homework6

Sunny Lee

3/23/2021

7)

```
#test
?plot.svm
```

```
## No documentation for 'plot.svm' in specified packages and libraries:
## you could try '??plot.svm'
```

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.0.4
```

```
auto <- read.csv("Auto.csv")
```

a)

```
library(ISLR)
gas.med = median(Auto$mpg)
new.var = ifelse(Auto$mpg > gas.med, 1, 0)
Auto$mpg1 = as.factor(new.var)
```

b)

```
tune.lin <- tune(svm, mpg1 ~ ., data = Auto, kernel = "linear", ranges = list(cost = c(0.01, 0.1, 1, 5,
summary(tune.lin)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.01019231
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-02 0.07897436 0.05705152
## 2 1e-01 0.05089744 0.05107437
## 3 1e+00 0.01019231 0.01786828
## 4 5e+00 0.01025641 0.01792836
## 5 1e+01 0.01782051 0.01724506
## 6 1e+02 0.03057692 0.02010376
```

From the errors we get above, we find that the svm does best at cost = 1 as it has the lowest error and going lower or higher increases the CV error.

c)

```
tune.rad <- tune(svm, mpg1~., data = Auto, kernel = "radial", ranges = list(cost = c(0.1, 1, 5, 10), gam
summary(tune.rad)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##    10   0.1
##
## - best performance: 0.02044872
##
## - Detailed performance results:
##    cost gamma      error dispersion
## 1   0.1 1e-02 0.08660256 0.03596154
## 2   1.0 1e-02 0.07628205 0.03516370
## 3   5.0 1e-02 0.05333333 0.02967825
## 4  10.0 1e-02 0.03294872 0.02648056
## 5   0.1 1e-01 0.07634615 0.03536754
## 6   1.0 1e-01 0.06102564 0.03352019
## 7   5.0 1e-01 0.02807692 0.02537217
## 8  10.0 1e-01 0.02044872 0.02354784
## 9   0.1 1e+00 0.55621795 0.04070617
## 10  1.0 1e+00 0.06102564 0.02658827
## 11  5.0 1e+00 0.06358974 0.02677991
## 12 10.0 1e+00 0.06358974 0.02677991
## 13  0.1 5e+00 0.55621795 0.04070617
## 14  1.0 5e+00 0.49506410 0.04553443
## 15  5.0 5e+00 0.49506410 0.04711144
## 16 10.0 5e+00 0.49506410 0.04711144
## 17  0.1 1e+01 0.55621795 0.04070617
## 18  1.0 1e+01 0.52051282 0.03972291
## 19  5.0 1e+01 0.51032051 0.04128352
## 20 10.0 1e+01 0.51032051 0.04128352
## 21  0.1 1e+02 0.55621795 0.04070617
## 22  1.0 1e+02 0.55621795 0.04070617
## 23  5.0 1e+02 0.55621795 0.04070617
## 24 10.0 1e+02 0.55621795 0.04070617
```

```
tune.pol <- tune(svm, mpg1~., data = Auto, kernel = "polynomial", ranges = list(cost = c(0.1, 1, 5, 10)
summary(tune.pol)
```

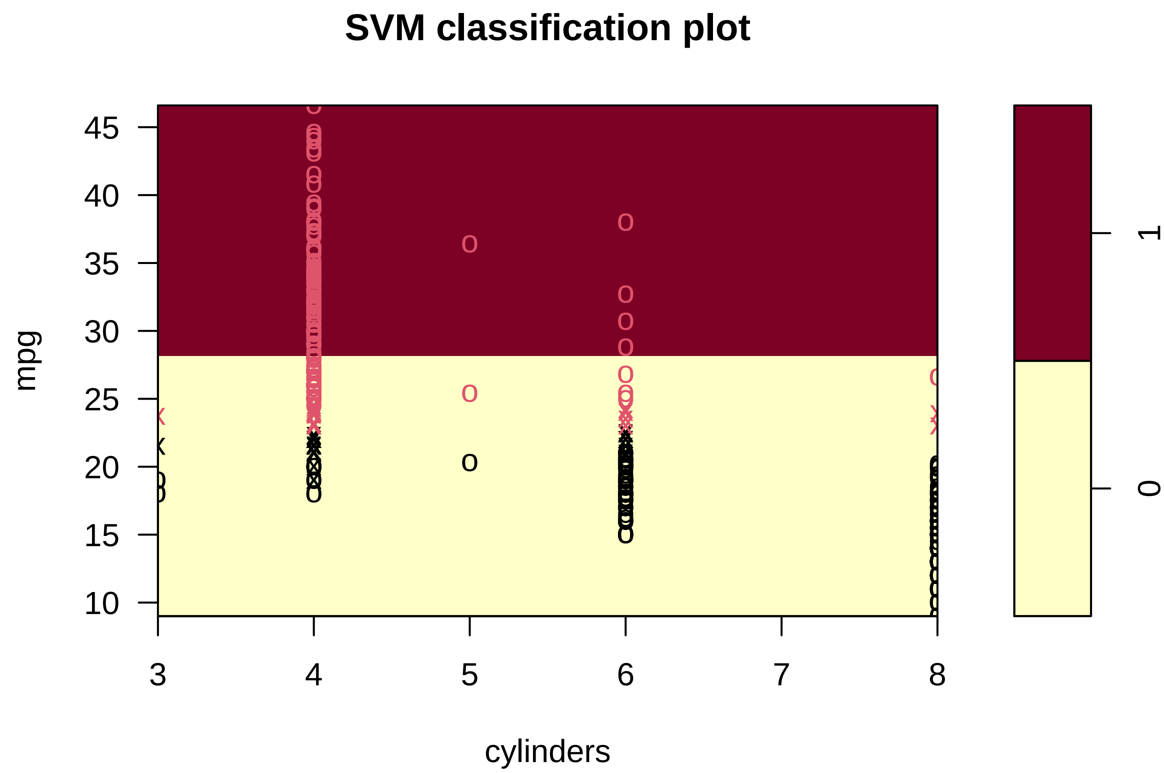
```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##    10      2
```

```
##
## - best performance: 0.5357051
##
## - Detailed performance results:
##    cost degree     error dispersion
## 1   0.1      2 0.5740385 0.07190877
## 2   1.0      2 0.5740385 0.07190877
## 3   5.0      2 0.5740385 0.07190877
## 4  10.0      2 0.5357051 0.12425909
## 5   0.1      3 0.5740385 0.07190877
## 6   1.0      3 0.5740385 0.07190877
## 7   5.0      3 0.5740385 0.07190877
## 8  10.0      3 0.5740385 0.07190877
## 9   0.1      4 0.5740385 0.07190877
## 10  1.0      4 0.5740385 0.07190877
## 11  5.0      4 0.5740385 0.07190877
## 12 10.0      4 0.5740385 0.07190877
```

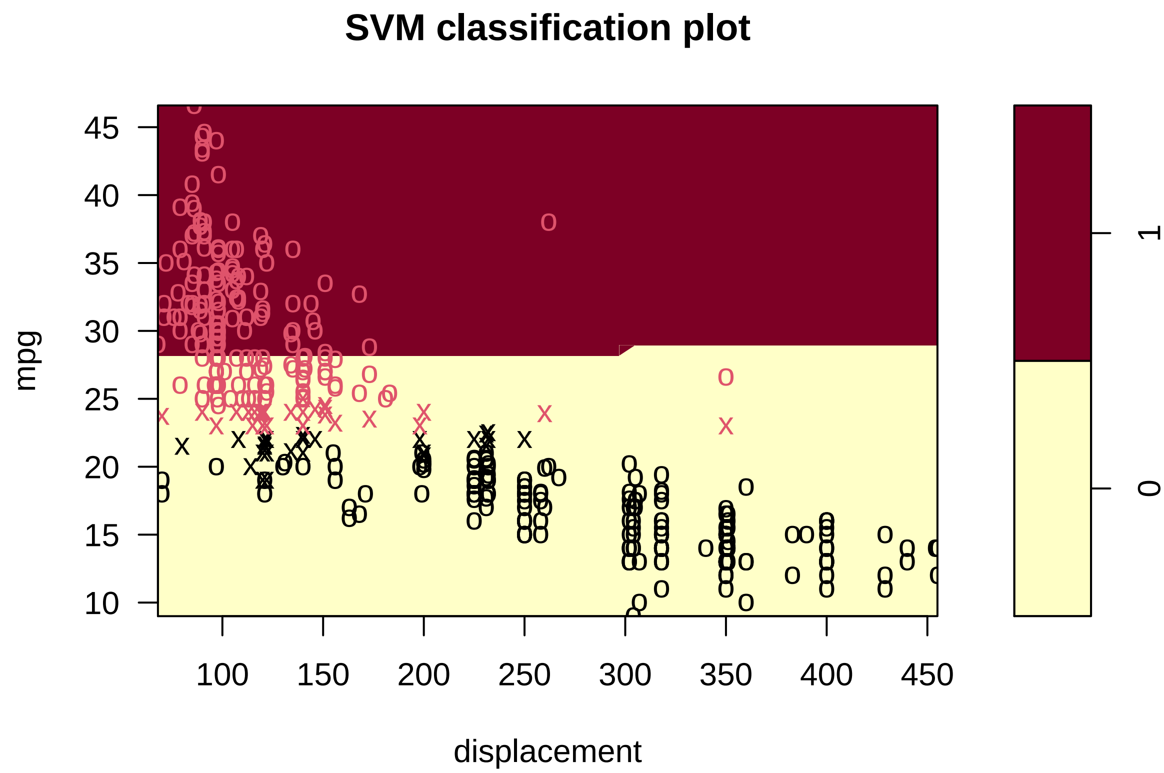
Looking at the summaries for the radial and polynomial tuned svms, we find a cost of 10 and a gamma of .01 works best for the radial and a cost of 10 and a degree of 2 works best for the polynomial svm. This means we must allow for more misclassifications for both our radial and polynomial svms. Comparing the errors for the cross validations of the three svms, we also see that the linear model has the lowest cross validation error, and thus we should see our linear model perform the best out of the three svms.

d)

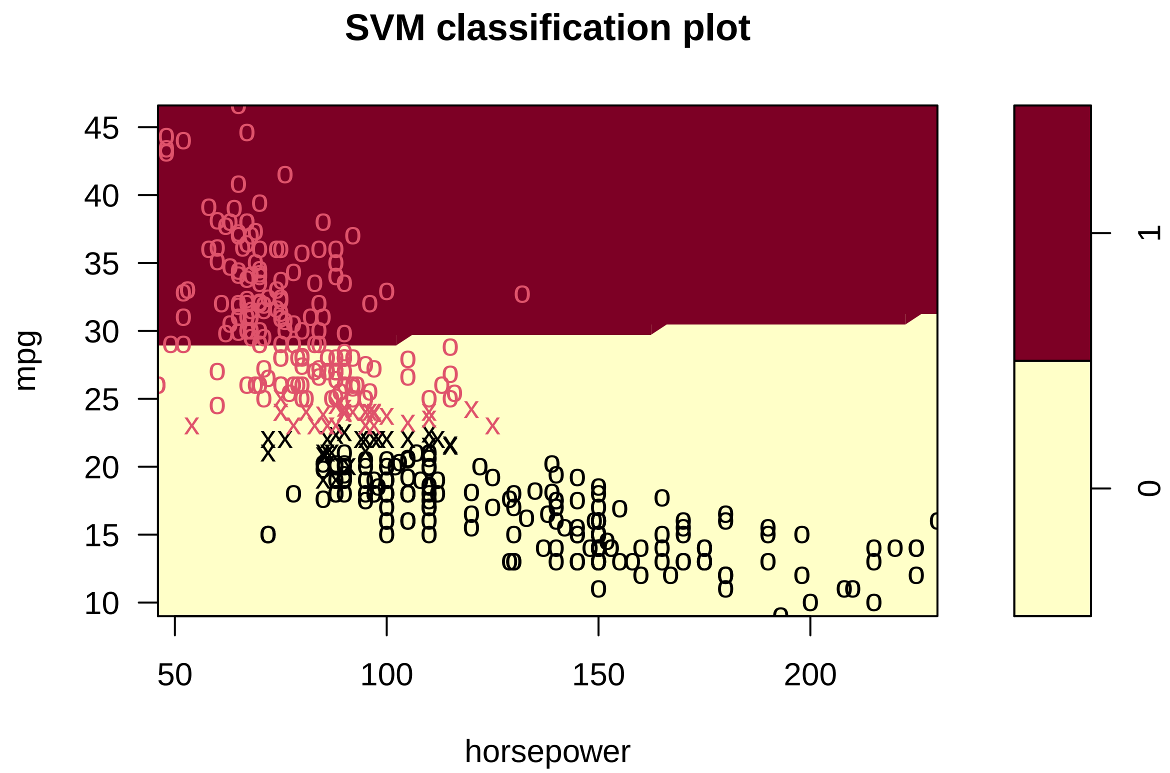
```
svm.lin <- tune.lin$best.model
svm.pol <- tune.pol$best.model
svm.rad <- tune.rad$best.model

plot(svm.lin, Auto, mpg~cylinders)
```



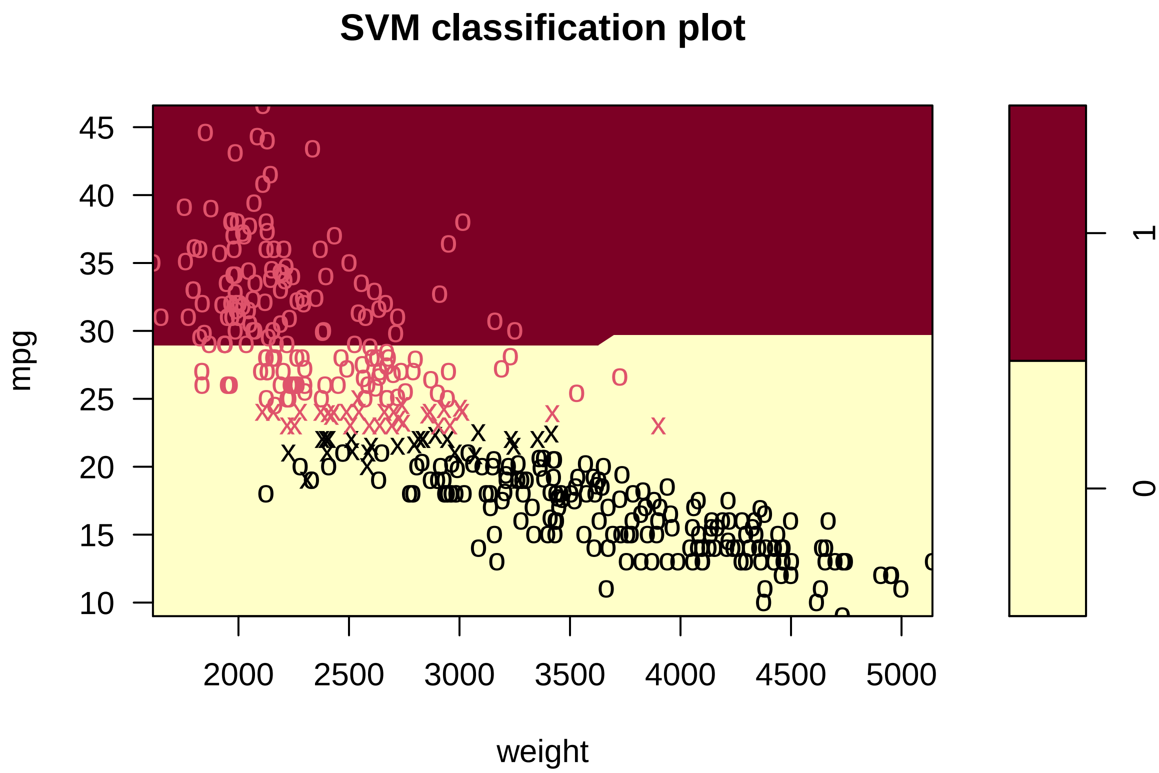

```
plot(svm.lin, Auto, mpg~displacement)
```




```
plot(svm.lin, Auto, mpg~horsepower)
```

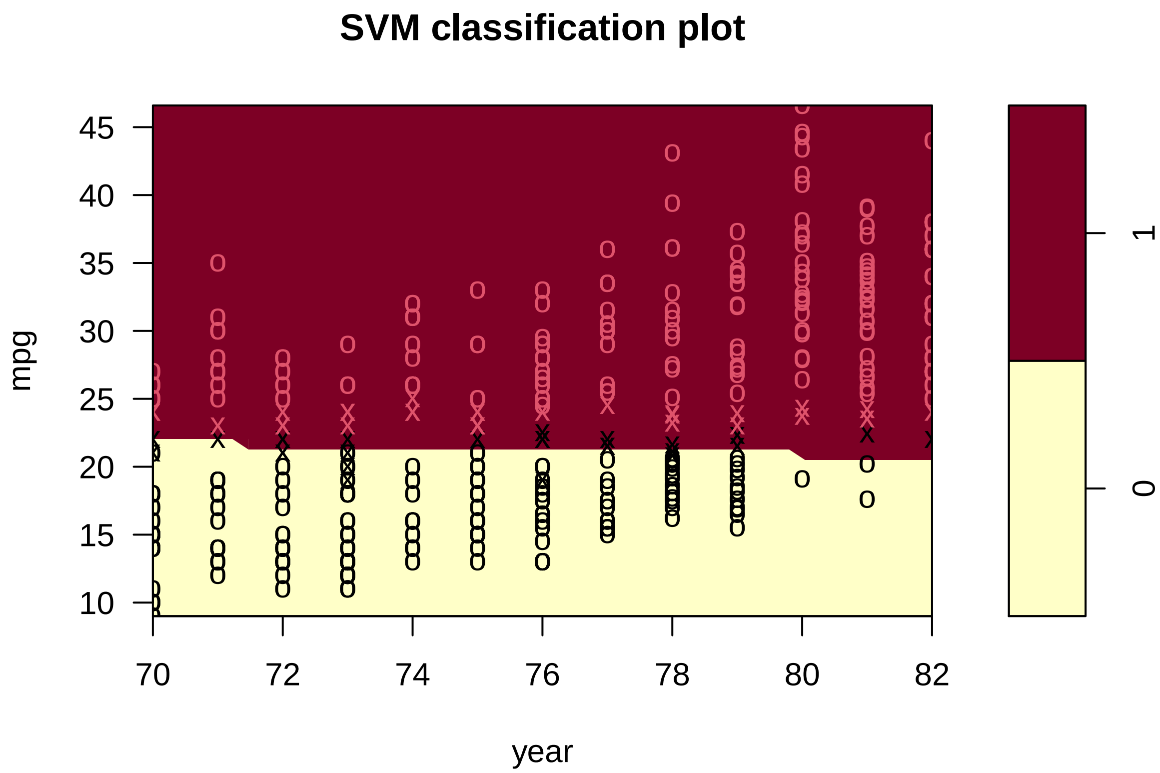



```
plot(svm.lin, Auto, mpg~weight)
```

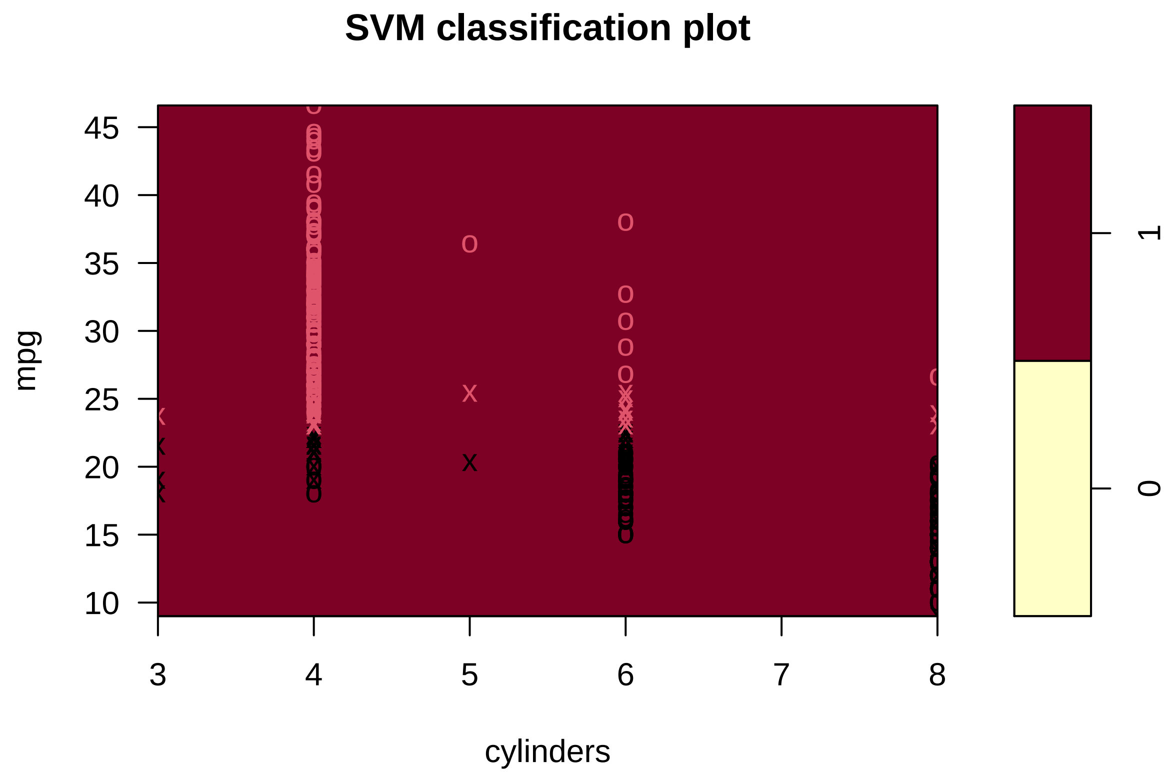



```
plot(svm.lin, Auto, mpg~acceleration)
```

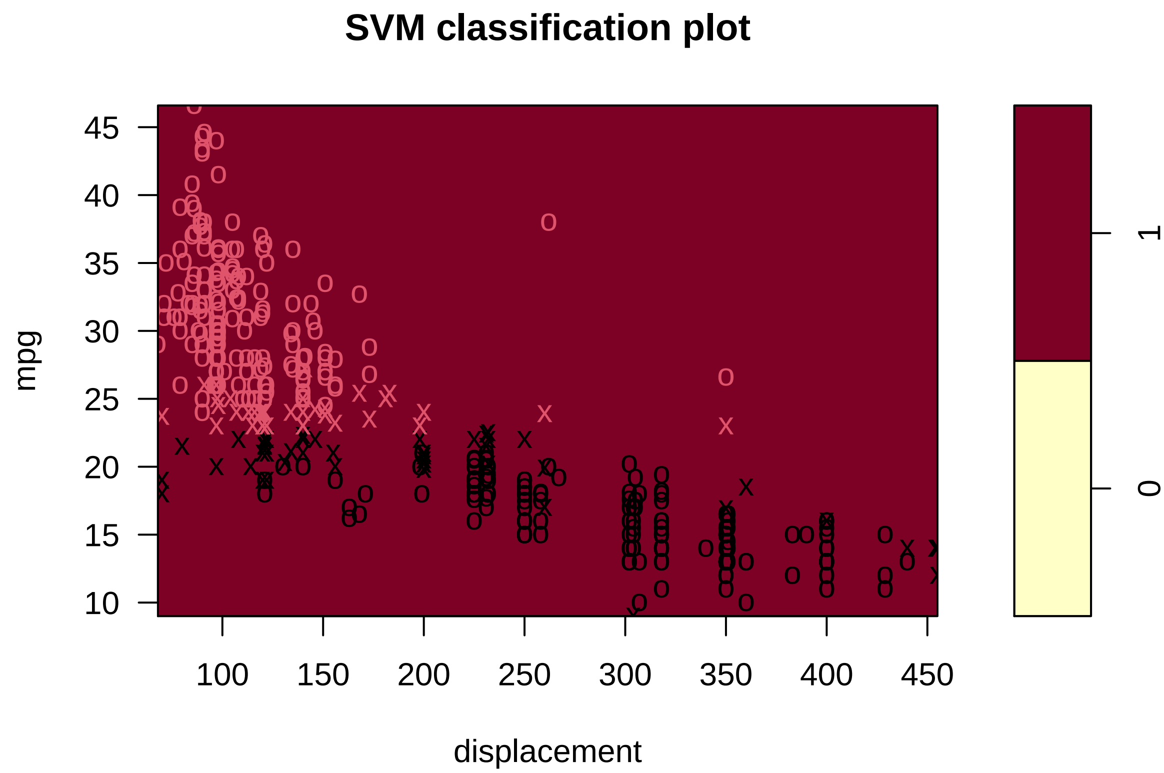

```
plot(svm.lin, Auto, mpg~year)
```



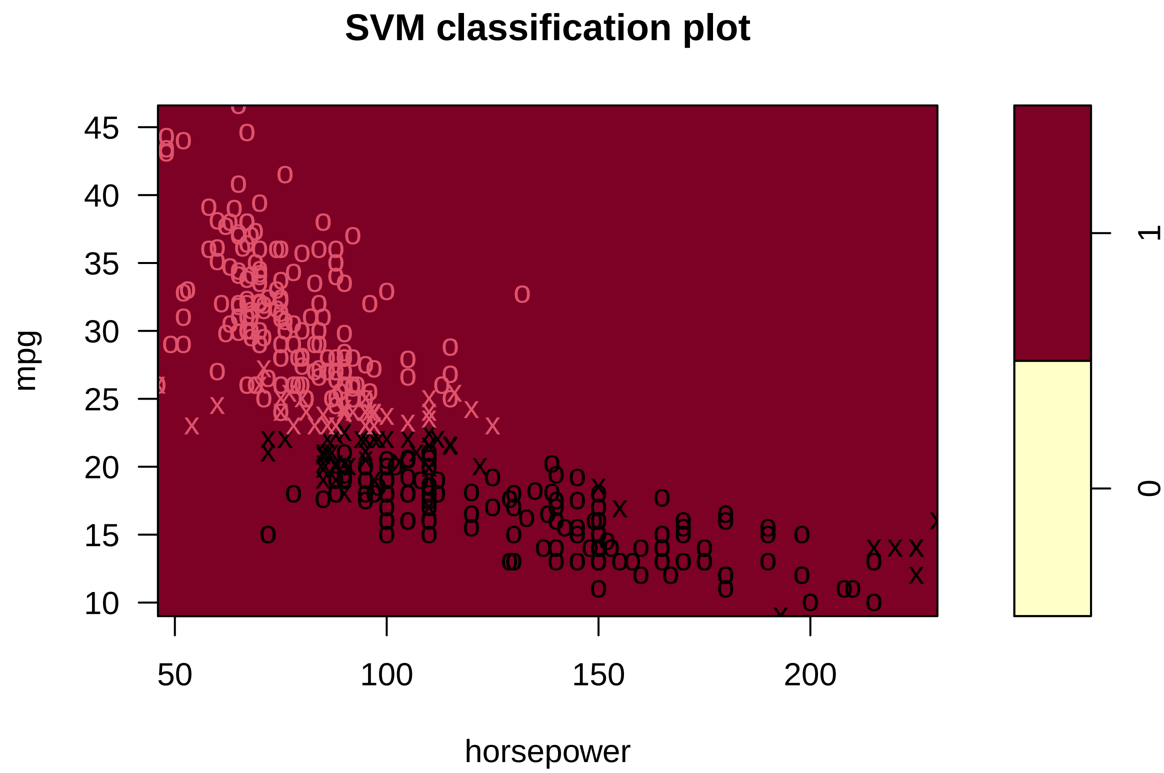

```
plot(svm.rad, Auto, mpg~cylinders)
```



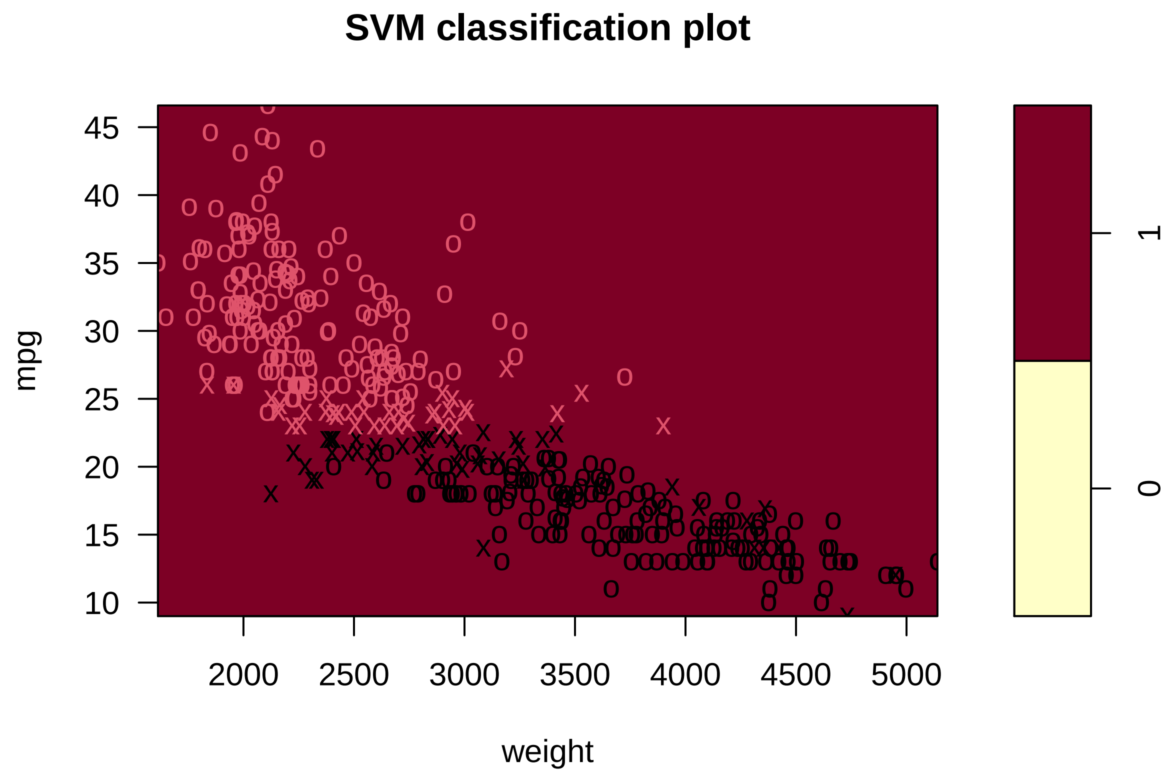

```
plot(svm.rad, Auto, mpg~displacement)
```



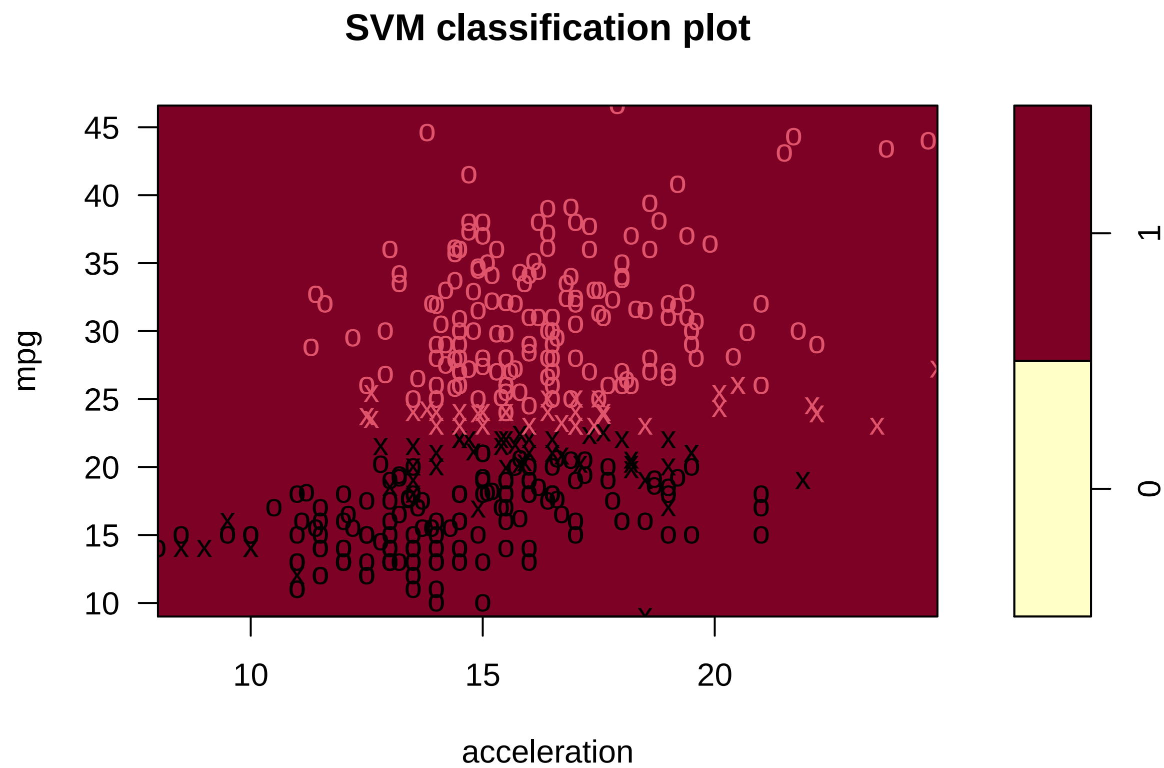

```
plot(svm.rad, Auto, mpg~horsepower)
```



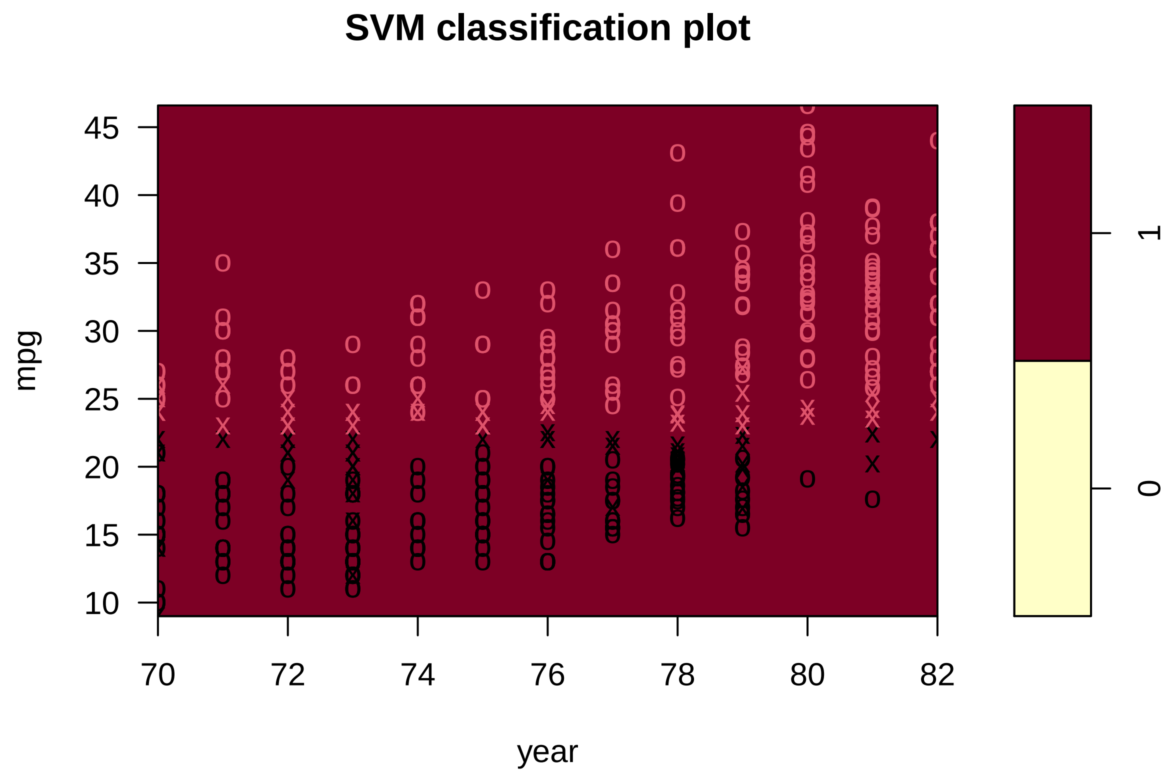

```
plot(svm.rad, Auto, mpg~weight)
```



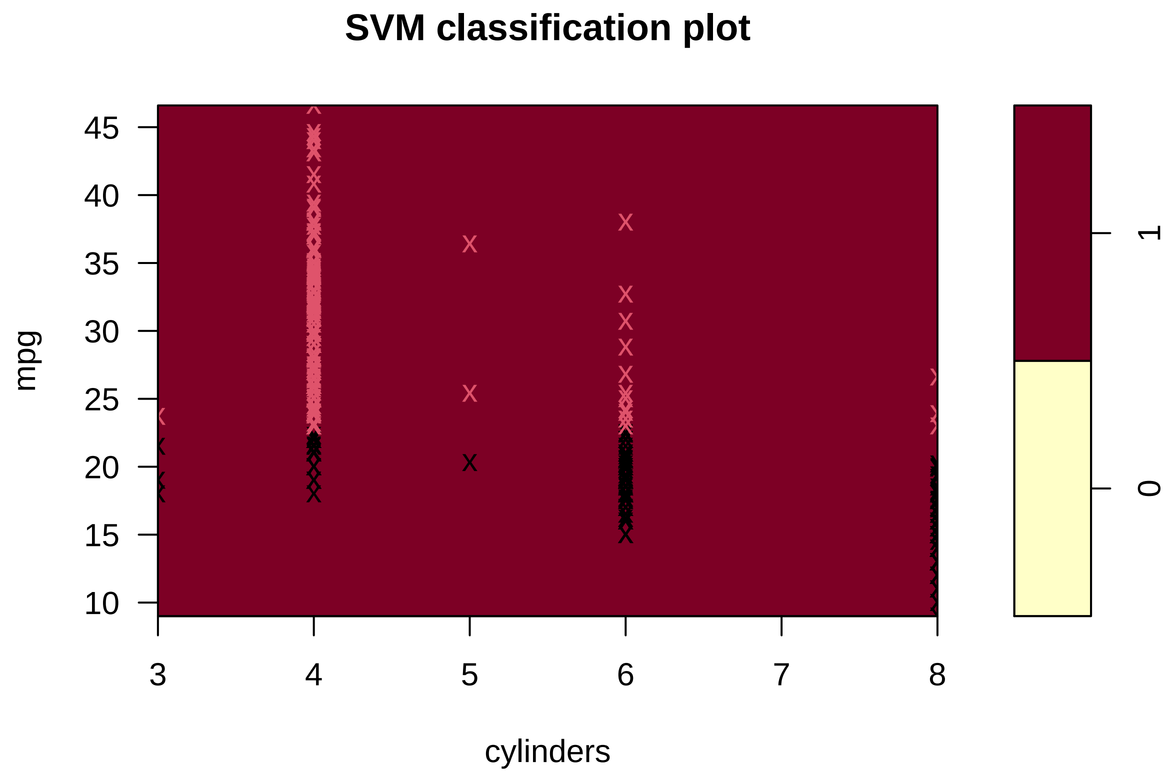

```
plot(svm.rad, Auto, mpg~acceleration)
```



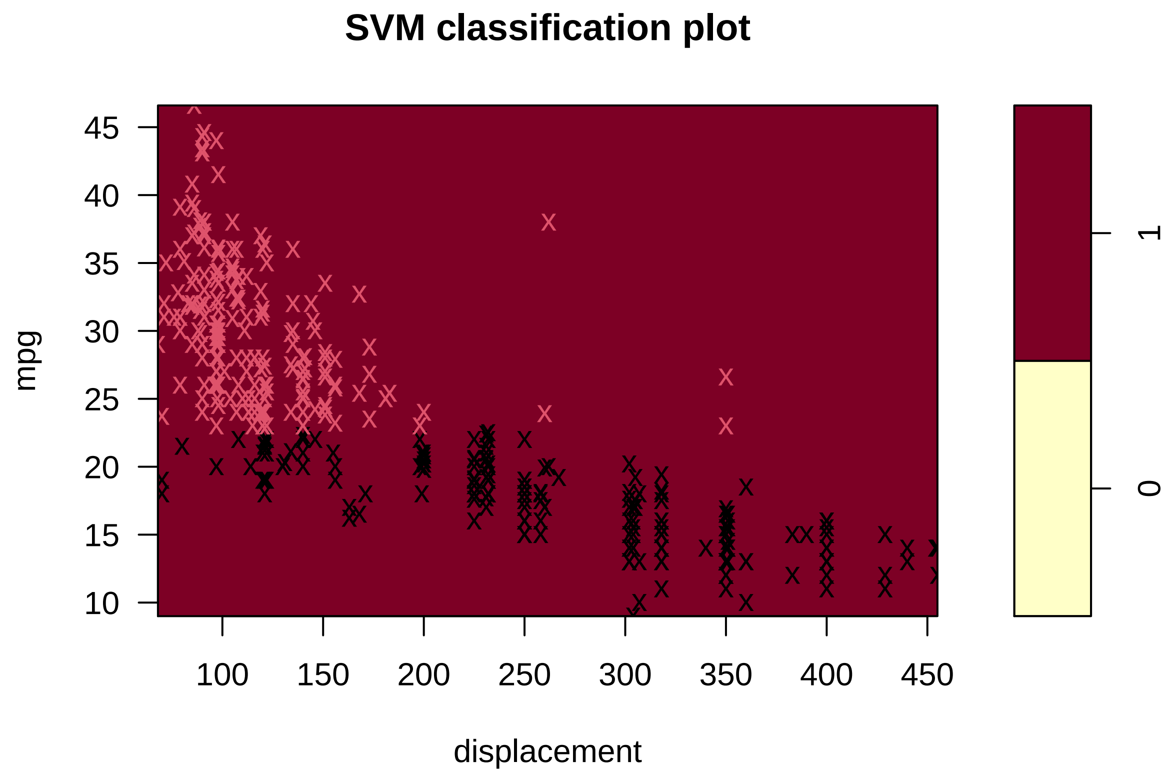

```
plot(svm.rad, Auto, mpg~year)
```



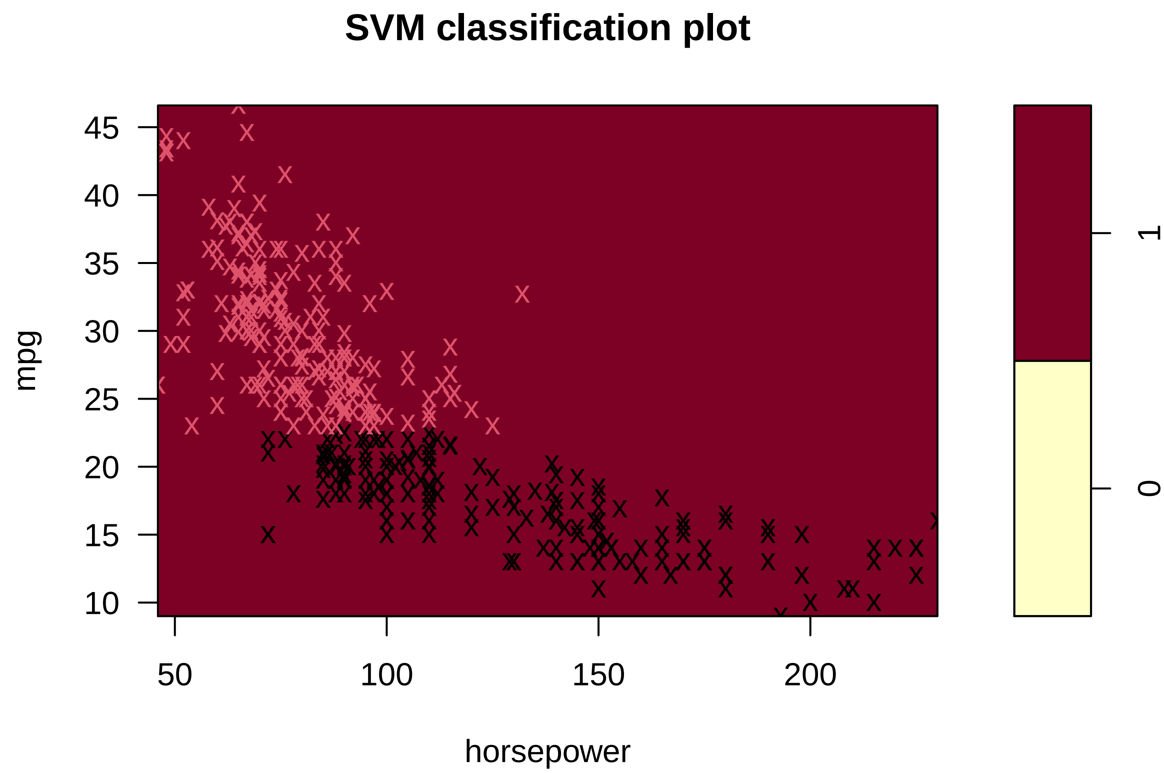

```
plot(svm.pol, Auto, mpg~cylinders)
```



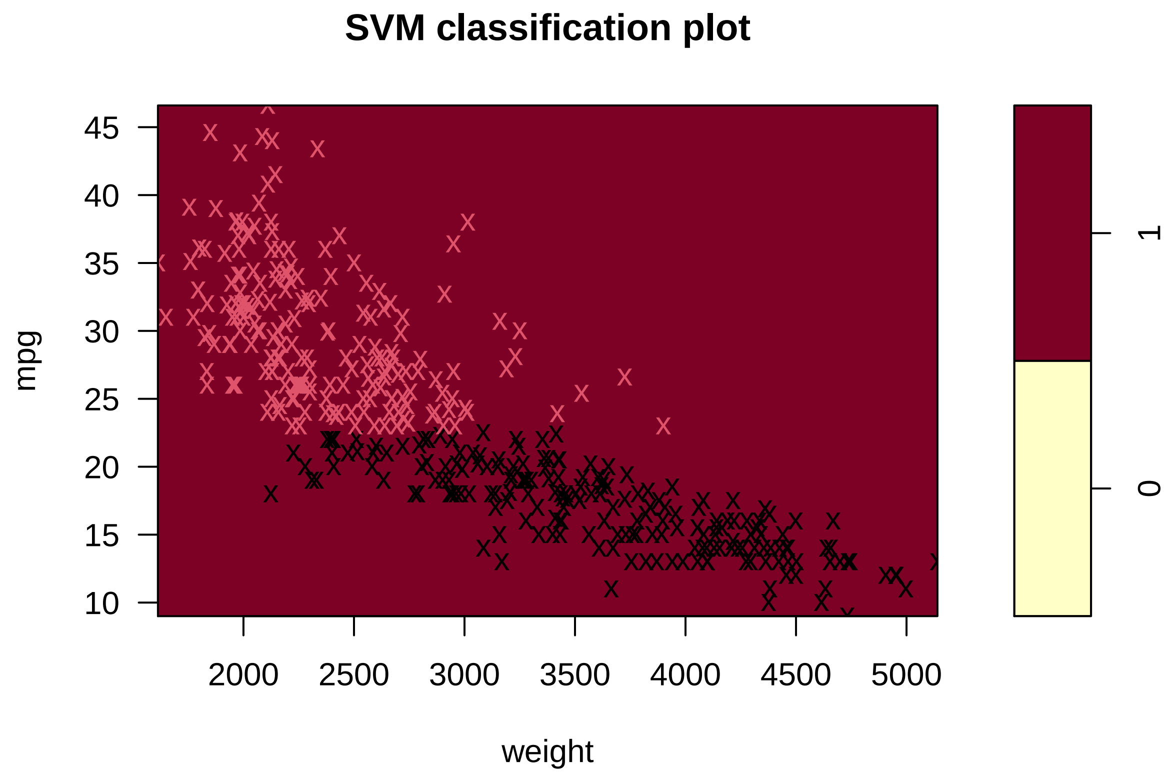

```
plot(svm.pol, Auto, mpg~displacement)
```




```
plot(svm.pol, Auto, mpg~horsepower)
```

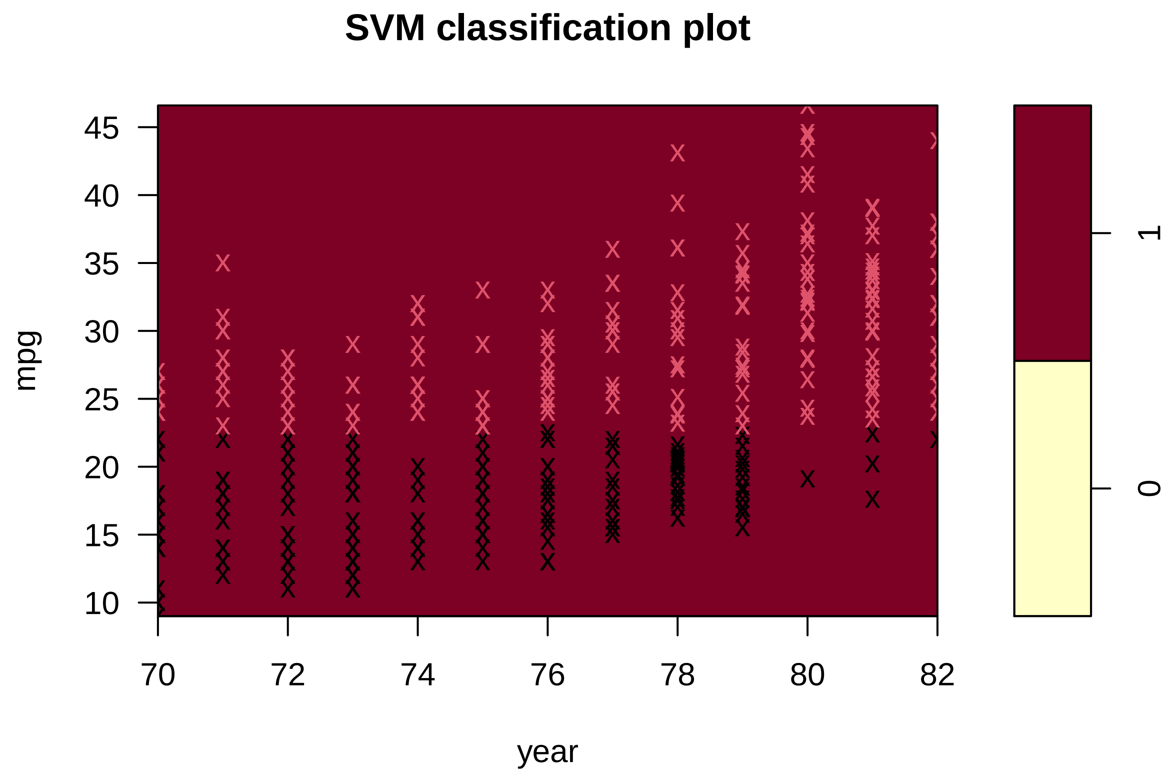



```
plot(svm.pol, Auto, mpg~weight)
```




```
plot(svm.pol, Auto, mpg~acceleration)
```


```
plot(svm.pol, Auto, mpg~year)
```



From the graphs above, we see our prediction was right. The linear model is splitting the data quite well. However, the graphs from the radial and polynomial models we see the whole dataset is being classified as one class, meaning our svms are not even classifying at all. Thus, our radial and polynomial svms are useless and the only one we can use is the linear kernel.