Biostat 212a Homework 4

Due Mar. 4, 2025 @ 11:59PM

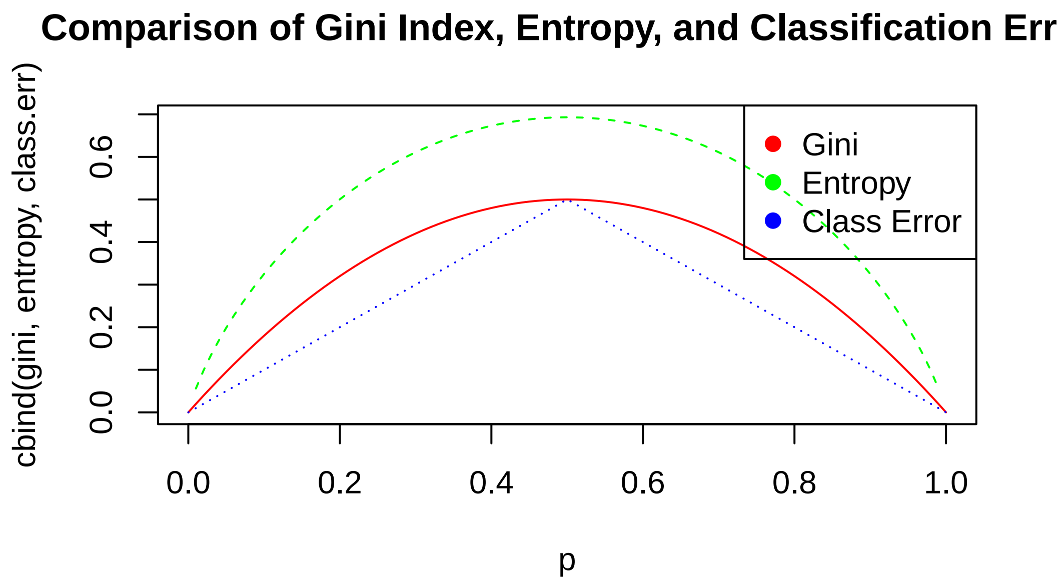
Jiaye Tian UID: 306541095

2025-03-04

Table of contents

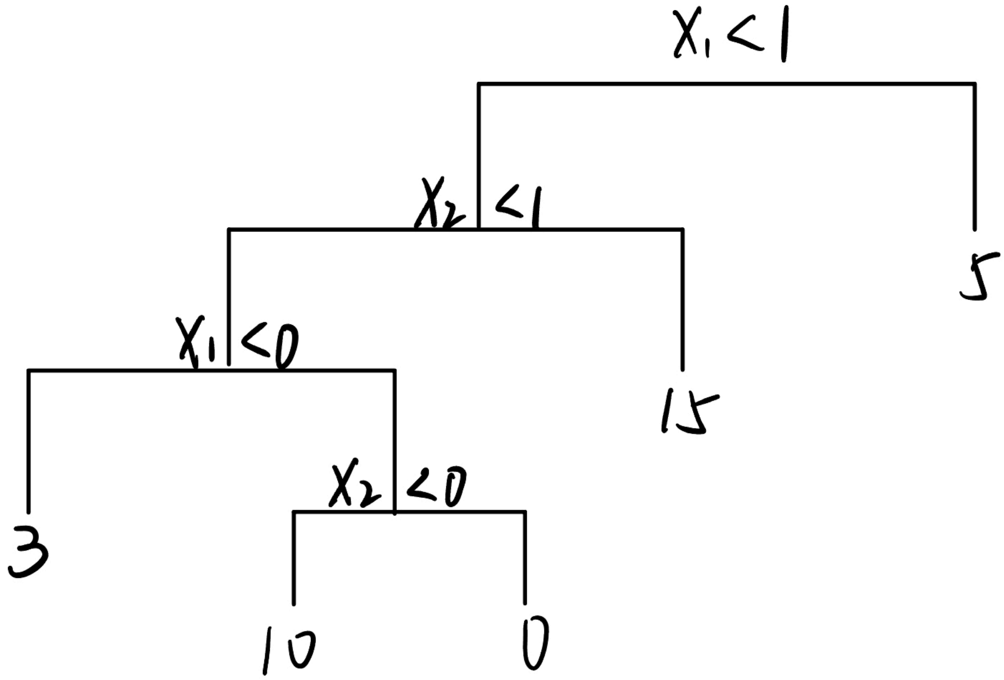
0.1 ISL Exercise 8.4.3 (10pts)

```r
p = seq(0, 1, 0.01)
gini = p * (1 - p) * 2
entropy = -(p * log(p) + (1 - p) * log(1 - p))
class.err = 1 - pmax(p, 1 - p)

matplot(p, cbind(gini, entropy, class.err), type = "l",
        col = c("red", "green", "blue"),
        main = "Comparison of Gini Index, Entropy, and Classification Error")

legend("topright",legend=c("Gini","Entropy", "Class Error"), pch=19,
       col=c("red", "green", "blue"))
```

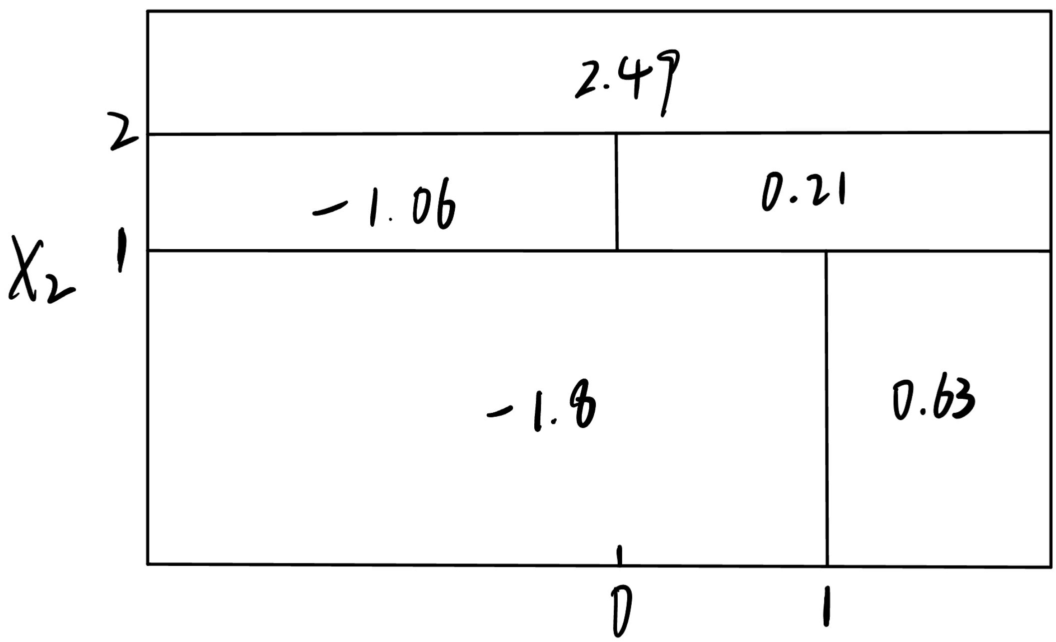


0.2 ISL Exercise 8.4.4 (10pts)

(a)



(b)



0.3 ISL Exercise 8.4.5 (10pts)

Majority Vote → Red Averaging Probability → Green

If $P(Red|X) > 0.5 \Rightarrow$ Classifier predicts Red.

If $P(Red|X) < 0.5 \Rightarrow$ Classifier predicts Green

$P(Red|X) \leq 0.5$: $0.1, 0.15, 0.2, 0.2 \Rightarrow$ 4 classifiers predict Green.

$P(Red|X) > 0.5$: $0.55, 0.6, 0.6, 0.65, 0.7, 0.75 \Rightarrow$ 6 classifiers predict Red.

Thus, the majority (6 out of 10) Vote $\Rightarrow$ Red. $\blacksquare$

$$P_{avg}(Red|X) = \frac{0.1+0.15+0.2+0.2+0.55+0.6+0.6+0.65+0.7+0.75}{10}$$

$$= 0.45$$

$P_{avg}(Green|X) = 1 - 0.45 = 0.55 \Rightarrow$ Green.

0.4 ISL Lab 8.3. `Boston` data set (30pts)

Follow the machine learning workflow to train regression tree, random forest, and boosting methods for predicting `medv`. Evaluate out-of-sample performance on a test set.

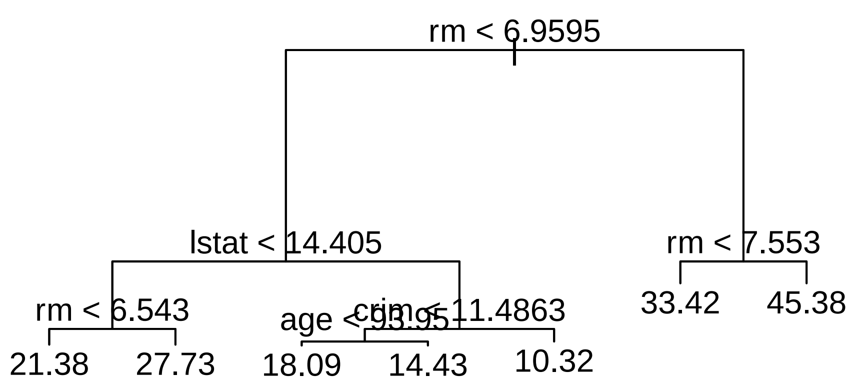
```r
# Load necessary libraries
library(MASS)          # Boston dataset
library(tree)          # Regression Tree
library(randomForest) # Random Forest
library(gbm)         # Boosting (Gradient Boosting)
# library(ISLR2)
# attach(Carseats)

# Set seed for reproducibility
set.seed(1)

# Split the dataset into training (50%) and testing (50%)
train <- sample(1:nrow(Boston), nrow(Boston) / 2)
Boston.test <- Boston[-train, "medv"]  # Extract true test set values
```

1. Regression tree

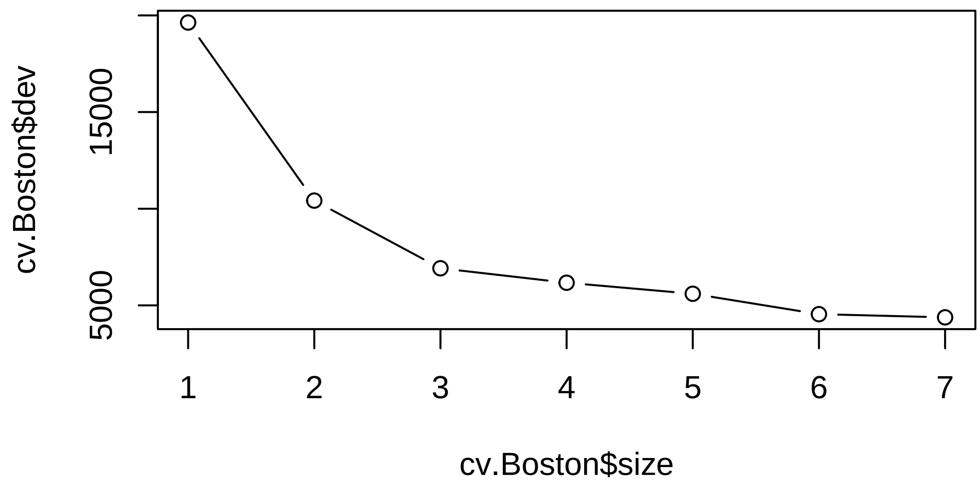
```
# Train Regression Tree
tree.Boston <- tree(medv ~ ., Boston, subset = train)
summary(tree.Boston)
```

```
Regression tree:
tree(formula = medv ~ ., data = Boston, subset = train)
Variables actually used in tree construction:
[1] "rm"    "lstat" "crim"  "age"
Number of terminal nodes:  7
Residual mean deviance:  10.38 = 2555 / 246
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-10.1800  -1.7770  -0.1775   0.0000   1.9230  16.5800
```

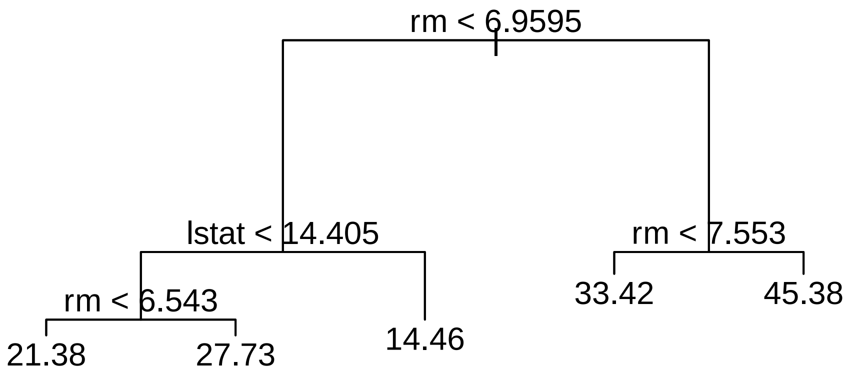
```
# Plot the tree
plot(tree.Boston)
text(tree.Boston)
```



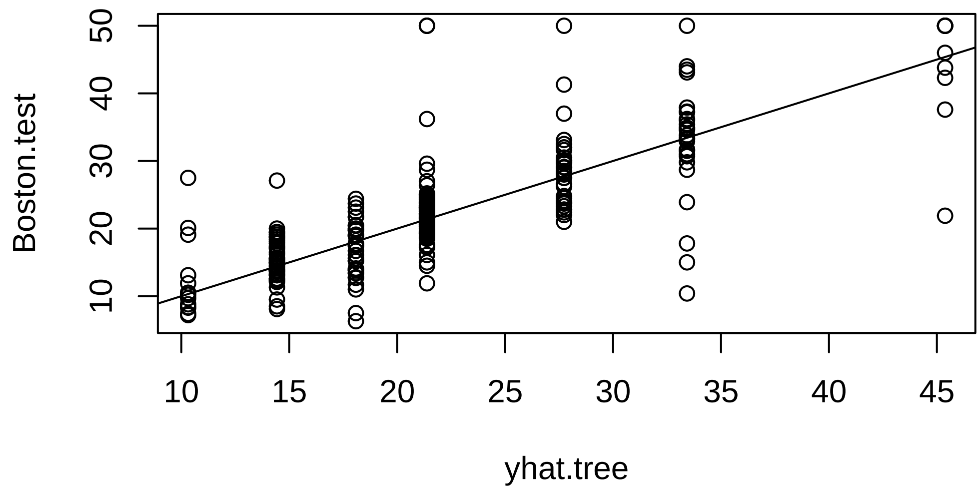
```r
# Cross-validation for pruning
cv.Boston <- cv.tree(tree.Boston)
plot(cv.Boston$size, cv.Boston$dev, type = "b")
```




```r
# Prune the tree with best = 5 (per original code)
prune.Boston <- prune.tree(tree.Boston, best = 5)
plot(prune.Boston)
text(prune.Boston, pretty = 5)
```




```
# Predict on test set and compute MSE
yhat.tree <- predict(tree.Boston, newdata = Boston[-train, ])
Boston.test <- Boston[-train, "medv"]
plot(yhat.tree, Boston.test)
abline(0, 1)
```




```
# Compute the test set MSE
tree.mse <- mean((yhat.tree - Boston.test)^2)
tree.mse
```

```
[1] 35.28688
```

In other words, the test set MSE associated with the regression tree is 35.29. The square root of the MSE is therefore around 5.941, indicating that this model leads to test predictions that are (on average) within approximately $5,941 of the true median home value for the census tract.

2. RandomForest

```
# Train Bagging model (mtry = 12, per original code)
set.seed(1)
bag.Boston <- randomForest(medv ~ ., data = Boston,
                           subset = train, mtry = 12,
                           importance = TRUE)

# Test set prediction & MSE
yhat.bag <- predict(bag.Boston, newdata = Boston[-train, ])
bag.mse <- mean((yhat.bag - Boston.test)^2)
bag.mse
```

```
[1] 23.38773
```

```r
# Train Bagging with ntree = 25 (per original code)
bag.Boston <- randomForest(medv ~ ., data = Boston,
                           subset = train, mtry = 12, ntree = 25)
yhat.bag <- predict(bag.Boston, newdata = Boston[-train, ])
bag.mse2 <- mean((yhat.bag - Boston.test)^2)
bag.mse2
```

```
[1] 25.19144
```

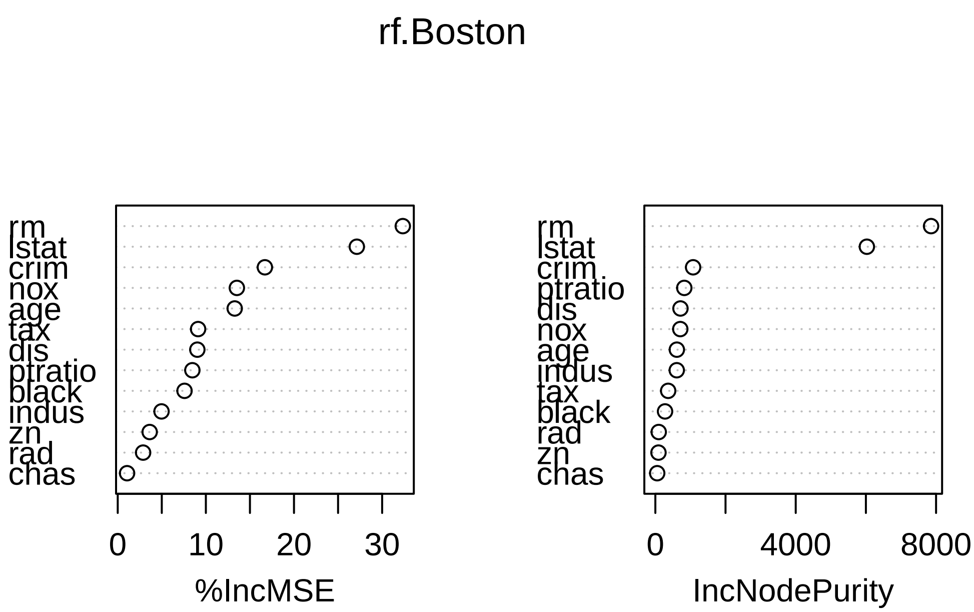
```r
# Train Random Forest (mtry = 6, per original code)
set.seed(1)
rf.Boston <- randomForest(medv ~ ., data = Boston,
                          subset = train, mtry = 6, importance = TRUE)

# Test set prediction & MSE
yhat.rf <- predict(rf.Boston, newdata = Boston[-train, ])
rf.mse <- mean((yhat.rf - Boston.test)^2)
rf.mse
```

```
[1] 19.62021
```

```r
# Feature importance & visualization
importance(rf.Boston)
```

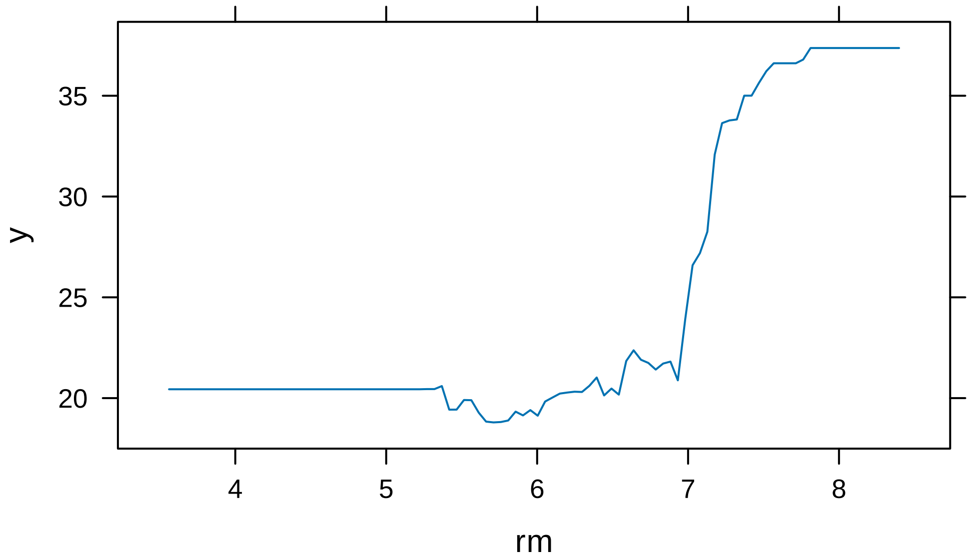
```
          %IncMSE IncNodePurity
crim    16.697017    1076.08786
zn       3.625784      88.35342
indus    4.968621     609.53356
chas     1.061432      52.21793
nox     13.518179     709.87339
rm      32.343305    7857.65451
age     13.272498     612.21424
dis      9.032477     714.94674
rad      2.878434      95.80598
tax      9.118801     364.92479
ptratio  8.467062     823.93341
black    7.579482     275.62272
lstat   27.129817    6027.63740
```

```r
varImpPlot(rf.Boston)
```

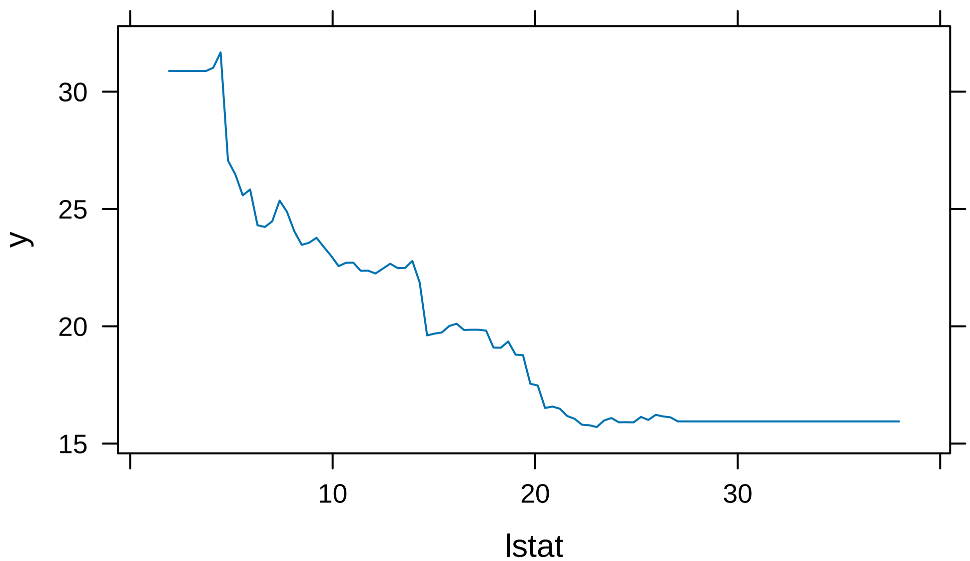


3. Boosting

```r
# Train Boosting model (default shrinkage = 0.1)
set.seed(1)
boost.Boston <- gbm(medv ~ ., data = Boston[train, ],
distribution = "gaussian", n.trees = 5000, interaction.depth = 4)

# Visualize variable importance
plot(boost.Boston, i = "rm")
```




```
plot(boost.Boston, i = "lstat")
```



```r
# Test set prediction & MSE
yhat.boost <- predict(boost.Boston, newdata = Boston[-train, ], n.trees = 5000)
boost.mse <- mean((yhat.boost - Boston.test)^2)
boost.mse
```

```
[1] 18.84709
```

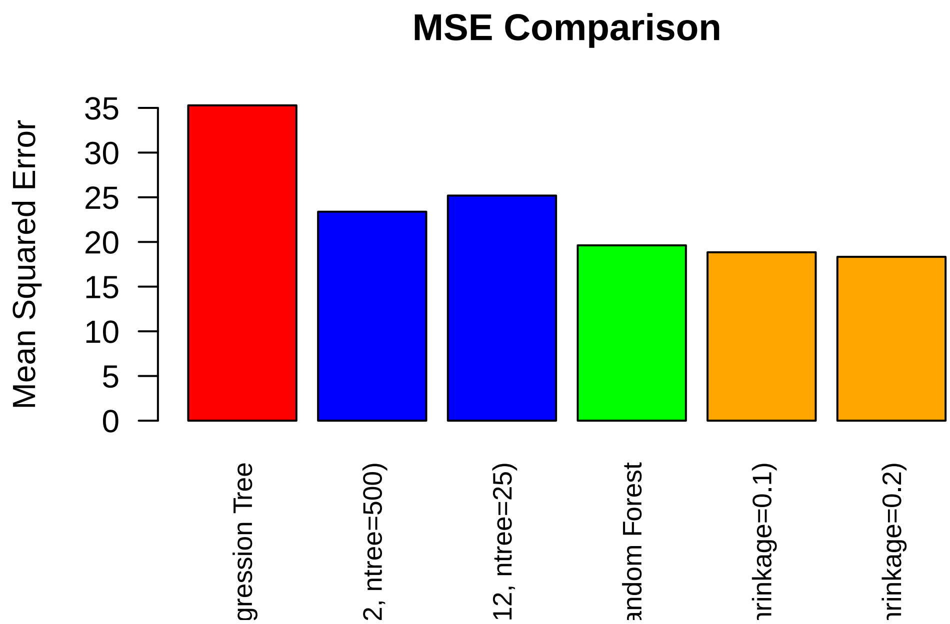
```r
# Train Boosting with shrinkage = 0.2 (per original code)
boost.Boston <- gbm(medv ~ ., data = Boston[train, ],
                    distribution = "gaussian", n.trees = 5000,
                    interaction.depth = 4, shrinkage = 0.2, verbose = F)

# Test set prediction & MSE
yhat.boost <- predict(boost.Boston, newdata = Boston[-train, ], n.trees = 5000)
boost.mse2 <- mean((yhat.boost - Boston.test)^2)
boost.mse2
```

```
[1] 18.33455
```

```r
# Compare MSE results
mse_results <- data.frame(
  Model = c("Regression Tree", "Bagging (mtry=12, ntree=500)",
            "Bagging (mtry=12, ntree=25)", "Random Forest",
            "Boosting (shrinkage=0.1)", "Boosting (shrinkage=0.2)"),
  MSE = c(tree.mse, bag.mse, bag.mse2, rf.mse, boost.mse, boost.mse2)
)
print(mse_results)
```

```
                         Model      MSE
1              Regression Tree 35.28688
2 Bagging (mtry=12, ntree=500) 23.38773
3  Bagging (mtry=12, ntree=25) 25.19144
4                Random Forest 19.62021
5     Boosting (shrinkage=0.1) 18.84709
6     Boosting (shrinkage=0.2) 18.33455
```

```r
# Plot MSE comparison
barplot(mse_results$MSE, names.arg = mse_results$Model,
        col = c("red", "blue", "blue", "green", "orange", "orange"),
        main = "MSE Comparison", ylab = "Mean Squared Error", las = 2, cex.names = 0.8)
```



0.5 ISL Lab 8.3 `Carseats` data set (30pts)

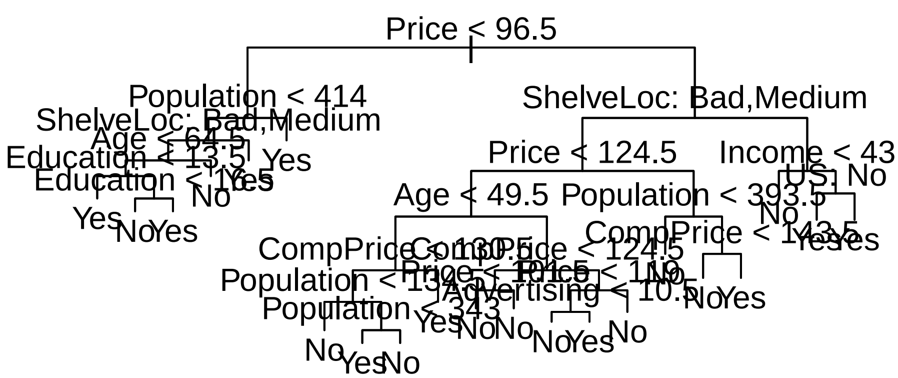
Follow the machine learning workflow to train classification tree, random forest, and boosting methods for classifying `Sales <= 8` versus `Sales > 8`. Evaluate out-of-sample performance on a test set.

```
# Load required libraries
library(tree)          # Classification Tree
library(randomForest)  # Random Forest
library(gbm)           # Gradient Boosting
library(caret)         # Confusion Matrix
library(ISLR2)

# Set seed for reproducibility
set.seed(2)

# Convert Sales into a binary variable
Carseats$High <- factor(ifelse(Carseats$Sales > 8, "Yes", "No"))

# Remove the Sales column
Carseats <- subset(Carseats, select = -Sales)
```

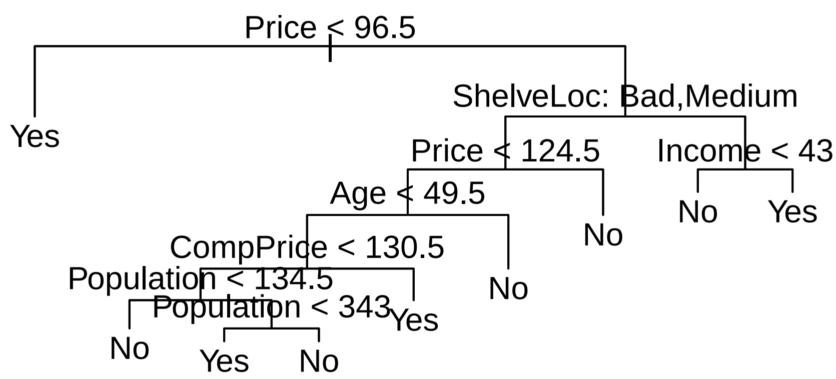
```r
# Train-test split (50-50)
train <- sample(1:nrow(Carseats), nrow(Carseats) / 2)
Carseats.train <- Carseats[train, ]
Carseats.test <- Carseats[-train, ]
High.test <- Carseats.test$High

# ---- Classification Tree ----
tree.carseats <- tree(High ~ ., data = Carseats.train)
summary(tree.carseats)
```

```
Classification tree:
tree(formula = High ~ ., data = Carseats.train)
Variables actually used in tree construction:
[1] "Price"       "Population"  "ShelveLoc"   "Age"         "Education"
[6] "CompPrice"   "Advertising" "Income"      "US"
Number of terminal nodes:  21
Residual mean deviance:  0.5543 = 99.22 / 179
Misclassification error rate: 0.115 = 23 / 200
```

```r
plot(tree.carseats)
text(tree.carseats, pretty = 0)
```



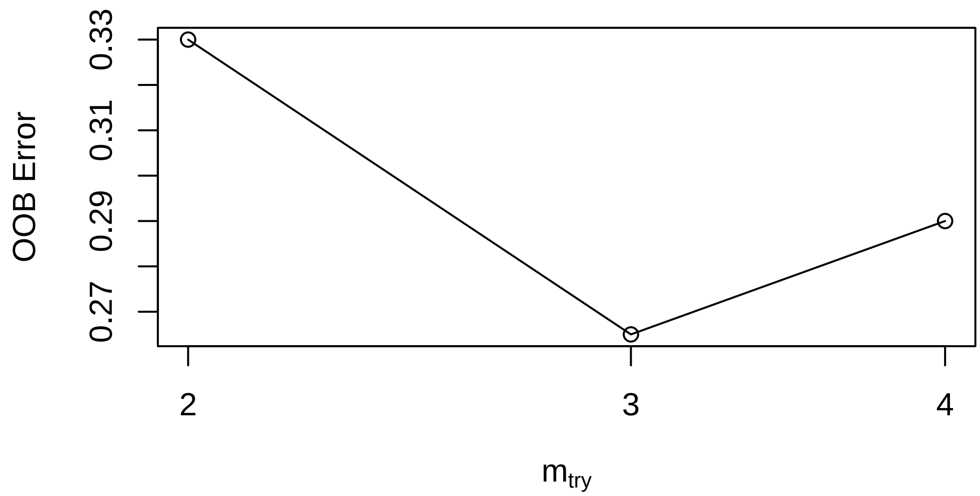
```r
# Test set prediction
tree.pred <- predict(tree.carseats, Carseats.test, type = "class")
confusion_matrix_tree <- table(tree.pred, High.test)
tree_accuracy <- sum(diag(confusion_matrix_tree)) / sum(confusion_matrix_tree)

# ---- Pruning ----
set.seed(7)
cv.carseats <- cv.tree(tree.carseats, FUN = prune.misclass)
prune_size <- cv.carseats$size[which.min(cv.carseats$dev)]  # Select pruning size with minimu
prune.carseats <- prune.misclass(tree.carseats, best = prune_size)

plot(prune.carseats)
text(prune.carseats, pretty = 0)
```




```r
# Evaluate pruned tree
tree.pred.pruned <- predict(prune.carseats, Carseats.test, type = "class")
confusion_matrix_pruned <- table(tree.pred.pruned, High.test)
pruned_accuracy <- sum(diag(confusion_matrix_pruned)) / sum(confusion_matrix_pruned)

# ---- Random Forest ----
p <- ncol(Carseats.train) - 1  # Number of features excluding 'High'
```

```
# Tune mtry using tuneRF()
set.seed(1)
best_mtry <- tuneRF(Carseats.train[-ncol(Carseats.train)], Carseats.train$High,
                     stepFactor = 1.5, improve = 0.01, trace = FALSE)
```

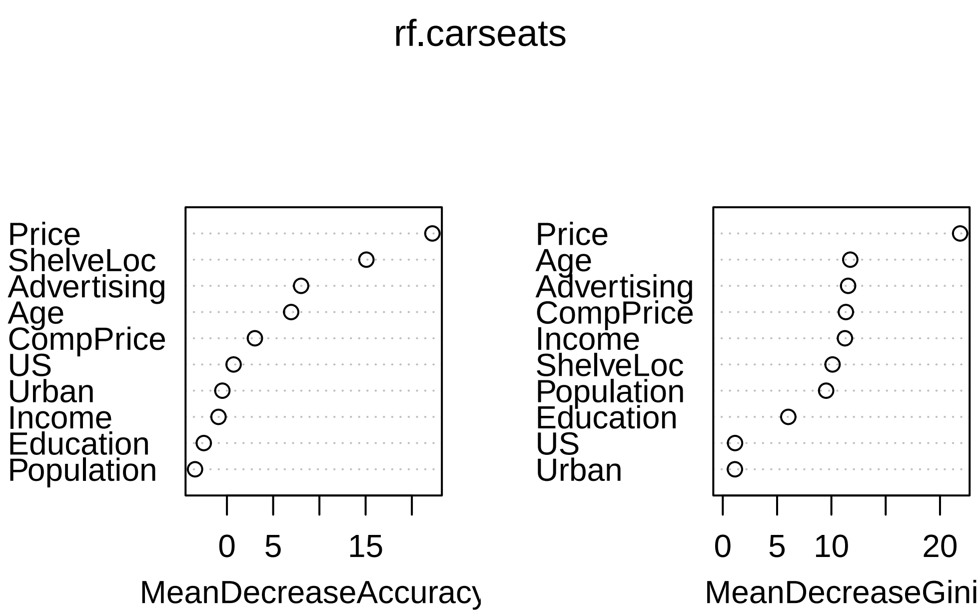
```
-0.245283 0.01
-0.09433962 0.01
```




```
mtry_best <- best_mtry[which.min(best_mtry[,2]), 1]

# Train Random Forest
set.seed(1)
rf.carseats <- randomForest(High ~ ., data = Carseats.train, ntree = 500, mtry = mtry_best,

# Test set prediction
rf.pred <- predict(rf.carseats, Carseats.test)
confusion_matrix_rf <- table(rf.pred, High.test)
rf_accuracy <- sum(diag(confusion_matrix_rf)) / sum(confusion_matrix_rf)

# Variable Importance
importance(rf.carseats)
```

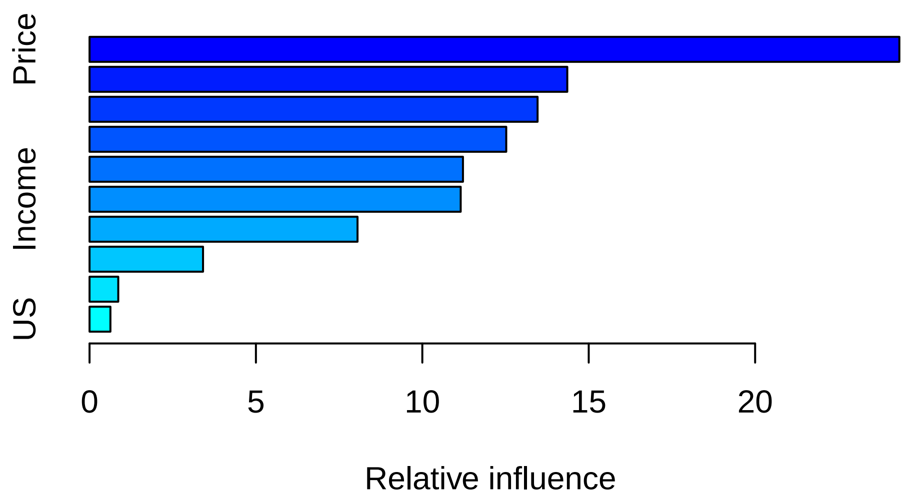
```
                     No         Yes MeanDecreaseAccuracy MeanDecreaseGini
CompPrice     2.1304913  2.21583500            3.0233624        11.332341
Income       -1.2187382  0.08332858           -0.9127061        11.247612
Advertising   2.7107740  9.28221142            8.0200032        11.539777
Population   -2.5279538 -1.96748887           -3.4496266         9.515984
Price        17.6102844 16.18029714           22.2205678        21.853751
ShelveLoc    11.8799961 11.50873822           15.0797611        10.106813
Age           3.7038805  6.83255965            6.9440472        11.738663
Education    -2.9218854 -0.16331173           -2.5039109         6.031744
Urban        -0.9337001  0.16405241           -0.4965739         1.114652
US           -0.4330561  1.48555372            0.7195542         1.127466
```

```r
varImpPlot(rf.carseats)
```




```r
# ---- Boosting ----
boost.train <- Carseats.train
boost.test <- Carseats.test
boost.train$High <- ifelse(boost.train$High == "Yes", 1, 0)  # Convert to 0/1 for Boosting

set.seed(1)
boost.carseats <- gbm(High ~ ., data = boost.train, distribution = "bernoulli",
                      n.trees = 5000, interaction.depth = 4, shrinkage = 0.01, verbose = FALS
```

```r
# Prediction
boost.prob <- predict(boost.carseats, newdata = boost.test, n.trees = 5000, type = "response"
boost.pred <- factor(ifelse(boost.prob > 0.5, "Yes", "No"), levels = levels(High.test))

confusion_matrix_boost <- table(boost.pred, High.test)
boost_accuracy <- sum(diag(confusion_matrix_boost)) / sum(confusion_matrix_boost)

# Variable Importance
summary(boost.carseats)
```




```
                    var    rel.inf
Price             Price 24.3344414
CompPrice     CompPrice 14.3555860
ShelveLoc     ShelveLoc 13.4625168
Age                 Age 12.5211889
Advertising Advertising 11.2205000
Income           Income 11.1522623
Population   Population  8.0522931
Education     Education  3.4101404
Urban             Urban  0.8634069
US                   US  0.6276643
```

```r
# Test Boosting with different shrinkage (0.2)
set.seed(1)
boost.carseats2 <- gbm(High ~ ., data = boost.train, distribution = "bernoulli",
                        n.trees = 5000, interaction.depth = 4, shrinkage = 0.2, verbose = FAL

boost.prob2 <- predict(boost.carseats2, newdata = boost.test, n.trees = 5000, type = "respons
boost.pred2 <- factor(ifelse(boost.prob2 > 0.5, "Yes", "No"), levels = levels(High.test))

confusion_matrix_boost2 <- table(boost.pred2, High.test)
boost_accuracy2 <- sum(diag(confusion_matrix_boost2)) / sum(confusion_matrix_boost2)

# ---- Confusion Matrix Visualization ----
cat("Classification Tree:\n")
```

```
Classification Tree:
```

```r
print(confusionMatrix(as.factor(tree.pred), High.test))
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  104  33
       Yes  13  50

               Accuracy : 0.77
                 95% CI : (0.7054, 0.8264)
    No Information Rate : 0.585
    P-Value [Acc > NIR] : 2.938e-08

                  Kappa : 0.5091

 Mcnemar's Test P-Value : 0.005088

            Sensitivity : 0.8889
            Specificity : 0.6024
         Pos Pred Value : 0.7591
         Neg Pred Value : 0.7937
             Prevalence : 0.5850
         Detection Rate : 0.5200
   Detection Prevalence : 0.6850
```

```
      Balanced Accuracy : 0.7456

       'Positive' Class : No
```

```r
cat("\nPruned Tree:\n")
```

```
Pruned Tree:
```

```r
print(confusionMatrix(as.factor(tree.pred.pruned), High.test))
```

```
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No  97  25
       Yes 20  58

               Accuracy : 0.775
                 95% CI : (0.7108, 0.8309)
    No Information Rate : 0.585
    P-Value [Acc > NIR] : 1.206e-08

                  Kappa : 0.5325

 Mcnemar's Test P-Value : 0.551

            Sensitivity : 0.8291
            Specificity : 0.6988
         Pos Pred Value : 0.7951
         Neg Pred Value : 0.7436
             Prevalence : 0.5850
         Detection Rate : 0.4850
   Detection Prevalence : 0.6100
      Balanced Accuracy : 0.7639

       'Positive' Class : No
```

```r
cat("\nRandom Forest:\n")
```

```
Random Forest:
```

```r
print(confusionMatrix(as.factor(rf.pred), High.test))
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  110  24
       Yes   7  59

               Accuracy : 0.845
                 95% CI : (0.7873, 0.8922)
    No Information Rate : 0.585
    P-Value [Acc > NIR] : 1.939e-15

                  Kappa : 0.671

 Mcnemar's Test P-Value : 0.004057

            Sensitivity : 0.9402
            Specificity : 0.7108
         Pos Pred Value : 0.8209
         Neg Pred Value : 0.8939
             Prevalence : 0.5850
         Detection Rate : 0.5500
   Detection Prevalence : 0.6700
      Balanced Accuracy : 0.8255

       'Positive' Class : No
```

```r
cat("\nBoosting (shrinkage=0.01):\n")
```

```
Boosting (shrinkage=0.01):
```

```r
print(confusionMatrix(as.factor(boost.pred), High.test))
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  109  16
       Yes   8  67

               Accuracy : 0.88
                 95% CI : (0.8267, 0.9216)
    No Information Rate : 0.585
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7493

 Mcnemar's Test P-Value : 0.153

            Sensitivity : 0.9316
            Specificity : 0.8072
         Pos Pred Value : 0.8720
         Neg Pred Value : 0.8933
             Prevalence : 0.5850
         Detection Rate : 0.5450
   Detection Prevalence : 0.6250
      Balanced Accuracy : 0.8694

       'Positive' Class : No
```

```r
cat("\nBoosting (shrinkage=0.2):\n")
```

```
Boosting (shrinkage=0.2):
```

```r
print(confusionMatrix(as.factor(boost.pred2), High.test))
```

```
Confusion Matrix and Statistics

          Reference
```

```
Prediction  No Yes
       No  111  36
       Yes   6  47

               Accuracy : 0.79
                 95% CI : (0.7269, 0.8443)
    No Information Rate : 0.585
    P-Value [Acc > NIR] : 7.046e-10

                  Kappa : 0.5435

 Mcnemar's Test P-Value : 7.648e-06

            Sensitivity : 0.9487
            Specificity : 0.5663
         Pos Pred Value : 0.7551
         Neg Pred Value : 0.8868
             Prevalence : 0.5850
         Detection Rate : 0.5550
   Detection Prevalence : 0.7350
      Balanced Accuracy : 0.7575

       'Positive' Class : No
```

```r
# ---- Accuracy Comparison ----
accuracy_results <- data.frame(
  Model = c("Classification Tree", "Pruned Tree", "Random Forest", "Boosting (0.01)", "Boost
  Accuracy = c(tree_accuracy, pruned_accuracy, rf_accuracy, boost_accuracy, boost_accuracy2)
)

print(accuracy_results)
```

```
                Model Accuracy
1 Classification Tree    0.770
2         Pruned Tree    0.775
3       Random Forest    0.845
4     Boosting (0.01)    0.880
5      Boosting (0.2)    0.790
```

```
barplot(accuracy_results$Accuracy, names.arg = accuracy_results$Model,
        col = c("red", "blue", "green", "orange", "purple"),
        main = "Accuracy Comparison", ylab = "Accuracy", las = 2, cex.names = 0.8)
```