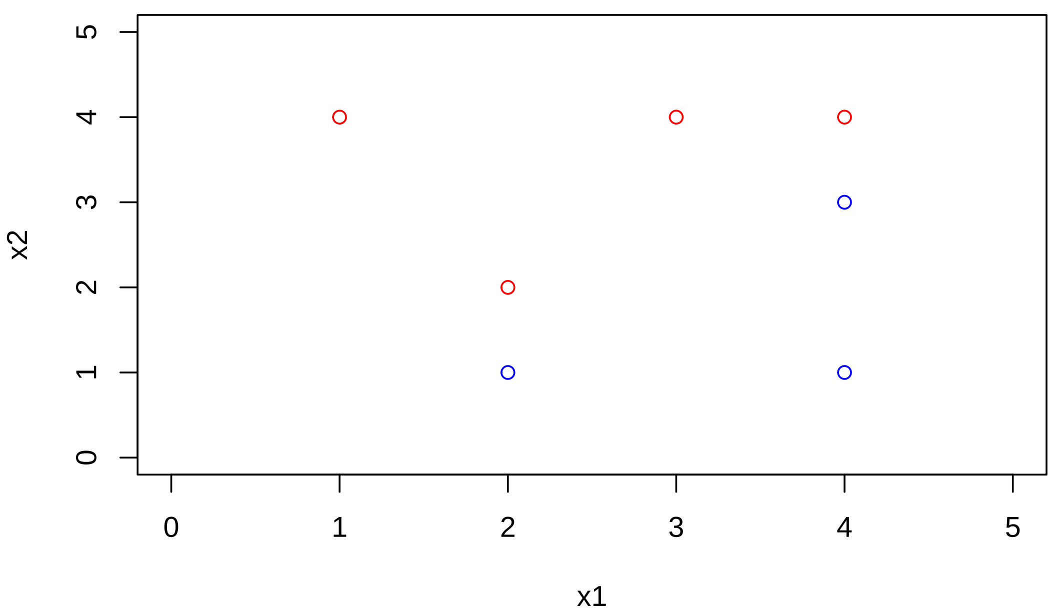
Chapter 9

Tong Sun

3/3/2022

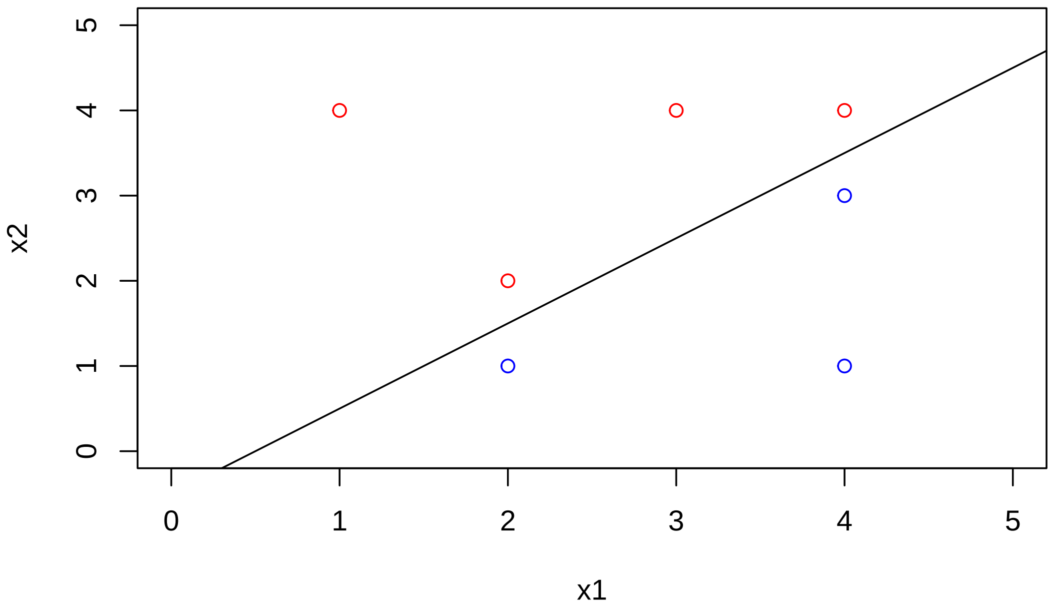
9.3

Here we explore the maximal margin classifier on a toy data set. ##(a) We are given n = 7 observations in p = 2 dimensions. For each observation, there is an associated class label. Sketch the observations.



##(b) Sketch the optimal separating hyperplane, and provide the equation for this hyperplane (of the form (9.1)).

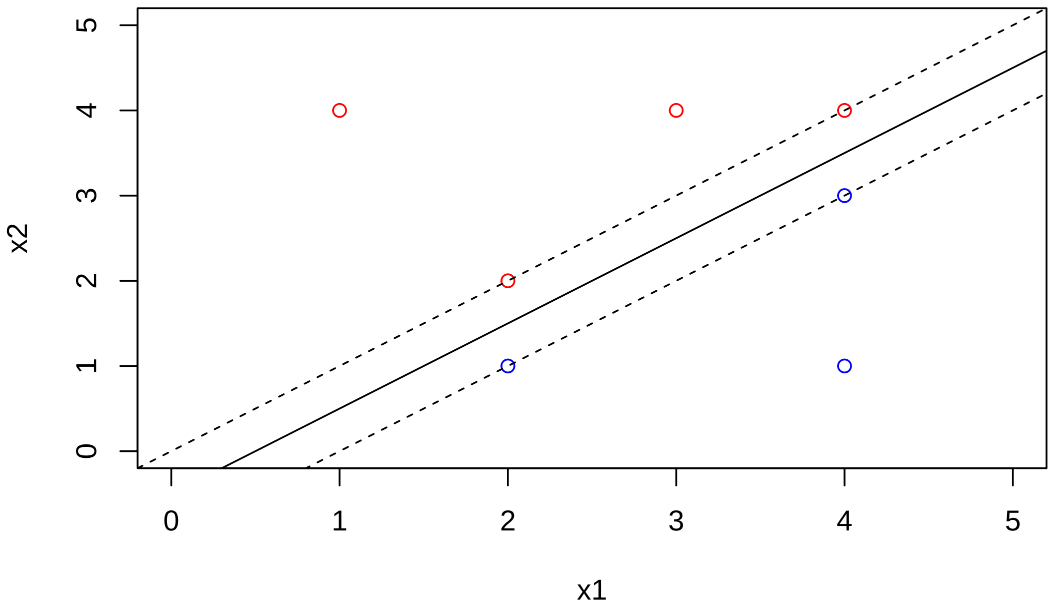
From the plot, the optimal separating hyperplane has to be between the group1 including points (2,1) and (2,2), and between the group2 including points (4,3) and (4,4). So with averaging the y-values of two group of points, the line is the one that passes through the points (2,1.5) and (4,3.5) which equation is $X_1 - X_2 - 0.5 = 0$.



##(c) Describe the classification rule for the maximal margin classifier. It should be something along the lines of "Classify to Red if $\beta_0 + \beta_1 X_1 + \beta_2 X_2 > 0$,and classify to Blue otherwise." Provide the values for β_0, β_1, and β_2.

The classification rule is that classifying to Red if $X_1 - X_2 - 0.5 < 0$, and classify to Blue otherwise.

##(d) On your sketch, indicate the margin for the maximal margin hyperplane.



The margin is here equal to 0.25.

##(e) Indicate the support vectors for the maximal margin classifier.

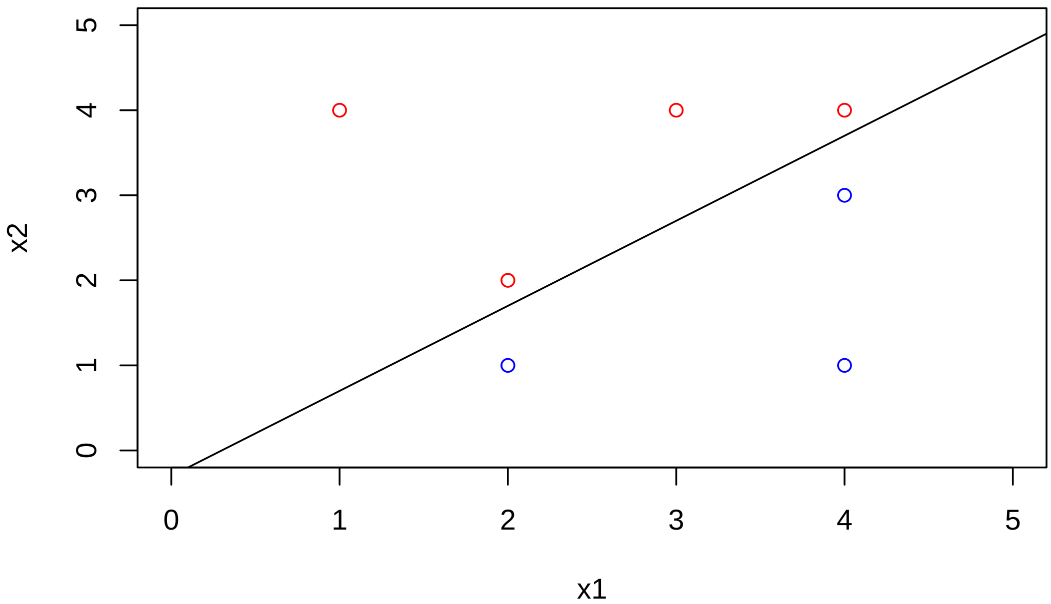
The support vectors are the points (2,1), (2,2), (4,3) and (4,4).

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

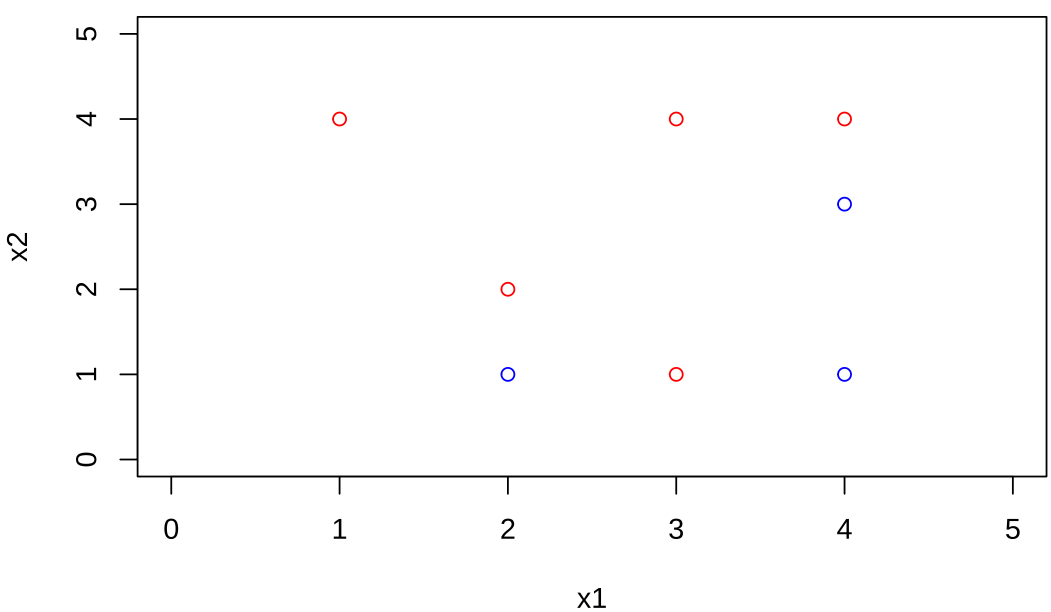
By examining the plot, it is clear that if we moved the observation(4,1), we would not change the maximal margin hyperplane as it is not a support vector.

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

For example, the hyperplane which equation is $X_1 - X_2 - 0.3 = 0$ is not the optimal separating hyperplane.



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

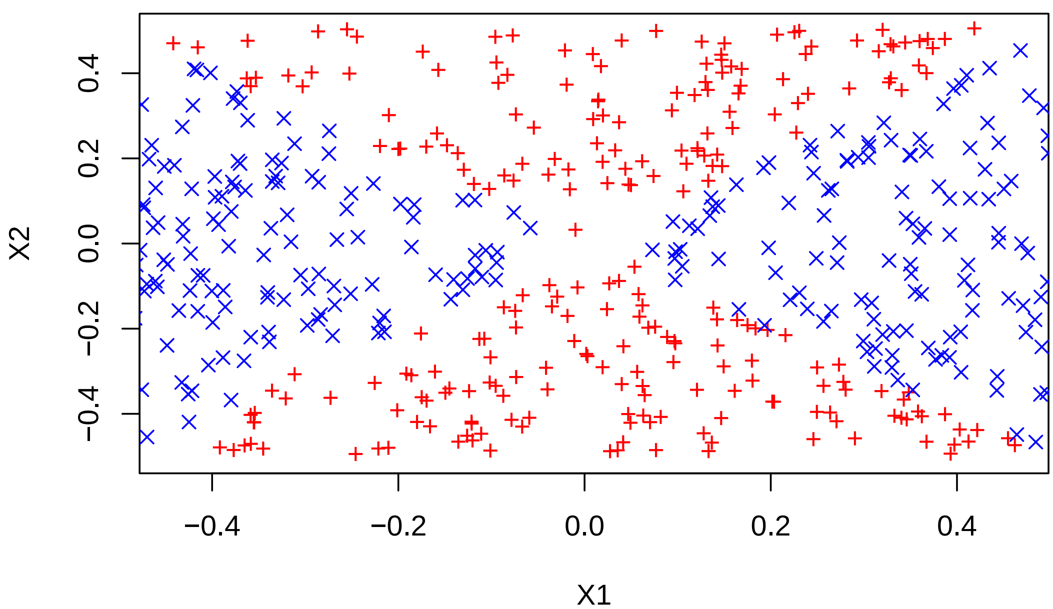


When the red point(3,1) is added to th plot, the two classes are obviously not separable by a hyperplane anymore.

9.5

We have seen that we can fit an SVM with a non-linear kernel in order to perform classification using a non-linear decision boundary. We will now see that we can also obtain a non-linear decision boundary by performing logistic regression using non-linear transformations of the features. ##(a) Generate a data set with n = 500 and p = 2, such that the observations belong to two classes with a quadratic decision boundary between them. For instance, you can do this as follows:

##(b) Plot the observations, colored according to their class labels.Your plot should display X_1 on the x-axis, and X_2 on the y-axis.

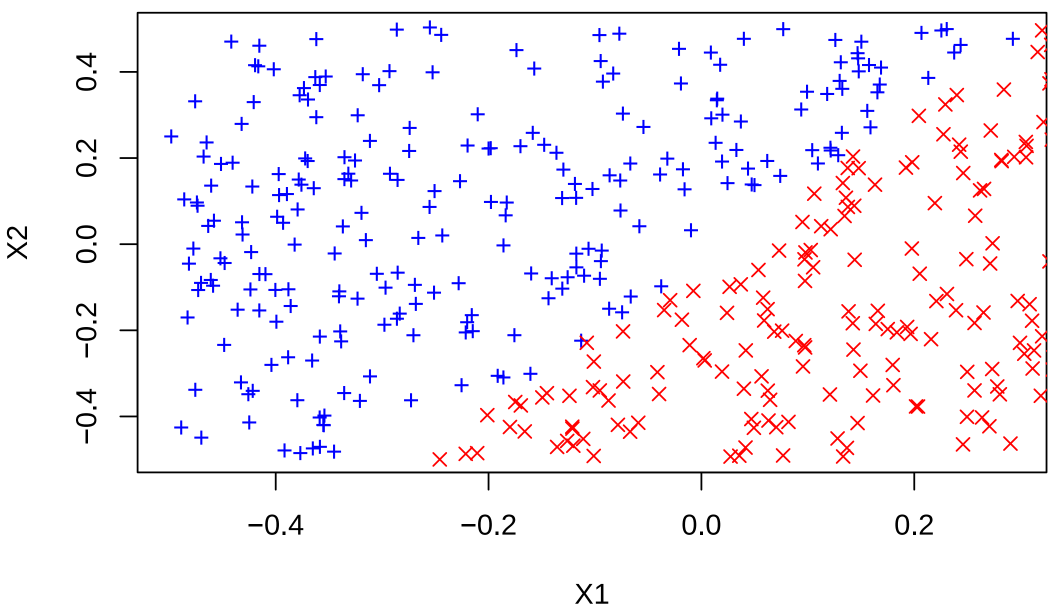


##(c) Fit a logistic regression model to the data, using X_1 and X_2 as predictors.

```
##
## Call:
## glm(formula = y ~ x1 + x2, family = "binomial")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3633  -1.1724   0.0033   1.1550   1.3627
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.01340    0.09011   0.149   0.8818
```

```
## x1          -0.58219    0.31095  -1.872   0.0612 .
## x2           0.33647    0.31313   1.075   0.2826
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 693.15  on 499  degrees of freedom
## Residual deviance: 688.53  on 497  degrees of freedom
## AIC: 694.53
##
## Number of Fisher Scoring iterations: 3
```

##(d) Apply this model to the training data in order to obtain a predicted class label for each training observation.Plot the observations, colored according to the predicted class labels. The decision boundary should be linear.



From the plot, I can make the conclusion that the decision boundary is obviously linear.

##(e) Now fit a logistic regression model to the data using non-linear functions of X_1 and X_2 as predictors (e.g.${X_1}^2$, $X_1 * X_2$, $log(X_2)$,and so forth).

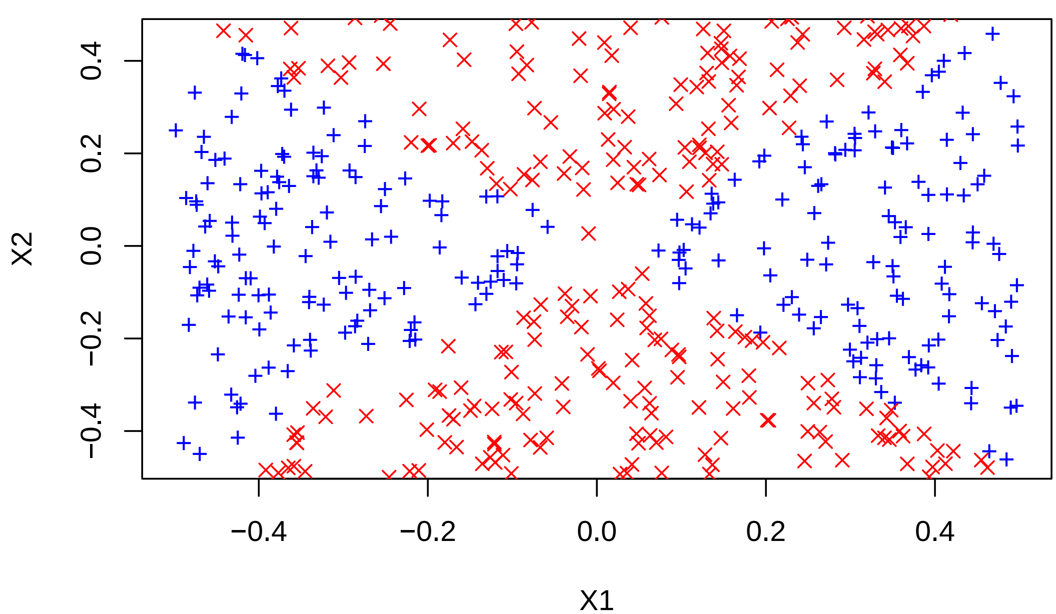
```
## Warning: glm.fit: algorithm did not converge

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

##
```

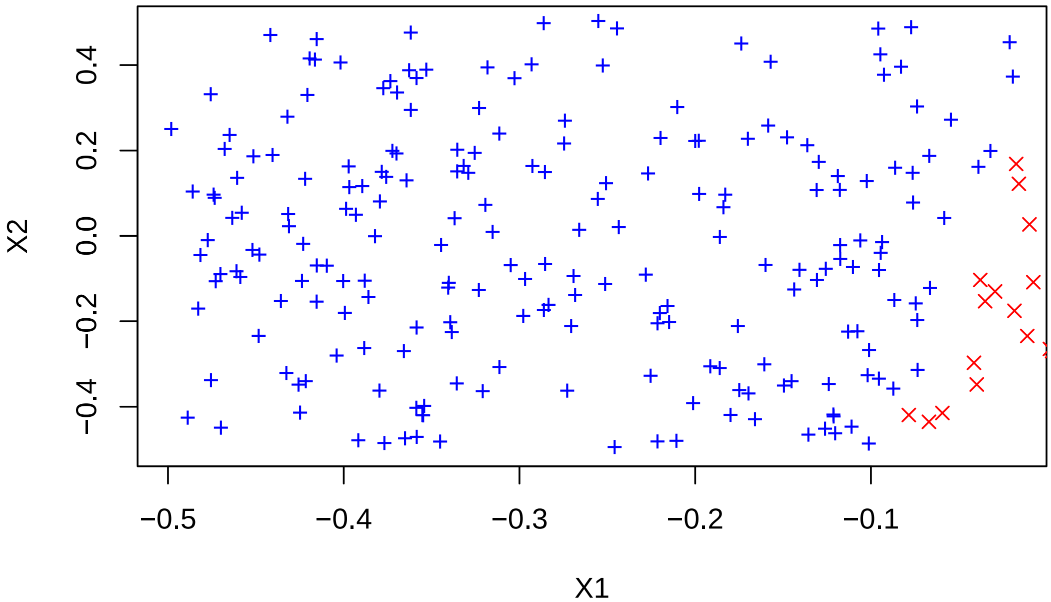
```
## Call:
## glm(formula = y ~ poly(x1, 2) + poly(x2, 2) + I(x1 * x2), family = "binomial",
##     data = data)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -1.351e-03  -2.000e-08   0.000e+00   2.000e-08   1.585e-03
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)      22.58    3238.06   0.007    0.994
## poly(x1, 2)1  -1113.79   68883.32  -0.016    0.987
## poly(x1, 2)2  33828.12  881545.10   0.038    0.969
## poly(x2, 2)1   1739.84   82221.68   0.021    0.983
## poly(x2, 2)2 -35239.84  921552.80  -0.038    0.969
## I(x1 * x2)     -404.32   40198.38  -0.010    0.992
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6.9315e+02  on 499  degrees of freedom
## Residual deviance: 5.4803e-06  on 494  degrees of freedom
## AIC: 12
##
## Number of Fisher Scoring iterations: 25
```

##(f) Apply this model to the training data in order to obtain a predicted class label for each training observation. Plot the observations, colored according to the predicted class labels. The decision boundary should be obviously non-linear. If it is not, then repeat (a)-(e) until you come up with an example in which the predicted class labels are obviously non-linear.

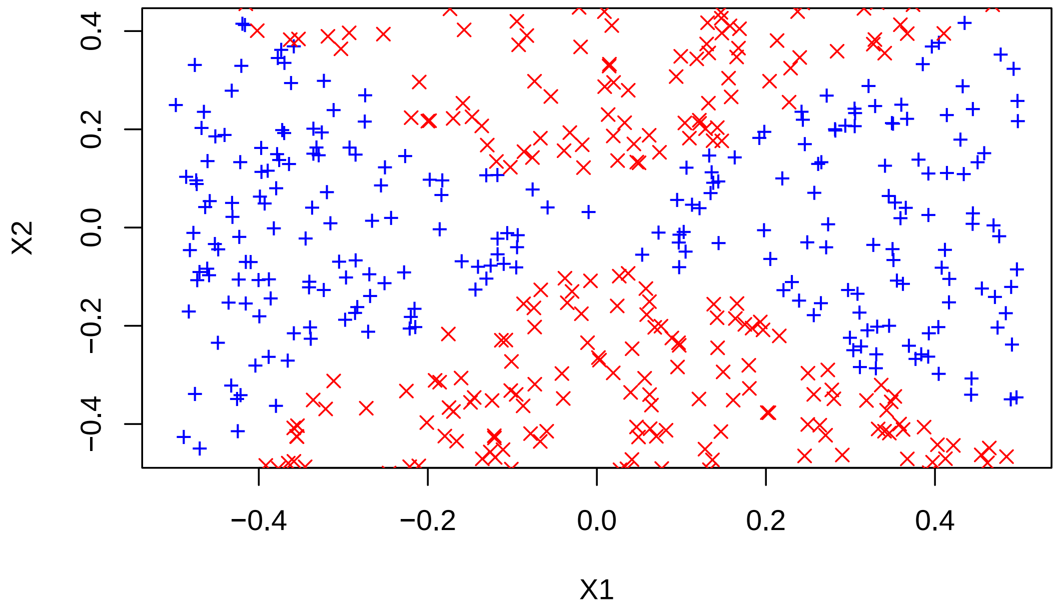


The non-linear decision boundary is surprisingly very similar to the true decision boundary.

##(g) Fit a support vector classifier to the data with X_1 and X_2 as predictors. Obtain a class prediction for each training observation. Plot the observations,colored according to the predicted class labels.



##(h) Fit a SVM using a non-linear kernel to the data.Obtain a class prediction for each training observation.Plot the observations,colored according to the predicted class labels.



##(i) Comment on your results.

The effect of linear logistic regression on nonlinear boundary processing is poor, the effect of SVM linear kernel with small cost is good, and the effect of nonlinear logistic regression and SVM on nonlinear boundary processing is perfect. ### 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. Note you will need to fit the classifier without the gas mileage variable to produce sensible results.

```
##
## Parameter tuning of 'e1071::svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.01275641
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-02 0.07653846 0.04350608
## 2 1e-01 0.04858974 0.04432285
```

```
## 3 1e+00 0.01275641 0.01344780
## 4 5e+00 0.01794872 0.02110955
## 5 1e+01 0.02551282 0.02417610
## 6 1e+02 0.03057692 0.02638876
## 7 1e+03 0.03057692 0.02638876
```

A cost of 0.01 seems to perform best.

##(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.

```
##
## Parameter tuning of 'e1071::svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##   100      2
##
## - best performance: 0.3162179
##
## - Detailed performance results:
##     cost degree     error dispersion
## 1  1e-02      2 0.5791667 0.04897523
## 2  1e-01      2 0.5791667 0.04897523
## 3  1e+00      2 0.5791667 0.04897523
## 4  5e+00      2 0.5791667 0.04897523
## 5  1e+01      2 0.5587179 0.06226145
## 6  1e+02      2 0.3162179 0.09160665
## 7  1e-02      3 0.5791667 0.04897523
## 8  1e-01      3 0.5791667 0.04897523
## 9  1e+00      3 0.5791667 0.04897523
## 10 5e+00      3 0.5791667 0.04897523
## 11 1e+01      3 0.5791667 0.04897523
## 12 1e+02      3 0.3287821 0.12789151
## 13 1e-02      4 0.5791667 0.04897523
## 14 1e-01      4 0.5791667 0.04897523
## 15 1e+00      4 0.5791667 0.04897523
## 16 5e+00      4 0.5791667 0.04897523
## 17 1e+01      4 0.5791667 0.04897523
## 18 1e+02      4 0.5791667 0.04897523
```

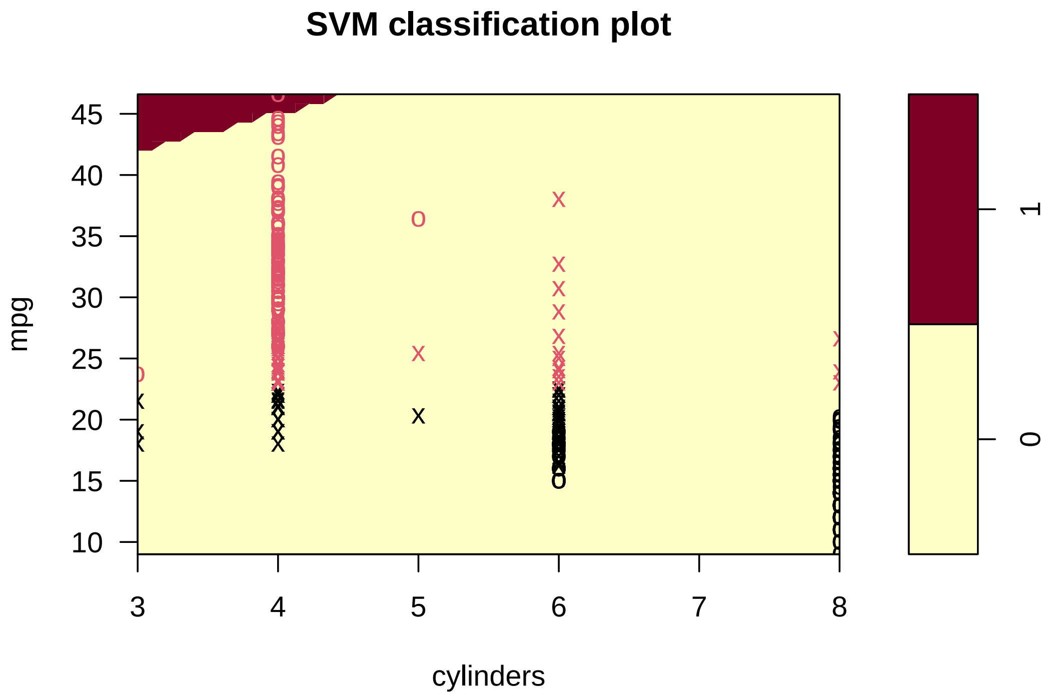
For a polynomial kernel, the lowest cross-validation error is obtained for a degree of 3 and a cost of 100.

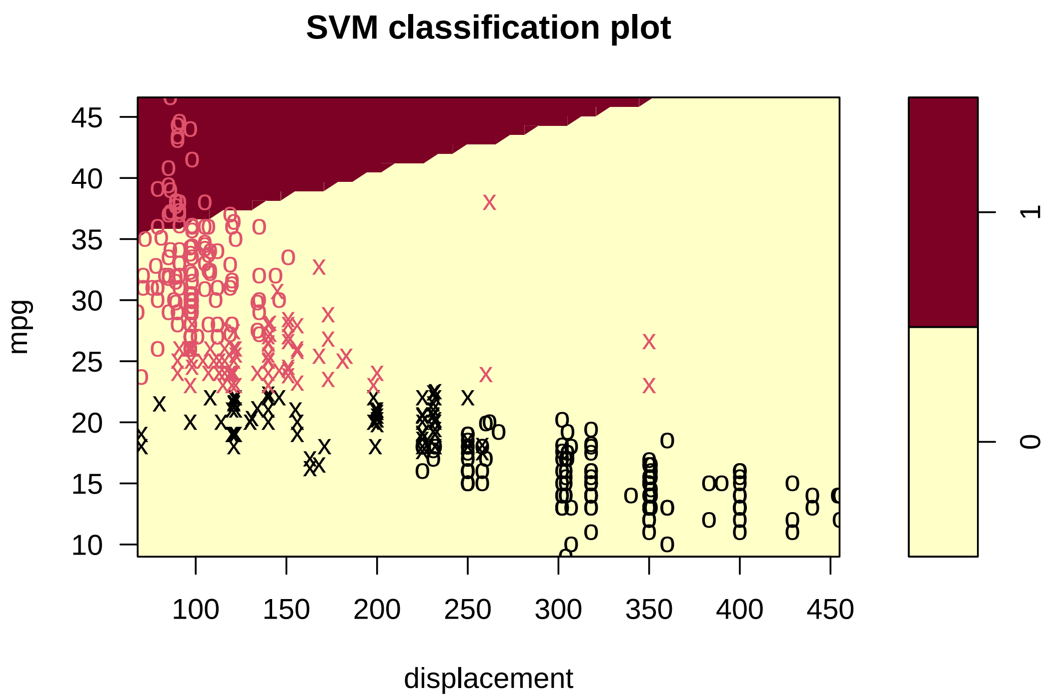
```
##
## Parameter tuning of 'e1071::svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##   100  0.01
##
```

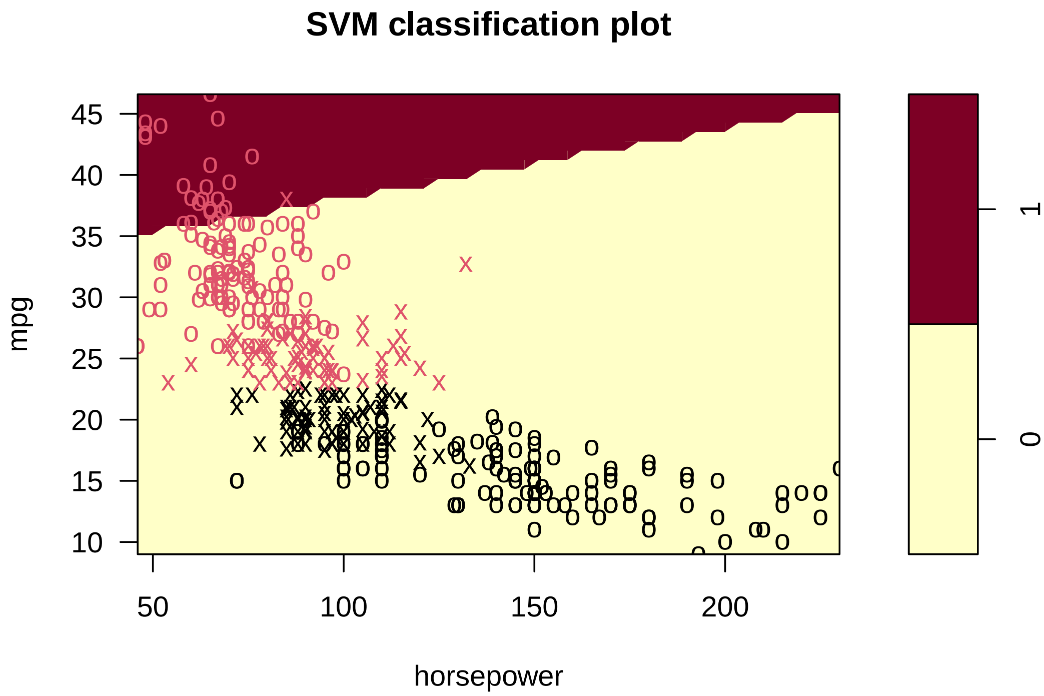
```
## - best performance: 0.01282051
##
## - Detailed performance results:
##     cost gamma      error dispersion
## 1  1e-02 1e-02 0.57916667 0.04897523
## 2  1e-01 1e-02 0.08929487 0.05827503
## 3  1e+00 1e-02 0.07141026 0.04478033
## 4  5e+00 1e-02 0.04858974 0.04432285
## 5  1e+01 1e-02 0.02051282 0.02648194
## 6  1e+02 1e-02 0.01282051 0.01813094
## 7  1e-02 1e-01 0.22211538 0.07344788
## 8  1e-01 1e-01 0.07903846 0.04582628
## 9  1e+00 1e-01 0.05628205 0.04492041
## 10 5e+00 1e-01 0.03076923 0.03151981
## 11 1e+01 1e-01 0.02564103 0.03197998
## 12 1e+02 1e-01 0.03326923 0.02974993
## 13 1e-02 1e+00 0.57916667 0.04897523
## 14 1e-01 1e+00 0.57916667 0.04897523
## 15 1e+00 1e+00 0.05365385 0.03520658
## 16 5e+00 1e+00 0.05628205 0.04334690
## 17 1e+01 1e+00 0.05628205 0.04334690
## 18 1e+02 1e+00 0.05628205 0.04334690
## 19 1e-02 5e+00 0.57916667 0.04897523
## 20 1e-01 5e+00 0.57916667 0.04897523
## 21 1e+00 5e+00 0.51820513 0.07134333
## 22 5e+00 5e+00 0.51570513 0.07151659
## 23 1e+01 5e+00 0.51570513 0.07151659
## 24 1e+02 5e+00 0.51570513 0.07151659
## 25 1e-02 1e+01 0.57916667 0.04897523
## 26 1e-01 1e+01 0.57916667 0.04897523
## 27 1e+00 1e+01 0.53333333 0.07558729
## 28 5e+00 1e+01 0.53076923 0.07387664
## 29 1e+01 1e+01 0.53076923 0.07387664
## 30 1e+02 1e+01 0.53076923 0.07387664
## 31 1e-02 1e+02 0.57916667 0.04897523
## 32 1e-01 1e+02 0.57916667 0.04897523
## 33 1e+00 1e+02 0.57916667 0.04897523
## 34 5e+00 1e+02 0.57916667 0.04897523
## 35 1e+01 1e+02 0.57916667 0.04897523
## 36 1e+02 1e+02 0.57916667 0.04897523
```

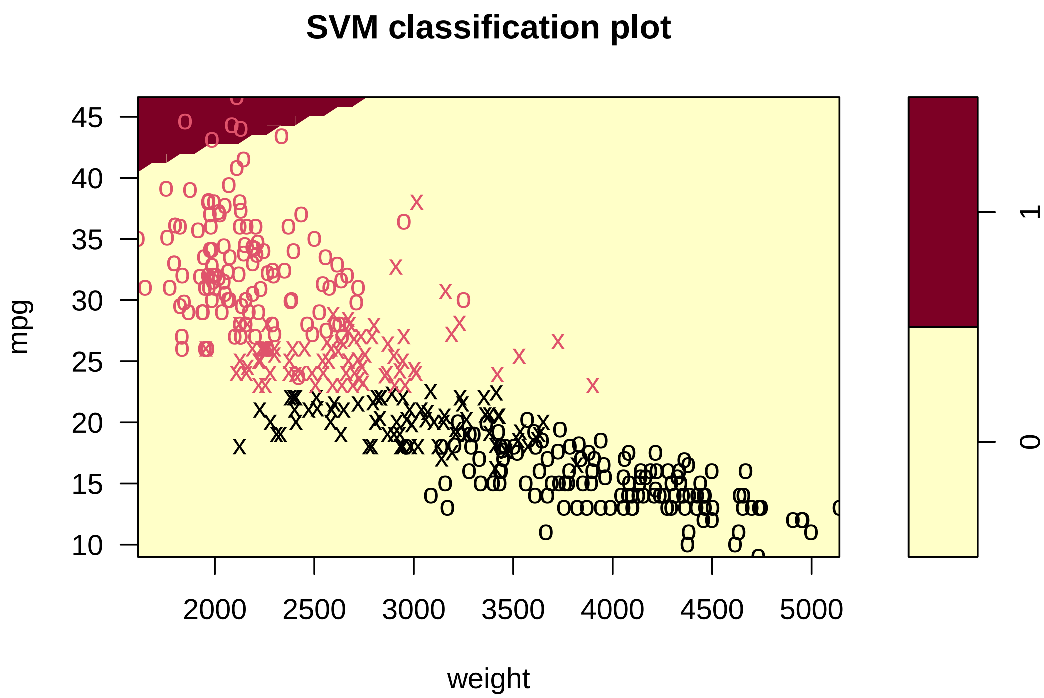
For the radial kernel, the lowest cross-validation error is obtained for a gamma of 0.01 and a cost of 0.1.

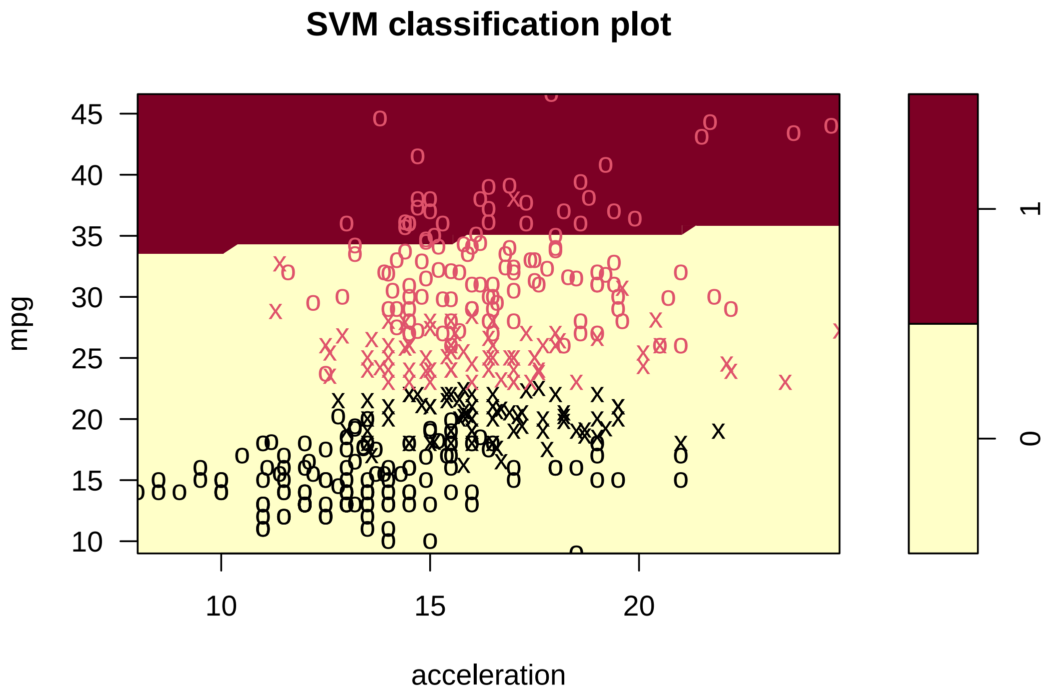
##(d) Make some plots to back up your assertions in (b) and (c).

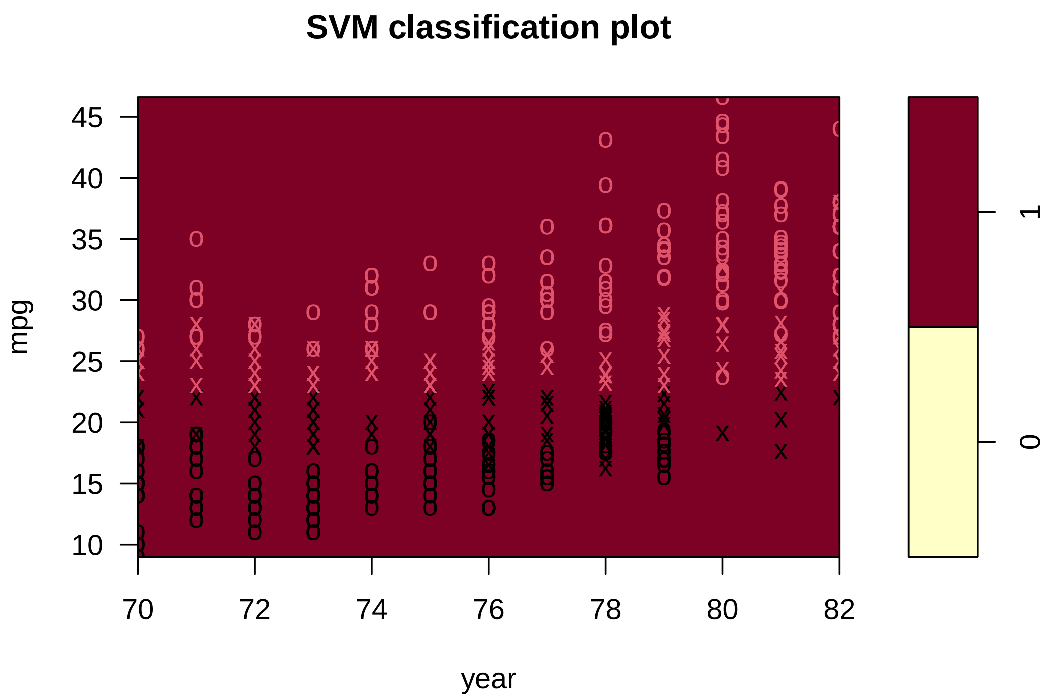

SVM classification plot
mpg
cylinders
45
40
35
30
25
20
15
10
3
4
5
6
7
8
1
0

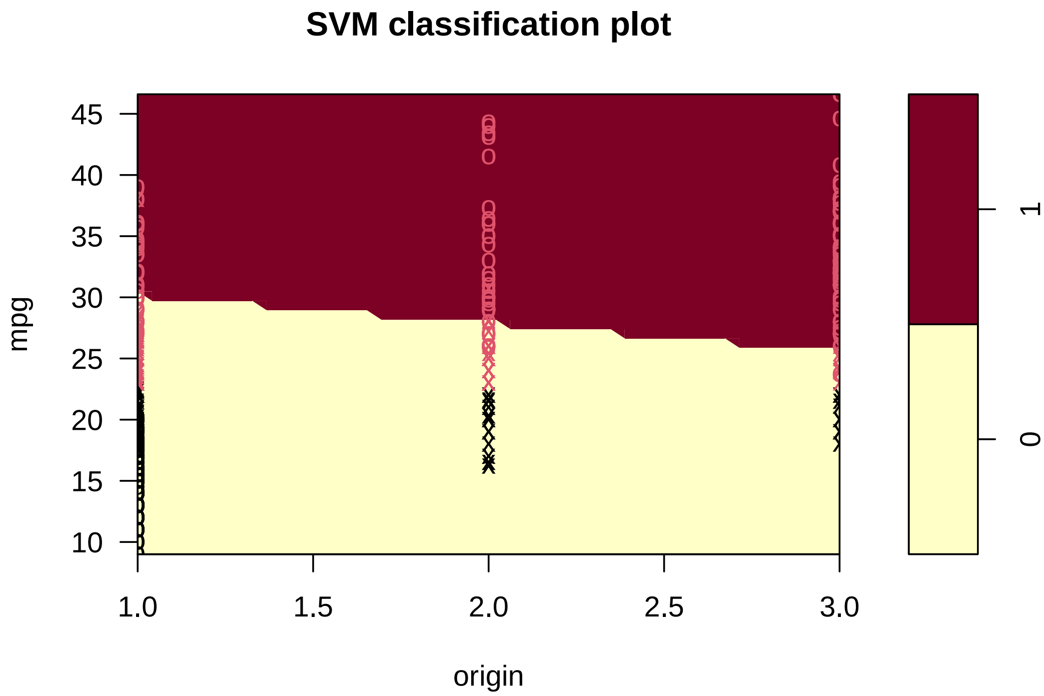

SVM classification plot
mpg
displacement
45
40
35
30
25
20
15
10
100
150
200
250
300
350
400
450
1
0


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

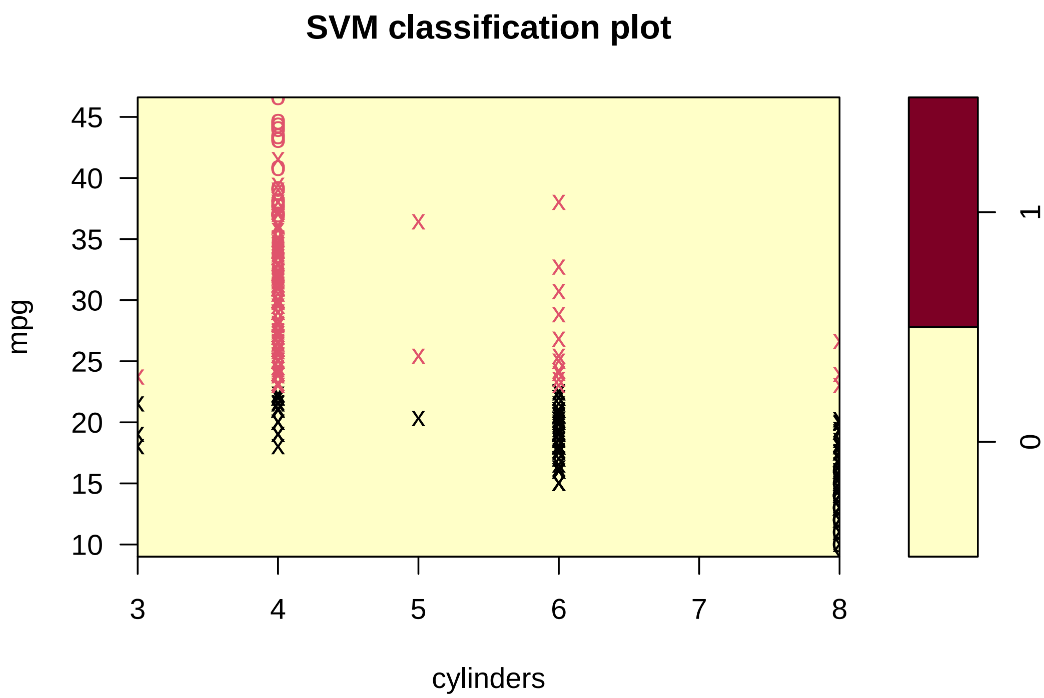

SVM classification plot
mpg
45
40
35
30
25
20
15
10
2000
2500
3000
3500
4000
4500
5000
weight
1
0


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

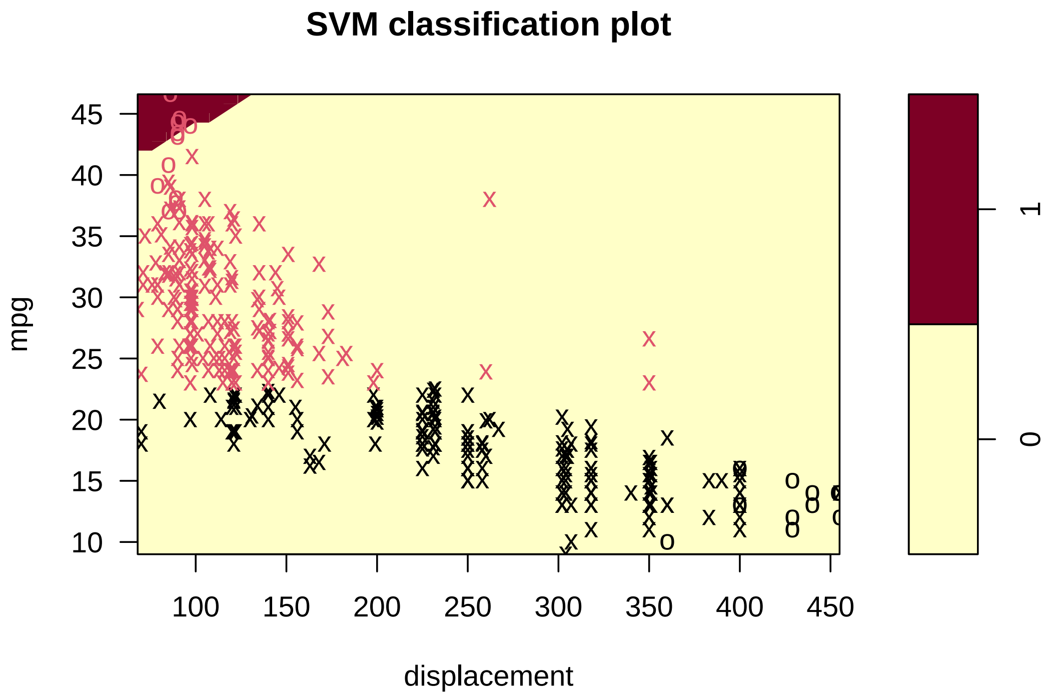

SVM classification plot
mpg
45
40
35
30
25
20
15
10
year
70
72
74
76
78
80
82
1
0

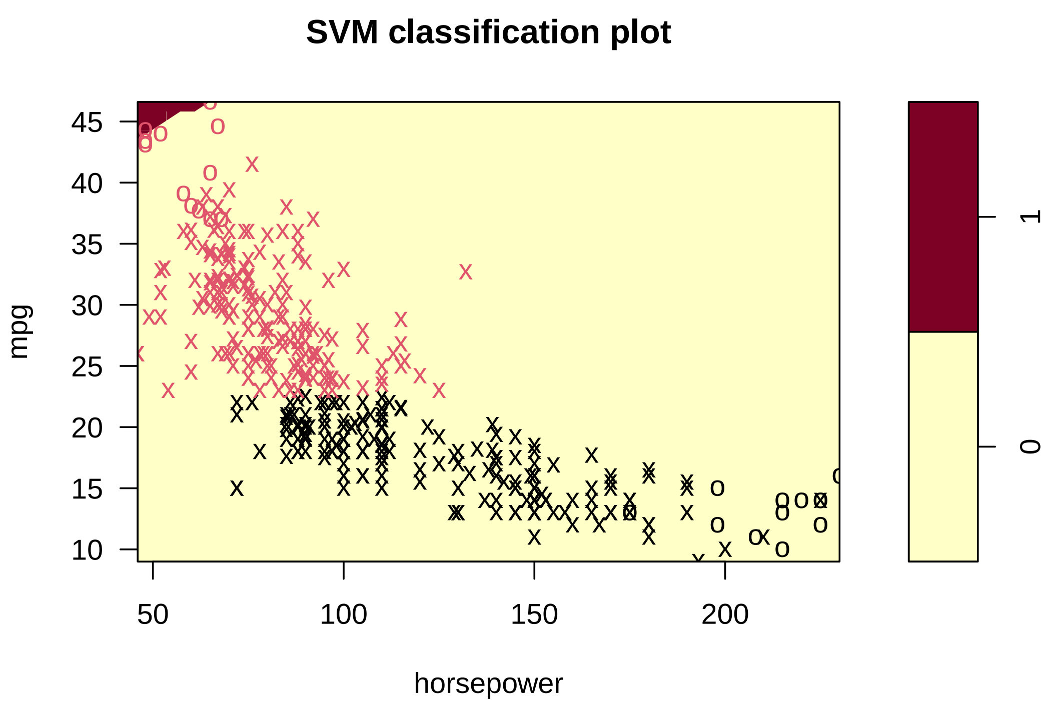


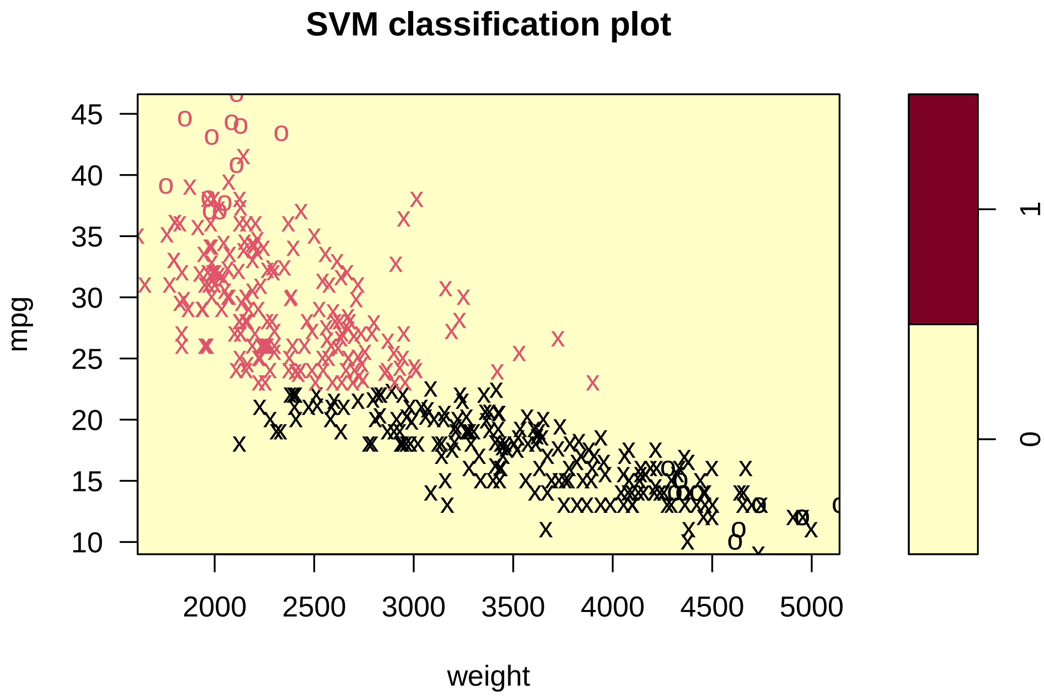
SVM classification plot
mpg
45
40
35
30
25
20
15
10
1.0
1.5
2.0
2.5
3.0
origin
1
0

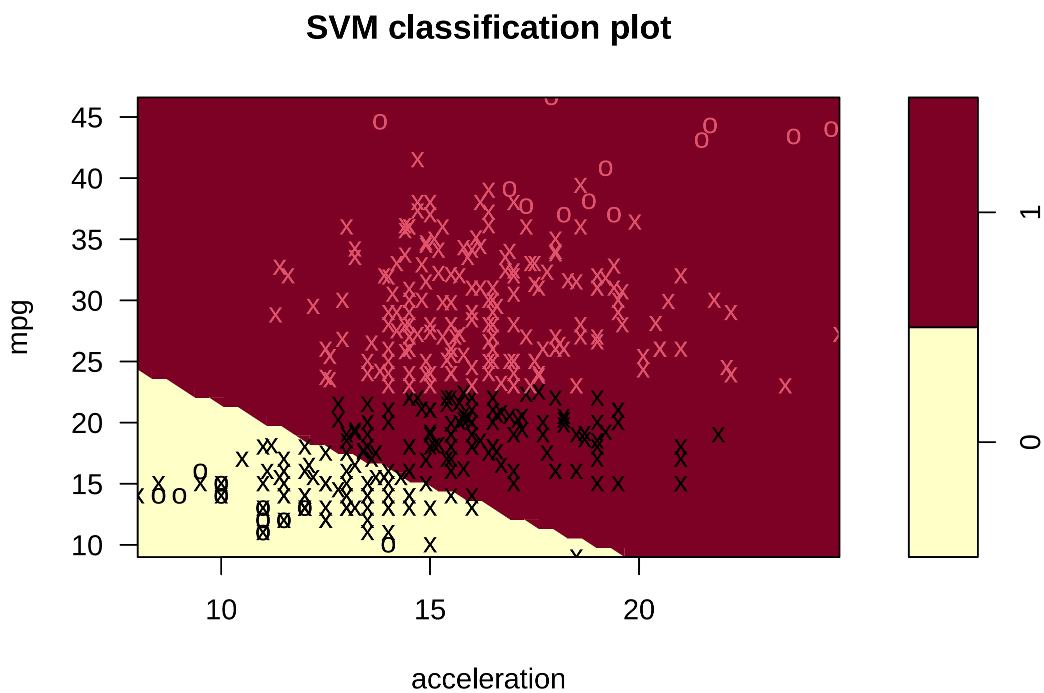


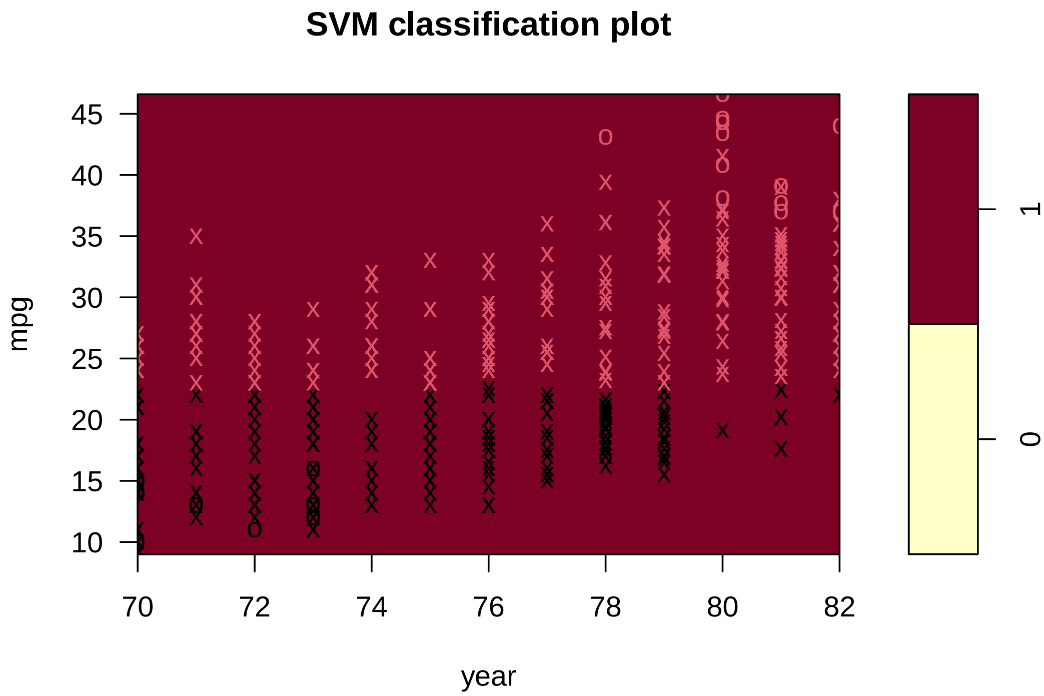
SVM classification plot
mpg
45
40
35
30
25
20
15
10
3
4
5
6
7
8
cylinders
1
0

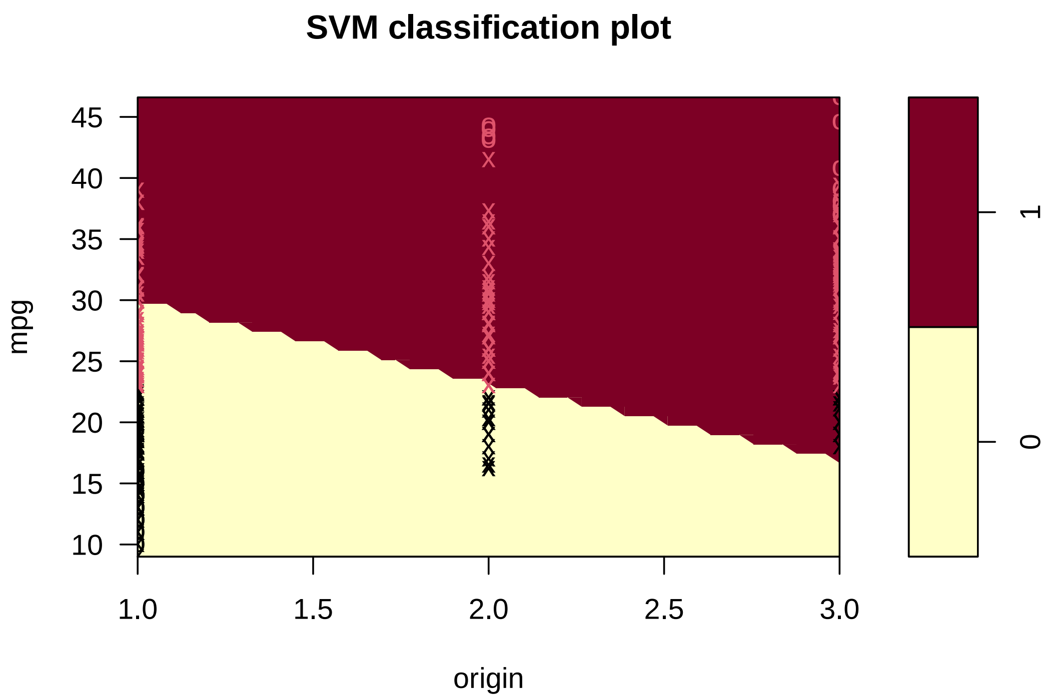

SVM classification plot
mpg
displacement
0
1


SVM classification plot
mpg
horsepower
0
1

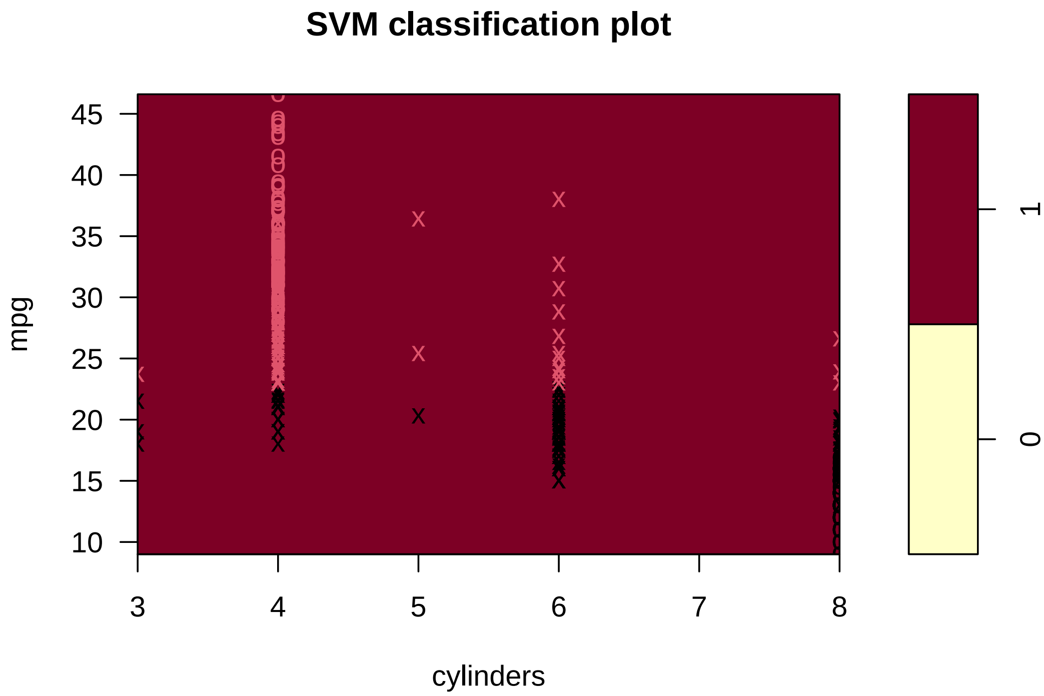

SVM classification plot
mpg
weight
45
40
35
30
25
20
15
10
2000
2500
3000
3500
4000
4500
5000
1
0

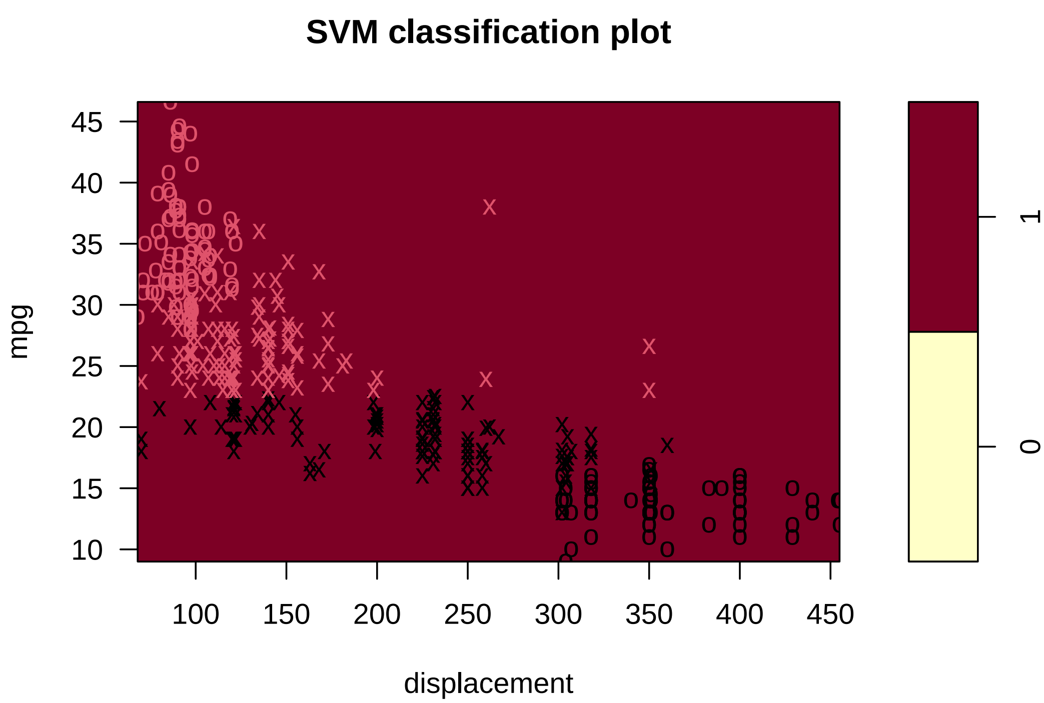

SVM classification plot
mpg
acceleration
45
40
35
30
25
20
15
10
10
15
20
1
0

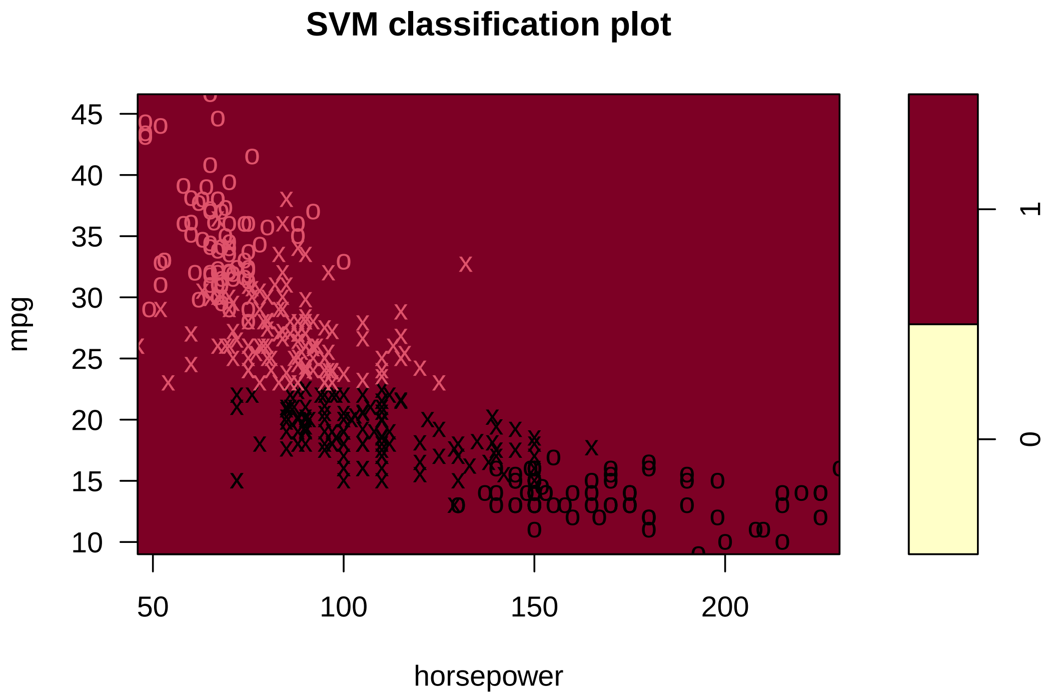


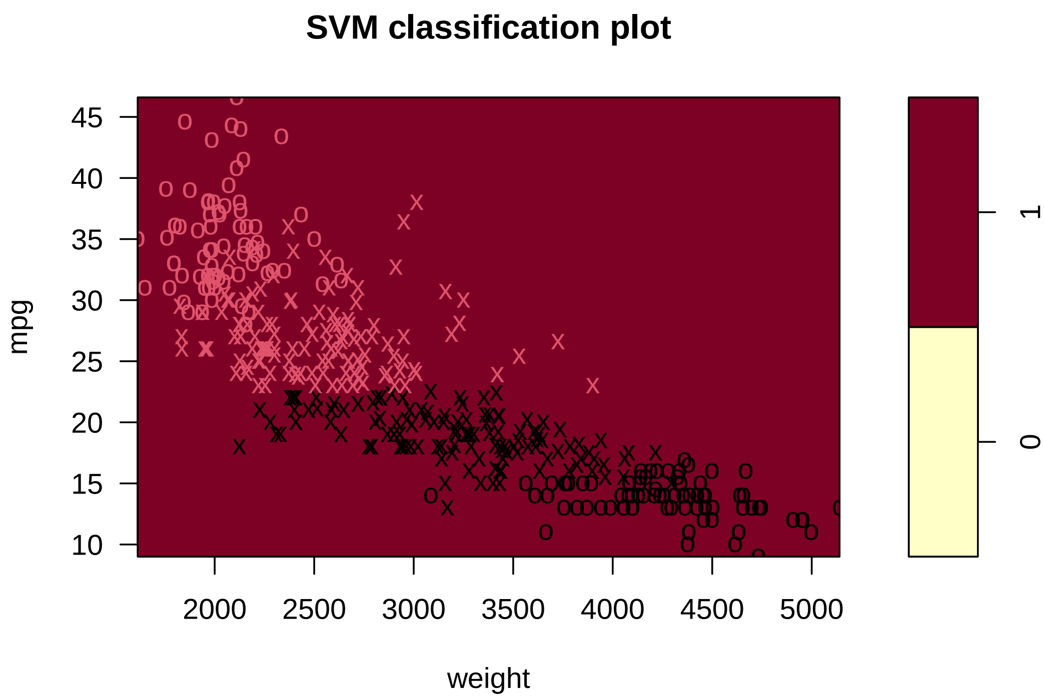



```
plotpairs(radial)
```


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

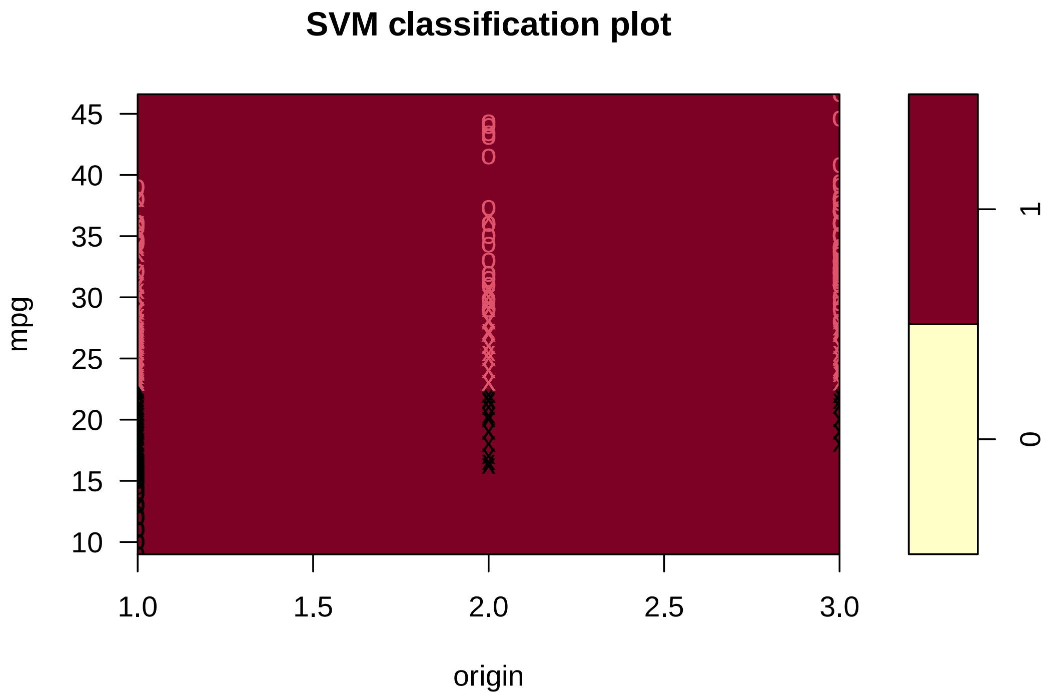

SVM classification plot
mpg
45
40
35
30
25
20
15
10
100
150
200
250
300
350
400
450
displacement
1
0


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
2500
3000
3500
4000
4500
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



9.8

This problem involves the OJ data set which is part of the ISLR2 package. ##(a) Create a training set containing a random sample of 800 observations, and a test set containing the remaining observations.

##(b) Fit a support vector classifier to the training data using cost = 0.01, with Purchase as the response and the other variables as predictors. Use the summary() function to produce summary statistics, and describe the results obtained.

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, kernel = "linear", cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##
## Number of Support Vectors:  437
##
##  ( 218 219 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

Support vector classifier creates 435 support vectors out of 800 training points. Out of these, 216 belong to level MM and remaining 219 belong to level CH.

##(c) What are the training and test error rates?

```
##     train.pred
##       CH  MM
##   CH 440  53
##   MM  74 233

## [1] 0.175

##     test.pred
##       CH  MM
##   CH 138  22
##   MM  32  78

## [1] 0.1777778
```

The training error rate is 17.5% , the test error rate is 17.8%.

##(d) Use the tune() function to select an optimal cost. Consider values in the range 0.01 to 10.

```
##
## Parameter tuning of 'e1071::svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##        cost
##  0.03162278
##
## - best performance: 0.16125
##
## - Detailed performance results:
##           cost   error dispersion
## 1   0.01000000 0.16250 0.01863390
## 2   0.01778279 0.16500 0.02108185
## 3   0.03162278 0.16125 0.02316157
## 4   0.05623413 0.16375 0.02316157
## 5   0.10000000 0.16500 0.02751262
## 6   0.17782794 0.16125 0.02161050
## 7   0.31622777 0.16875 0.02585349
## 8   0.56234133 0.16875 0.02447363
## 9   1.00000000 0.16500 0.02108185
## 10  1.77827941 0.16750 0.02648375
## 11  3.16227766 0.16750 0.02898755
## 12  5.62341325 0.17000 0.02838231
## 13 10.00000000 0.16625 0.02571883
```

Here the optimal cost is 0.03.

##(e) Compute the training and test error rates using this new value for cost.

```
##     train.pred
##       CH  MM
##   CH 435  58
##   MM  70 237
```

```
## [1] 0.16
```

```
##     test.pred
##       CH  MM
##   CH 137  23
##   MM  31  79
```

```
## [1] 0.2
```

We may see that, with the best cost, the trainging error rate is 16% and the test error rate is 20%.

##(f) Repeat parts (b) through (e) using a support vector machine with a radial kernel. Use the default value for gamma.

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  368
##
##  ( 181 187 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```
##     train.pred
##       CH  MM
##   CH 461  32
##   MM  77 230
```

```
## [1] 0.13625
```

```
##     test.pred
##       CH  MM
##   CH 138  22
##   MM  32  78
```

```
## [1] 0.2
```

Radial kernel with default gamma creates 368 support vectors, out of which, 181 belong to level CH and remaining 187 belong to level MM.The classifier has a training error of 13.63% and a test error of 20% which is a slight improvement over linear kernel.Next I use cross validation to find optimal cost.

```
##
## Parameter tuning of 'e1071::svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##      cost
##  1.778279
##
## - best performance: 0.155
##
## - Detailed performance results:
##           cost   error dispersion
## 1   0.01000000 0.38375 0.04566256
## 2   0.01778279 0.38375 0.04566256
## 3   0.03162278 0.36375 0.05084358
## 4   0.05623413 0.20750 0.03782269
## 5   0.10000000 0.17625 0.02791978
## 6   0.17782794 0.16625 0.03910900
## 7   0.31622777 0.16500 0.04440971
## 8   0.56234133 0.15625 0.04686342
## 9   1.00000000 0.15500 0.04609772
## 10  1.77827941 0.15500 0.04005205
## 11  3.16227766 0.16250 0.03584302
## 12  5.62341325 0.16375 0.03087272
## 13 10.00000000 0.16750 0.04048319
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, kernel = "radial", cost = tune.out$best.parameters$cost)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1.778279
##
## Number of Support Vectors:  350
##
##  ( 172 178 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```
##     train.pred1
##       CH  MM
```

```
##   CH 462  31
##   MM  78 229
```

```
## [1] 0.13625
```

```
##     test.pred1
##       CH  MM
##   CH 138  22
##   MM  33  77
```

```
## [1] 0.2037037
```

Tuning does not reduce train and test error rates significantly when I use the optimal best cost.

##(g) Repeat parts (b) through (e) using a support vector machine with a polynomial kernel. Set degree = 2.

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, kernel = "polynomial",
##     degree = 2)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  2
##      coef.0:  0
##
## Number of Support Vectors:  449
##
##  ( 223 226 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```
##     train.pred
##       CH  MM
##   CH 467  26
##   MM 107 200
```

```
## [1] 0.16625
```

```
##     test.pred
##       CH  MM
##   CH 141  19
##   MM  41  69
```

```
## [1] 0.2222222
```

Polynomial kernel with default gamma creates 449 support vectors, out of which, 223 belong to level CH and 226 belong to level MM.The classifier has a training error of 16.63% and a test error of 22.22% which is no improvement over linear kernel.

```
##
## Parameter tuning of 'e1071::svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##      cost
##  5.623413
##
## - best performance: 0.1625
##
## - Detailed performance results:
##           cost   error dispersion
## 1   0.01000000 0.38375 0.04566256
## 2   0.01778279 0.37000 0.04533824
## 3   0.03162278 0.33125 0.04299952
## 4   0.05623413 0.31625 0.03175973
## 5   0.10000000 0.31250 0.03004626
## 6   0.17782794 0.25375 0.03387579
## 7   0.31622777 0.19875 0.02972676
## 8   0.56234133 0.19375 0.03186887
## 9   1.00000000 0.18125 0.03186887
## 10  1.77827941 0.18000 0.03827895
## 11  3.16227766 0.17375 0.04101575
## 12  5.62341325 0.16250 0.04289846
## 13 10.00000000 0.16375 0.04427267
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, kernel = "polynomial",
##     degree = 2, cost = tune.out$best.parameters$cost)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  5.623413
##      degree:  2
##      coef.0:  0
##
## Number of Support Vectors:  354
##
##  ( 172 182 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```
##     train.pred
##       CH  MM
##   CH 464  29
##   MM  86 221
```

```
## [1] 0.14375
```

```
##     test.pred
##       CH  MM
##   CH 141  19
##   MM  37  73
```

```
## [1] 0.2074074
```

Here I find that tuning reduce train error rate to 14.38% and test error rate to 20.74%.

##(h) Overall, which approach seems to give the best results on this data?

Overall, radial basis kernel seems to be producing minimum misclassification error on both train and test data.