The Final Project

Milica Cudina

2024-04-28

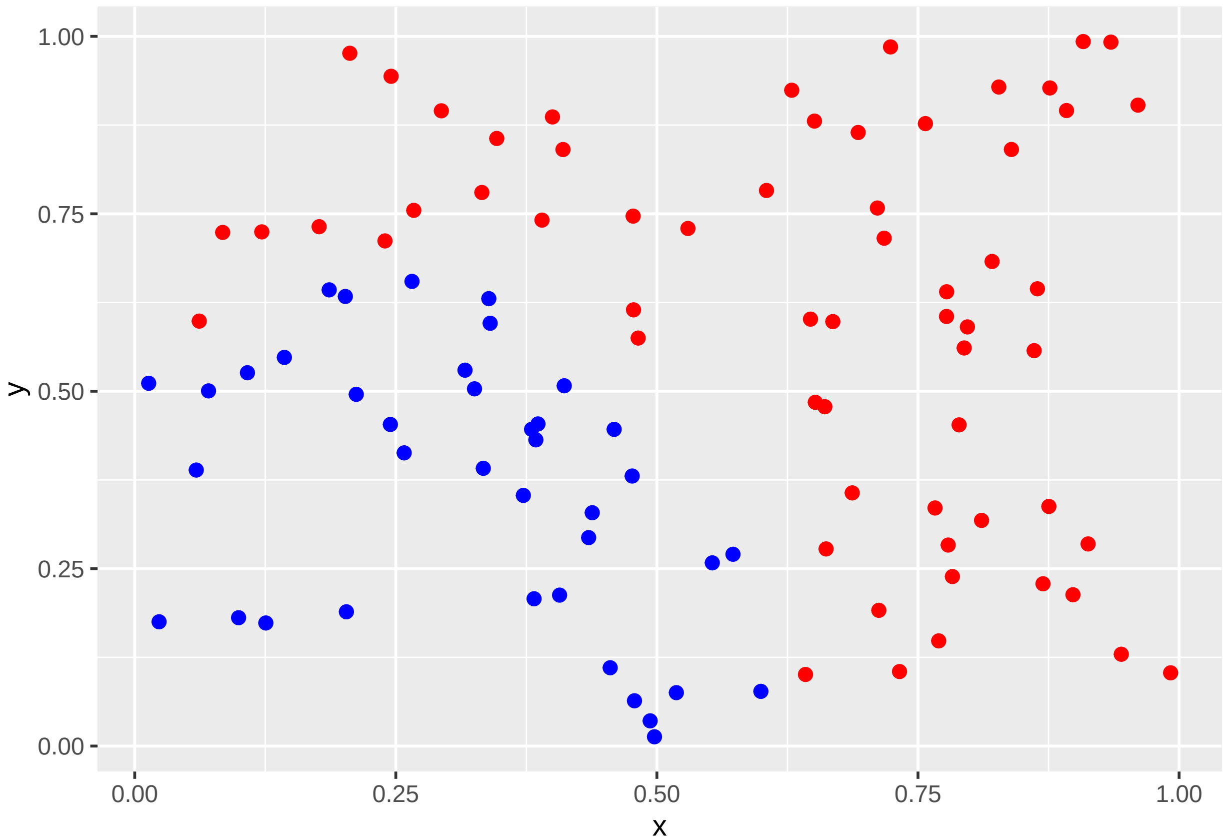
```r
library(nimble)
```

Problem #1 (45 points)

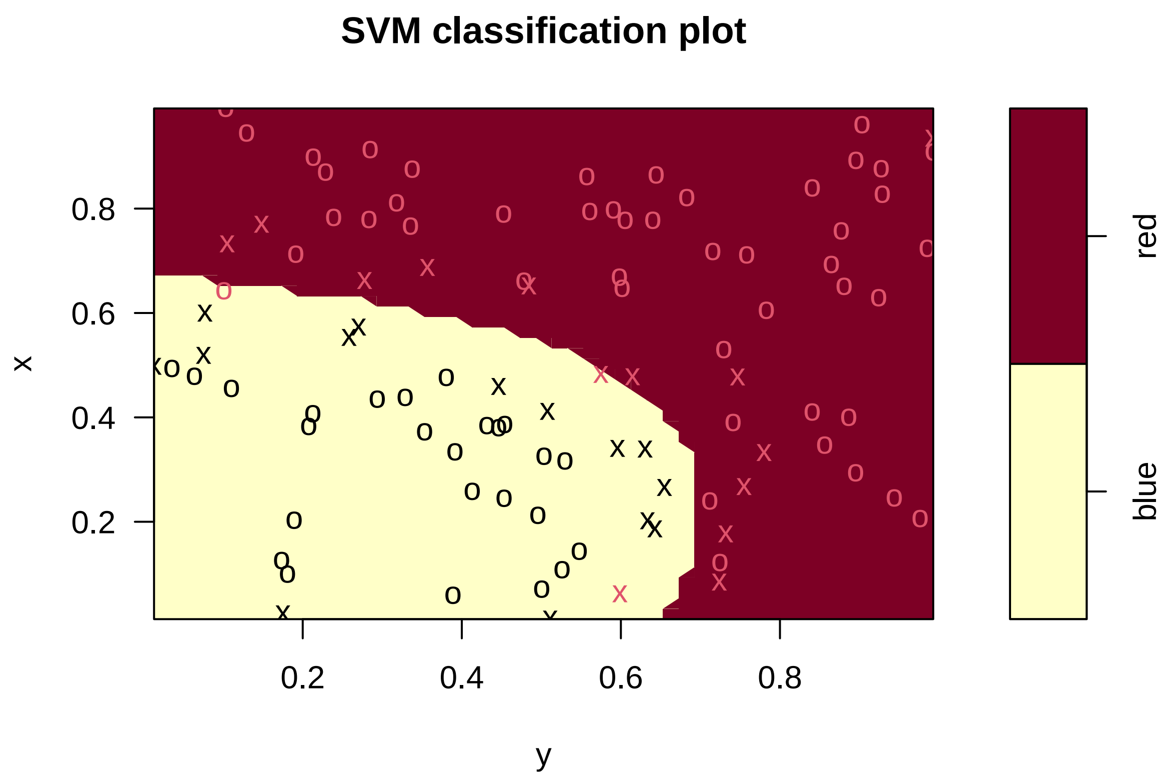
Solve Problem **9.7.4** from the textbook (page 399).

Solution:

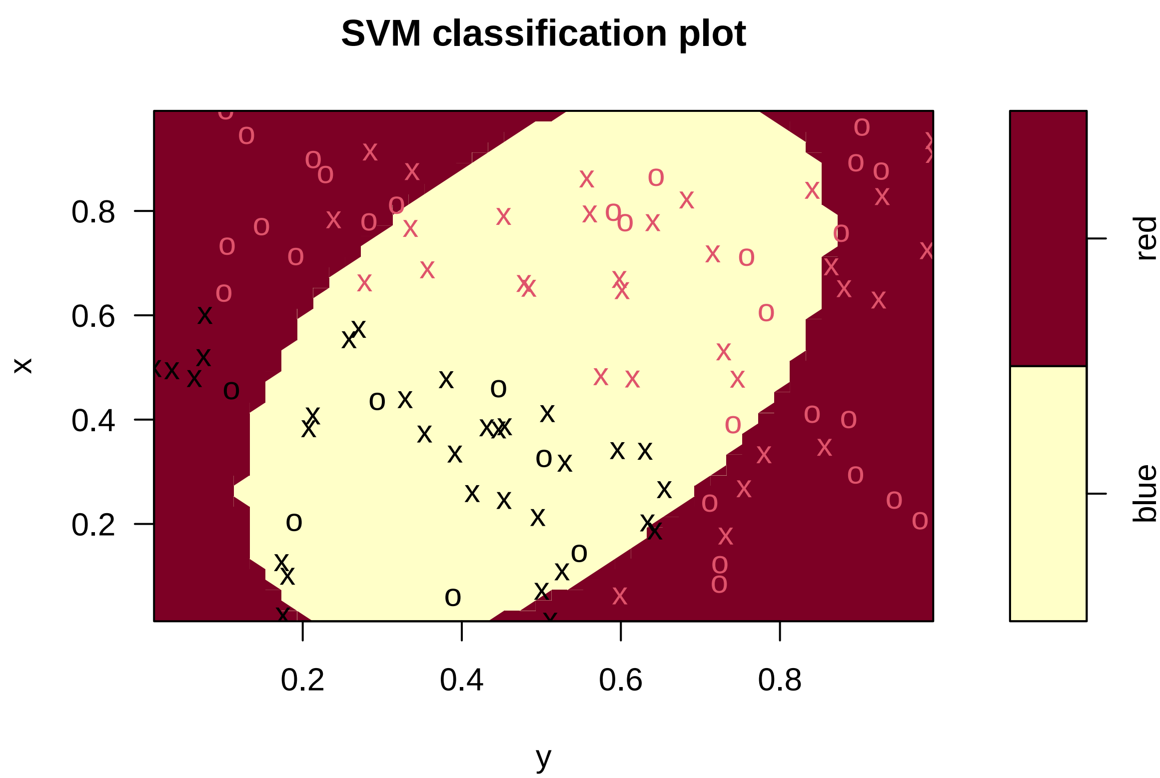
```r
library(ggplot2)
library(e1071)
set.seed(1)
data <- data.frame(
  x = runif(100),
  y = runif(100)
)
score <- (2*data$x-0.5)^2 + (data$y)^2 - 0.5
data$class <- factor(ifelse(score > 0, "red", "blue"))

p <- ggplot(data, aes(x = x, y = y, color = class)) +
  geom_point(size = 2) + scale_colour_identity()
p
```



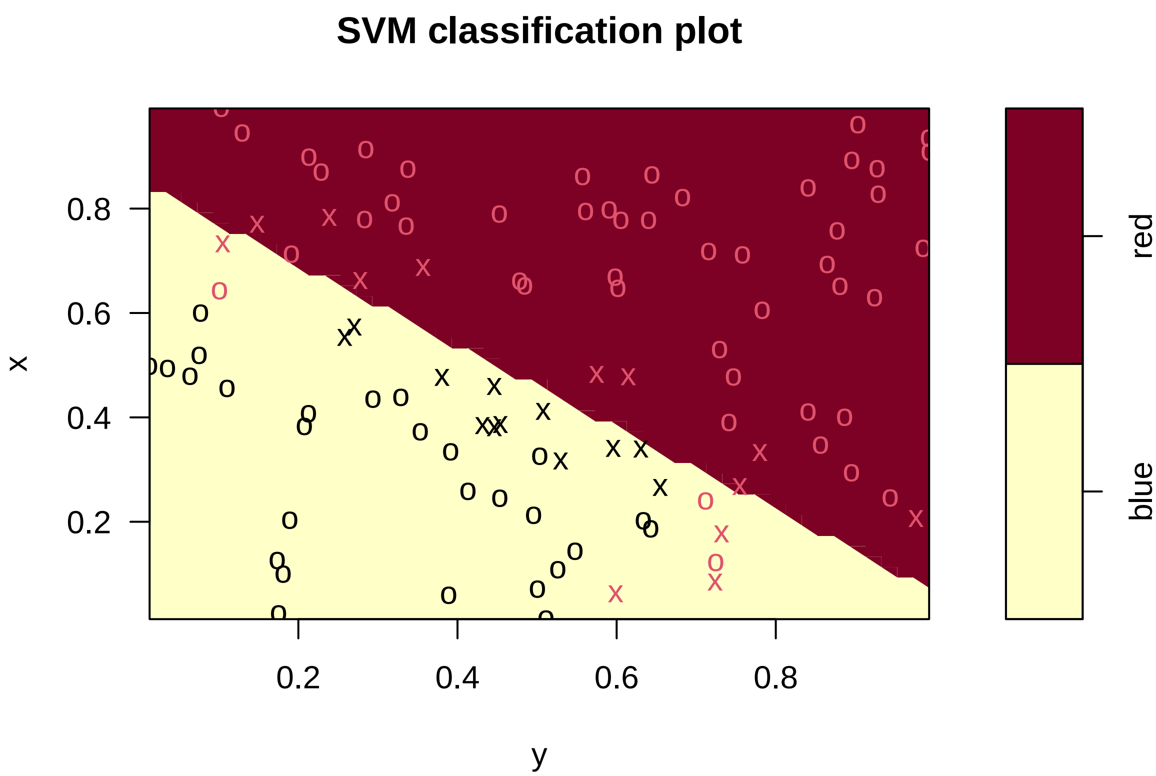

```r
train <- 1:75
test <- 76:100

fits <- list(
  "Radial" = svm(class ~ ., data = data[train, ], kernel = "radial"),
  "Polynomial" = svm(class ~ ., data = data[train, ], kernel = "polynomial", degree = 2),
  "Linear" = svm(class ~ ., data = data[train, ], kernel = "linear")
)

err <- function(model, data) {
  out <- table(predict(model, data), data$class)
  (out[1, 2] + out[2, 1]) / sum(out)
}
plot(fits[[1]], data)
```




```r
plot(fits[[2]], data)
```




```r
plot(fits[[3]], data)
```




```
sapply(fits, err, data = data[train, ])
##      Radial Polynomial     Linear
## 0.02666667 0.34666667 0.05333333
sapply(fits, err, data = data[test, ])
##      Radial Polynomial     Linear
##        0.04       0.28       0.12
```

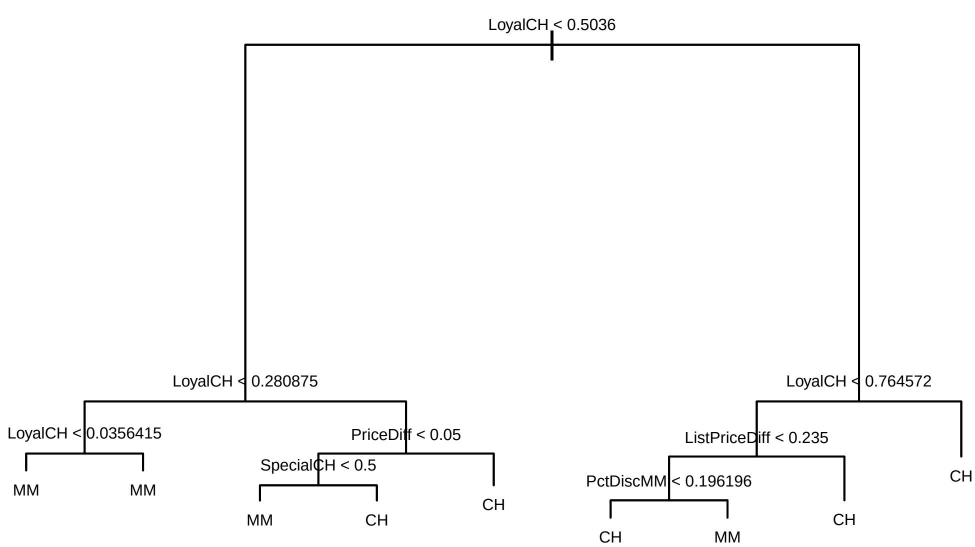
Problem #2 ($5 \times 11 = 55$ points)

Solve Problem **8.4.9** from the textbook (pages 363-364).

Solution:

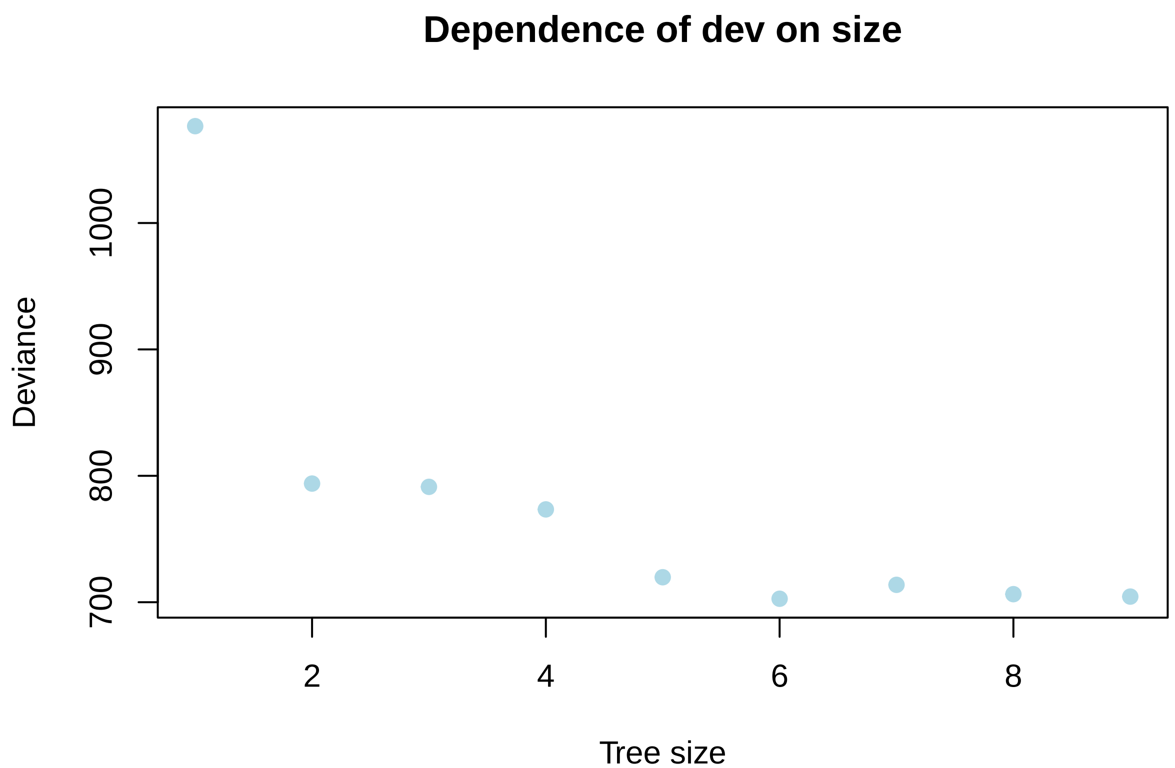
```
library(ISLR2)
data(OJ)
set.seed(1)

#training set
train <- sample(1:nrow(OJ), floor(nrow(OJ)*0.75))

#fitting a tree
library(tree)
stablo <- tree(Purchase ~ ., data = OJ[train, ])
summary(stablo)
##
## Classification tree:
## tree(formula = Purchase ~ ., data = OJ[train, ])
## Variables actually used in tree construction:
## [1] "LoyalCH"       "PriceDiff"     "SpecialCH"     "ListPriceDiff"
## [5] "PctDiscMM"
```

```r
## Number of terminal nodes:  9
## Residual mean deviance:  0.7417 = 588.1 / 793
## Misclassification error rate: 0.1584 = 127 / 802
#for to find paths to terminal nodes
stablo
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 802 1075.00 CH ( 0.60723 0.39277 )
##    2) LoyalCH < 0.5036 365  441.60 MM ( 0.29315 0.70685 )
##      4) LoyalCH < 0.280875 177  140.50 MM ( 0.13559 0.86441 )
##        8) LoyalCH < 0.0356415 59   10.14 MM ( 0.01695 0.98305 ) *
##        9) LoyalCH > 0.0356415 118  116.40 MM ( 0.19492 0.80508 ) *
##      5) LoyalCH > 0.280875 188  258.00 MM ( 0.44149 0.55851 )
##       10) PriceDiff < 0.05 79   84.79 MM ( 0.22785 0.77215 )
##         20) SpecialCH < 0.5 64   51.98 MM ( 0.14062 0.85938 ) *
##         21) SpecialCH > 0.5 15   20.19 CH ( 0.60000 0.40000 ) *
##       11) PriceDiff > 0.05 109  147.00 CH ( 0.59633 0.40367 ) *
##    3) LoyalCH > 0.5036 437  338.40 CH ( 0.86957 0.13043 )
##      6) LoyalCH < 0.764572 175  201.60 CH ( 0.73714 0.26286 )
##       12) ListPriceDiff < 0.235 72   99.81 MM ( 0.50000 0.50000 )
##         24) PctDiscMM < 0.196196 55   73.14 CH ( 0.61818 0.38182 ) *
##         25) PctDiscMM > 0.196196 17   12.32 MM ( 0.11765 0.88235 ) *
##       13) ListPriceDiff > 0.235 103   65.64 CH ( 0.90291 0.09709 ) *
##      7) LoyalCH > 0.764572 262   91.28 CH ( 0.95802 0.04198 ) *
#plotting the tree
plot(stablo)
text(stablo, pretty = 0, digits = 2, cex = 0.5)
```



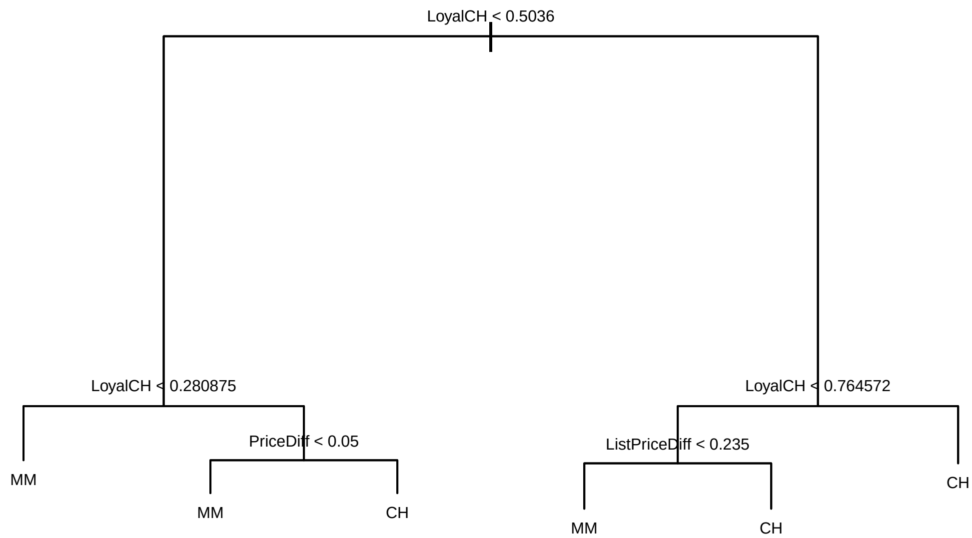

```r
#response table
tab=table(predict(stablo, OJ[test, ], type = "class"), OJ[test, "Purchase"])
tab
##
##       CH MM
##    CH 24  0
##    MM  1  0
#cross-validation
```

```
cv.stablo= cv.tree(stablo)
cv.stablo
## $size
## [1] 9 8 7 6 5 4 3 2 1
##
## $dev
## [1]  704.4675  706.3754  713.7588  702.7565  719.7714  773.4425  791.2789
## [8]  793.8671 1076.6373
##
## $k
## [1]      -Inf  12.62207  13.94616  14.35384  26.21539  36.15632  43.07317
## [8]  45.53369 294.59602
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
#scatterplot of size and deviance
plot(cv.stablo$size, cv.stablo$dev,
     main="Dependence of dev on size",
     pch=19, col="lightblue",
     xlab="Tree size", ylab="Deviance")
```




```
#which is optimal?
arg.min=which.min(cv.stablo$dev)
cv.stablo$size[arg.min]
## [1] 6
#prune the tree
```

```r
snip <- prune.tree(stablo, best = cv.stablo$size[arg.min])
plot(snip)
text(snip, pretty = 0, digits = 2, cex = 0.5)
```




```r
#training error for pruned vs. unpruned
class.err <- function(fit) {
  summary(fit)$misclass[1] / summary(fit)$misclass[2]
}
class.err(stablo)
## [1] 0.1583541
class.err(snip)
## [1] 0.1783042
#testing error for pruned vs. unpruned
new.err <- function(fit) {
  pred <- predict(fit, newdata = OJ[-train, ], type = "class")
  mean(pred != OJ[-train, "Purchase"])
}
new.err(stablo)
## [1] 0.1716418
new.err(snip)
## [1] 0.1902985
```