Stat 427/627 Statistical Machine Learning

In-class Lab 10: Support Vector Machine

Contents

We will use `svm()` function in package `e1071` to fit support vector machines for classification.

```
# install.packages("e1071")
library(e1071)
```

1 Carseat data

Recall the `Carseats` data in package `ISLR2` (and in Lab 9, Trees).

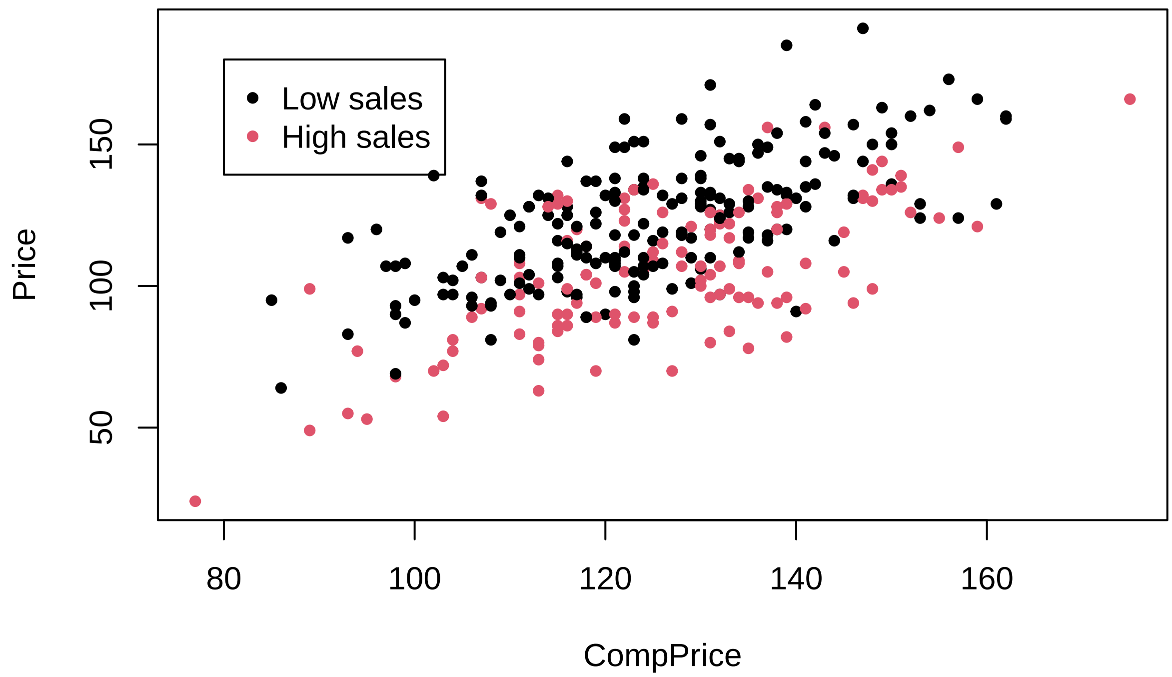
```
library(ISLR2)
names(Carseats)
```

```
##  [1] "Sales"       "CompPrice"   "Income"      "Advertising" "Population"
##  [6] "Price"       "ShelveLoc"   "Age"         "Education"   "Urban"
## [11] "US"
```

```
carseat.data <- Carseats
carseat.data$High <- factor(ifelse(Carseats$Sales <= 8, "No", "Yes"))

set.seed(2023)
n <- nrow(carseat.data)
z <- sample(n, floor(n*0.8))
carseat <- carseat.data[z, ]
row.names(carseat) <- NULL
carseat.new <- carseat.data[-z, ]  # hold-out for testing at the end.
```

```
plot(Price ~ CompPrice, col=High, pch=20, data=carseat)
legend(80, 180, pch=20, col=c(1, 2), legend = c("Low sales", "High sales"))
```



- The two classes cannot be separated by a hyperplane. Hence the maximum margin classifier won't work. But Support Vector Classifier and Support Vector Machine will!

2 `svm()` fits SVM with various kernels

- Old version of `svm()` may require that you use a data frame that **only** has the response and the predictors used in the model.
- To plot the classification boundary (in 2D), we'll need to declare the predictors in the `plot()` function.

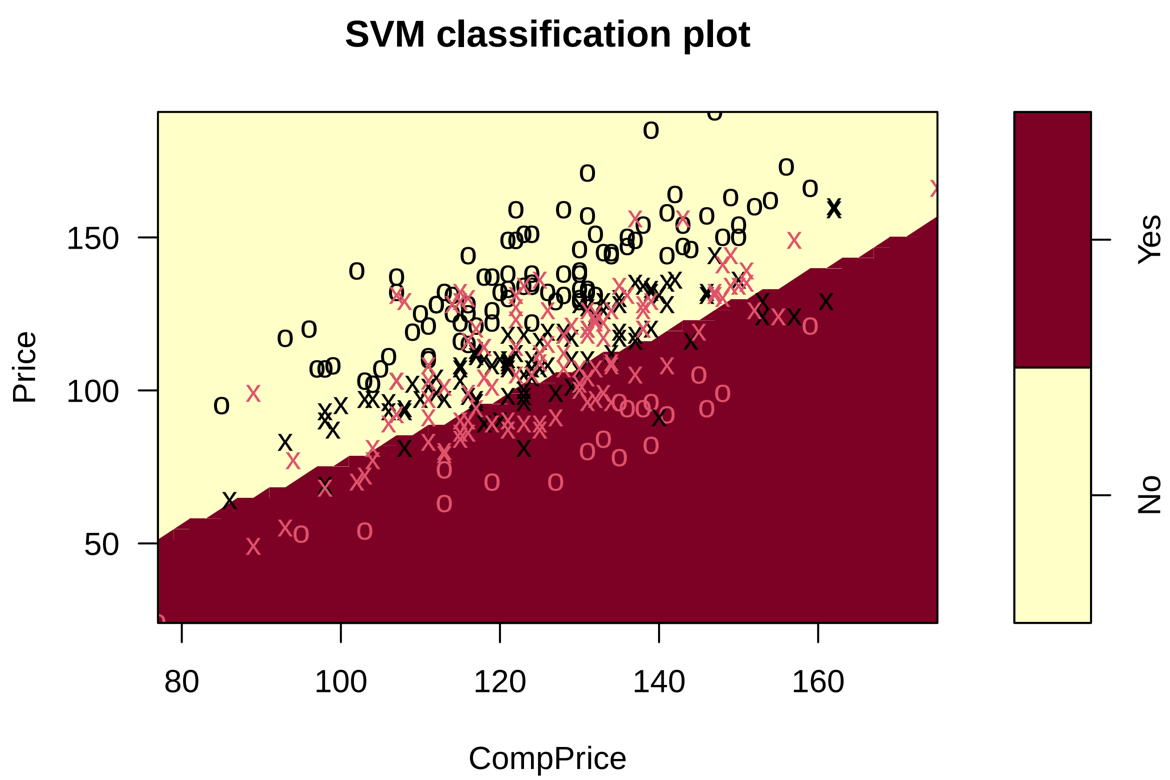
2.1 Linear kernel (Support vector classifier)

```
svm.lin <- svm(High ~ Price + CompPrice, kernel="linear",
                data=carseat)
summary(svm.lin)
```

```
##
## Call:
## svm(formula = High ~ Price + CompPrice, data = carseat, kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
```

```
## Number of Support Vectors:  211
##
##  ( 105 106 )
##
##
## Number of Classes:  2
##
## Levels:
##  No Yes
```

```r
plot(svm.lin, data=carseat, Price ~ CompPrice)
```



- Support vectors are marked as "x" in the plots (default).
- Other observations are marked as "o" in the plots (default).
- If you want to identify the support vectors:

```r
svm.lin$index[1:5] # index of the support vectors
```

```
## [1]  1  7  8  9 14
```

```r
head(carseat[svm.lin$index, c(2, 6, 12)])
```

```
##    CompPrice Price High
## 1        121    98   No
## 7        108    93   No
## 8        103    97   No
## 9        116    98   No
```

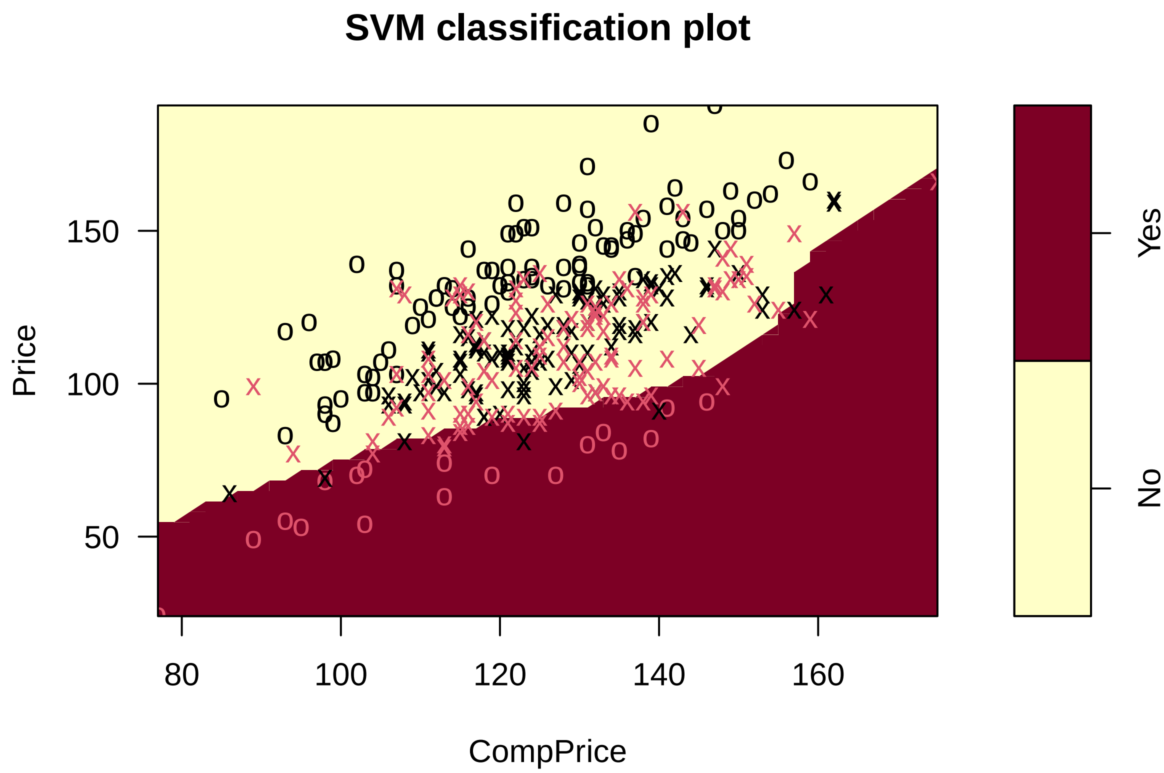
```
## 14        162   160   No
## 16        123   100   No
```

2.2 Polynomial

```r
svm.pol <- svm(High ~ Price + CompPrice, kernel="polynomial",
                data=carseat)
summary(svm.pol)
```

```
##
## Call:
## svm(formula = High ~ Price + CompPrice, data = carseat, kernel = "polynomial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  3
##      coef.0:  0
##
## Number of Support Vectors:  215
##
##  ( 107 108 )
##
##
## Number of Classes:  2
##
## Levels:
##  No Yes
```

```r
plot(svm.pol, data=carseat, Price ~ CompPrice)
```

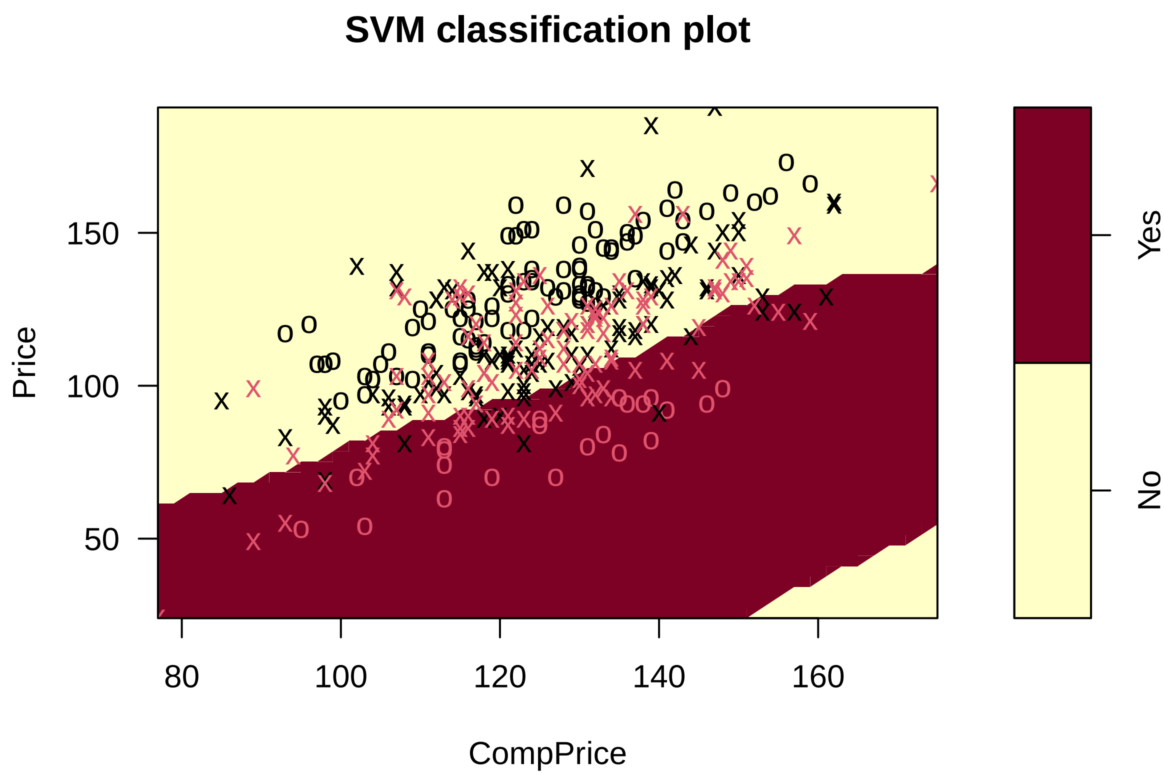


2.3 Radial

```
svm.rad <- svm(High ~ Price + CompPrice, kernel="radial",
                data=carseat)
summary(svm.rad)
```

```
##
## Call:
## svm(formula = High ~ Price + CompPrice, data = carseat, kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  213
##
##  ( 109 104 )
##
##
## Number of Classes:  2
##
## Levels:
##  No Yes
```

```r
plot(svm.rad, data=carseat, Price ~ CompPrice)
```



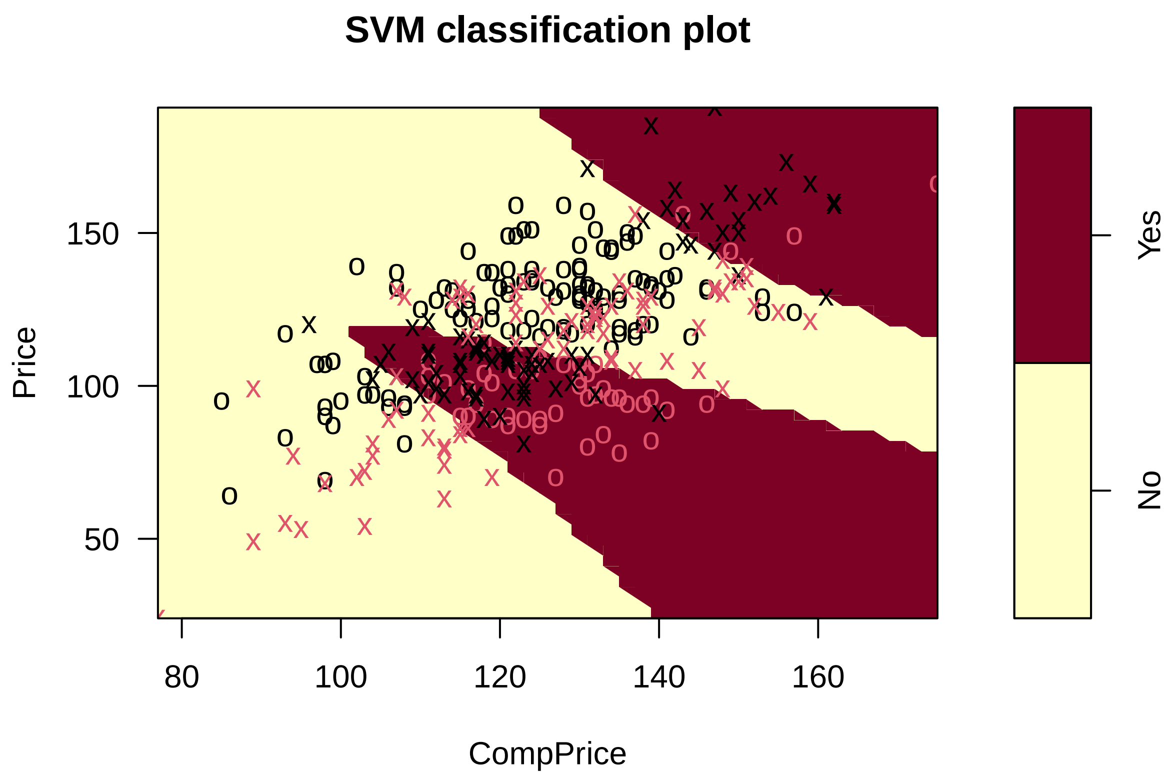
2.4 Sigmoid

```r
svm.sig <- svm(High ~ Price + CompPrice, kernel="sigmoid",
               data=carseat)
summary(svm.sig)
```

```
##
## Call:
## svm(formula = High ~ Price + CompPrice, data = carseat, kernel = "sigmoid")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  sigmoid
##        cost:  1
##      coef.0:  0
##
## Number of Support Vectors:  158
##
##  ( 79 79 )
##
##
## Number of Classes:  2
```

```
##
## Levels:
##  No Yes
```

```
plot(svm.sig, data=carseat, Price ~ CompPrice)
```



3 `tune()` the cost and the kernel

The function `tune()` conducts K-10 fold (by default) cross-validation over a "grid" of supplied parameter range. For classification, the CV prediction error is calculated. For regression, the CV mean squared error is calculated.

The "`cost =`" argument in `svm()` sets the cost of violations (this is not the C is the textbook, but they are related. When the `cost` argument is small, then the margins will be wide and many support vectors will be on the margin or will violate the margin.

Let's try 7 `cost` budget values (0.001, 0.01, 0.1, 1, 10, 100, 1000) and 4 kernels. The "grid" is 7×4 and 28 models will be evaluated.

```
set.seed(2023)
svm.tuning <- tune(svm, High ~ Price + CompPrice, data = carseat,
                   ranges = list(cost = 10^seq(-3, 3),
                                 kernel = c("linear", "polynomial",
                                            "radial", "sigmoid")))
summary(svm.tuning)
```

```
##
## Parameter tuning of 'svm':
```

```
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost kernel
##   100 radial
##
## - best performance: 0.28125
##
## - Detailed performance results:
##     cost     kernel    error dispersion
## 1  1e-03     linear 0.393750 0.11985958
## 2  1e-02     linear 0.393750 0.11985958
## 3  1e-01     linear 0.290625 0.04179667
## 4  1e+00     linear 0.303125 0.04670107
## 5  1e+01     linear 0.296875 0.04716346
## 6  1e+02     linear 0.293750 0.05145454
## 7  1e+03     linear 0.293750 0.05145454
## 8  1e-03 polynomial 0.393750 0.11985958
## 9  1e-02 polynomial 0.375000 0.10825318
## 10 1e-01 polynomial 0.365625 0.08339841
## 11 1e+00 polynomial 0.306250 0.06878156
## 12 1e+01 polynomial 0.300000 0.05352180
## 13 1e+02 polynomial 0.296875 0.04716346
## 14 1e+03 polynomial 0.296875 0.04716346
## 15 1e-03     radial 0.393750 0.11985958
## 16 1e-02     radial 0.393750 0.11985958
## 17 1e-01     radial 0.340625 0.10147566
## 18 1e+00     radial 0.306250 0.05855612
## 19 1e+01     radial 0.296875 0.07254369
## 20 1e+02     radial 0.281250 0.07933097
## 21 1e+03     radial 0.287500 0.07767230
## 22 1e-03    sigmoid 0.393750 0.11985958
## 23 1e-02    sigmoid 0.393750 0.11985958
## 24 1e-01    sigmoid 0.331250 0.09682458
## 25 1e+00    sigmoid 0.396875 0.09552714
## 26 1e+01    sigmoid 0.431250 0.10805252
## 27 1e+02    sigmoid 0.450000 0.11141452
## 28 1e+03    sigmoid 0.450000 0.11141452
```

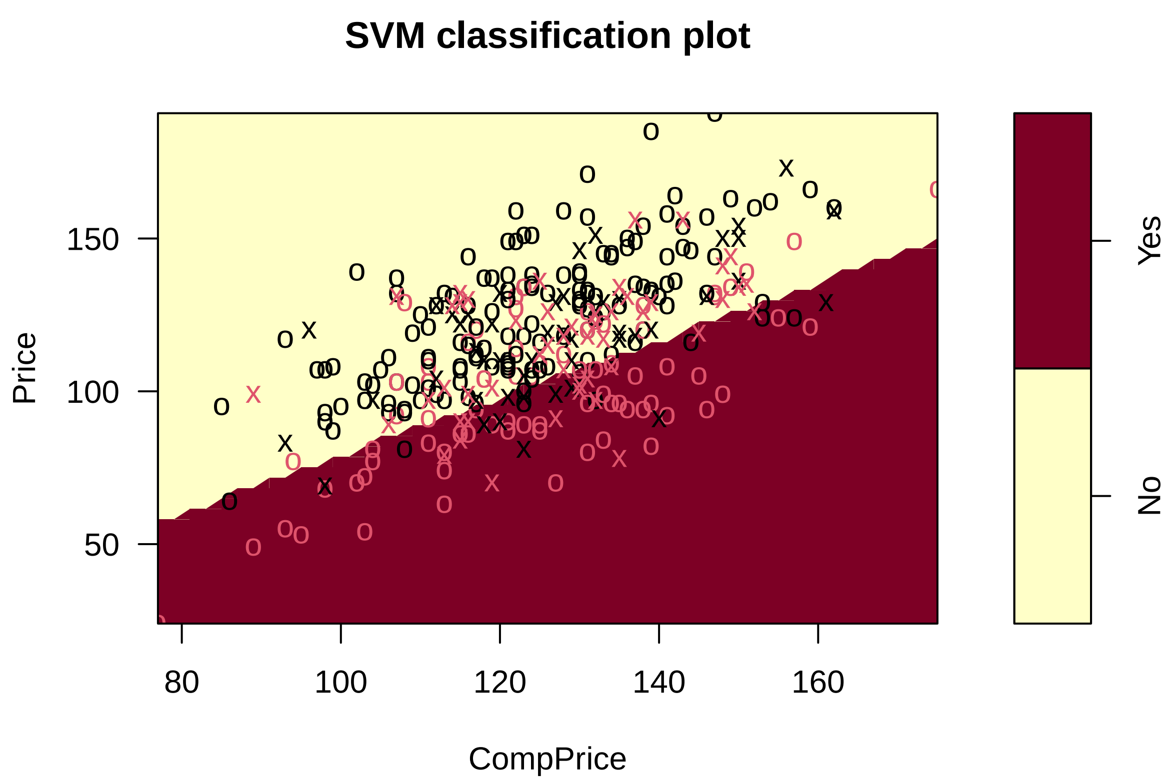
- When using 2 predictors, using `cost = 100` and `kernel="radial"` gives the smallest cross-validation error rate: 28.1%
- In practice, one may run another grid search, using finer grids, near the optimal parameters from the previous grid search.
- *Recall in 10-fold cross-validation*:
  - The results may be different when a different random seed, or function version, is used.
  - The misclassification rate (or other accuracy measurements such as MSE) calculated from cross-validation is a valid estimate of the model's misclassification rate (or MSE, etc.), because they are calcuated based on the observations in "testing" folds.

4 More predictors

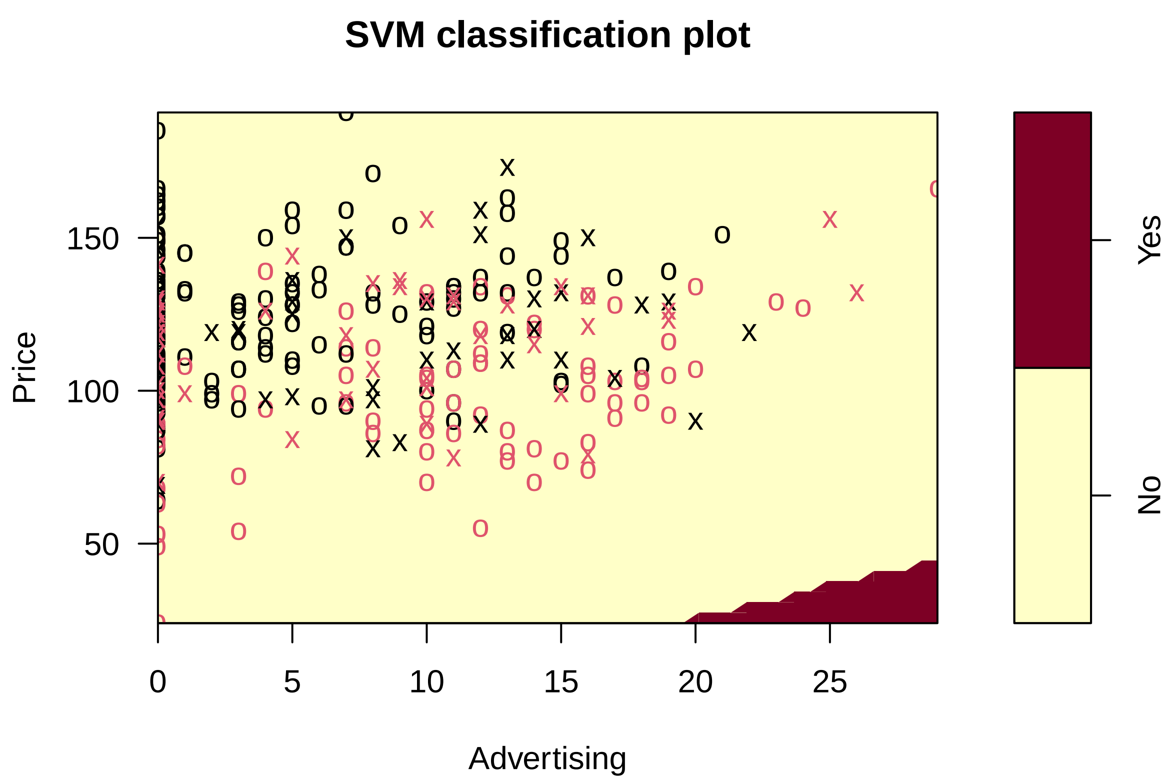
```r
svm.mlin <- svm(High ~ ., kernel="linear", data=carseat[, -1])
summary(svm.mlin)
```

```
##
## Call:
## svm(formula = High ~ ., data = carseat[, -1], kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  99
##
##  ( 50 49 )
##
##
## Number of Classes:  2
##
## Levels:
##  No Yes
```

```r
plot(svm.mlin, data=carseat, Price ~ CompPrice)
```



```r
plot(svm.mlin, data=carseat, Price ~ Advertising)
```



- The above 2-D plots are "sliced" by holding the other predictors at 0.

Let's tune the SVM with more predictors. We'll remove `Sales` (column 1) when we fit the model because `High` was created from `Sales`.

```r
set.seed(2023)
svm2.tuning <- tune(svm, High ~ . , data = carseat[, -1],
                   ranges = list(cost = 10^seq(-3, 3),
                                 kernel = c("linear", "polynomial",
                                            "radial", "sigmoid")))
svm2.tuning
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost kernel
##    10 linear
##
## - best performance: 0.1125
```

5 Predict the "new" data (and another run of validation)

Based on the previous tuning, let's consider the following 2 models:

```
svm.tuning$best.parameters
```

```
##    cost kernel
## 20  100 radial
```

```
svm.rad2 <- svm(High ~ Price + CompPrice, data = carseat, cost = 100, kernel = "radial")
yhat.new <- predict(svm.rad2, newdata=carseat.new)
err.rad2 <- c(mean(carseat.new$High != yhat.new), svm.tuning$best.performance)

svm2.tuning$best.parameters
```

```
##   cost kernel
## 5   10 linear
```

```
svm.mlin2 <- svm(High ~ ., data=carseat[, -1], cost=10, kernel="linear")
yhat.new <- predict(svm.mlin2, newdata=carseat.new)
err.mlin2 <- c(mean(carseat.new$High != yhat.new), svm2.tuning$best.performance)

err.table <- rbind(err.rad2, err.mlin2)
colnames(err.table) <- c("new.test.data", "Cross-validation")
err.table
```

```
##           new.test.data Cross-validation
## err.rad2          0.300          0.28125
## err.mlin2         0.125          0.11250
```

6 More than two classes

`svm()` handles response variable with more than two categories. But the response must be a factor!

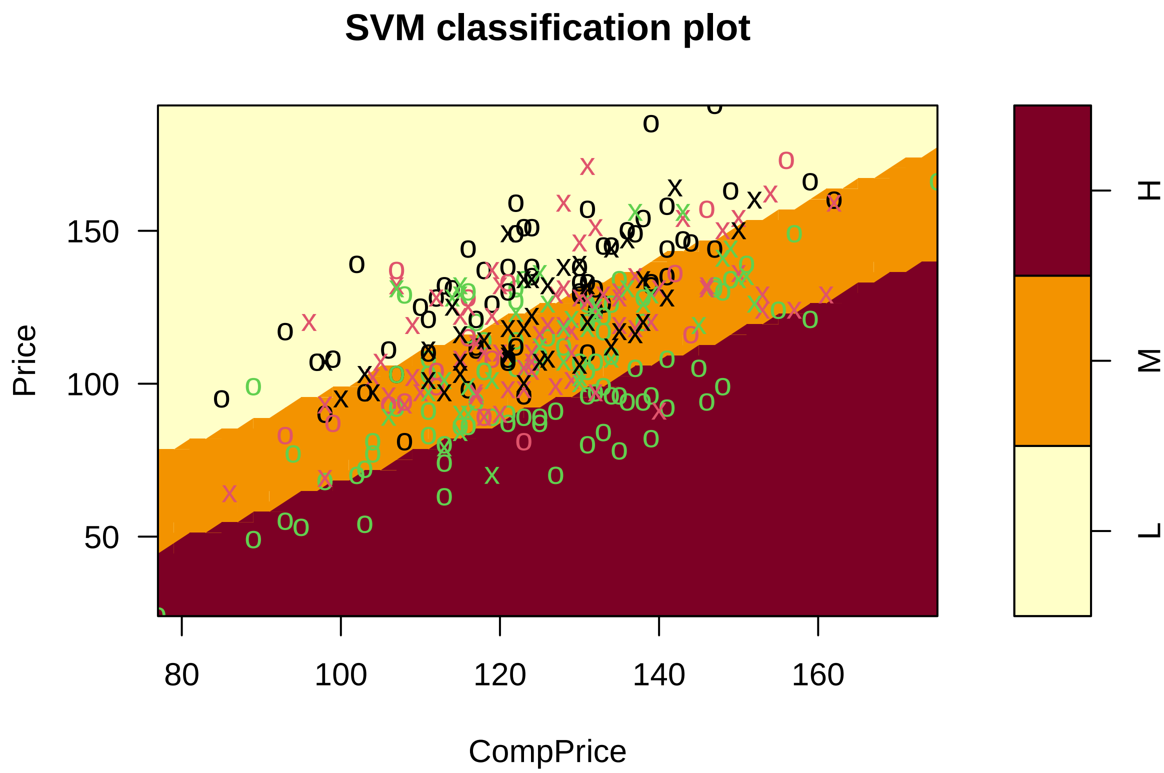
```
carseat$LMH <- cut(carseat$Sales, breaks = c(-Inf, 6, 8, Inf) , label=c("L", "M", "H"))
table(carseat$LMH)
```

```
##
##   L   M   H
## 103  91 126
```

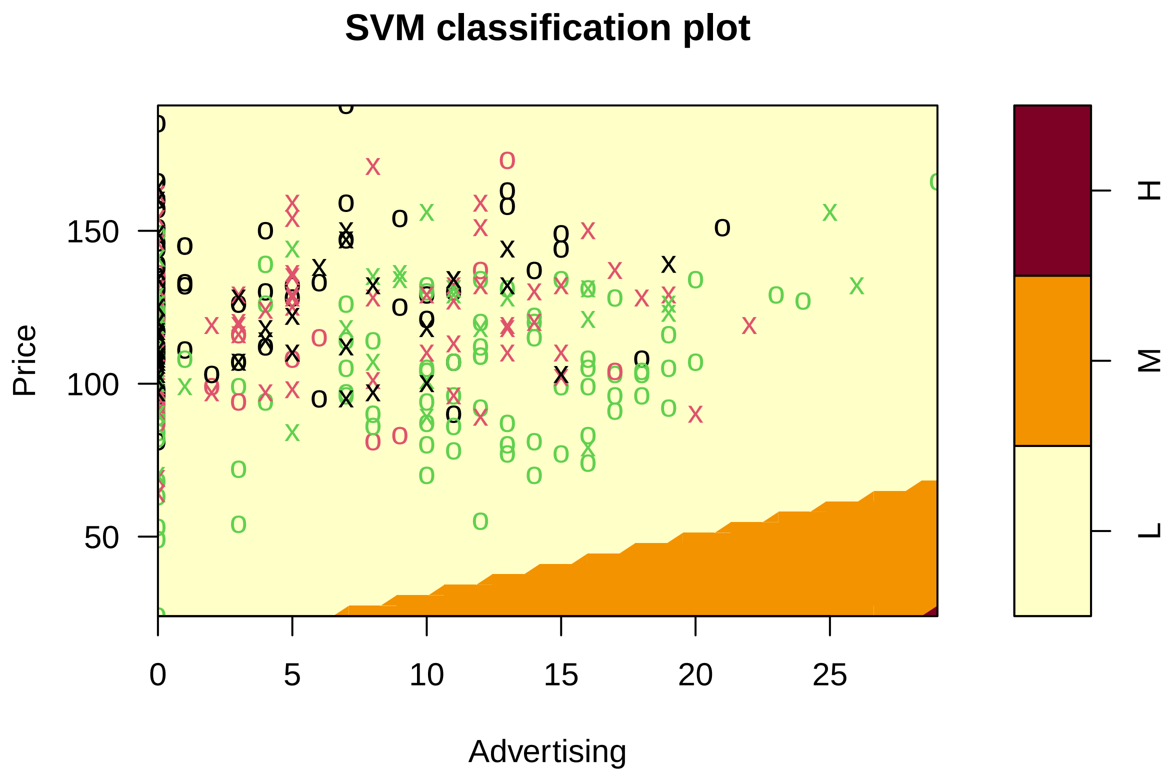
```
set.seed(2023)
LMH.tune <- tune(svm, LMH ~. -Sales - High, data=carseat,
                 ranges = list(cost = 10^seq(-3, 3),
                               kernel = c("linear", "polynomial",
                                          "radial", "sigmoid")))
LMH.tune$best.parameters
```

```
##   cost kernel
## 7 1000 linear
```

```
# Remove Sales (column 1) and High (column 12) to avoid plotting error.
LMH.svm <- svm(LMH ~., data=carseat[, -c(1, 12)], cost = 1000, kernel = "linear")
plot(LMH.svm, data=carseat, Price ~ CompPrice)
```




```
plot(LMH.svm, data=carseat, Price ~ Advertising)
```




```
yhat.new <- predict(LMH.svm, newdata=carseat.new)

carseat.new$LMH <- cut(carseat.new$Sales, breaks = c(-Inf, 6, 8, Inf),
                       label=c("L", "M", "H"))
c(mean(carseat.new$LMH != yhat.new), LMH.tune$best.performance)
```

```
## [1] 0.237500 0.221875
```

- In general, it is more difficult to predict finer/more classes.