Statistical Learning for Biological and Information Systems
Problem Set #3

Chenye Yang cy2540@columbia.edu

October 22, 2019

P1

(a)

Ans:
Assume $A =$ "receive an A", $\bar{A} =$ "not receive an A". Thus:

$$\log \frac{p_A(X)}{p_{\bar{A}}(X)} = \beta_0 + \beta_1 X_1 + \beta_2 X_2$$
$$= -6 + 0.05 X_1 + X_2$$

For $X_1 = 40,\ X_2 = 3.5$,

$$\log \frac{p_A(X)}{1 - p_{\bar{A}}(X)} = -6 + 0.05 \times 40 + 3.5 = -0.5$$

$\Rightarrow$

$$p_A(X) = 0.38$$

Therefore, the probability that a student who studies for 40 hours and has an undergrad GPA of 3.5 gets an A in the class is 0.38

(b)

Ans:
When $p_A(X) = 0.5$, $\log \frac{p_A(X)}{p_{\bar{A}}(X)} = 0$

$$\begin{cases} -6 + 0.05 X_1 + X_2 = 0 \\ X_2 = 3.5 \end{cases}$$

$\Rightarrow$

$$X_1 = 50$$

Therefore, the student in part (a) need to study 50 hours to have a 50% chance of getting an A in the class.

P2

Ans:
Given:

$$\begin{array}{lllll} Y = \text{Yes} & \bar{X} = 10 & \hat{\sigma}^2 = 36 & \pi_{\text{Yes}} = 0.8 & X \sim N(10, 36) \\ Y = \text{No} & \bar{X} = 0 & \hat{\sigma}^2 = 36 & \pi_{\text{No}} = 0.2 & X \sim N(0, 36) \end{array}$$

Thus,

$$\begin{aligned} P(Y = \text{Yes}|X = 4) &= \frac{\pi_{\text{Yes}} f_{\text{Yes}}(X)}{\pi_{\text{Yes}} f_{\text{Yes}}(X) + \pi_{\text{No}} f_{\text{No}}(X)} \\ &= \frac{0.8 \times \frac{1}{6\times\sqrt{2\pi}} e^{-\frac{(4-10)^2}{2\times 36}}}{0.8 \times \frac{1}{6\times\sqrt{2\pi}} e^{-\frac{(4-10)^2}{2\times 36}} + 0.2 \times \frac{1}{6\times\sqrt{2\pi}} e^{-\frac{(4-0)^2}{2\times 36}}} \\ &= \frac{4 \times e^{-1/2}}{4 \times e^{-1/2} + e^{-2/9}} \\ &= 0.752 \end{aligned}$$

Therefore, the probability that a company will issue a dividend this year given that its percentage profit was $X = 4$ last year is 0.752

P3

Ans:
Assume $\boldsymbol{X_1} = [1, 0.0]^T,\ \boldsymbol{X_2} = [1, 0.2]^T,\ \boldsymbol{X_3} = [1, 0.4]^T,\ \boldsymbol{X_4} = [1, 0.6]^T,\ \boldsymbol{X_5} = [1, 0.8]^T,\ \boldsymbol{X_6} = [1, 1.0]^T$ and $y_1 = 0,\ y_2 = 0,\ y_3 = 0,\ y_4 = 1,\ y_5 = 0,\ y_6 = 1$ and $\boldsymbol{\beta} = [\beta_0, \beta_1]^T$.
The log-likelihood function, $\ell(\boldsymbol{\beta}) = \ell(\beta_0, \beta_1)$:

$$\begin{aligned}
\ell(\boldsymbol{\beta}) &= \sum_{i=1}^{6} \left[y_i \log p\left(\boldsymbol{X_i}; \boldsymbol{\beta}\right) + (1 - y_i) \log\left(1 - p\left(\boldsymbol{X_i}; \boldsymbol{\beta}\right)\right)\right] \\
&= \sum_{i=1}^{6} \left[y_i \boldsymbol{\beta}^T \boldsymbol{X_i} - \log\left(1 + e^{\boldsymbol{\beta}^T \boldsymbol{X_i}}\right)\right] \\
&= -\log(1 + e^{\boldsymbol{\beta}^T \boldsymbol{X_1}}) - \log(1 + e^{\boldsymbol{\beta}^T \boldsymbol{X_2}}) - \log(1 + e^{\boldsymbol{\beta}^T \boldsymbol{X_3}}) + \boldsymbol{\beta}^T \boldsymbol{X_4} - \log(1 + e^{\boldsymbol{\beta}^T \boldsymbol{X_4}}) \\
&\quad - \log(1 + e^{\boldsymbol{\beta}^T \boldsymbol{X_5}}) + \boldsymbol{\beta}^T \boldsymbol{X_6} - \log(1 + e^{\boldsymbol{\beta}^T \boldsymbol{X_6}}) \\
&= -\log(1 + e^{\beta_0}) - \log(1 + e^{\beta_0 + \beta_1 \times 0.2}) - \log(1 + e^{\beta_0 + \beta_1 \times 0.4}) + (\beta_0 + \beta_1 \times 0.6) \\
&\quad - \log(1 + e^{\beta_0 + \beta_1 \times 0.6}) - \log(1 + e^{\beta_0 + \beta_1 \times 0.8}) + (\beta_0 + \beta_1 \times 1.0) - \log(1 + e^{\beta_0 + \beta_1 \times 1.0})
\end{aligned} \tag{1}$$

To maximize the log-likelihood, we set its derivatives to zero:

$$\frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{6} \boldsymbol{X_i}\left[y_i - p\left(\boldsymbol{X_i}; \boldsymbol{\beta}\right)\right] = 0 \tag{2}$$

In practice, we use the Newton–Raphson algorithm to solve the maximizing problem:

$$\begin{aligned}
\frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= \sum_{i=1}^{6} \boldsymbol{X_i}\left[y_i - p\left(\boldsymbol{X_i}; \boldsymbol{\beta}\right)\right] \\
\frac{\partial^2 \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T} &= -\sum_{i=1}^{6} \boldsymbol{X_i} \boldsymbol{X_i}^T p\left(\boldsymbol{X_i}; \boldsymbol{\beta}\right)\left(1 - p\left(\boldsymbol{X_i}; \boldsymbol{\beta}\right)\right) \\
\boldsymbol{\beta}^{\text{new}} &= \boldsymbol{\beta}^{\text{old}} - \left(\frac{\partial^2 \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T}\right)^{-1} \frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}
\end{aligned} \tag{3}$$

Code:

```
beta_old = c(0,0)
beta_new = c(0,0)
# initiate X
X = matrix(data = rep(0,12), nrow = 2, ncol = 6)
for (i in 1:6){
  X[,i] = c(1,(i-1)/5)
}
# initiate y
y = c(0, 0, 0, 1, 0, 1)
# initiate derivatives
l_b = c(0, 0)
for (i in 1:6){
```

```
  l_b = l_b + X[,i]*(y[i]-exp(X[,i] %*% beta_old)/(1+exp(X[,i] %*% beta_old)))
}
# initiate second derivatives
l_bb = matrix(data = rep(0,4), nrow = 2, ncol = 2)
for (i in 1:6){
  l_bb = l_bb - (X[,i] %*% t(X[,i]))*((exp(X[,i] %*% beta_old)/(1+exp(X[,i] %*% beta_old)))*(1-(exp(X[,i] %*% beta_old)/(1+exp(X[,i] %*% beta_old)))))[1]
}
# iteration
for (i in 1:10){
  beta_new = t(t(beta_old)) - solve(l_bb) %*% t(t(l_b))
  beta_old = t(t(beta_new))
}
t(t(beta_new))
```

Result:

```
> t(t(beta_new))
          [,1]
[1,] -23.80952
[2,]  34.28571
```

The estimated coefficients of this logistic regression problem is $\hat{\beta}_0 = -23.81,\ \hat{\beta}_1 = 34.29$.

P4

Ans:
From given,

$$\begin{aligned}\operatorname{cov}[\boldsymbol{Y}] &= \operatorname{cov}[\boldsymbol{AX}] \\ &= \boldsymbol{A}\operatorname{cov}[\boldsymbol{X}]\boldsymbol{A}^T \\ &= \boldsymbol{A\Sigma A}^T = \boldsymbol{I}\end{aligned} \tag{4}$$

Because $\boldsymbol{\Sigma}$ is symmetric, assume $\boldsymbol{\Sigma}$ has non-degenerate eigenvalues $\lambda_1,\ \lambda_2$ and corresponding linearly independent eigenvectors $v_1,\ v_2$. Define:

$$\begin{aligned}\boldsymbol{Q} &= \begin{bmatrix} \boldsymbol{v_1} & \boldsymbol{v_2} \end{bmatrix} = \begin{bmatrix} v_1^1 & v_2^1 \\ v_1^2 & v_2^2 \end{bmatrix} \\ \boldsymbol{\Lambda} &= \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}\end{aligned} \tag{5}$$

Thus we have,

$$\boldsymbol{\Sigma} = \boldsymbol{Q\Lambda Q}^T = \boldsymbol{Q\Lambda Q}^{-1} \tag{6}$$

Therefore,

$$\begin{aligned}\boldsymbol{AQ\Lambda Q}^T\boldsymbol{A}^T &= \boldsymbol{I} \\ (\boldsymbol{AQ})\boldsymbol{\Lambda}(\boldsymbol{AQ})^T &= \boldsymbol{I}\end{aligned} \tag{7}$$

$\Rightarrow$

$$\begin{aligned}\boldsymbol{AQ} &= \left(\boldsymbol{\Lambda}^{-1}\right)^{\frac{1}{2}} \\ \boldsymbol{A} &= \left(\boldsymbol{\Lambda}^{-1}\right)^{\frac{1}{2}}\boldsymbol{Q}^{-1}\end{aligned} \tag{8}$$

Using Matlab program to calculate the eigenvalues and eigenvectors of $\boldsymbol{\Lambda}$:
Code:

```
# Matlab
syms sigma_1 sigma_2 rho
A = [sigma_1^2, rho*sigma_1*sigma_2;
     rho*sigma_1*sigma_2, sigma_2^2];
[eigenvector,eigenvalue] = eig(A)
```

Result:

$$\begin{aligned}\boldsymbol{Q} &= \begin{bmatrix} \boldsymbol{v_1} & \boldsymbol{v_2} \end{bmatrix} \\ \boldsymbol{v_1} &= \begin{bmatrix} (\sigma_1^2/2 - (4\rho^2\sigma_1^2\sigma_2^2 + \sigma_1^4 - 2\sigma_1^2\sigma_2^2 + \sigma_2^4)^{1/2}/2 + \sigma_2^2/2)/(\rho\sigma_1\sigma_2) - \sigma_2/(\rho\sigma_1) \\ 1 \end{bmatrix} \\ \boldsymbol{v_2} &= \begin{bmatrix} ((4\rho^2\sigma_1^2\sigma_2^2 + \sigma_1^4 - 2\sigma_1^2\sigma_2^2 + \sigma_2^4)^{1/2}/2 + \sigma_1^2/2 + \sigma_2^2/2)/(\rho\sigma_1\sigma_2) - \sigma_2/(\rho\sigma_1) \\ 1 \end{bmatrix} \\ \boldsymbol{\Lambda} &= \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \\ \lambda_1 &= \sigma_1^2/2 - (4*\rho^2*\sigma_1^2*\sigma_2^2 + \sigma_1^4 - 2*\sigma_1^2*\sigma_2^2 + \sigma_2^4)^{1/2}/2 + \sigma_2^2/2 \\ \lambda_2 &= (4*\rho^2*\sigma_1^2*\sigma_2^2 + \sigma_1^4 - 2*\sigma_1^2*\sigma_2^2 + \sigma_2^4)^{1/2}/2 + \sigma_1^2/2 + \sigma_2^2/2\end{aligned} \tag{9}$$

Therefore, use Matlab program to calculate $\boldsymbol{A}$:

Code:

```
A = (inv(eigenvalue))^(1/2)*inv(eigenvector)
```

Result:

$$
\begin{aligned}
\boldsymbol{A} &= \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \\
a_{11} &= -(2^{1/2}\rho\sigma_1\sigma_2(1/(\sigma_1^2 - (4\rho^2\sigma_1^2\sigma_2^2 + \sigma_1^4 - 2\sigma_1^2\sigma_2^2 + \sigma_2^4)^{1/2} + \sigma_2^2))^{1/2})/(4\rho^2\sigma_1^2\sigma_2^2 + \sigma_1^4 - 2\sigma_1^2\sigma_2^2 + \sigma_2^4)^{1/2} \\
a_{12} &= (2^{1/2}(1/(\sigma_1^2 - (4\rho^2\sigma_1^2\sigma_2^2 + \sigma_1^4 - 2\sigma_1^2\sigma_2^2 + \sigma_2^4)^{1/2} + \sigma_2^2))^{1/2}((4\rho^2\sigma_1^2\sigma_2^2 + \sigma_1^4 - 2\sigma_1^2\sigma_2^2 + \sigma_2^4)^{1/2} + \sigma_1^2 - \sigma_2^2)) \\
&\quad /(2(4\rho^2\sigma_1^2\sigma_2^2 + \sigma_1^4 - 2\sigma_1^2\sigma_2^2 + \sigma_2^4)^{1/2}) \\
a_{21} &= (2^{1/2}\rho\sigma_1\sigma_2(1/((4\rho^2\sigma_1^2\sigma_2^2 + \sigma_1^4 - 2\sigma_1^2\sigma_2^2 + \sigma_2^4)^{1/2} + \sigma_1^2 + \sigma_2^2))^{1/2})/(4\rho^2\sigma_1^2\sigma_2^2 + \sigma_1^4 - 2\sigma_1^2\sigma_2^2 + \sigma_2^4)^{1/2} \\
a_{22} &= (2^{1/2}(1/((4\rho^2\sigma_1^2\sigma_2^2 + \sigma_1^4 - 2\sigma_1^2\sigma_2^2 + \sigma_2^4)^{1/2} + \sigma_1^2 + \sigma_2^2))^{1/2}((4\rho^2\sigma_1^2\sigma_2^2 + \sigma_1^4 - 2\sigma_1^2\sigma_2^2 + \sigma_2^4)^{1/2} - \sigma_1^2 + \sigma_2^2)) \\
&\quad /(2(4\rho^2\sigma_1^2\sigma_2^2 + \sigma_1^4 - 2\sigma_1^2\sigma_2^2 + \sigma_2^4)^{1/2})
\end{aligned} \tag{10}
$$

P5

Ans:
Because the variance of each population is the same (σ^2), we have:

$$\begin{aligned} \hat{\sigma}_k^2 &= \frac{1}{n_k - 1} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2 \\ \mathbb{E}\left[\hat{\sigma}_k^2\right] &= \sigma^2 \end{aligned} \tag{11}$$

Thus,

$$\begin{aligned} \mathbb{E}\left[\hat{\sigma}^2\right] &= \mathbb{E}\left[\sum_{k=1}^{K} \alpha_k \hat{\sigma}_k^2\right] = \sum_{k=1}^{K} \alpha_k \mathbb{E}\left[\hat{\sigma}_k^2\right] \\ &= \sigma^2 \sum_{k=1}^{K} \alpha_k = \sigma^2 \end{aligned} \tag{12}$$

Because x_i obeys Gaussian distribution, thus

$$\frac{(n_k - 1)\,\hat{\sigma}_k^2}{\sigma^2} \sim \mathcal{X}_{n_k-1}^2$$

Therefore,

$$\begin{aligned} \mathrm{var}\left[\frac{(n_k - 1)\,\hat{\sigma}_k^2}{\sigma^2}\right] &= 2\,(n_k - 1) \\ \frac{(n_k - 1)^2}{\sigma^4}\,\mathrm{var}\left[\hat{\sigma}_k^2\right] &= 2\,(n_k - 1) \\ \mathrm{var}\left[\hat{\sigma}_k^2\right] &= \frac{2\sigma^4}{n_k - 1} \end{aligned} \tag{13}$$

Considering,

$$\begin{aligned} \mathrm{var}\left[\hat{\sigma}^2\right] &= \mathrm{var}\left[\sum_{k=1}^{K} \alpha_k \hat{\sigma}_k^2\right] \\ &= \sum_{k=1}^{K} \alpha_k^2\,\mathrm{var}\left[\hat{\sigma}_k^2\right] \\ &= \sum_{k=1}^{K} \alpha_k^2 \frac{2\sigma^4}{n_k - 1} \\ &= 2\sigma^4 \sum_{k=1}^{K} \frac{\alpha_k^2}{n_k - 1} \sum_{i=1}^{K} \alpha_i \\ &= 2\sigma^4 \sum_{k=1}^{K} \frac{\alpha_k^2}{n_k - 1} \sum_{i=1}^{K} \frac{n_i - 1}{n - K} \\ &= \frac{2\sigma^4}{n - K} \sum_{k=1}^{K} \frac{\alpha_k^2}{n_k - 1} \sum_{i=1}^{K} (n_i - 1) \end{aligned} \tag{14}$$

According to Cauchy-Schwarz inequality,

$$\begin{aligned} \sum_{k=1}^{K} \frac{\alpha_k^2}{n_k - 1} \sum_{k=1}^{K} (n_k - 1) &\geq \left(\sum_{k=1}^{K} \frac{\alpha_k}{\sqrt{n_k - 1}} \times \sqrt{n_k - 1} \right)^2 \\ &= \left(\sum_{k=1}^{K} \alpha_k \right) \\ &= 1 \end{aligned} \tag{15}$$

Therefore,

$$\text{var}\left[\hat{\sigma}^2\right] \geq \frac{2\sigma^4}{n - K} \tag{16}$$

The equality happens if and only if:

$$\frac{\alpha_1}{n_1 - 1} = \frac{\alpha_2}{n_2 - 1} = ... = \frac{\alpha_K}{n_K - 1} \tag{17}$$

which can be satisfied by $\alpha_k = (n_k - 1)/(n - K)$, i.e.

$$\frac{\alpha_1}{n_1 - 1} = \frac{\alpha_2}{n_2 - 1} = ... = \frac{\alpha_K}{n_K - 1} = \frac{1}{n - K} \tag{18}$$

Therefore, $\alpha_k = (n_k - 1)/(n - K)$ minimizes the variance of σ^2 under the Gaussian assumption.

P6

Ans:
Majority vote approach:
X is classified to *Red*: 6 estimates, X is classified to *Green*: 4 estimates. Thus, X is classified to *Red*.

Average probability:

$$\begin{aligned}\bar{\mathbb{P}}[\text{Class is Red}|X] &= \frac{\sum^n \mathbb{P}[\text{Class is Red}|X]}{n} \\ &= \frac{0.1 + 0.15 + 0.2 + 0.2 + 0.55 + 0.6 + 0.6 + 0.65 + 0.7 + 0.75}{10} \\ &= 0.45\end{aligned}$$

Because $\bar{\mathbb{P}}[\text{Class is Red}|X] = 0.45 < 0.5$, X is classified to *Green*.

P7

(a)

Code:

```
# (a)
rm(list=ls())
set.seed(1000)
load('/Users/yangchenye/Downloads/OJ.rda')
dim(OJ)
OJ=na.omit(OJ) # remove incomplete cases
dim(OJ)
names(OJ)

# 800: the sample size
smp_size = floor(800)
train_ind = sample(seq_len(nrow(OJ)), size = smp_size)
# select train data and test data
train = subset(OJ[train_ind, ])
test = subset(OJ[-train_ind, ])
```

(b)

Code:

```
# (b)
library(tree)
tree.fit = tree(train$Purchase~., data=train)
summary(tree.fit)
```

Result:

```
> summary(tree.fit)

Classification tree:
tree(formula = train$Purchase ~ ., data = train)
Variables actually used in tree construction:
[1] "LoyalCH"     "PriceDiff"    "SalePriceMM"
Number of terminal nodes:  8
Residual mean deviance:  0.7486 = 592.9 / 792
Misclassification error rate: 0.16 = 128 / 800
```

Ans:
The training error rate is 128/800=0.16
The tree has 8 terminal nodes.

(c)

Code:

```
# (c)
tree.fit
```

Result:

```
> tree.fit
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

1) root 800 1066.00 CH ( 0.61500 0.38500 )
2) LoyalCH < 0.5036 353  422.60 MM ( 0.28612 0.71388 )
4) LoyalCH < 0.276142 170  131.00 MM ( 0.12941 0.87059 )
8) LoyalCH < 0.035047 57   10.07 MM ( 0.01754 0.98246 ) *
9) LoyalCH > 0.035047 113  108.50 MM ( 0.18584 0.81416 ) *
5) LoyalCH > 0.276142 183  250.30 MM ( 0.43169 0.56831 )
10) PriceDiff < 0.05 78   79.16 MM ( 0.20513 0.79487 ) *
11) PriceDiff > 0.05 105  141.30 CH ( 0.60000 0.40000 ) *
3) LoyalCH > 0.5036 447  337.30 CH ( 0.87472 0.12528 )
6) LoyalCH < 0.764572 187  206.40 CH ( 0.75936 0.24064 )
12) SalePriceMM < 2.125 120  156.60 CH ( 0.64167 0.35833 )
24) PriceDiff < -0.35 16   17.99 MM ( 0.25000 0.75000 ) *
25) PriceDiff > -0.35 104  126.70 CH ( 0.70192 0.29808 ) *
13) SalePriceMM > 2.125 67   17.99 CH ( 0.97015 0.02985 ) *
7) LoyalCH > 0.764572 260   91.11 CH ( 0.95769 0.04231 ) *
```

Ans:
For the terminal node:

$$8) \quad LoyalCH < 0.035047 \quad 57 \quad 10.07 \quad MM \quad (0.017540.98246)*$$

"8)" is the node number.
"$LoyalCH < 0.035047$" is a two-column matrix of the labels for the left and right splits at the node.
"57" is the number of cases reaching that node ($LoyalCH < 0.035047$).
"10.07" is the deviance of the node.
"MM" is the fitted value at the node (the mean for regression trees, a majority class for classification trees).
"(0.017540.98246)" is a matrix of fitted probabilities for each response level.

(d)

Code:

```
# (d)
plot(tree.fit)
text(tree.fit,pretty=0)
```

Result:

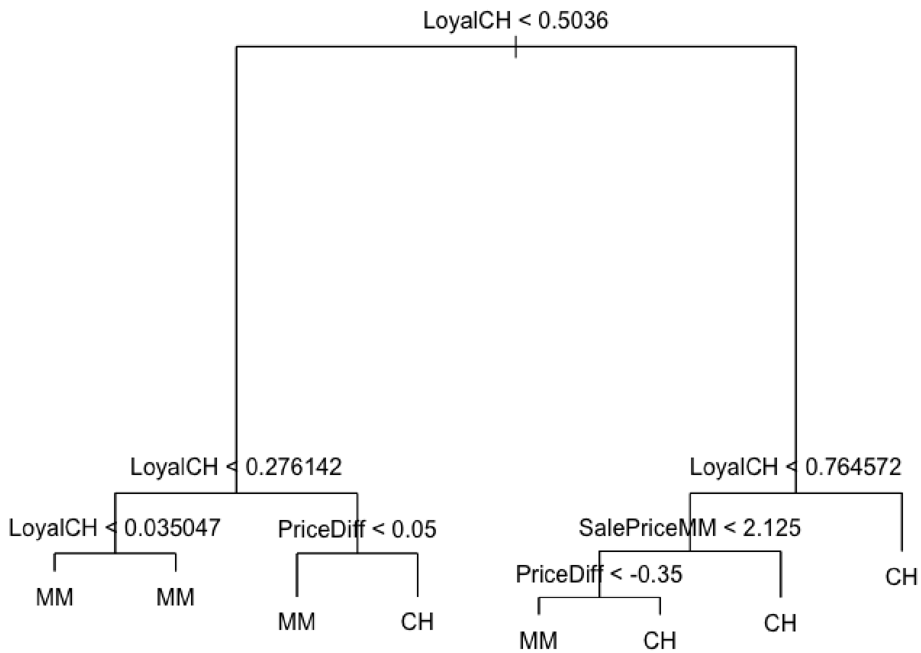

Figure 1: Regression tree with 8 terminal nodes

Ans:
For customers with $LoyalCH < 0.276142$, they will choose Minute Maid Orange Juice.
For customers with $0.276142 < LoyalCH < 0.5036$ & $PriceDiff < 0.05$, they will choose Minute Maid Orange Juice.
For customers with $0.5036 < LoyalCH < 0.764572$ & $SalePriceMM < 2.125$ & $PriceDiff < -0.35$, they will choose Minute Maid Orange Juice.
Otherwise, customers will choose Citrus Hill Orange Juice.

(e)

Code:

```
# (e)
tree.pred = predict(tree.fit, test, type="class")
with(test, table(tree.pred, Purchase))
```

Result:

```
> with(test, table(tree.pred, Purchase))
          Purchase
tree.pred  CH  MM
       CH 150  38
       MM  11  71
```

Ans:
The test error rate is $\frac{11+38}{270} = 0.18$

(f)

Code:

```
# (f)
cv.fit = cv.tree(tree.fit)
cv.fit
cv.fit$size[which.min(cv.fit$dev)]
```

Result:

```
> cv.fit
$size
[1] 8 7 6 5 4 3 2 1

$dev
[1]  673.8589  677.7294  675.2341  729.4281  727.2145  772.9214
   773.5642
[8] 1067.3829

$k
[1]      -Inf  11.87503  12.41171  29.77434  31.80546  39.82936
   41.38321
[8] 306.37571

$method
[1] "deviance"

attr(,"class")
[1] "prune"         "tree.sequence"
> cv.fit$size[which.min(cv.fit$dev)]
[1] 8
```

Ans:
The optimal tree size is 8.

(g)

Code:

```
# (g)
par(mfrow = c(1, 2))
# default plot
plot(cv.fit)
# better plot
plot(cv.fit$size, cv.fit$dev / smp_size, type = "b",
     xlab = "Tree Size", ylab = "CV Misclassification Rate")
```

Result:

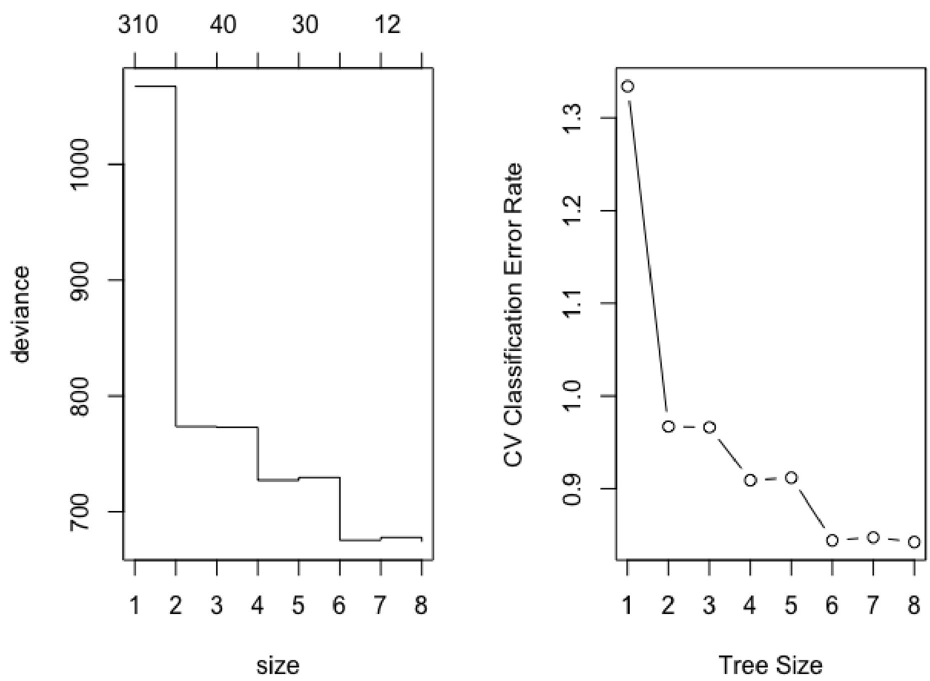

Figure 2: Error rate to tree size

(h)

Code:

```
# (h)
cv.best = cv.fit$size[which.min(cv.fit$dev / smp_size)] # misclassification rate of each tree
cv.best
```

Result:

```
> cv.best
[1] 8
```

Ans:
The tree size corresponds to the lowest cross-validated classification error rate is 8.

(i)

Code:

```
# (i)
prune.fit = prune.tree(tree.fit, best = 5)
plot(prune.fit)
text(prune.fit, pretty=0)
```

Result:

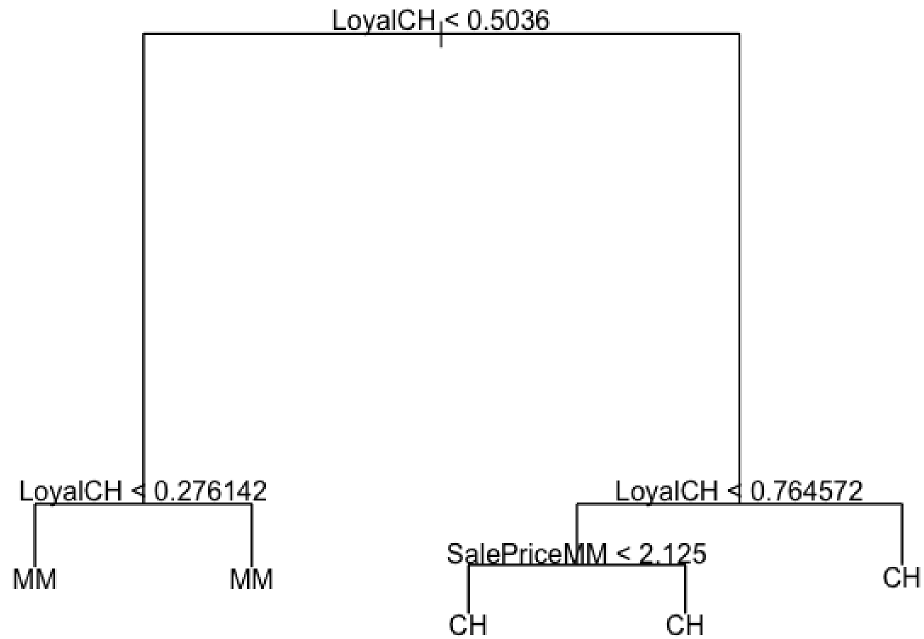

Figure 3: Pruned classification tree

(j)

Code:

```
# (j)
summary(prune.fit)
```

Result:

```
> summary(prune.fit)

Classification tree:
snip.tree(tree = tree.fit, nodes = c(12L, 4L, 5L))
Variables actually used in tree construction:
[1] "LoyalCH"     "SalePriceMM"
Number of terminal nodes:  5
Residual mean deviance:  0.8138 = 646.9 / 795
Misclassification error rate: 0.1962 = 157 / 800
```

Ans:
Training error rate of unpruned tree: 128/800=0.16
Training error rate of pruned tree: 157/800=0.1962
The training error rate of pruned tree is higher.

(k)

Code:

```
# (k)
```

```
tree.pred = predict(prune.fit, test, type="class")
with(test, table(tree.pred, Purchase))
```

Result:

```
> with(test, table(tree.pred, Purchase))
         Purchase
tree.pred  CH  MM
       CH 129  25
       MM  32  84
```

Ans:
Test error rate of unpruned tree: $(11 + 38)/270 = 0.18$
Test error rate of pruned tree: $(32 + 25)/270 = 0.21$
The test error rate of pruned tree is higher.