 69.80 16.93
```

The second set of data is return which we are going to combine lag 1 step and lage 2 steps of return into this set.

```r
head(sp500.return, 4)
```

```
##                  r.SPY       r.AAPL        r.MSFT       r.AMZN
## 2010-01-04 0.008543196  0.002717514  0.0107772694 -0.017247750
## 2010-01-05 0.002647093  0.001728854  0.0003230371  0.005899985
## 2010-01-06 0.000704057 -0.015906307 -0.0061369188 -0.018115688
## 2010-01-07 0.004221291 -0.001848595 -0.0103997075 -0.017013233
##                    r.GOOG      r.JPM         r.JNJ        r.XOM      r.BAC
## 2010-01-04 -0.0003189411 0.02536485 -0.0004635914  0.006257290 0.02952756
## 2010-01-05 -0.0044037068 0.01936994 -0.0115955473  0.003904497 0.03250484
## 2010-01-06 -0.0252087463 0.00549446  0.0081338495  0.008643028 0.01172827
## 2010-01-07 -0.0232794894 0.01980881 -0.0071372385 -0.003141874 0.03294698
```

```r
sp500.return.lag1 <- rbind(c(NA*1:ncol(sp500.return[,2:ncol(sp500.return)])),
                           sp500.return[1:(nrow(sp500.return)-1),2:ncol(sp500.return)])
colnames(sp500.return.lag1) <- paste("lag1.",Stock[2:length(Stock)], sep = "")

sp500.return.lag2 <- rbind(c(NA*1:ncol(sp500.return[,2:ncol(sp500.return)])),
                           sp500.return[1:(nrow(sp500.return)-2),2:ncol(sp500.return)])
sp500.return.lag2 <- rbind(c(NA*1:ncol(sp500.return)),sp500.return.lag2)
colnames(sp500.return.lag2) <- paste("lag2.",Stock[2:length(Stock)], sep = "")
```

The third set of data is direction which 1 means up direction and -1 means down direction. In this set, we convert the data type to factor for future step.

```r
sp500.direction <- NULL
sp500.direction <- cbind(sp500.direction, ifelse(sp500.return$r.SPY < 0, -1, 1))
```

```r
sp500.direction <- cbind(sp500.direction, ifelse(sp500.return$r.AAPL < 0, -1, 1))
sp500.direction <- cbind(sp500.direction, ifelse(sp500.return$r.MSFT < 0, -1, 1))
sp500.direction <- cbind(sp500.direction, ifelse(sp500.return$r.AMZN < 0, -1, 1))
sp500.direction <- cbind(sp500.direction, ifelse(sp500.return$r.GOOG < 0, -1, 1))
sp500.direction <- cbind(sp500.direction, ifelse(sp500.return$r.JPM < 0, -1, 1))
sp500.direction <- cbind(sp500.direction, ifelse(sp500.return$r.JNJ < 0, -1, 1))
sp500.direction <- cbind(sp500.direction, ifelse(sp500.return$r.XOM < 0, -1, 1))
sp500.direction <- cbind(sp500.direction, ifelse(sp500.return$r.BAC < 0, -1, 1))
sp500.direction <- data.frame(sp500.direction)
colnames(sp500.direction) <- paste("d.",Stock, sep = "")

head(sp500.direction, 4)
```

```
##   d.SPY d.AAPL d.MSFT d.AMZN d.GOOG d.JPM d.JNJ d.XOM d.BAC
## 1     1      1      1     -1     -1     1    -1     1     1
## 2     1      1      1      1     -1     1    -1     1     1
## 3     1     -1     -1     -1     -1     1     1     1     1
## 4     1     -1     -1     -1     -1     1    -1    -1     1
```

```r
class(sp500.direction$d.SPY)
```

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

```r
sp500.direction$d.SPY <- as.factor(sp500.direction$d.SPY)
sp500.direction$d.AAPL <- as.factor(sp500.direction$d.AAPL)
sp500.direction$d.MSFT <- as.factor(sp500.direction$d.MSFT)
sp500.direction$d.AMZN <- as.factor(sp500.direction$d.AMZN)
sp500.direction$d.GOOG <- as.factor(sp500.direction$d.GOOG)
sp500.direction$d.JPM <- as.factor(sp500.direction$d.JPM)
sp500.direction$d.JNJ <- as.factor(sp500.direction$d.JNJ)
sp500.direction$d.XOM <- as.factor(sp500.direction$d.XOM)
sp500.direction$d.BAC <- as.factor(sp500.direction$d.BAC)

class(sp500.direction$d.SPY)
```

```
## [1] "factor"
```

We combine the second and third datasets to be our master dataset that we are going to use.

```r
sp500 <- cbind(sp500.return, sp500.direction)
sp500 <- cbind(cbind(cbind(sp500.return, sp500.return.lag1), sp500.return.lag2 ),
               sp500.direction)

sp500 <- sp500[complete.cases(sp500),]
head(sp500, 4)
```

```
##                  r.SPY       r.AAPL       r.MSFT      r.AMZN       r.GOOG
## 2010-01-06 0.000704057 -0.015906307 -0.006136919 -0.01811569 -0.025208746
## 2010-01-07 0.004221291 -0.001848595 -0.010399708 -0.01701323 -0.023279489
## 2010-01-08 0.003327769  0.006648338  0.006896519  0.02707695  0.013331111
## 2010-01-11 0.001396552 -0.008821591 -0.012720157 -0.02404139 -0.001511568
##                  r.JPM         r.JNJ        r.XOM        r.BAC
## 2010-01-06  0.005494460  0.0081338495  0.008643028  0.011728271
## 2010-01-07  0.019808812 -0.0071372385 -0.003141874  0.032946982
## 2010-01-08 -0.002455928  0.0034379902 -0.004011547 -0.008859953
## 2010-01-11 -0.003357229  0.0001557701  0.011219880  0.008939153
##               lag1.AAPL     lag1.MSFT    lag1.AMZN    lag1.GOOG
## 2010-01-06  0.001728854  0.0003230371  0.005899985 -0.004403707
## 2010-01-07 -0.015906307 -0.0061369188 -0.018115688 -0.025208746
## 2010-01-08 -0.001848595 -0.0103997075 -0.017013233 -0.023279489
## 2010-01-11  0.006648338  0.0068965187  0.027076954  0.013331111
##                lag1.JPM     lag1.JNJ     lag1.XOM     lag1.BAC
## 2010-01-06  0.019369943 -0.011595547  0.003904497  0.032504844
## 2010-01-07  0.005494460  0.008133850  0.008643028  0.011728271
## 2010-01-08  0.019808812 -0.007137239 -0.003141874  0.032946982
## 2010-01-11 -0.002455928  0.003437990 -0.004011547 -0.008859953
##               lag2.AAPL     lag2.MSFT    lag2.AMZN     lag2.GOOG
## 2010-01-06  0.002717514  0.0107772694 -0.017247750 -0.0003189411
## 2010-01-07  0.001728854  0.0003230371  0.005899985 -0.0044037068
## 2010-01-08 -0.015906307 -0.0061369188 -0.018115688 -0.0252087463
## 2010-01-11 -0.001848595 -0.0103997075 -0.017013233 -0.0232794894
##              lag2.JPM      lag2.JNJ     lag2.XOM   lag2.BAC d.SPY d.AAPL
## 2010-01-06 0.02536485 -0.0004635914  0.006257290 0.02952756     1     -1
## 2010-01-07 0.01936994 -0.0115955473  0.003904497 0.03250484     1     -1
## 2010-01-08 0.00549446  0.0081338495  0.008643028 0.01172827     1      1
## 2010-01-11 0.01980881 -0.0071372385 -0.003141874 0.03294698     1     -1
##            d.MSFT d.AMZN d.GOOG d.JPM d.JNJ d.XOM d.BAC
## 2010-01-06     -1     -1     -1     1     1     1     1
## 2010-01-07     -1     -1     -1     1    -1    -1     1
## 2010-01-08      1      1      1    -1     1    -1    -1
## 2010-01-11     -1     -1     -1    -1     1     1     1
```

Question 2:

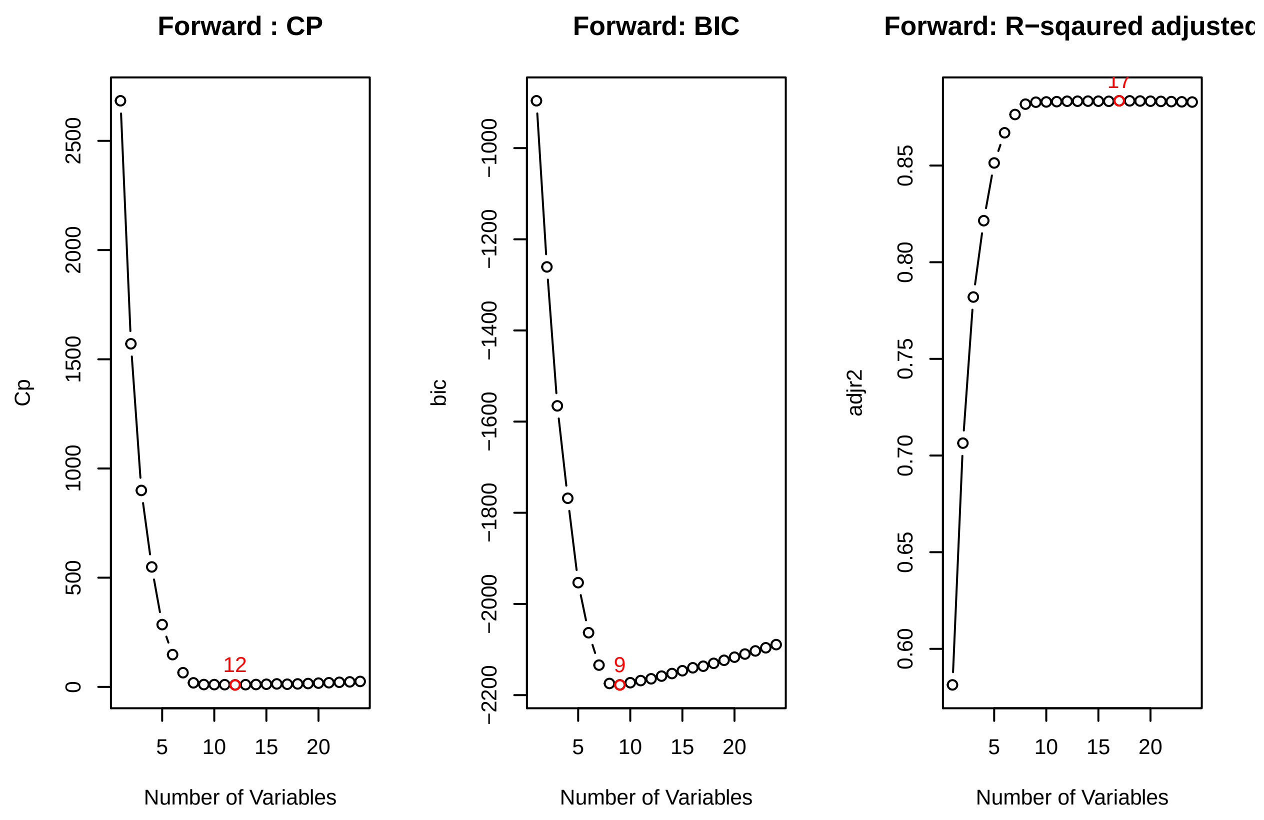
Pick a quantitative variable and fit at least four different models in order to predict that variable using the other predictors. Determine which of the models is the best fit. You will need to provide strong reason why the particular model you chose is the best one. You will need to confirm the model 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.

For quantitive data, we use the second dataset which is return dataset. We peform fitting model in the purpose of SPY ETF return prediction.

In the first step, we select the predictors by useing regsubsets function with forward, backword and exhaustive methods. The result shows that the number of 9 predictors is the optimal number including the return of APPL, BAC, GOOG, AMZN, JNJ, XOM, JPM, MSFT and lag2 return of JNJ.

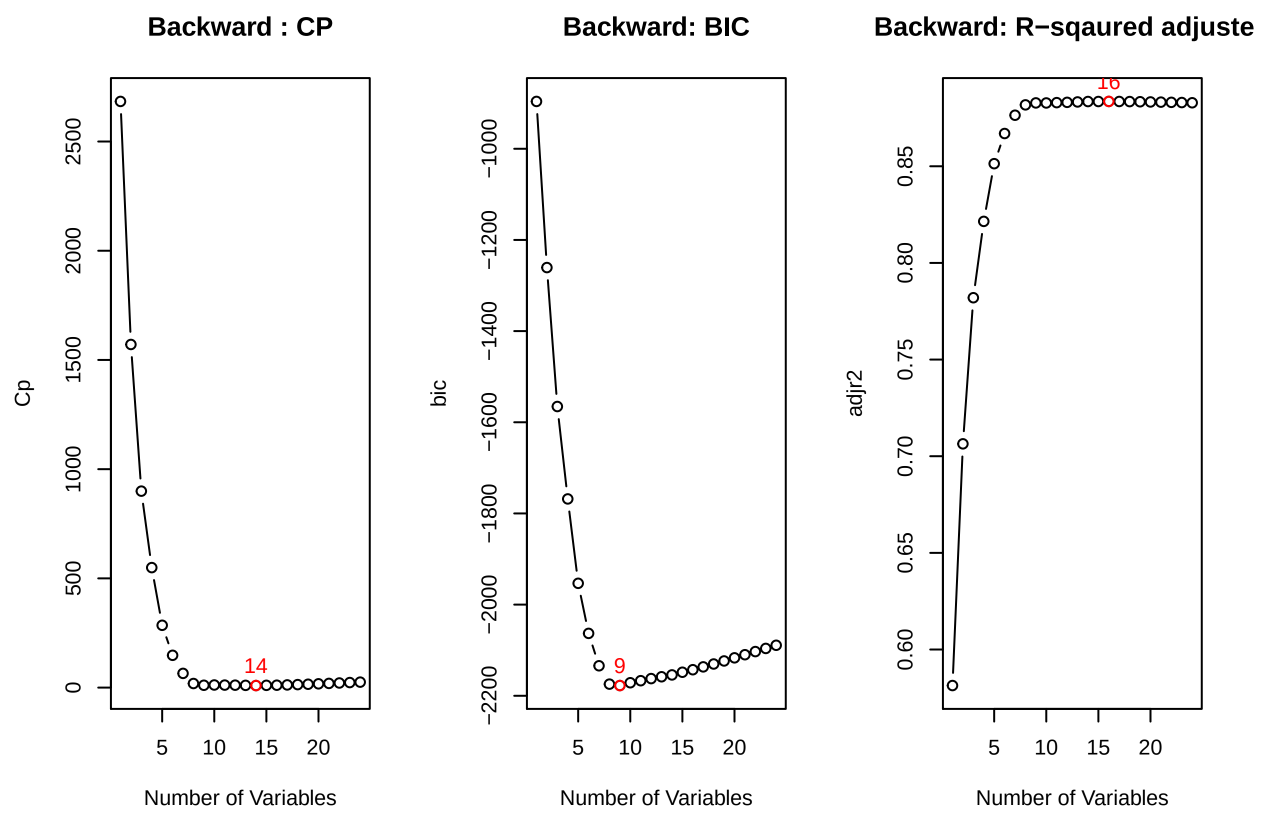
```r
library(leaps)
set.seed(personal)

train.index <- sample(1:nrow(sp500), nrow(sp500)/2)
test.index <- -train.index

sp500.qt.train <- sp500[train.index,1:(9+8+8)]
sp500.qt.test <- sp500[test.index,1:(9+8+8)]
```

```r
##### Forward
forward.fit.qt = regsubsets(r.SPY ~ ., data = sp500.qt.train, nvmax = (9+8+8), method = "forward")
forward.summary.qt = summary(forward.fit.qt)

par(mfrow = c(1, 3))

min.cp.qt = which.min(forward.summary.qt$cp)
plot(forward.summary.qt$cp, xlab = "Number of Variables", ylab = "Cp",
     type = "b",
main = "Forward : CP")
points(min.cp.qt,forward.summary.qt$cp[min.cp.qt ],col = "red")
text(min.cp.qt,forward.summary.qt$cp[min.cp.qt ],as.character(min.cp.qt),
     pos=3, col = "red")

min.bic.qt = which.min(forward.summary.qt$bic)
plot(forward.summary.qt$bic, xlab = "Number of Variables", ylab = "bic",
     type = "b",
main = "Forward: BIC")
points(min.bic.qt,forward.summary.qt$bic[min.bic.qt ],col = "red")
text(min.bic.qt,forward.summary.qt$bic[min.bic.qt ],as.character(min.bic.qt),
     pos=3, col = "red")

max.adjr2.qt = which.max(forward.summary.qt$adjr2)
plot(forward.summary.qt$adjr2, xlab = "Number of Variables", ylab = "adjr2",
     type = "b",
main = "Forward: R-sqaured adjusted")
points(max.adjr2.qt,forward.summary.qt$adjr2[max.adjr2.qt],col = "red")
text(max.adjr2.qt,forward.summary.qt$adjr2[max.adjr2.qt],as.character(max.adjr2.qt),
     pos=3, col = "red")
```



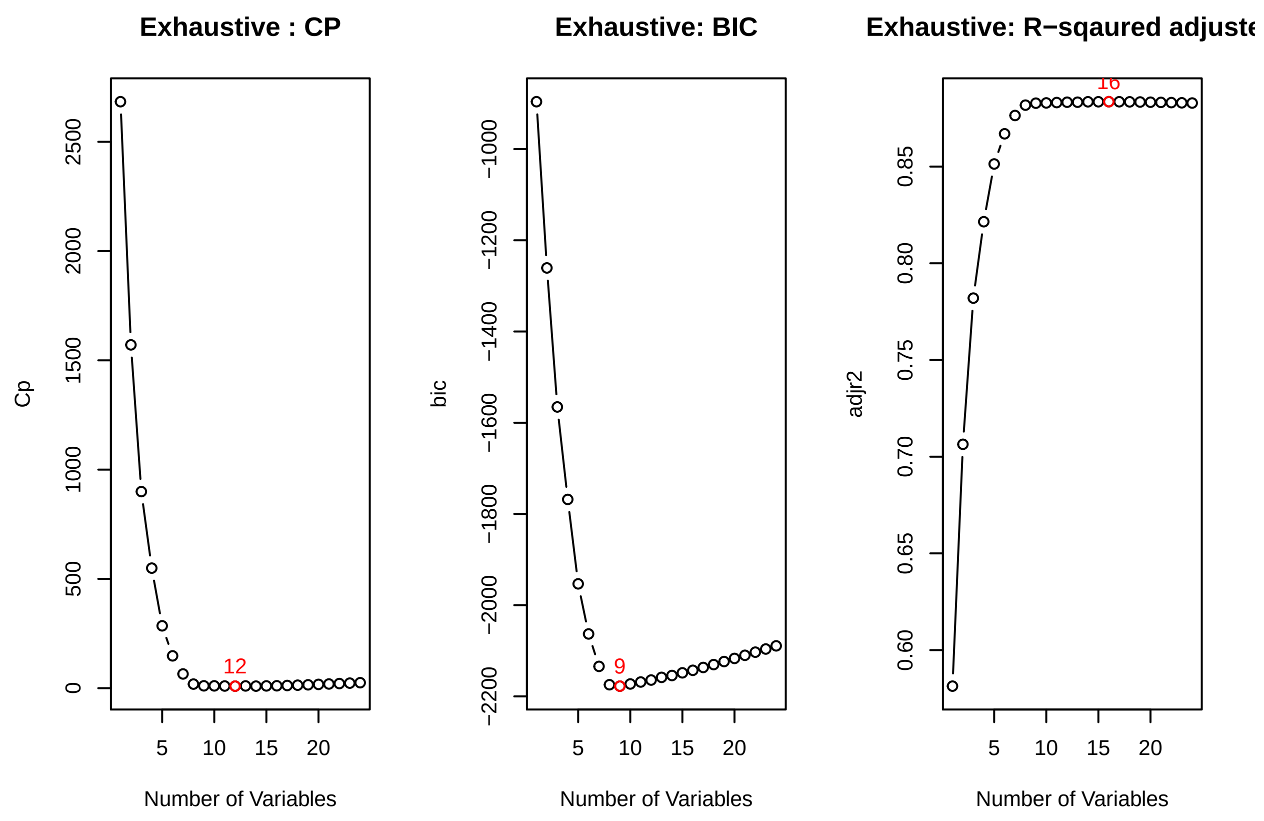
```
##### Backward
backward.fit.qt = regsubsets(r.SPY ~ ., data = sp500.qt.train, nvmax = (9+8+8),
                             method = "backward")
backward.summary.qt = summary(backward.fit.qt)

par(mfrow = c(1, 3))

min.cp.qt = which.min(backward.summary.qt$cp)
plot(backward.summary.qt$cp, xlab = "Number of Variables", ylab = "Cp",
     type = "b",
main = "Backward : CP")
points(min.cp.qt,backward.summary.qt$cp[min.cp.qt ],col = "red")
text(min.cp.qt,backward.summary.qt$cp[min.cp.qt ],as.character(min.cp.qt), pos=3,
     col = "red")

min.bic.qt = which.min(backward.summary.qt$bic)
plot(backward.summary.qt$bic, xlab = "Number of Variables", ylab = "bic",
     type = "b",
main = "Backward: BIC")
points(min.bic.qt,backward.summary.qt$bic[min.bic.qt ],col = "red")
text(min.bic.qt,backward.summary.qt$bic[min.bic.qt ],as.character(min.bic.qt), pos=3,
     col = "red")

max.adjr2.qt = which.max(backward.summary.qt$adjr2)
plot(backward.summary.qt$adjr2, xlab = "Number of Variables", ylab = "adjr2",
     type = "b",
```

```r
main = "Backward: R-sqaured adjusted")
points(max.adjr2.qt,backward.summary.qt$adjr2[max.adjr2.qt],col = "red")
text(max.adjr2.qt,backward.summary.qt$adjr2[max.adjr2.qt],as.character(max.adjr2.qt),
     pos=3, col = "red")
```



```r
##### Exhaustive
exhaustive.fit.qt = regsubsets(r.SPY ~ ., data = sp500.qt.train, nvmax = (9+8+8),
                               method = "exhaustive")
exhaustive.summary.qt = summary(exhaustive.fit.qt)

par(mfrow = c(1, 3))

min.cp.qt = which.min(exhaustive.summary.qt$cp)
plot(exhaustive.summary.qt$cp, xlab = "Number of Variables", ylab = "Cp",
     type = "b",
main = "Exhaustive : CP")
points(min.cp.qt,exhaustive.summary.qt$cp[min.cp.qt ],col = "red")
text(min.cp.qt,exhaustive.summary.qt$cp[min.cp.qt ],as.character(min.cp.qt), pos=3,
     col = "red")

min.bic.qt = which.min(exhaustive.summary.qt$bic)
plot(exhaustive.summary.qt$bic, xlab = "Number of Variables", ylab = "bic",
     type = "b", main = "Exhaustive: BIC")
points(min.bic.qt,exhaustive.summary.qt$bic[min.bic.qt ],col = "red")
text(min.bic.qt,exhaustive.summary.qt$bic[min.bic.qt ],as.character(min.bic.qt),
     pos=3, col = "red")
```

```
max.adjr2.qt = which.max(exhaustive.summary.qt$adjr2)
plot(exhaustive.summary.qt$adjr2, xlab = "Number of Variables", ylab = "adjr2",
     type = "b", main = "Exhaustive: R-sqaured adjusted")
points(max.adjr2.qt,exhaustive.summary.qt$adjr2[max.adjr2.qt],col = "red")
text(max.adjr2.qt,exhaustive.summary.qt$adjr2[max.adjr2.qt],as.character(max.adjr2.qt),
     pos=3, col = "red")
```



```
forward.coefi.qt = sort(abs(coef(forward.fit.qt, id = 24)), TRUE)[1:9]
names(forward.coefi.qt)
```

```
## [1] "r.JNJ"    "r.XOM"    "r.JPM"    "r.MSFT"   "r.AAPL"   "r.BAC"
## [7] "r.GOOG"   "lag2.JNJ" "r.AMZN"
```

```
backward.coefi.qt = sort(abs(coef(backward.fit.qt, id = 24)), TRUE)[1:9]
names(backward.coefi.qt)
```

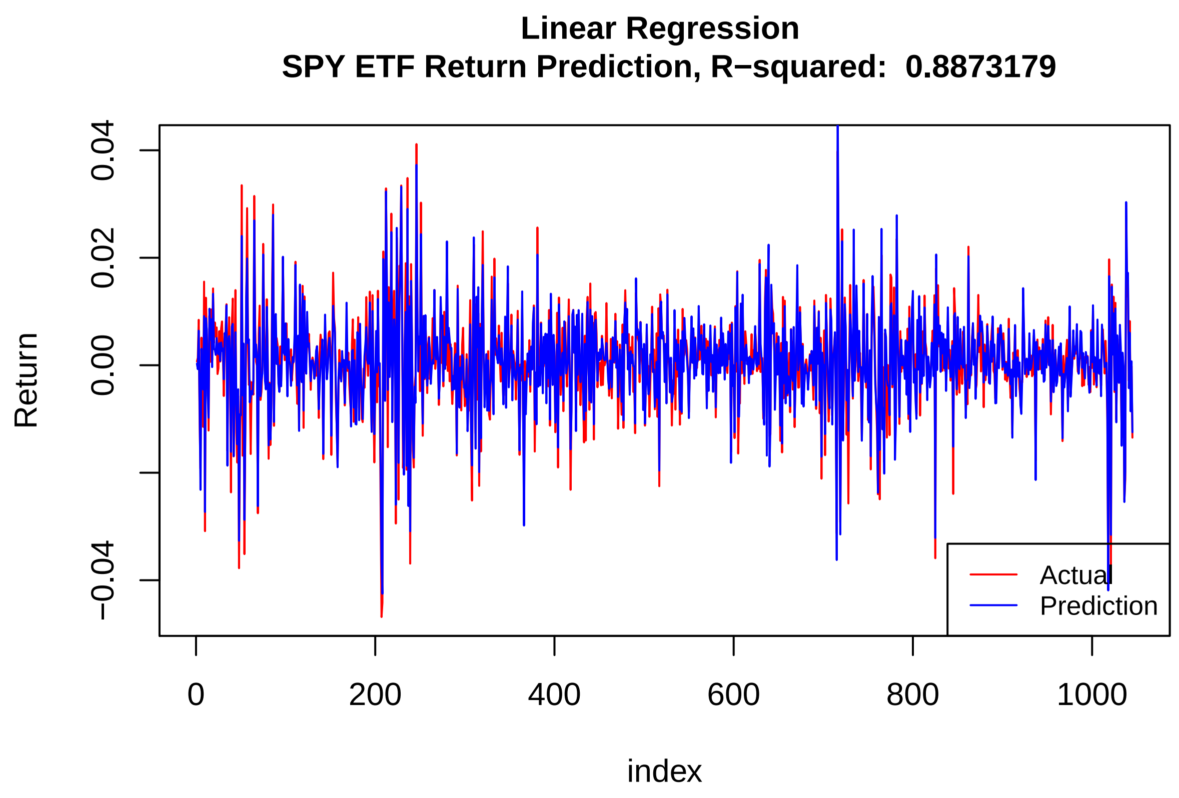
```
## [1] "r.JNJ"    "r.XOM"    "r.JPM"    "r.MSFT"   "r.AAPL"   "r.BAC"
## [7] "r.GOOG"   "lag2.JNJ" "r.AMZN"
```

```
exhaustive.coefi.qt = sort(abs(coef(exhaustive.fit.qt, id = 24)), TRUE)[1:9]
names(exhaustive.coefi.qt)
```

```
## [1] "r.JNJ"    "r.XOM"    "r.JPM"    "r.MSFT"   "r.AAPL"   "r.BAC"
## [7] "r.GOOG"   "lag2.JNJ" "r.AMZN"
```

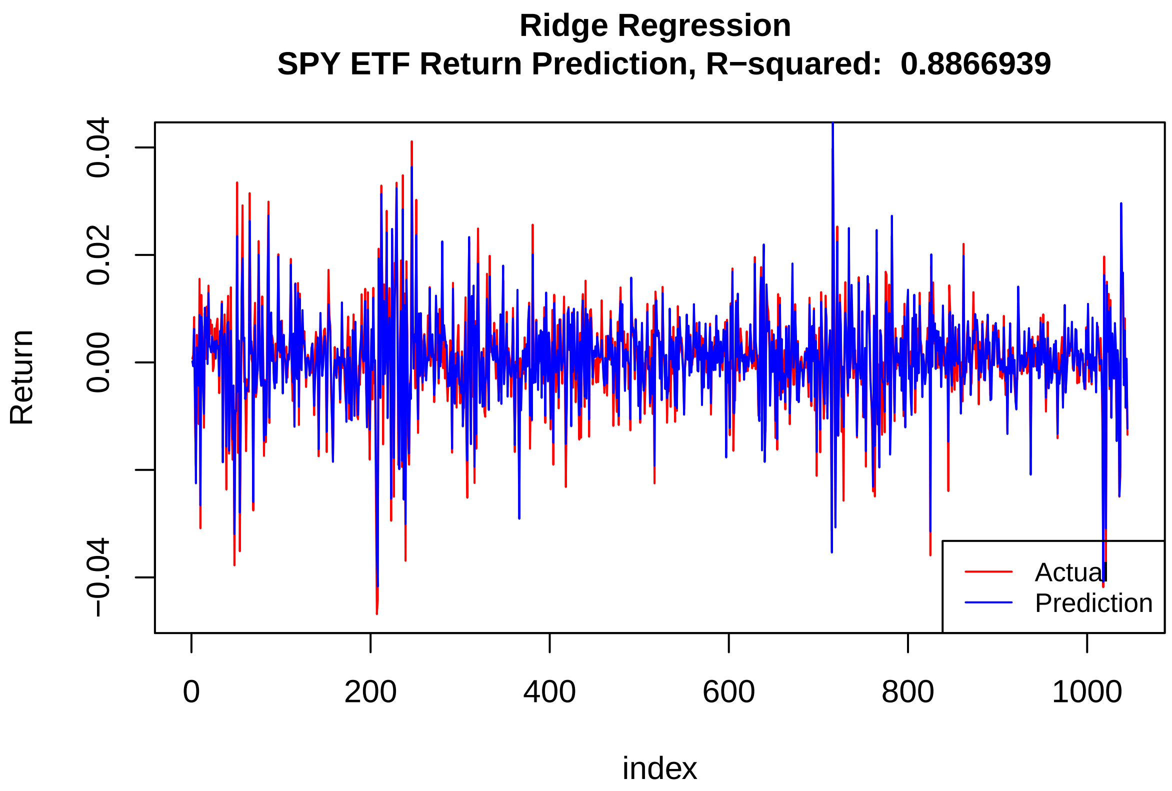
We use 4 models to fit which are Multiple linear regression, Ridge regression, Lasso regression and Bagging. The result shows that the Lasso regression is the best model with the highest r-squared. We will use Lasso regression to perfrom our strategy.

```
##### Multiple Linear regression
par(mfrow = c(1, 1))
lm.fit.qt <- lm(r.SPY ~ r.AAPL + r.MSFT + r.AMZN + r.GOOG +
                    r.JPM + r.JNJ + r.XOM + r.BAC + lag2.JNJ, data = sp500.qt.train)

lm.pred.qt <- predict(lm.fit.qt, sp500.qt.test)

r2.lm.qt <- 1 - (sum((sp500.qt.test$r.SPY-lm.pred.qt)^2)/
                     sum((sp500.qt.test$r.SPY-mean(sp500.qt.test$r.SPY))^2))

plot(1:length(sp500.qt.test$r.SPY), sp500.qt.test$r.SPY, type = "l", col = "red",
     xlab = "index", ylab = "Return",
     main = paste("Linear Regression \n SPY ETF Return Prediction, R-squared: ",
                  round(r2.lm.qt ,7)), cex.main = 1)
lines(1:length(lm.pred.qt),lm.pred.qt, type = "l", col = "blue")
legend("bottomright", legend = c("Actual", "Prediction"),
       col = c("red", "blue"), lty = c(1,1) , cex = 0.8)
```



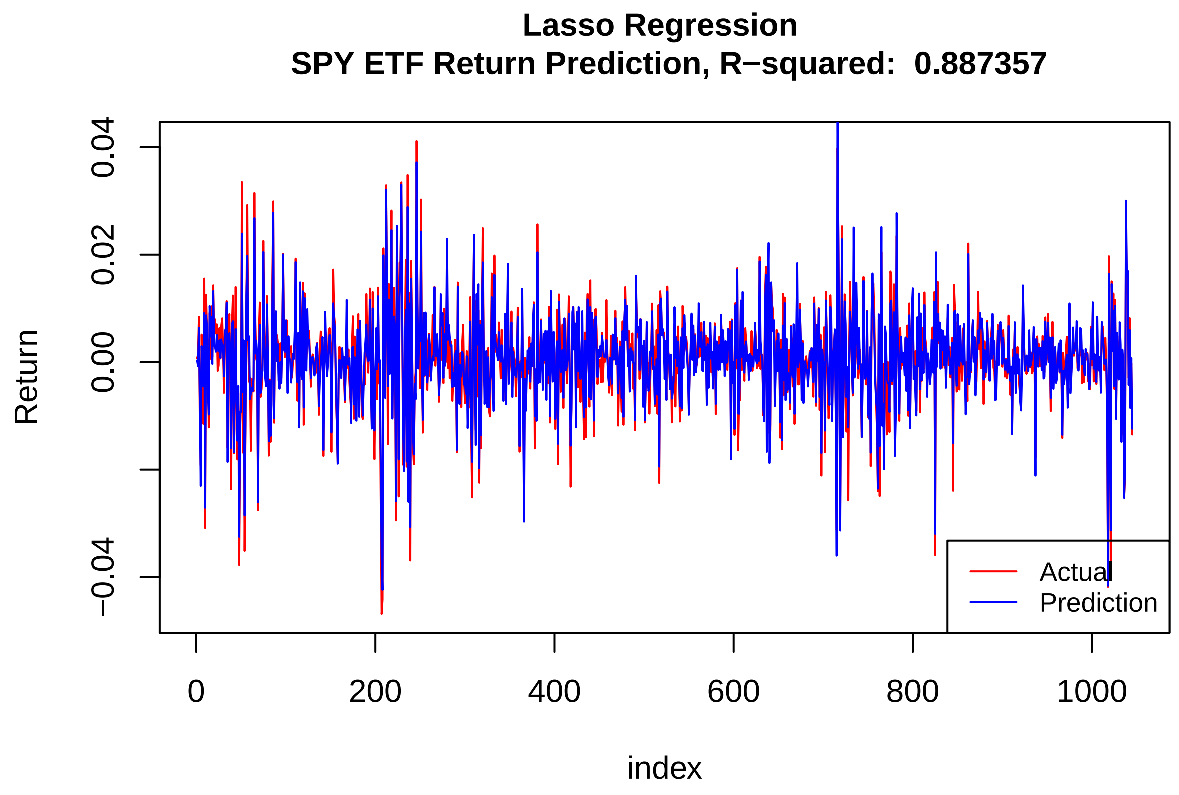
```
##### Ridge Regression
library(glmnet)

x.ridge = model.matrix(r.SPY ~ r.AAPL + r.MSFT + r.AMZN + r.GOOG +
                           r.JPM + r.JNJ + r.XOM + r.BAC + lag2.JNJ, data = sp500.qt.train)
y.ridge = sp500.qt.train$r.SPY
```

```r
cv.out.ridge <- cv.glmnet(x.ridge, y.ridge, alpha = 0)
bestlam.ridge <- cv.out.ridge$lambda.min

x.ridge.test = model.matrix(r.SPY ~ r.AAPL + r.MSFT + r.AMZN + r.GOOG +
                              r.JPM + r.JNJ + r.XOM + r.BAC + lag2.JNJ, data = sp500.qt.test)
ridge.fit = glmnet(x.ridge, y.ridge, alpha = 0)
ridge.pred = predict(ridge.fit, s = bestlam.ridge, newx = x.ridge.test)

r2.ridge.qt <- 1 - (sum((sp500.qt.test$r.SPY-ridge.pred)^2)/
                        sum((sp500.qt.test$r.SPY-mean(sp500.qt.test$r.SPY))^2))

plot(1:length(sp500.qt.test$r.SPY), sp500.qt.test$r.SPY, type = "l",
     col = "red", xlab = "index", ylab = "Return",
     main = paste("Ridge Regression \n SPY ETF Return Prediction, R-squared: ",
                  round(r2.ridge.qt ,7)), cex.main = 1)
lines(1:length(ridge.pred),ridge.pred, type = "l", col = "blue")
legend("bottomright", legend = c("Actual", "Prediction"), col = c("red", "blue"),
       lty = c(1,1) , cex = 0.8)
```



```r
##### Lasso Regression
library(glmnet)

x.lasso = model.matrix(r.SPY ~ r.AAPL + r.MSFT + r.AMZN + r.GOOG +
                         r.JPM + r.JNJ + r.XOM + r.BAC + lag2.JNJ, data = sp500.qt.train)
y.lasso = sp500.qt.train$r.SPY
```

```r
cv.out.lasso <- cv.glmnet(x.lasso, y.lasso, alpha = 1)
bestlam.lasso <- cv.out.lasso$lambda.min

x.lasso.test = model.matrix(r.SPY ~ r.AAPL + r.MSFT + r.AMZN + r.GOOG +
                                r.JPM + r.JNJ + r.XOM + r.BAC +
                                lag2.JNJ, data = sp500.qt.test)
lasso.fit = glmnet(x.lasso, y.lasso, alpha = 1)
lasso.pred = predict(lasso.fit, s = bestlam.lasso, newx = x.lasso.test)

r2.lasso.qt <- 1 - (sum((sp500.qt.test$r.SPY-lasso.pred)^2)/
                        sum((sp500.qt.test$r.SPY-mean(sp500.qt.test$r.SPY))^2))

plot(1:length(sp500.qt.test$r.SPY), sp500.qt.test$r.SPY, type = "l",
     col = "red", xlab = "index", ylab = "Return",
     main = paste("Lasso Regression \n SPY ETF Return Prediction, R-squared: ",
                  round(r2.lasso.qt ,7)), cex.main = 1)
lines(1:length(lasso.pred),lasso.pred, type = "l", col = "blue")
legend("bottomright", legend = c("Actual", "Prediction"), col = c("red", "blue"),
       lty = c(1,1) , cex = 0.8)
```



```r
##### Bagging/Random Forest
library(randomForest)
set.seed(personal)

tree.no <- seq(1000, 1900, by = 100)
rf.error <- rep(NA, length = length(tree.no))
```

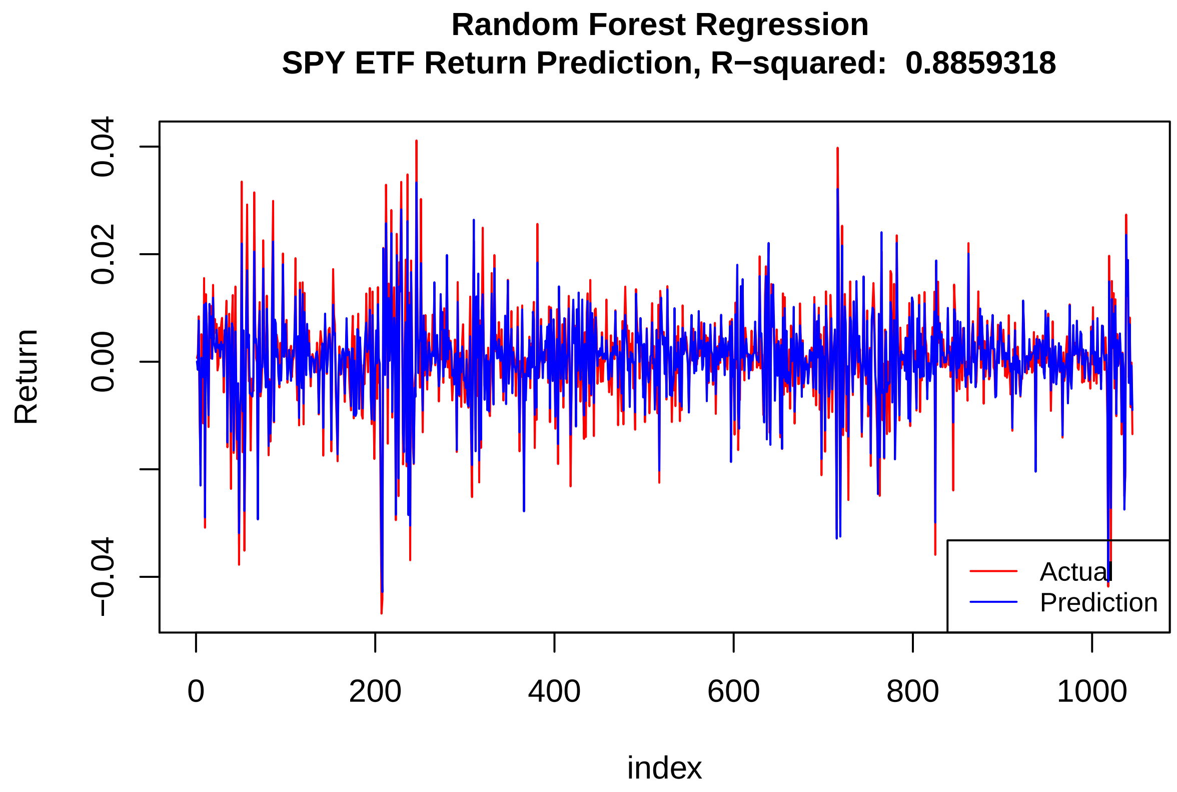
```
for(i in 1:length(tree.no)){
    rf.qt = randomForest(r.SPY ~ r.AAPL + r.MSFT + r.AMZN + r.GOOG +
                              r.JPM + r.JNJ + r.XOM + r.BAC + lag2.JNJ,
                         data = sp500.qt.train, ntree = tree.no[i], mtry = 9)
    rf.pred.qt = predict(rf.qt, sp500.qt.train)
    r2.rf.qt <- 1 - (sum((sp500.qt.train$r.SPY-rf.pred.qt)^2)/
                         sum((sp500.qt.train$r.SPY-mean(sp500.qt.train$r.SPY))^2))
    rf.error[i] <- r2.rf.qt

}

rf.train.error <- rf.error[which.max(rf.error)]
rf.train.error
```

```
## [1] 0.9766731
```

```
best.tree.no <- tree.no[which.max(rf.error)]
best.tree.no
```

```
## [1] 1100
```

```
rf.qt = randomForest(r.SPY ~ r.AAPL + r.MSFT + r.AMZN + r.GOOG +
                         r.JPM + r.JNJ + r.XOM + r.BAC + lag2.JNJ,
                     data = sp500.qt.train, ntree = best.tree.no, mtry = 9)
rf.pred.qt = predict(rf.qt, sp500.qt.test)

r2.rf.qt <- 1 - (sum((sp500.qt.test$r.SPY-rf.pred.qt)^2)/
                     sum((sp500.qt.test$r.SPY-mean(sp500.qt.test$r.SPY))^2))

plot(1:length(sp500.qt.test$r.SPY), sp500.qt.test$r.SPY, type = "l", col = "red",
     xlab = "index", ylab = "Return",
     main = paste("Random Forest Regression \n SPY ETF Return Prediction, R-squared: ",
                  round(r2.rf.qt ,7)), cex.main = 1)
lines(1:length(rf.pred.qt),rf.pred.qt, type = "l", col = "blue")
legend("bottomright", legend = c("Actual", "Prediction"), col = c("red", "blue"),
       lty = c(1,1) , cex = 0.8)
```



```
df.qt.compare <- data.frame(lm = r2.lm.qt, ridge = r2.ridge.qt, lasso = r2.lasso.qt,
                            rf = r2.rf.qt)
df.qt.compare
```

```
##          lm     ridge    lasso        rf
## 1 0.8873179 0.8866939 0.887357 0.8859318
```

Question 3:

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

For qualitative data, we use the third dataset which is direction combined with the second dataset which is return to perform fitting model in the purpose of SPY ETF direction prediction.

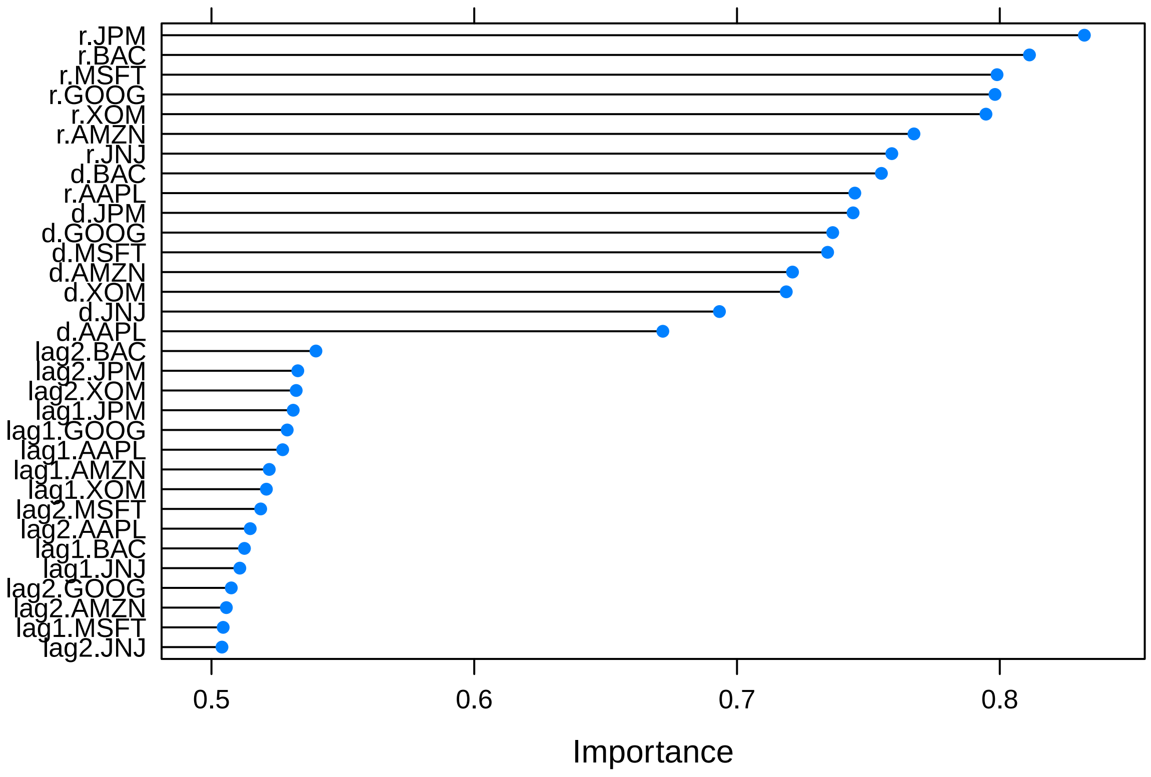
Since we have many variables in the dataset, we need to select the most important predictors. We then use feature selection with random forest to rank the importance of predictiors to the response. The result shows there are 21 important predictors with the highest accuracy of cross-validation as shown in the table below.

```
set.seed(personal)
sp500.ql.train <- sp500[train.index,2:ncol(sp500)]
sp500.ql.test <- sp500[test.index,2:ncol(sp500)]

###### Rank Features By Importance
library(mlbench)
```

```r
library(caret)

# prepare training scheme
control.rfi <- trainControl(method="repeatedcv", number=10, repeats=3)
# train the model
model.rfi <- train(d.SPY ~ ., data=sp500.ql.train, method="lvq", preProcess="scale",
                   trControl=control.rfi)
# estimate variable importance
importance.rfi <- varImp(model.rfi, scale=FALSE)
# summarize importance
print(importance.rfi)
```

```
## ROC curve variable importance
##
##   only 20 most important variables shown (out of 32)
##
##          Importance
## r.JPM        0.8322
## r.BAC        0.8113
## r.MSFT       0.7990
## r.GOOG       0.7982
## r.XOM        0.7947
## r.AMZN       0.7673
## r.JNJ        0.7589
## d.BAC        0.7550
## r.AAPL       0.7448
## d.JPM        0.7442
## d.GOOG       0.7364
## d.MSFT       0.7345
## d.AMZN       0.7211
## d.XOM        0.7188
## d.JNJ        0.6933
## d.AAPL       0.6718
## lag2.BAC     0.5398
## lag2.JPM     0.5329
## lag2.XOM     0.5322
## lag1.JPM     0.5311
```

```r
# plot importance
plot(importance.rfi)
```



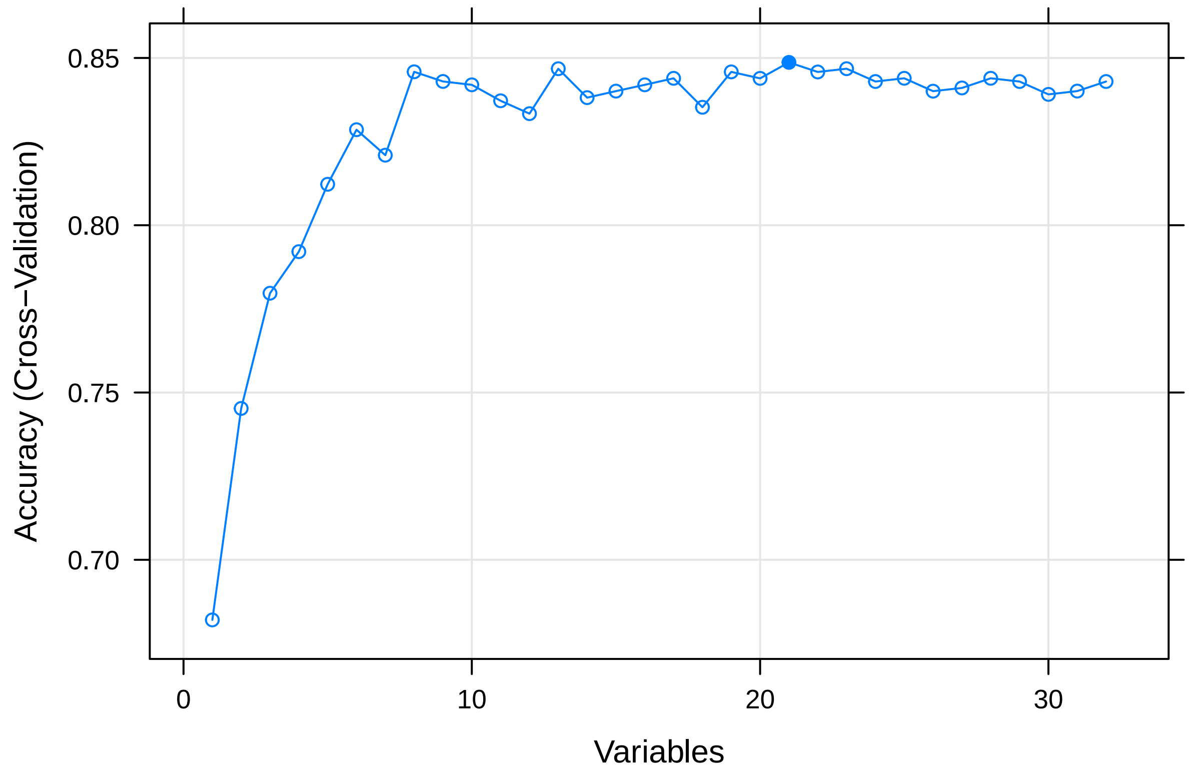
```
###### Feature Selection
# define the control using a random forest selection function
control.rfe <- rfeControl(functions=rfFuncs, method="cv", number=10)
# run the RFE algorithm
results.rfe <- rfe(sp500.ql.train[,-25], sp500.ql.train$d.SPY, sizes=c(1:32),
                   rfeControl=control.rfe)
# summarize the results
print(results.rfe)
```

```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (10 fold)
##
## Resampling performance over subset size:
##
##  Variables Accuracy  Kappa AccuracySD KappaSD Selected
##          1   0.6820 0.3579    0.05000 0.09954
##          2   0.7452 0.4813    0.03975 0.08035
##          3   0.7797 0.5520    0.04747 0.09636
##          4   0.7921 0.5781    0.05220 0.10601
##          5   0.8122 0.6183    0.04426 0.09000
##          6   0.8286 0.6522    0.03674 0.07471
##          7   0.8209 0.6355    0.03611 0.07245
##          8   0.8459 0.6869    0.03476 0.06990
##          9   0.8430 0.6815    0.03702 0.07452
##          10   0.8420 0.6785    0.04247 0.08614
##          11   0.8372 0.6695    0.04104 0.08230
##          12   0.8334 0.6623    0.04260 0.08510
##          13   0.8468 0.6887    0.03633 0.07357
##          14   0.8381 0.6711    0.03702 0.07463
##          15   0.8401 0.6753    0.03565 0.07193
##          16   0.8420 0.6788    0.03594 0.07277
##          17   0.8439 0.6829    0.03867 0.07817
##          18   0.8353 0.6659    0.04448 0.08971
##          19   0.8458 0.6875    0.04255 0.08541
##          20   0.8439 0.6837    0.04057 0.08129
##          21   0.8487 0.6927    0.03371 0.06781        *
##          22   0.8458 0.6871    0.03377 0.06719
##          23   0.8468 0.6892    0.03786 0.07533
##          24   0.8430 0.6813    0.03428 0.06880
##          25   0.8439 0.6828    0.03232 0.06514
##          26   0.8401 0.6758    0.03688 0.07398
##          27   0.8410 0.6774    0.03769 0.07556
##          28   0.8439 0.6832    0.03650 0.07299
##          29   0.8429 0.6810    0.03334 0.06640
##          30   0.8391 0.6737    0.03759 0.07520
##          31   0.8401 0.6750    0.03760 0.07532
##          32   0.8430 0.6811    0.03113 0.06221
##
## The top 5 variables (out of 21):
##    r.JPM, r.BAC, r.XOM, r.MSFT, r.GOOG
```

```
# list the chosen features
predictors(results.rfe)
```

```
##  [1] "r.JPM"     "r.BAC"     "r.XOM"     "r.MSFT"    "r.GOOG"
##  [6] "r.AMZN"    "r.AAPL"    "r.JNJ"     "d.BAC"     "d.MSFT"
## [11] "d.JPM"     "d.GOOG"    "d.AMZN"    "d.XOM"     "d.JNJ"
## [16] "d.AAPL"    "lag2.JPM"  "lag2.JNJ"  "lag2.XOM"  "lag1.AMZN"
## [21] "lag1.JPM"
```

```
# plot the results
plot(results.rfe, type=c("g", "o"))
```



We use 4 models to fit which are Logistic regression, Linear discriminant analysis, Quadratic discriminant analysis and KNN. The result shows that the Logistic regression is the best model with the lowest error. We will use Logistic regression to perfrom our strategy.

```
###### Logistic Regression
library(MASS)

glm.fit.ql <- glm(d.SPY ~ r.JPM + r.BAC + r.XOM + r.MSFT + r.GOOG +
                      r.AMZN + r.AAPL + r.JNJ + d.BAC + d.MSFT + d.JPM + d.GOOG +
                      d.AMZN + d.XOM + d.JNJ + d.AAPL + lag2.JPM + lag2.JNJ +
                      lag2.XOM + lag1.AMZN + lag1.JPM, family=binomial,data=sp500.ql.train)

glm.prob.ql <- predict(glm.fit.ql, sp500.ql.test, type="response")
glm.pred.ql <- rep(1, dim(sp500.ql.test)[1])
glm.pred.ql[glm.prob.ql < 0.5] <- -1
table(glm.pred.ql, sp500.ql.test$d.SPY)
```

```
##
## glm.pred.ql  -1   1
##          -1 370  58
##          1  103 514
```

```
mean.error.glm <- mean(glm.pred.ql != sp500.ql.test$d.SPY)
mean.error.glm
```

```
## [1] 0.154067
```

```
###### Linear discriminant Analysis

lda.fit.ql <- lda(d.SPY ~ r.JPM + r.BAC + r.XOM + r.MSFT + r.GOOG +
                      r.AMZN + r.AAPL + r.JNJ + d.BAC + d.MSFT + d.JPM + d.GOOG +
                      d.AMZN + d.XOM + d.JNJ + d.AAPL + lag2.JPM + lag2.JNJ +
                      lag2.XOM + lag1.AMZN + lag1.JPM, data = sp500.ql.train)

lda.pred.ql <- predict(lda.fit.ql, sp500.ql.test)
table(lda.pred.ql$class, sp500.ql.test$d.SPY)
```

```
##
##       -1   1
##   -1 373  65
##   1  100 507
```

```
mean.error.lda <- mean(lda.pred.ql$class != sp500.ql.test$d.SPY)
mean.error.lda
```

```
## [1] 0.1578947
```

```
###### Quadratic Discriminant Analysis

qda.fit.ql <- qda(d.SPY ~ r.JPM + r.BAC + r.XOM + r.MSFT + r.GOOG +
                      r.AMZN + r.AAPL + r.JNJ + d.BAC + d.MSFT + d.JPM + d.GOOG +
                      d.AMZN + d.XOM + d.JNJ + d.AAPL + lag2.JPM + lag2.JNJ +
                      lag2.XOM + lag1.AMZN + lag1.JPM, data = sp500.ql.train)

qda.pred.ql <- predict(qda.fit.ql, sp500.ql.test)
table(qda.pred.ql$class, sp500.ql.test$d.SPY)
```

```
##
##       -1   1
##   -1 370  97
##   1  103 475
```

```
mean.error.qda <- mean(qda.pred.ql$class != sp500.ql.test$d.SPY)
mean.error.qda
```

```
## [1] 0.1913876
```

```
###### KNN

knn.train.ql <- data.frame(sp500.ql.train$r.JPM, sp500.ql.train$r.BAC,
                           sp500.ql.train$r.XOM, sp500.ql.train$r.MSFT,
                           sp500.ql.train$r.GOOG, sp500.ql.train$r.AMZN,
                           sp500.ql.train$r.AAPL,
                           sp500.ql.train$r.JNJ, sp500.ql.train$d.BAC,
                           sp500.ql.train$d.MSFT, sp500.ql.train$d.JPM,
                           sp500.ql.train$d.GOOG,
                           sp500.ql.train$d.AMZN, sp500.ql.train$d.XOM,
                           sp500.ql.train$d.JNJ, sp500.ql.train$d.AAPL,
                           sp500.ql.train$lag2.JPM,
                           sp500.ql.train$lag2.JNJ, sp500.ql.train$lag2.XOM,
                           sp500.ql.train$lag1.AMZN,
                           sp500.ql.train$lag1.JPM)
colnames(knn.train.ql) <- c("r.JPM", "r.BAC", "r.XOM", "r.MSFT", "r.GOOG",
```

```r
                            "r.AMZN", "r.AAPL", "r.JNJ", "d.BAC", "d.MSFT", "d.JPM",
                            "d.GOOG", "d.AMZN", "d.XOM", "d.JNJ", "d.AAPL", "lag2.JPM",
                            "lag2.JNJ", "lag2.XOM", "lag1.AMZN", "lag1.JPM")

knn.test.ql <- data.frame(sp500.ql.test$r.JPM, sp500.ql.test$r.BAC,
                          sp500.ql.test$r.XOM, sp500.ql.test$r.MSFT,
                          sp500.ql.test$r.GOOG, sp500.ql.test$r.AMZN,
                          sp500.ql.test$r.AAPL, sp500.ql.test$r.JNJ,
                          sp500.ql.test$d.BAC, sp500.ql.test$d.MSFT,
                          sp500.ql.test$d.JPM, sp500.ql.test$d.GOOG,
                          sp500.ql.test$d.AMZN, sp500.ql.test$d.XOM,
                          sp500.ql.test$d.JNJ, sp500.ql.test$d.AAPL,
                          sp500.ql.test$lag2.JPM, sp500.ql.test$lag2.JNJ,
                          sp500.ql.test$lag2.XOM, sp500.ql.test$lag1.AMZN,
                          sp500.ql.test$lag1.JPM)
colnames(knn.test.ql) <- c("r.JPM", "r.BAC", "r.XOM", "r.MSFT", "r.GOOG",
                           "r.AMZN", "r.AAPL", "r.JNJ", "d.BAC", "d.MSFT", "d.JPM",
                           "d.GOOG", "d.AMZN", "d.XOM", "d.JNJ", "d.AAPL", "lag2.JPM",
                           "lag2.JNJ", "lag2.XOM", "lag1.AMZN", "lag1.JPM")

library(class)

k <- seq(1, 30, by = 1)
knn.error <- rep(NA, length(k))

for(i in k){
    knn.fit.ql <- knn(data.frame(knn.train.ql),
    data.frame(knn.test.ql), sp500.ql.train$d.SPY, k=i)
    knn.pred.ql <- knn.fit.ql
    table(knn.pred.ql, sp500.ql.test$d.SPY)

    mean.error.knn <- mean(knn.pred.ql != sp500.ql.test$d.SPY)
    mean.error.knn
    knn.error[i] <- mean.error.knn
}
min.knn <- which.min(knn.error)

knn.fit.ql <- knn(data.frame(knn.train.ql),
data.frame(knn.test.ql), sp500.ql.train$d.SPY, k=min.knn)
knn.pred.ql <- knn.fit.ql
table(knn.pred.ql, sp500.ql.test$d.SPY)
```

```
##
## knn.pred.ql  -1   1
##          -1 370  72
##          1  103 500
```

```r
mean.error.knn <- mean(knn.pred.ql != sp500.ql.test$d.SPY)
mean.error.knn
```

```
## [1] 0.1674641
```

```r
result<- data.frame(err.glm = mean.error.glm,
err.lda = mean.error.lda, err.qda = mean.error.qda,
err.knn = mean.error.knn)
```

```
row.names(result) <- "percent.error"
t(result)
```

```
##         percent.error
## err.glm     0.1540670
## err.lda     0.1578947
## err.qda     0.1913876
## err.knn     0.1674641
```

For our strategy, we are going to use the prediction of return and direction of SPY ETF to determine the buy signal and sell signal. In our idea, when the return becomes very low with down direction, it is time for us to buy, and when return becomes very high with up direction, it is time for us to sell.

We pick the buy and sell boundary from the distribution of return that we gain from Lasso regression.

```
sp500.price.backtest <- sp500.close[3:nrow(sp500.close),]

x.lasso.bt = model.matrix(r.SPY ~ r.AAPL + r.MSFT + r.AMZN + r.GOOG +
                              r.JPM + r.JNJ + r.XOM + r.BAC + lag2.JNJ, data = sp500)
y.lasso.bt = sp500$r.SPY

lasso.pred.backtest = predict(lasso.fit, s = bestlam.lasso, newx = x.lasso.bt)
r2.lasso.bt <- 1 - (sum((sp500$r.SPY-lasso.pred.backtest)^2)/
                        sum((sp500$r.SPY-mean(sp500$r.SPY))^2))

glm.prob.bt <- predict(glm.fit.ql, sp500, type="response")
glm.pred.bt <- rep(1, dim(sp500)[1])
glm.pred.bt[glm.prob.bt < 0.5] <- -1
table(glm.pred.bt , sp500$d.SPY)
```

```
##
## glm.pred.bt   -1    1
##          -1  755  125
##          1   182 1027
```

```
mean(glm.pred.bt != sp500$d.SPY)
```

```
## [1] 0.1469603
```

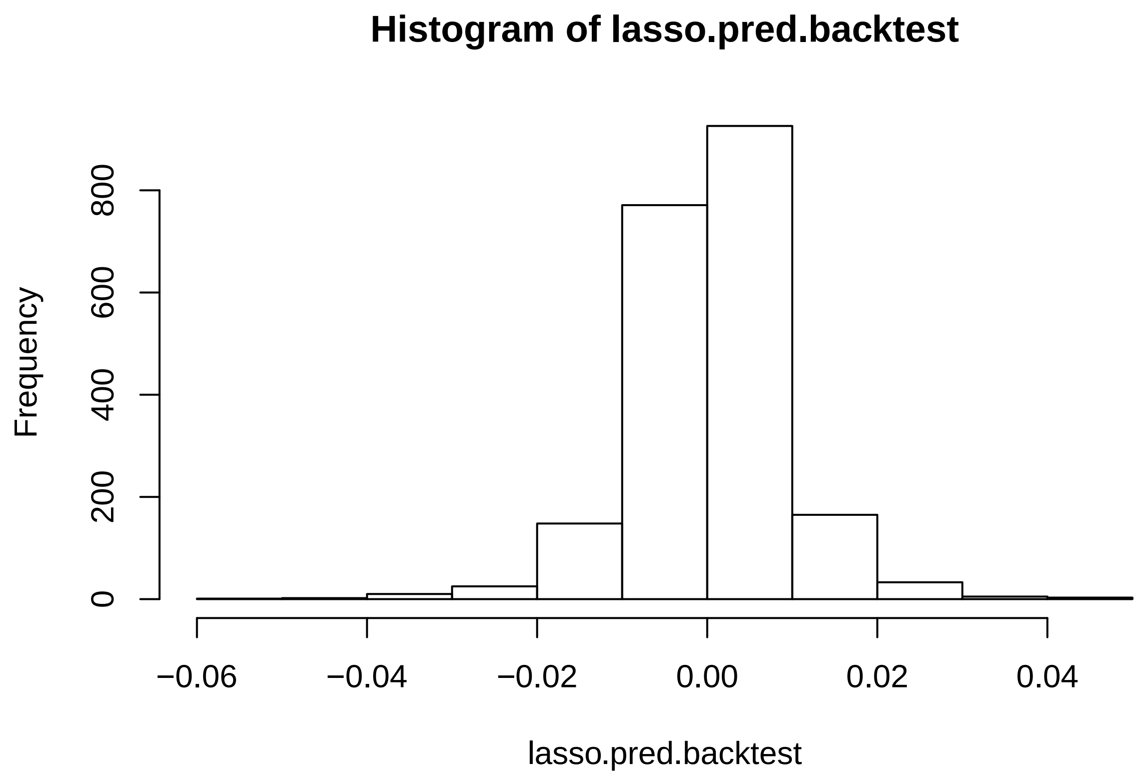
```
summary(lasso.pred.backtest)
```

```
##        1
##  Min.   :-0.0569192
##  1st Qu.:-0.0035216
##  Median : 0.0005024
##  Mean   : 0.0004569
##  3rd Qu.: 0.0049096
##  Max.   : 0.0493598
```

```
sd(lasso.pred.backtest)
```

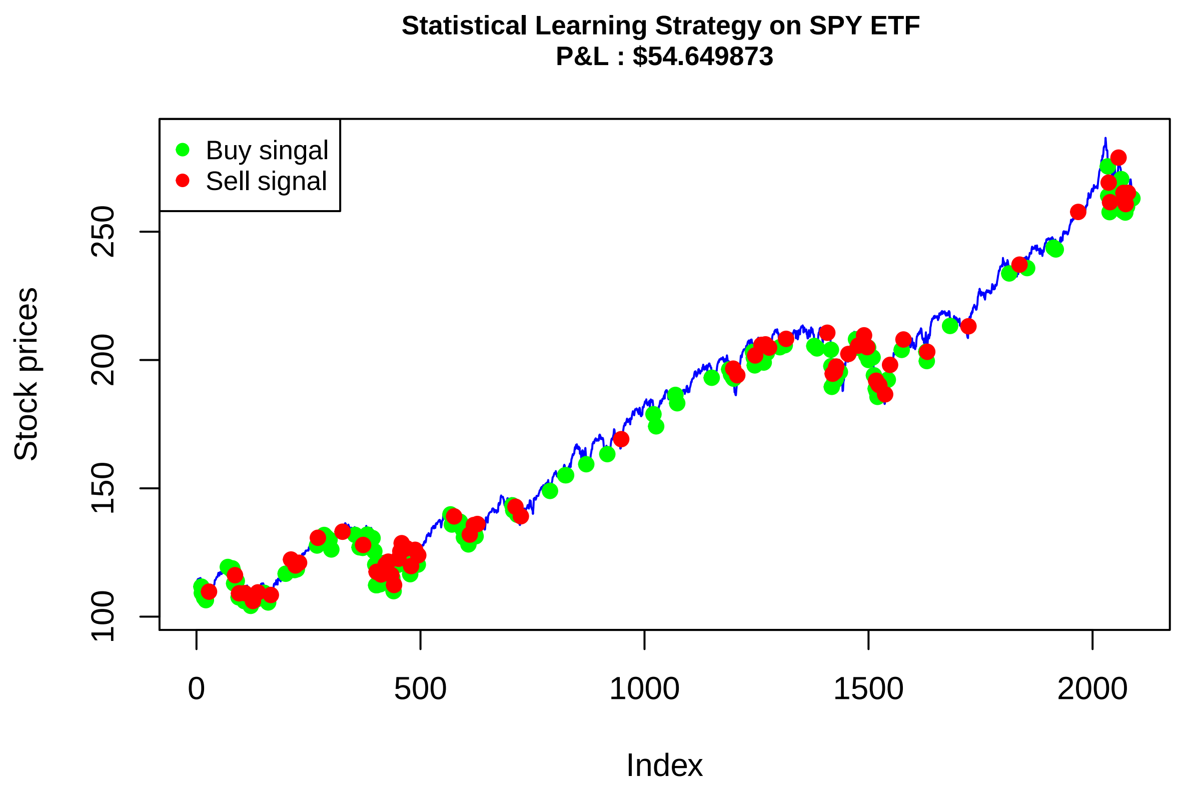
```
## [1] 0.008743504
```

```
hist(lasso.pred.backtest)
```



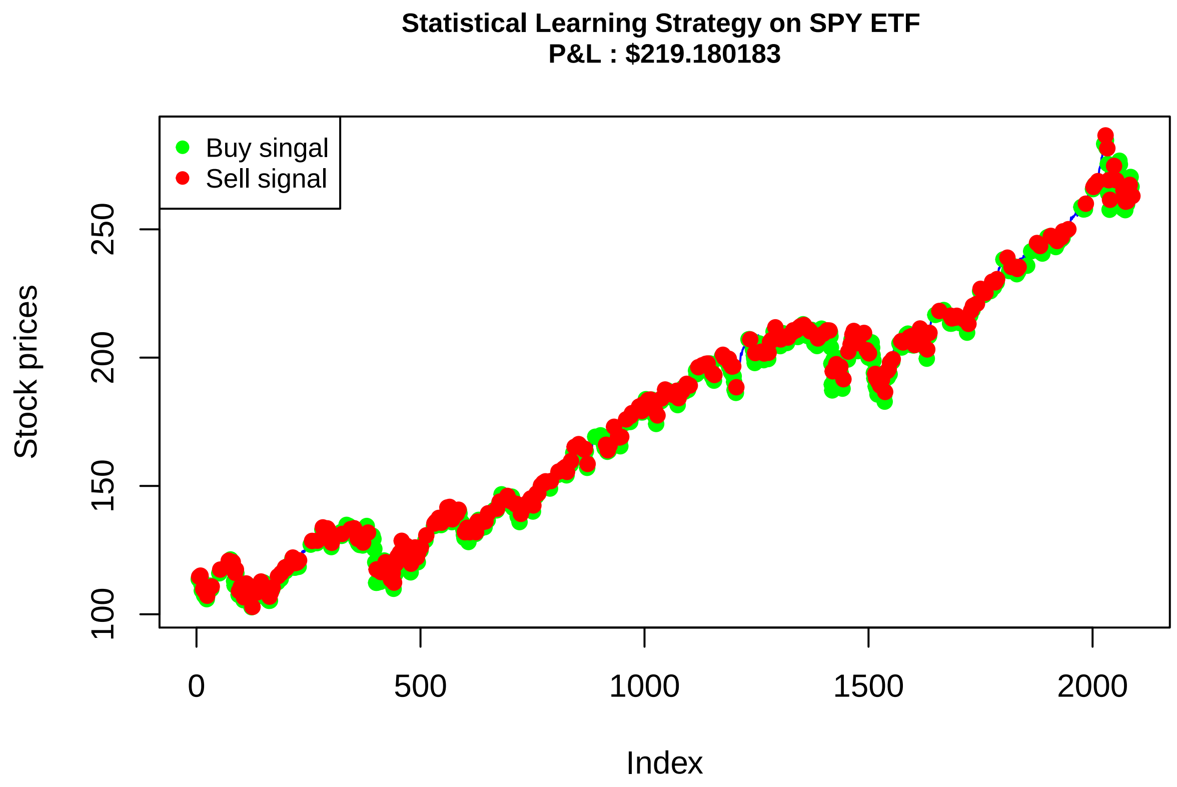
For the first case, we will buy when the return becomes lower than 1rd quartile of return minus one SD, and we will sell when the return becomes higher than 3rd quartile of return distribution plus SD. From the performance result, we gain $54.65 profit with this strategy.

```r
bounds <- summary(lasso.pred.backtest)

signal.buy <- as.numeric(substr(bounds[2], 9, 19))  - sd(lasso.pred.backtest)
signal.sell <- as.numeric(substr(bounds[5], 9, 19)) + sd(lasso.pred.backtest)

gain <- 0
quantity <- 0

for(i in 1:(nrow(sp500))){

    if(i == nrow(sp500) && sum(quantity) != 0){
        gain[i] <- sum(quantity)*sp500.price.backtest$SPY[i]
        quantity[i] <- -1*sum(quantity)
        break
    }

    if(glm.pred.bt[i] == -1 && (lasso.pred.backtest[i] <= signal.buy)){
        gain[i] <- -1*sp500.price.backtest$SPY[i]
        quantity[i] <- 1
    }else if(glm.pred.bt[i] == 1 && lasso.pred.backtest[i] >=
             signal.sell && sum(quantity) != 0){
        gain[i] <- sum(quantity)*sp500.price.backtest$SPY[i]
        quantity[i] <- -1*sum(quantity)
    }else{
        gain[i] <- 0
        quantity[i] <- 0
    }

}

signal.all <- sign(gain)
buy.index <- which(signal.all == -1)
sell.index <- which(signal.all == 1)

plot(1:nrow(sp500.price.backtest), sp500.price.backtest$SPY, type = "l", col="blue",
     main = paste("Statistical Learning Strategy on SPY ETF \n", "P&L : $",
                  round(sum(gain),6),sep = ""),
     xlab = "Index", ylab = "Stock prices", cex.main = 0.8)
points(buy.index, sp500.price.backtest$SPY[buy.index], col = "green", pch = 19)
points(sell.index, sp500.price.backtest$SPY[sell.index], col = "red", pch = 19)
legend("topleft", legend = c("Buy singal", "Sell signal"), col = c("green", "red"),
       pch = c(19,19) , cex = 0.8)
```



For the second case, we will buy when the return becomes lower than 1rd quartile of return, and we will sell when the return becomes higher than 3rd quartile of return distribution. From the performance result, we gain $219.18 profit. It seems like this strategy can perfrom very well with the upward trend of SPY ETF.

```r
bounds <- summary(lasso.pred.backtest)

signal.buy <- as.numeric(substr(bounds[2], 9, 19))
signal.sell <- as.numeric(substr(bounds[5], 9, 19))

gain <- 0
quantity <- 0

for(i in 1:(nrow(sp500))){

    if(i == nrow(sp500) && sum(quantity) != 0){
        gain[i] <- sum(quantity)*sp500.price.backtest$SPY[i]
        quantity[i] <- -1*sum(quantity)
        break
    }

    if(glm.pred.bt[i] == -1 && (lasso.pred.backtest[i] <= signal.buy)){
        gain[i] <- -1*sp500.price.backtest$SPY[i]
        quantity[i] <- 1

    }else if(glm.pred.bt[i] == 1 && lasso.pred.backtest[i] >=
             signal.sell && sum(quantity) != 0){
        gain[i] <- sum(quantity)*sp500.price.backtest$SPY[i]
        quantity[i] <- -1*sum(quantity)
    }else{
        gain[i] <- 0
        quantity[i] <- 0
    }

}

signal.all <- sign(gain)
buy.index <- which(signal.all == -1)
sell.index <- which(signal.all == 1)

plot(1:nrow(sp500.price.backtest), sp500.price.backtest$SPY, type = "l", col="blue",
     main = paste("Statistical Learning Strategy on SPY ETF \n", "P&L : $",
                  round(sum(gain),6),sep = ""),
     xlab = "Index", ylab = "Stock prices", cex.main = 0.8)
points(buy.index, sp500.price.backtest$SPY[buy.index], col = "green", pch = 19)
points(sell.index, sp500.price.backtest$SPY[sell.index], col = "red", pch = 19)
legend("topleft", legend = c("Buy singal", "Sell signal"), col = c("green", "red"),
       pch = c(19,19) , cex = 0.8)
```



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.

(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.

(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.

(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.

```
##part a
library(ISLR)
gas.med <-  median(Auto$mpg)
new.var <- ifelse(Auto$mpg > gas.med, 1, 0)
Auto$mpglevel <- as.factor(new.var)

##part b
library(e1071)
set.seed(personal)
tune.out <- tune(svm, mpglevel ~ ., data = Auto, kernel = "linear",
                 ranges = list(cost = c(0.01, 0.1, 0.8, 1, 2, 5, 10, 100)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.8
##
## - best performance: 0.01025641
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-02 0.07403846 0.04097358
## 2 1e-01 0.04596154 0.04806320
## 3 8e-01 0.01025641 0.01792836
## 4 1e+00 0.01025641 0.01792836
## 5 2e+00 0.01282051 0.01813094
## 6 5e+00 0.01538462 0.01792836
## 7 1e+01 0.01794872 0.01730637
## 8 1e+02 0.03057692 0.01058092
```

```
#With 10-fold cross-validation, the best model that function picks is cost 0.8.
#However, when we look at the table, cost 0.8 and codt 1 have the same result.


##part c
set.seed(personal)
tune.out <- tune(svm, mpglevel ~ ., data = Auto, kernel = "polynomial",
                 ranges = list(cost = c(0.8, 1, 2, 5, 10, 100), degree = c(2, 3, 4)))
summary(tune.out)
```

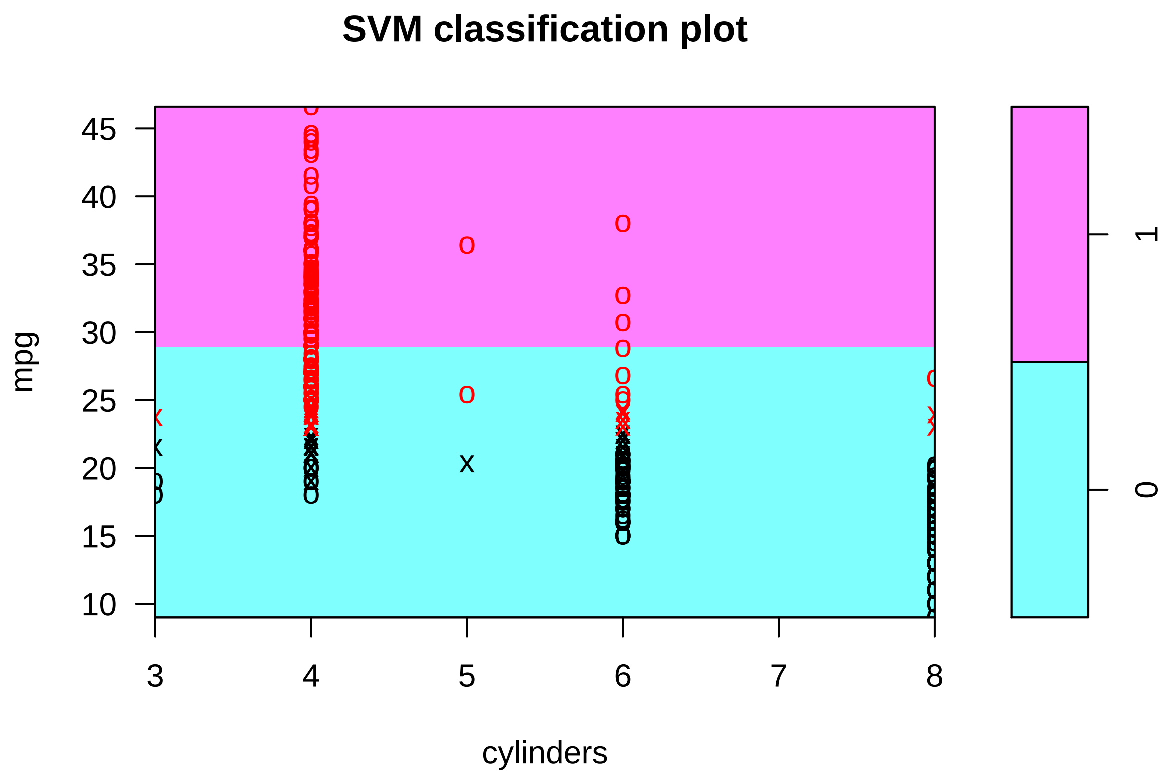
```
##
```

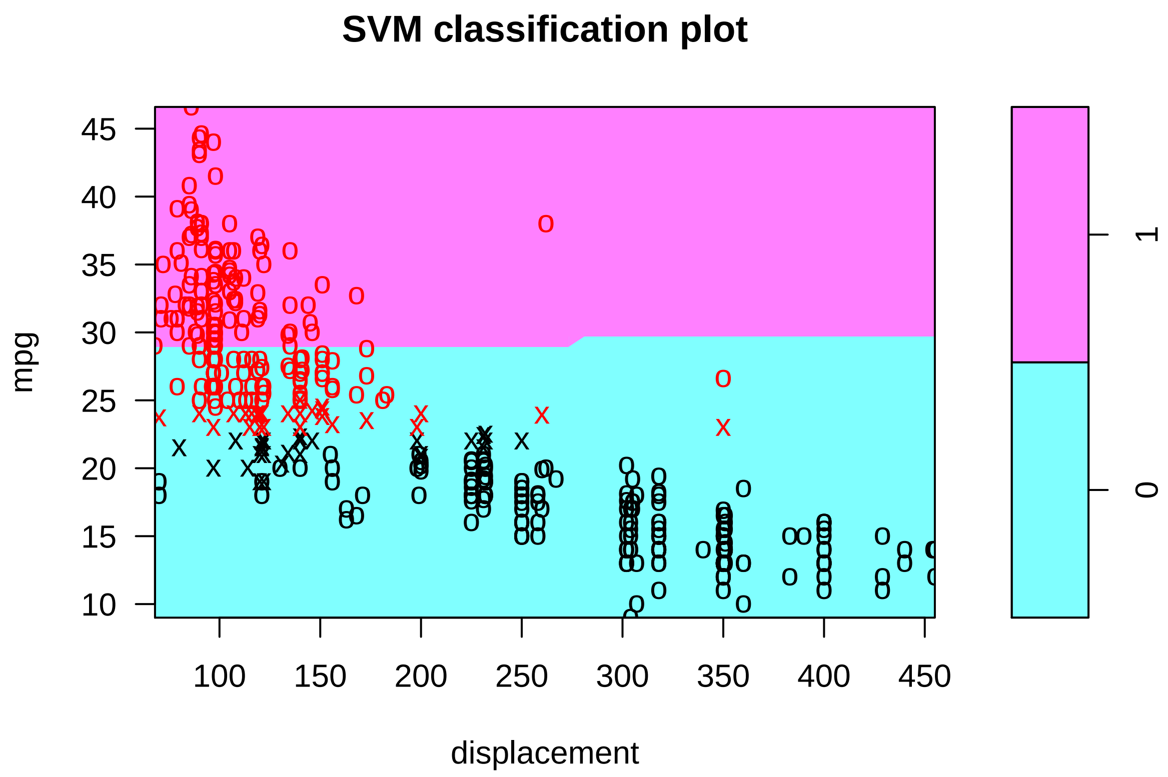
```
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##   100      2
##
## - best performance: 0.3010256
##
## - Detailed performance results:
##     cost degree     error dispersion
## 1    0.8      2 0.5587821 0.04374663
## 2    1.0      2 0.5587821 0.04374663
## 3    2.0      2 0.5587821 0.04374663
## 4    5.0      2 0.5587821 0.04374663
## 5   10.0      2 0.5234615 0.07732575
## 6  100.0      2 0.3010256 0.07578226
## 7    0.8      3 0.5587821 0.04374663
## 8    1.0      3 0.5587821 0.04374663
## 9    2.0      3 0.5587821 0.04374663
## 10   5.0      3 0.5587821 0.04374663
## 11  10.0      3 0.5587821 0.04374663
## 12 100.0      3 0.3315385 0.09280778
## 13   0.8      4 0.5587821 0.04374663
## 14   1.0      4 0.5587821 0.04374663
## 15   2.0      4 0.5587821 0.04374663
## 16   5.0      4 0.5587821 0.04374663
## 17  10.0      4 0.5587821 0.04374663
## 18 100.0      4 0.5587821 0.04374663
```

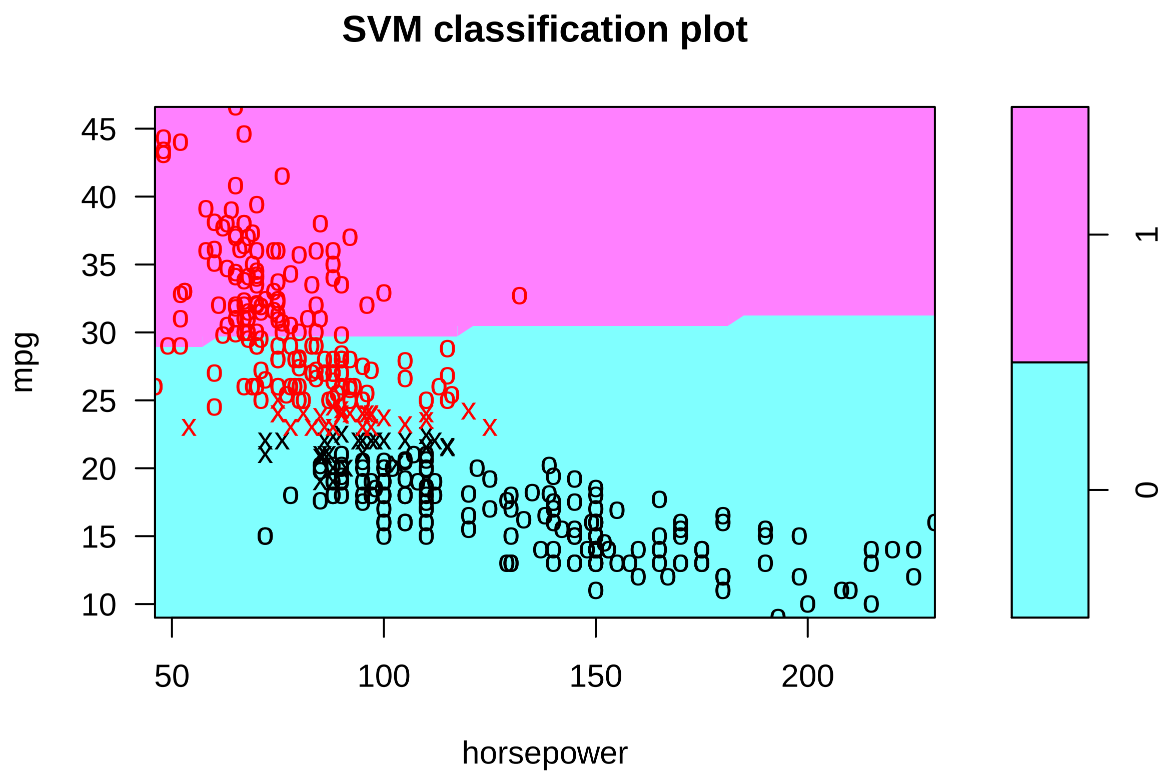
```r
#With 10-fold cross-validation on polynomial, the best model is cost 100 with degree 2.
set.seed(personal)
tune.out <- tune(svm, mpglevel ~ ., data = Auto, kernel = "radial",
                ranges = list(cost = c(0.8, 1, 2, 5, 10, 100),
                              gamma = c(0.01, 0.1, 1, 5, 10, 100)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##   100  0.01
##
## - best performance: 0.01538462
##
## - Detailed performance results:
##     cost gamma      error dispersion
## 1    0.8 1e-02 0.07403846 0.04097358
## 2    1.0 1e-02 0.07403846 0.04097358
## 3    2.0 1e-02 0.06121795 0.04387918
## 4    5.0 1e-02 0.04596154 0.04651847
## 5   10.0 1e-02 0.02294872 0.03502423
## 6  100.0 1e-02 0.01538462 0.02162241
## 7    0.8 1e-01 0.05871795 0.04363602
## 8    1.0 1e-01 0.05621795 0.04331450
## 9    2.0 1e-01 0.03583333 0.03463444
## 10   5.0 1e-01 0.03326923 0.02113646
## 11  10.0 1e-01 0.02814103 0.02550614
## 12 100.0 1e-01 0.03070513 0.02649650
## 13   0.8 1e+00 0.06391026 0.03688860
## 14   1.0 1e+00 0.05884615 0.03649382
## 15   2.0 1e+00 0.06141026 0.03476122
## 16   5.0 1e+00 0.06141026 0.03476122
## 17  10.0 1e+00 0.06141026 0.03476122
## 18 100.0 1e+00 0.06141026 0.03476122
## 19   0.8 5e+00 0.53064103 0.04317910
## 20   1.0 5e+00 0.50788462 0.05784725
## 21   2.0 5e+00 0.49762821 0.06657212
## 22   5.0 5e+00 0.49762821 0.06657212
## 23  10.0 5e+00 0.49762821 0.06657212
## 24 100.0 5e+00 0.49762821 0.06657212
## 25   0.8 1e+01 0.54596154 0.04572655
## 26   1.0 1e+01 0.52551282 0.04349936
## 27   2.0 1e+01 0.52551282 0.04349936
## 28   5.0 1e+01 0.52551282 0.04349936
## 29  10.0 1e+01 0.52551282 0.04349936
## 30 100.0 1e+01 0.52551282 0.04349936
## 31   0.8 1e+02 0.55878205 0.04374663
## 32   1.0 1e+02 0.55878205 0.04374663
## 33   2.0 1e+02 0.55878205 0.04374663
## 34   5.0 1e+02 0.55878205 0.04374663
## 35  10.0 1e+02 0.55878205 0.04374663
## 36 100.0 1e+02 0.55878205 0.04374663
```

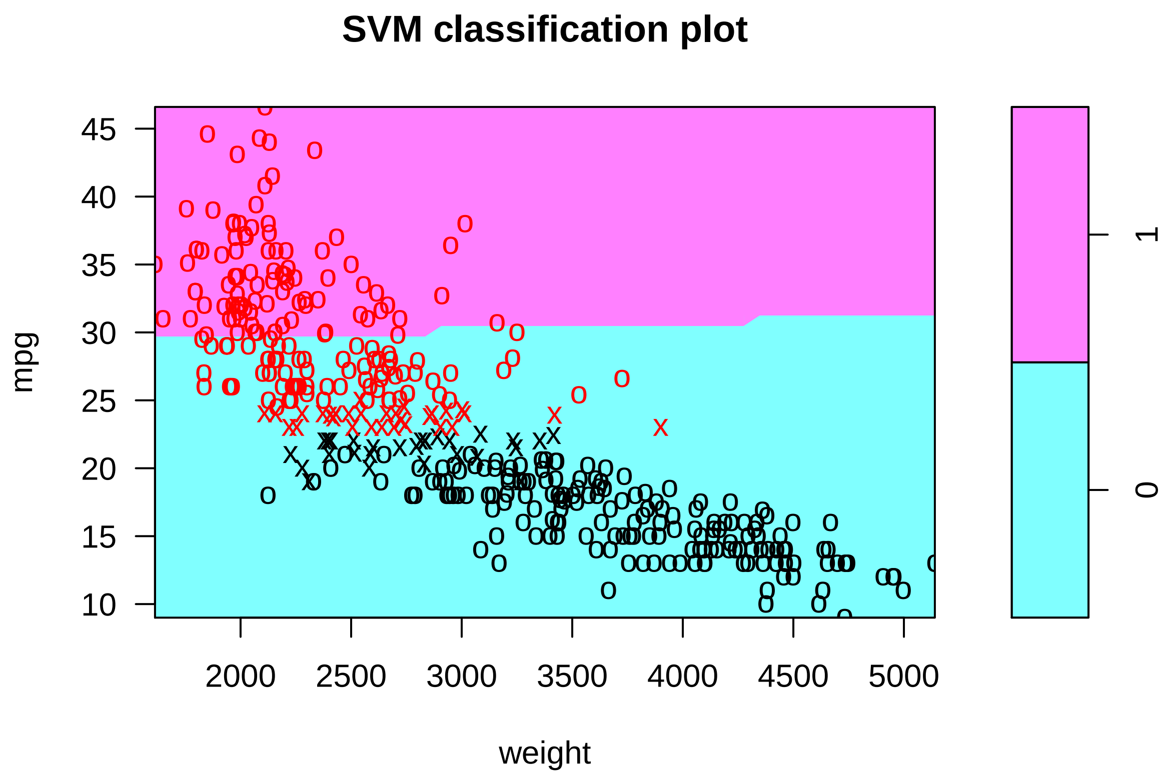
```r
#With 10-fold cross-validation on radial, the best model is cost 100 with gamma 0.01.

#part d
svm.linear <- svm(mpglevel ~ ., data = Auto, kernel = "linear", cost = 0.8)
svm.poly <- svm(mpglevel ~ ., data = Auto, kernel = "polynomial", cost = 100,
    degree = 2)
svm.radial = svm(mpglevel ~ ., data = Auto, kernel = "radial", cost = 100,
                 gamma = 0.01)
plotpairs = function(fit) {
    for (name in names(Auto)[!(names(Auto) %in% c("mpg", "mpglevel", "name"))]) {
        plot(fit, Auto, as.formula(paste("mpg~", name, sep = "")))
    }
}
plotpairs(svm.linear)
```

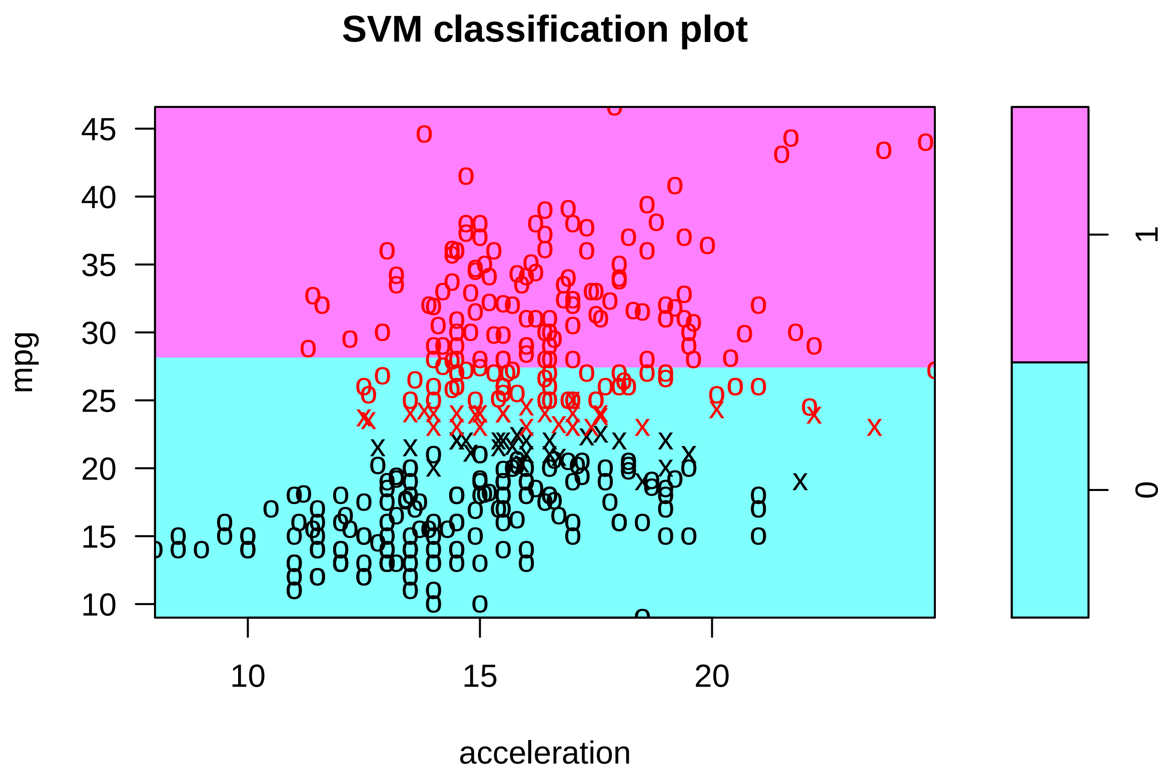

SVM classification plot
mpg
cylinders
45
40
35
30
25
20
15
10
3
4
5
6
7
8
1
0

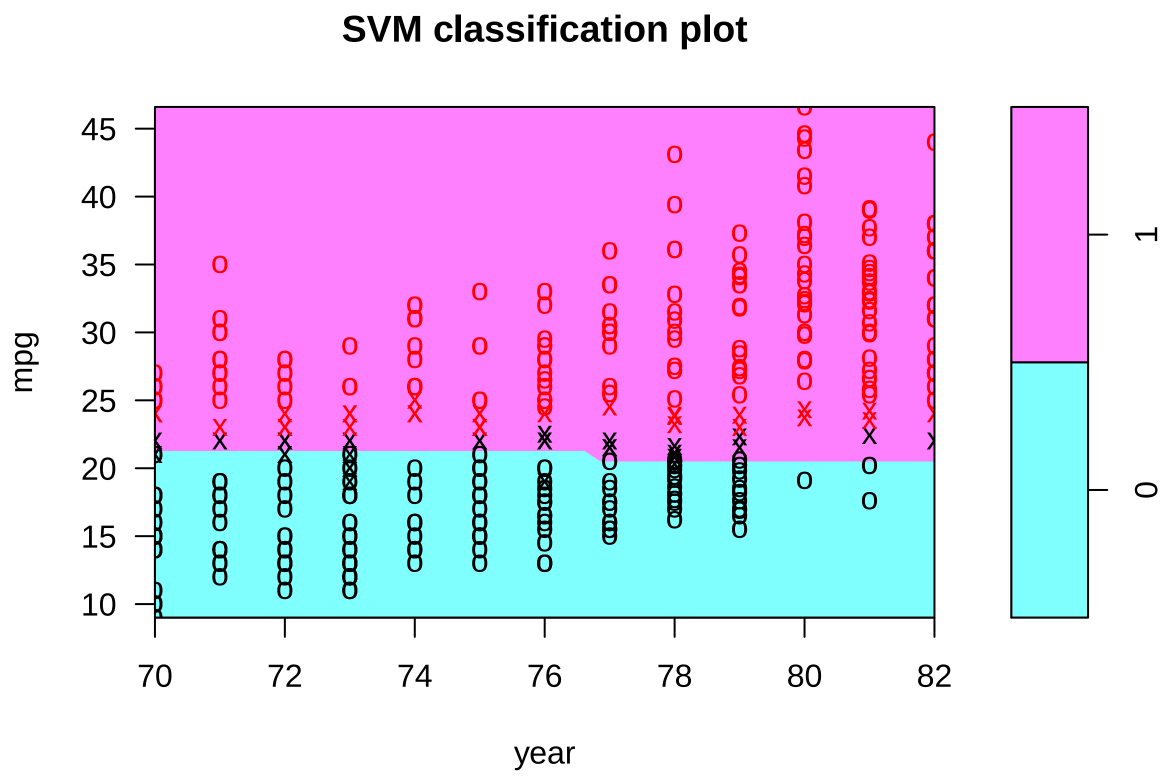

SVM classification plot
mpg
displacement
45
40
35
30
25
20
15
10
100
150
200
250
300
350
400
450
1
0

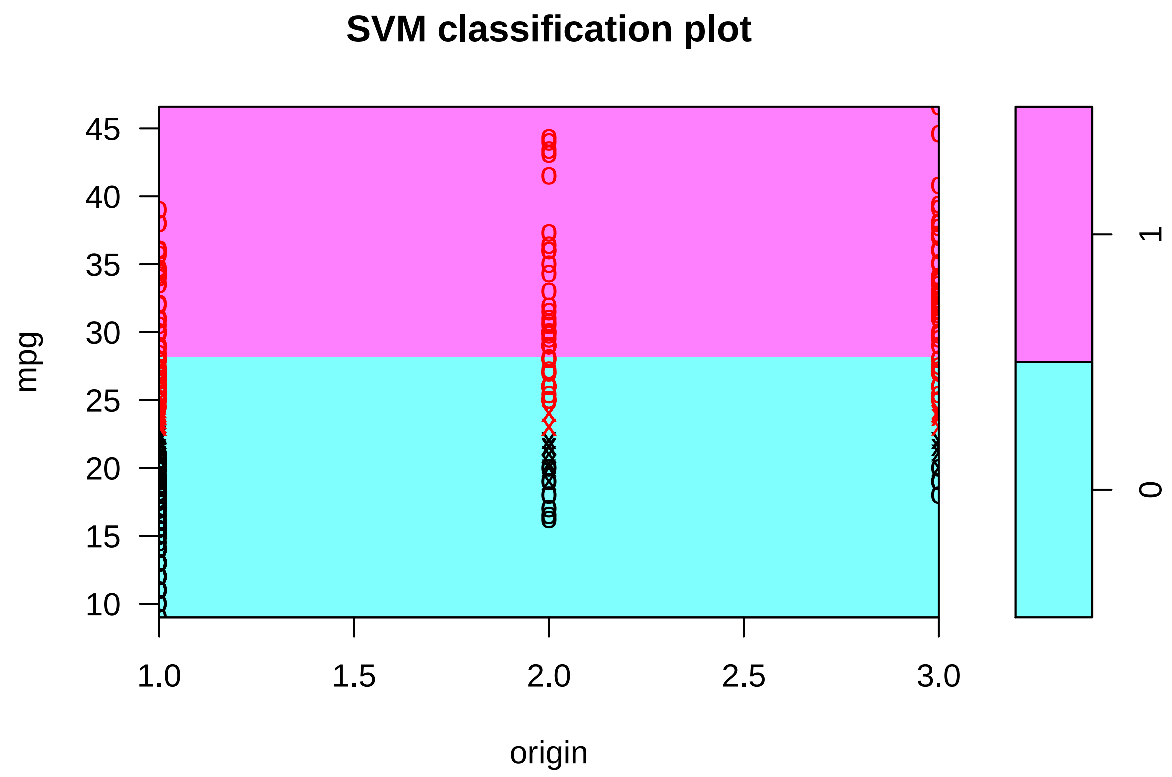


SVM classification plot
mpg
horsepower
45
40
35
30
25
20
15
10
50
100
150
200
1
0

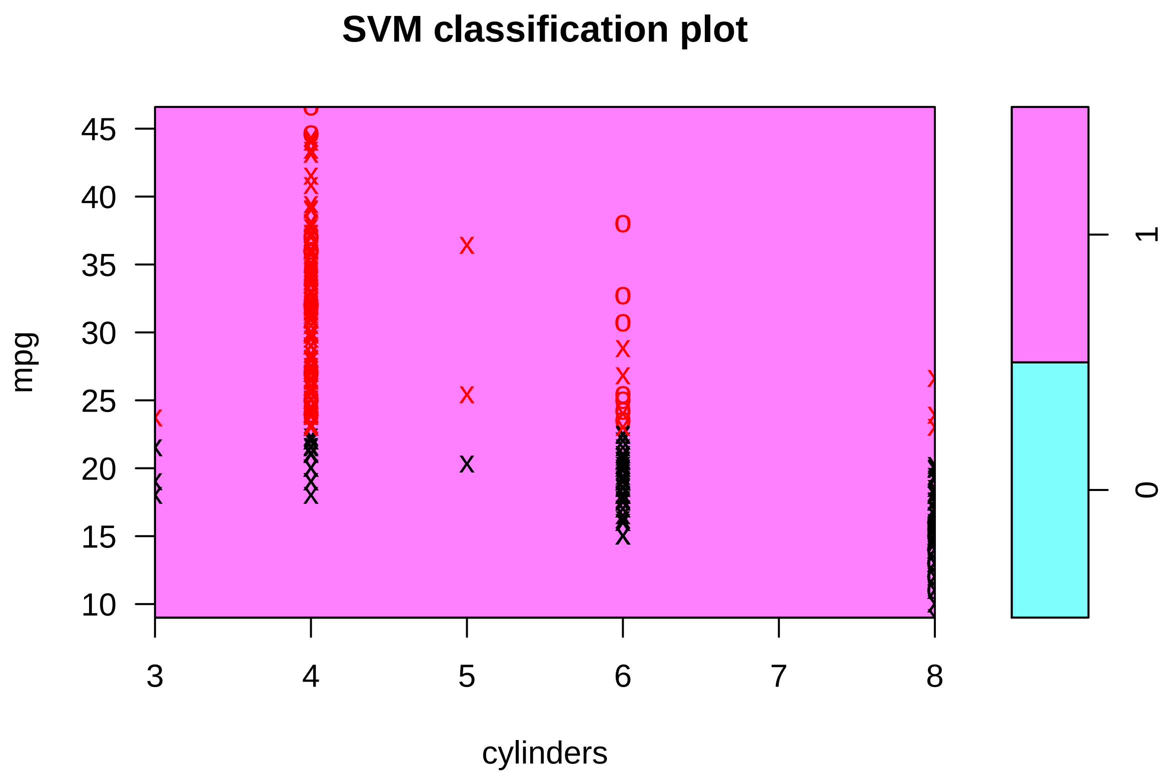

SVM classification plot
mpg
weight
45
40
35
30
25
20
15
10
2000
2500
3000
3500
4000
4500
5000
1
0

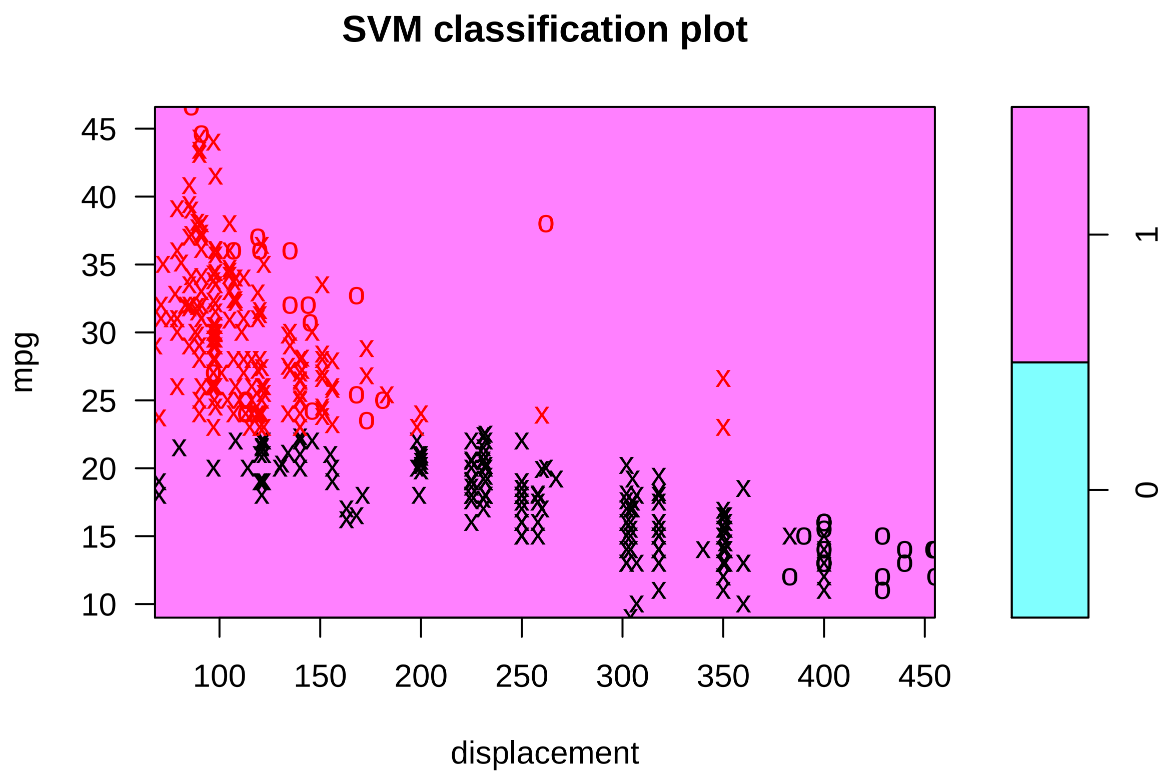

SVM classification plot
mpg
acceleration
45
40
35
30
25
20
15
10
10
15
20
1
0


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

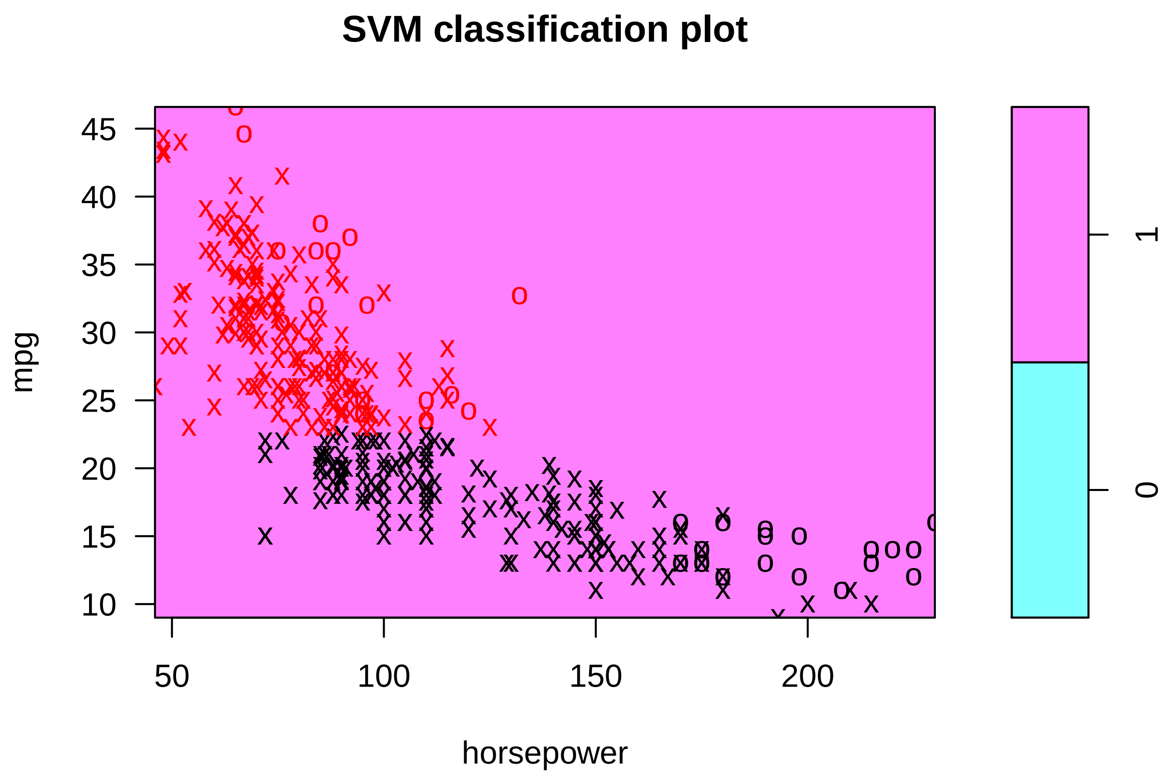


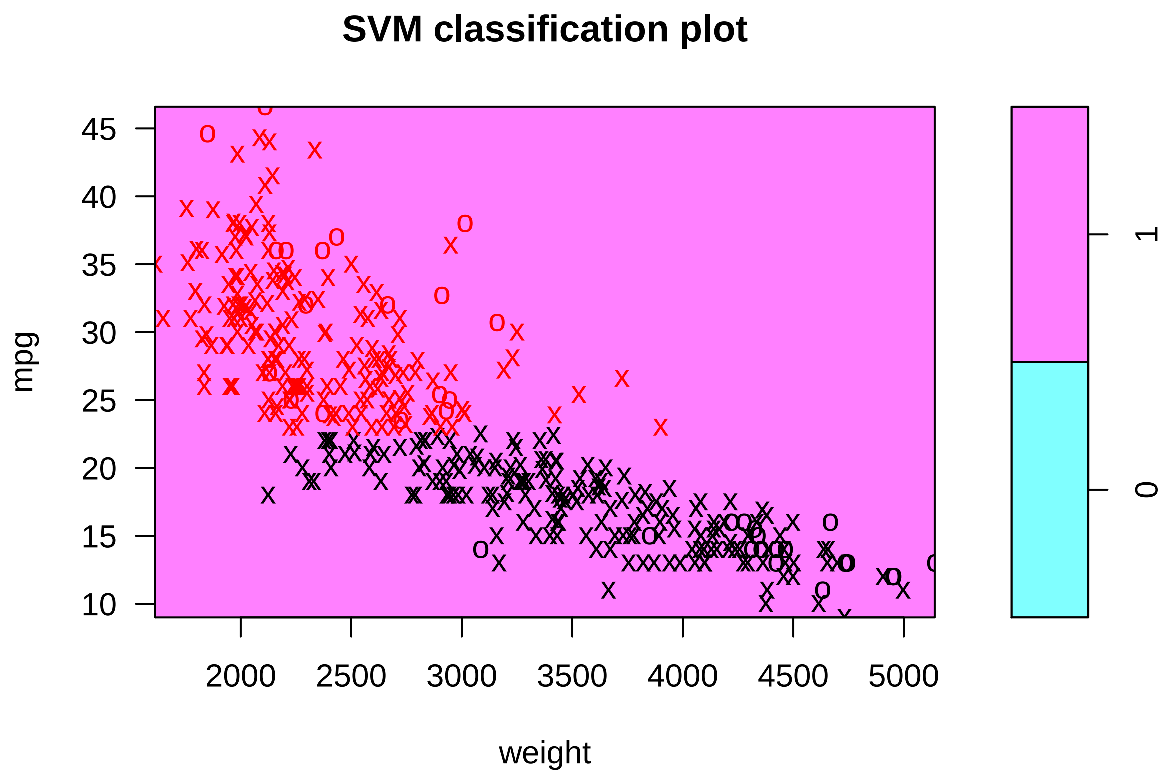
```r
plotpairs(svm.poly)
```

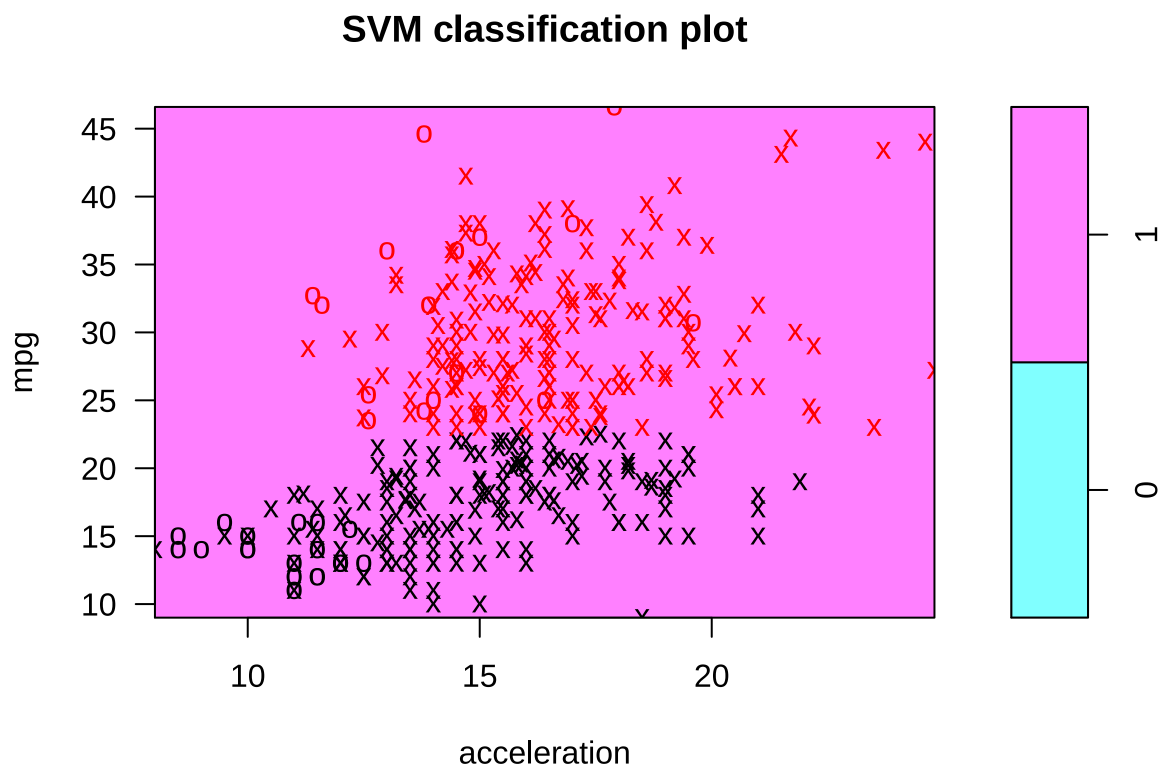

SVM classification plot
mpg
cylinders
45
40
35
30
25
20
15
10
3
4
5
6
7
8
1
0

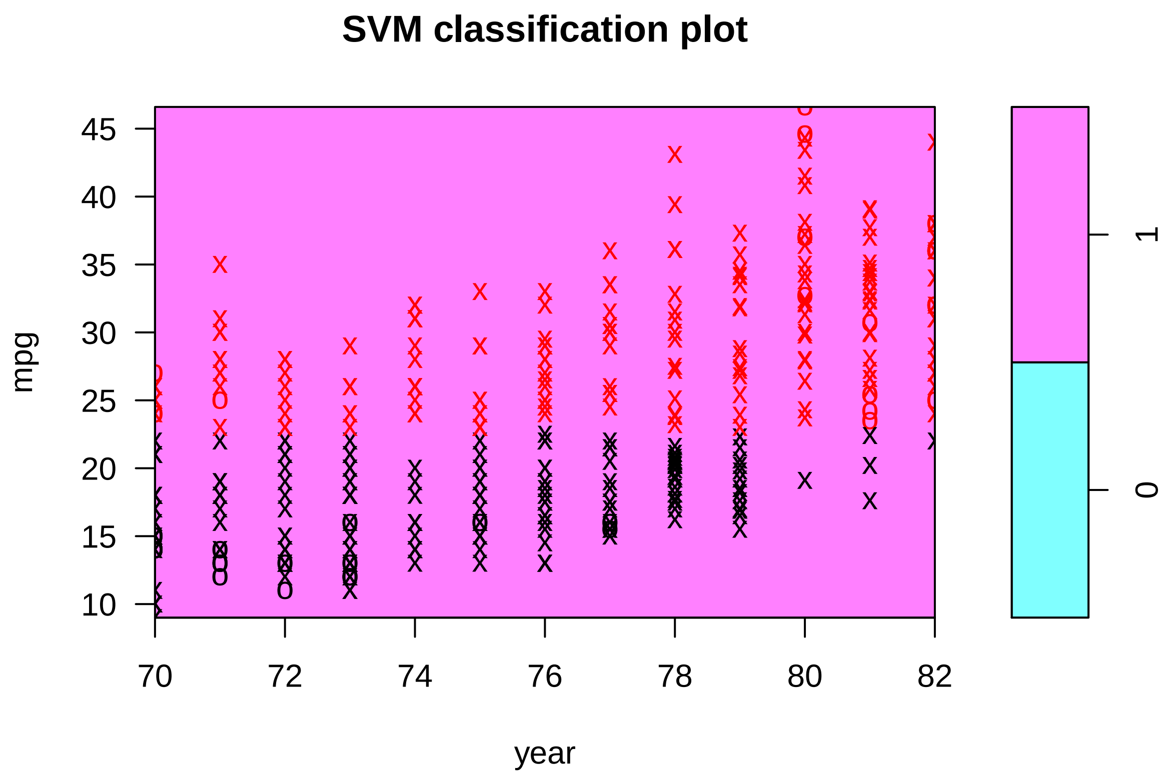


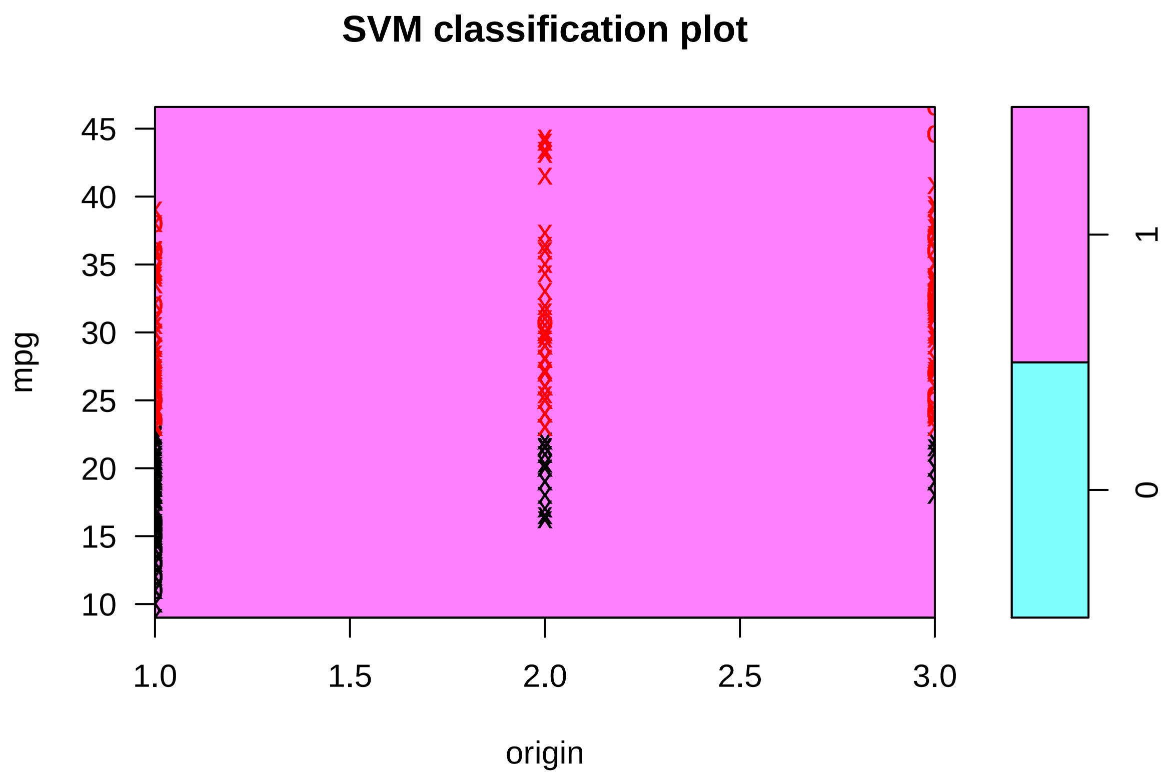
SVM classification plot
mpg
45
40
35
30
25
20
15
10
100
150
200
250
300
350
400
450
displacement
1
0


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

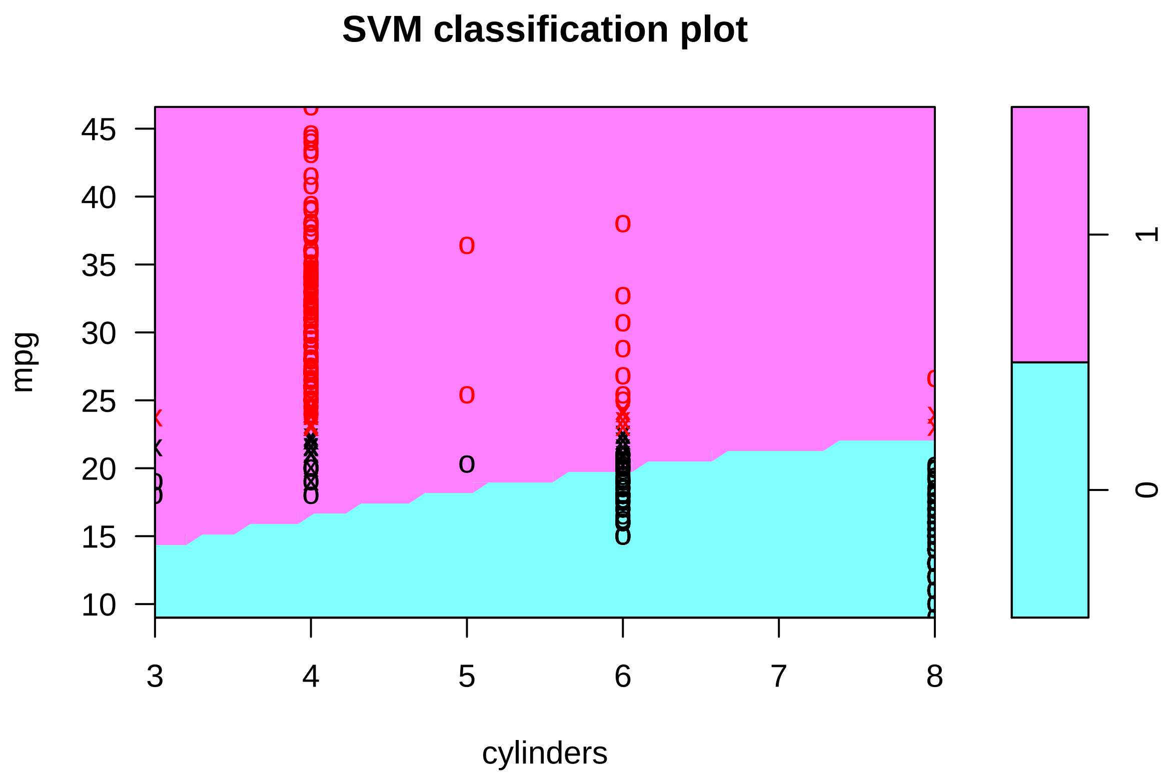

SVM classification plot
mpg
weight
45
40
35
30
25
20
15
10
2000
2500
3000
3500
4000
4500
5000
1
0

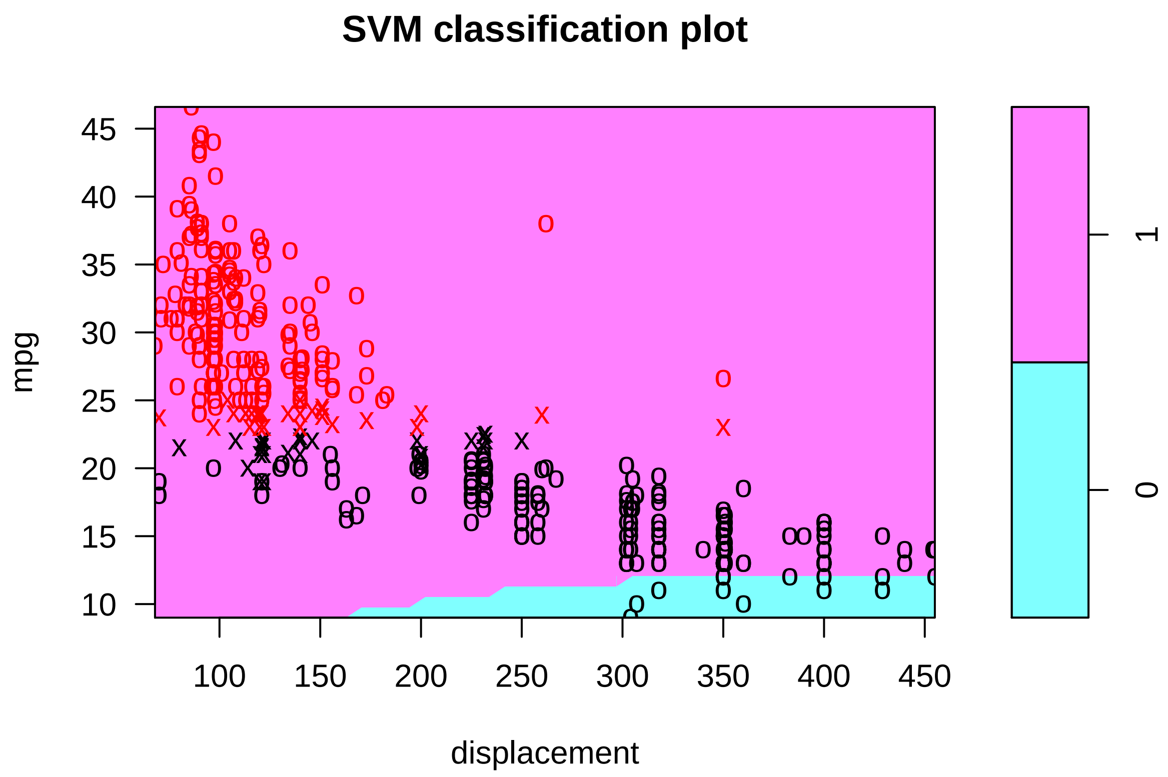

SVM classification plot
mpg
acceleration
45
40
35
30
25
20
15
10
10
15
20
1
0

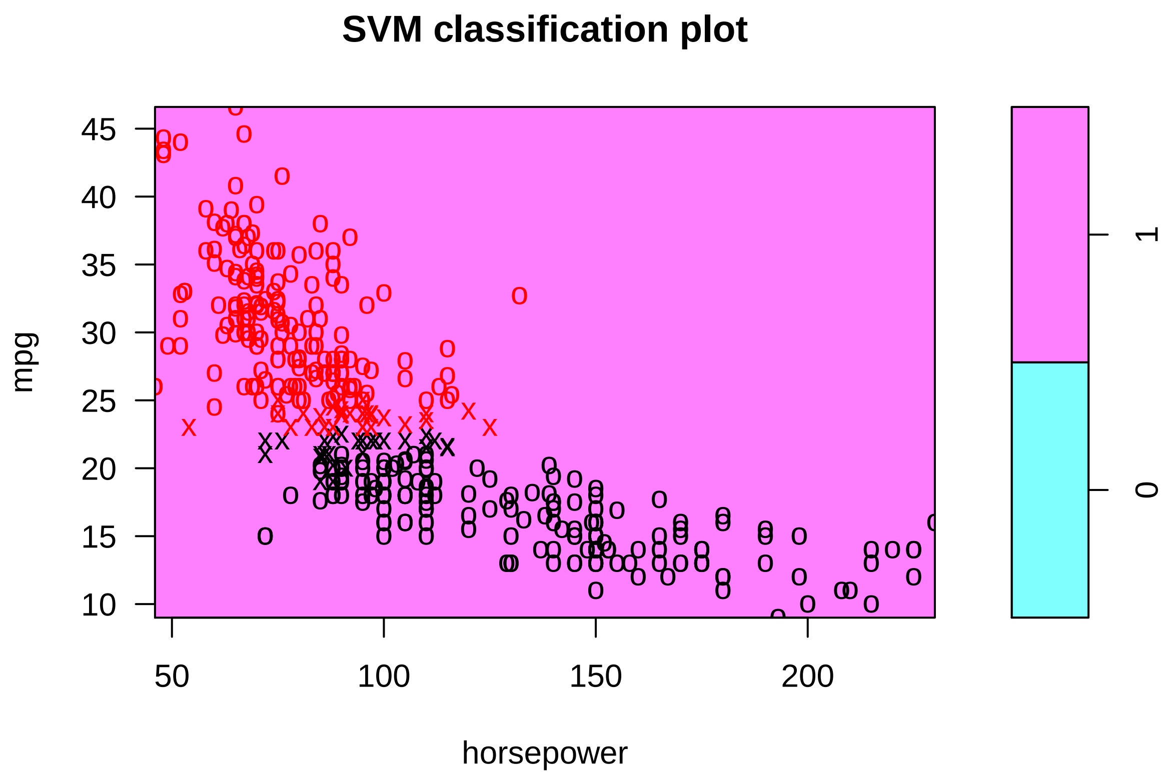

SVM classification plot
mpg
year
45
40
35
30
25
20
15
10
70
72
74
76
78
80
82
1
0

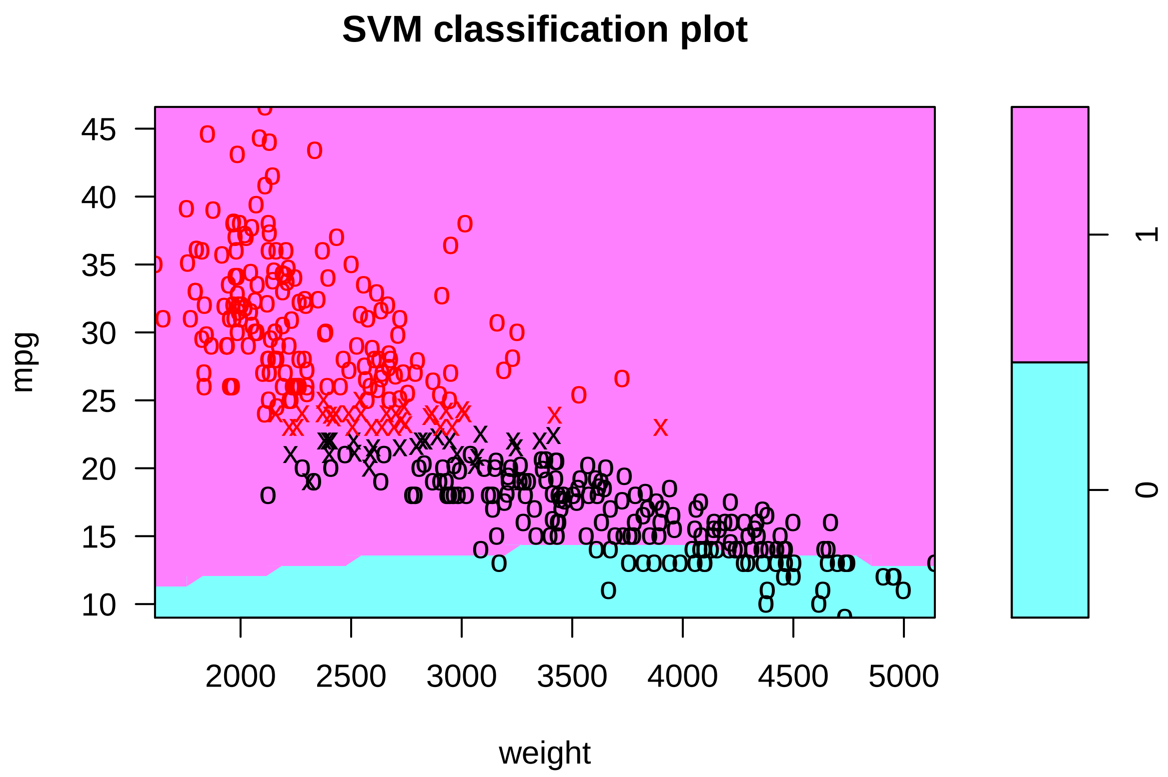


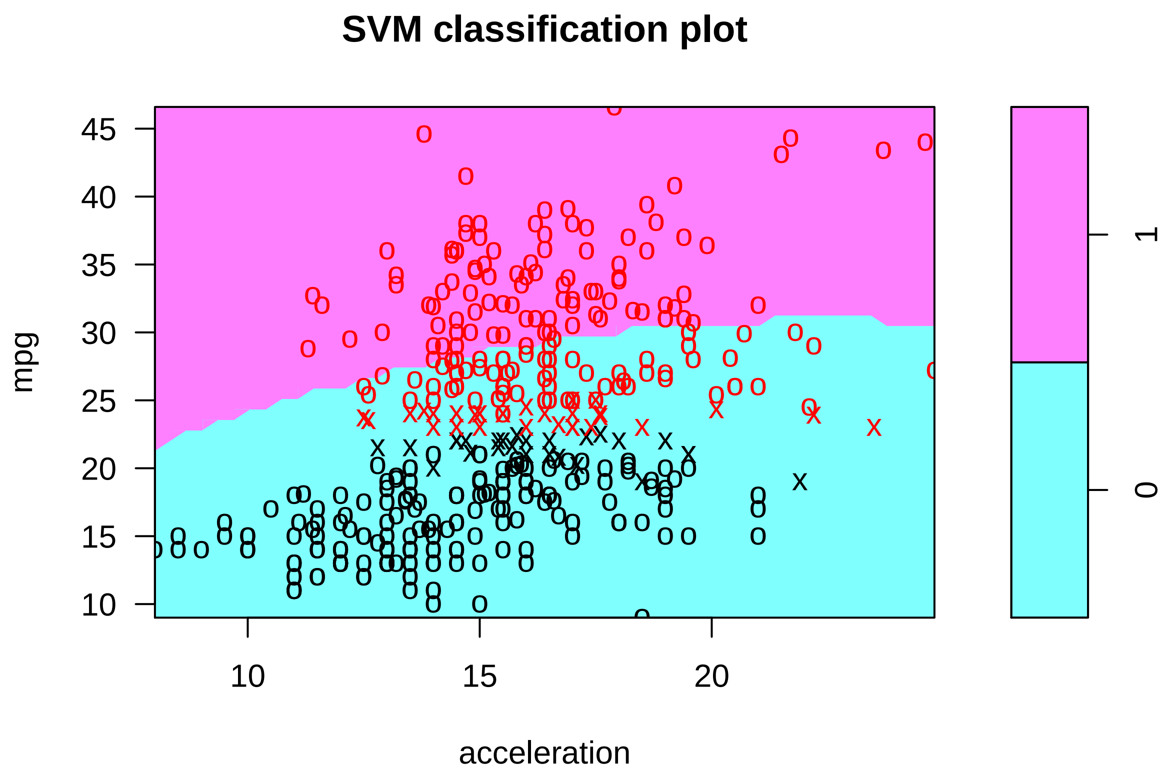
```r
plotpairs(svm.radial)
```

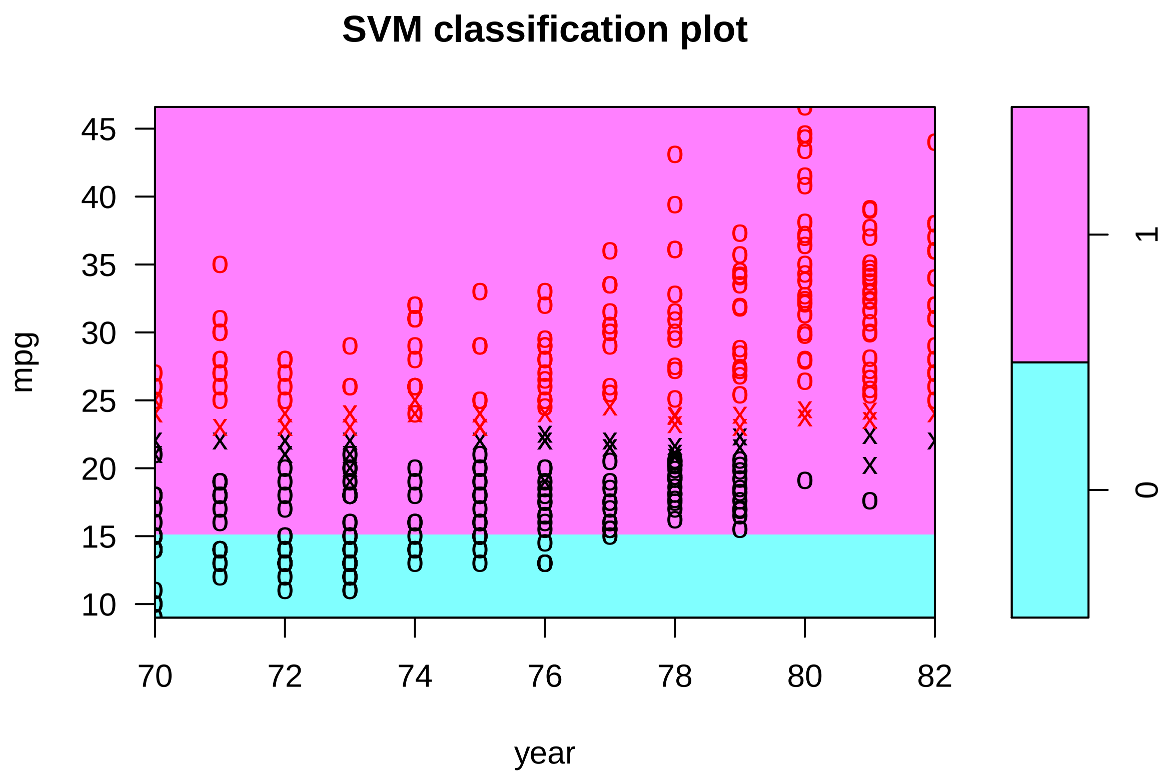

SVM classification plot
mpg
cylinders
45
40
35
30
25
20
15
10
3
4
5
6
7
8
1
0

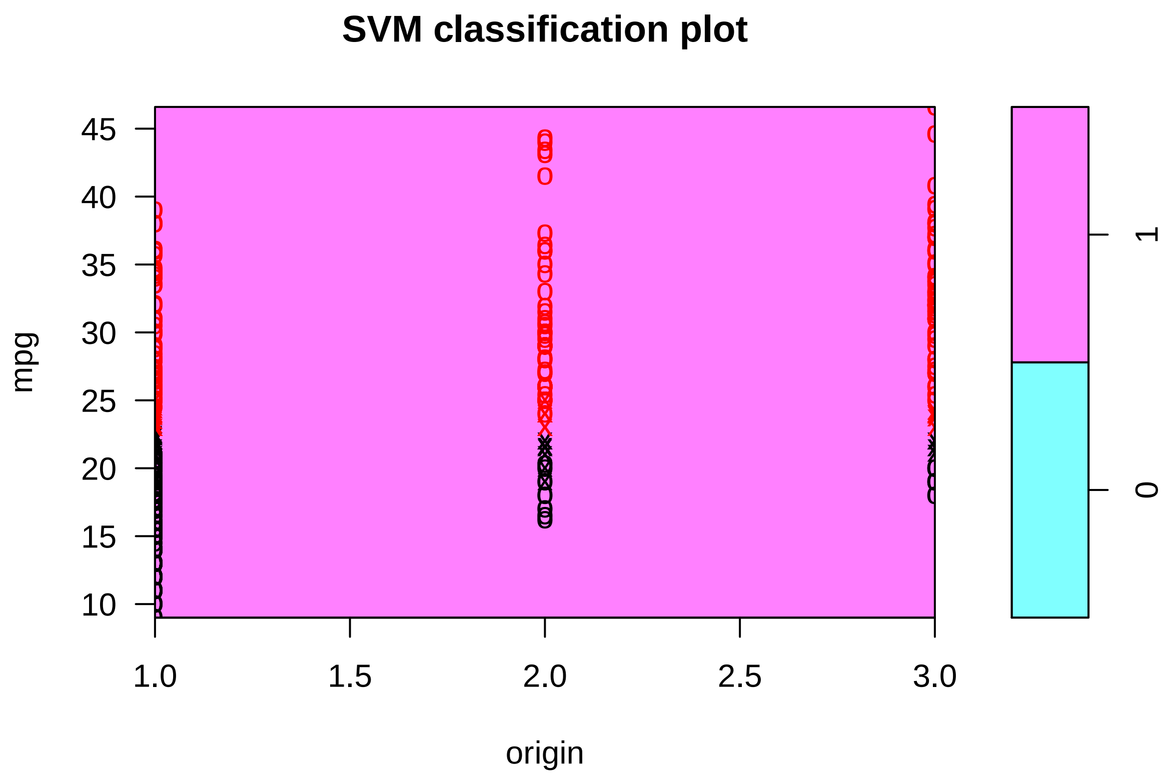

SVM classification plot
mpg
45
40
35
30
25
20
15
10
100
150
200
250
300
350
400
450
displacement
1
0


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


SVM classification plot
mpg
45
40
35
30
25
20
15
10
2000
2500
3000
3500
4000
4500
5000
weight
1
0


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


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


SVM classification plot
mpg
origin
1
0