# Homework #8: Tree Ensembles

**Hyun Ko**

Due: Wed Nov 2 | 11:45am

**DS 6030 | Fall 2022 | University of Virginia**

This is an **independent assignment**. Do not discuss or work with classmates.

## Required R packages and Directories

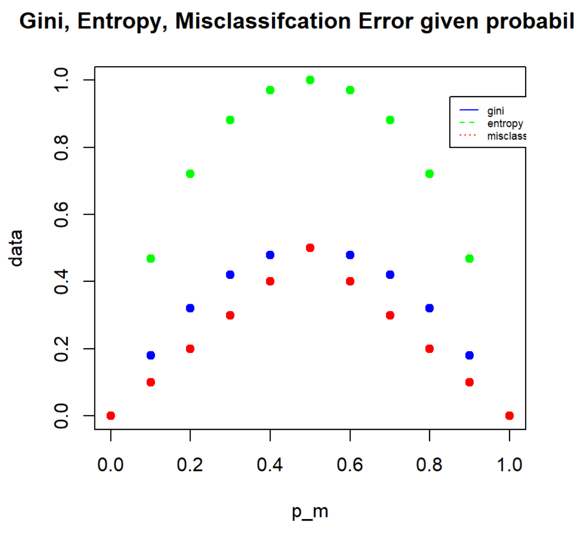
```
data.dir = 'https://mdporter.github.io/DS6030/data/' # data directory
library(R6030)     # functions for DS-6030
library(tidyverse) # functions for data manipulation
library(randomForest)
library(MASS)
library(gridExtra)
```

## Problem 1: Tree Splitting for classification

Consider the Gini index, classification error, and entropy impurity measures in a simple classification setting with two classes.

Create a single plot that displays each of these quantities as a function of $p_m$, the estimated probability of an observation in node $m$ being from class 1. The x-axis should display $p_m$, ranging from 0 to 1, and the y-axis should display the value of the Gini index, classification error, and entropy.

```
get_info = function(p_m){
  gini = 1 - ( (p_m ^ 2) + (1 - p_m) ^ 2)
  entropy = -(p_m * log2(p_m)) - ((1 - p_m) * log2(1 - p_m))
  misclass = 1 - max(p_m, 1-p_m)

  return(c(gini,entropy,misclass))
}

plot(0:1, 0:1, type = "n", xlab = "p_m", ylab = "data", main ="Gini, Entropy, Mi
sclassifcation Error given probability")
for (p_m in seq(0,1,0.1)){
  return_vals = get_info(p_m)
  points(p_m, return_vals[1], col = "blue", bg = "blue", pch = 19)
  points(p_m, return_vals[2], col = "green", bg = "green", pch = 19)
  points(p_m, return_vals[3], col = "red", bg = "red", pch = 19)
}

legend(0.85, 0.95, legend=c("gini", "entropy", "misclass error"), col=c("blue",
"green", "red"), lty=1:3, cex=0.55)
```



## Problem 2: Combining bootstrap estimates

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 the following 10 estimates of $\Pr(\text{Class is Red} \mid X)$: $\{0.2, 0.25, 0.3, 0.4, 0.4, 0.45, 0.7, 0.85, 0.9, 0.9\}$.

a. ISLR 8.2 describes the *majority vote* approach for making a hard classification from a set of bagged classifiers. What is the final classification for this example using majority voting?

We can do hard classification by comparing each probability with the threshold, 0.5. Since we have total 6 sample probabilities below 0.5, while 4 above 0.5, we can conclude that it is green.

b. An alternative is to base the final classification on the average probability. What is the final classification for this example using average probability?

```
mean(p_red)
```

```
#> [1] 0.535
```

Since the average bootstrap probabilities is above 0.5, we can conclude that it is red.

c. Suppose the cost of mis-classifying a Red observation (as Green) is twice as costly as mis-classifying a Green observation (as Red). How would you modify both approaches to make better final classifications under these unequal costs? Report the final classifications.

As the cost of mis-classifying Red as Green (False Negative) is twice the cost of mis-classifying Green as Red (False Positive), we should be careful classifying an observation as Green, therefore leading to lower the threshold (below 0.5). Let's say we set the threshold as 0.333. Then, first 3 observations are considered green and last 7 observations are considered red. As the majority of bootstrap samples judge the observation as Red, our final classification would be Red.

## Problem 3: Random Forest Tuning

Random forest has several tuning parameters that you will explore in this problem. We will use the `Boston` housing data from the `MASS` R package (See the ISLR Lab in section 8.3.3 for example code).

- Note: remember that `MASS` can mask the `dplyr::select()` function.

a. List all of the random forest tuning parameters in the `randomForest::randomForest()` function. Note any tuning parameters that are specific to classification or regression problems. Indicate the tuning parameters you think will be most important to optimize?

ntree = number of trees to grow

mtry = number of variables randomly sampled as candidates at each split

nodesize = minimum size of terminal nodes

maxnodes = maximum number of terminal nodes trees in the forest can have

The number of variables evaluated for eachsplit (mtry) seems to be the most important parameter among all. Small mtry leads to high bias, while large mtry leads to high variance.

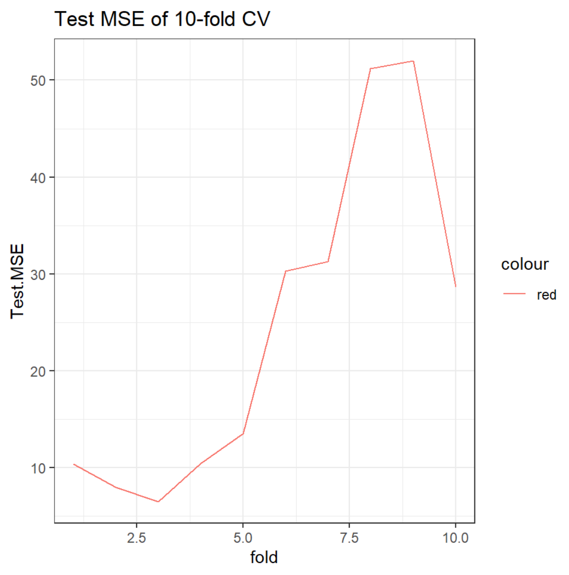
b. Use a random forest model to predict `medv`, the median value of owner-occupied homes (in $1000s). Use the default parameters and report the 10-fold cross-validation MSE.

```
test_MSE = c()
cv_result = data.frame("Test MSE" = NA)
for (i in 1:10) {
  start_idx = i * 50 +1
  if (i != 10){
    end_idx = i*2
  }
  else{
    end_idx = nrow(Boston)
  }

  test = Boston[start_idx : end_idx, ]
  train = Boston[-c(start_idx:end_idx),]
  x_train = train[,1:13]
  y_train = train[,"medv"]
  x_test = test[,1:13]
  y_test = test[,"medv"]

  rf_model = randomForest(x = x_train, y = y_train)
  prediction = predict(rf_model, x_test)
  MSE = mean((prediction - y_test)^2)
  #test_MSE = append(test_MSE, MSE)
  cv_result[nrow(cv_result) + 1,] = MSE
}

cv_result = drop_na(cv_result)
cv_result$fold = c(1:10)
cv_result
```

| Test.MSE <dbl> | fold <int> |
|---|---|
| 10.359 | 1 |
| 7.957 | 2 |
| 6.507 | 3 |
| 10.404 | 4 |
| 13.481 | 5 |
| 30.271 | 6 |
| 31.254 | 7 |
| 51.179 | 8 |
| 52.028 | 9 |
| 28.617 | 10 |

1-10 of 10 rows

```
ggplot(cv_result, aes(x = fold, y = Test.MSE))+
  geom_line(aes(color = "red")) +
  labs(xlab = "Fold", ylab = "Test MSE", title ="Test MSE of 10-fold CV")
```



c. Now we will vary the tuning parameters of `mtry` and `ntree` to see what effect they have on performance.

- Use a range of reasonable `mtry` and `ntree` values.
- Use 5 times repeated out-of-bag (OOB) to assess performance. That is, run random forest 5 times for each tuning set, calculate the OOB MSE each time and use the average for the MSE associated with the tuning parameters.
- Use a plot to show the average MSE as a function of `mtry` and `ntree`.
- Report the best tuning parameter combination.
- Note: random forest is a stochastic model; it will be different every time it runs. Set the random seed to control the uncertainty associated with the stochasticity.
- Hint: If you use the `randomForest` package, the `mse` element in the output is a vector of OOB MSE values for `1:ntree` trees in the forest. This means that you can set `ntree` to some maximum value and get the MSE for any number of trees up to `ntree`.

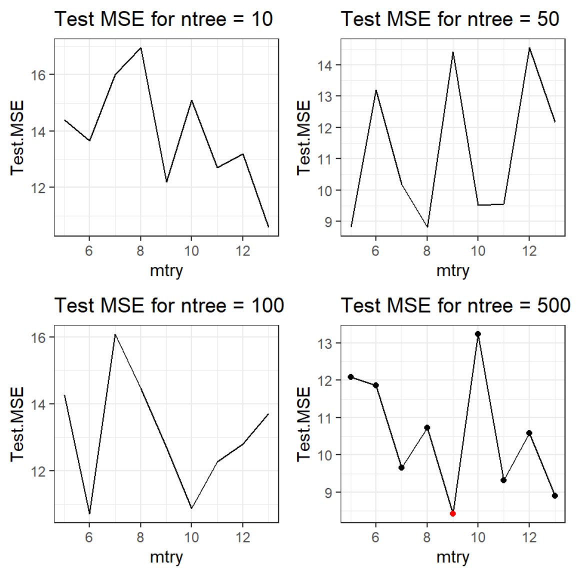
```
mtry_vals = seq(5, 13, by = 1)
ntree_vals = c(10, 50, 100, 500)

set.seed(123)
test_MSE = c()
ntee_param = c()
mtry_param = c()
result = data.frame("ntree"= NA, "mtry" = NA, "Test MSE" = NA)

for (ntree in ntree_vals){
  for (mtry in mtry_vals) {
    oob_MSE = c()
    for (i in 1:5) {
      ind = sample(nrow(Boston), size = 100, replace = FALSE)
      train = Boston[-ind,]
      test = Boston[ind,]
      x_train = train[, 1:13]
      y_train = train[,"medv"]
      x_test = test[,1:13]
      y_test = test[,"medv"]

      rf_model = randomForest(x = x_train, y = y_train, ntree = ntree, mtry = mt
ry)
      prediction = predict(rf_model, x_test)
      MSE = mean((prediction - y_test)^2)
      oob_MSE = append(oob_MSE, MSE)
    }
  result[nrow(result) + 1,] <- c(ntree,mtry,mean(oob_MSE))
  }
}
```

```
result = drop_na(result)
result = result %>% arrange(Test.MSE)
result
```

| ntree | mtry | Test.MSE |
|---:|---:|---:|
| <dbl> | <dbl> | <dbl> |
| 500 | 9 | 8.415 |
| 50 | 8 | 8.801 |
| 50 | 5 | 8.810 |
| 500 | 13 | 8.905 |
| 500 | 11 | 9.312 |
| 50 | 10 | 9.511 |
| 50 | 11 | 9.546 |
| 500 | 7 | 9.652 |
| 50 | 7 | 10.175 |
| 10 | 13 | 10.583 |

1-10 of 36 rows Previous **1** 2 3 4 Next

```
result$dot.color = c("red",rep("black", 35))

plot1 = ggplot(result %>% filter(ntree == 10), aes(x = mtry, y = Test.MSE), col
 = "blue") +
  geom_line()+labs(title = "Test MSE for ntree = 10") + scale_color_identity()

plot2 = ggplot(result %>% filter(ntree == 50), aes(x = mtry, y = Test.MSE), col
 = "blue") +
  geom_line()+ labs(title = "Test MSE for ntree = 50") + scale_color_identity()

plot3 = ggplot(result %>% filter(ntree == 100), aes(x = mtry, y = Test.MSE), col
= "blue") +
  geom_line()+ labs(title = "Test MSE for ntree = 100") + scale_color_identity()

plot4 = ggplot(result %>% filter(ntree == 500), aes(x = mtry, y = Test.MSE), col
= "blue") +
  geom_line() + geom_point(aes(colour = dot.color)) + labs(title = "Test MSE for
ntree = 500")+ scale_color_identity()

grid.arrange(plot1, plot2, plot3, plot4, ncol=2, nrow = 2)
```



∴ The optimal tuning parameter combination (ntree = 500, mtry = 9) gives the lowest Test MSE (8.415).