hw3

Xinwei Zhang

3/8/2018

P3

```
X = c(0.0,0.2,0.4,0.6,0.8,1.0)
ones = c(1,1,1,1,1,1)
X = cbind(ones,X)
y = c(0,0,0,1,0,1)

lr = function(X,y){
  get_p = function(X,beta){
    beta = as.vector(beta)
    return (exp(X%*%beta) / (1 + exp(X%*%beta)))
  }

  beta = rep(0,ncol(X))

  for (i in 0:9){
    p = as.vector(get_p(X,beta))
    W = diag(p * (1 - p))
    delta = solve(t(X)%*%W%*%X) %*% t(X)%*%(y - p)
    beta = beta + delta
  }
  print (beta)
  return (beta)
}

result = lr(X,y)
```

```
##            [,1]
## ones -4.097970
## X     5.723309
```

P6

```
p = c(0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, 0.75)
```

majority approach

```
sum(p>=0.5) > sum(p<0.5)
```

```
## [1] TRUE
```

The number of red prediction is greater than that of green predictions. So the prediction is red.

average approach

```
mean(p)
```

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

The average of possibilities is less than 0.5. So the prediction is green.

P7

(a)

```
library(ISLR)
attach(OJ)

set.seed(1000)
train = sample(dim(OJ)[1], 800)
OJ.train = OJ[train, ]
OJ.test = OJ[-train, ]
```

(b)

```
library(tree)
oj.tree = tree(Purchase ~ ., data = OJ.train)
summary(oj.tree)
```

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = OJ.train)
## Variables actually used in tree construction:
## [1] "LoyalCH"        "PriceDiff"      "WeekofPurchase"
## Number of terminal nodes:  7
## Residual mean deviance:  0.7848 = 622.4 / 793
## Misclassification error rate: 0.175 = 140 / 800
```

The tree use three variables: LoyalCH, PriceDiff, WeekofPurchase. The training error rate is 0.175. It has 7 terminal nodes.

(c)

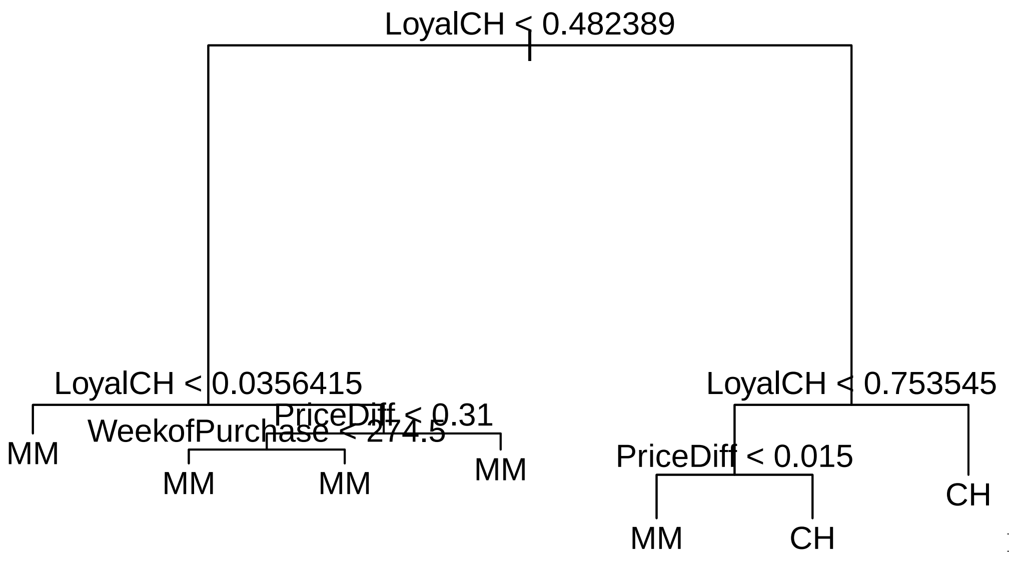
```
oj.tree
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 800 1069.000 CH ( 0.61125 0.38875 )
##    2) LoyalCH < 0.482389 297  319.600 MM ( 0.22896 0.77104 )
##      4) LoyalCH < 0.0356415 55    9.996 MM ( 0.01818 0.98182 ) *
##      5) LoyalCH > 0.0356415 242  285.500 MM ( 0.27686 0.72314 )
##       10) PriceDiff < 0.31 188  197.200 MM ( 0.21809 0.78191 )
##         20) WeekofPurchase < 274.5 166  185.600 MM ( 0.24699 0.75301 ) *
##         21) WeekofPurchase > 274.5 22    0.000 MM ( 0.00000 1.00000 ) *
##       11) PriceDiff > 0.31 54   74.790 MM ( 0.48148 0.51852 ) *
##    3) LoyalCH > 0.482389 503  447.300 CH ( 0.83698 0.16302 )
##      6) LoyalCH < 0.753545 235  284.500 CH ( 0.70638 0.29362 )
##       12) PriceDiff < 0.015 72   98.420 MM ( 0.43056 0.56944 ) *
##       13) PriceDiff > 0.015 163  149.500 CH ( 0.82822 0.17178 ) *
##      7) LoyalCH > 0.753545 268  104.000 CH ( 0.95149 0.04851 ) *
```

I choose to pick 12). The spliting variable is PriceDiff. The spliting value of this node is 0.015. 72 points are below this node. This is terminal node. The prediciton is Sales=MM. The deviance for all points below this node is 80. 43% points in this node have CH as value of Sales. 57% points have MM as value of Sales.

(d)

```r
plot(oj.tree)
text(oj.tree, pretty = 0)
```



LoyalCH is the most important variable. Top three nodes contain LoyalCH. If LoyalCH<0.0356415, the prediction is MM. If LoyalCH>0.753545, the prediction is CH. In between, the prediction depends on WeekofPurchase and PriceDiff.

(e)

```r
oj.pred = predict(oj.tree, OJ.test, type="class")
table(OJ.test$Purchase, oj.pred)
```

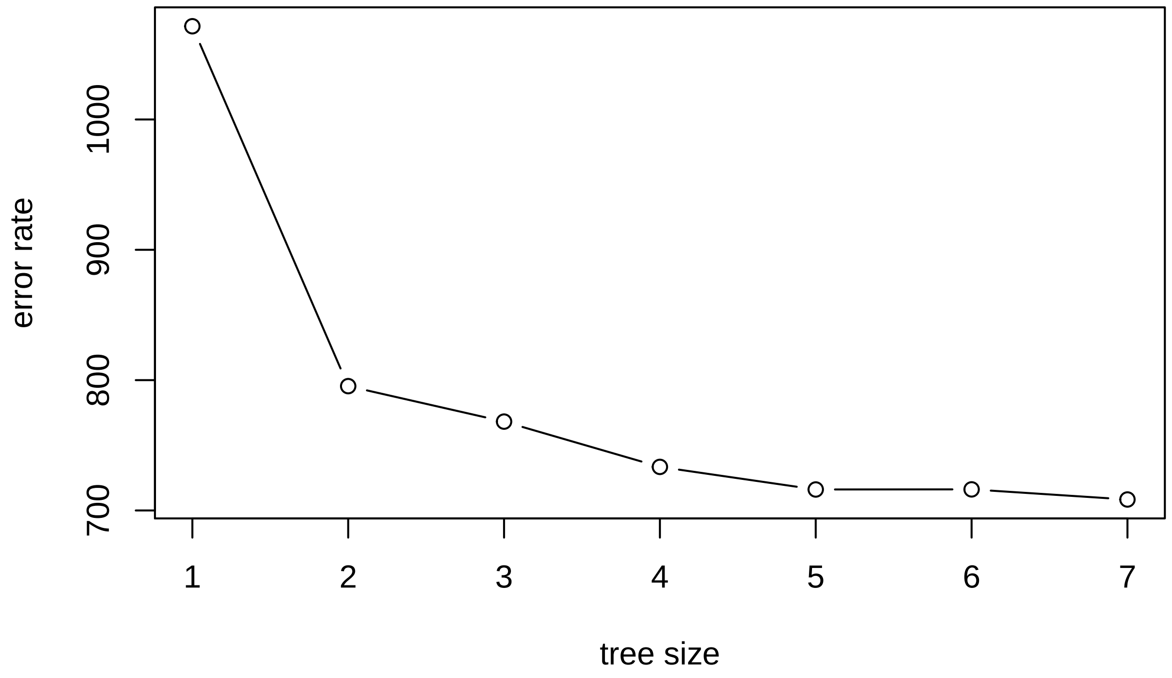
```
##     oj.pred
##       CH  MM
##   CH 123  41
##   MM  12  94
```

(f)

```
cv.oj = cv.tree(oj.tree, FUN = prune.tree)
```

(g)

```
plot(cv.oj$size, cv.oj$dev, type = "b", xlab = "tree size", ylab = "error rate")
```



(h)

Size of 7 gives the lowest cross-validated classifiation error rate.

(i)

```
oj.pruned = prune.tree(oj.tree, best = 7)
```

(j)

```
summary(oj.pruned)
```

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = OJ.train)
## Variables actually used in tree construction:
## [1] "LoyalCH"        "PriceDiff"      "WeekofPurchase"
## Number of terminal nodes:  7
## Residual mean deviance:  0.7848 = 622.4 / 793
## Misclassification error rate: 0.175 = 140 / 800
```

The training error tate are the same, which is 0.175.

(k)

```
pred.unpruned = predict(oj.tree, OJ.test, type="class")
misclass.unpruned = sum(OJ.test$Purchase != pred.unpruned)
misclass.unpruned/length(pred.unpruned)
```

```
## [1] 0.1962963
```

```
pred.pruned = predict(oj.pruned, OJ.test, type="class")
misclass.pruned = sum(OJ.test$Purchase != pred.pruned)
misclass.pruned/length(pred.pruned)
```

```
## [1] 0.1962963
```

The test error rate are the same, which is 0.196.