Assignment-4 Statistical Modelling

Mughundhan Chandrasekar

11/20/2017

Creating Environment

```r
rm(list = ls())
setwd("/Users/Mughundhan/UIC/UIC Academics/FALL 2017/BIZ ANALYTICS
STATS/Assignment 4")
```

Question 1.

Here we explore the maximal margin classifier on a toy data set.

Question 1.(a)

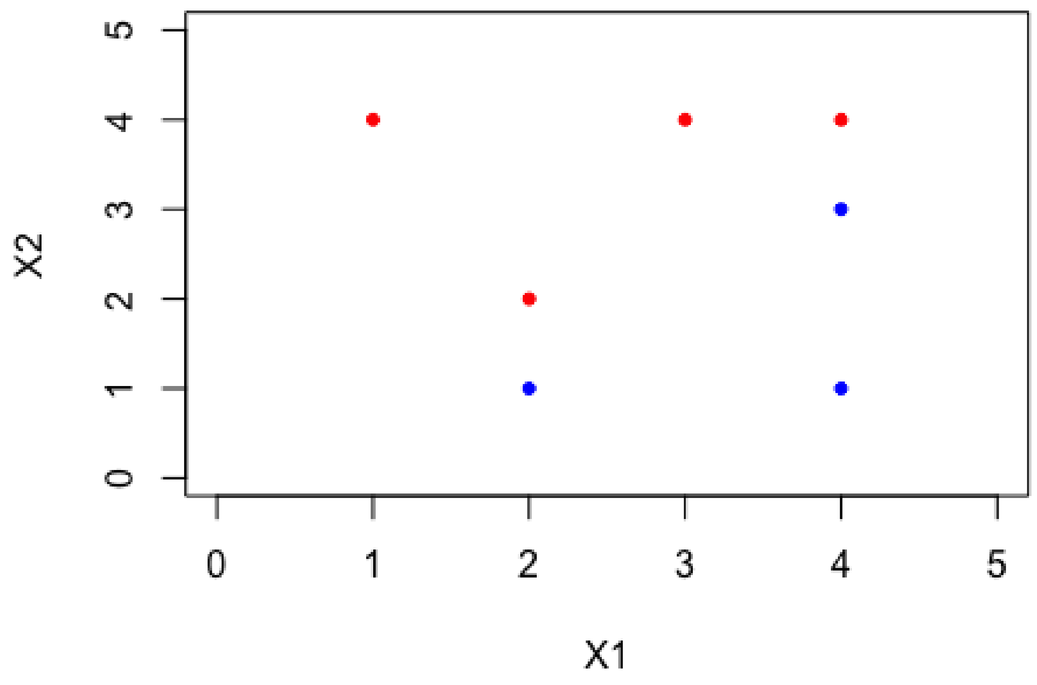
We are given n = 7 observations in p = 2 dimensions. For each observation, there is an associated class label. Sketch the observations.

Answer: The observation is sketched as shown below.

```r
X1 <- c(3, 2, 4, 1, 2, 4, 4)
X2 <- c(4, 2, 4, 4, 1, 3, 1)
Y <- c("red", "red", "red", "red", "blue", "blue", "blue")
fdata <- as.data.frame(cbind(X1, X2, Y))
fdata
```

```
##   X1 X2    Y
## 1  3  4  red
## 2  2  2  red
## 3  4  4  red
## 4  1  4  red
## 5  2  1 blue
## 6  4  3 blue
## 7  4  1 blue
```

```r
plot(X1, X2, col = Y, xlim = c(0, 5), ylim = c(0, 5), pch=20)
```



Question 1.(b)

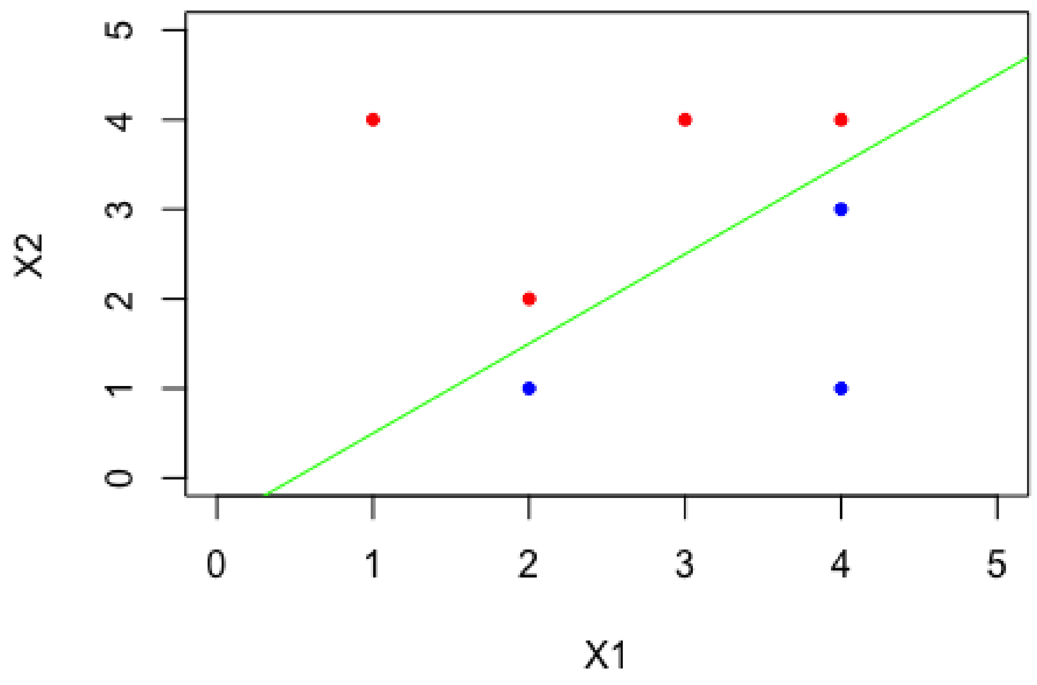
Sketch the optimal separating hyperplane, and provide the equation for this hyperplane.

Answer:

As shown in the plot, the optimal separating hyperplane has to be between the observations **(2,1)** and **(2,2)**, and between the observations **(4,3)** and **(4,4)**.

So it is obvious that the line passes through the points **(2, 1.5)** and **(4, 3.5)**. The corresponding equation is given as follows: **X1 - X2 - 0.5 = 0**

```
library(e1071)
library(dplyr)
slope=1
intercept=-0.5
plot(X1, X2, col = Y, xlim = c(0, 5), ylim = c(0, 5), pch=20)
abline(intercept, slope, col="green")
```



Question 1.(c)

Describe the classification rule for the maximal margin classifier. It should be something along the lines of ***Classify to Red if b0 + b1X1 + b2X2 > 0, and classify to Blue otherwise***. Provide the values for b0, b1, and b2.

NOTE: "b"" denotes beta

Answer:

```
beta <- c(-intercept, solve(matrix(c(2, 1.5, 4, 3.5), nrow = 2, byrow = T),
c(intercept, intercept)))
beta

## [1]  0.5 -1.0  1.0
```

Observation: The beta values are given as follows:

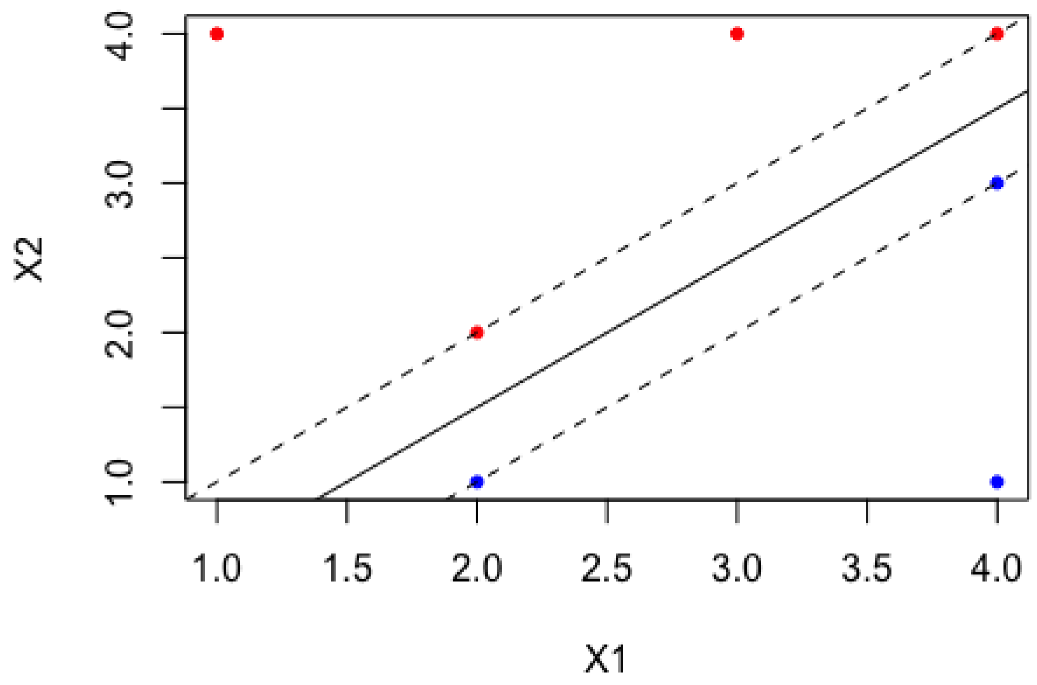
- b0 = 0.5
- b1 = -1
- b2 = 1

Question 1.(d)

On your sketch, indicate the margin for the maximal margin hyperplane.

Answer: The sketch shown below, which is followed by the observation.

```r
plot(X1, X2, col=Y, pch=20, data=fdata)
abline(-beta[1], beta[3])
abline(beta[2], beta[3], lty = 2)
abline(0, beta[3], lty = 2)
```



Observation:

The maximal margin hyperplane is shown as a solid line. The "Margin Width" is the distance from the solid line to either of the dashed lines which is 1/4.

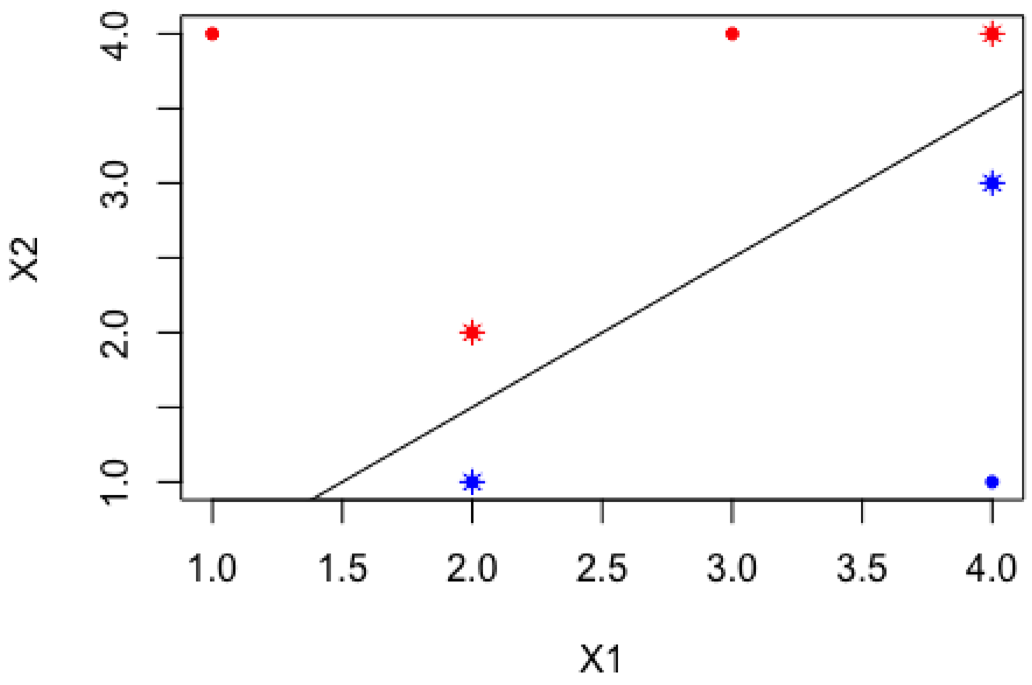
Question 1.(e)

Indicate the support vectors for the maximal margin classifier.

Answer: The corresponding coordinates are indicated as * (asterisk)

```r
# Replot, this time indicating the support vectors for the maximal margin
classifier.
```

```r
plot(X1, X2, col=Y, pch=20, data=fdata)

abline(intercept, slope)
points(c(2, 4), c(2, 4), pch = 8, col = "red")
points(c(2, 4), c(1, 3), pch = 8, col = "blue")
```



Observation: We can see that the coordinates ***(2, 1), (2, 2), (4, 3) and (4, 4)*** are the support vectors for the maximal margin classifier.

Question 1.(f)

Argue that a slight movement of the seventh observation would not affect the maximal margin hyperplane.

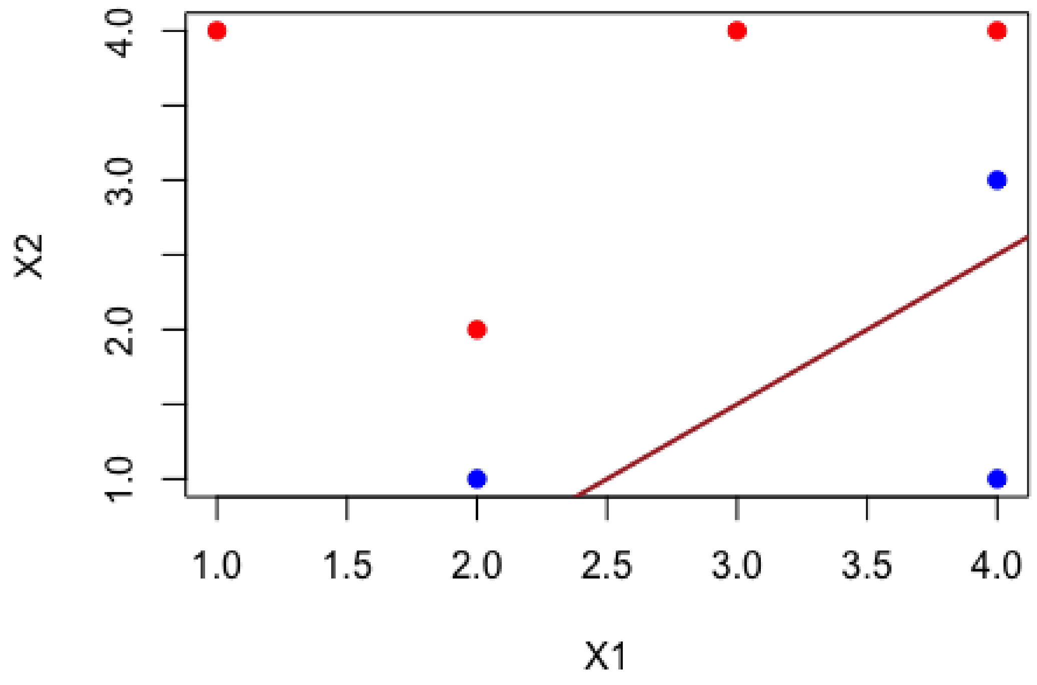
Answer: The seventh observation is (4, 1). Slight movement of the seventh observation would not affect the maximal margin hyperplane ***because the seventh observation (4, 1) is not a support vector*** and is quite far from the margins of the separating hyperplane with maximum margin

Question 1.(g)

Sketch a hyperplane that is not the optimal separating hyperplane, and provide the equation for this hyperplane.

Answer: For example, the hyperplane which equation is **X1 - X2 -1.5 = 0** is not the optimal separating hyperplane.

```r
plot(X1, X2, col=Y, pch=19, data=fdata)
abline(-1.5, 1, col="brown",lwd=2)
```

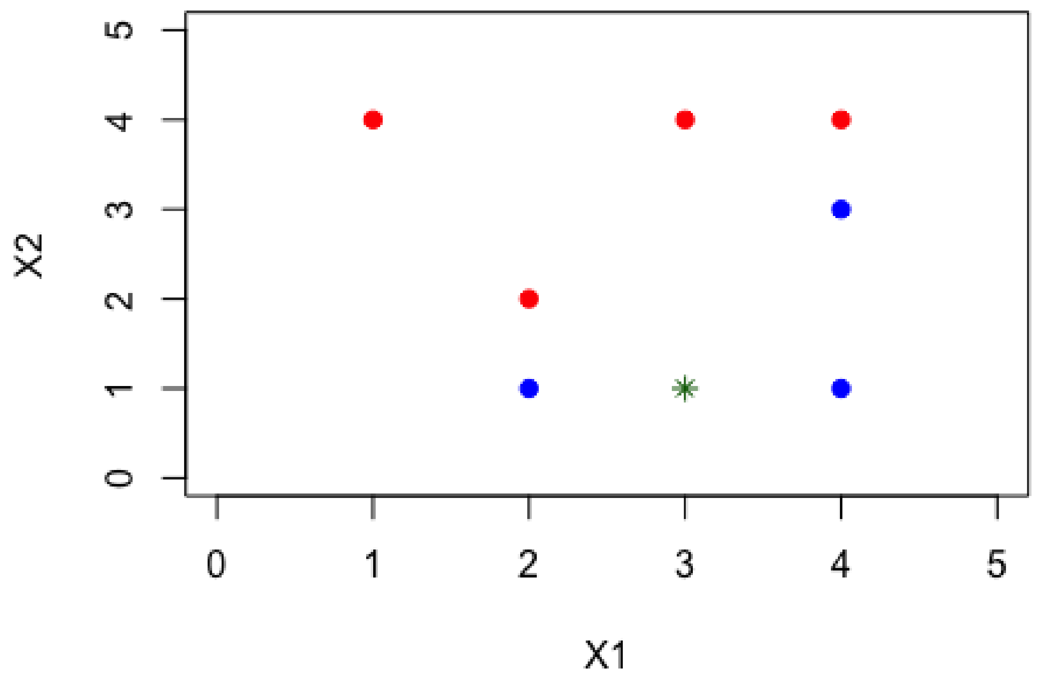


Question 1.(h)

Draw an additional observation on the plot so that the two classes are no longer separable by a hyperplane.

Answer: The additional observation is (3, 1) which is represented by green coloured *.

```r
plot(X1, X2, col = Y, xlim = c(0, 5), ylim = c(0, 5), pch=19)
points(c(3), c(1), col = c("darkgreen"), pch=8)
```



Observation: On including the additional observation **(3, 1)**, the two classes prove to be no longer separable by a hyperplane.

Question 2.

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.

Question 2.(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.

Answer: The binary variable is created as shown below, which is followed by the observtions.

```
library(ISLR)
median_mileage <- median(Auto$mpg)

#Median for the mileage
median_mileage

## [1] 22.75
```

```
mpg_level <- ifelse(Auto$mpg > median(Auto$mpg), 1, 0)
Auto$mpg_level <- as.factor(mpg_level)
head(Auto)
```

```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                        name mpg_level
## 1 chevrolet chevelle malibu         0
## 2         buick skylark 320         0
## 3        plymouth satellite         0
## 4             amc rebel sst         0
## 5               ford torino         0
## 6          ford galaxie 500         0
```

Observation:

1. We can see that the median for the mileage is 22.75
2. mpg_level has two levels:

- "1" if the mileage greater than the median mileage (found to be 22.75)
- "0" if the mileage lesser than or equal to the median mileage

Question 2.(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.

Answer: The comments are given below the r-code and the corresponding output.

```
library(e1071)
set.seed(1)
tune_linear <- tune(svm, mpg_level ~ ., data = Auto, kernel = "linear",
ranges = list(cost = c(0.01, 0.1, 1, 5, 10, 100, 1000)))
summary(tune_linear)
```

```
##
## 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
## 7 1e+03 0.03820513 0.01773427
```

Observation:

1. We can see that the 10-fold Cross-validation has worked perfecty.
2. For a Linear Kernel, the Lowest cross-validation error is obtained for the **Cost of 1**.

Question 2.(c)

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

Answer: Now let us fit a svm model using Radial and then using Polynomial.

SVM with Radial

```
set.seed(1)
tune_radial <- tune(svm, mpg_level ~ ., data = Auto, kernel = "radial",
ranges = list(cost = c(0.01, 0.1, 1, 5, 10, 100), gamma = c(0.01, 0.1, 1, 5,
10, 100)))
summary(tune_radial)

##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##   100  0.01
##
## - best performance: 0.01532051
##
## - Detailed performance results:
##     cost gamma      error dispersion
## 1  1e-02 1e-02 0.56115385 0.04344202
## 2  1e-01 1e-02 0.09185897 0.03862507
## 3  1e+00 1e-02 0.07147436 0.05103685
## 4  5e+00 1e-02 0.04326923 0.04975032
## 5  1e+01 1e-02 0.02551282 0.03812986
## 6  1e+02 1e-02 0.01532051 0.01788871
## 7  1e-02 1e-01 0.19153846 0.07612945
## 8  1e-01 1e-01 0.07916667 0.05201159
## 9  1e+00 1e-01 0.05608974 0.05092939
## 10 5e+00 1e-01 0.03064103 0.02637448
## 11 1e+01 1e-01 0.02551282 0.02076457
## 12 1e+02 1e-01 0.02807692 0.01458261
## 13 1e-02 1e+00 0.56115385 0.04344202
## 14 1e-01 1e+00 0.56115385 0.04344202
## 15 1e+00 1e+00 0.06634615 0.06187383
## 16 5e+00 1e+00 0.06128205 0.06186124
## 17 1e+01 1e+00 0.06128205 0.06186124
## 18 1e+02 1e+00 0.06128205 0.06186124
## 19 1e-02 5e+00 0.56115385 0.04344202
## 20 1e-01 5e+00 0.56115385 0.04344202
## 21 1e+00 5e+00 0.49224359 0.03806832
## 22 5e+00 5e+00 0.48967949 0.03738577
## 23 1e+01 5e+00 0.48967949 0.03738577
## 24 1e+02 5e+00 0.48967949 0.03738577
## 25 1e-02 1e+01 0.56115385 0.04344202
## 26 1e-01 1e+01 0.56115385 0.04344202
## 27 1e+00 1e+01 0.51775641 0.04471079
## 28 5e+00 1e+01 0.51012821 0.03817175
## 29 1e+01 1e+01 0.51012821 0.03817175
## 30 1e+02 1e+01 0.51012821 0.03817175
## 31 1e-02 1e+02 0.56115385 0.04344202
## 32 1e-01 1e+02 0.56115385 0.04344202
## 33 1e+00 1e+02 0.56115385 0.04344202
## 34 5e+00 1e+02 0.56115385 0.04344202
## 35 1e+01 1e+02 0.56115385 0.04344202
## 36 1e+02 1e+02 0.56115385 0.04344202
```

SVM with Polynomial

```
set.seed(1)
tune_poly <- tune(svm, mpg_level ~ ., data = Auto, kernel = "polynomial",
ranges = list(cost = c(0.01, 0.1, 1, 5, 10, 100), degree = c(2, 3, 4)))
summary(tune_poly)

##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##   100      2
##
## - best performance: 0.3013462
##
## - Detailed performance results:
##     cost degree     error dispersion
## 1  1e-02      2 0.5611538 0.04344202
## 2  1e-01      2 0.5611538 0.04344202
## 3  1e+00      2 0.5611538 0.04344202
## 4  5e+00      2 0.5611538 0.04344202
## 5  1e+01      2 0.5382051 0.05829238
## 6  1e+02      2 0.3013462 0.09040277
## 7  1e-02      3 0.5611538 0.04344202
## 8  1e-01      3 0.5611538 0.04344202
## 9  1e+00      3 0.5611538 0.04344202
## 10 5e+00      3 0.5611538 0.04344202
## 11 1e+01      3 0.5611538 0.04344202
## 12 1e+02      3 0.3322436 0.11140578
## 13 1e-02      4 0.5611538 0.04344202
## 14 1e-01      4 0.5611538 0.04344202
## 15 1e+00      4 0.5611538 0.04344202
## 16 5e+00      4 0.5611538 0.04344202
## 17 1e+01      4 0.5611538 0.04344202
## 18 1e+02      4 0.5611538 0.04344202
```

Observation:

1. For a **radial kernel**, the lowest cross-validation error is obtained for a ***gamma of 0.01 and a cost of 100.***
2. For a **polynomial kernel**, the lowest cross-validation error is obtained for a ***degree of 2 and a cost of 100.***

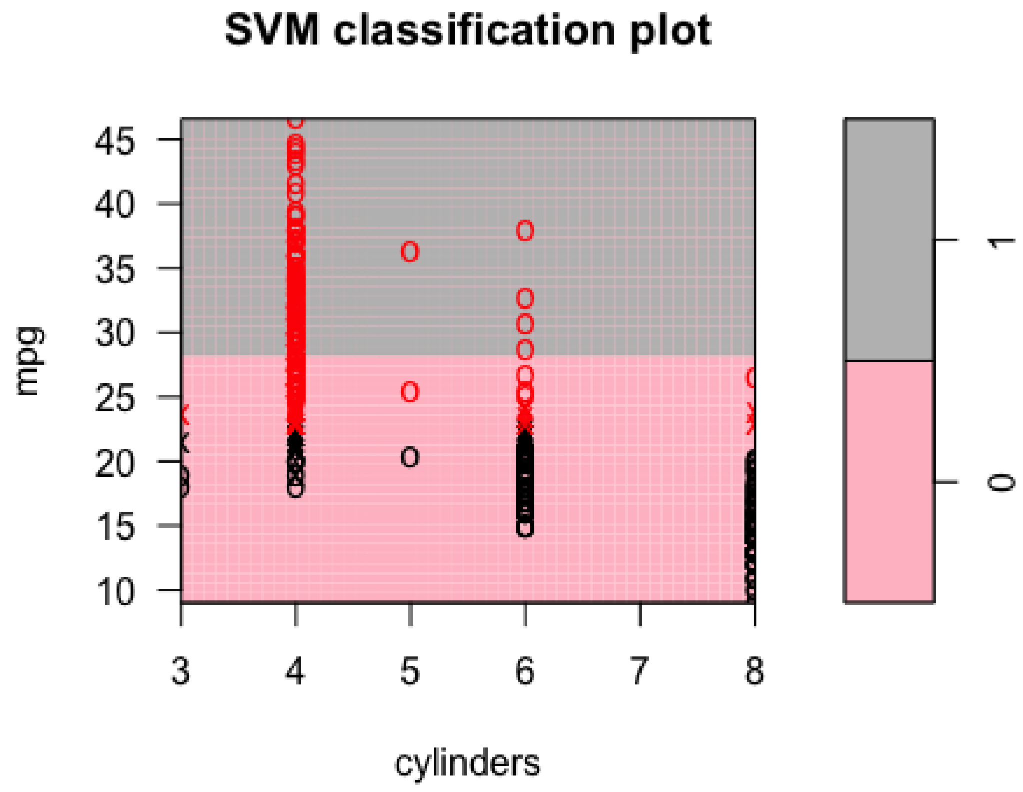
Question 2.(d)

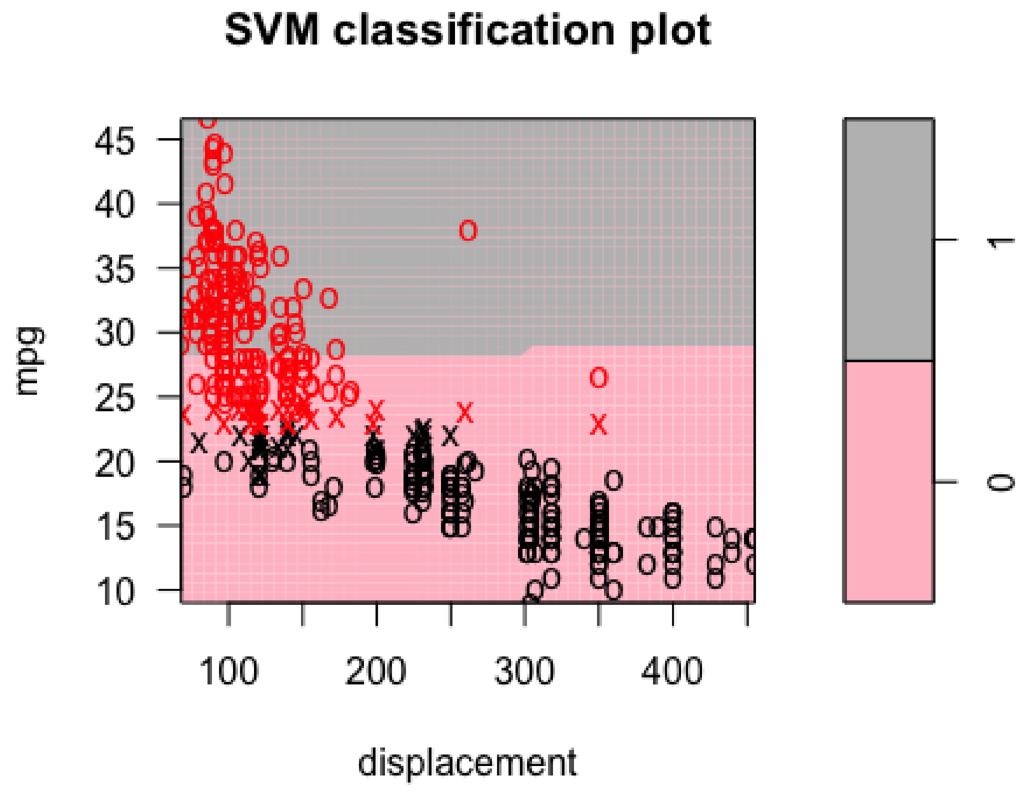
Make some plots to back up your assertions in (b) and (c).

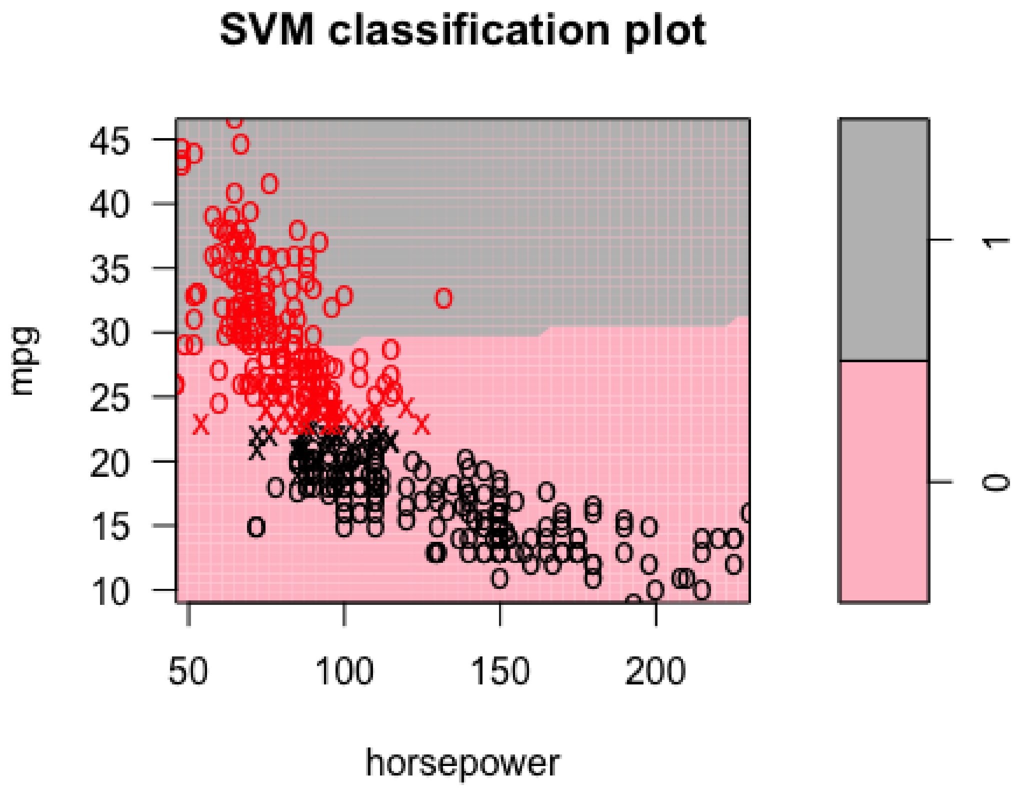
```
Auto$mpg_level <- as.factor(Auto$mpg_level)

svm_linear <- svm(mpg_level ~ ., data = Auto, kernel = "linear", cost = 1)
svm_radial <- svm(mpg_level ~ ., data = Auto, kernel = "radial", cost = 100,
gamma = 0.01)
svm_poly <- svm(mpg_level ~ ., data = Auto, kernel = "polynomial", cost =
100, degree = 2)
plotpairs = function(fit) {
    for (name in names(Auto)[!(names(Auto) %in% c("mpg", "mpg_level",
"name"))]) {
        plot(fit, Auto, col = c("pink", "grey"), as.formula(paste("mpg~",
name, sep = "")))
    }
}
```

Now let us make some plots in support of assertions made for SVM with Linear Kernel

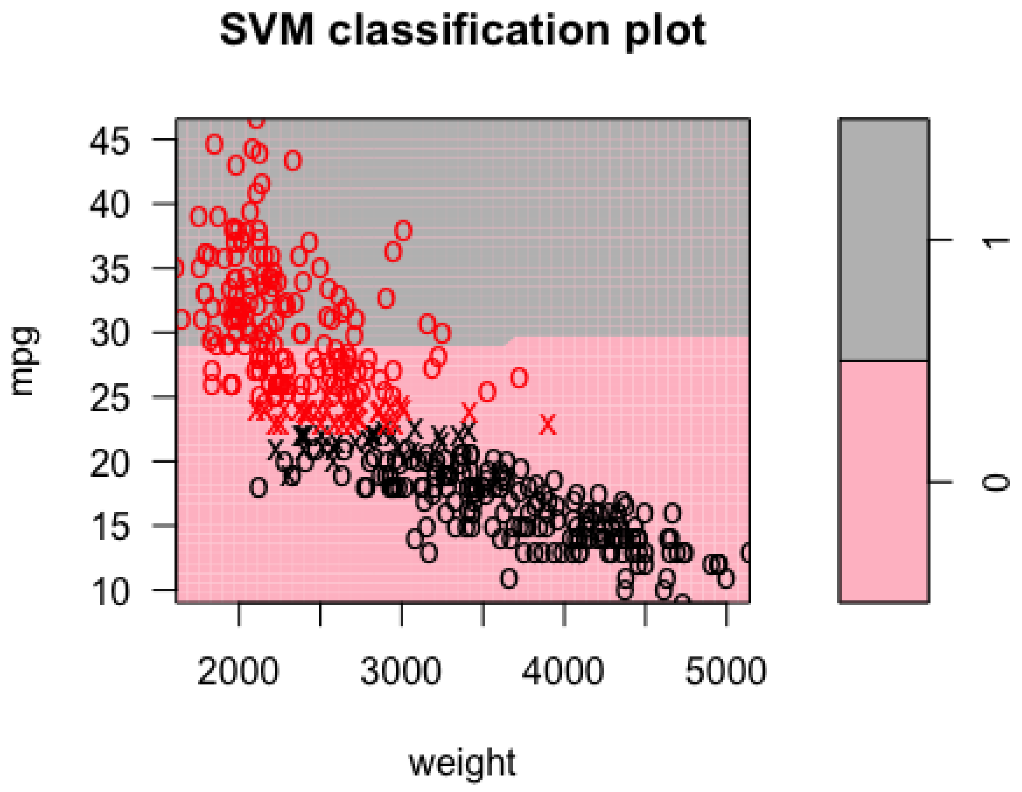
```
plotpairs(svm_linear)
```

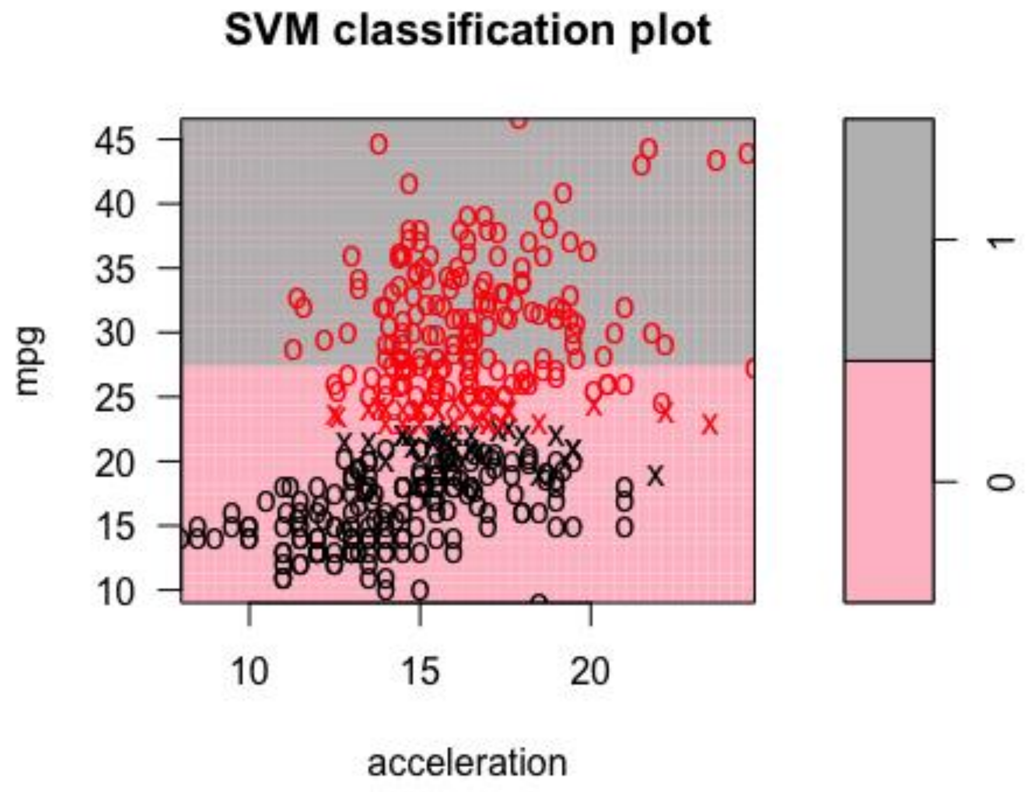

SVM classification plot
mpg
cylinders
45
40
35
30
25
20
15
10
3
4
5
6
7
8
1
0


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

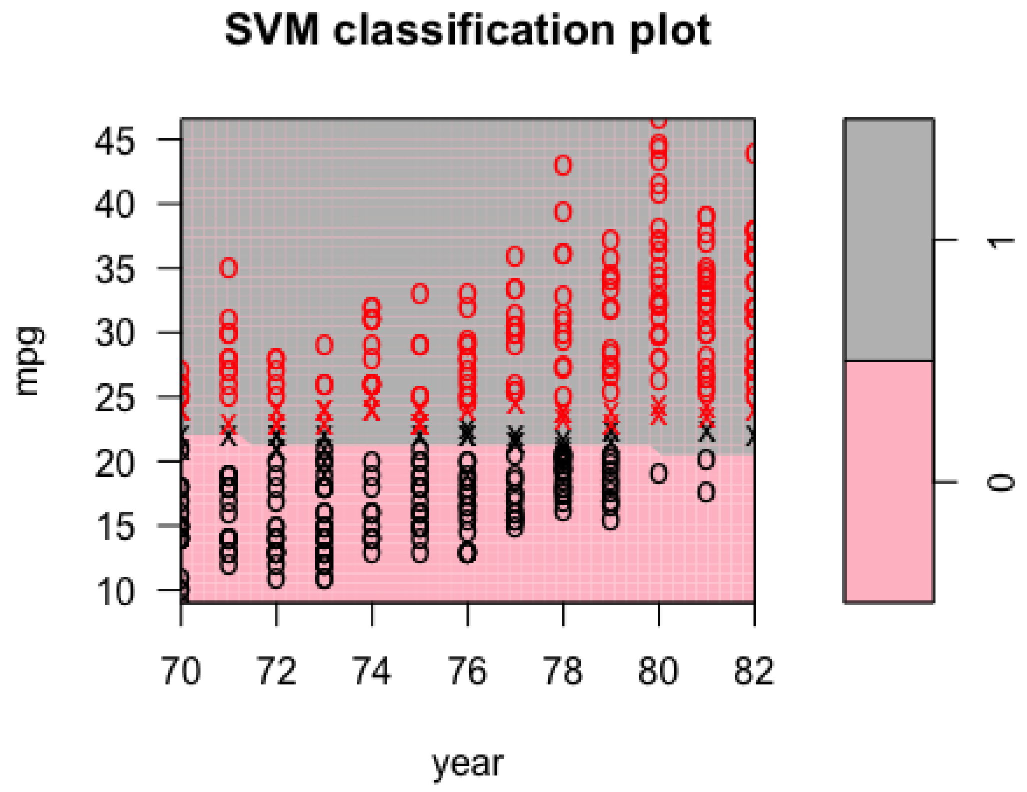


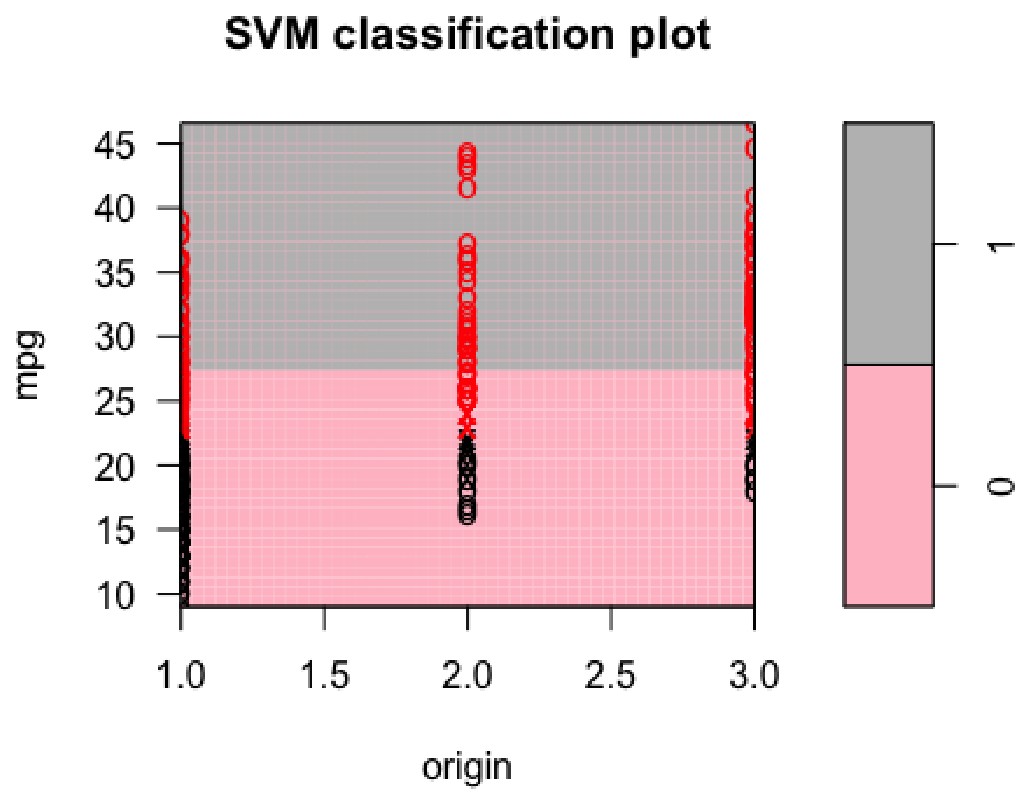
SVM classification plot
mpg
horsepower
45
40
35
30
25
20
15
10
50
100
150
200
1
0


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



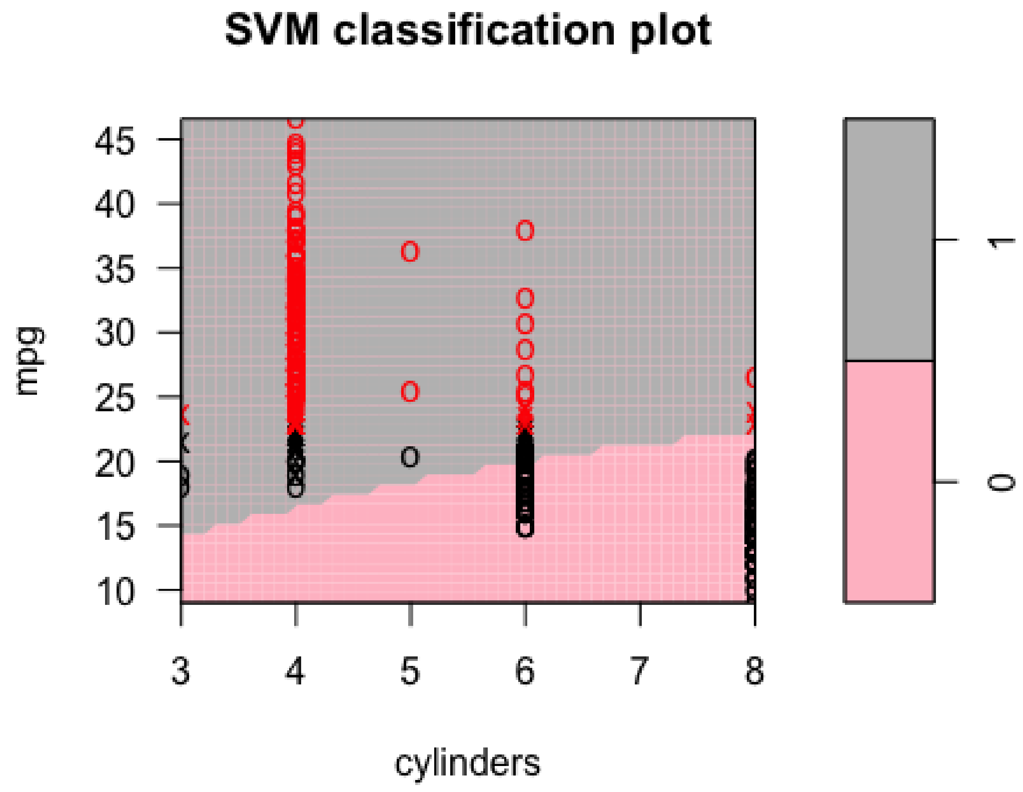
SVM classification plot
mpg
acceleration
45
40
35
30
25
20
15
10
10
15
20
1
0

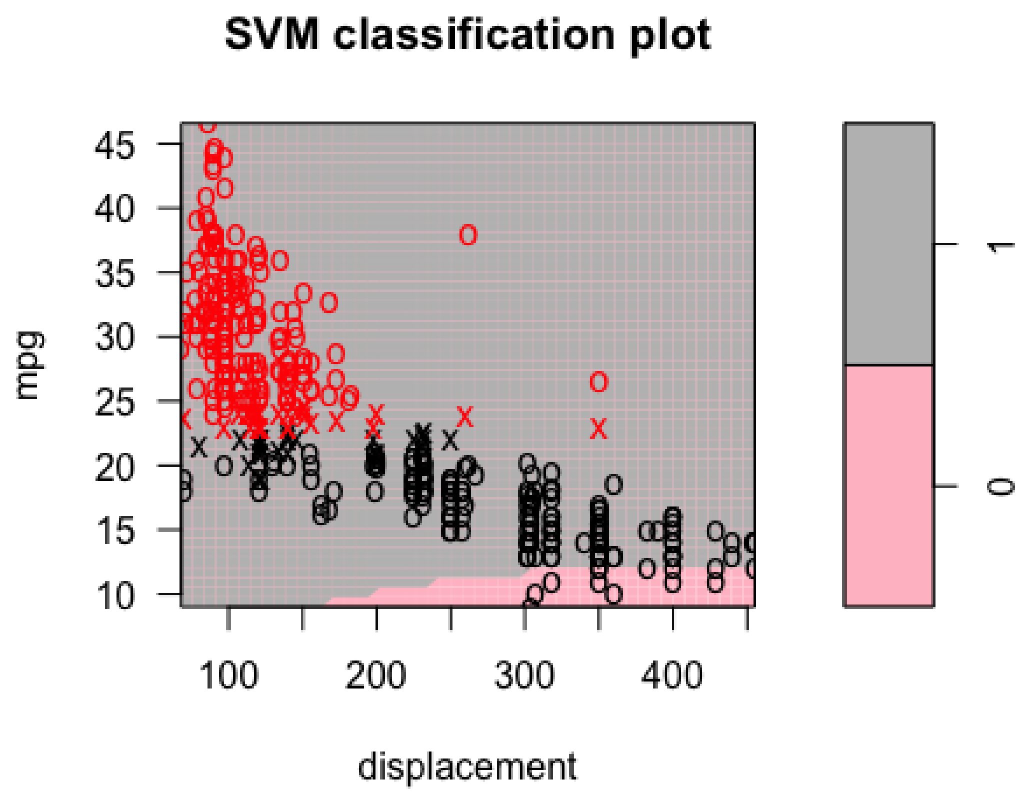

SVM classification plot
mpg
year
45
40
35
30
25
20
15
10
70
72
74
76
78
80
82
1
0

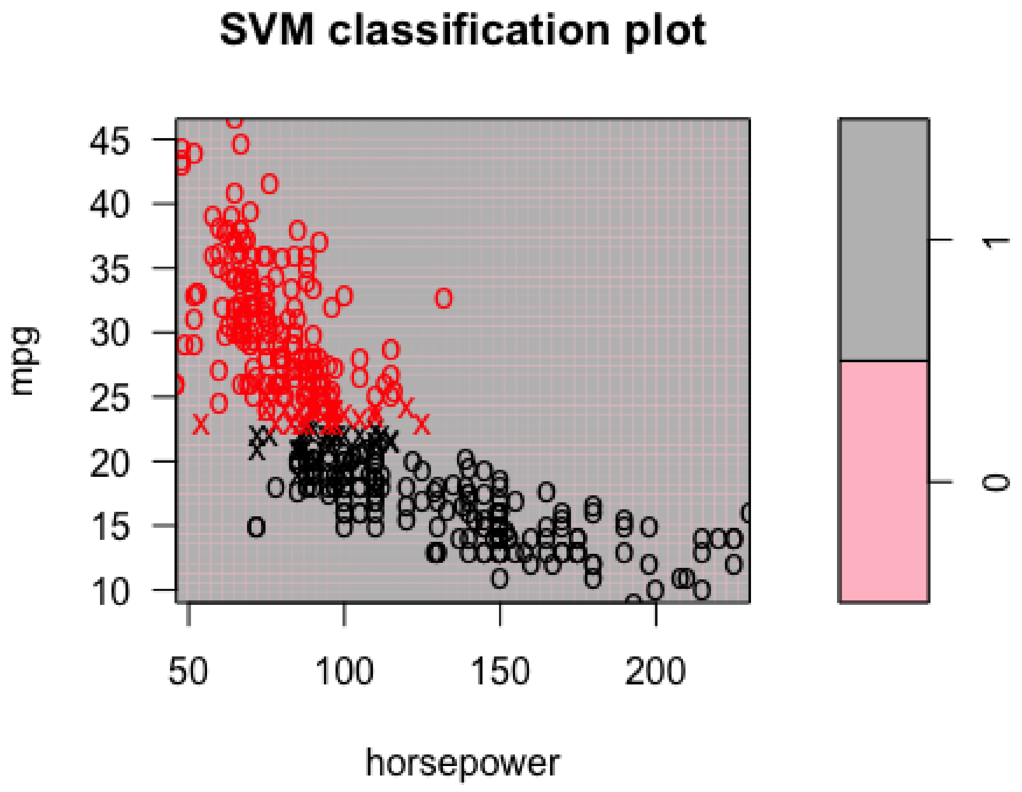


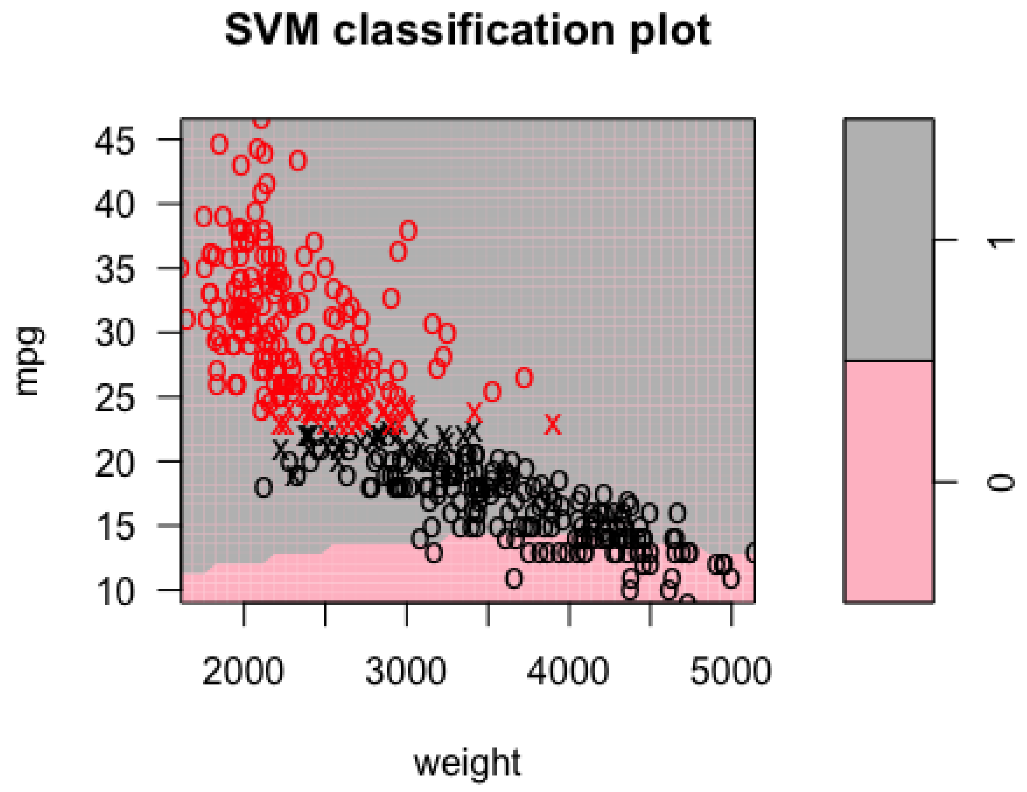
Now let us make some plots in support of assertions made for SVM with Radial Kernel

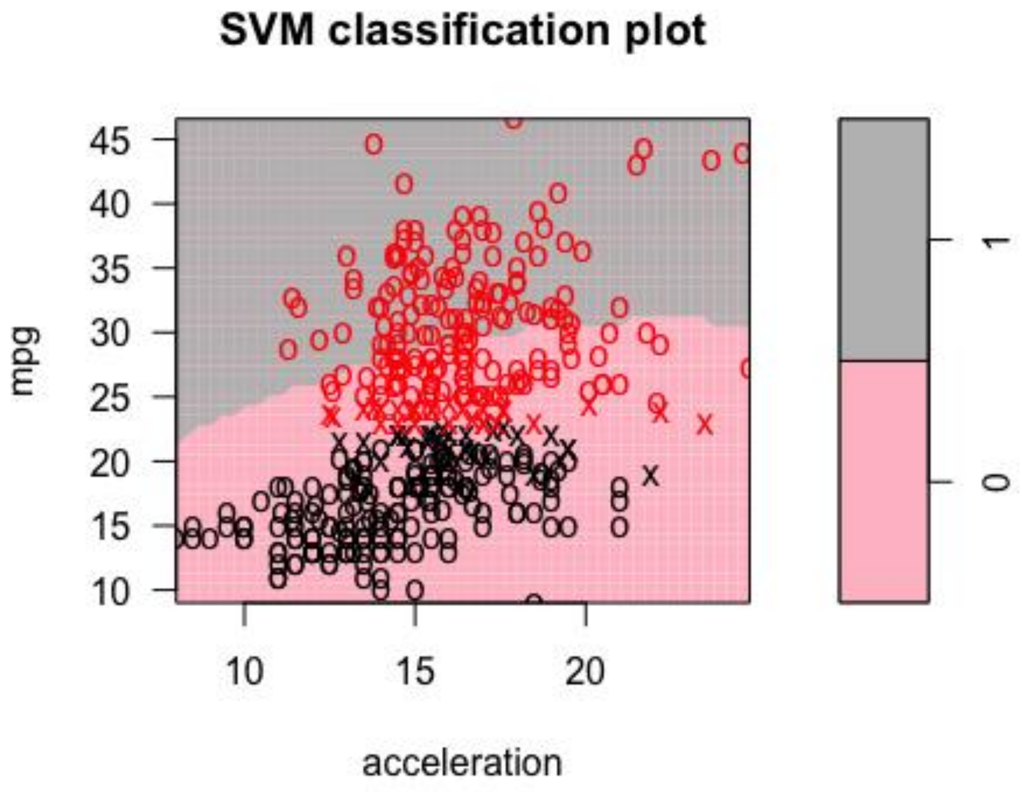
```
plotpairs(svm_radial)
```

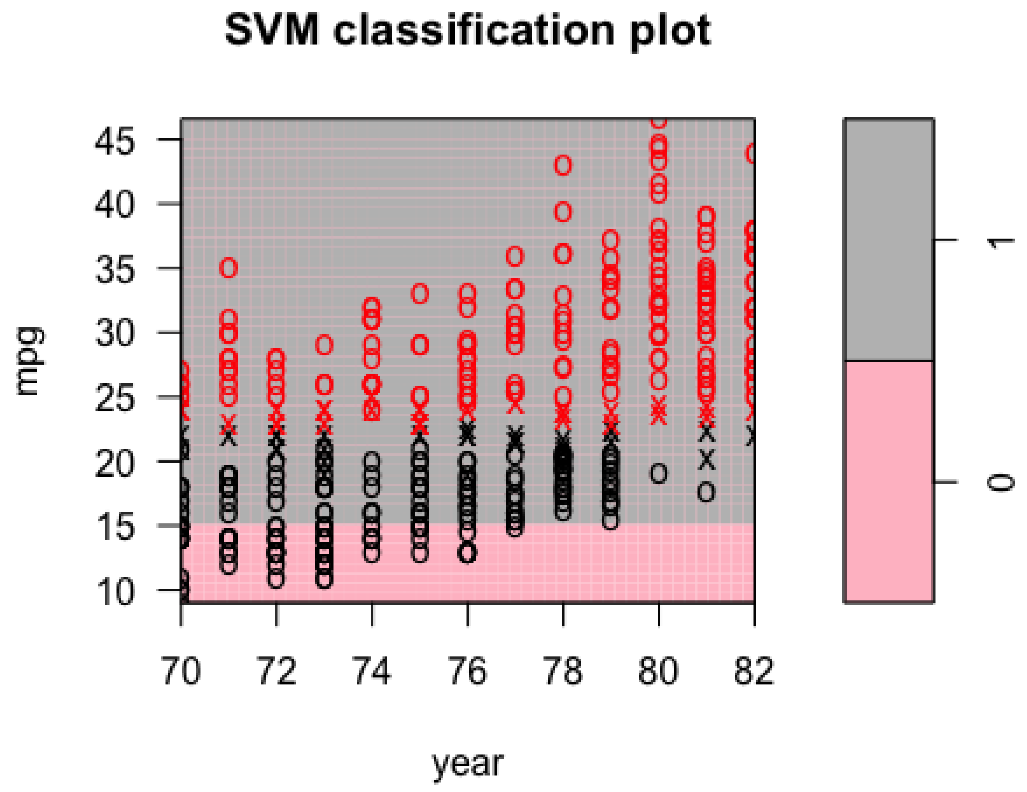

SVM classification plot
mpg
cylinders
45
40
35
30
25
20
15
10
3
4
5
6
7
8
1
0

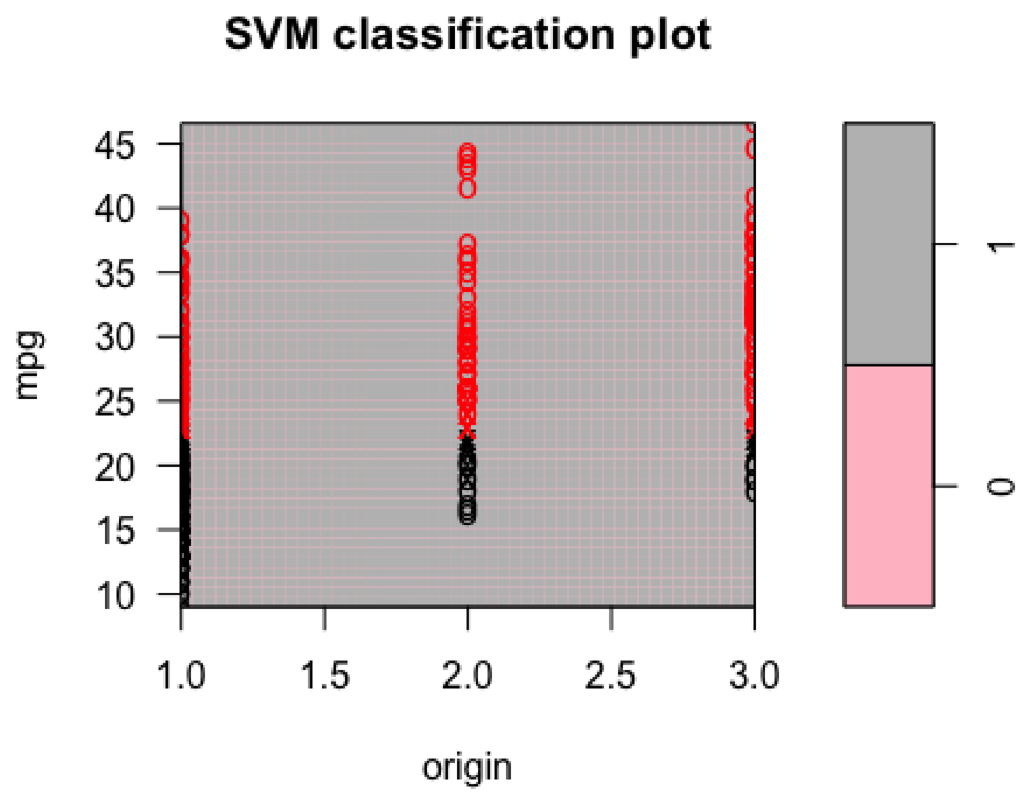

SVM classification plot
mpg
displacement
45
40
35
30
25
20
15
10
100
200
300
400
1
0


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


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

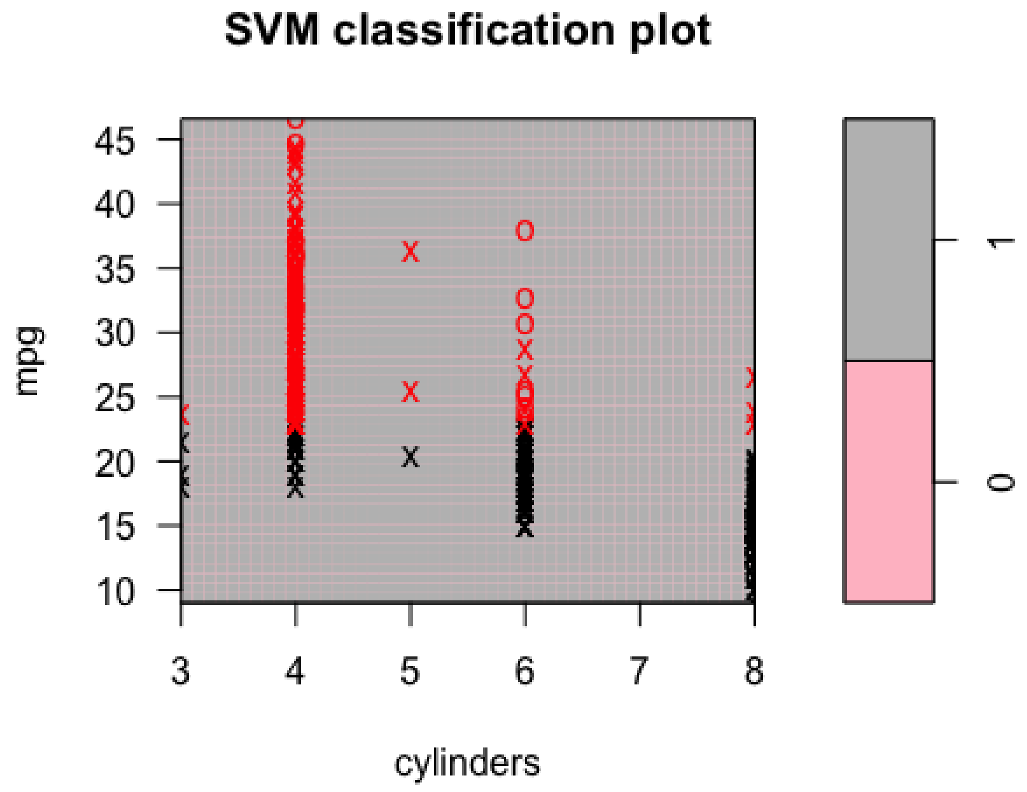

SVM classification plot
mpg
acceleration
45
40
35
30
25
20
15
10
10
15
20
1
0

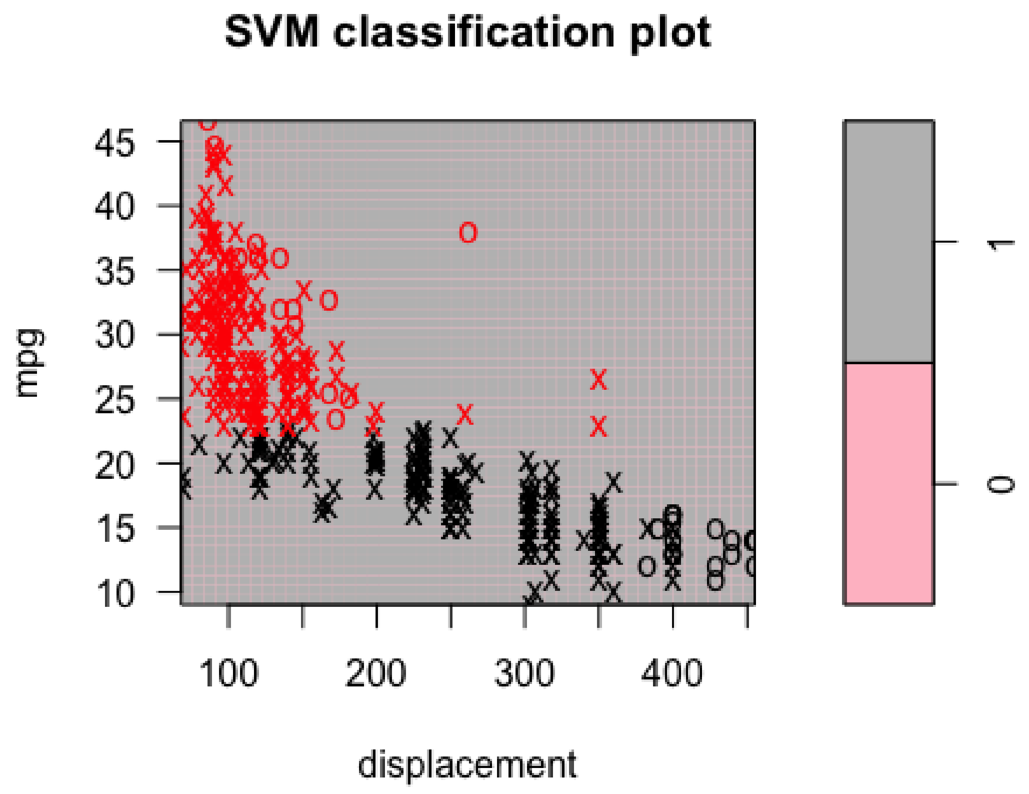

SVM classification plot
mpg
45
40
35
30
25
20
15
10
70
72
74
76
78
80
82
year
1
0



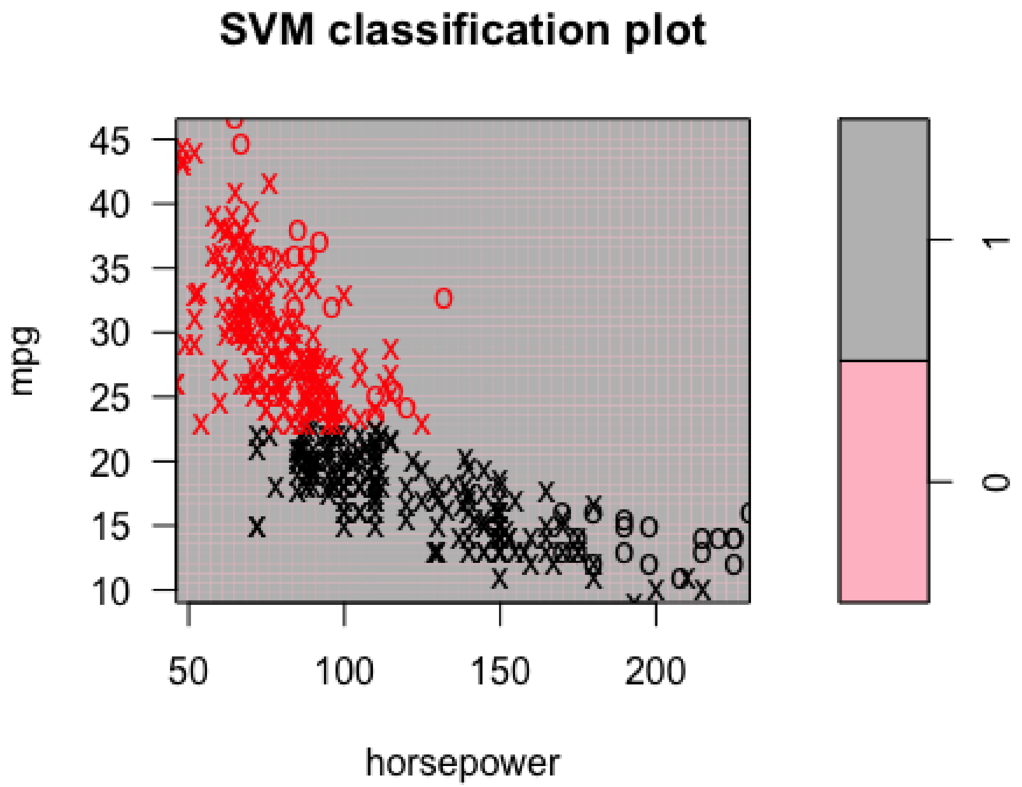
Now let us make some plots in support of assertions made for SVM with Polynomial Kernel

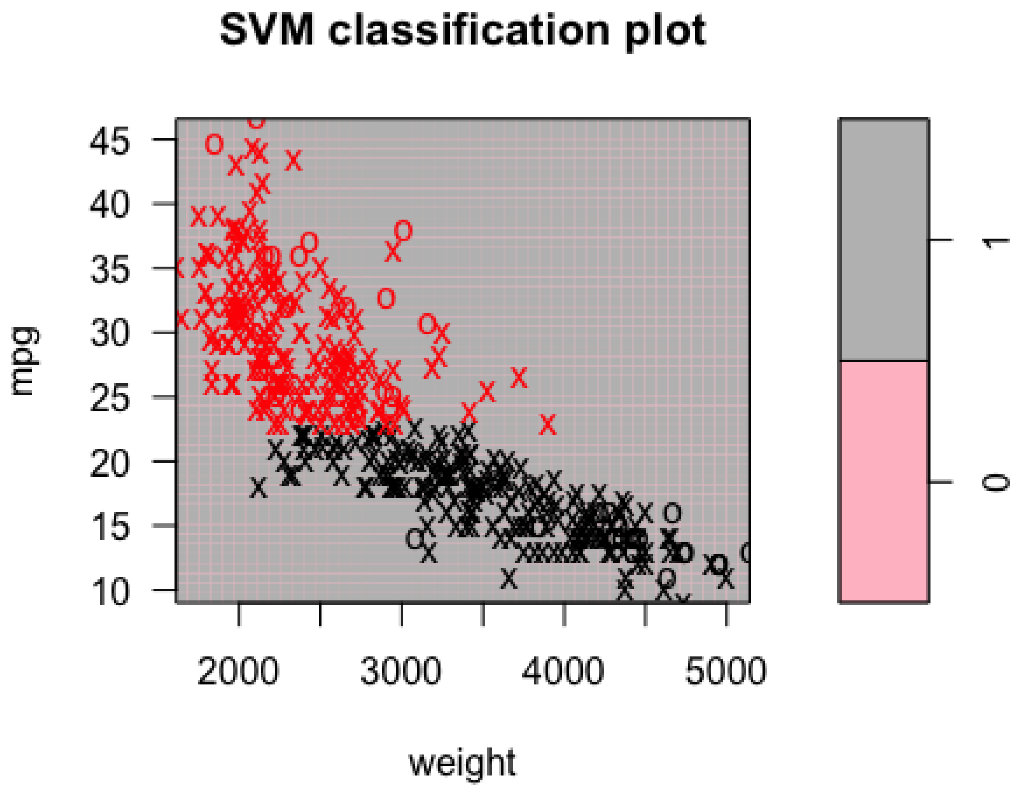
```
plotpairs(svm_poly)
```

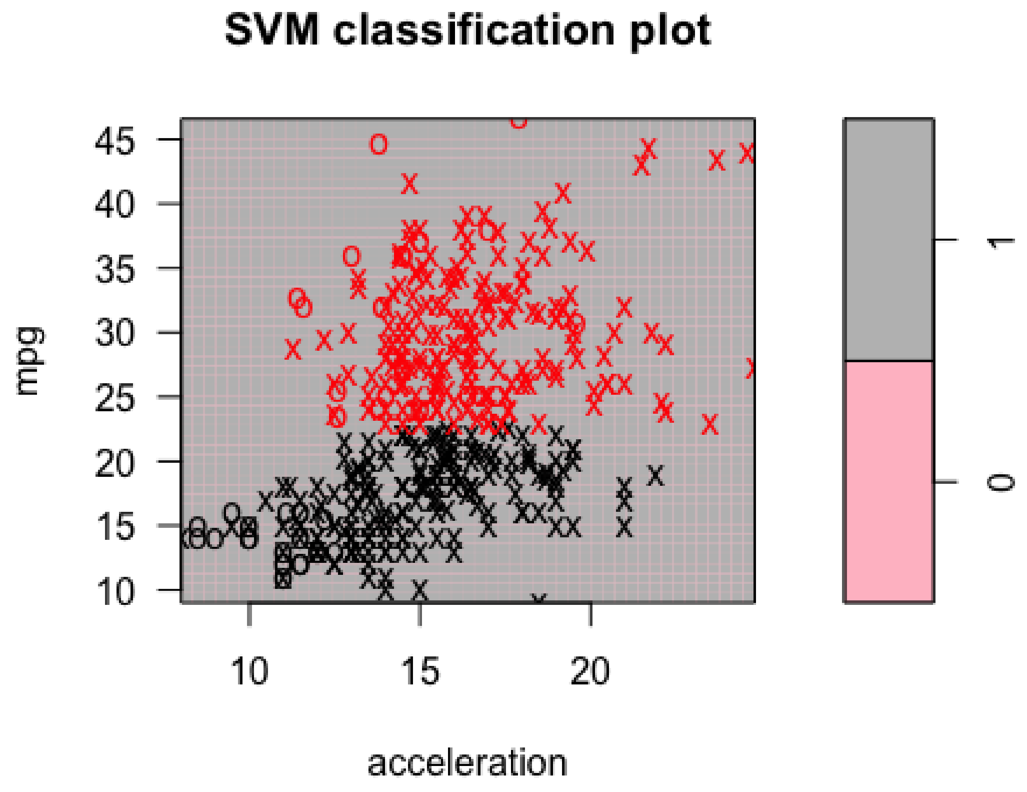

SVM classification plot
mpg
45
40
35
30
25
20
15
10
3
4
5
6
7
8
cylinders
1
0



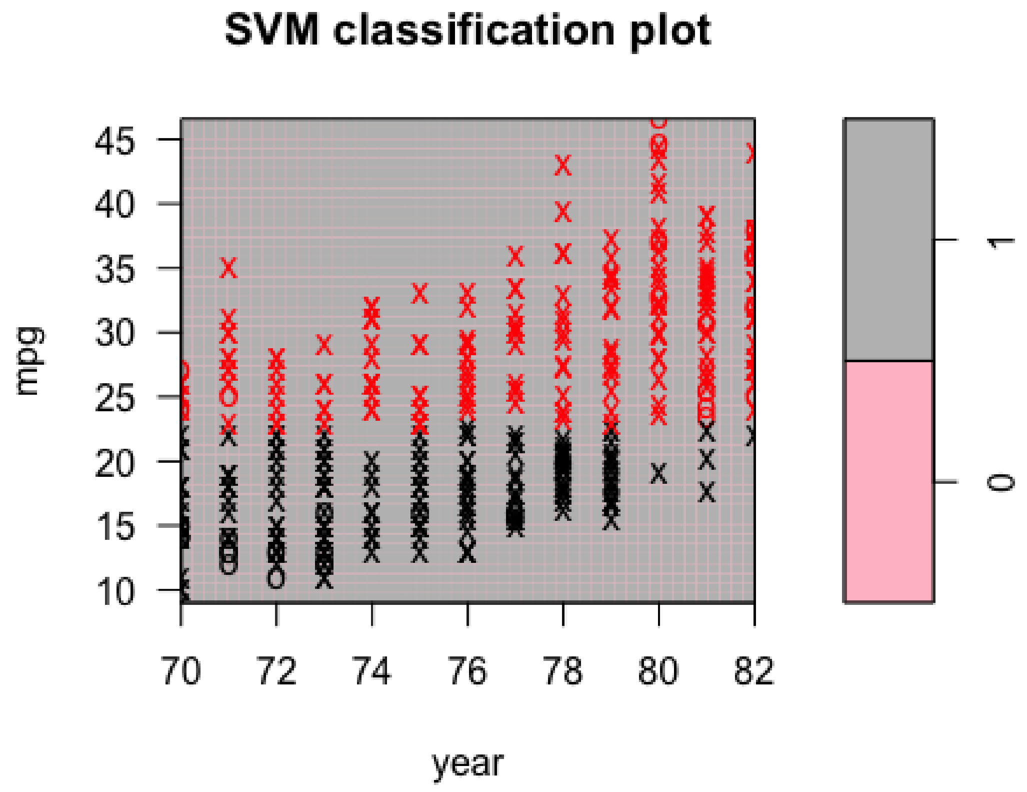
SVM classification plot
mpg
displacement
45
40
35
30
25
20
15
10
100
200
300
400
1
0

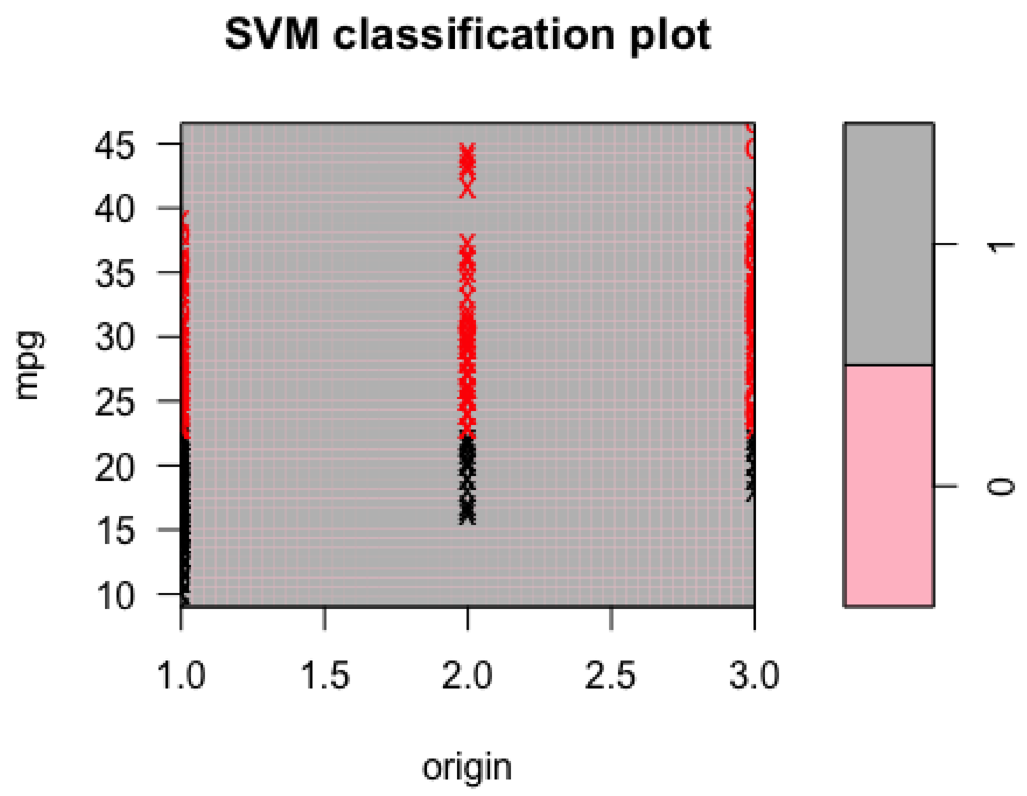

SVM classification plot
mpg
horsepower
45
40
35
30
25
20
15
10
50
100
150
200
1
0


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



Question 3.

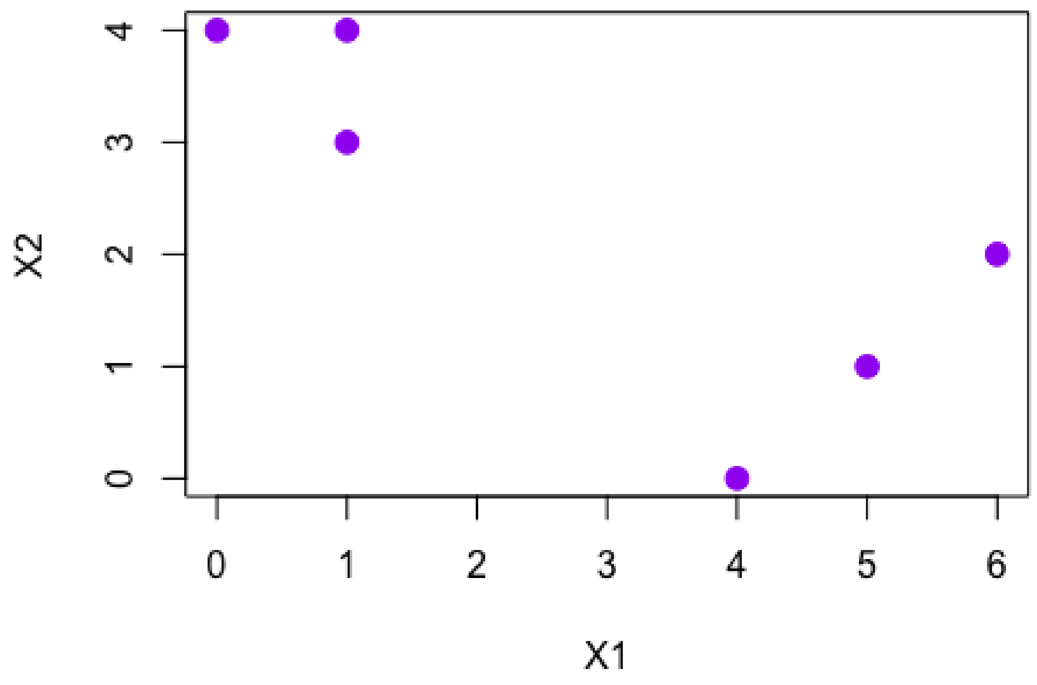
In this problem, you will perform K-means clustering manually, with K = 2, on a small example with n = 6 observations and p = 2 features. The observations are as follows.

```
rm(list=ls())
X1 <- c(1,1,0,5,6,4)
X2 <- c(4,3,4,1,2,0)
fdata <- cbind(X1, X2)
fdata

##      X1 X2
## [1,]  1  4
## [2,]  1  3
## [3,]  0  4
## [4,]  5  1
## [5,]  6  2
## [6,]  4  0
```

Question 3.(a)

Plot the observations.

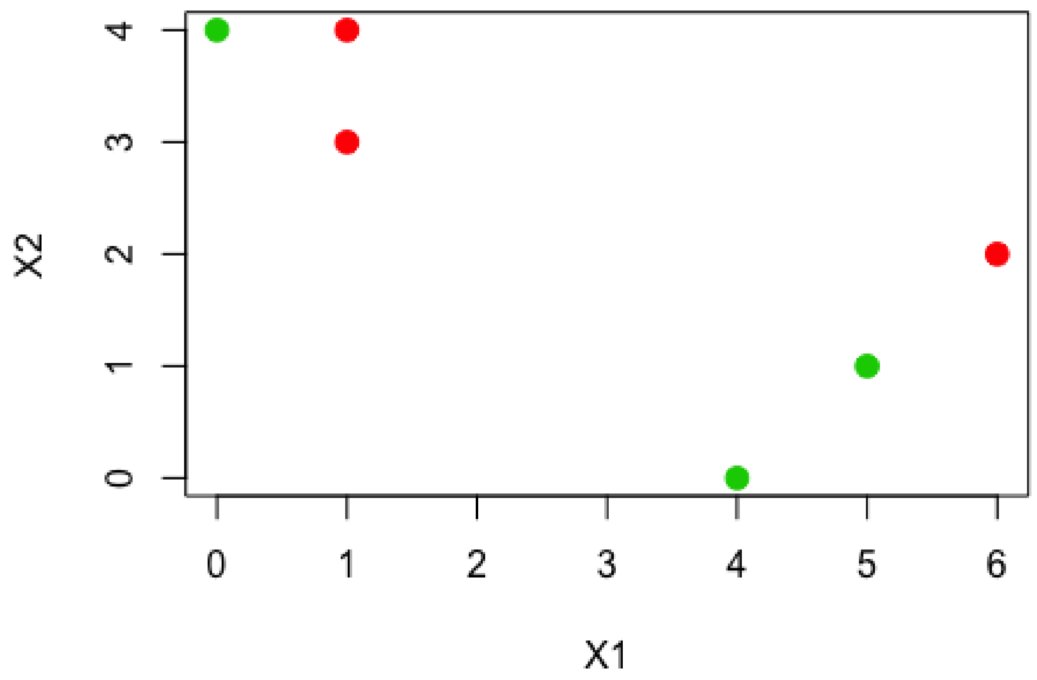
```
r  plot(fdata, pch=20, cex=2, col="purple")
```



Question 3.(b)

Randomly assign a cluster label to each observation. You can use the sample() command in R to do this. Report the cluster labels for each observation.

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

## [1] 1 1 2 2 1 2

plot(fdata, col = (labels + 1), pch = 20, cex = 2)
```

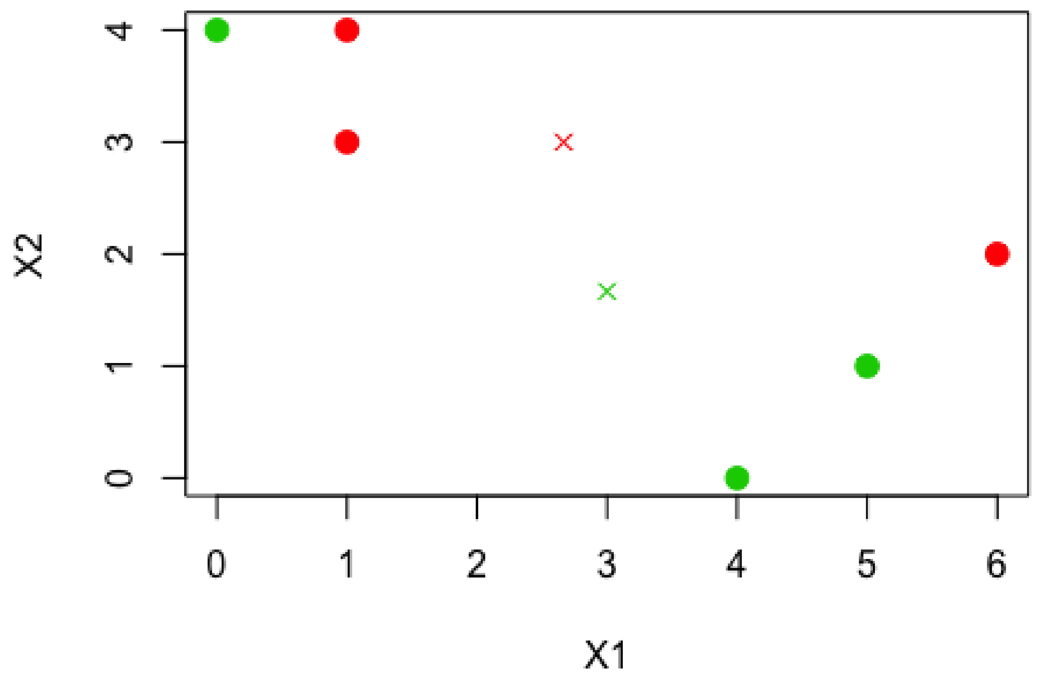


Question 3.(c)

Compute the centroid for each cluster.

Computations:

```
set.seed(1)
centroid1 <- c(mean(fdata[labels == 1, 1]), mean(fdata[labels == 1, 2]))
centroid2 <- c(mean(fdata[labels == 2, 1]), mean(fdata[labels == 2, 2]))
plot(fdata, 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)
```



We can compute the **centroid for green cluster** as shown below:

```
X11 <- (1/3)*(0+4+5)
X12 <- (1/3)*(4+0+1)
cent_green_clust <- cbind(X11, X12)
cent_green_clust

##      X11      X12
## [1,]   3 1.666667
```

We can compute the **centroid for red cluster** as shown below:

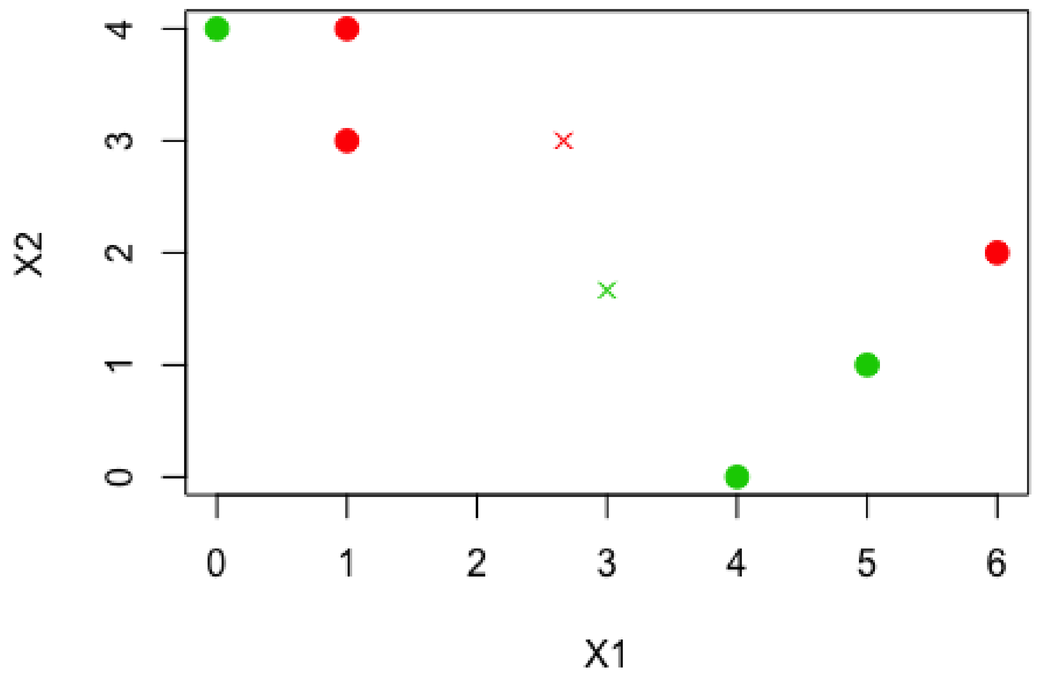
```
X21 <- (1/3)*(1+1+6)
X22 <- (1/3)*(2+4+3)
cent_red_clust <- cbind(X21, X22)
cent_red_clust

##           X21 X22
## [1,] 2.666667   3
```

Question 3.(d)

Assign each observation to the centroid to which it is closest, in terms of Euclidean distance. Report the cluster labels for each observation.

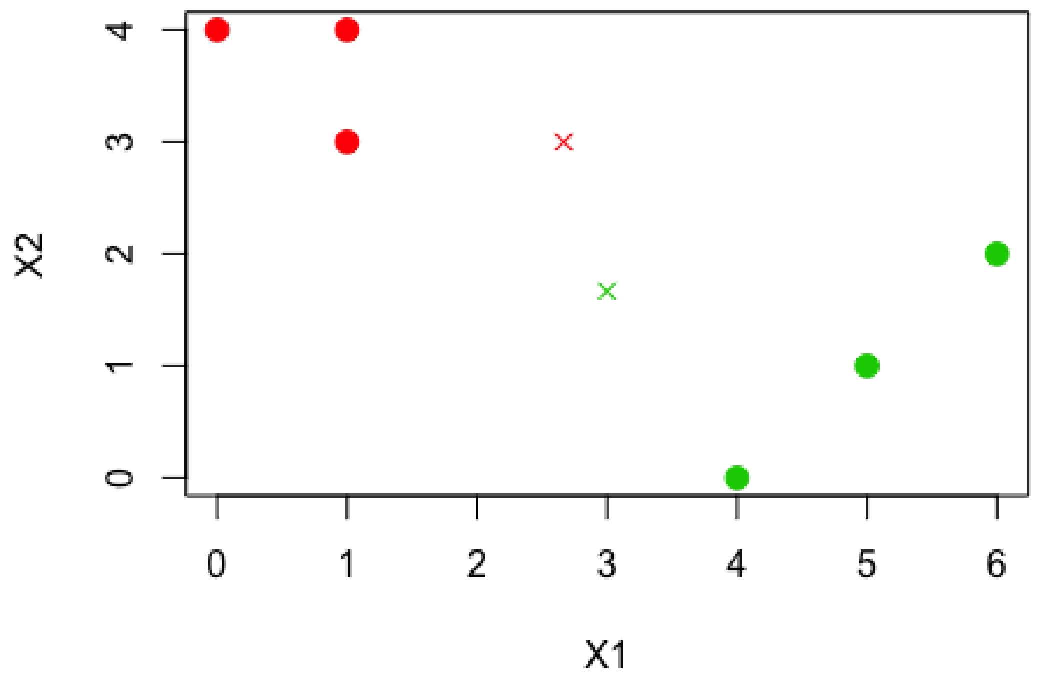
Solution: Let us have a look at the plot (before assigning to the closest centroid w.r.to Euclidean distance)

```r
plot(fdata, 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)
```



Now let us assign each observation to the closest centroid [w.r.to Euclidean distance]

```r
labels <- c(1, 1, 1, 2, 2, 2)
plot(fdata, 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)
```



Question 3.(e)

Repeat (c) and (d) until the answers obtained stop changing.

Answer:

We can compute the **centroid for green cluster** as shown below:

```
X11 <- (1/3)*(4+5+6)
X12 <- (1/3)*(0+1+2)
cent_green_clust <- cbind(X11, X12)
cent_green_clust

##      X11 X12
## [1,]   5   1
```

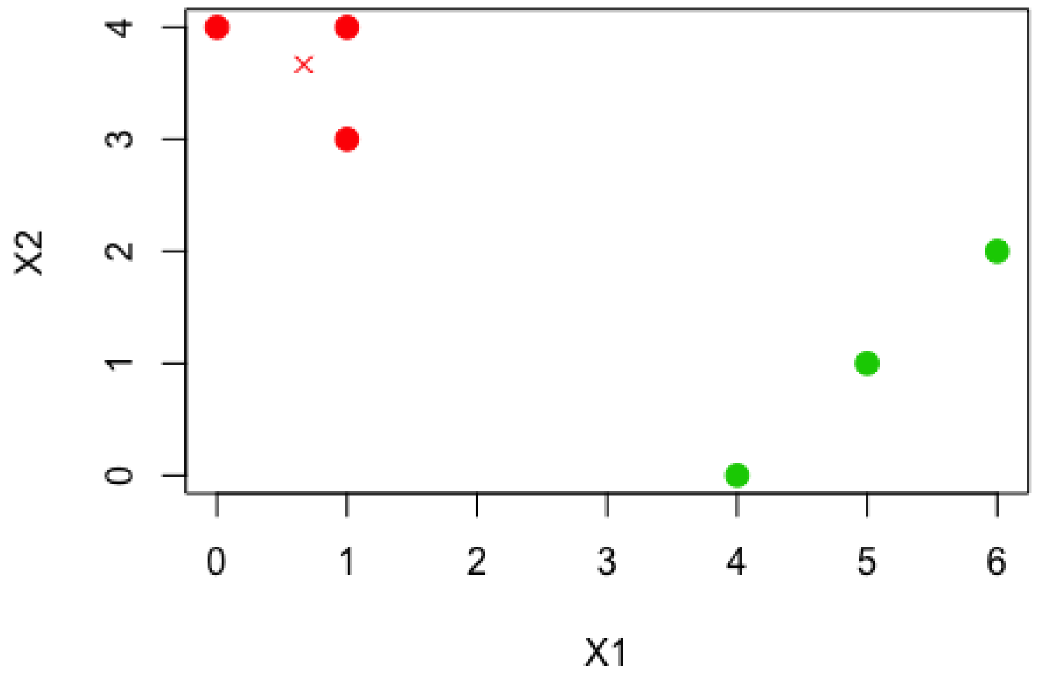
We can compute the **centroid for red cluster** as shown below:

```
X21 <- (1/3)*(0+1+1)
X22 <- (1/3)*(3+4+4)
cent_red_clust <- cbind(X21, X22)
cent_red_clust
```

```
##             X21      X22
## [1,] 0.6666667 3.666667
```

Let us plot when the answers stop changing

```
centroid1 <- c(mean(fdata[labels == 1, 1]), mean(fdata[labels == 1, 2]))
centroid2 <- c(mean(fdata[labels == 2, 1]), mean(fdata[labels == 2, 2]))
plot(fdata, 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)
```



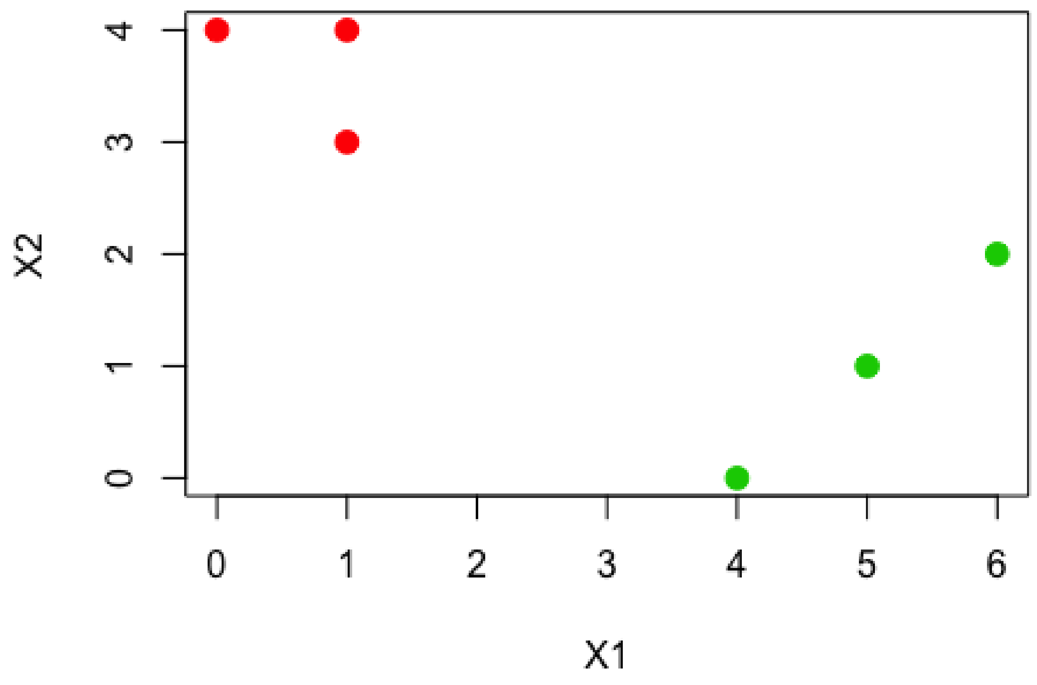
Observation: When we assign each observation to the centroid to which it is the closest, then nothing changes, so the algorithm stops at this step.

Question 3.(f)

In your plot from (a), color the observations according to the cluster labels obtained.

Answer:

```
plot(fdata, col=(labels + 1), pch = 20, cex = 2)
```



Question 4.

In this problem, you will generate simulated data, and then perform PCA and K-means clustering on the data.

Question 4.(a)

Generate a simulated data set with 20 observations in each of three classes (i.e. 60 observations total), and 50 variables. Hint: There are a number of functions in R that you can use to generate data. One example is the rnorm() function; runif() is another option. Be sure to add a mean shift to the observations in each class so that there are three distinct classes.

Answer:

```
set.seed(2)
x <- matrix(rnorm(20 * 3 * 50, mean = 0, sd = 0.001), ncol = 50)
x[1:20, 2] <- 1
x[21:40, 1] <- 2
x[21:40, 2] <- 2
x[41:60, 1] <- 1
true_labels <- c(rep(1, 20), rep(2, 20), rep(3, 20))
head(x)
```

```
##               [,1] [,2]          [,3]          [,4]          [,5]
## [1,] -8.969145e-04    1  0.0012411998 -0.0010304037 -8.178004e-05
## [2,]  1.848492e-04    1  0.0001388584 -0.0012619293  2.888418e-03
## [3,]  1.587845e-03    1  0.0017106316  0.0003921846  1.172825e-03
## [4,] -1.130376e-03    1 -0.0004306410 -0.0011314383  5.496432e-04
## [5,] -8.025176e-05    1 -0.0010442296  0.0005441445  7.791045e-04
## [6,]  1.324203e-04    1  0.0005375795  0.0011766089 -5.651313e-04
##               [,6]          [,7]          [,8]          [,9]         [,10]
## [1,] -0.0003181198  0.0015553636  0.0007395299 -1.071887e-03  9.210408e-05
## [2,] -0.0003154903  0.0014232581 -0.0012555745  1.373661e-04 -6.383213e-05
## [3,]  0.0008843223 -0.0003587572  0.0014119511 -8.949973e-04 -8.360449e-04
## [4,] -0.0018854213 -0.0004977432  0.0007334870 -7.856582e-05  1.728745e-03
## [5,]  0.0007321793 -0.0018849880  0.0012817215  1.595836e-03 -9.670830e-04
## [6,]  0.0007905447 -0.0011396263  0.0008988005 -1.204026e-03  5.663589e-04
##              [,11]         [,12]         [,13]         [,14]         [,15]
## [1,] -0.0011150718 -0.0008851107  0.0001133897 -1.150544e-05  0.0019891898
## [2,] -0.0001142184  0.0019720096  0.0003749700 -1.149609e-03  0.0009834269
## [3,] -0.0008946214  0.0017217516  0.0014645065  1.163992e-04  0.0010103531
## [4,] -0.0006540889 -0.0009608869 -0.0012422331  3.473042e-04 -0.0019333986
## [5,]  0.0011787163  0.0011570388 -0.0007026556  5.469328e-04  0.0004879395
## [6,]  0.0009515165 -0.0005944494  0.0008672734 -5.180769e-04 -0.0004625082
##              [,16]         [,17]         [,18]         [,19]         [,20]
## [1,]  2.297330e-04 -0.0004174203 -0.0002941476  0.0013217584 -0.0008399767
## [2,]  1.092479e-03 -0.0005495990  0.0008561112  0.0025652873  0.0005287176
## [3,]  2.880342e-06  0.0002639009 -0.0005727162  0.0013697219  0.0008203881
## [4,] -1.325724e-04 -0.0008402018 -0.0001708121  0.0011550379  0.0002353379
## [5,] -6.789119e-05 -0.0009785698 -0.0004060722  0.0001299466  0.0013115394
## [6,]  3.058845e-04 -0.0008425745  0.0013306101 -0.0014847332 -0.0040752252
##              [,21]         [,22]        [,23]         [,24]         [,25]
## [1,]  0.0004685524  0.0004966514 7.053075e-05  0.0012835423  0.0006669912
## [2,] -0.0013005815  0.0009085848 6.266342e-04 -0.0009730500  0.0003223934
## [3,] -0.0002526671 -0.0005196211 6.184692e-04 -0.0003180604 -0.0000856340
## [4,]  0.0009656920  0.0020058045 5.619092e-04 -0.0020033926 -0.0012136848
## [5,] -0.0002110945  0.0009073413 3.493727e-04 -0.0012418042 -0.0004929722
## [6,] -0.0013920929 -0.0013993379 6.846007e-04 -0.0019046738  0.0009443771
##              [,26]         [,27]         [,28]         [,29]         [,30]
## [1,] -8.472442e-04 -0.0004999697 -0.0023931058 -7.315099e-04 -0.0005453050
## [2,]  7.042928e-05  0.0005881623  0.0001886540 -1.918124e-03  0.0001049698
## [3,]  1.581195e-03  0.0005812152  0.0006670037 -1.654804e-05 -0.0017034898
## [4,]  2.059799e-05  0.0006293431 -0.0005340189  3.034383e-04  0.0008765051
## [5,] -1.233705e-04  0.0008116190 -0.0022403304  8.031262e-04 -0.0008858937
## [6,] -1.257379e-03  0.0014636101 -0.0002206335 -1.578393e-04  0.0001175607
##              [,31]         [,32]         [,33]         [,34]         [,35]
## [1,]  0.0020610545  3.405397e-05  0.0002035076  0.0023465549 -0.0003668638
## [2,] -0.0005347649  1.744328e-03  0.0008992397  0.0000392687 -0.0013691004
## [3,]  0.0008995549 -7.896841e-06 -0.0004640080 -0.0015911692 -0.0005324749
## [4,]  0.0016049680  8.833709e-04 -0.0001801725 -0.0007163666  0.0002917585
## [5,] -0.0018106745  1.893180e-04  0.0002918624  0.0003897614 -0.0005152670
## [6,] -0.0018428077  1.437093e-03  0.0004484821 -0.0002873721 -0.0013162111
##              [,36]         [,37]         [,38]         [,39]         [,40]
## [1,]  0.0004687672  0.0002820207  0.0014324294 -0.0004130050  0.0002012512
## [2,]  0.0012329601  0.0004324943 -0.0003559032  0.0008173732 -0.0018133509
## [3,] -0.0015416580  0.0004192031  0.0003484009 -0.0011617299 -0.0009541452
## [4,]  0.0019894919  0.0003714494  0.0005019369 -0.0005539248 -0.0013232117
## [5,]  0.0002449807  0.0005524890  0.0002254110 -0.0008701356 -0.0017001044
## [6,]  0.0007913569 -0.0007647828  0.0004710608  0.0012343010 -0.0011348630
##              [,41]         [,42]         [,43]         [,44]         [,45]
## [1,] -1.237421e-03 -0.0005206907  0.0013949181  8.056052e-04  0.0011360469
## [2,]  4.970946e-04  0.0006096293 -0.0001051587  1.431615e-05 -0.0003964177
## [3,]  1.152604e-03  0.0009179604 -0.0010472475 -7.737464e-04  0.0004610868
## [4,] -2.765654e-04  0.0006481462  0.0003900601 -3.642246e-04  0.0002723962
## [5,] -1.328038e-03  0.0015945181  0.0007290500  3.085749e-04  0.0009437149
## [6,]  4.724361e-05 -0.0003235029  0.0013952141  1.008076e-03  0.0007767642
##              [,46]         [,47]         [,48]         [,49]         [,50]
## [1,] -0.0013534991 -0.0003870643 -2.983695e-04  0.0012979893  0.0009731223
## [2,]  0.0003408542  0.0005775024  5.469815e-04 -0.0006041793  0.0017952227
## [3,]  0.0002087469 -0.0001407346  3.947386e-04 -0.0016079685  0.0002564498
## [4,]  0.0013237901 -0.0002609810 -4.697918e-05  0.0002487063  0.0013531616
## [5,] -0.0017835545 -0.0016734887  7.000310e-04 -0.0001665372  0.0016209731
## [6,] -0.0008143231 -0.0002048545  1.402526e-03 -0.0013887137 -0.0024293425
```

```
true_labels
```

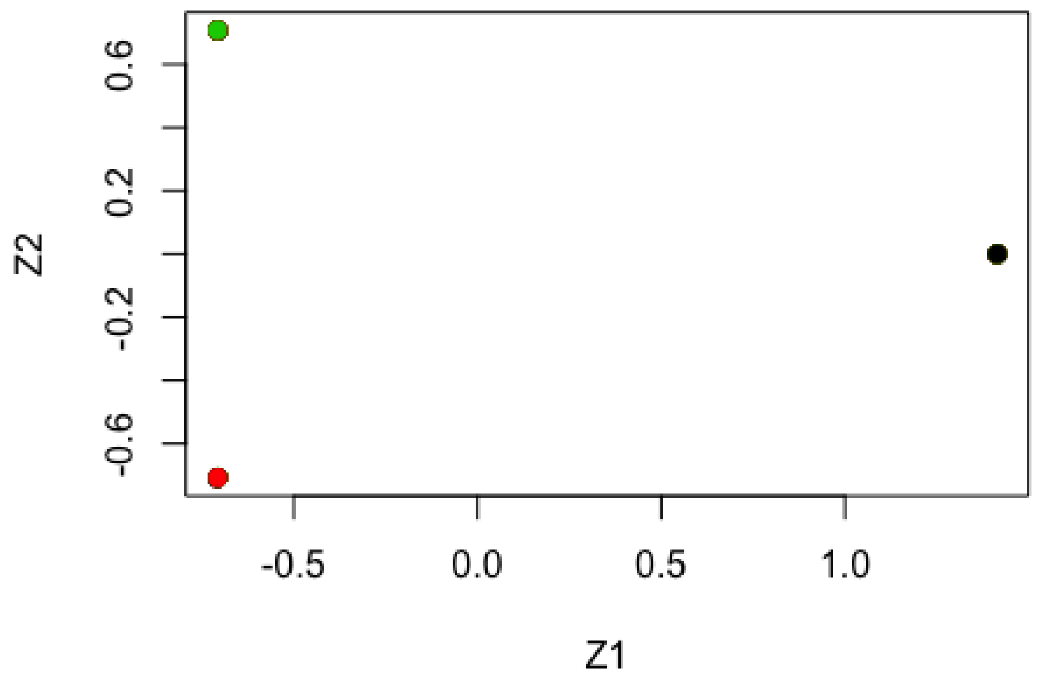
```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [36] 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

Question 4.(b)

Perform PCA on the 60 observations and plot the first two principal component score vectors. Use a different color to indicate the observations in each of the three classes. If the three classes appear separated in this plot, then continue on to part (c). If not, then return to part (a) and modify the simulation so that there is greater separation between the three classes. Do not continue to part (c) until the three classes show at least some separation in the first two principal component score vectors.

Answer:

```r
pr_compx <- prcomp(x)
plot(pr_compx$x[, 1:2], col = 1:3, xlab = "Z1", ylab = "Z2", pch = 19)
```



Observation:

1. We can see that three classes are uniquely identified based on color - red, green, blue.
2. All three classes looks to be well-separated, thus we can move to part 4.(c) on completion.

Question 4.(c)

Perform K-means clustering of the observations with K = 3. How well do the clusters that you obtained in K-means clustering compare to the true class labels? Hint: You can use the table() function in R to compare the true class labels to the class labels obtained by clustering. Be careful how you interpret the results: K-means clustering will arbitrarily number the clusters, so you cannot simply check whether the true class labels and clustering labels are the same.

Answer:

```
km_three <- kmeans(x, 3, nstart = 20)
table(true_labels, km_three$cluster)

##
## true_labels  1  2  3
##           1 20  0  0
##            2  0 20  0
##            3  0  0 20
```

Observation:

1. We can say that the clusters we just obtained via K-means Clustering perform pretty well.
2. We can see that there are 3 clusters.
3. Thus, we can conclude that, **the observations are perfectly clustered**.

Question 4.(d)

Perform K-means clustering with K = 2. Describe your results.

Answer:

```
km_two <- kmeans(x, 2, nstart = 20)
table(true_labels, km_two$cluster)

##
## true_labels  1  2
##            1 20  0
##            2  0 20
##            3 20  0
```

Observation:

1. We can see that there are two clusters.

2. On performing K-means clustering with k=2, we can find that all the observations of one of the three clusters (previously seen) is now absorbed in one of the two clusters.

3. Further, the result shows that the second cluster has absorbed most of the observations.

Question 4.(e)

Now perform K-means clustering with K = 4, and describe your results.

Answer:

```
km_four <- kmeans(x, 4, nstart = 20)
table(true_labels, km_four$cluster)

##
## true_labels  1  2  3  4
##            1  9  0  0 11
##            2  0 20  0  0
##            3  0  0 20  0
```

Observation:

1. We can see that there are four clusters
2. The first cluster and the second cluster is split in-order to facilitate the presence of observations in the third and the fourth cluster.
3. Looks like, first cluster is split to form cluster 1 and cluster 4
4. Further, second cluster is split to form cluster 2 and cluster 3.

Question 4.(f)

Now perform K-means clustering with K = 3 on the first two principal component score vectors, rather than on the raw data. That is, perform K-means clustering on the 60 ?? 2 matrix of which the first column is the first principal component score vector, and the second column is the second principal component score vector. Comment on the results.

Answer:

```
km_pc <- kmeans(pr_compx$x[, 1:2], 3, nstart = 20)
table(true_labels, km_pc$cluster)

##
## true_labels  1  2  3
##           1  0  0 20
##           2  0 20  0
##           3 20  0  0
```

Observation:

1. We can say that the clusters we just obtained via K-means Clustering on the first two PCs (Principal Components) perform pretty well.
2. We can see that there are 3 clusters.
3. Thus, we can conclude that, **the observations are perfectly clustered**.
4. Unlike the previous cases, the first observation is mostly absorbed by the last cluster.
5. On the other hand, the last observation is mostly absorbed by the first cluster.

Question 4.(g)

Using the scale() function, perform K-means clustering with K = 3 on the data after scaling each variable to have standard deviation one. How do these results compare to those obtained in (b)? Explain.

Answer:

```
km_scale <- kmeans(scale(x), 3, nstart = 20)
table(true_labels, km_scale$cluster)

##
## true_labels  1  2  3
##           1  8  2 10
##           2  0 19  1
##           3 11  1  8
```

Observation:

1. When compared with the results we obtained from 4.(b), we can say that we got very poor results after scaling each variable to have standard deviation one. than with unscaled data
2. This is basically because scaling affects the distance between the observations.
3. Thus we can conclude that scaled data has yielded worse results when compared with the unscaled data.