hw5

Nan Tang

5/31/2020

Problem 1

(a)

$$\begin{aligned}
p(X) &= \sum_y p(x|Y)p(Y) \\
&= p(x|Y=1)p(Y=1) + p(x|Y=2)p(Y=2) \\
&= \frac{1}{4} \cdot I(-4 < x < -2) + \frac{1}{4} \cdot I(2 < x < 4)
\end{aligned}$$

Marginal distribution of X follows uniform distribution on seperate intervals [-4, -2] and [2, 4], pdf of X is $\frac{1}{4}$ if x falls in intervals.

$$\begin{aligned}
p(Y=1|X \in [-4,-2]) &= [p(X|Y=1) \cdot p(Y=1)]/[p(X)] \\
&= (\frac{1}{2} \cdot 1 \cdot \frac{1}{2})/(\frac{1}{4}) \\
&= 1 \\
p(Y=2|X \in [-4,-2]) &= 0, \text{ since } p(x \in [-4,-2]|Y=2) = 0
\end{aligned}$$

similiarly, $p(Y=2|X \in [2,4]) = 1$, $p(Y=1|X \in [2,4]) = 0$

(b)

Since the conditional probability distribution of Y base on X can be represented as

$$\begin{aligned}
p(Y=1|X \in [-4,-2]) &= 1 \\
p(Y=2|X \in [-4,-2]) &= 0 \\
p(Y=1|X \in [2,4]) &= 0 \\
p(Y=2|X \in [2,4]) &= 1
\end{aligned}$$

$$\begin{aligned}
p(Y=1|X \in [-4,-2]) &> p(Y=2|X \in [-4,-2]) \\
p(Y=2|X \in [2,4]) &> p(Y=1|X \in [2,4])
\end{aligned}$$

Therefore, estimator of bayes rule for y is $f_B(x \in [-4,-2]) = 1$, $f_B(x \in [2,4]) = 2$.
In this case, risk is zero, since $p(Y=2|X \in [-4,-2]) = p(Y=1|X \in [2,4]) = 0$, their sum is zero as well.

(c)

For any query point (x_0, y_0), where $x_0 \in [-4, -2]$ or $[2, 4]$, $y_0 = 1$ or 2. Misclassification only occure when all samples (x_i) are draw from one interval, while query point x_0 in another interval.

For example, in the case when all sample points (x_i, y_i) satisfies $x_i \in [-4, -2]$, then classification rule S will classify any y_0 as 1, even if $x_0 \in [2, 4]$.

Conditional probability of Y given X is pure on both intervals of X, the training risk for KNN is zero. Risk only exists on independent test data:

$$
\begin{aligned}
p(y_0 \neq f_1(\hat{x_0}; S)) &= p(\text{all sample } x_i \in [-4, -2] \cap x_0 \in [2, 4]) + p(\text{all sample } x_i \in [2, 4] \cap x_0 \in [-4, -2]) \\
&= p(\text{all sample } x_i \in [-4, -2])p(x_0 \in [2, 4]) + p(\text{all sample } x_i \in [2, 4])p(x_0 \in [-4, -2]) \\
&= (\frac{1}{2})^n \frac{1}{2} + (\frac{1}{2})^n \frac{1}{2} \text{ , since x uniformly distributed on two intervals} \\
&= (\frac{1}{2})^n, \text{ where n is sample size}
\end{aligned}
$$

(d)

Risk for KNN classification with K=3 occurs when there are less than two (one or zero) training data x_i in the same interval as query point x_0.

For example, when only query point $x_0 \in [2, 4]$ and one sample point $x_j \in [2, 4]$, while other sample points $x_{i \neq j} \in [-4, -2]$. Classification rule will incorrectly predict $f\hat{x}_0$ as 1, since two of the three nearest training points have value $y_i = 1$.

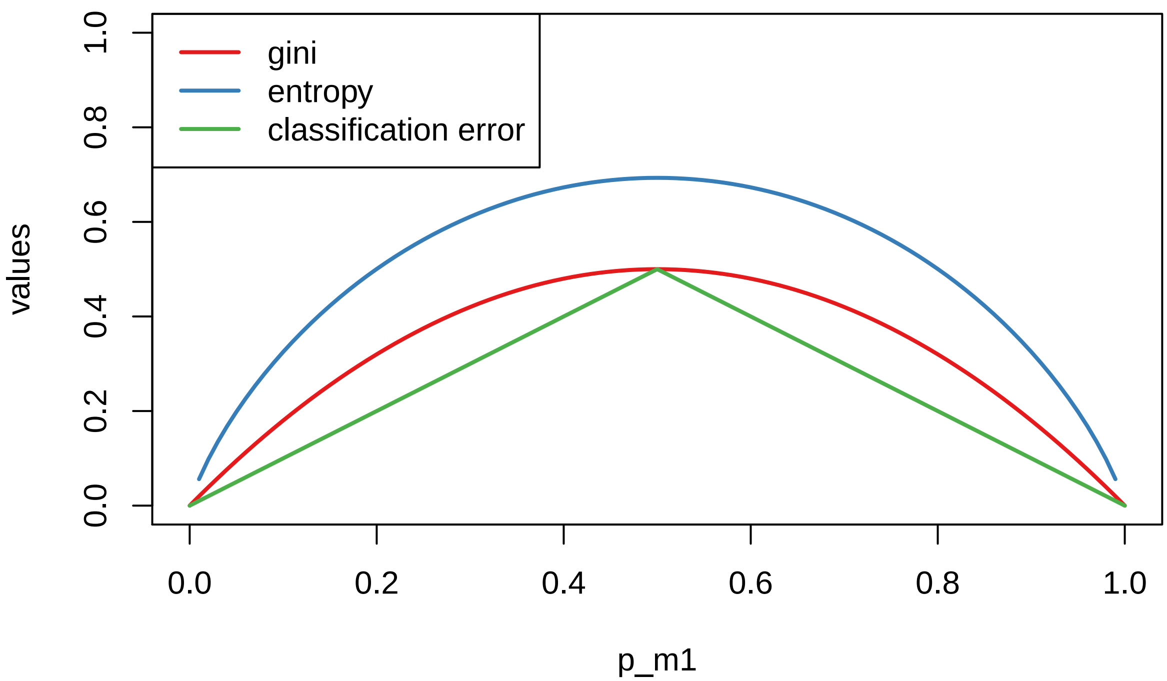
$$
\begin{aligned}
p(y_0 \neq f_3(\hat{x_0}; S)) &= p(\text{less than two sample } x_i \in [-4, -2] \cap x_0 \in [-4, -2]) + p(\text{less than two sample} x_i \in [2, 4] \cap x_0 \in [2, 4]) \\
&= p(\text{all sample } x_i \in [-4, -2] \cap x_0 \in [2, 4]) + p(\text{all sample } x_i \in [2, 4] \cap x_0 \in [-4, -2]) \\
&+ p(\text{only one } x_i \in [2, 4] \cap x_0 \in [2, 4]) + p(\text{only one } x_i \in [-4, -2] \cap x_0 \in [-4, -2]) \\
&= p(\text{all sample } x_i \in [-4, -2])p(x_0 \in [2, 4]) + p(\text{all sample } x_i \in [2, 4])p(x_0 \in [-4, -2]) \\
&+ p(\text{only one } x_i \in [2, 4])p(x_0 \in [2, 4]) + p(\text{only one } x_i \in [-4, -2])p(x_0 \in [-4, -2]) \\
&= (\frac{1}{2})^n \frac{1}{2} + (\frac{1}{2})^n \frac{1}{2} + \binom{n}{1} (\frac{1}{2})^n \frac{1}{2} + \binom{n}{1} (\frac{1}{2})^n \frac{1}{2} \\
&= (\frac{1}{2})^n + \binom{n}{1} (\frac{1}{2})^n \\
&= (n + 1)(\frac{1}{2})^n
\end{aligned}
$$

(e)

In this case, 1-nearest neighbor classifier has smaller risk than 3-nearest neighbor classifier.

8.4.3

```
p_m1 = seq(from=0, to=1, by=0.01)

gini_val = 2 * p_m1 * (1-p_m1)

entropy_val = - (p_m1 * log(p_m1) + (1-p_m1) * log(1-p_m1))

## 1 - max(p, 1-p)
classerror_val = 1 - pmax(p_m1, 1-p_m1)

colors = brewer.pal(n = 3, name = "Set1")

plot(NA, NA, xlim=c(0, 1), ylim=c(0, 1), xlab='p_m1', ylab='values')
lines(p_m1, gini_val, col=colors[1], lwd=2)
lines(p_m1, entropy_val, col=colors[2], lwd=2)
lines(p_m1, classerror_val, col=colors[3], lwd=2)

legend('topleft', legend=c('gini', 'entropy', 'classification error'), col=colors, lwd=2)
```



The plot implies gini index and entropy are more sensitive to p_{mi}'s with small value.

8.4.9

(a)

```
set.seed(1)

train_index <- sample(1:nrow(OJ), 800)
train_dt <- OJ[train_index,]
test_dt <- OJ[-train_index,]
```

(b)

```r
train_tree <- tree(formula=Purchase~., data=train_dt)
summary(train_tree)
```

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = train_dt)
## Variables actually used in tree construction:
## [1] "LoyalCH"       "PriceDiff"     "SpecialCH"     "ListPriceDiff"
## [5] "PctDiscMM"
## Number of terminal nodes:  9
## Residual mean deviance:  0.7432 = 587.8 / 791
## Misclassification error rate: 0.1588 = 127 / 800
```

The tree has 9 terminal nodes, the training error rate is 0.1588.

(c)

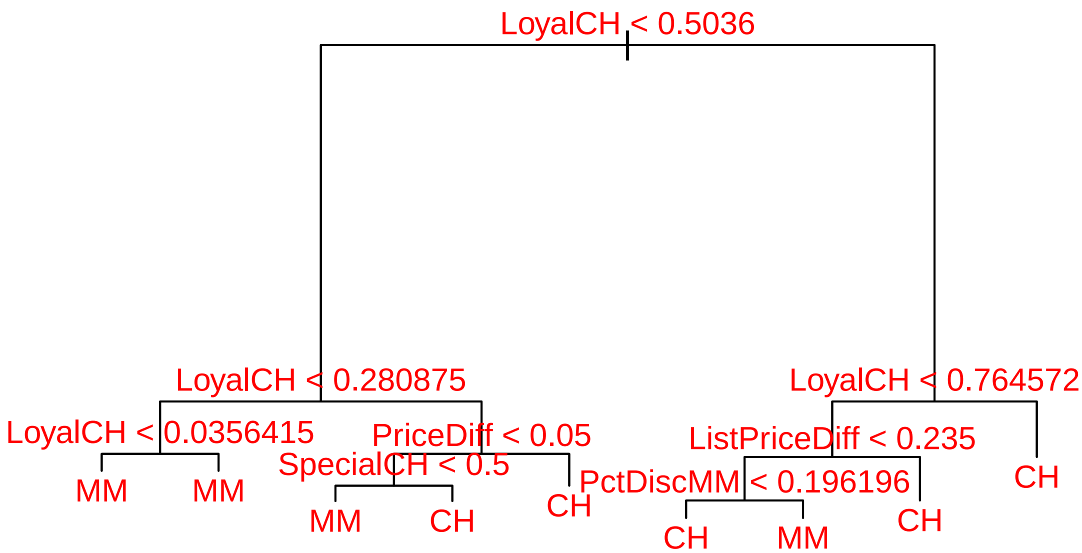
```r
train_tree
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 800 1073.00 CH ( 0.60625 0.39375 )
##    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 435  337.90 CH ( 0.86897 0.13103 )
##      6) LoyalCH < 0.764572 174  201.00 CH ( 0.73563 0.26437 )
##       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 > 0.235 102   65.43 CH ( 0.90196 0.09804 ) *
##      7) LoyalCH > 0.764572 261   91.20 CH ( 0.95785 0.04215 ) *
```

Pick the terminal node (7) LoyalCH > 0.764572 261 91.20 CH (0.95785 0.04215) * as example. (7) is a branch of (3), while (3) is a branch of root. If 'LoyalCh' > 0.5036 and 'LoyalCH' > 0.7646, then we can classify 'Purchase' as CH.
This leaf (15) contains 261 training observations; 95.8% are CH and 4.2% are MM; deviance is 91.2.

(d)

```r
plot(train_tree)
text(train_tree, col='red')
```



for example: this tree model will predict 'Purchase' of an object with 'LoyalCH' < 0.0356 as MM; 'Purchase' of an object with 'LoyalCH' < 0.2809 and 'PriceDiff' > 0.05 as CH.

(e)

```
test_pred <- predict(train_tree, test_dt, type='class')

table(test_dt[,'Purchase'], test_pred)
```

```
##     test_pred
##       CH  MM
##   CH 160   8
##   MM  38  64
```

```
test_err <- (table(test_dt[,'Purchase'], test_pred)[1,2] + table(test_dt[,'Purchase'], test_pred)[2,1] 
print(test_err)
```

```
## [1] 0.1703704
```

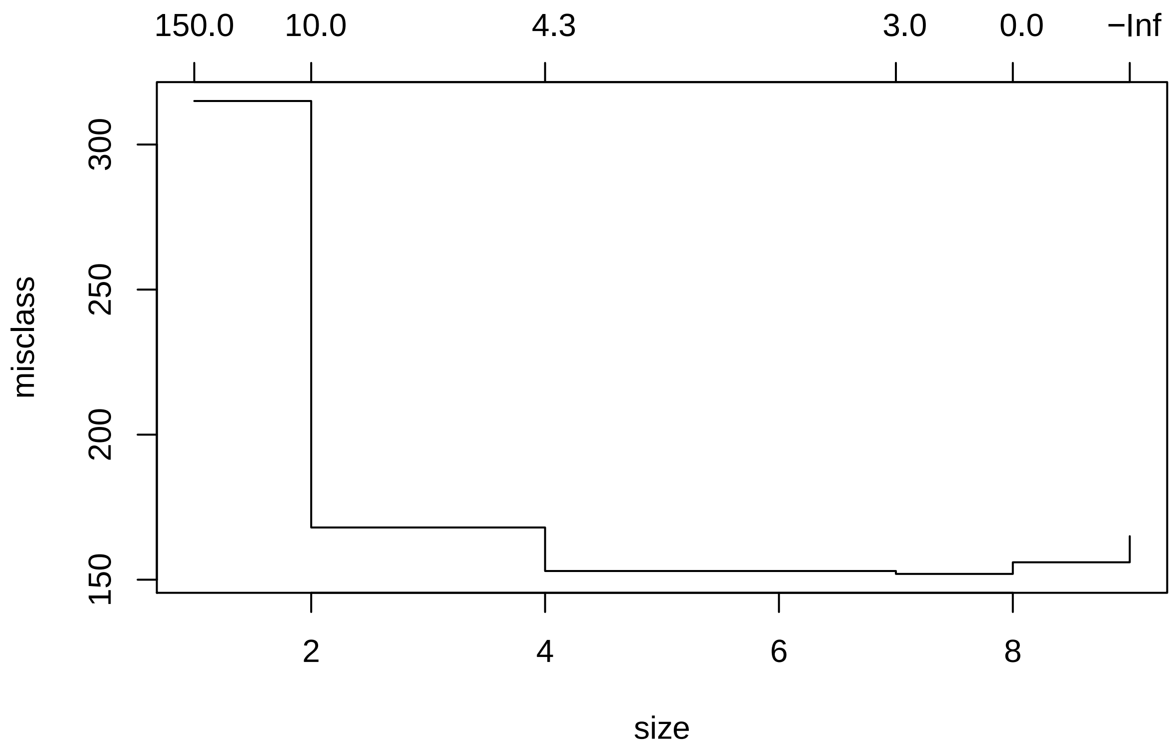
Test error rate is 17.04%

(f)

```
set.seed(2)

train_tree_cv <- cv.tree(train_tree, FUN=prune.misclass, K=5)
```

(g)

```
plot(train_tree_cv)
```



(h)

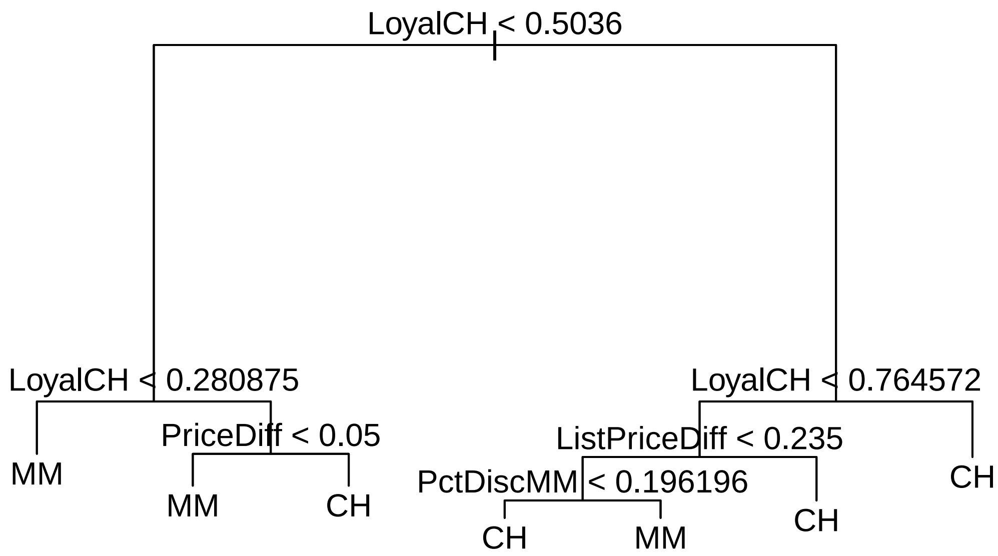
```
opt_size <- train_tree_cv$size[train_tree_cv$dev == min(train_tree_cv$dev)]

print(opt_size)
```

```
## [1] 7
```

Tree size 7 has lowest mis-classification rate in training cross-validation.

(i)

```
opt_tree <- prune.misclass(train_tree, best=7)

plot(opt_tree)
text(opt_tree)
```



(j)

```r
opt_train_pred <- predict(opt_tree, train_dt, type='class')

table(train_dt[,'Purchase'], opt_train_pred )
```

```
##     opt_train_pred
##       CH  MM
##   CH 441  44
##   MM  86 229
```

```r
opt_train_err <- (table(opt_train_pred, train_dt[,'Purchase'])[1, 2] + table(opt_train_pred, train_dt[,
print(opt_train_err)
```

```
## [1] 0.1625
```

Training error rate of tree with 7 terminal nodes is 16.25%. Comparing to unpruned tree model, training error rate of pruned model is slightly higher.

(k)

```r
opt_test_pred <- predict(opt_tree, test_dt, type='class')

table(test_dt[, 'Purchase'], opt_test_pred)
```

```
##     opt_test_pred
##       CH  MM
##   CH 160   8
##   MM  36  66
```

```r
opt_test_err <- (table(test_dt[, 'Purchase'], opt_test_pred)[1, 2] + table(test_dt[, 'Purchase'], opt_te
print(opt_test_err)
```

```
## [1] 0.162963
```

Test error rate of tree model with 7 terminal nodes is 16.3%. Approximately 1% lower than testing error rate of unpruned model, implying the pruned model performs better on testing data.