Homework 3

E6690: Statistical Learning for Bio & Info Systems

P1. Suppose we collect data for a group of students in a statistics class with variables X_1 (hours studied), X_2 (undergrad GPA), and Y (receive an A). We fit a logistic regression and produce estimated coefficient, $\hat{\beta}_0 = -6$, $\hat{\beta}_1 = 0.05$, $\hat{\beta}_2 = 1$.

(a) (5pt) Estimate the probability that a student who studies for 40 hours and has an undergrad GPA of 3.5 gets an A in the class.

Answer: We just need to put the parameters in the equation for predicted probability

$$\hat{p}(X) = \frac{e^{-6+0.05X_1+X_2}}{1+e^{-6+0.05X_1+X_2}} = 0.3775.$$

(b) (5pt) How many hours would the student in part (a) need to study to have a 50% chance of getting an A in the class?

Answer: According to the equation, we have

$$\hat{p}(X) = \frac{e^{-6+0.05X_1+3.5}}{1+e^{-6+0.05X_1+3.5}} = 0.5,$$

which is equivalent to $e^{-6+0.05X_1+3.5} = 1$. By taking $\log$ of both sides, we get $X_1 = 50$.

P2. (10pt) Suppose that we wish to predict whether a given stock will issue a dividend this year ("Yes" or "No") based on X, last year's percent profit. We examine a large number of companies and discover that the mean value of X for companies that issued a dividend was $\bar{X} = 10$, while the mean for those that didn't was $\bar{X} = 0$. In addition, the variance of X for these two sets of companies was $\hat{\sigma}^2 = 36$. Finally, 80% of companies issued dividends. Assuming that X follows a normal distribution, predict the probability that a company will issue a dividend this year given that its percentage profit was $X = 4$ last year.

Hint: Use Bayes' theorem.

Answer: For the given parameters, the equation for $p_k(x)$ is

$$p_1(4) = \frac{0.8\,e^{-\frac{(4-10)^2}{72}}}{0.8\,e^{-\frac{(4-10)^2}{72}} + 0.2\,e^{-\frac{(4-0)^2}{72}}} = 0.752.$$

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

P3. (20pt) Consider $X = [0.0\ 0.2\ 0.4\ 0.6\ 0.8\ 1.0]^\top$ as the independent variable and $y =$ [false false false true false true]$^\top$ as the response. Write down the log-likelihood function, $l(\beta_0, \beta_1)$, for the logistic regression problem and first order optimality conditions. Note that this problem of finding optimal (β_0, β_1) can only be solved numerically. Use NewtonRaphson algorithm from Section 4.4.1, pp. 120-121, [ESL] book and perform 10 iterations. Hint: Use library(matlib) for calculating matrix inverses.

Answer: Recall

$$\mathbb{P}[Y = \text{true} \,|\, X = x] = p(x) = \frac{e^{\beta_0+\beta_1 x}}{1 + e^{\beta_0+\beta_1 x}}.$$

The likelihood function is given by

$$L(\beta_0, \beta_1) = \prod_{i=1}^{6} p(x_i)^{y_i}(1 - p(x_i))^{(1-y_i)},$$

where $y_i = 0$ and $y_i = 1$ correspond to $y_i =$ false and $y_i =$ true, respectively. Consequently,

$$\begin{aligned}\ln L(\beta_0, \beta_1) = l(\beta_0, \beta_1) &= \sum_{i=1}^{6} \left[y_i \log(p(x_i)) + (1 - y_i)\,(1 - \log(p(x_i)))\right] \\ &= \sum_{i=1}^{6} \left[-\log(1 + e^{\beta_0+\beta_1 x_i}) + y_i(\beta_0 + \beta_1 x_i)\right].\end{aligned}$$

The objective is to maximize $l(\beta_0, \beta_1)$. To this end, we find the gradient of $l(\boldsymbol{\beta})$ and set it equal to 0 (the first order optimality condition):

$$\begin{aligned}\nabla l(\beta_0, \beta_1) = g(\boldsymbol{\beta}) &= \begin{bmatrix} g_1(\boldsymbol{\beta}) \\ g_2(\boldsymbol{\beta}) \end{bmatrix} \\ &= \begin{bmatrix} -\sum_{i=1}^{6} \left(\frac{e^{\beta_0+\beta_1 x_i}}{1+e^{\beta_0+\beta_1 x_i}} - y_i\right) \\ -\sum_{i=1}^{6} \left(\frac{e^{\beta_0+\beta_1 x_i}}{1+e^{\beta_0+\beta_1 x_i}} - y_i\right) x_i \end{bmatrix} \\ &= \begin{bmatrix} -\sum_{i=1}^{6} \frac{e^{\beta_0+\beta_1 x_i}}{1+e^{\beta_0+\beta_1 x_i}} + 2 \\ -\sum_{i=1}^{6} \frac{e^{\beta_0+\beta_1 x_i}}{1+e^{\beta_0+\beta_1 x_i}} x_i + 1.6 \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ 0 \end{bmatrix}.\end{aligned}$$

From Section 4.4.1, pp. 120-121, [ESL], the update step in Newton's method is:

$$\boldsymbol{\beta}^{(n+1)} = \boldsymbol{\beta}^{(n)} - J(\boldsymbol{\beta}^{(n)})^{-1}\, g(\boldsymbol{\beta}^{(n)}),$$

where

$$\begin{aligned}J(\boldsymbol{\beta}) = J(\beta_0, \beta_1) &= \begin{bmatrix} \nabla g_1(\boldsymbol{\beta})^\top \\ \nabla g_2(\boldsymbol{\beta})^\top \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial g_1(\boldsymbol{\beta})}{\partial \beta_0} & \frac{\partial g_1(\boldsymbol{\beta})}{\partial \beta_1} \\ \frac{\partial g_2(\boldsymbol{\beta})}{\partial \beta_0} & \frac{\partial g_2(\boldsymbol{\beta})}{\partial \beta_1} \end{bmatrix} \\ &= \begin{bmatrix} \sum_{i=1}^{6} \frac{e^{\beta_0+\beta_1 x_i}}{(1+e^{\beta_0+\beta_1 x_i})^2} & \sum_{i=1}^{6} \frac{e^{\beta_0+\beta_1 x_i}}{(1+e^{\beta_0+\beta_1 x_i})^2} x_i \\ \sum_{i=1}^{6} \frac{e^{\beta_0+\beta_1 x_i}}{(1+e^{\beta_0+\beta_1 x_i})^2} x_i & \sum_{i=1}^{6} \frac{e^{\beta_0+\beta_1 x_i}}{(1+e^{\beta_0+\beta_1 x_i})^2} x_i^2 \end{bmatrix}.\end{aligned}$$

The corresponding code in R is as follows:

```
> #for calculationg inverse of a matrix, install "matlib" package
> library(matlib)
> x<-c(0,0.2,0.4,0.6,0.8,1)
> y<-c(0,0,0,1,0,1)
> g<-function(beta){
>   f1<- sum(((exp(beta[1]+beta[2]*x))/(1+exp(beta[1]+beta[2]*x)))-y)
>   f2<- sum((((exp(beta[1]+beta[2]*x))/(1+exp(beta[1]+beta[2]*x)))-y)*x)
>   matrix(c(f1,f2),ncol=1,nrow = 2)
> }
> J<-function(beta){
>   f11<-sum((exp(beta[1]+beta[2]*x))/(1+exp(beta[1]+beta[2]*x))^2)
>   f12<-sum(x*exp(beta[1]+beta[2]*x)/((1+exp(beta[1]+beta[2]*x))^2))
>   f21<-f12
>   f22<-sum((x^2)*exp(beta[1]+beta[2]*x)/((1+exp(beta[1]+beta[2]*x))^2))
>   matrix(c(f11,f21,f12,f22),ncol = 2,nrow = 2)
> }
> #starting point for Beta
> beta<-c(-1,1)
> cat("Newton's method- starting point : beta0=",beta[1],",  beta1=",beta[2])
> #Run Newton's method for 10 iterations
> for(i in 1:10){
>   beta<-beta-inv(J(beta))%*%g(beta)
>   cat('\n',"Newton's method- iteration" ,i, "beta0=",beta[1],",  beta1=",beta[2])
> }
> model_glm<-glm(y~x,family = "binomial")
> cat("By using logistic regression function in R, beta0=",model_glm$coefficients[1],
    ",   beta1=",model_glm$coefficients[2])
## Newton's method- starting point : beta0= -1 ,  beta1= 1
## Newton's method- iteration 1 beta0= -2.705639 ,  beta1= 3.841178
## Newton's method- iteration 2 beta0= -3.70133 ,  beta1= 5.200004
## Newton's method- iteration 3 beta0= -4.061292 ,  beta1= 5.675472
## Newton's method- iteration 4 beta0= -4.097643 ,  beta1= 5.722885
## Newton's method- iteration 5 beta0= -4.09797 ,  beta1= 5.723309
## Newton's method- iteration 6 beta0= -4.09797 ,  beta1= 5.723309
## Newton's method- iteration 7 beta0= -4.09797 ,  beta1= 5.723309
## Newton's method- iteration 8 beta0= -4.09797 ,  beta1= 5.723309
## Newton's method- iteration 9 beta0= -4.09797 ,  beta1= 5.723309
## Newton's method- iteration 10 beta0= -4.09797 ,  beta1= 5.723309
## By using logistic regression function in R, beta0= -4.09797 ,   beta1= 5.723309
```

P4. (20pt) Let $\boldsymbol{X} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$. Find the distribution of $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}$. For the bivariate case with

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix},$$

find a 2×2 $\boldsymbol{A}$ such that cov$(\boldsymbol{Y})$ is an identity matrix.

Hint: Consider eigen-decomposition.

Answer: Since $\boldsymbol{X}$ has normal distribution and $\boldsymbol{Y}$ is a linear transformation of $\boldsymbol{X}$, $\boldsymbol{Y}$ is normally distributed, with

$$\mathbb{E}[\boldsymbol{Y}] = \mathbb{E}[\boldsymbol{A}\boldsymbol{X}] = \boldsymbol{A}\mathbb{E}[\boldsymbol{X}] = 0$$

and

$$\begin{aligned}
\operatorname{cov}(\boldsymbol{Y}) = \operatorname{cov}(\boldsymbol{A}\boldsymbol{X}) &= \mathbb{E}\left[(\boldsymbol{A}\boldsymbol{X} - \mathbb{E}(\boldsymbol{A}\boldsymbol{X}))(\boldsymbol{A}\boldsymbol{X} - \mathbb{E}(\boldsymbol{A}\boldsymbol{X}))^\top\right] \\
&= \mathbb{E}\left[\boldsymbol{A}(\boldsymbol{X} - \mathbb{E}(\boldsymbol{X}))(\boldsymbol{X} - \mathbb{E}(\boldsymbol{X}))^\top \boldsymbol{A}^\top\right] \\
&= \boldsymbol{A}\mathbb{E}\left[(\boldsymbol{X} - \mathbb{E}(\boldsymbol{X}))(\boldsymbol{X} - \mathbb{E}(\boldsymbol{X}))^\top\right]\boldsymbol{A}^\top \\
&= \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\top.
\end{aligned}$$

In order to find $\boldsymbol{A}$ such that $\boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\top = I$, we consider the eigenvalues of $\boldsymbol{\Sigma}$. To this end

$$\begin{aligned}
\det(\boldsymbol{\Sigma} - \lambda \boldsymbol{I}) &= (\lambda - \sigma_1^2)(\lambda - \sigma_2^2) - \rho^2\sigma_1^2\sigma_2^2 \\
&= \lambda^2 - \lambda(\sigma_1^2 + \sigma_2^2) + (1-\rho^2)\sigma_1^2\sigma_2^2 = 0.
\end{aligned}$$

This yields

$$\begin{aligned}
\lambda_1 &= \frac{\sigma_1^2 + \sigma_2^2 + \sqrt{(\sigma_1^2 + \sigma_2^2)^2 - 4(1-\rho^2)\sigma_1^2\sigma_2^2}}{2}, \\
\lambda_2 &= \frac{\sigma_1^2 + \sigma_2^2 - \sqrt{(\sigma_1^2 + \sigma_2^2)^2 - 4(1-\rho^2)\sigma_1^2\sigma_2^2}}{2}.
\end{aligned}$$

Since eigenvalues are real and $\boldsymbol{\Sigma}$ is symmetric, $\boldsymbol{\Sigma}$ can be written as

$$\boldsymbol{\Sigma} = \boldsymbol{Q}\Lambda\boldsymbol{Q}^\top = (\boldsymbol{Q}\Lambda^{\frac{1}{2}})(\boldsymbol{Q}\Lambda^{\frac{1}{2}})^\top,$$

where columns of $\boldsymbol{Q}$ are the eigenvectors of $\boldsymbol{\Sigma}$ that form an orthonormal basis ($\boldsymbol{Q}^\top\boldsymbol{Q} = \boldsymbol{I}$) and the diagonal entries of diagonal matrix $\boldsymbol{\Lambda}$ are the corresponding eigenvalues of $\boldsymbol{\Sigma}$. Therefore, we have

$$\boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\top = \boldsymbol{A}(\boldsymbol{Q}\boldsymbol{\Lambda}^{\frac{1}{2}})(\boldsymbol{Q}\boldsymbol{\Lambda}^{\frac{1}{2}})^\top\boldsymbol{A}^\top.$$

Letting $\boldsymbol{A}$ to be $(\boldsymbol{Q}\boldsymbol{\Lambda}^{\frac{1}{2}})^{-1} = \boldsymbol{\Lambda}^{-\frac{1}{2}}\boldsymbol{Q}^{-1} = \boldsymbol{\Lambda}^{-\frac{1}{2}}\boldsymbol{Q}^\top$, yields $\boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\top = \boldsymbol{I}$.

P5. (20pt) Consider K different populations when the mean of each population may be different, but one may assume that the variance of each population is the same (σ^2). Let

$$\hat{\mu}_k = \frac{1}{n_k}\sum_{i:\, y_i = k} x_i \quad \text{and} \quad \hat{\sigma}_k^2 = \frac{1}{n_k - 1}\sum_{i:\, y_i = k}(x_i - \hat{\mu}_k)^2,$$

n_k is the sample size of population k and the pair (x_i, y_i) corresponds to the ith observation; $y_i \in \{1, \ldots, K\}$. For $\alpha_1, \ldots, \alpha_K$, such that $\sum_{i=1}^K \alpha_i = 1$, define an unbiased pooled variance estimator:

$$\hat{\sigma}^2 = \sum_{k=1}^K \alpha_k \hat{\sigma}_k^2.$$

If $n = \sum_{k=1}^K n_k$, show that $\alpha_k = (n_k - 1)/(n - K)$ minimizes the variance of $\hat{\sigma}^2$ under the Gaussian assumption.

Answer: Note that $\hat{\sigma}^2$ is unbiased for all feasible $\{\alpha_k\}$

$$\mathbb{E}\hat{\sigma}^2 = \sum_{k=1}^{K} \alpha_k \mathbb{E}\hat{\sigma}_k^2 = \sum_{k=1}^{K} \alpha_k \sigma^2 = \sigma^2,$$

since $\hat{\sigma}_k^2$ is unbiased. The variance of $\hat{\sigma}^2$ can be estimated as follows:

$$\mathsf{Var}(\hat{\sigma}^2) = \sum_{k=1}^{K} \alpha_k^2 \, \mathsf{Var}(\hat{\sigma}_k^2) = \sum_{k=1}^{K} \alpha_k^2 \frac{2\sigma^4}{n_k - 1},$$

due to

$$\frac{n_k - 1}{\sigma^2}\hat{\sigma}_k^2 \sim \chi^2_{n_k - 1}$$

and the fact that $\mathbb{E}(\mathcal{N}(0,1))^4 = 3, \mathbb{E}(\mathcal{N}(0,1))^2 = 1$. Then, the problem is

$$\min_{\alpha_1,\ldots,\alpha_K} \sum_{k=1}^{K} \frac{\alpha_k^2}{n_k - 1},$$

subject to $\sum_{k=1}^{K} \alpha_k = 1$. The unconstrained version is given by

$$\sum_{k=1}^{K} \frac{\alpha_k^2}{n_k - 1} + \lambda \left(\sum_{k=1}^{K} \alpha_k - 1 \right).$$

Consequently, the first-order optimality condition is, for $k = 1, \ldots, K$,

$$2\frac{\alpha_k}{n_k - 1} + \lambda = 0.$$

That implies that α_k is proportional to $(n_k - 1)$, which in turn yields

$$\alpha_k = \frac{n_k - 1}{n - K}.$$

P6. (10pt) Suppose we produce ten bootstrapped samples from a data set containing red and green classes. We then apply a classification tree to each bootstrapped sample and, for a specific value of X, produce 10 estimates of $\mathbb{P}[\text{Class is Red}|X]$:

$$0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, \text{ and } 0.75.$$

There are two common ways to combine these results together into a single class prediction. One is the majority vote approach. The second approach is to classify based on the average probability. In this problem, what is the final classification under each of these two approaches?

Answer:

```
> prob<- c(0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, 0.75)
> a<-sum(prob >= 0.5)
> b<- sum(prob < 0.5)
> a>b
## [1] TRUE
> mean(prob)
## [1] 0.45
```

In the first approach, the number of red predictions is greater than the number of green predictions based on a 50% threshold, so the final classification would be Red. In the second approach, the average of the probabilities is less than the 50% threshold, which means Green.

P7. This problem involves `OJ` data set which is part of the `ISLR` package.

(a) (2pt) First run set.seed(1000), and then create a training set containing a random sample of 800 observations, and a test set containing the remaining observations.

Answer:

```
> library(ISLR)
> attach(OJ)
> set.seed(1000)
>
> train <- sample(dim(OJ)[1], 800)
> OJ.train <- OJ[train, ]
> OJ.test <- OJ[-train, ]
```

(b) (2pt) Fit a tree to the training data, with `Purchase` as the response and the other variables as predictors. Use the `summary()` function to produce summary statistics about the tree, and describe the results obtained. What is the training error rate? How many terminal nodes does the tree have?

Answer:

```
> 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 only uses three variables: LoyalCH, PriceDiff and WeekofPurchase . It has 7 terminal nodes. Training error rate (misclassification error) for the tree is 0.175.

(c) (3pt) Type in the name of the tree object in order to get a detailed text output. Pick one of the terminal nodes, and interpret the information displayed.

Answer:

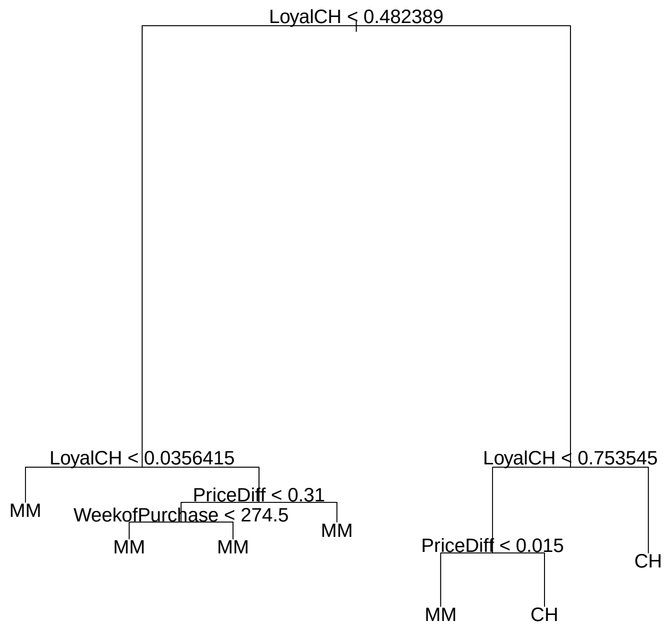
```
> oj.tree
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 800 1069.000 CH ( 0.61125 0.38875 )
##    2) LoyalCH < 0.482389 297  319.600 MM ( 0.22896 0.77104 )
##      4) LoyalCH < 0.0356415 55    9.996 MM ( 0.01818 0.98182 ) *
##      5) LoyalCH > 0.0356415 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 ) *
```

Let's pick terminal node labeled 4. The splitting variable at this node is LoyalCH. The splitting value of this node is 0.0356415. There are 55 points in the subtree below this node. The deviance for all points contained in region below this node is 9.996. A * in the line denotes that this is in fact a terminal node. The prediction at this node is Sales $=$ MM. About 18% points in this node have CH as value of Sales. Remaining 98% points have MM as value of Sales.

(d) (2pt) Create a plot of the tree, and interpret the results.

Answer:

```
> pdf(file="4-d.pdf",width = 10,height = 10)
> plot(oj.tree)
> text(oj.tree)
> dev.off()
```



LoyalCH is the most important variable of the tree, in fact top 3 nodes contain LoyalCH. If $LoyalCH < 0.035$, the tree predicts MM. If $LoyalCH > 0.7535$, the tree predicts CH. For intermediate values of LoyalCH, the decision also depends on the values of PriceDiff and WeekofPurchase.

(e) (3pt) Predict the response on the test data, and produce a confusion matrix comparing the test labels to the predicted test labels. What is the test error rate?

Answer:

```
> 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
```

Test error rate is equal to $(41 + 12)/270 = 0.196$.

(f) (2pt) Apply the `cv.tree()` function to the training set in order to determine the optimal tree size.

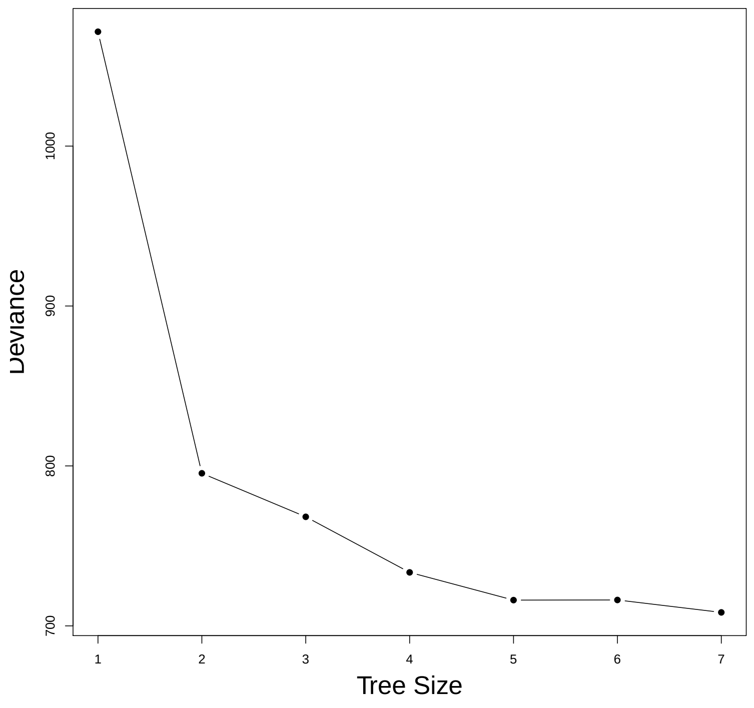
Answer:

```
> cv.oj <- cv.tree(oj.tree, FUN = prune.tree)
```

(g) (3pt) Produce a plot with tree size on the x-axis and cross-validated classification error rate on the y-axis.

Answer:

```
> pdf(file="4-g.pdf",width = 10,height = 10)
> plot(cv.oj$size, cv.oj$dev, type = "b", xlab = "Tree Size", ylab = "Deviance",pch=19)
> dev.off()
```



(h) (1pt) Which tree size corresponds to the lowest cross-validated classification error rate?

Answer: Size of 7 gives lowest cross-validation error.

(i) (3pt) Produce a pruned tree corresponding to the optimal tree size obtained using cross-validation. If cross-validation does not lead to selection of a pruned tree, then create a pruned tree with five terminal nodes.

Answer:

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

(j) (2pt) Compare the training error rate between the pruned and unpruned tree. Which is higher?

Answer:

```
> 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
```

Misclassification error of pruned tree is exactly same as that of original tree.

(k) (2pt) Compare the test error rates between the pruned and unpruned trees. Which is higher?

Answer:

```
> 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
```

Pruned and unpruned trees have same test error rate of 0.196.