HW 6

Runze Wang

You can run the following code to prepare the analysis.

```
library(r02pro)     #INSTALL IF NECESSARY
library(tidyverse)  #INSTALL IF NECESSARY
library(MASS)
my_ahp <- ahp %>% dplyr::select(gar_car, liv_area, lot_area, oa_qual, sale_price) %>%
  na.omit() %>%
  mutate(type = factor(ifelse(sale_price > median(sale_price), "Expensive", "Cheap")))
tr_ind <- 1:(nrow(my_ahp)/20)
my_ahp_train <- my_ahp[tr_ind, ]
my_ahp_test <- my_ahp[-tr_ind, ]
```

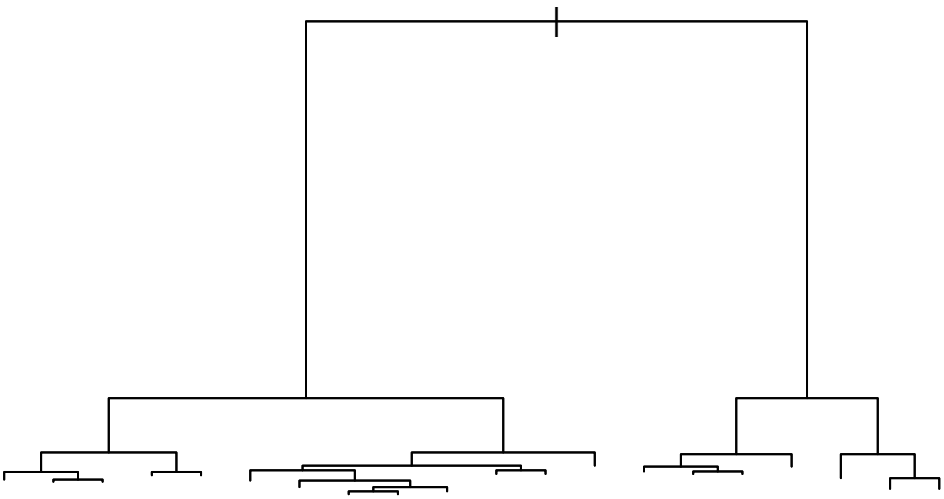
Suppose we want to build a tree to predict `sale_price` and `type`. Please answer the following questions.

1. First, we fit a deep regression tree to predict `sale_price` using the training data `my_ahp_train`. Note that, here we use `tree.control` to generate such a deep tree.

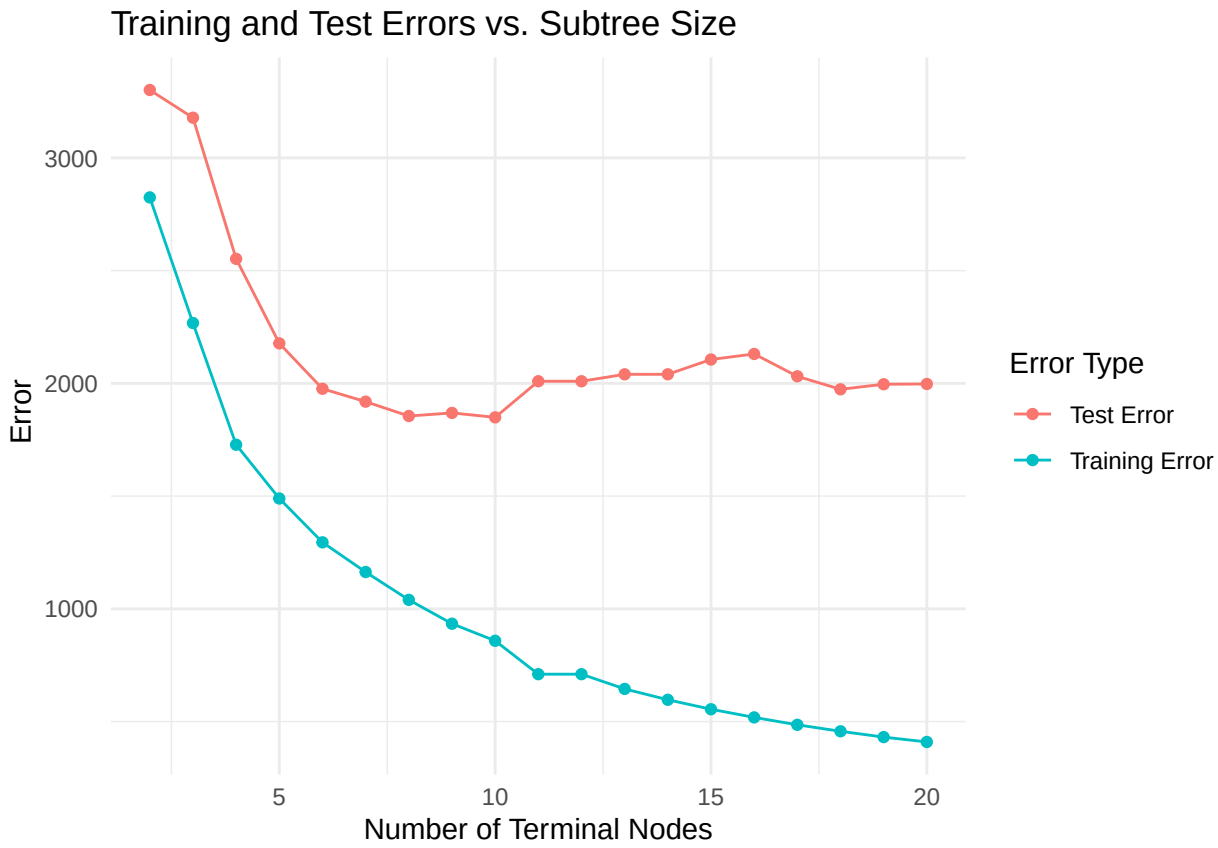
```
library(tree)
my_control <- tree.control(nrow(my_ahp_train), minsize = 2, mindev = 0)
fit <- tree(sale_price ~ gar_car + liv_area + oa_qual,
            control = my_control,
            data = my_ahp_train)
```

a Prune the tree to different subtrees with the number of terminals nodes ranging from 2 to 20 with increment 1. For each subtree, compute the training error and prediction error on the test data `my_ahp_test`. Visualize the relationship of the two errors vs. the subtree size.

```
library(tree)
training_errors <- numeric(length = 19)
test_errors <- numeric(length = 19)
for (i in 2:20) {
  pruned_tree <- prune.tree(fit, best = i)
  train_preds <- predict(pruned_tree, newdata = my_ahp_train)
  test_preds <- predict(pruned_tree, newdata = my_ahp_test)

  training_errors[i-1] <- mean((train_preds - my_ahp_train$sale_price)^2)
  test_errors[i-1] <- mean((test_preds - my_ahp_test$sale_price)^2)
}
plot(pruned_tree)
```

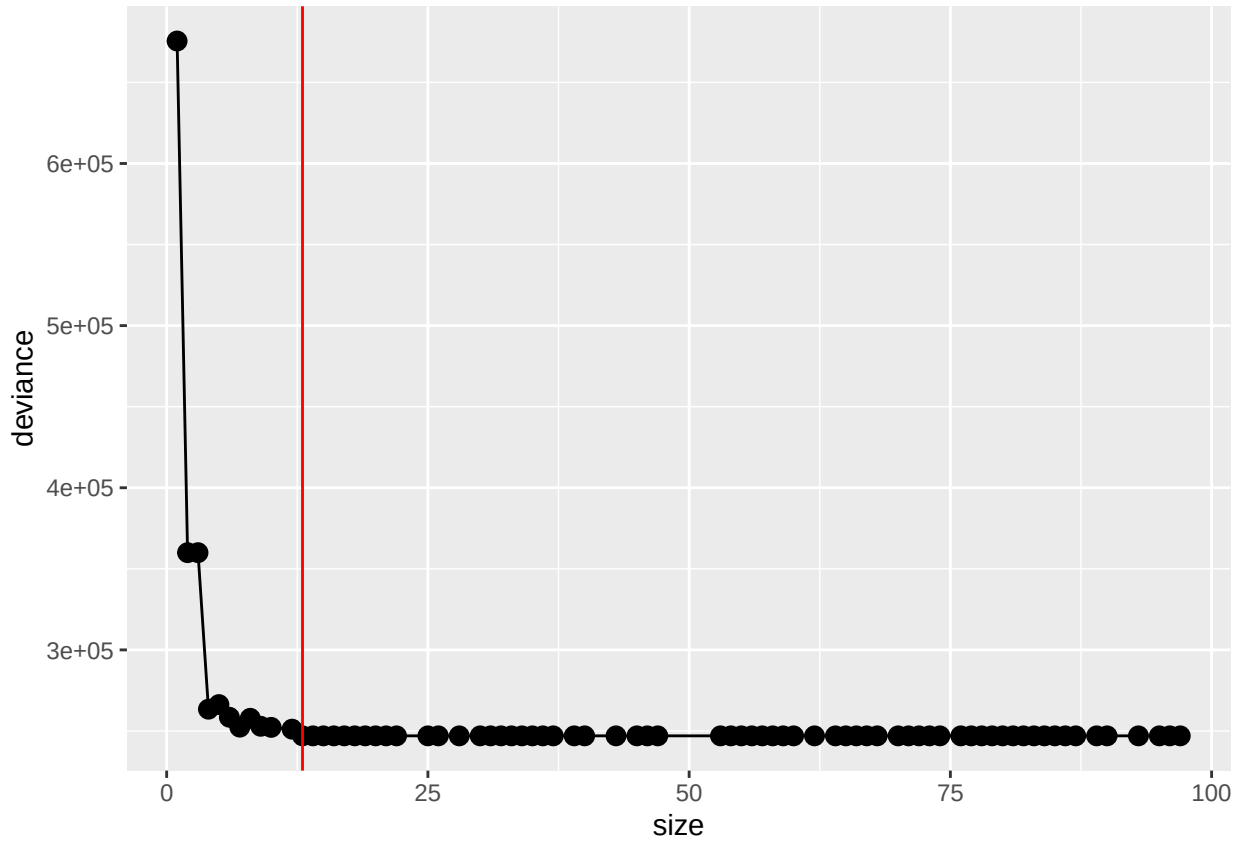


```r
errors_df <- data.frame(
  TerminalNodes = rep(2:20, times = 2),
  Error = c(training_errors, test_errors),
  Type = rep(c("Training Error", "Test Error"), each = 19)
)

ggplot(errors_df, aes(x = TerminalNodes, y = Error, color = Type)) +
  geom_line() +
  geom_point() +
  theme_minimal() +
  labs(title = "Training and Test Errors vs. Subtree Size",
       x = "Number of Terminal Nodes", y = " Error",
       color = "Error Type")
```

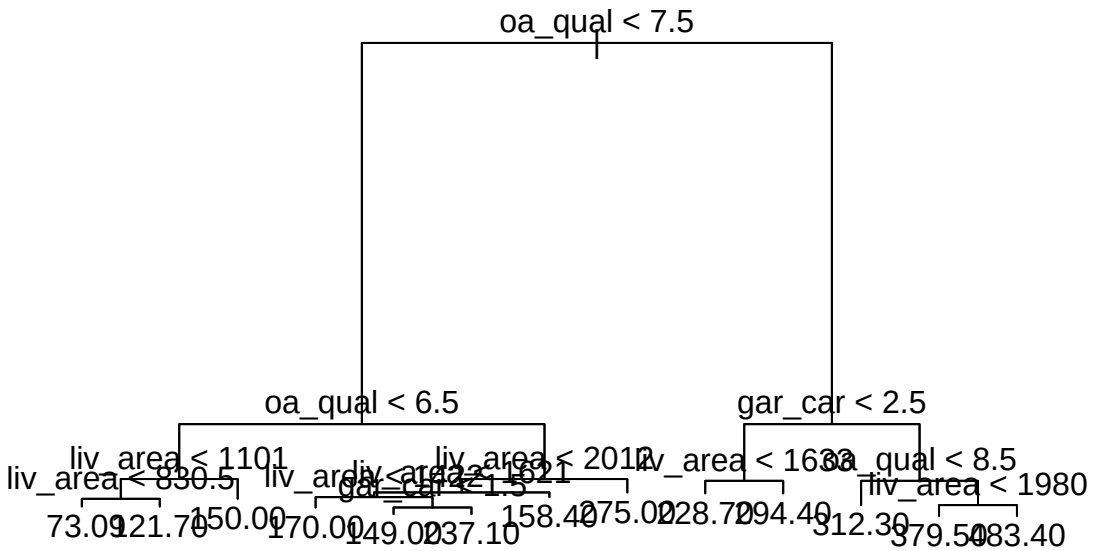


b Generate a pruned tree with the number of terminal nodes determined by cross-validation. Then visualize the tree and provide interpretations. Note that if many trees have the same cross-validation error, you want to choose the tree with the fewest terminal nodes.

```
set.seed(0)
cv.fit <- cv.tree(fit)
cv.fit_df <- data.frame(size = cv.fit$size, deviance = cv.fit$dev)
min_deviance <- min(cv.fit$dev)
sizes_with_min_deviance <- cv.fit$size[cv.fit$dev == min_deviance]
best_size <- min(sizes_with_min_deviance)

ggplot(cv.fit_df, mapping = aes(x = size, y = deviance)) +
  geom_point(size = 3) +
  geom_line() +
  geom_vline(xintercept = best_size, col = "red")
```




```
fit.tree.final <- prune.tree(fit, best = best_size)
plot(fit.tree.final)
text(fit.tree.final)
```



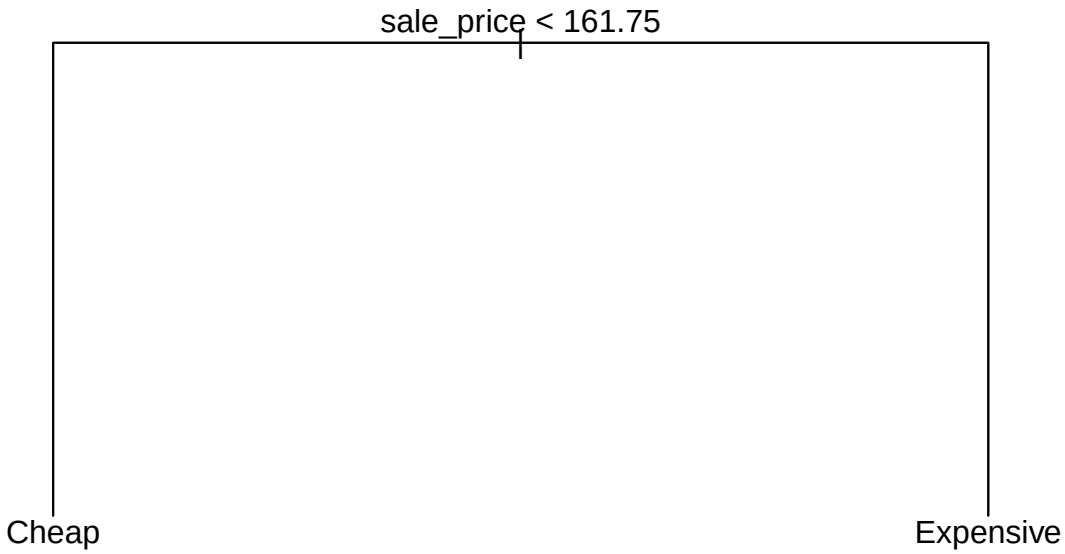

```
best_size
```

```
## [1] 13
```

When the terminal nodes are equal to 13, we can get the minimum deviance. The deviance drops dramatically and then becomes stable after size 13.

2. Build a classification tree with the number of terminal nodes determined via cross-validation to predict `expensive` using the training data `my_ahp_train`. Then compute the training and test classification error.

```
library(tree)
fit_tree <- tree(type ~ ., data = my_ahp_train)

set.seed(0)
cv_fit <- cv.tree(fit_tree)
best_size <- cv_fit$size[which.min(cv_fit$dev)]

pruned_tree <- prune.tree(fit_tree, best = best_size)

plot(pruned_tree)
text(pruned_tree, pretty = 0)
```




```
train_pred <- predict(pruned_tree, newdata = my_ahp_train, type = "class")
train_error <- mean(train_pred != my_ahp_train$type)

test_pred <- predict(pruned_tree, newdata = my_ahp_test, type = "class")
test_error <- mean(test_pred != my_ahp_test$type)

train_error
```

```
## [1] 0
```

```
test_error
```

```
## [1] 0.004123711
```

3. Question 8.4.3 on Page 361 in ISLRv2.

3. Consider the Gini index, classification error, and entropy in a simple classification setting with two classes. Create a single plot that displays each of these quantities as a function of pm1. The x-axis should display pm1, ranging from 0 to 1, and the y-axis should display the value of the Gini index, classification error, and entropy. Hint In a setting with two classes, pm1 = 1- pm2. You could make this plot by hand, but it will be much easier to make in R.

As we know, the Gini index is

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

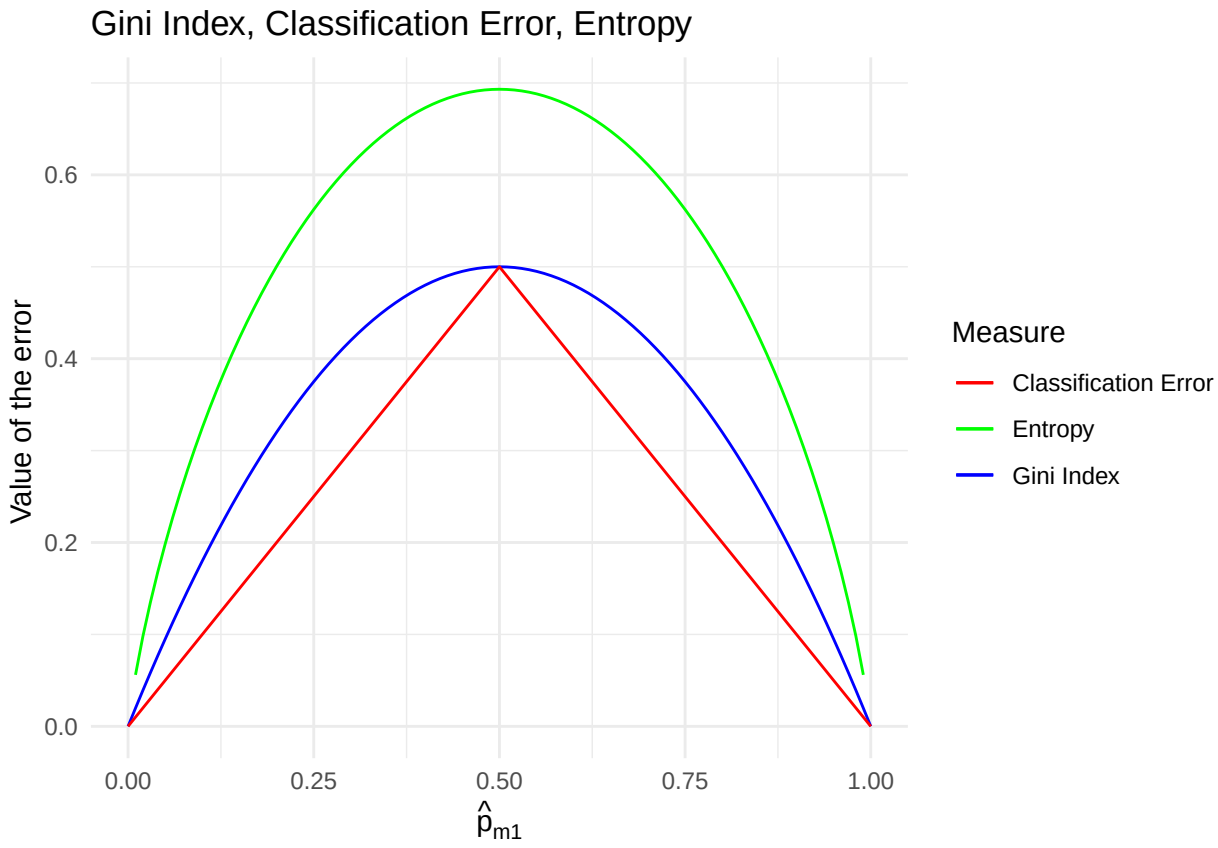
, the classification error is $E = 1 - \max(\hat{p}_{mk})$. the entropy is

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log(\hat{p}_{mk})$$

. so we can write a r code.

```
p = seq(0, 1, 0.01)
gini = p * (1 - p) * 2
classification_error = 1 - pmax(p, 1 - p)
entropy = -(p * log(p) + (1 - p) * log(1 - p))

data = data.frame(p, gini, classification_error, entropy)

ggplot(data, aes(x = p)) +
  geom_line(aes(y = gini, color = "Gini Index")) +
  geom_line(aes(y = classification_error, color = "Classification Error")) +
  geom_line(aes(y = entropy, color = "Entropy")) +
  labs(x = expression(hat(p)[m1]),
       y = "Value of the error ",
       color = "Measure",
       title = "Gini Index, Classification Error, Entropy") +
  theme_minimal() +
  scale_color_manual(values = c("Gini Index" = "blue", "Classification Error" = "red", "Entropy" = "gre
```

```
## Warning: Removed 2 rows containing missing values (`geom_line()`).
```


Gini Index, Classification Error, Entropy
Value of the error
0.0
0.2
0.4
0.6
0.00
0.25
0.50
0.75
1.00
$\hat{p}_{m1}$
Measure
Classification Error
Entropy
Gini Index