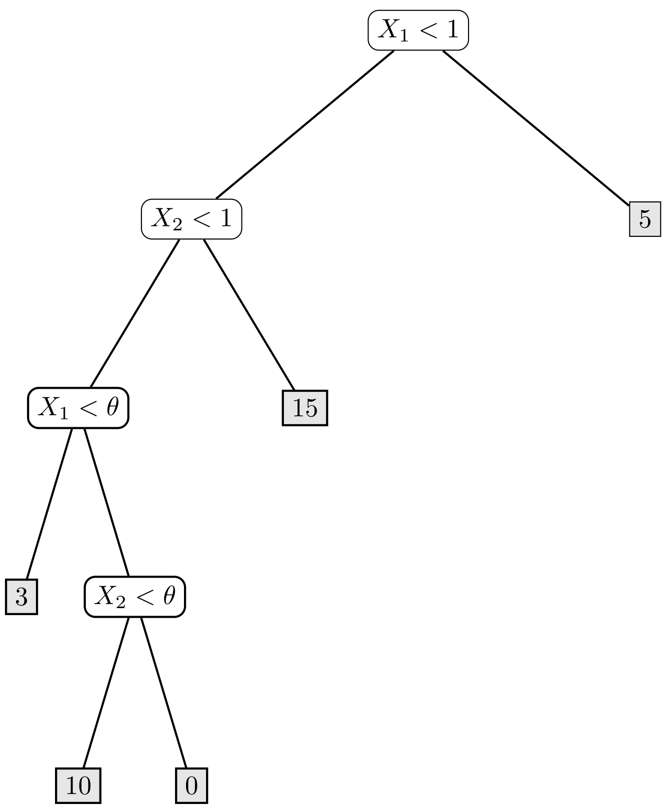
chapter 08 hw

Fabiani Rafael

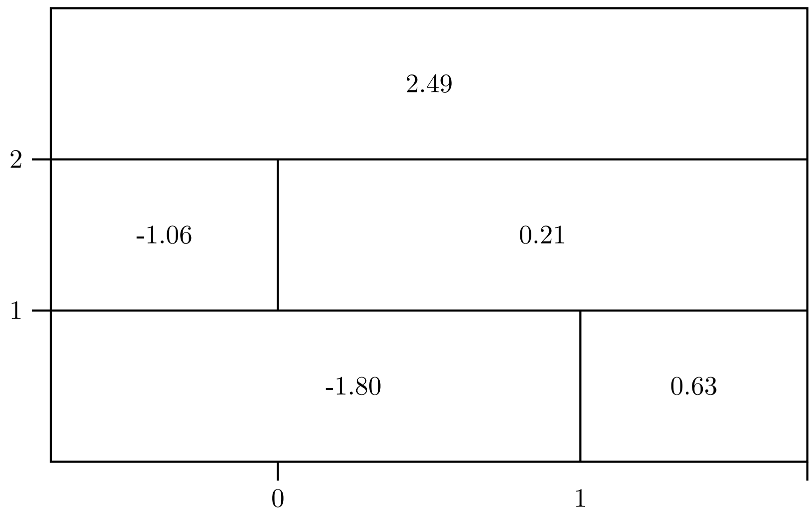
Conceptual Questions

Exercise 4: This question relates to the plots in Figure 8.14.

(a) Sketch the tree corresponding to the partition of the predictor space illustrated in the left-hand panel of Figure 8.14. The numbers inside the boxes indicate the mean of Y within each region.



(b) Create a diagram similar to the left-hand panel of Figure 8.14, using the tree illustrated in the right-hand panel of the same figure. You should divide up the predictor space into the correct regions, and indicate the mean for each region.



Applied Questions

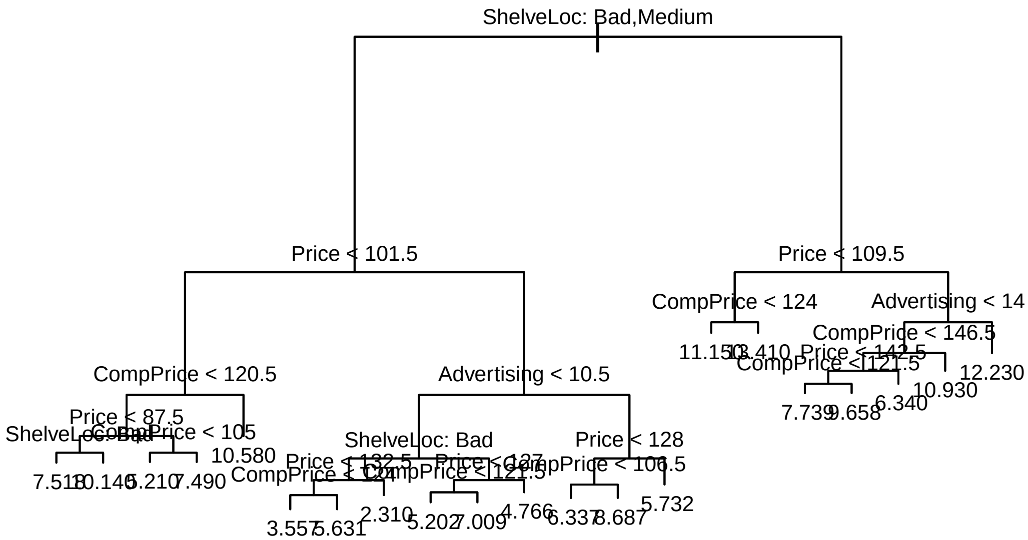
Exercise 8 a-e: In the lab, a classification tree was applied to the Carseats data set after converting Sales into a qualitative response variable. Now we will seek to predict Sales using regression trees and related approaches, treating the response as a quantitative variable.

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

```
set.seed(448)
data("Carseats")

#  train/test split
train_indices <- sample(1:nrow(Carseats), nrow(Carseats)/2)
Carseats.train <- Carseats[train_indices, ]
Carseats.test <- Carseats[-train_indices, ]
```

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

```
full_tree <- tree(Sales ~ ., data = Carseats.train)

# plot & label  tree
plot(full_tree); text(full_tree, pretty = 0, cex = 0.7)
```



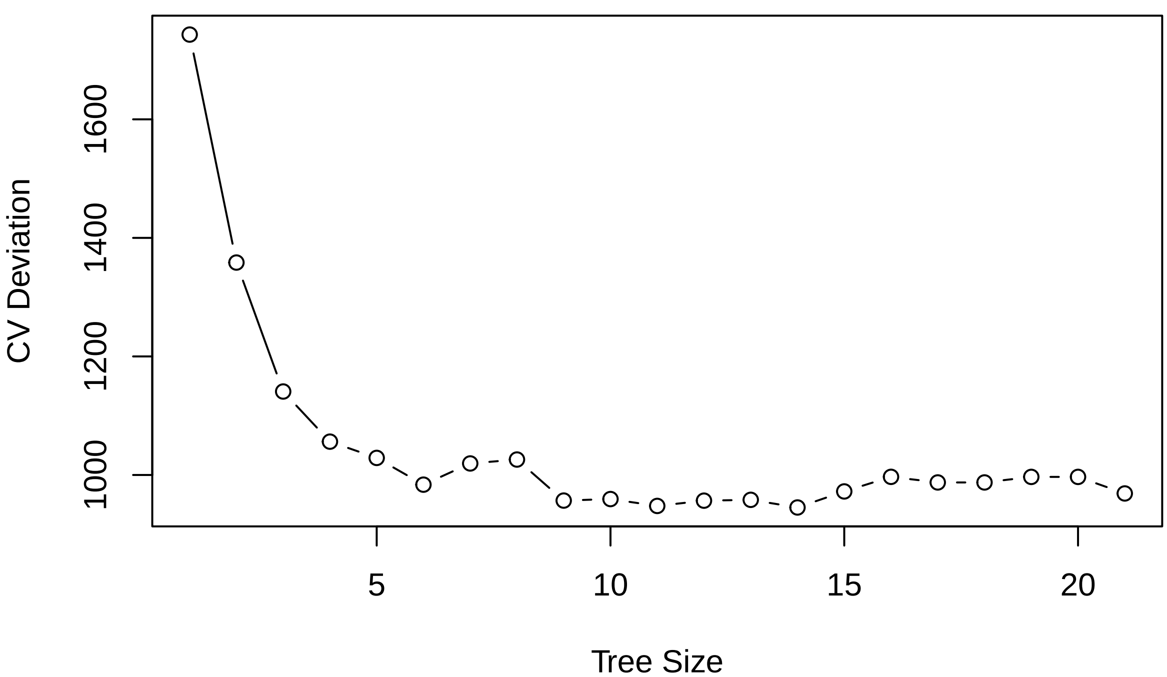

```
full_pred <- predict(full_tree, Carseats.test)
full_mse  <- mean((full_pred - Carseats.test$Sales)^2)
cat("(b)  Unpruned tree  ·  test MSE =", round(full_mse, 3), "\n\n")
```

```
## (b)  Unpruned tree  ·  test MSE = 4.593
```

- The test MSE is 4.593 or about 4.6.

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

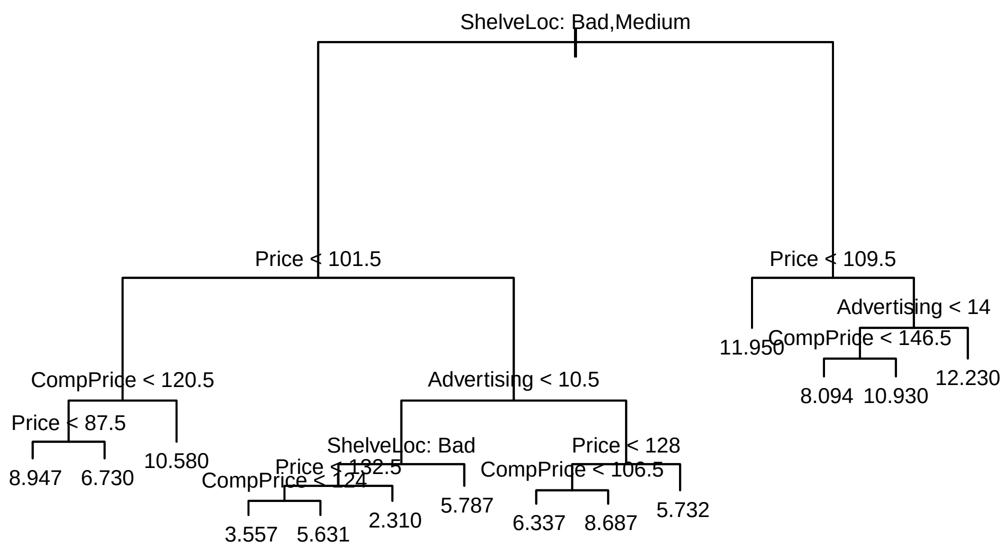
```
set.seed(448)
cv_tree <- cv.tree(full_tree, FUN = prune.tree)


# plot cv results
plot(cv_tree$size, cv_tree$dev, type = "b", xlab = "Tree Size", ylab = "CV Deviation")
```



```r
# optimal tree size
optimal_size <- cv_tree$size[which.min(cv_tree$dev)]
cat("(c)  Optimal tree size =", optimal_size, "\n")
```

```
## (c)  Optimal tree size = 14
```

```r
# prune
pruned_tree <- prune.tree(full_tree, best = optimal_size)

# plot pruned tree
plot(pruned_tree); text(pruned_tree, pretty = 0, cex = 0.7)
```




```r
# pred on the test set using the pruned tree
pruned_pred <- predict(pruned_tree, Carseats.test)
pruned_mse  <- mean((pruned_pred - Carseats.test$Sales)^2)
cat("(c)  Pruned tree  ·  test MSE =", round(pruned_mse, 3), "\n\n")
```

```
## (c)  Pruned tree  ·  test MSE = 4.928
```

- Pruning the tree seemed to increase MSE to 4.928 or about 4.9.

(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.

```r
# fit a bagging model
bagging_model <- randomForest(Sales ~ ., data = Carseats.train, mtry = ncol(Carseats.train) - 1, ntree =

# predict on the test set
bagging_pred <- predict(bagging_model, Carseats.test)

# calculate test MSE
bagging_mse <- mean((bagging_pred - Carseats.test$Sales)^2)
cat("(d)  Bagging model  ·  test MSE =", round(bagging_mse, 3), "\n")
```

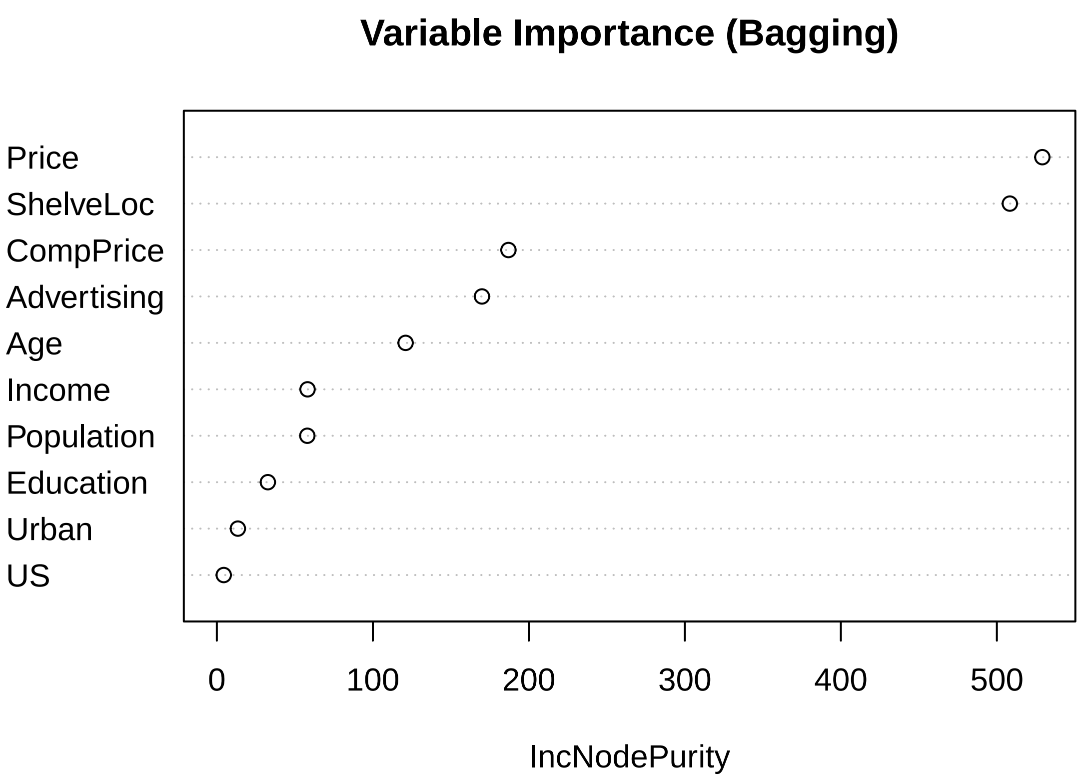
```
## (d)  Bagging model  ·  test MSE = 2.634
```

```r
# variable importance
importance(bagging_model)
```

```
##             IncNodePurity
```

```
## CompPrice      186.927580
## Income          58.152790
## Advertising    169.950258
## Population      57.997518
## Price          529.142034
## ShelveLoc      508.306432
## Age            121.001179
## Education       32.708028
## Urban           13.478081
## US               4.428693
```

```r
# plot variable importance
varImpPlot(bagging_model, main = "Variable Importance (Bagging)")
```



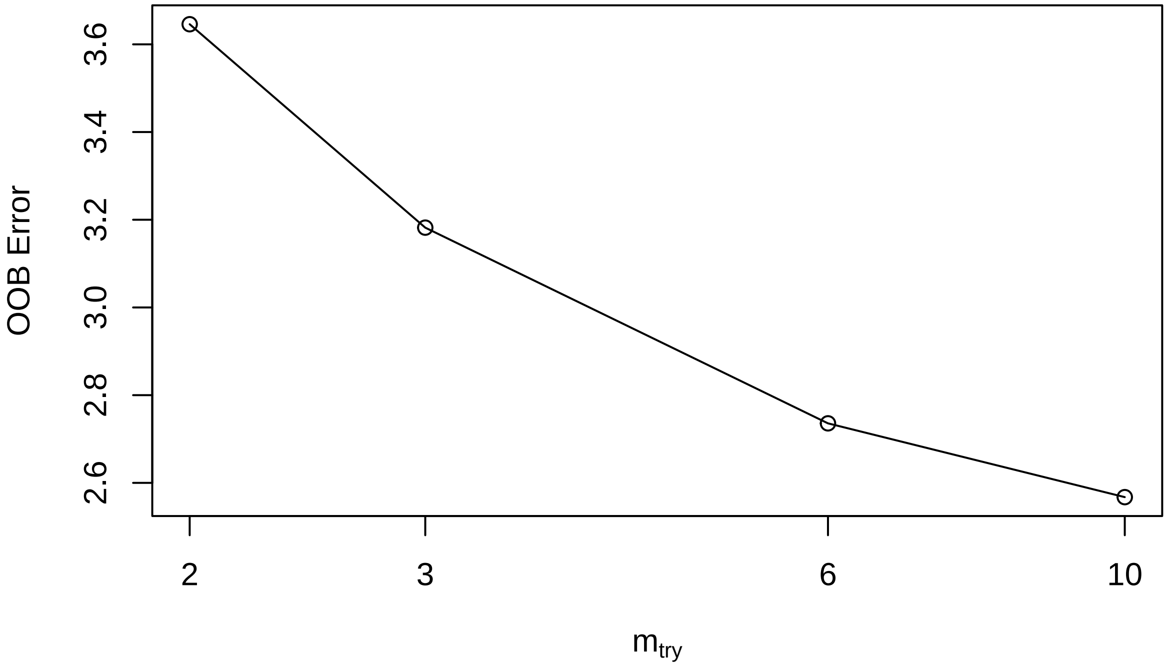
- The test MSE obtained is 2.634 or about 2.6.

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

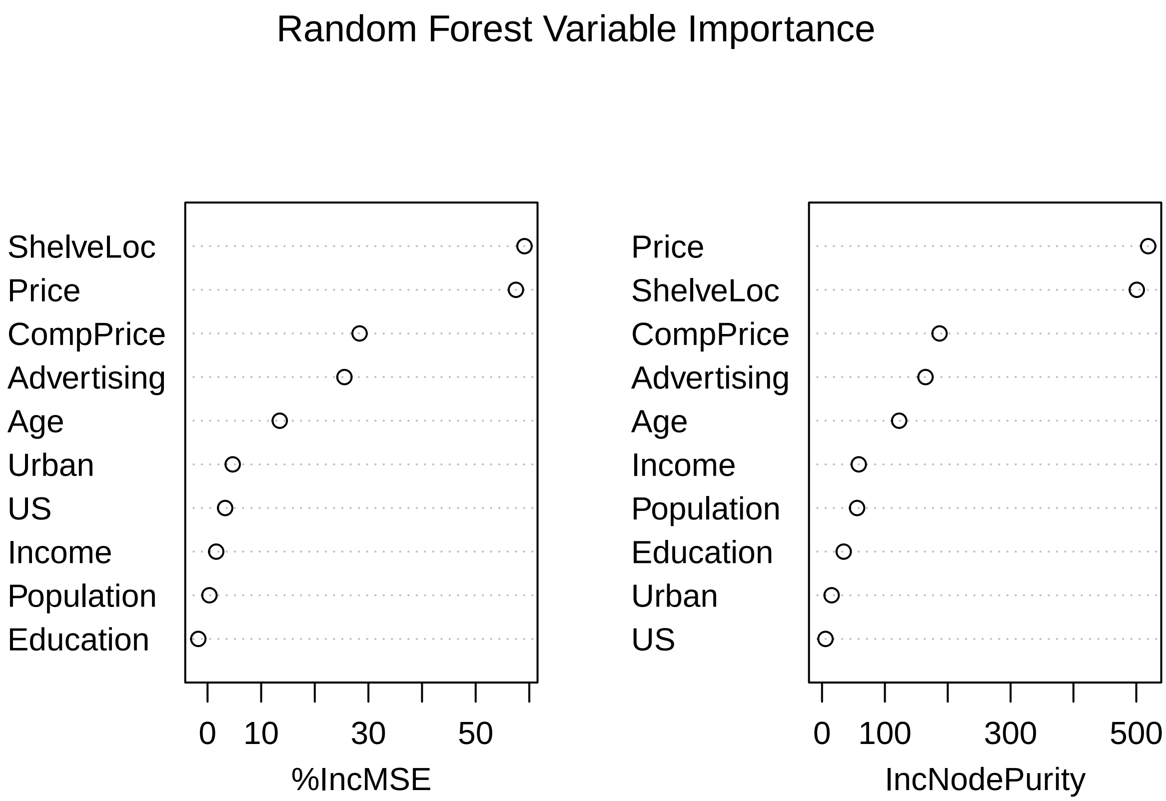
```r
set.seed(448)
predictors <- Carseats.train[, -which(names(Carseats.train) == "Sales")]
response <- Carseats.train$Sales

# optimal mtry with error handling
best_mtry <- tuneRF(
  x = predictors,
  y = response,
  ntreeTry = 500,
  improve = 0.01,
  trace = FALSE
```

```
)
```

```
## -0.145751 0.01
## 0.1402443 0.01
## 0.06150049 0.01
```




```r
# edge case no improvement
if (length(best_mtry) == 0) {
  best_mtry_value <- sqrt(ncol(predictors)) %>% round()
} else {
  best_mtry_value <- best_mtry[which.min(best_mtry[,2]), 1]
}

# fit fina model
rf_fit <- randomForest(
  Sales ~ .,
  data = Carseats.train,
  mtry = best_mtry_value,
  importance = TRUE,
  ntree = 500
)

pred_rf <- predict(rf_fit, Carseats.test)
mse_rf <- round(mean((pred_rf - Carseats.test$Sales)^2), 3)

# results
varImpPlot(rf_fit, main = "Random Forest Variable Importance")
```




```r
cat("Optimal mtry:", best_mtry_value, "\nRF Test MSE:", mse_rf)
```

```
## Optimal mtry: 10
## RF Test MSE: 2.611
```

- It seems that this increased the MSE to over 2.6, at 2.611.

Exercise 10: We now 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.

```r
data("Hitters")
# rm rows wit NA vals
Hitters <- na.omit(Hitters)

# log-transform salary
Hitters$Salary <- log(Hitters$Salary)
head(Hitters)
```

```
##                    AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun
## -Alan Ashby          315   81     7   24  38    39    14   3449   835     69
## -Alvin Davis         479  130    18   66  72    76     3   1624   457     63
## -Andre Dawson        496  141    20   65  78    37    11   5628  1575    225
## -Andres Galarraga    321   87    10   39  42    30     2    396   101     12
## -Alfredo Griffin     594  169     4   74  51    35    11   4408  1133     19
## -Al Newman           185   37     1   23   8    21     2    214    42      1
##                    CRuns CRBI CWalks League Division PutOuts Assists Errors
## -Alan Ashby          321  414    375      N        W     632      43     10
## -Alvin Davis         224  266    263      A        W     880      82     14
## -Andre Dawson        828  838    354      N        E     200      11      3
## -Andres Galarraga     48   46     33      N        E     805      40      4
```

```
## -Alfredo Griffin    501  336    194      A         W      282      421      25
## -Al Newman           30    9     24      N         E       76      127       7
##                     Salary NewLeague
## -Alan Ashby       6.163315         N
## -Alvin Davis      6.173786         A
## -Andre Dawson     6.214608         N
## -Andres Galarraga 4.516339         N
## -Alfredo Griffin  6.620073         A
## -Al Newman        4.248495         A
```

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

```
# create training and test sets
set.seed(448)
train_indices <- 1:200
Hitters.train <- Hitters[train_indices, ]
Hitters.test <- Hitters[-train_indices, ]
# check dimensions
cat("Training set dimensions:", dim(Hitters.train), "\n")
```

```
## Training set dimensions: 200 20
```

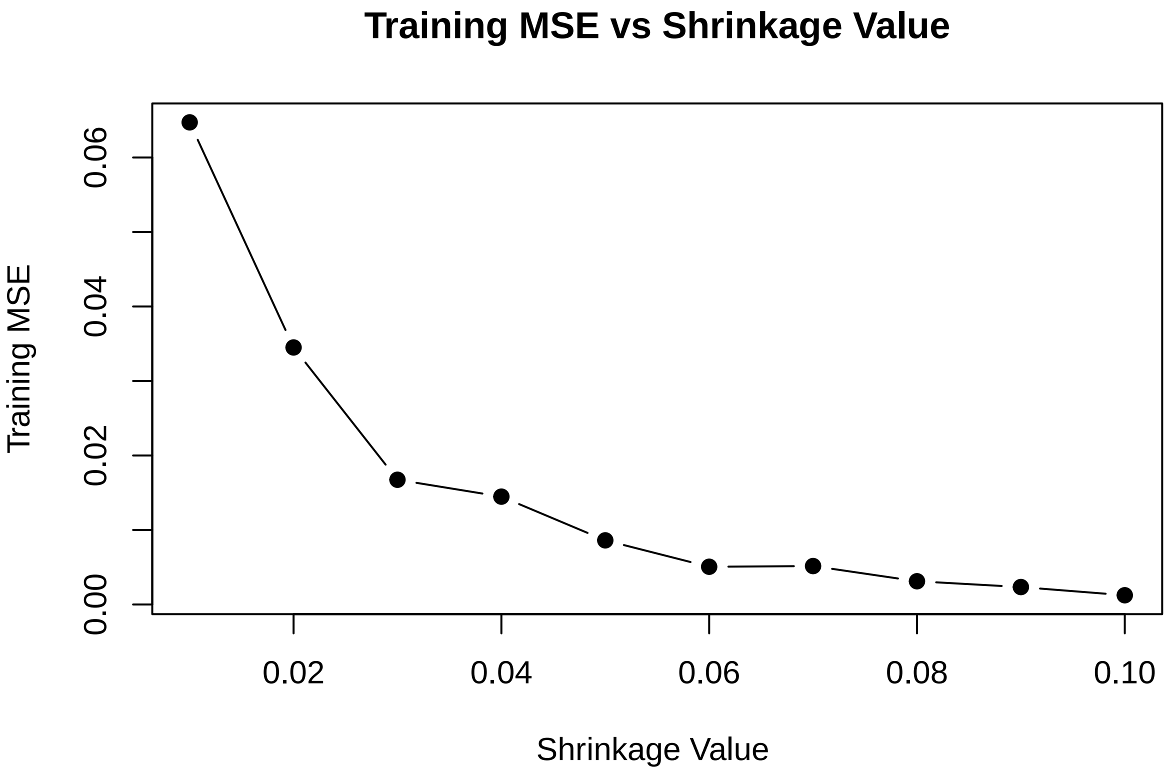
```
cat("Test set dimensions:", dim(Hitters.test), "\n")
```

```
## Test set dimensions: 63 20
```

(c) Perform boosting on the training set with 1,000 trees for a range of values of the shrinkage parameter λ. Produce a plot with different shrinkage values on the x-axis and the corresponding training set MSE on the y-axis.

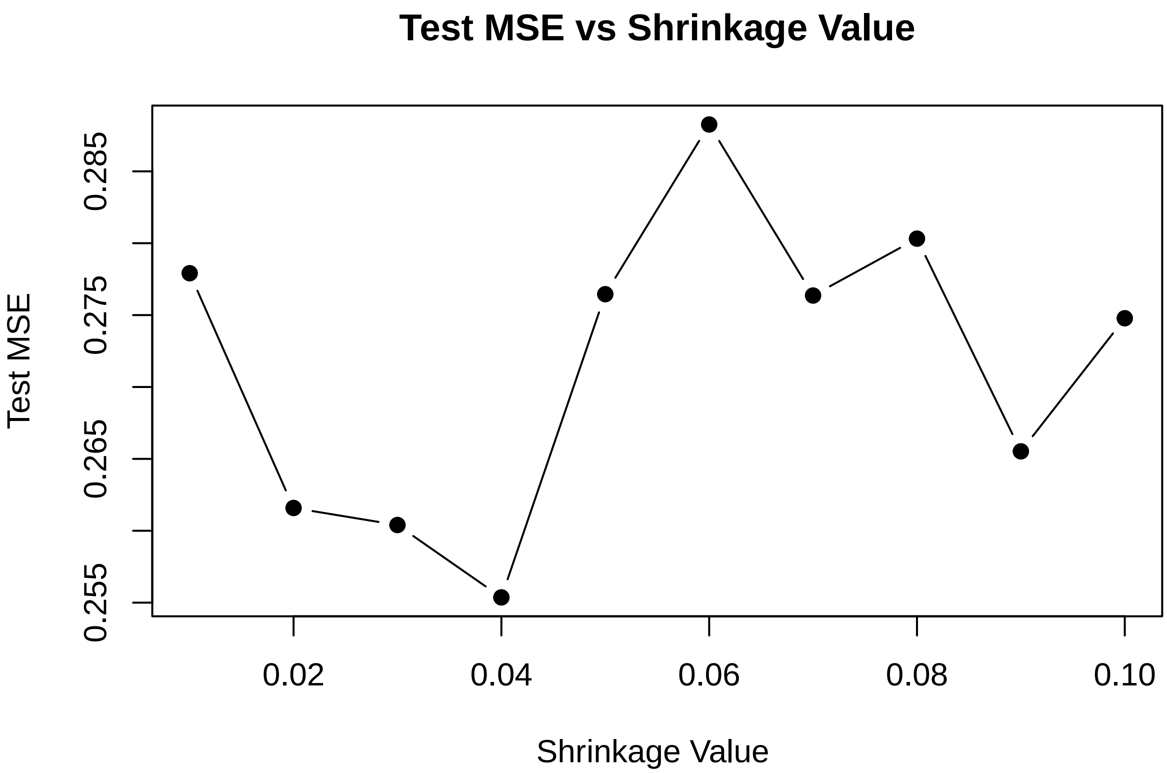
```
# range of shrinkage values
shrinkage_values <- seq(0.01, 0.1, by = 0.01)

#  init training MSE vecto
train_mse <- numeric(length(shrinkage_values))

# for each shrinkage value
for (i in seq_along(shrinkage_values)) {
  # fit boosting model
  boost_model <- gbm(
    formula = Salary ~ .,
    data = Hitters.train,
    distribution = "gaussian",
    n.trees = 1000,
    interaction.depth = 4,
    shrinkage = shrinkage_values[i],
    bag.fraction = 0.5,
    verbose = FALSE
  )
  # pred on the training set
  train_pred <- predict(boost_model, Hitters.train, n.trees = 1000)
  # get training MSE
  train_mse[i] <- mean((train_pred - Hitters.train$Salary)^2)
}
# plot training MSE v shrinkage values
plot(shrinkage_values, train_mse, type = "b", pch = 19,
```

```r
     xlab = "Shrinkage Value ", ylab = "Training MSE",
     main = "Training MSE vs Shrinkage Value")
```



(d) Produce a plot with different shrinkage values on the x-axis and the corresponding test set MSE on the y-axis.

```r
#  init training MSE vector
test_mse <- numeric(length(shrinkage_values))

for (i in seq_along(shrinkage_values)) {
  # fit boosting model
  boost_model <- gbm(
    formula = Salary ~ .,
    data = Hitters.train,
    distribution = "gaussian",
    n.trees = 1000,
    interaction.depth = 4,
    shrinkage = shrinkage_values[i],
    bag.fraction = 0.5,
    verbose = FALSE
  )
  #  test set pred
  test_pred <- predict(boost_model, Hitters.test, n.trees = 1000)
  # test MSE
  test_mse[i] <- mean((test_pred - Hitters.test$Salary)^2)
}
# plot test MSE v shrinkage vals
plot(shrinkage_values, test_mse, type = "b", pch = 19,
     xlab = "Shrinkage Value ", ylab = "Test MSE",
     main = "Test MSE vs Shrinkage Value")
```



(e) Compare the test MSE of boosting to the test MSE that results from applying two of the regression approaches seen in Chapters 3 and 6.

```r
# pick best shrinkage value from part d
best_lambda    <- shrinkage_values[which.min(test_mse)]
best_mse_boost <- min(test_mse)

cat("(e) boosting best shrinkage =", best_lambda,
    "test mse =", round(best_mse_boost, 3), "\n\n")
```

```
## (e) boosting best shrinkage = 0.04 test mse = 0.255
```

```r
# baseline: ordinary least squares
lm_fit  <- lm(Salary ~ ., data = Hitters.train)
lm_pred <- predict(lm_fit, Hitters.test)
lm_mse  <- mean((lm_pred - Hitters.test$Salary)^2)
cat("linear regression test mse =", round(lm_mse, 3), "\n")
```

```
## linear regression test mse = 0.492
```

```r
# baseline: ridge regression
x_train <- model.matrix(Salary ~ ., Hitters.train)[, -1]
y_train <- Hitters.train$Salary
x_test  <- model.matrix(Salary ~ ., Hitters.test)[, -1]
y_test  <- Hitters.test$Salary

set.seed(448)
cv_ridge  <- cv.glmnet(x_train, y_train, alpha = 0, nfolds = 10)
ridge_pred <- predict(cv_ridge, s = cv_ridge$lambda.min, newx = x_test)
ridge_mse  <- mean((ridge_pred - y_test)^2)
cat("ridge regression test mse =", round(ridge_mse, 3), "\n\n")
```

```
## ridge regression test mse = 0.457
```

```r
# side-by-side mse comparison
mse_tbl <- tibble::tibble(
  method   = c("boosting", "linear regression", "ridge regression"),
  test_mse = round(c(best_mse_boost, lm_mse, ridge_mse), 3)
)
print(mse_tbl)
```

```
## # A tibble: 3 x 2
##   method            test_mse
##   <chr>                <dbl>
## 1 boosting             0.255
## 2 linear regression    0.492
## 3 ridge regression     0.457
```

(f) Which variables appear to be the most important predictors in the boosted model?

```r
# variable importance from the final boosting model
best_boost <- gbm(
  Salary ~ ., data = Hitters.train,
  distribution      = "gaussian",
  n.trees           = 1000,
  interaction.depth = 4,
  shrinkage         = best_lambda,
  bag.fraction      = 0.5,
  verbose           = FALSE
)

imp <- summary(best_boost, plotit = FALSE)
knitr::kable(
  head(imp, 10),
  digits    = 2,
  col.names = c("variable", "relative influence (%)"),
  caption   = "top 10 important predictors - boosting"
)
```

Table 1: top 10 important predictors – boosting

	variable	relative influence (%)
CAtBat	CAtBat	22.60
CRBI	CRBI	10.37
CWalks	CWalks	9.83
CHits	CHits	7.16
CRuns	CRuns	6.28
Walks	Walks	5.90
PutOuts	PutOuts	5.67
Years	Years	4.51
CHmRun	CHmRun	4.20
AtBat	AtBat	3.97

- The top 10 most important predictors for the data set appear to be AtBat, Hits, HmRun, Runs, RBI, Walks, Years, CAtBat, CHits, and CHmRun. From these the most important predictors are AtBat, Hits, HmRun, Runs, and RBI.

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

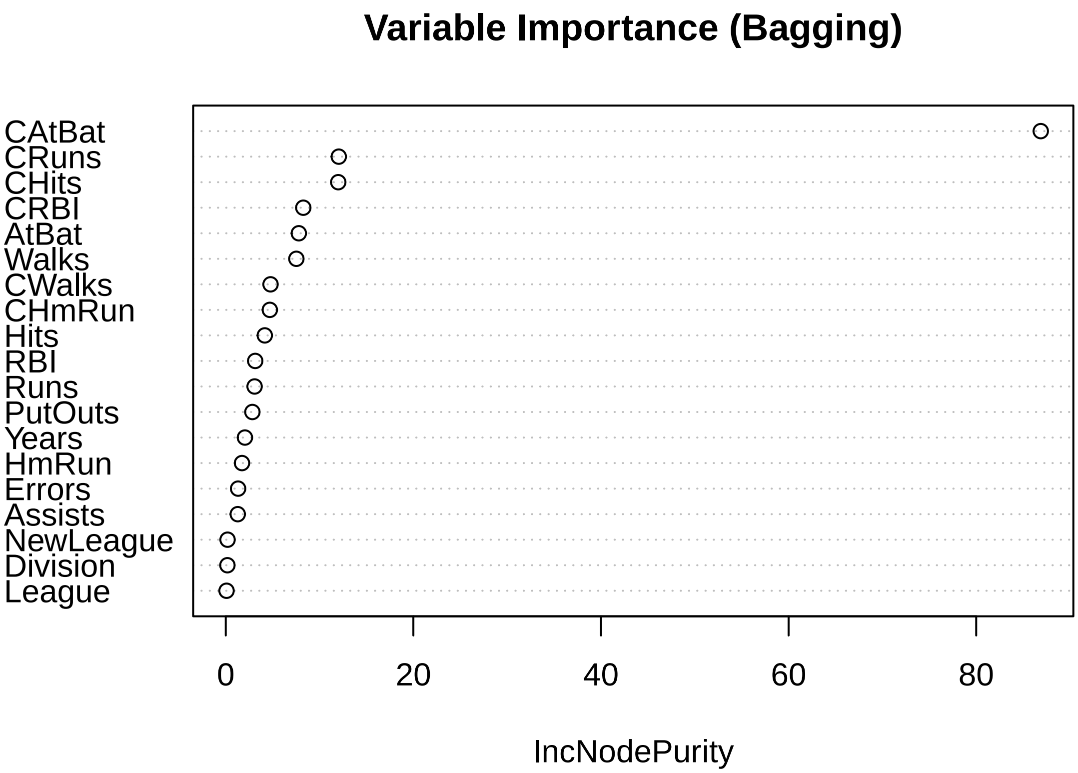
```r
# bagging model
bagging_model <- randomForest(
  Salary ~ .,
  data = Hitters.train,
  mtry = ncol(Hitters.train) - 1,
  ntree = 500
)
# pred on the test set
bagging_pred <- predict(bagging_model, Hitters.test)

# get test MSE
bagging_mse <- mean((bagging_pred - Hitters.test$Salary)^2)
cat("Bagging Test MSE:", round(bagging_mse, 3), "\n")
```

```
## Bagging Test MSE: 0.232
```

```r
# Variable importance
importance(bagging_model)
```

```
##           IncNodePurity
## AtBat         7.7889311
## Hits          4.1595227
## HmRun         1.7309295
## Runs          3.0744015
## RBI           3.1400975
## Walks         7.5256481
## Years         2.0414694
## CAtBat       86.8814940
## CHits        11.9923234
## CHmRun        4.6965302
## CRuns        12.0465909
## CRBI          8.2637426
## CWalks        4.7731320
## League        0.0836081
## Division      0.1749444
## PutOuts       2.8387601
## Assists       1.2776929
## Errors        1.3149954
## NewLeague     0.1927522
```

```r
# Plot variable importance
varImpPlot(bagging_model, main = "Variable Importance (Bagging)")
```



- The test MSE for the bagging approach is 0.232.