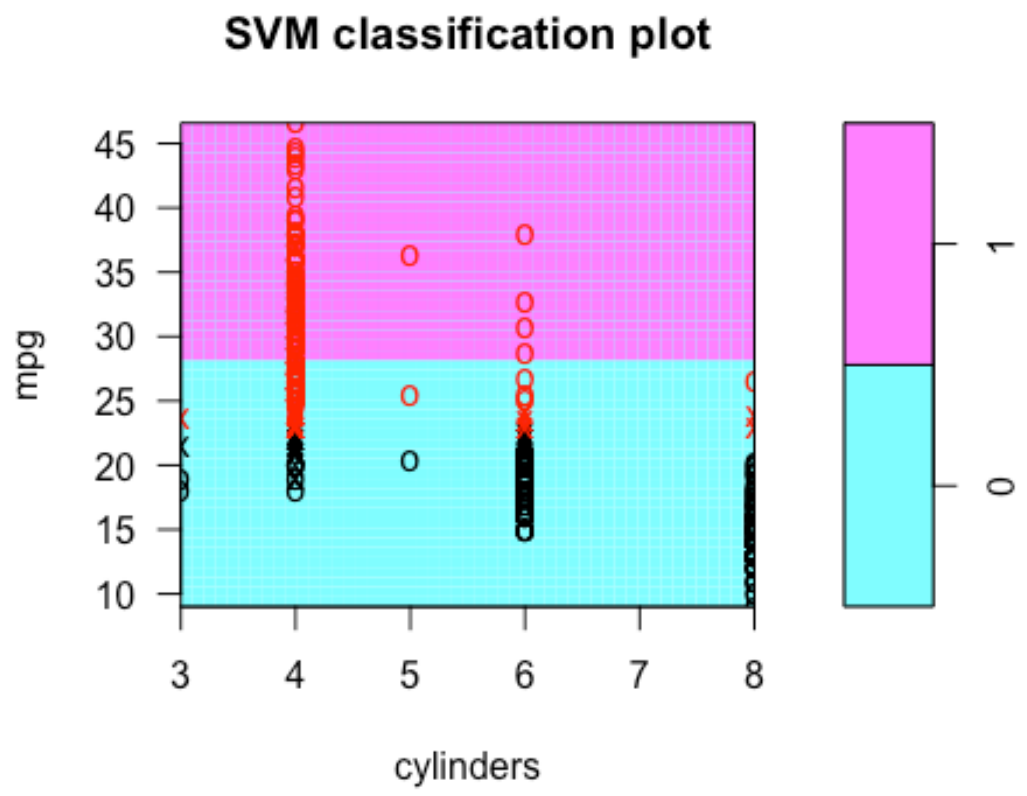
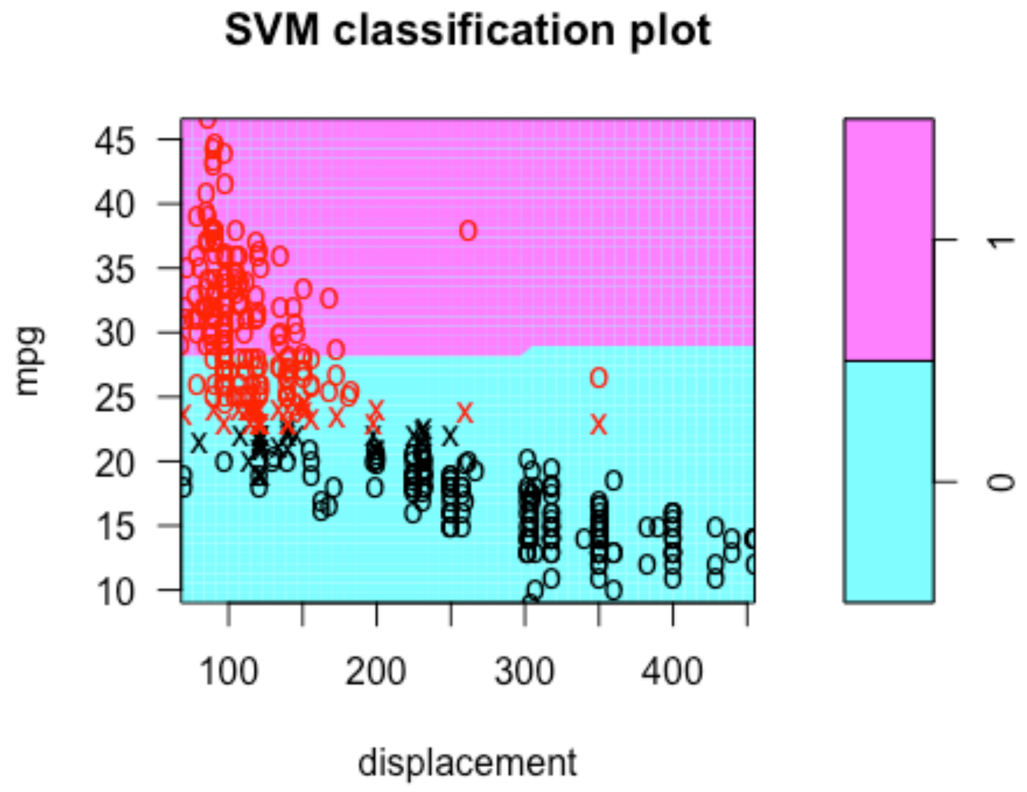
R Notebook

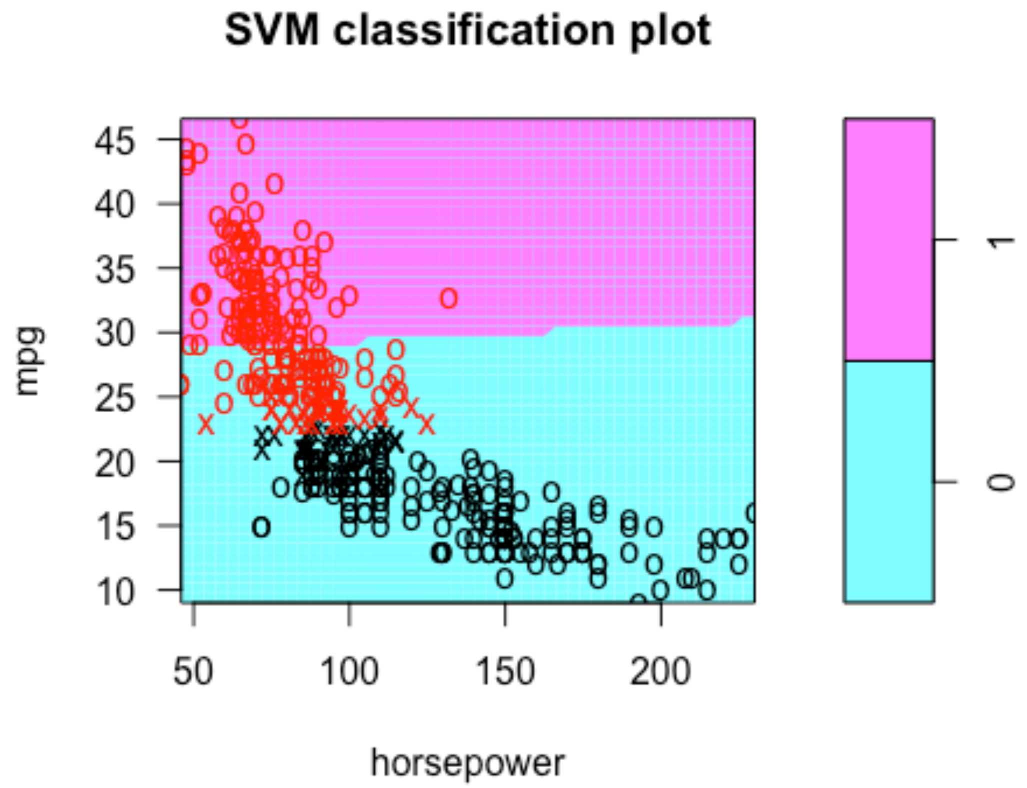
```
#Ex7
library(ISLR)
require(tidyverse)

## Loading required package: tidyverse

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'tidyverse'

library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(e1071)

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

attach(Auto)

## The following object is masked from package:ggplot2:
##
##     mpg

library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

#a
gas_median = median(Auto$mpg)
variation = ifelse(Auto$mpg > gas_median, 1, 0)
Auto$mpglevel = as.factor(variation)
#b
set.seed(1)
library(e1071)

tune_x = tune(svm, mpglevel ~ ., data = Auto, kernel = "linear", ranges =
list(cost = c(0.01, 0.1, 1, 5, 10, 100)))
summary(tune_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 0.07403846 0.05471525
## 2 1e-01 0.03826923 0.05148114
## 3 1e+00 0.01275641 0.01344780
## 4 5e+00 0.01782051 0.01229997
## 5 1e+01 0.02038462 0.01074682
## 6 1e+02 0.03820513 0.01773427

#the best result is achieved when cost = 1. we get an error
# of 0.0127


#c
tune_x = tune(svm, mpglevel ~ ., data = Auto, kernel = "polynomial", ranges =
list(cost = c(0.1,
    1, 5, 10), degree = c(2, 3, 4)))
summary(tune_x)

##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##    10      2
##
## - best performance: 0.5228205
##
## - Detailed performance results:
##    cost degree     error dispersion
## 1   0.1      2 0.5587179 0.05068311
## 2   1.0      2 0.5587179 0.05068311
## 3   5.0      2 0.5587179 0.05068311
## 4  10.0      2 0.5228205 0.09271988
## 5   0.1      3 0.5587179 0.05068311
## 6   1.0      3 0.5587179 0.05068311
## 7   5.0      3 0.5587179 0.05068311
## 8  10.0      3 0.5587179 0.05068311
## 9   0.1      4 0.5587179 0.05068311
## 10  1.0      4 0.5587179 0.05068311
## 11  5.0      4 0.5587179 0.05068311
## 12 10.0      4 0.5587179 0.05068311
```

```
##From the summary we can see that the smallest cross-validation error is
obtained for cost=10 and degree=2.

tune_x = tune(svm, mpglevel ~ ., data = Auto, kernel = "radial", ranges =
list(cost = c(0.1,
    1, 5, 10), gamma = c(0.01, 0.1, 1, 5, 10, 100)))
summary(tune_x)
```

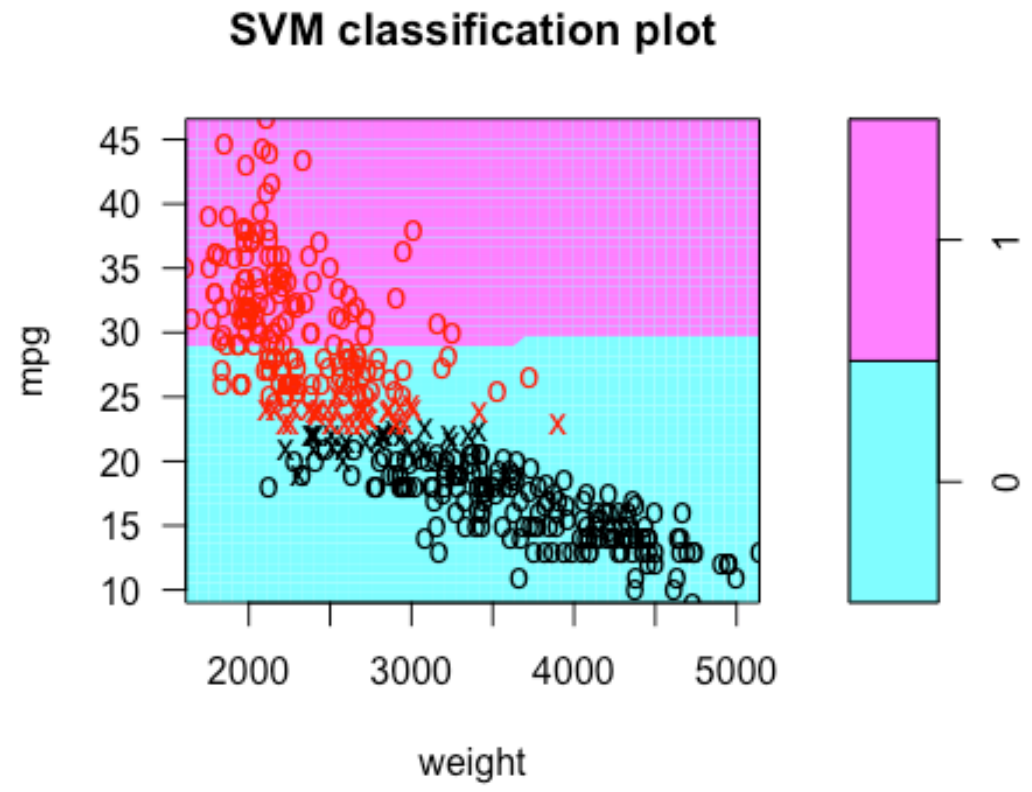
```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##    10  0.01
##
## - best performance: 0.02038462
##
## - Detailed performance results:
##    cost gamma      error dispersion
## 1   0.1 1e-02 0.08673077 0.04966103
## 2   1.0 1e-02 0.07147436 0.04638992
## 3   5.0 1e-02 0.04846154 0.03501693
## 4  10.0 1e-02 0.02038462 0.01617396
## 5   0.1 1e-01 0.07660256 0.04674289
## 6   1.0 1e-01 0.05608974 0.03555911
## 7   5.0 1e-01 0.02544872 0.02662645
## 8  10.0 1e-01 0.02544872 0.02662645
## 9   0.1 1e+00 0.58673077 0.03560377
## 10  1.0 1e+00 0.06121795 0.03216114
## 11  5.0 1e+00 0.06115385 0.02720346
## 12 10.0 1e+00 0.06115385 0.02720346
## 13  0.1 5e+00 0.58673077 0.03560377
## 14  1.0 5e+00 0.52032051 0.02994512
## 15  5.0 5e+00 0.51525641 0.03415169
## 16 10.0 5e+00 0.51525641 0.03415169
## 17  0.1 1e+01 0.58673077 0.03560377
## 18  1.0 1e+01 0.55602564 0.03521326
## 19  5.0 1e+01 0.54083333 0.02859746
## 20 10.0 1e+01 0.54083333 0.02859746
## 21  0.1 1e+02 0.58673077 0.03560377
## 22  1.0 1e+02 0.58673077 0.03560377
## 23  5.0 1e+02 0.58673077 0.03560377
## 24 10.0 1e+02 0.58673077 0.03560377

#with radial the best results are achieved for cost 10
# and gamma 0.01



#d
svm_linear_model = svm(mpglevel ~ ., data = Auto, kernel = "linear", cost =
1)
svm_polynomial = svm(mpglevel ~ ., data = Auto, kernel = "polynomial", cost =
10,
    degree = 2)
svm_radial = svm(mpglevel ~ ., data = Auto, kernel = "radial", cost = 10,
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_model)
```

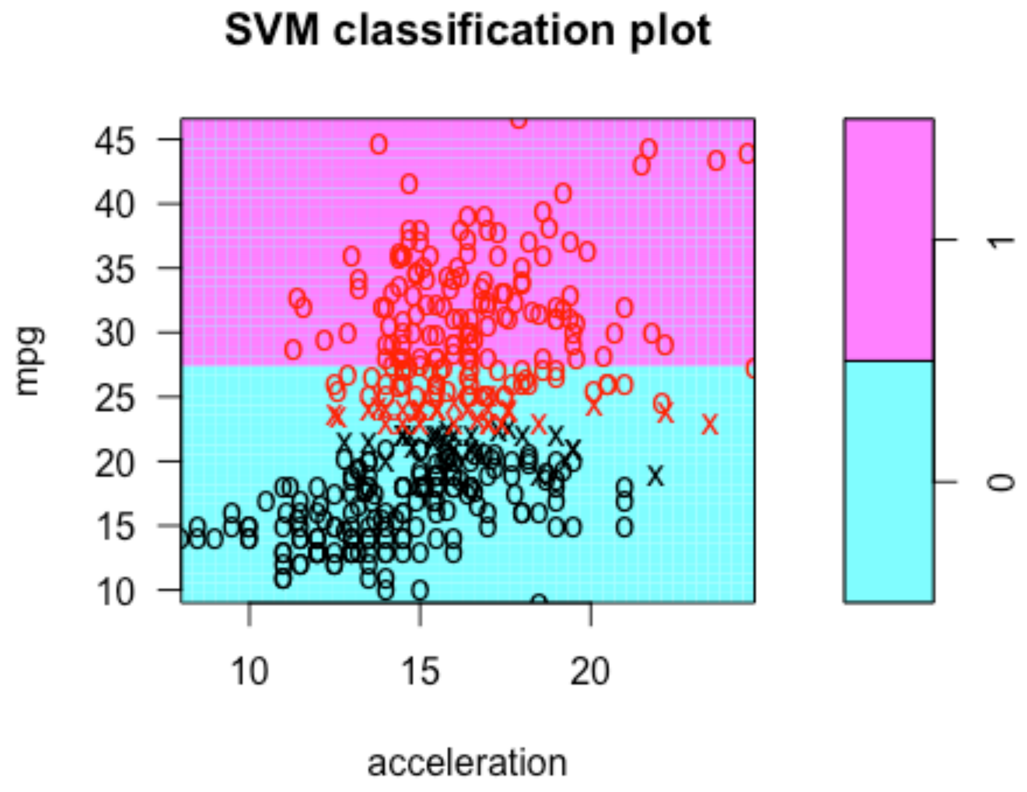

SVM classification plot
mpg
cylinders
45
40
35
30
25
20
15
10
3
4
5
6
7
8
1
0

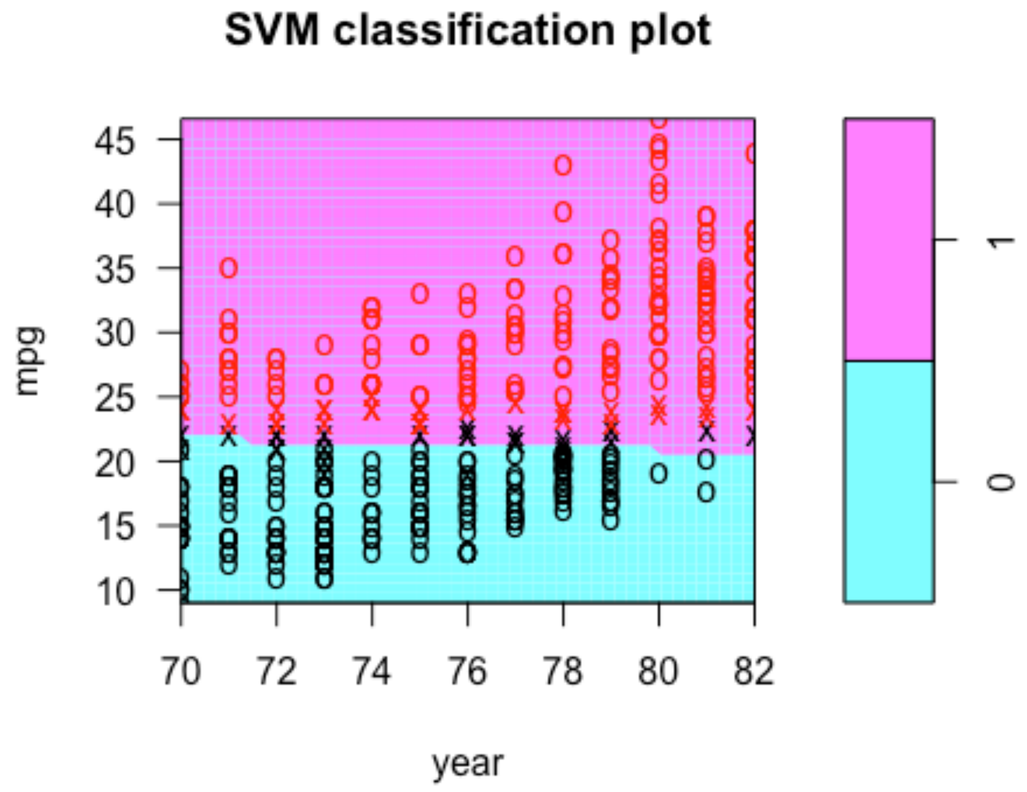

SVM classification plot
mpg
displacement
45
40
35
30
25
20
15
10
100
200
300
400
1
0

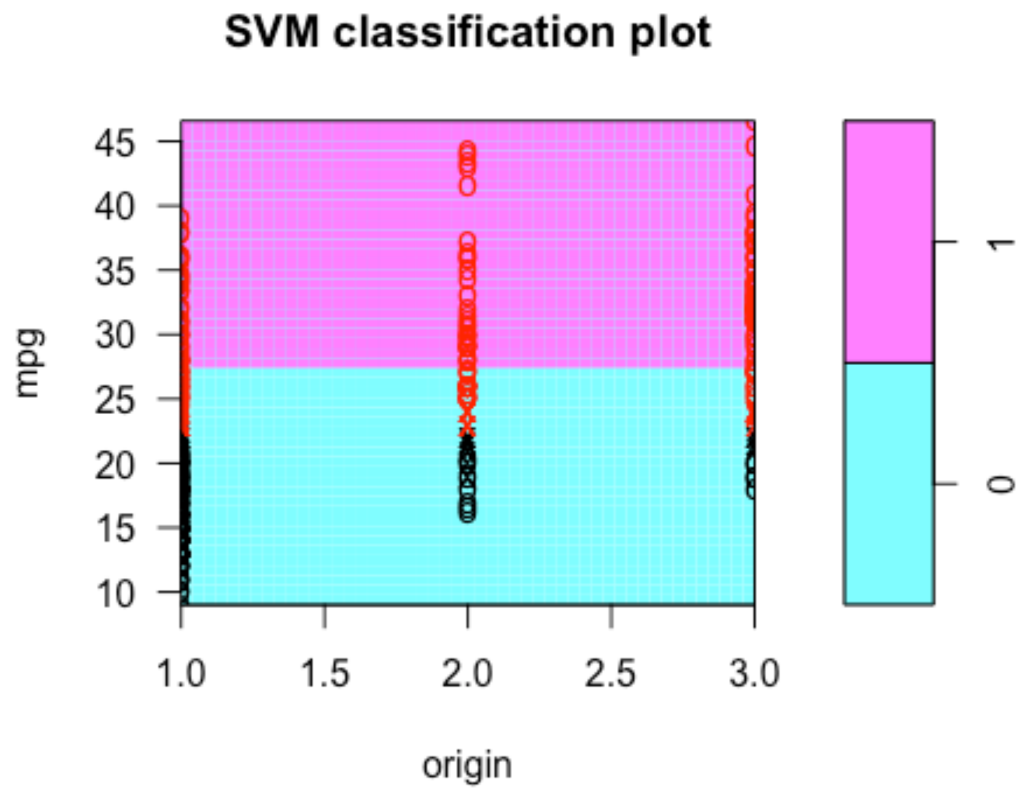


SVM classification plot
mpg
horsepower
45
40
35
30
25
20
15
10
50
100
150
200
1
0

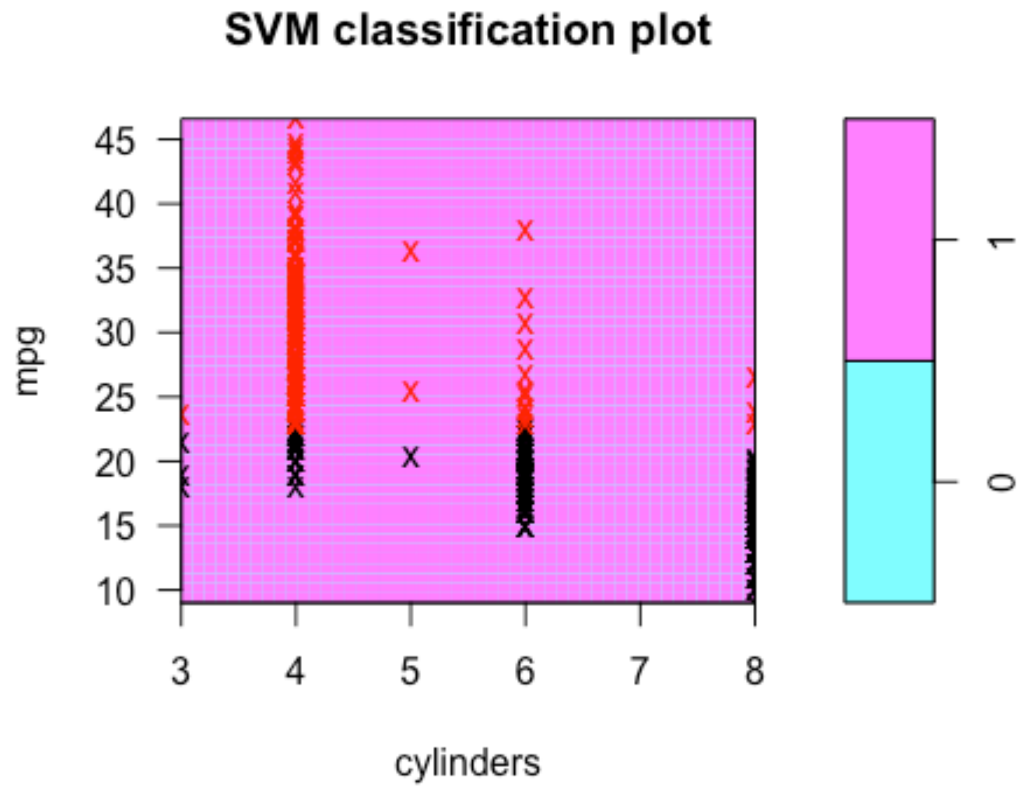

SVM classification plot
mpg
weight
45
40
35
30
25
20
15
10
2000
3000
4000
5000
1
0

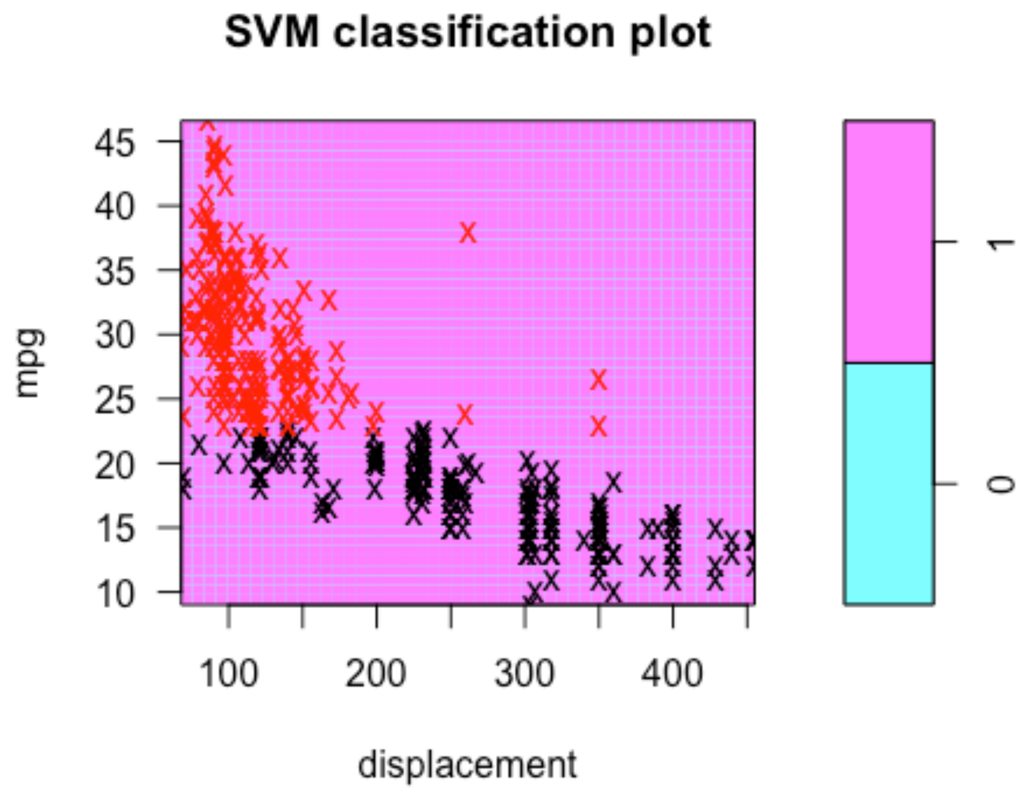

SVM classification plot
mpg
acceleration
45
40
35
30
25
20
15
10
10
15
20
1
0

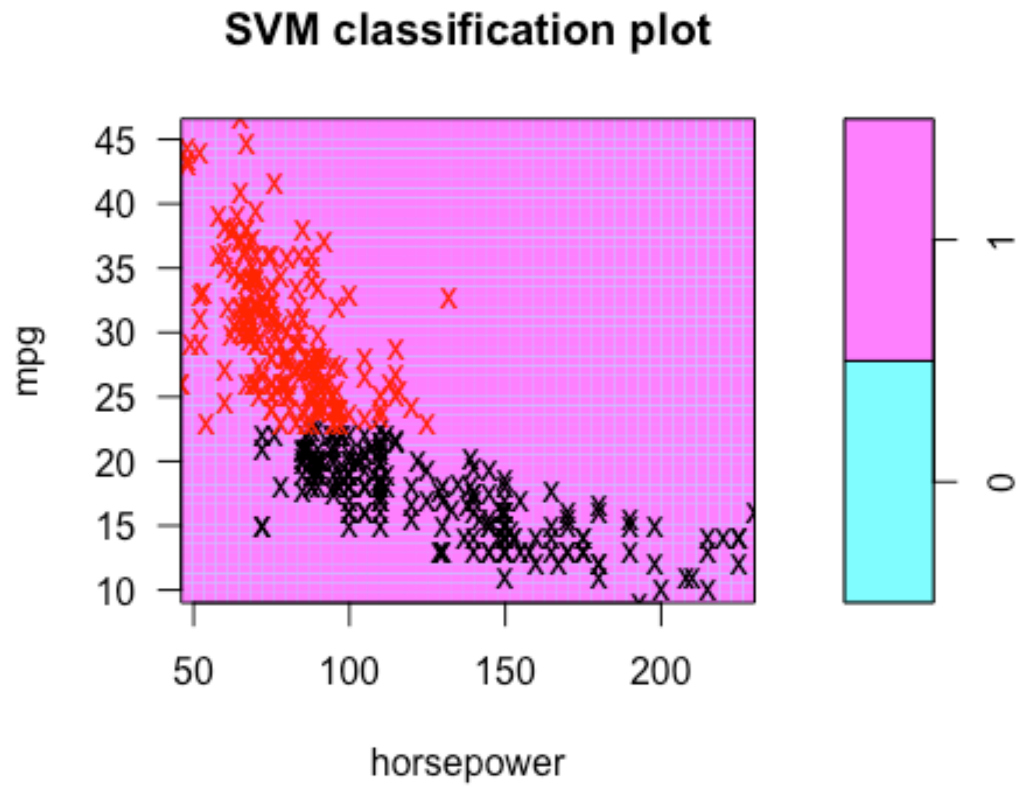

SVM classification plot
mpg
year
45
40
35
30
25
20
15
10
70
72
74
76
78
80
82
1
0

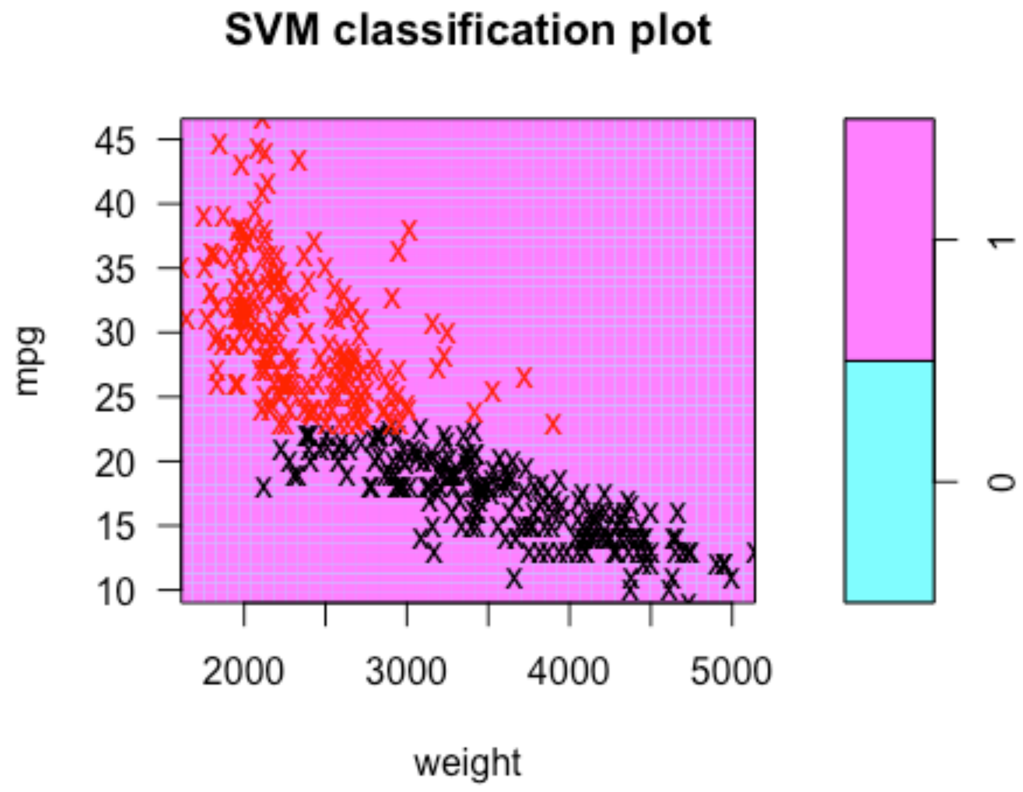


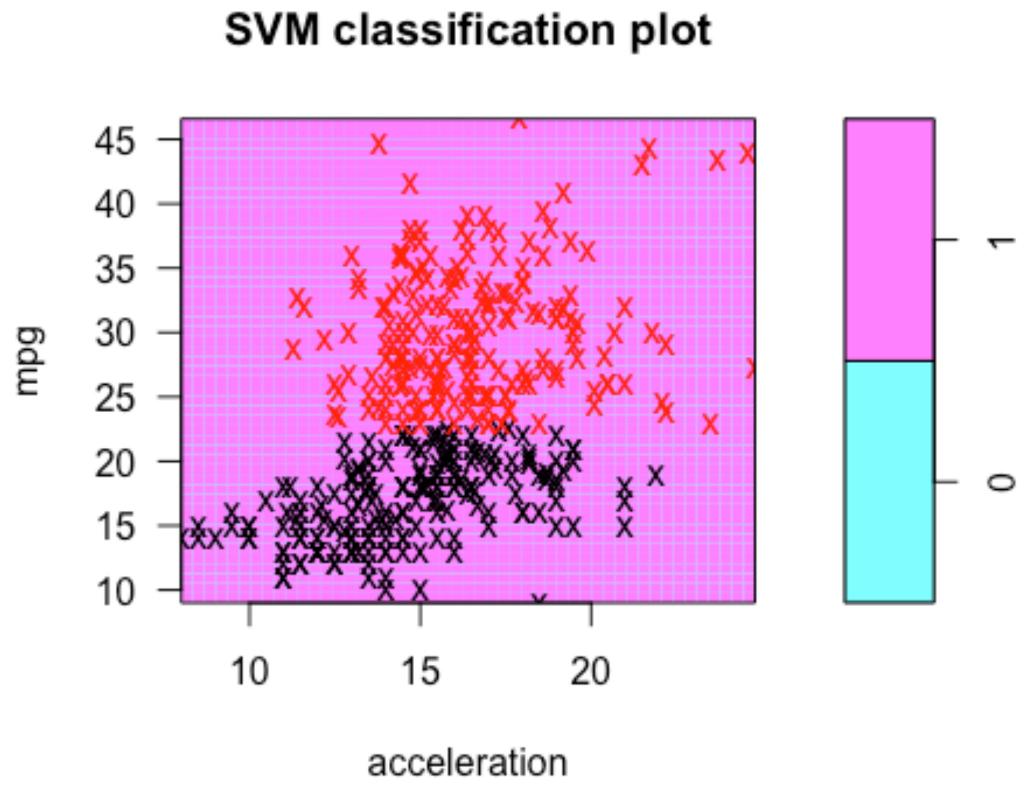
```
plotpairs(svm_polynomial)
```

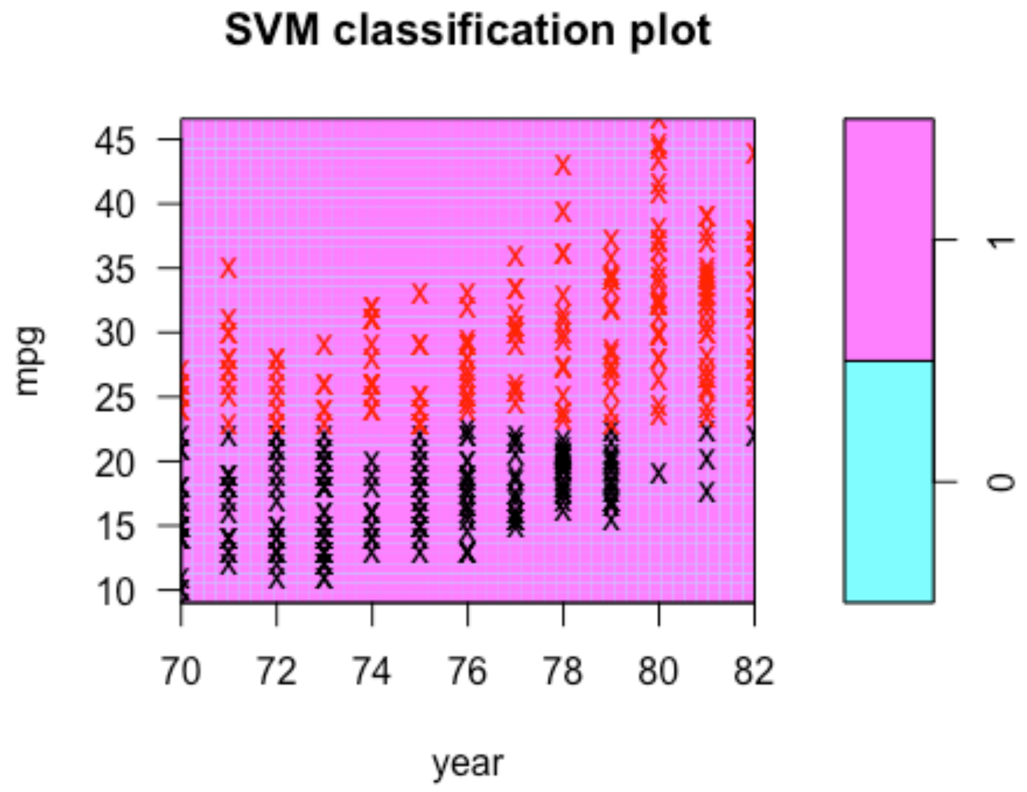

SVM classification plot
mpg
cylinders
45
40
35
30
25
20
15
10
3
4
5
6
7
8
1
0

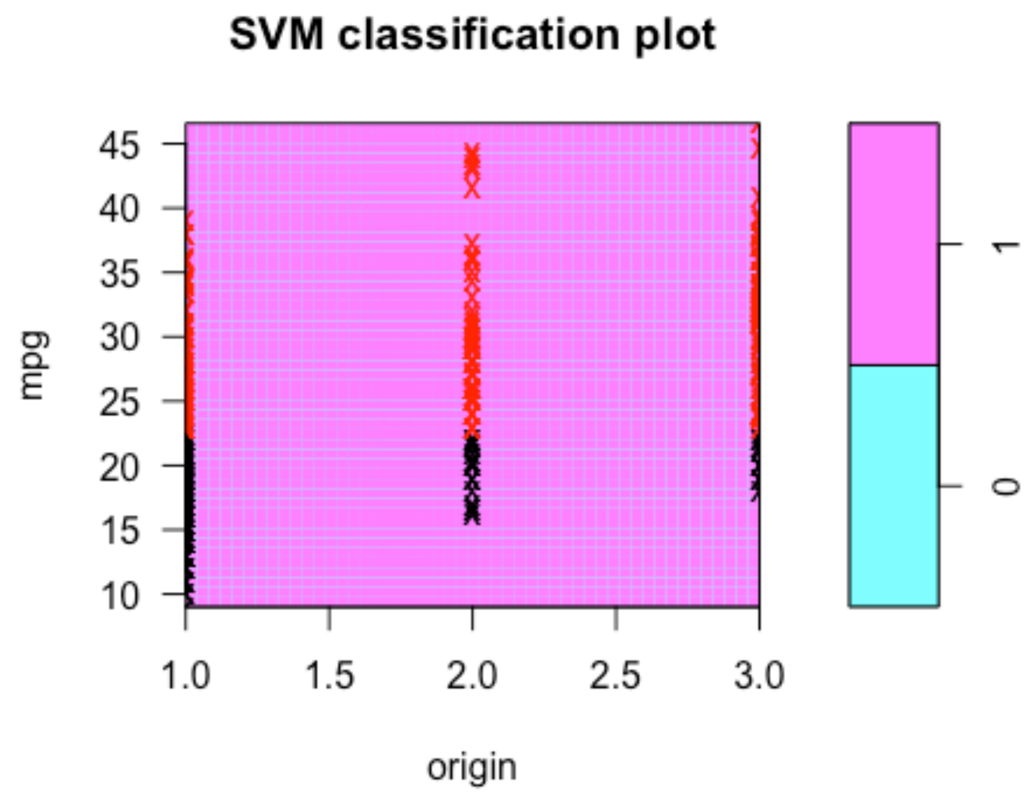

SVM classification plot
mpg
displacement
45
40
35
30
25
20
15
10
100
200
300
400
1
0


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

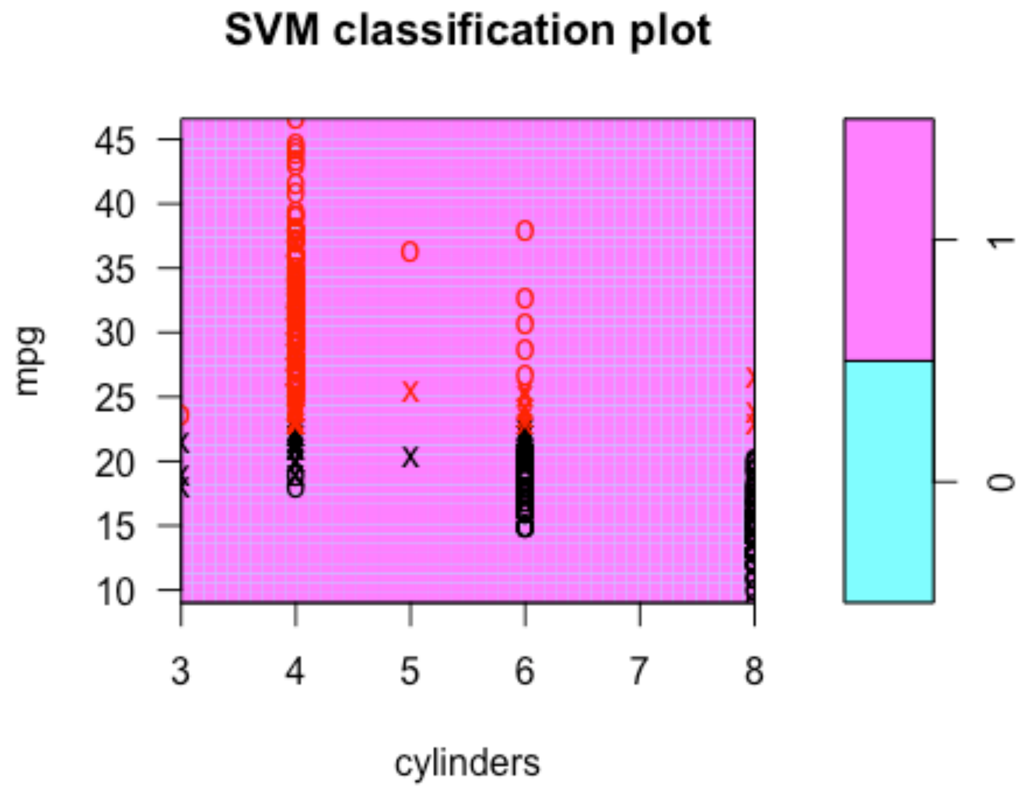

SVM classification plot
mpg
weight
45
40
35
30
25
20
15
10
2000
3000
4000
5000
1
0

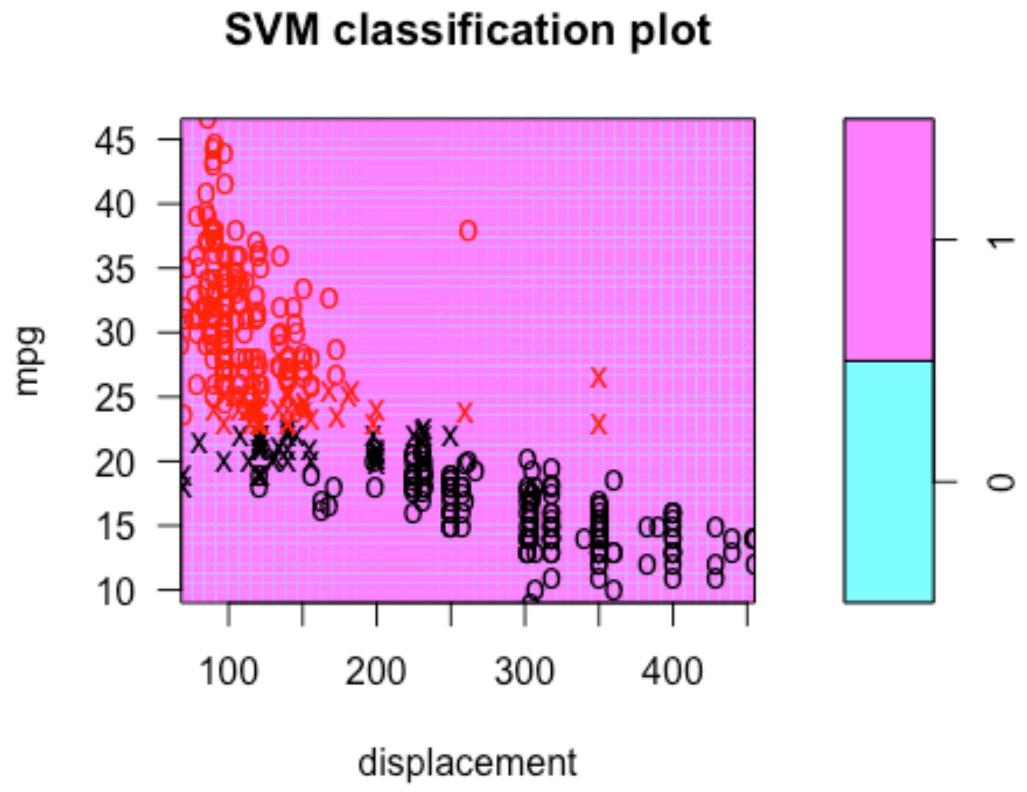

SVM classification plot
mpg
45
40
35
30
25
20
15
10
10
15
20
acceleration
1
0

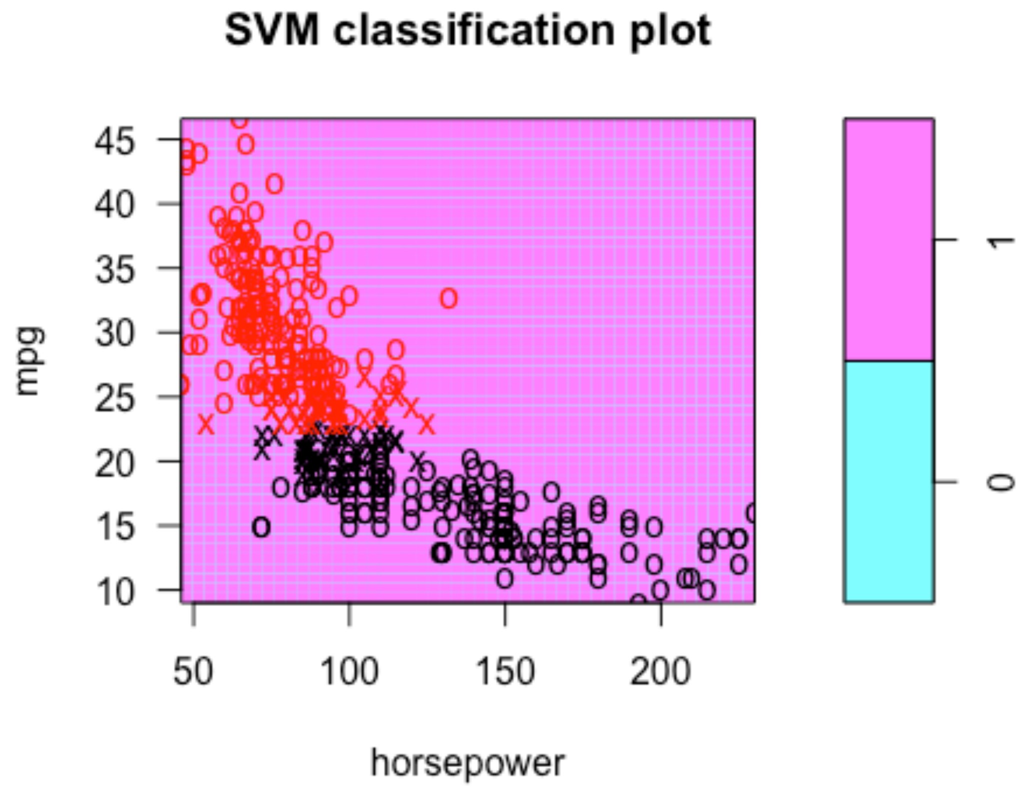

SVM classification plot
mpg
year
45
40
35
30
25
20
15
10
70
72
74
76
78
80
82
1
0

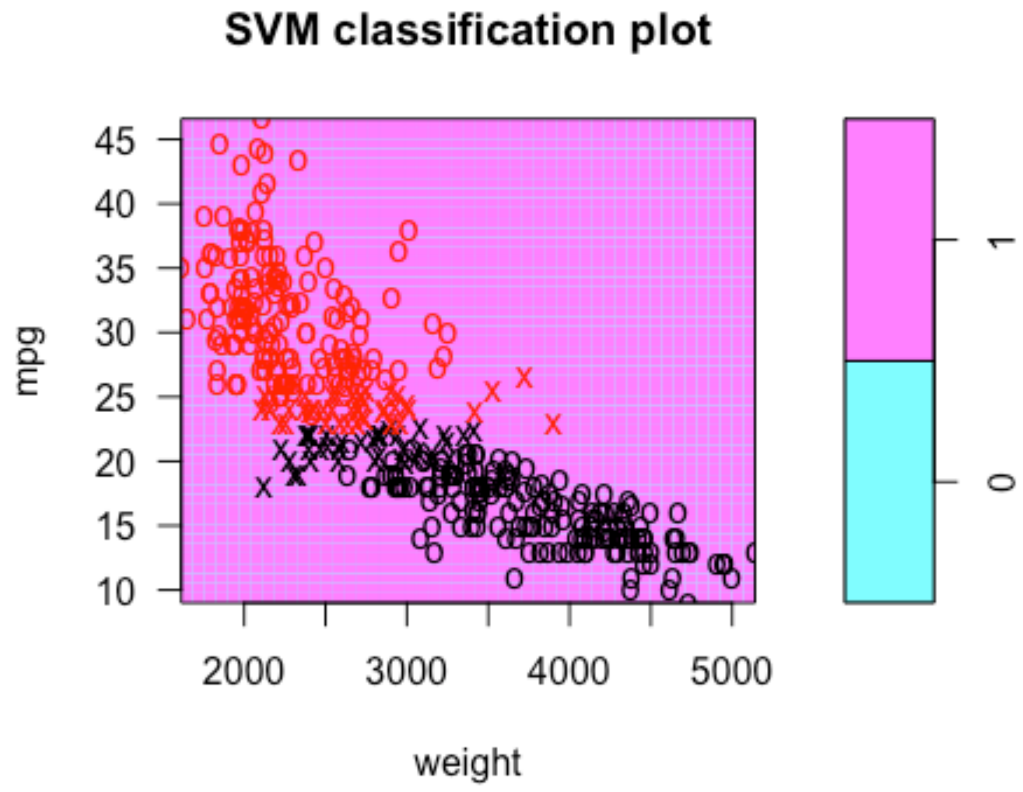


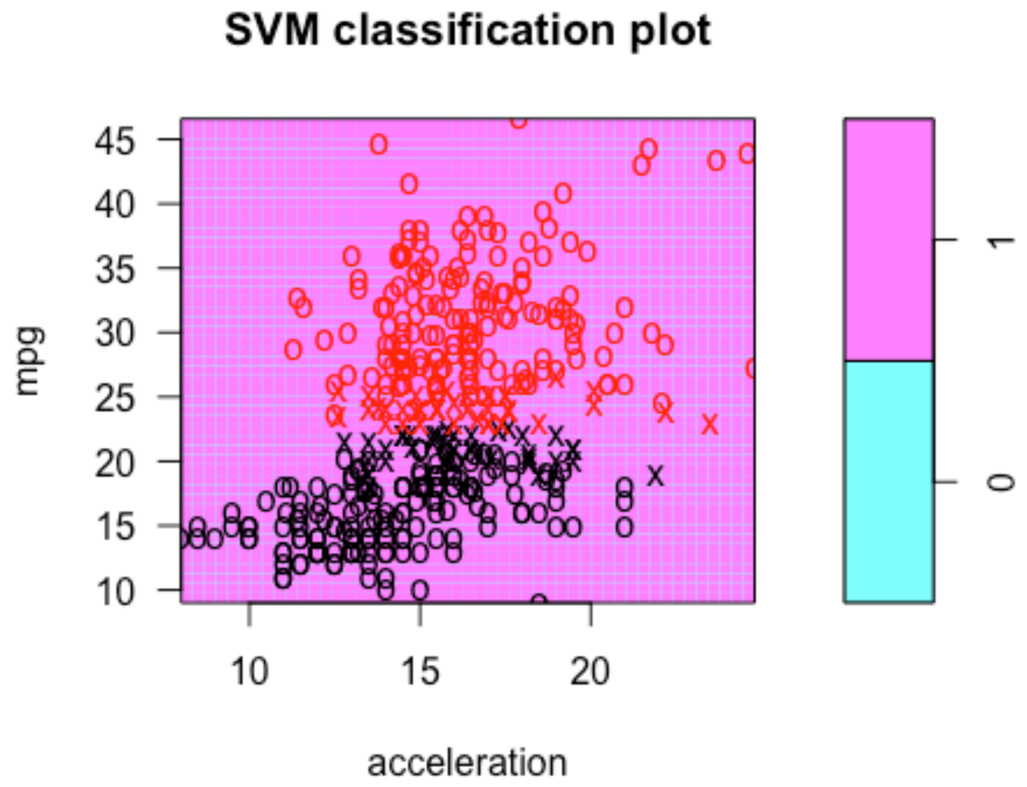
```
plotpairs(svm_radial)
```

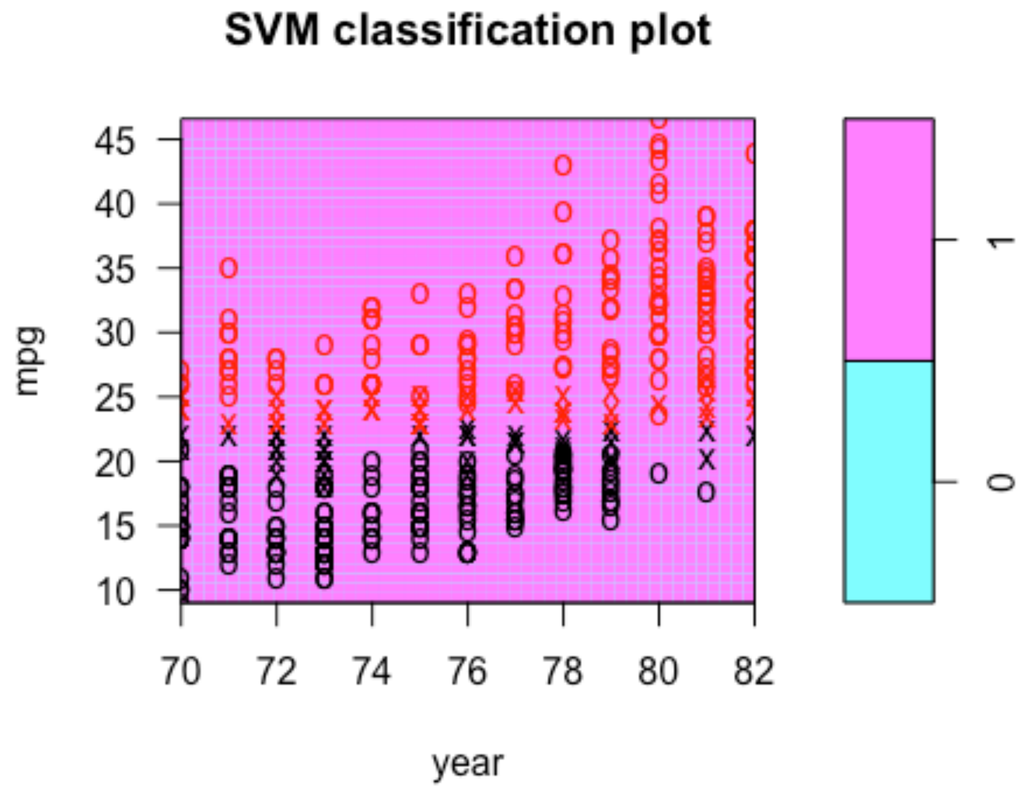

SVM classification plot
mpg
cylinders
45
40
35
30
25
20
15
10
3
4
5
6
7
8
1
0

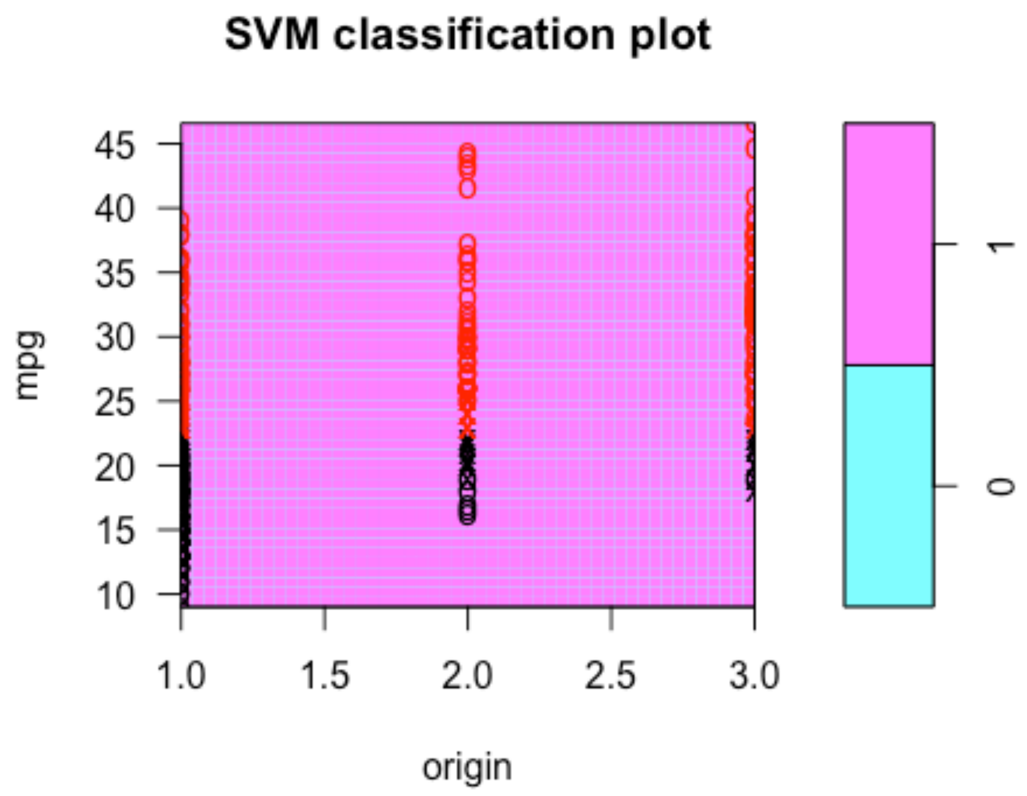

SVM classification plot
mpg
displacement
45
40
35
30
25
20
15
10
100
200
300
400
1
0


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

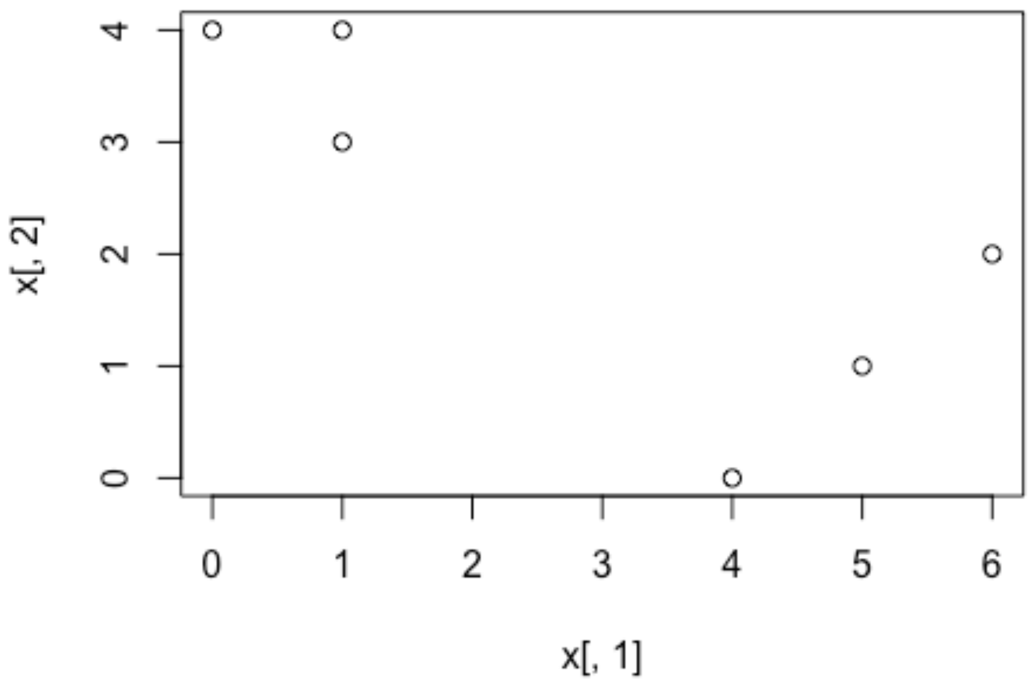

SVM classification plot
mpg
45
40
35
30
25
20
15
10
2000
3000
4000
5000
weight
1
0


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

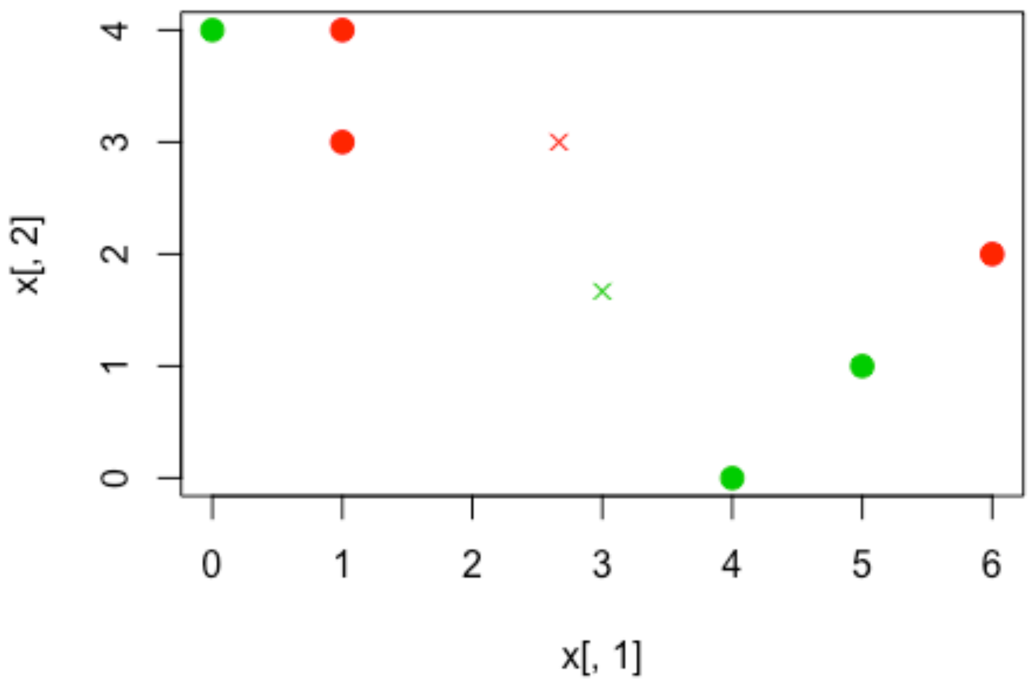

SVM classification plot
mpg
year
45
40
35
30
25
20
15
10
70
72
74
76
78
80
82
1
0



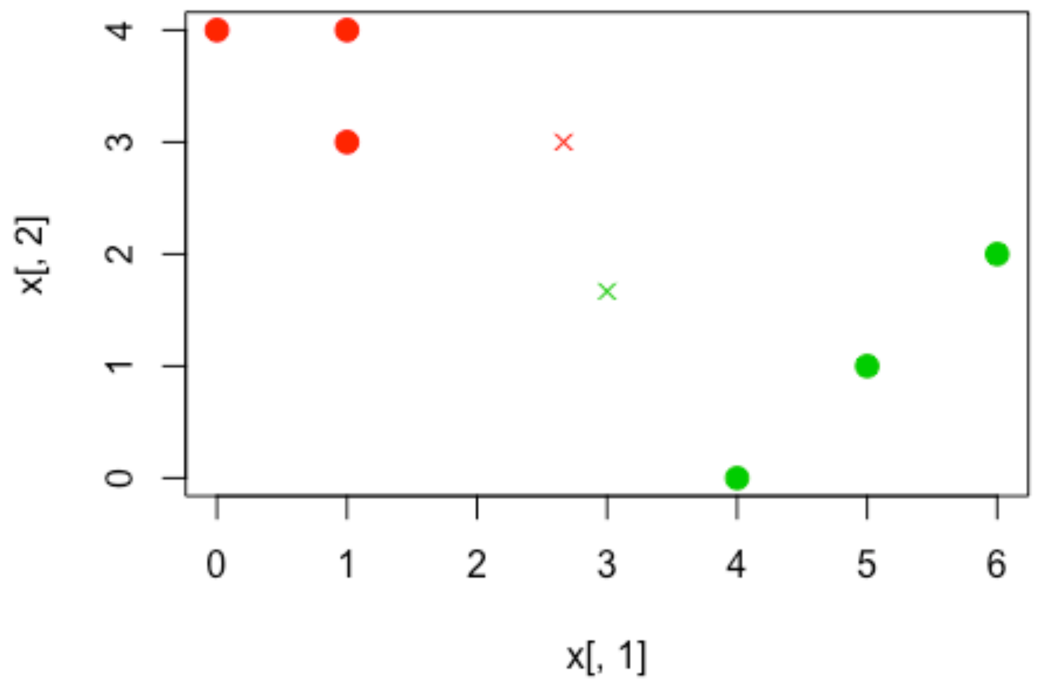
```
#Ex3
#a
set.seed(1)
x <- cbind(c(1, 1, 0, 5, 6, 4), c(4, 3, 4, 1, 2, 0))
plot(x[,1], x[,2])
```



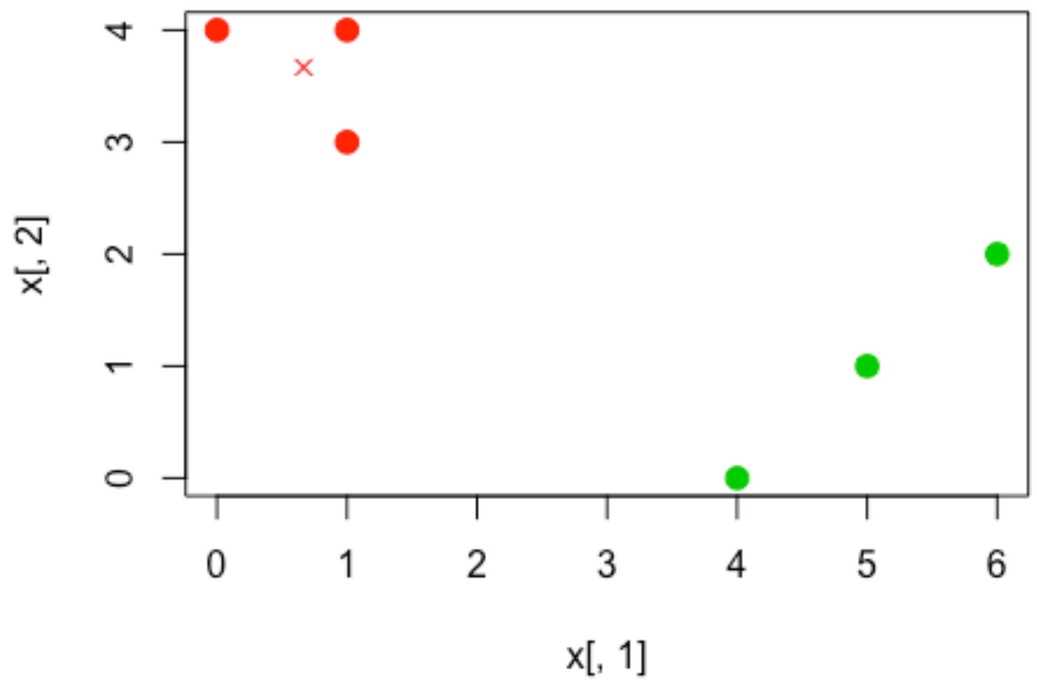
```
#b
set.seed(1)
labels <- sample(2, nrow(x), replace = T)
labels

## [1] 1 1 2 2 1 2

plot(x[, 1], x[, 2], col = (labels + 1), pch = 20, cex = 2)

#c
centroid1 <- c(mean(x[labels == 1, 1]), mean(x[labels == 1, 2]))
centroid2 <- c(mean(x[labels == 2, 1]), mean(x[labels == 2, 2]))
plot(x[,1], x[,2], col=(labels + 1), pch = 20, cex = 2)
points(centroid1[1], centroid1[2], col = 2, pch = 4)
points(centroid2[1], centroid2[2], col = 3, pch = 4)
```



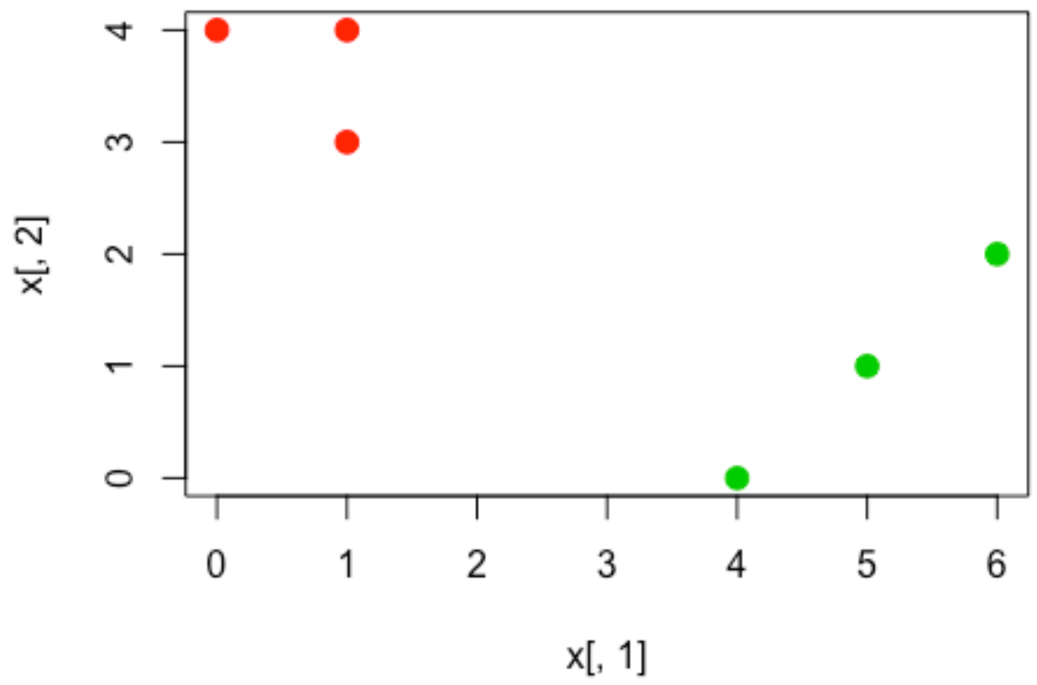
```
#d
labels <- c(1, 1, 1, 2, 2, 2)
plot(x[, 1], x[, 2], col = (labels + 1), pch = 20, cex = 2)
points(centroid1[1], centroid1[2], col = 2, pch = 4)
points(centroid2[1], centroid2[2], col = 3, pch = 4)
```



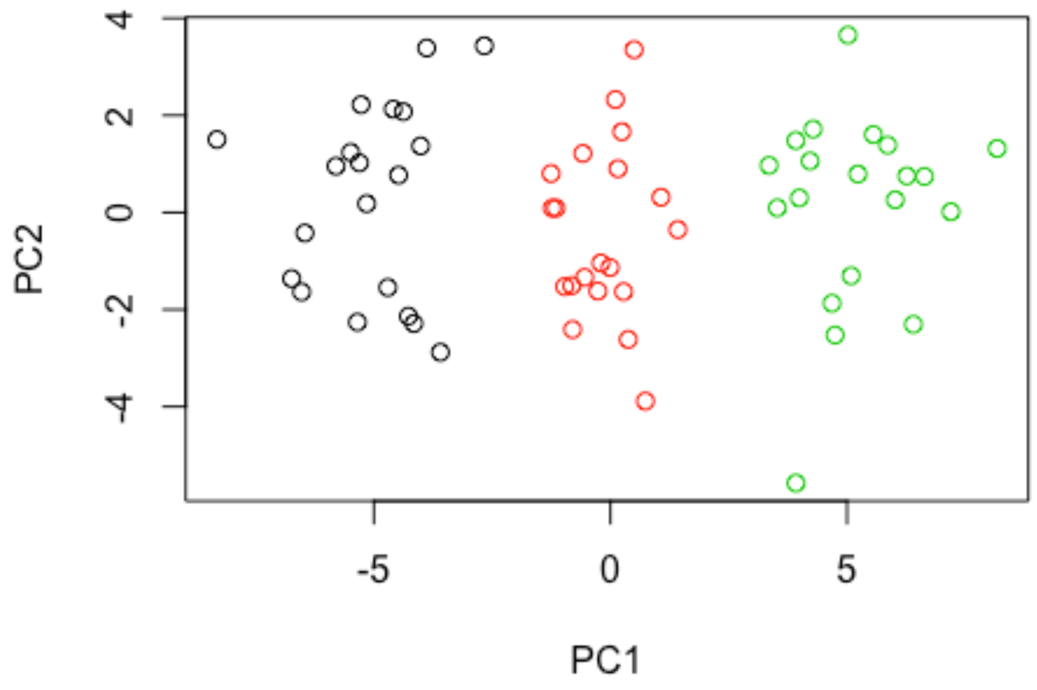
```
#e
centroid1 <- c(mean(x[labels == 1, 1]), mean(x[labels == 1, 2]))
centroid2 <- c(mean(x[labels == 2, 1]), mean(x[labels == 2, 2]))
plot(x[,1], x[,2], col=(labels + 1), pch = 20, cex = 2)
points(centroid1[1], centroid1[2], col = 2, pch = 4)
points(centroid2[1], centroid2[2], col = 3, pch = 4)
```



```r
#f
plot(x[, 1], x[, 2], col=(labels + 1), pch = 20, cex = 2)
```



```
#Ex 10
#a
set.seed(1)
X <- rbind(matrix(rnorm(20*50, mean = 0), nrow = 20),
matrix(rnorm(20*50, mean=0.7), nrow = 20),
matrix(rnorm(20*50, mean=1.4), nrow = 20))

#b
X.pca = prcomp(X)$x
plot(X.pca[,1:2], col=c(rep(1,20), rep(2,20), rep(3,20)))
```



```
#c
res = kmeans(X, centers = 3)
true_class = c(rep(1,20), rep(2,20), rep(3,20))
table(res$cluster, true_class)

##    true_class
##      1  2  3
##   1  0  0 20
##   2  0 20  0
##   3 20  0  0

#class 2 is classified correctly. 1 and 3 classes
# seem to be mixed up when classified.
#d
res = kmeans(X, centers = 2)
true = c(rep(1,20), rep(2,20), rep(3,20))
table(res$cluster, true_class)

##    true_class
##      1  2  3
##   1 20 19  0
##   2  0  1 20
```

```
#it seems middle class is grouped in wrong class. Around
#half of classes from group 2 are classified correctly.

#e
res = kmeans(X, centers = 4)
true = c(rep(1,20), rep(2,20), rep(3,20))
table(res$cluster, true_class)

##    true_class
##      1  2  3
##   1  0  0  7
##   2 20  0  0
##   3  0 20  0
##   4  0  0 13

#one of the classes is split into 2 classes

#f
res = kmeans(X.pca[,1:2], centers = 3)
true = c(rep(1,20), rep(2,20), rep(3,20))
table(res$cluster, true_class)

##    true_class
##      1  2  3
##   1  0  0 20
##   2  0 20  0
##   3 20  0  0

#g
res = kmeans(scale(X), centers = 3)
true = c(rep(1,20), rep(2,20), rep(3,20))
table(res$cluster, true_class)

##    true_class
##      1  2  3
##   1  0  0 20
##   2 20  0  0
##   3  0 20  0

# it seems points are now perfectly classified.
```