Homework 7

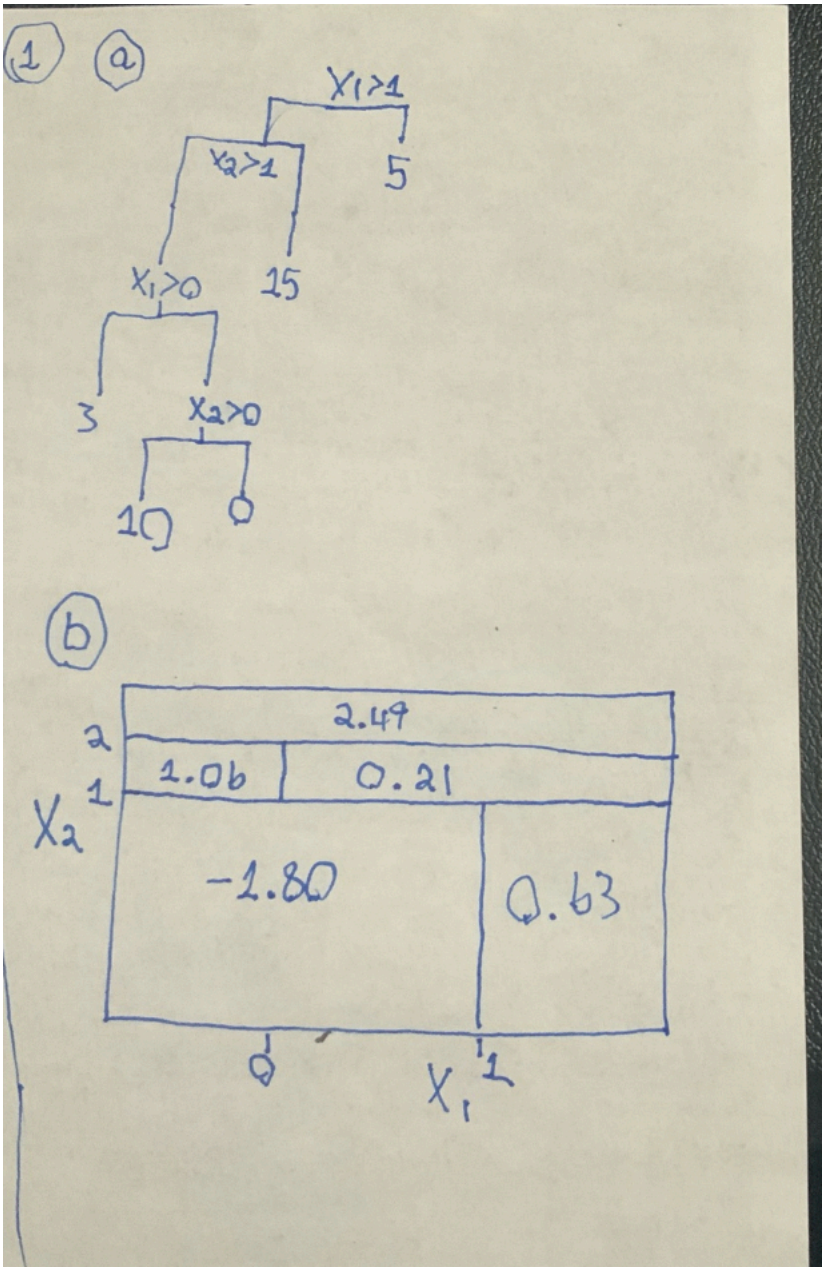
Daniel Tshiani

2025-06-16

1

I drew it by hand

```
knitr::include_graphics("Screenshot 2025-06-17 at 10.31.35 PM.png")
```



2

if we do the majority vote. we would count those whose probability of red is greater than 0.5 6 estimates have a probability greater than 0.5 and 4 estimates have a probability less than 0.5 so the majority method would classify red.

the second method is the average probability method. we would take the average and if its greater or equal to 0.5 we would classify red.

```
X <- c(0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6,
0.65, 0.7, 0.75)
mean(X)
```

```
## [1] 0.45
```

0.45 is less than 0.5 so we would classify it as green

3

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(ISLR)
attach(OJ)
OJ <- OJ
glimpse(OJ)
```

```
## Rows: 1,070
## Columns: 18
## $ Purchase       <fct> CH, CH, CH, MM, CH, CH, CH, CH, CH, CH, CH, CH, CH, CH,~
## $ WeekofPurchase <dbl> 237, 239, 245, 227, 228, 230, 232, 234, 235, 238, 240, ~
## $ StoreID        <dbl> 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 1, 2, 2~
## $ PriceCH        <dbl> 1.75, 1.75, 1.86, 1.69, 1.69, 1.69, 1.69, 1.75, 1.75, 1~
## $ PriceMM        <dbl> 1.99, 1.99, 2.09, 1.69, 1.69, 1.99, 1.99, 1.99, 1.99, 1~
## $ DiscCH         <dbl> 0.00, 0.00, 0.17, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0~
## $ DiscMM         <dbl> 0.00, 0.30, 0.00, 0.00, 0.00, 0.00, 0.40, 0.40, 0.40, 0~
## $ SpecialCH      <dbl> 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ SpecialMM      <dbl> 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0~
## $ LoyalCH        <dbl> 0.500000, 0.600000, 0.680000, 0.400000, 0.956535, 0.965~
## $ SalePriceMM    <dbl> 1.99, 1.69, 2.09, 1.69, 1.69, 1.99, 1.59, 1.59, 1.59, 1~
## $ SalePriceCH    <dbl> 1.75, 1.75, 1.69, 1.69, 1.69, 1.69, 1.69, 1.75, 1.75, 1~
## $ PriceDiff      <dbl> 0.24, -0.06, 0.40, 0.00, 0.00, 0.30, -0.10, -0.16, -0.1~
## $ Store7         <fct> No, No, No, No, Yes, Yes, Yes, Yes, Yes, Yes, Yes, Yes,~
## $ PctDiscMM      <dbl> 0.000000, 0.150754, 0.000000, 0.000000, 0.000000, 0.000~
## $ PctDiscCH      <dbl> 0.000000, 0.000000, 0.091398, 0.000000, 0.000000, 0.000~
## $ ListPriceDiff  <dbl> 0.24, 0.24, 0.23, 0.00, 0.00, 0.30, 0.30, 0.24, 0.24, 0~
## $ STORE          <dbl> 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2~
```

a

```
set.seed(123)
train_index <- sample(1:nrow(OJ), 800)
train_data <- OJ[train_index, ]
test_data <- OJ[-train_index, ]
```

b

```
library(tree)

tree_model <- tree(Purchase ~ . ,data = OJ) # except for buy as predictors but dont know which one that

summary(tree_model)
```

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = OJ)
## Variables actually used in tree construction:
## [1] "LoyalCH"       "PriceDiff"     "ListPriceDiff"
## Number of terminal nodes:  8
## Residual mean deviance:  0.7571 = 804 / 1062
## Misclassification error rate: 0.1636 = 175 / 1070
```

There are 8 terminal nodes. the misclassification error rate is about 16%.

c

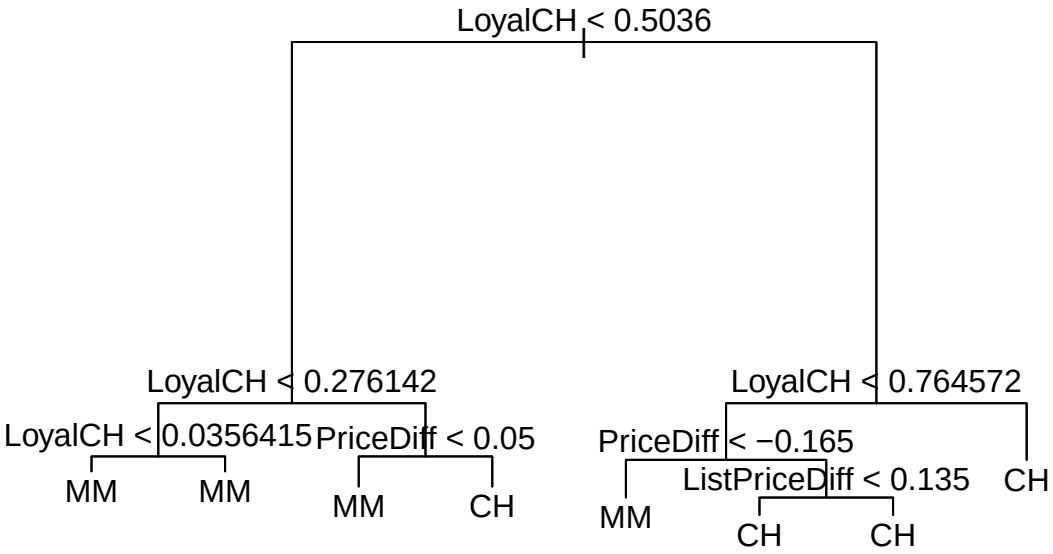
```
tree_model
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 1070 1431.00 CH ( 0.61028 0.38972 )
##    2) LoyalCH < 0.5036 469  559.30 MM ( 0.28358 0.71642 )
##      4) LoyalCH < 0.276142 223  164.60 MM ( 0.12108 0.87892 )
##        8) LoyalCH < 0.0356415 75   10.62 MM ( 0.01333 0.98667 ) *
##        9) LoyalCH > 0.0356415 148  137.60 MM ( 0.17568 0.82432 ) *
##      5) LoyalCH > 0.276142 246  336.30 MM ( 0.43089 0.56911 )
##       10) PriceDiff < 0.05 101  105.90 MM ( 0.21782 0.78218 ) *
##       11) PriceDiff > 0.05 145  197.30 CH ( 0.57931 0.42069 ) *
##    3) LoyalCH > 0.5036 601  475.20 CH ( 0.86522 0.13478 )
##      6) LoyalCH < 0.764572 251  289.20 CH ( 0.73705 0.26295 )
##       12) PriceDiff < -0.165 40   48.87 MM ( 0.30000 0.70000 ) *
##       13) PriceDiff > -0.165 211  199.00 CH ( 0.81991 0.18009 )
##         26) ListPriceDiff < 0.135 34   47.02 CH ( 0.52941 0.47059 ) *
##         27) ListPriceDiff > 0.135 177  132.90 CH ( 0.87571 0.12429 ) *
##      7) LoyalCH > 0.764572 350  123.80 CH ( 0.95714 0.04286 ) *
```

if LoyalCH is less than 0.5036, and LoyalCH is greater than 0.276142, and PriceDiff is greater than 0.05, the we would estimate the response would be of the purchase would be CH.

d

```
plot(tree_model)
text(tree_model)
```



if LoyalCH is less than 0.5036, and LoyalCH is greater than 0.276142, and PriceDiff is greater than 0.05, the we would estimate the response would be of the purchase would be CH.

e

```
Yhat <- predict(tree_model, newdata = test_data, type = "class")
summary(Yhat)
```

```
##  CH  MM
## 181  89
```

```
table(Yhat, test_data$Purchase)
```

```
##
## Yhat  CH  MM
##   CH 151  30
##   MM  15  74
```

```
(30+15)/(151+30+15+74)
```

```
## [1] 0.1666667
```

test error rate is about 16%

f

```
cv <- cv.tree(tree_model)
cv
```

```
## $size
## [1] 8 7 6 5 4 3 2 1
##
## $dev
## [1]  904.2243  917.3516  928.5913  949.2100  962.8427 1037.0930 1039.9048
## [8] 1432.3182
##
## $k
## [1]      -Inf  16.40821  19.08305  33.08948  41.35346  58.41721  62.17284
## [8] 396.28995
```

```
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
cv$size[which.min(cv$dev)]
```

```
## [1] 8
```

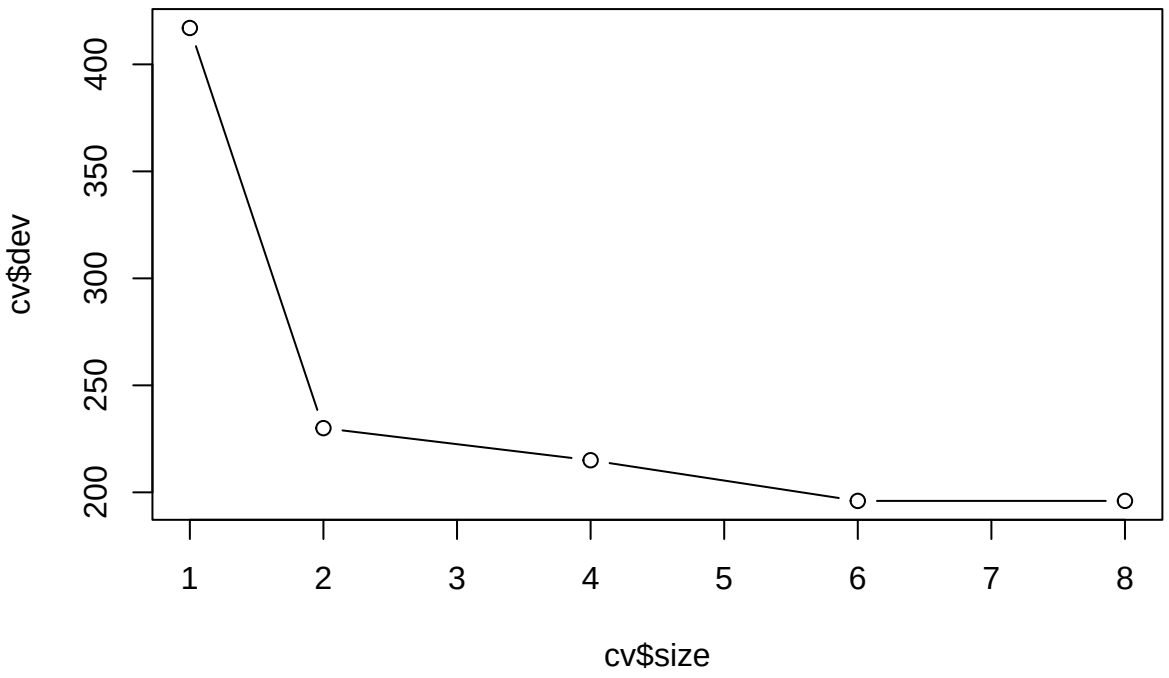
the optimal tree size is 8 terminal nodes

g

```
cv <- cv.tree(tree_model, FUN = prune.misclass)
cv
```

```
## $size
## [1] 8 6 4 2 1
##
## $dev
## [1] 196 196 215 230 417
##
## $k
## [1]  -Inf   0.0   8.0  11.5 203.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
plot(cv$size, cv$dev, type = "b")
```



h

```
min_index <- which.min(cv$dev)
best_size <- cv$size[min_index]
min_error <- cv$dev[min_index] / nrow(OJ)

best_size
```

```
## [1] 8
```

```
min_error
```

```
## [1] 0.1831776
```

i

```
tr_opt <- prune.tree(tree_model, best =8)
summary(tr_opt)
```

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = OJ)
## Variables actually used in tree construction:
## [1] "LoyalCH"       "PriceDiff"     "ListPriceDiff"
## Number of terminal nodes:  8
## Residual mean deviance:  0.7571 = 804 / 1062
## Misclassification error rate: 0.1636 = 175 / 1070
```

j

the training error rate for the pruned tree is lower than the error rate for the unprune tree.

k

the test error rates for the pruned tree is lower than the test error rate for the unpruned tree.