# MATH 4322 Homework 5

Phu Nguyen

Fall 2021

## Problem 1

The questions relate to the following plots:

a) Sketch the tree corresponding to the partition of the predictor space illustrated on the left-hand plot. The numbers inside the boxes indicate the mean of $Y$ within each region.

b) Create a diagram similar to the left-hand plot using the tree illustrated in the right-hand plot. You should divide up the predictor space inot the correct regions, and indicate the mean for each region.

## Problem 2

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 $P(\text{Class is Red}|X)$:

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

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

- The estimates of $P(green) = 0.9$ and \$ P(red) = 0.1\$. Anything probability above 0.5 is $P(red)$ and any probability below is $P(green)$. Therefore, we see that as per estimates of $P(classisRed|X)$, class red receives the maximum votes of 6 versus class of green which have 4 votes. THe final classification under the majority vote mothod is red.

## Problem 3

Provide a detailed explanation of the algorithm that is used to fit a regression tree.

- Recursive Binary Splitting: Steps
- $R(j, s) = X|X_j < s$ is the region of predictor space $(X_1, ....X_p)$ where $X_j < s(J^{th} predictorislessthanvalues)$.

Steps of recursive binary splitting

- 1. Start with full predictor space $R = (X_1, ...., X_p)$
- 2. For any $j$ and $s$, we define the pair of half-planes
  - $R_1(j, s) = X|X_j < s$

  - $R_2(j,s) = X|X_j \geq s$

- 3. We seek for values of $j$ and $s$ that minimize

  - $RSS = \sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}R_1)^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}R_2)^2$

- 4. We repeat steps 2 and 3 trying to split the data further by minimizing the RSS, until a stopping criterion (e.g. "no region contains more than five observations") is reached.

- 5. We get a final set of regions $R_1, ...., R_J$, and later predict the response for a test observation from region $R_j$ via the mean of training observations $\in R_j, j = 1, ...., J$

## Problem 4

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

a) Create a training set containing a random sample of 800 observations, and a test set containing the remaining observations.

```
library(ISLR)
set.seed(100)

train = sample(1:nrow(OJ), 800) #800 observation
OJ.train = OJ[train, ]
OJ.test = OJ[-train,]
```

b) 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?

```
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"     "ListPriceDiff" "SalePriceMM"
## Number of terminal nodes:  8
## Residual mean deviance:  0.7235 = 573 / 792
## Misclassification error rate: 0.1588 = 127 / 800
```

- The training error rate is 0.1588 and there are 8 terminal nodes.

c) 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.

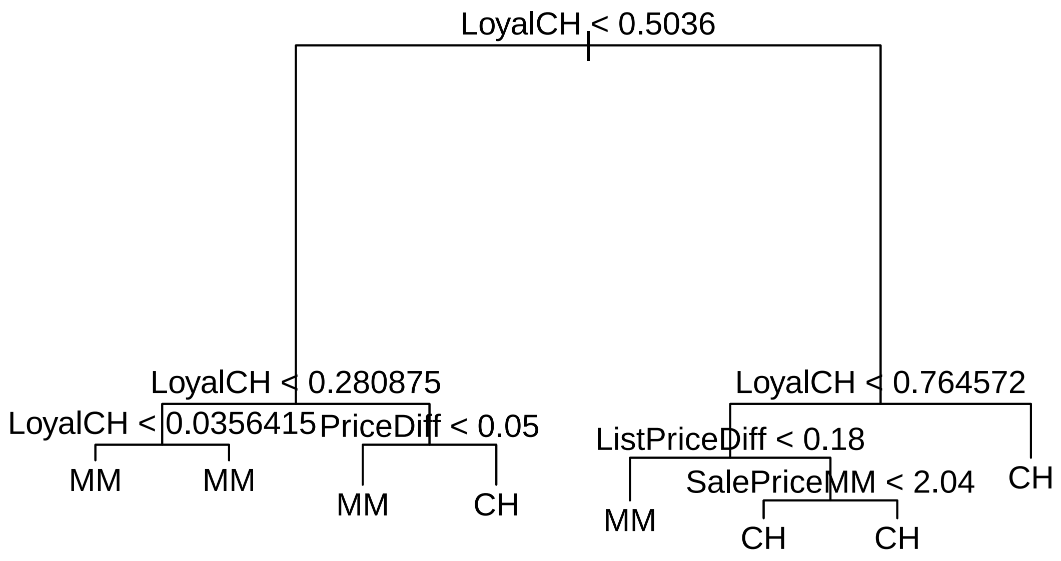
```
OJ.tree
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 800 1070.00 CH ( 0.61000 0.39000 )
##    2) LoyalCH < 0.5036 355  420.20 MM ( 0.27887 0.72113 )
##      4) LoyalCH < 0.280875 167  130.20 MM ( 0.13174 0.86826 )
##        8) LoyalCH < 0.0356415 59   10.14 MM ( 0.01695 0.98305 ) *
##        9) LoyalCH > 0.0356415 108  106.40 MM ( 0.19444 0.80556 ) *
##      5) LoyalCH > 0.280875 188  254.40 MM ( 0.40957 0.59043 )
##       10) PriceDiff < 0.05 81   74.58 MM ( 0.17284 0.82716 ) *
##       11) PriceDiff > 0.05 107  144.90 CH ( 0.58879 0.41121 ) *
##    3) LoyalCH > 0.5036 445  336.80 CH ( 0.87416 0.12584 )
##      6) LoyalCH < 0.764572 180  204.60 CH ( 0.74444 0.25556 )
##       12) ListPriceDiff < 0.18 49   66.27 MM ( 0.40816 0.59184 ) *
##       13) ListPriceDiff > 0.18 131  101.10 CH ( 0.87023 0.12977 )
##         26) SalePriceMM < 2.04 51   59.94 CH ( 0.72549 0.27451 ) *
##         27) SalePriceMM > 2.04 80   25.59 CH ( 0.96250 0.03750 ) *
##      7) LoyalCH > 0.764572 265   85.16 CH ( 0.96226 0.03774 ) *
```

- I will pick terminal node 8, it is a terminal node because of the asterisk. The split criterion is LoyalCH $< 0.0356$, the number of observation is 59 with a deviance of 10.14 and an overall prediction for the branch of MM. About 1.6% of the observations in that branch take the value of CH, and the remaining 98% take the value of MM.

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

```
plot(OJ.tree)
text(OJ.tree, pretty = 0)
```



* The top three nodes contain LoyalCH, the most important indicator of Purchase appears to be LoyalCH. since the first branch differentiates the intensity of customer brand loyalty to CH.

e) 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?

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

```
##
## OJ.tree.pred  CH  MM
##           CH 142  36
##           MM  23  69
```

```
(23+26)/(142+36+23+69)
```

```
## [1] 0.1814815
```

- The test error rate is 0.1814

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

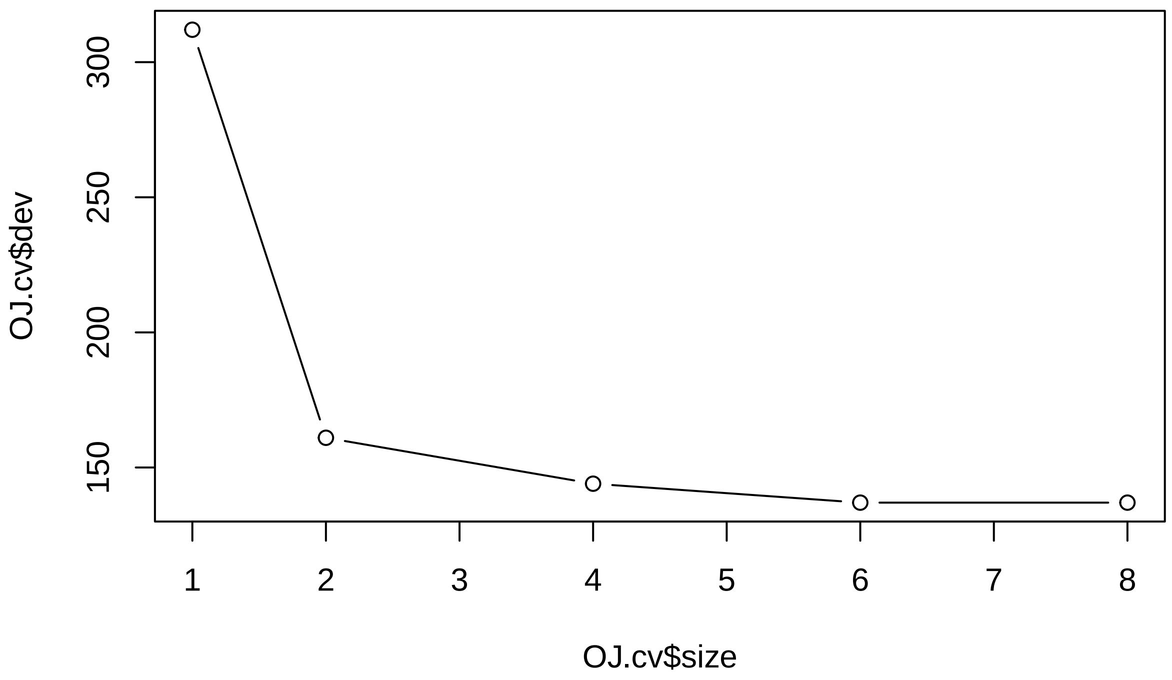
```
OJ.cv = cv.tree(OJ.tree, FUN = prune.misclass)
OJ.cv
```

```
## $size
## [1] 8 6 4 2 1
##
## $dev
## [1] 137 137 144 161 312
##
## $k
## [1]  -Inf   0.0   4.5   9.5 157.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

- The optimal tree size is 4

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

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



h) Which tree size corresponds to the lowest cross-validated classification error rate?
* Tree size of 4 terminal node have lower classification error rate.

i) 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.

```r
prune.OJ = prune.misclass(OJ.tree,best = 4)
summary(prune.OJ)
```

```
##
## Classification tree:
## snip.tree(tree = OJ.tree, nodes = 4:3)
## Variables actually used in tree construction:
## [1] "LoyalCH"   "PriceDiff"
## Number of terminal nodes:  4
## Residual mean deviance:  0.8624 = 686.5 / 796
## Misclassification error rate: 0.17 = 136 / 800
```

- Tree size of terminal node 4 have a little lower misclassifications error rate compare to 8 terminal node.

j) Compare the training error rates between the pruned and unpruned trees. Which is higher?

- Unpruned trees have a training error rate of 0.18 and pruned tree have a training error rate of 0.17. Therefore unpruned tree have a higher training error rate.

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

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

```
##
## test.pred  CH  MM
##        CH 152  42
##        MM  13  63
```

```r
(13+42)/(152+42+13+63)
```

```
## [1] 0.2037037
```

- Test error rate for unpruned tree = 0.2037
- Test error rate for pruned tree = 0.2037
- Both test error rate are the same

## Problem 5

We will use the `Carseats` data set that is in the `ISLR` package to see to predict `Sales` using regression trees and related approaches.

a) Split the data set into a training set and a test set.

```
train = sample(1:nrow(Carseats),nrow(Carseats)/2)
carseats.train = Carseats[train,]
carseats.test = Carseats[-train,]
```

b) Fit a regression tree to the training set. Plot the tree, and interpret the results. What test MSE do you obtain?

```
Carseats$ShelveLoc = as.factor(Carseats$ShelveLoc)
Carseats$Urban = as.factor(Carseats$Urban)
Carseats$US = as.factor(Carseats$US)
carseat.tree = tree(Sales ~., data = carseats.train)
summary(carseat.tree)
```

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = carseats.train)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Advertising" "CompPrice"   "Age"
## Number of terminal nodes:  16
## Residual mean deviance:  2.367 = 435.5 / 184
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -3.17500 -0.99630 -0.08358  0.00000  0.99420  4.68600
```

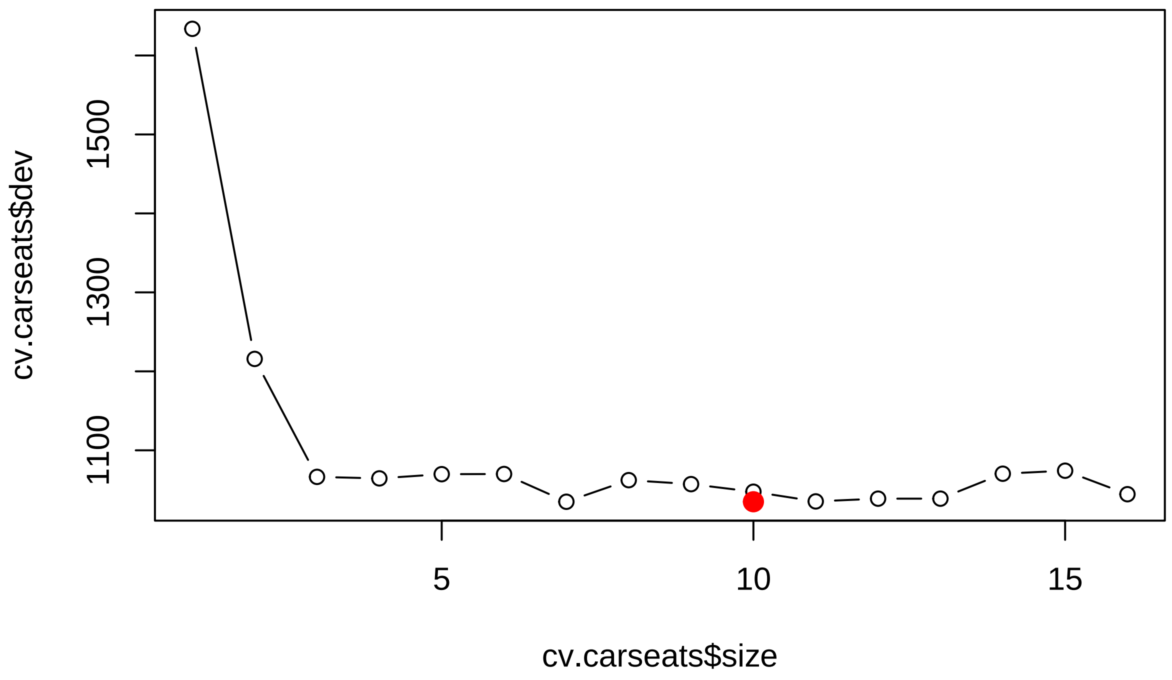
```
yhat = predict(carseat.tree, newdata = carseats.test)
mean((yhat - carseats.test$Sales)^2)
```

```
## [1] 4.636518
```

- The MSE is 2.943

c) Use cross-validation in order to determine the optimal level of tree complexity. Does pruning the tree improve the test MSE?

```
cv.carseats = cv.tree(carseat.tree)
plot(cv.carseats$size, cv.carseats$dev, type = "b")
tree.min <- which.min(cv.carseats$dev)
points(tree.min, cv.carseats$dev[tree.min], col = "red", cex = 2, pch = 20)
```



* Tree size 7 is selected by cross-validation

```
prune.carseats = prune.tree(carseat.tree, best = 7)
yhat = predict(prune.carseats, newdata = carseats.test)
mean((yhat - carseats.test$Sales)^2)
```

```
## [1] 5.412116
```

- The MSE is just a little higher, hence it does not improve the MSE

d) Use the bagging approach in order to analyze this data. What test MSE do you obtain? Use the `importance()` function to determine which variables are most important.

```
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.
```

```
bag.carseats = randomForest(Sales ~.,
                            data = carseats.train,
                            mtry = 10,
                            ntree = 500,
                            importance = TRUE)
yhat.bag = predict(bag.carseats, newdata = carseats.test)
mean((yhat.bag - carseats.test$Sales)^2)
```

```
## [1] 2.88899
```

- The MSE is now 2.08

```
importance(bag.carseats)
```

```
##               %IncMSE IncNodePurity
## CompPrice   23.3937561    172.852349
## Income       6.4534197     68.190496
## Advertising 19.7505535    166.393701
## Population  -2.8906125     58.341759
## Price       50.3342926    458.352689
## ShelveLoc   62.5977602    470.210734
## Age          9.9553934     95.686567
## Education    6.1777174     46.617836
## Urban       -0.5641462      6.863093
## US           1.1154939      6.505655
```

- Price and ShelveLoc is the most important variables

(e) Use random forests to analyze this data. What test MSE do you obtain? Use the `importance()` function to determine which variables aremost important. Describe the effect of $m$, the number of variables considered at each split, on the error rate obtained.

```
rf.carseats = randomForest(Sales ~.,
                           data = carseats.train,
                           mtry = 3,
                           ntree = 500,
                           importance = TRUE)
yhat.rf = predict(rf.carseats, newdata = carseats.test)
mean((yhat.rf - carseats.test$Sales)^2)
```

```
## [1] 3.284984
```

- MSE = 1.84, the best out of all method.

## Problem 6

We will use boosting to predict `Salary` in the `Hitters` data set.

a) Remove the observations for whom the salary information is unknown, and then log-transform the salaries.

```
Hitters = na.omit(Hitters)
```

b) Create a training set consisting of the first 200 observations, and a test set consisting of the remaining observations.

```
train = Hitters[1:200,]
testing = Hitters[-(1:200),]
```

c) Perform boosting on the training set with 1,000 trees. What is the the test set MSE? Compare this to the MSE for the regression tree we did in class.

```
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```
set.seed(1)
boost.hitters = gbm(log(Salary) ~.,
                    data = Hitters,
                    distribution = "gaussian",
                    n.tree = 1000)
yhat.boost = predict(boost.hitters, newdata = testing)
```

```
## Using 1000 trees...
```

```
#mean((yhat.boost - log(testing))^2)
```

d) Which variables appear to be the most imporatnt predictors in the boosted model?

- CatBat and CRuns

e) Now apply bagging to the training set. What is the test set MSE for this approach?