STATS 415 - Homework 9 - Support Vector Machines

Marian L. Schmidt

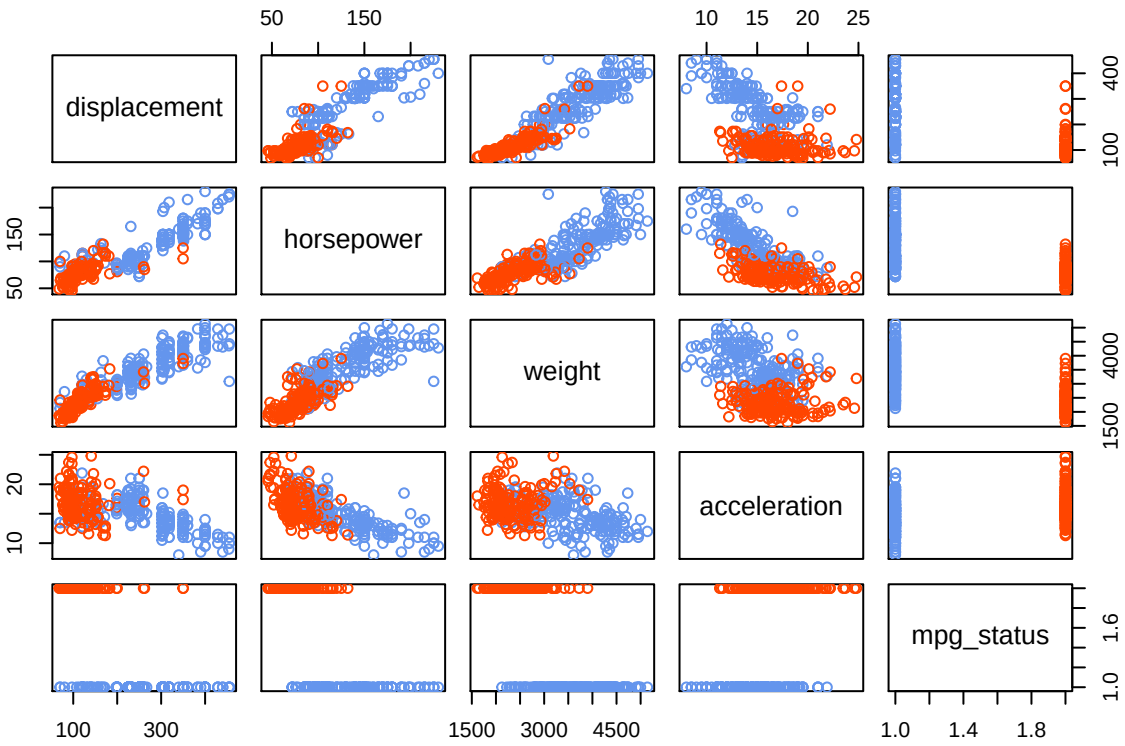
April 5, 2016

1. 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.

```
mpg_status <- ifelse(mpg > median(mpg), 1, 0)
Auto$mpg_status <- as.factor(mpg_status)
auto_data <- select(Auto, c(displacement, horsepower, weight, acceleration, mpg_status))
```

```
plot(auto_data, col = (cols))
```



(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.

```
set.seed(1)
### divide into equal sets of testing and training data
train <-  sample(1:dim(auto_data)[1], dim(auto_data)[1] / 2)
test <- -train
train_auto <- auto_data[train, ]
test_auto <- auto_data[test, ]
# Run the SVM
```

```
tune_out_linear <- tune(svm, mpg_status ~ ., data = train_auto, kernel = "linear",
                        ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)))
best_auto_linear <- tune_out_linear$best.model
summary(best_auto_linear)
```

```
##
## Call:
## best.tune(method = svm, train.x = mpg_status ~ ., data = train_auto,
##     ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)),
##     kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.1
##       gamma:  0.25
##
## Number of Support Vectors:  78
##
##  ( 39 39 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
y_prediction_linear <- predict(best_auto_linear, test_auto)
linear <- table(predict = y_prediction_linear, truth = test_auto$mpg_status);linear
```

```
##        truth
## predict  0  1
##       0 79  5
##       1 14 98
```

For a linear kernel, the lowest cross-validation error is obtained for a gamma of 0.25 and a cost of 0.1. The test error rate of this linear kernel is 10.7344633%

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

```
set.seed(1)
tune_out_radial <- tune(svm, mpg_status ~ ., data = train_auto, kernel = "radial",
                        ranges = list(cost = c(0.01, 0.1, 1, 5, 10, 100, 1000),
                                      gamma = c(0.01, 0.1, 1, 5, 10, 100, 1000)))
best_auto_radial <- tune_out_radial$best.model
summary(best_auto_radial)
```

```
##
## Call:
```

```
## best.tune(method = svm, train.x = mpg_status ~ ., data = train_auto,
##     ranges = list(cost = c(0.01, 0.1, 1, 5, 10, 100, 1000), gamma = c(0.01,
##         0.1, 1, 5, 10, 100, 1000)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  5
##       gamma:  1
##
## Number of Support Vectors:  66
##
##  ( 38 28 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
y_prediction_radial <- predict(best_auto_radial, test_auto)
radial <- table(predict = y_prediction_radial, truth = test_auto$mpg_status);radial
```

```
##        truth
## predict  0  1
##       0 78  6
##       1 15 97
```

For a radial kernel, the lowest cross-validation error is obtained for a degree of 3 and a cost of 5. The test error rate of this radial kernel is 12%

```
set.seed(1)
tune_out_poly <- tune(svm, mpg_status ~ ., data = train_auto, kernel = "polynomial",
                      gamma = 1, coef0=1,
                      ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100, 1000),
                      degree = c(2, 3, 4, 5, 6)))
best_auto_poly <- tune_out_poly$best.model
summary(best_auto_poly)
```

```
##
## Call:
## best.tune(method = svm, train.x = mpg_status ~ ., data = train_auto,
##     ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100, 1000),
##         degree = c(2, 3, 4, 5, 6)), kernel = "polynomial", gamma = 1,
##     coef0 = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  10
```

```
##       degree:  5
##        gamma:  1
##      coef.0:  1
##
## Number of Support Vectors:  33
##
##  ( 14 19 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
y_prediction_poly <- predict(best_auto_poly, test_auto)
poly <- table(predict = y_prediction_poly, truth = test_auto$mpg_status);poly
```

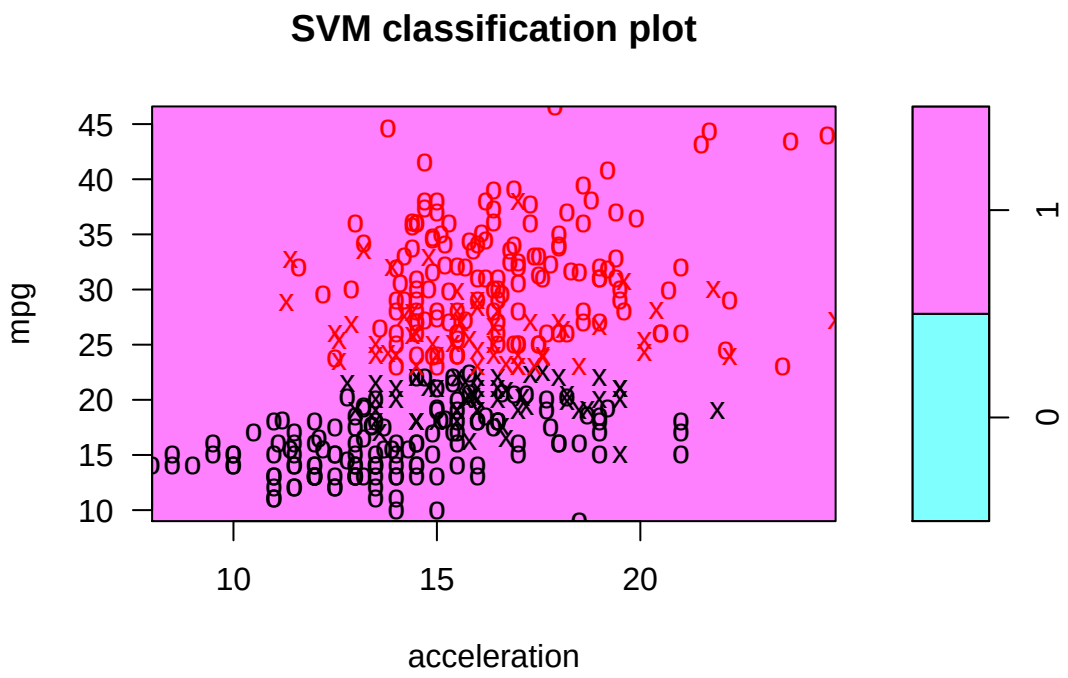
```
##        truth
## predict  0  1
##       0 80 11
##       1 13 92
```

For a polynomial kernel, the lowest cross-validation error is obtained for a degree of 5 and a cost of 10. The test error rate of this polynomial kernel is 13.9534884%

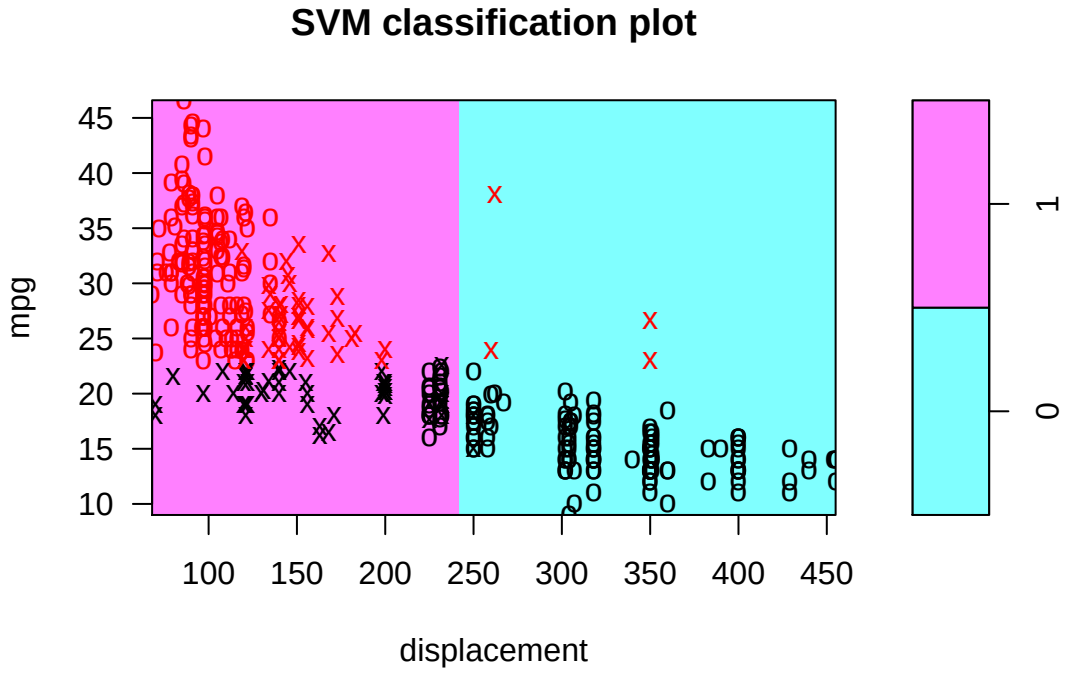
(d) Make some plots to back up your assertions in (b) and (c). Hint: In the lab, we used the `plot()` function for svm objects only in cases with `p = 2`. When `p > 2`, you can use the `plot()` function to create plots displaying pairs of variables at a time. Essentially, instead of typing `plot(svmfit, 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".

This time we will input all of the data into the models. I have tried hard to change the title and to make the divide between the cyan and pink along the y-axis, instead of the x-axis, to reflect the real support vector classifier. Unfortunately, I was unable to figure this out after a long time.

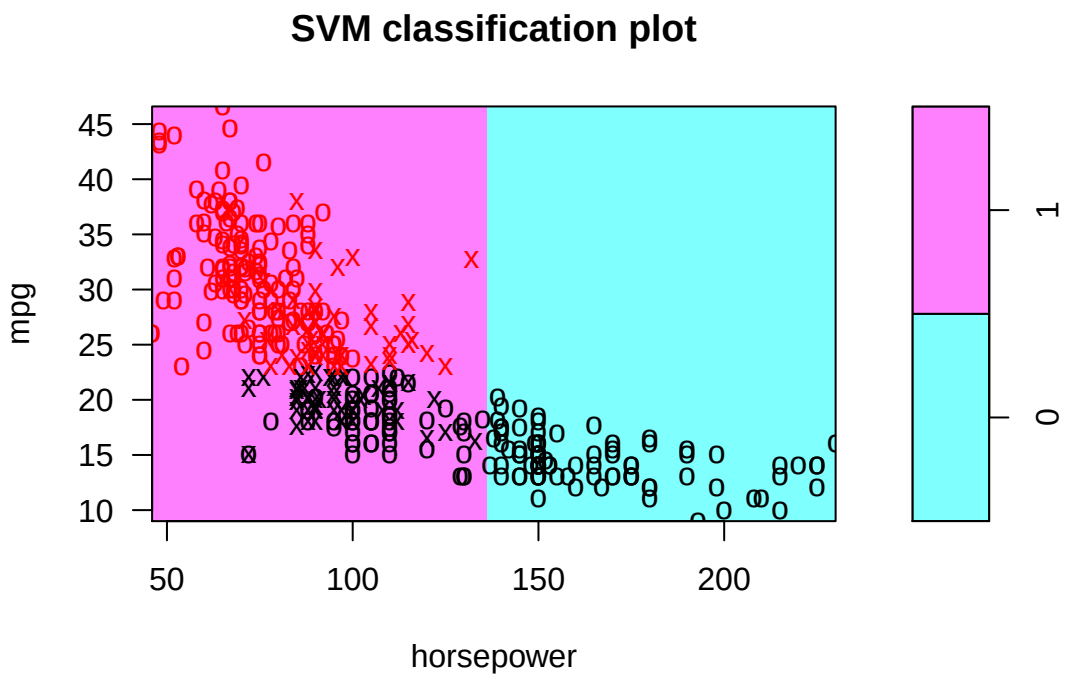
```
svm_linear <- svm(mpg_status ~ ., data = auto_data, kernel = "linear", cost = 1)
svm_radial = svm(mpg_status ~ ., data = auto_data, kernel = "radial", cost = 10, gamma = 0.01)
svm_poly = svm(mpg_status ~ ., data = auto_data, kernel = "polynomial", cost = 0.10, degree = 2,
               gamma = 1, coef0=1)
auto_data$mpg <- Auto$mpg # add mpg back in for plotting
# For slicing the plot variables
mean_acceleration <- mean(auto_data$acceleration); mean_weight <- mean(auto_data$weight)
mean_horsepower <- mean(auto_data$horsepower); mean_displacement <- mean(auto_data$displacement)
mean_mpg <- mean(auto_data$mpg)
# Plot
plot(svm_linear, auto_data, mpg ~ acceleration, main = "Linear: mpg~acceleration",
     slice = list(displacement = mean_displacement, horsepower = mean_horsepower))
```



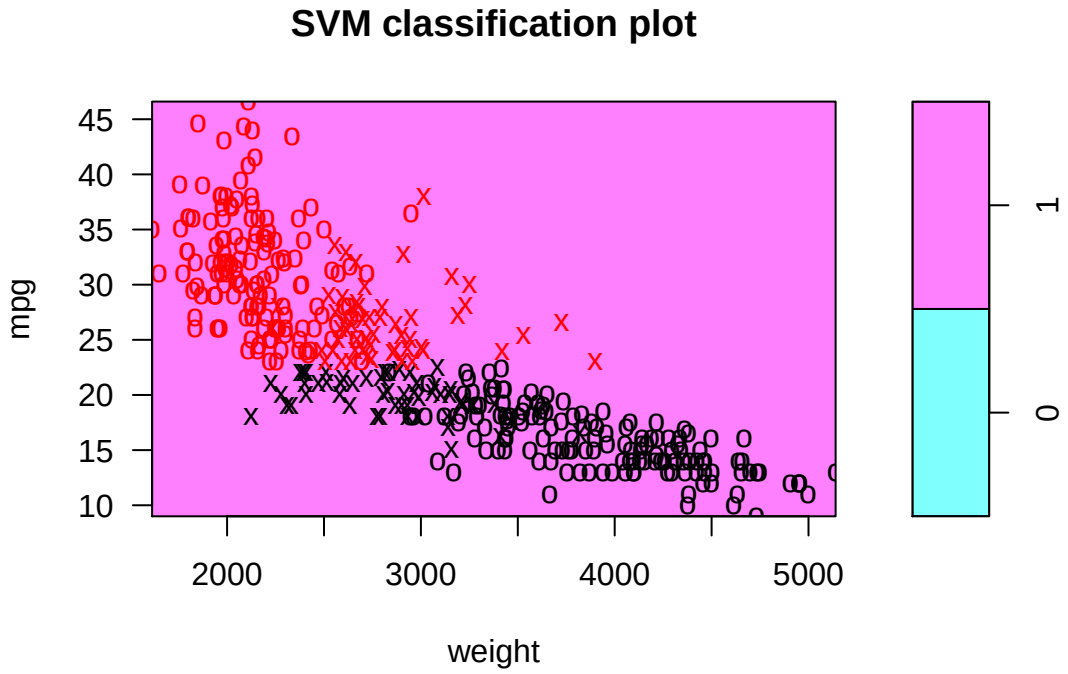

```
plot(svm_linear, auto_data, mpg ~ displacement, main = "Linear: mpg~displacement",
     slice = list(acceleration = mean_acceleration, horsepower = mean_horsepower))
```



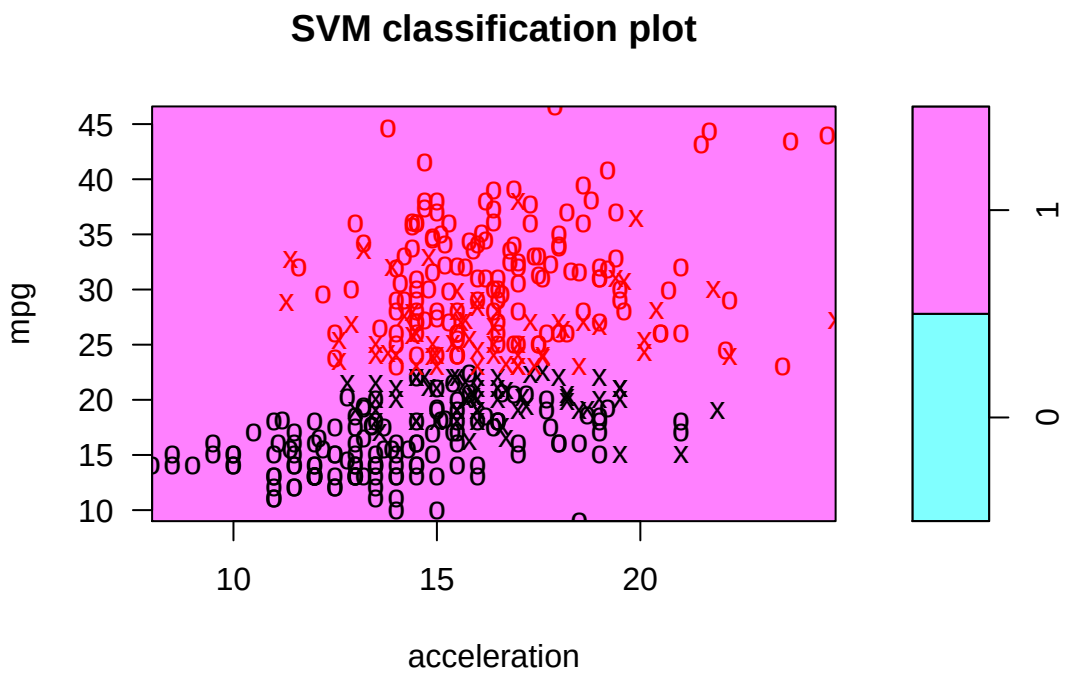

```
plot(svm_linear, auto_data, mpg ~ horsepower, main = "Linear: mpg~horsepower",
     slice = list(acceleration = mean_acceleration, displacement = mean_displacement))
```



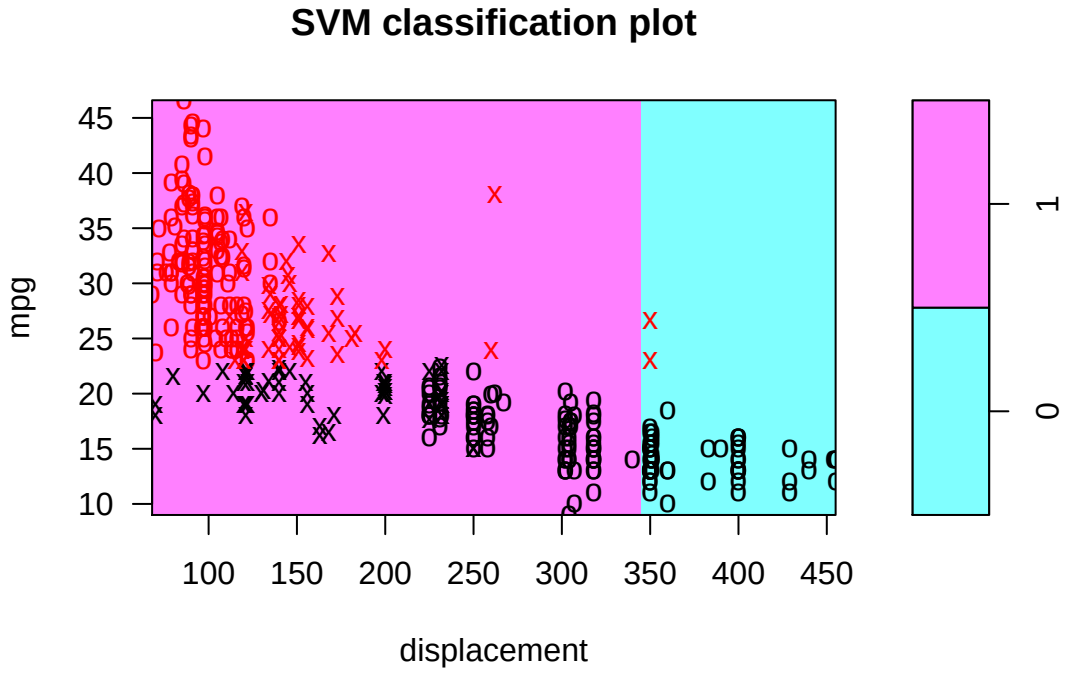

```
plot(svm_linear, auto_data, mpg ~ weight, main = "Linear: mpg~weight",
     slice = list(acceleration = mean_acceleration, displacement = mean_displacement))
```



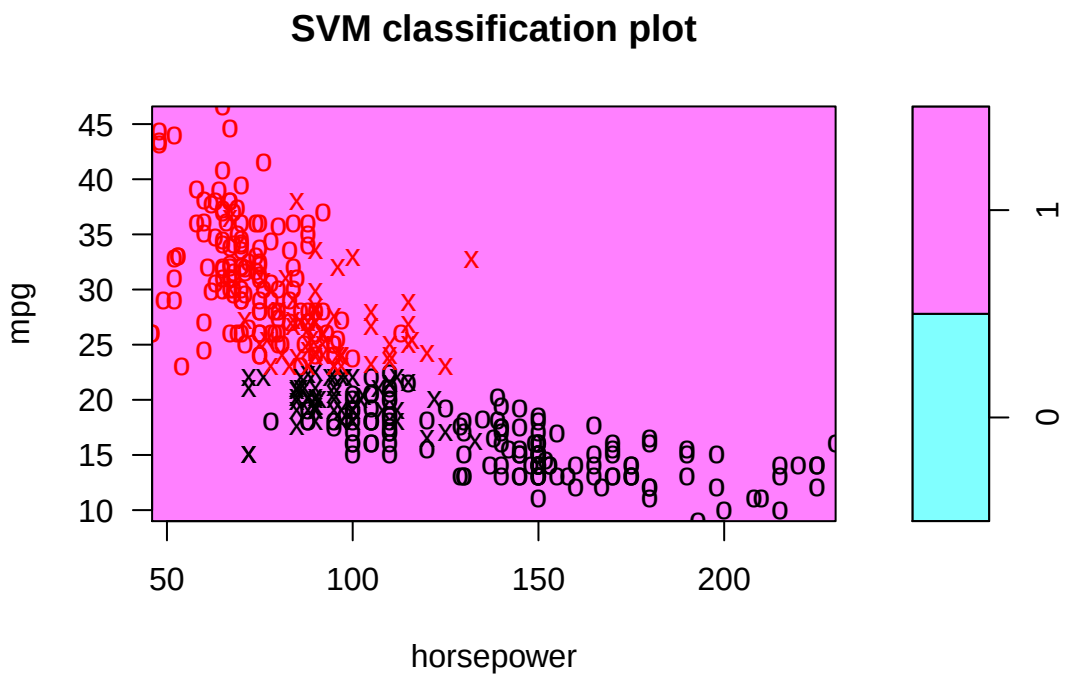

```
# Radial plots
plot(svm_radial, auto_data, mpg ~ acceleration, main = "Radial: mpg~acceleration",
     slice = list(displacement = mean_displacement, horsepower = mean_horsepower))
```



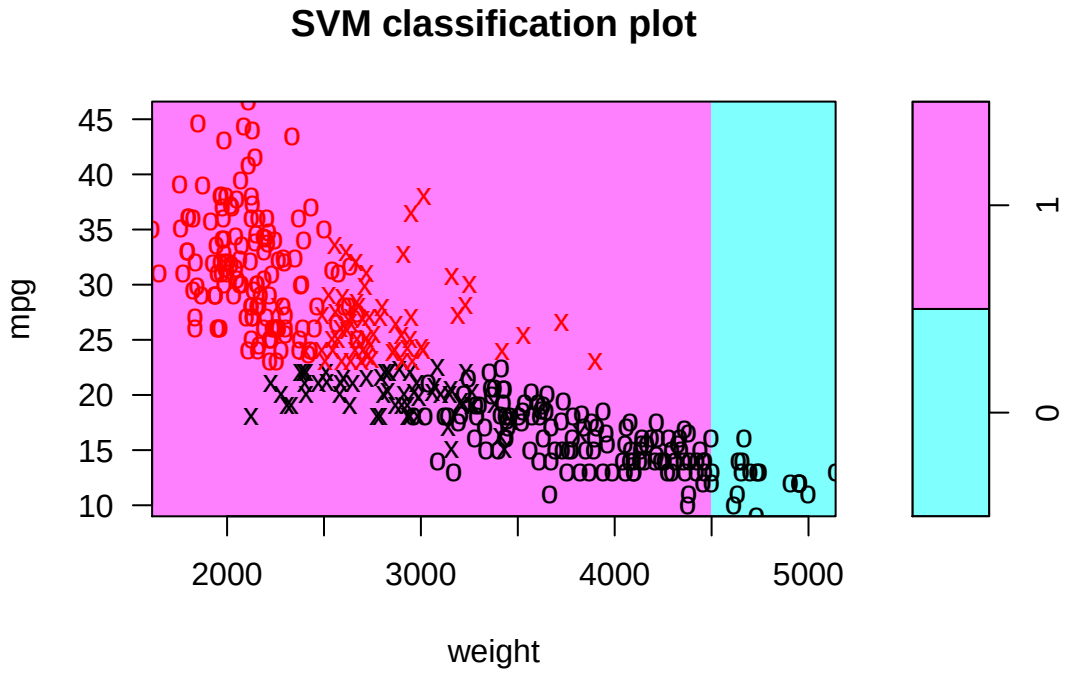

```
plot(svm_radial, auto_data, mpg ~ displacement, main = "Radial: mpg~displacement",
     slice = list(acceleration = mean_acceleration, horsepower = mean_horsepower))
```



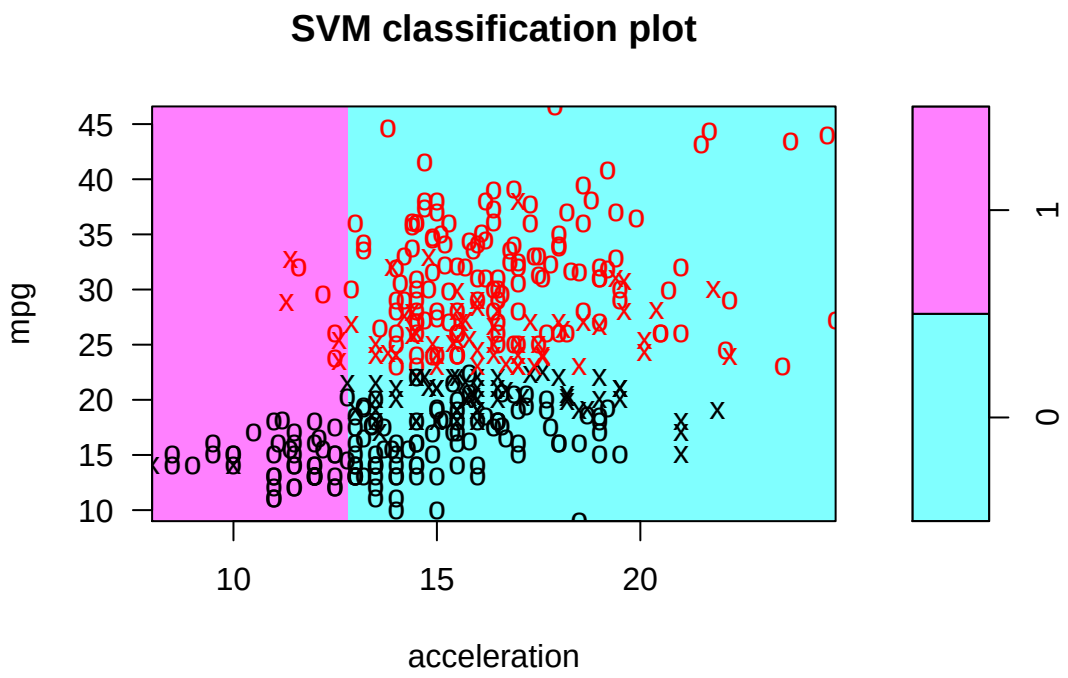

```
plot(svm_radial, auto_data, mpg ~ horsepower, main = "Radial: mpg~horsepower",
     slice = list(acceleration = mean_acceleration, displacement = mean_displacement))
```



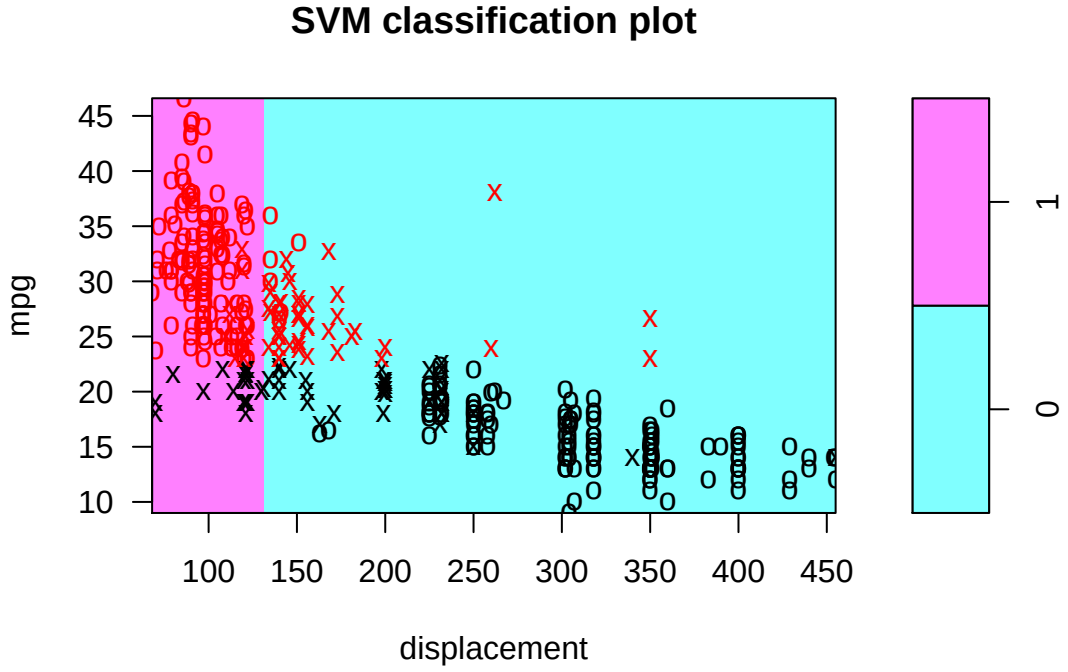

```
plot(svm_radial, auto_data, mpg ~ weight, main = "Radial: mpg~weight",
     slice = list(acceleration = mean_acceleration, displacement = mean_displacement))
```



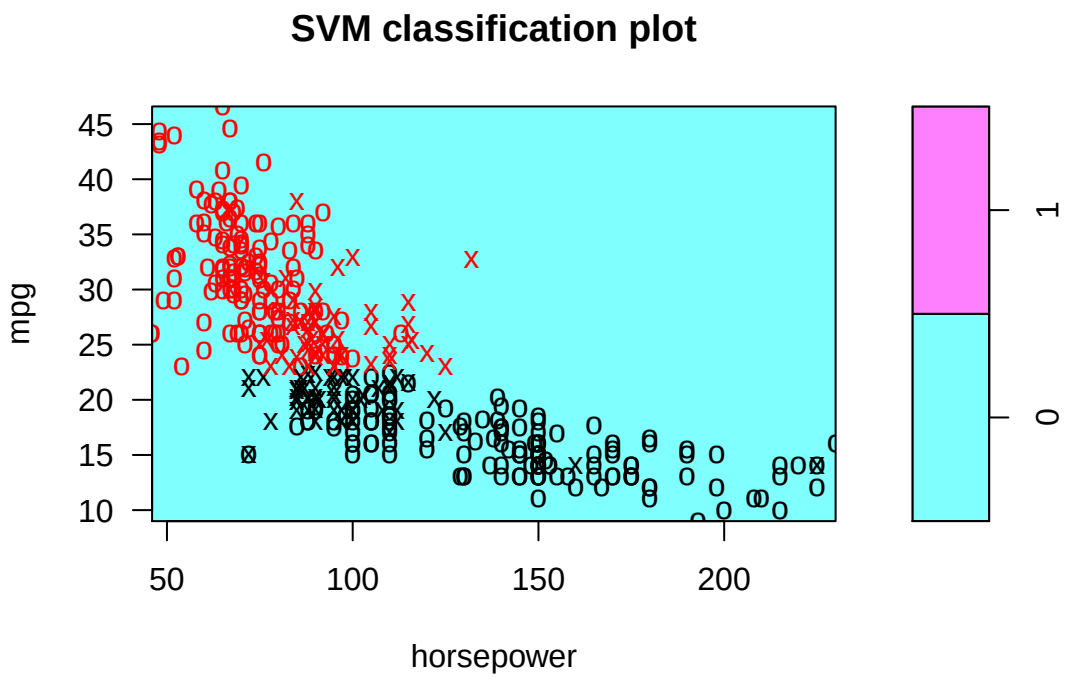

```
# Polynomial
plot(svm_poly, auto_data, mpg ~ acceleration, main = "Polynomial: mpg~acceleration",
     slice = list(displacement = mean_displacement, horsepower = mean_horsepower))
```




```
plot(svm_poly, auto_data, mpg ~ displacement, main = "Polynomial: mpg~displacement",
     slice = list(acceleration = mean_acceleration, horsepower = mean_horsepower))
```




```
plot(svm_poly, auto_data, mpg ~ horsepower, main = "Polynomial: mpg~horsepower",
     slice = list(acceleration = mean_acceleration, displacement = mean_displacement))
```




```
plot(svm_poly, auto_data, mpg ~ weight, main = "Polynomial: mpg~weight",
     slice = list(acceleration = mean_acceleration, displacement = mean_displacement))
```