SDS Homework 7

Matthew Bradley, Ayanna Fisher, Hayley Zorkic, Ben Howell

4/29/2022

Question 7:

a:

```
library(ISLR)
library(e1071)
data(Auto)

Auto$gasMedian <- ifelse(Auto$mpg > median(Auto$mpg), 1, 0)
```

b:

```
# set seed so markdown output is the same each time
set.seed(1)
Auto$gasMedian = as.factor(Auto$gasMedian)
newAuto = Auto[2:10]
tune_out = tune(svm, gasMedian ~ ., kernel = "linear", data = newAuto, ranges = list(cost = c(.001, .01
summary(tune_out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.08673077
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-03 0.13525641 0.05661708
## 2 1e-02 0.08923077 0.04698309
## 3 1e-01 0.08673077 0.04040897
## 4 1e+00 0.09961538 0.04923181
## 5 5e+00 0.11230769 0.05826857
## 6 1e+01 0.11237179 0.05701890
## 7 1e+02 0.11750000 0.06208951
## 8 1e+03 0.13525641 0.06613861
```

The tune function uses 10-fold cross validation to find the linear svm with the best cost. The best linear svm has a cost of 0.01, which has the lowest error at 0.0867. For all of the fitted costs, the errors were below 0.14. Dispersions are all similar, between 0.05 and 0.06.

c:

```
set.seed(1)
tune_outRadial = tune(svm, gasMedian ~ ., data = newAuto, kernel = "radial", ranges = list(cost = c(.00

tune_outPoly = tune(svm, gasMedian ~ ., data = newAuto, kernel = "polynomial", ranges = list(cost = c(.0

summary(tune_outRadial)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##    10     1
##
## - best performance: 0.07897436
##
## - Detailed performance results:
##     cost gamma      error dispersion
## 1  1e-03  0.01 0.55115385 0.04366593
## 2  1e-02  0.01 0.55115385 0.04366593
## 3  1e-01  0.01 0.11224359 0.03836937
## 4  1e+00  0.01 0.08673077 0.04551036
## 5  5e+00  0.01 0.08673077 0.04040897
## 6  1e+01  0.01 0.08673077 0.03855882
## 7  1e+02  0.01 0.09692308 0.05742483
## 8  1e-03  0.10 0.55115385 0.04366593
## 9  1e-02  0.10 0.26564103 0.10022083
## 10 1e-01  0.10 0.08666667 0.04193895
## 11 1e+00  0.10 0.08923077 0.04376306
## 12 5e+00  0.10 0.08423077 0.04689205
## 13 1e+01  0.10 0.08416667 0.05256241
## 14 1e+02  0.10 0.10211538 0.04535762
## 15 1e-03  1.00 0.55115385 0.04366593
## 16 1e-02  1.00 0.55115385 0.04366593
## 17 1e-01  1.00 0.55115385 0.04366593
## 18 1e+00  1.00 0.07903846 0.04891067
## 19 5e+00  1.00 0.08147436 0.04910668
## 20 1e+01  1.00 0.07897436 0.04869339
## 21 1e+02  1.00 0.07897436 0.04869339
## 22 1e-03  5.00 0.55115385 0.04366593
## 23 1e-02  5.00 0.55115385 0.04366593
## 24 1e-01  5.00 0.55115385 0.04366593
## 25 1e+00  5.00 0.48967949 0.05080301
## 26 5e+00  5.00 0.48211538 0.05914633
## 27 1e+01  5.00 0.48211538 0.05914633
## 28 1e+02  5.00 0.48211538 0.05914633
## 29 1e-03 10.00 0.55115385 0.04366593
## 30 1e-02 10.00 0.55115385 0.04366593
## 31 1e-01 10.00 0.55115385 0.04366593
## 32 1e+00 10.00 0.51794872 0.04766442
## 33 5e+00 10.00 0.51794872 0.04766442
## 34 1e+01 10.00 0.51794872 0.04766442
## 35 1e+02 10.00 0.51794872 0.04766442
```

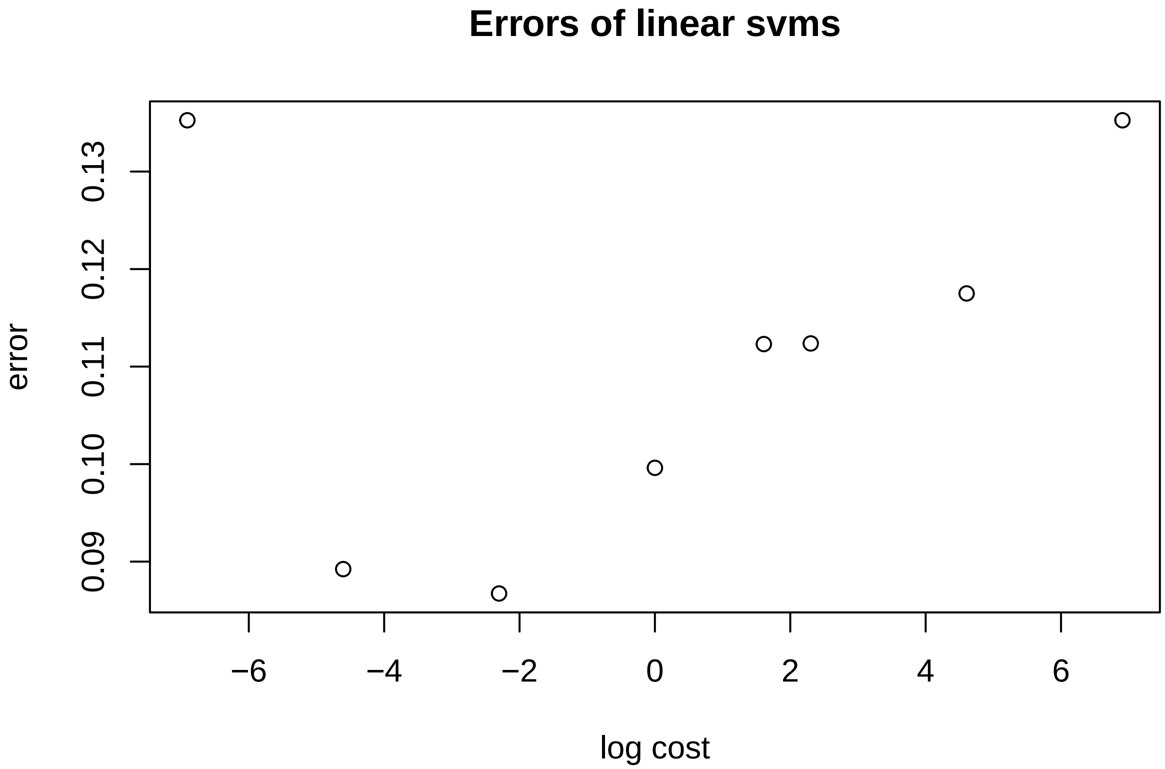
```r
summary(tune_outPoly)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##   100      1
##
## - best performance: 0.08403846
##
## - Detailed performance results:
##     cost degree      error dispersion
## 1  1e-03      1 0.60192308 0.06346118
## 2  1e-02      1 0.60192308 0.06346118
## 3  1e-01      1 0.36461538 0.09994785
## 4  1e+00      1 0.10705128 0.07510919
## 5  5e+00      1 0.08910256 0.05253109
## 6  1e+01      1 0.08653846 0.04956160
## 7  1e+02      1 0.08403846 0.06254843
## 8  1e+03      1 0.09698718 0.07021827
## 9  1e-03      2 0.60192308 0.06346118
## 10 1e-02      2 0.60192308 0.06346118
## 11 1e-01      2 0.60192308 0.06346118
## 12 1e+00      2 0.60192308 0.06346118
## 13 5e+00      2 0.60192308 0.06346118
## 14 1e+01      2 0.58416667 0.07806609
## 15 1e+02      2 0.31634615 0.07262899
## 16 1e+03      2 0.29346154 0.07790084
## 17 1e-03      3 0.60192308 0.06346118
## 18 1e-02      3 0.60192308 0.06346118
## 19 1e-01      3 0.60192308 0.06346118
## 20 1e+00      3 0.60192308 0.06346118
## 21 5e+00      3 0.60192308 0.06346118
## 22 1e+01      3 0.60192308 0.06346118
## 23 1e+02      3 0.44852564 0.13181806
## 24 1e+03      3 0.25750000 0.08857504
## 25 1e-03      4 0.60192308 0.06346118
## 26 1e-02      4 0.60192308 0.06346118
## 27 1e-01      4 0.60192308 0.06346118
## 28 1e+00      4 0.60192308 0.06346118
## 29 5e+00      4 0.60192308 0.06346118
## 30 1e+01      4 0.60192308 0.06346118
## 31 1e+02      4 0.60192308 0.06346118
## 32 1e+03      4 0.60192308 0.06346118
## 33 1e-03      5 0.60192308 0.06346118
## 34 1e-02      5 0.60192308 0.06346118
## 35 1e-01      5 0.60192308 0.06346118
## 36 1e+00      5 0.60192308 0.06346118
## 37 5e+00      5 0.60192308 0.06346118
## 38 1e+01      5 0.60192308 0.06346118
## 39 1e+02      5 0.60192308 0.06346118
## 40 1e+03      5 0.60192308 0.06346118
```

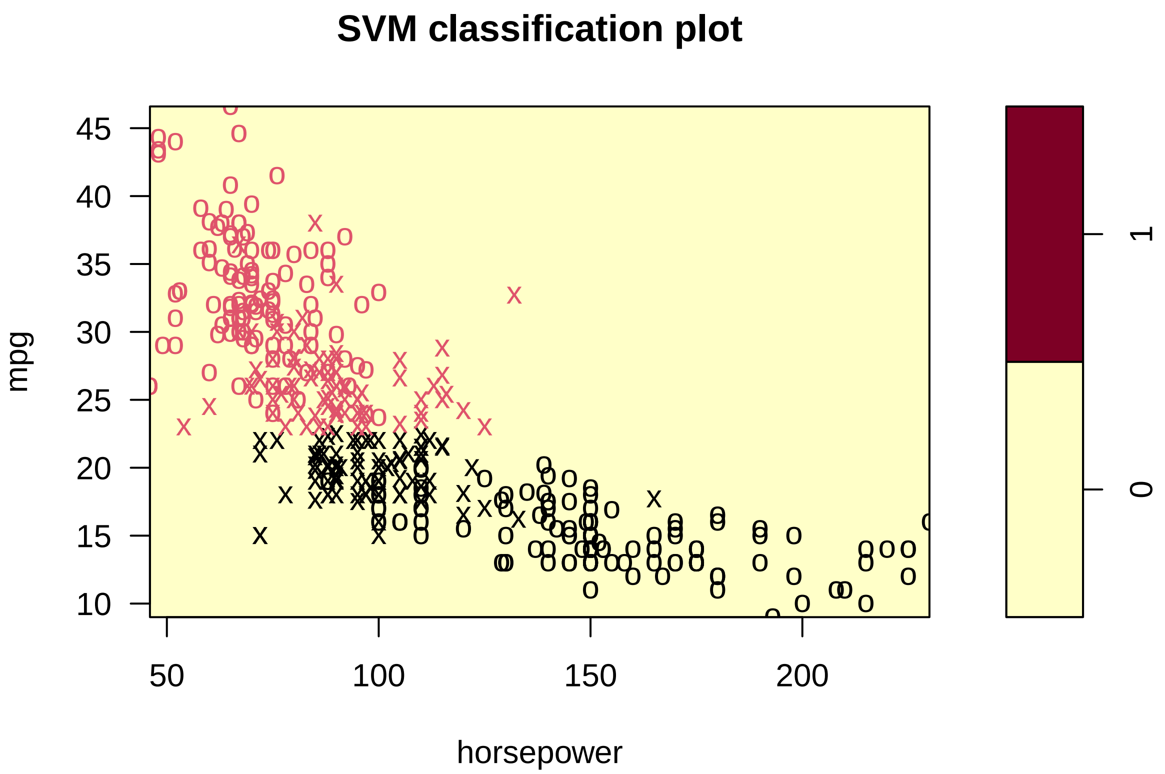
Here we used the tune function again, which is the same as we used in part b. For the radial svm model, it found the best values of cost and gamma to be 1 for each. For the polynomial model it found the best cost and degree to be 10 and 1 respectively. The lowest errors for the best radial and polynomial models are 0.07897 and 0.084038 respectively.

d:

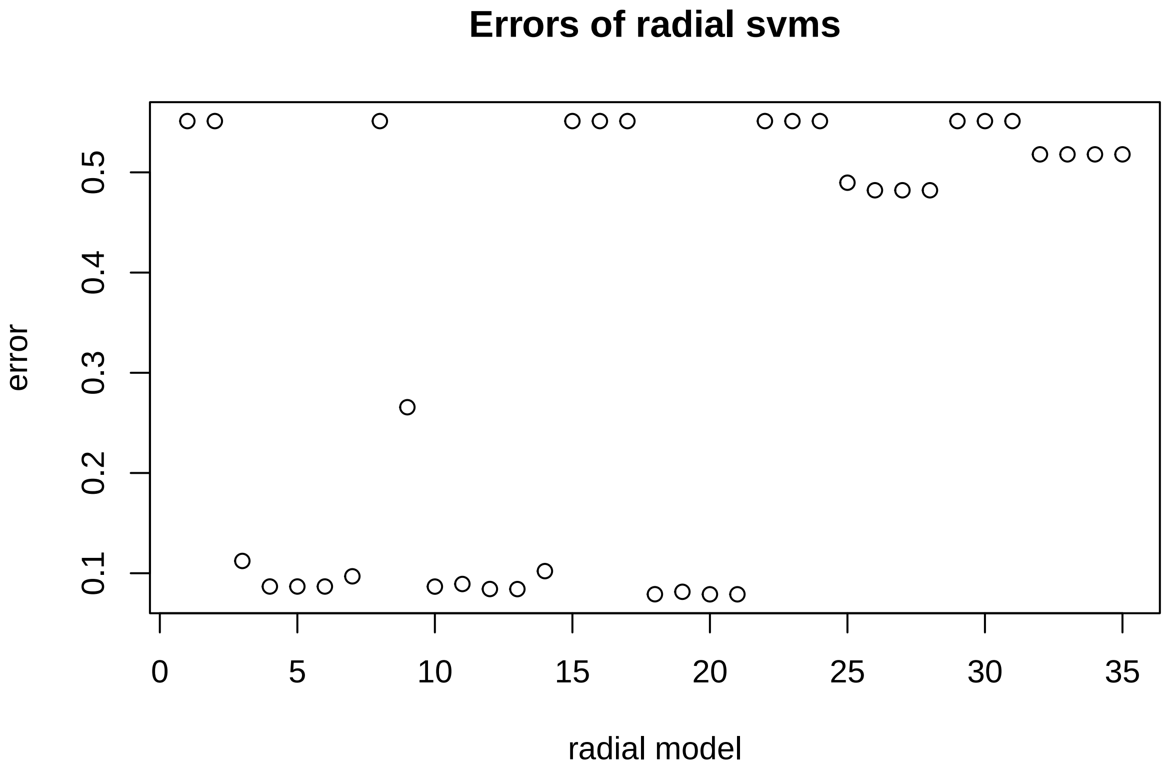
```
plot(y = tune_out$performances$error, x = log(tune_out$performances$cost), ylab = "error", xlab = "log
      main = "Errors of linear svms")
```



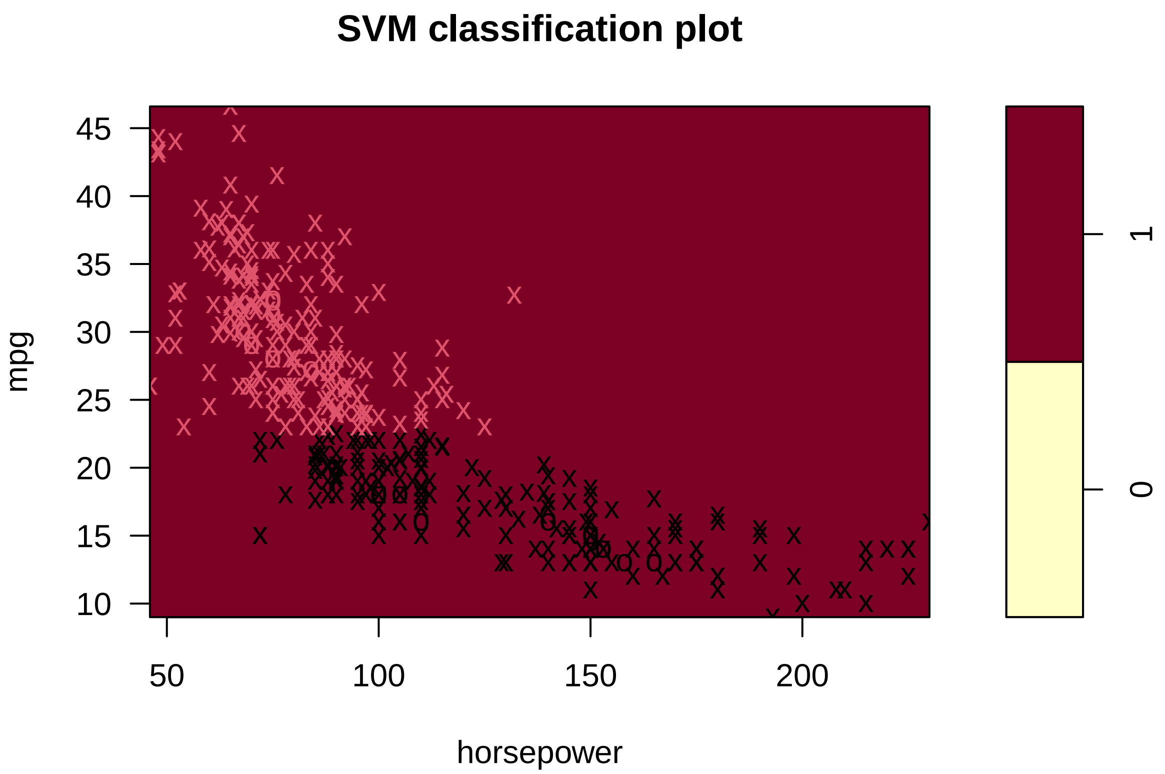
```
svmLinear <- svm(gasMedian ~ ., data = newAuto, kernel = "linear", cost = 0.01)
plot(svmLinear, Auto, mpg ~ horsepower)
```



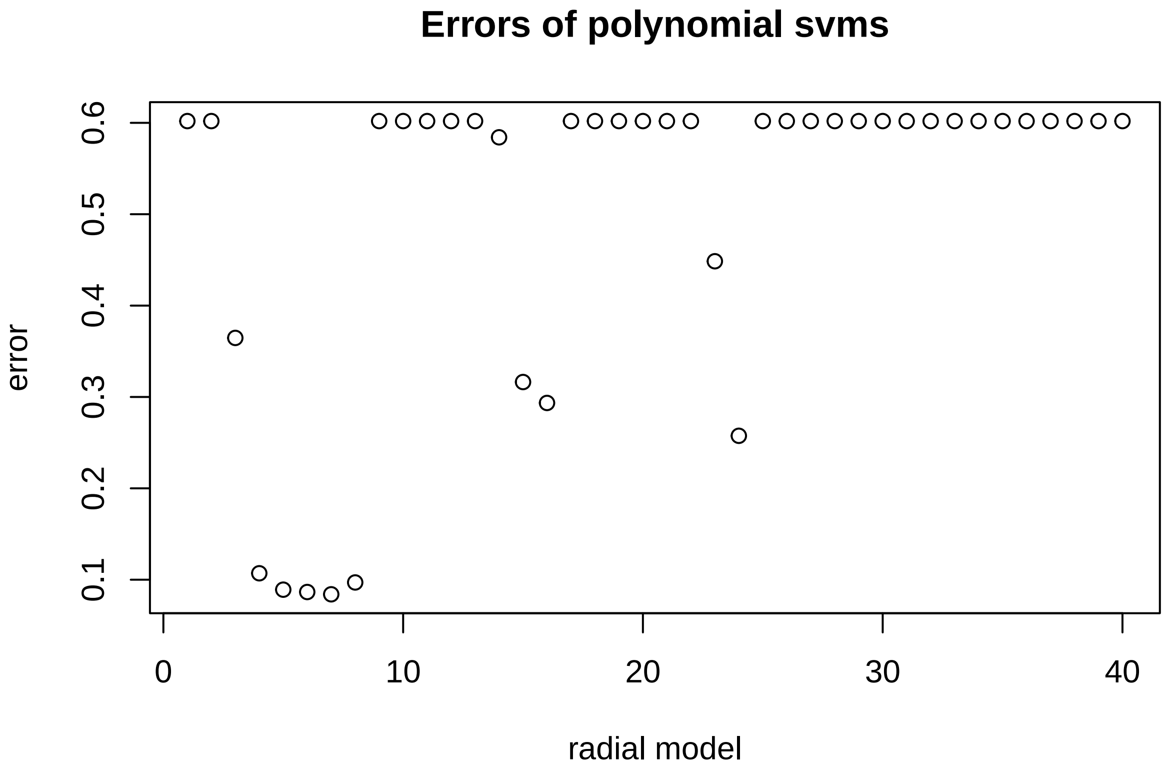
```
plot(y =  tune_outRadial$performances$error, x = 1:length(tune_outRadial$performances$error), ylab = "e
```



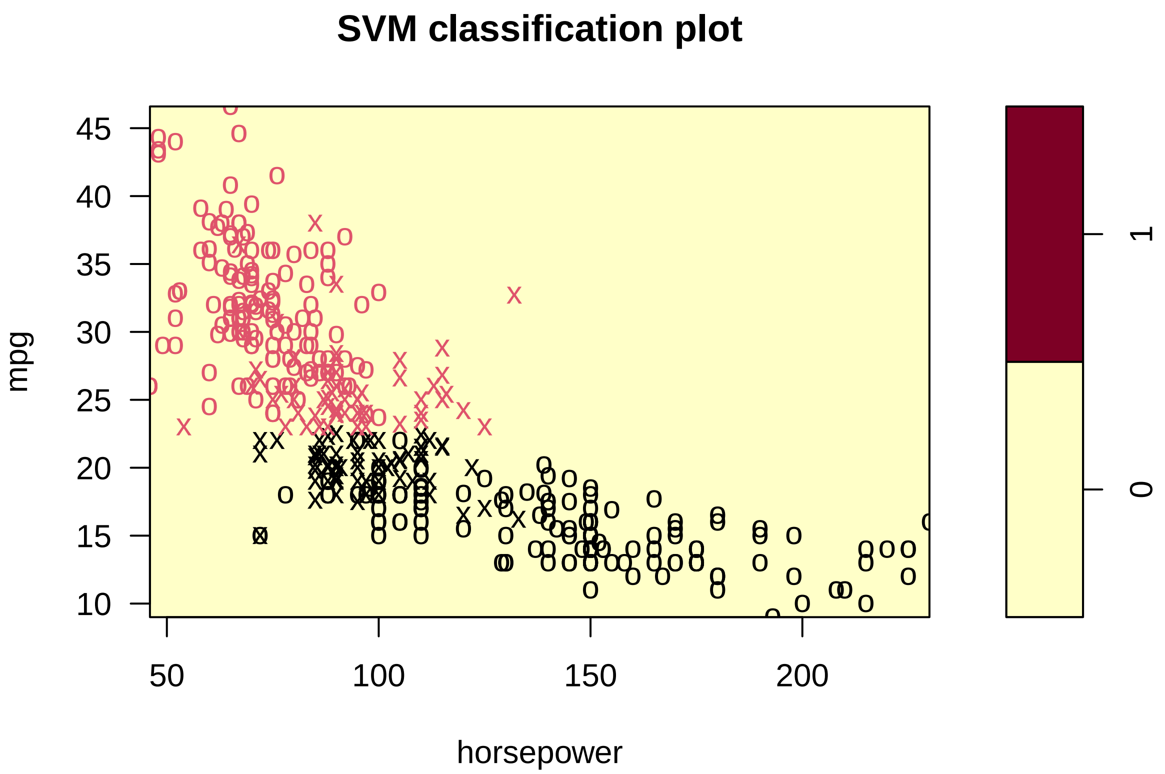
```
svmRadial <- svm(gasMedian~., data = newAuto, kernel = "radial", cost = 1, gamma = 1)
plot(svmRadial, Auto, mpg ~ horsepower)
```



```
plot(y =  tune_outPoly$performances$error, x = 1:length(tune_outPoly$performances$error), ylab = "error
```



```
svmPoly = svm(gasMedian~., data = newAuto, kernel = "polynomial", cost = 10, degree = 1)
plot(svmPoly, Auto, mpg~horsepower)
```


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