Assignment 9 by Team 4

Shun-Lung Chang, Muhammad Hammad Hassan, Kavish Tyagi

In this assignment we aim at predicting **Salary** in the **Hitters** data set which is available in the ISLR package in R. The assignment was carried out in R, and the source code can be found here.

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

To meet the requirements of this task, the 'Salary' variable can be processed as the code below shows.

```
dat <- Hitters[!is.na(Hitters$Salary), ]
dat$Salary <- log(dat$Salary)
```

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

The dataset has been partitioned by selecting the first 200 rows as the training dataset and the rest as the test dataset.

```
train_dat <- dat[1:200, ]
test_dat <- dat[201:nrow(dat), ]
```

3. Fit a tree to the training data, with Salary 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?

The tree model can obtained by `tree()` in the **tree** package. The results shows that the variables used in the tree construction, and the number of terminal nodes is **10**.

```
tree_mod <- tree(Salary ~ ., data = train_dat)
summary(tree_mod)
```

```
Regression tree:
tree(formula = Salary ~ ., data = train_dat)
Variables actually used in tree construction:
[1] "CAtBat"  "CRuns"   "AtBat"   "Walks"   "Hits"    "CHits"   "CRBI"
[8] "PutOuts"
Number of terminal nodes:  10
Residual mean deviance:  0.1665 = 31.64 / 190
Distribution of residuals:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-1.003000 -0.221300  0.009429  0.000000  0.246800  2.342000
```

Also, the training error in this study, is acquired by the definition of mean square error, that is,

$$\frac{\sum_i (predict_i - true_i)^2}{N}$$

The training error of this tree model is 0.1581972.

```
pred_train <- predict(tree_mod, train_dat)
mse_train <- mean((pred_train - train_dat$Salary) ^ 2)
mse_train
```

```
[1] 0.1581972
```

4. 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.

The information of the root and nodes is demonstrated below. Each node consists of split, n, deviance, and yval, which are: 1. split: splitting criterion, 2. n: number of observations in this node, 3. deviance: variance in this node, 4. yval: predicted value.

Node 5, for instance, contains observations with 'CAtBat' < 1322 (parent node) and 'CRun' > 92.5. This node includes 27 observations of the training dataset, and the variance of these observations is 2.261. All the samples being classified into node 5 will be given a predicted log salary 5.409.

```
tree_mod
```

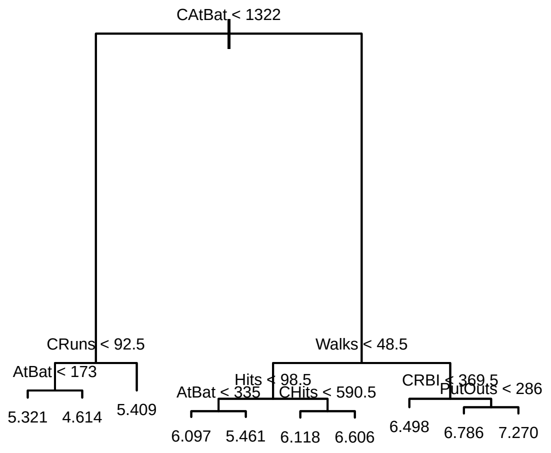
```
node), split, n, deviance, yval
      * denotes terminal node

 1) root 200 166.4000 5.940
   2) CAtBat < 1322 70  23.3000 4.971
     4) CRuns < 92.5 43  12.6000 4.696
       8) AtBat < 173 5   7.2760 5.321 *
       9) AtBat > 173 38   3.1220 4.614 *
     5) CRuns > 92.5 27   2.2610 5.409 *
   3) CAtBat > 1322 130  42.0400 6.462
     6) Walks < 48.5 80  20.2600 6.232
      12) Hits < 98.5 41   9.8330 6.019
        24) AtBat < 335 36   6.8960 6.097 *
        25) AtBat > 335 5   1.1580 5.461 *
      13) Hits > 98.5 39   6.6220 6.456
        26) CHits < 590.5 12   1.8470 6.118 *
        27) CHits > 590.5 27   2.7950 6.606 *
     7) Walks > 48.5 50  10.8100 6.829
      14) CRBI < 369.5 16   0.9873 6.498 *
      15) CRBI > 369.5 34   7.2280 6.985
        30) PutOuts < 286 20   2.6210 6.786 *
        31) PutOuts > 286 14   2.6770 7.270 *
```

5. Create a plot of the tree, and interpret the results.

The plot below evidently indicates how an observation will be predicted. For example, the prediction of an observation with 'CAtBat' < 1322 and 'CRun' > 92.5 is 5.409, and this node is exactly the node 5 in task 4.

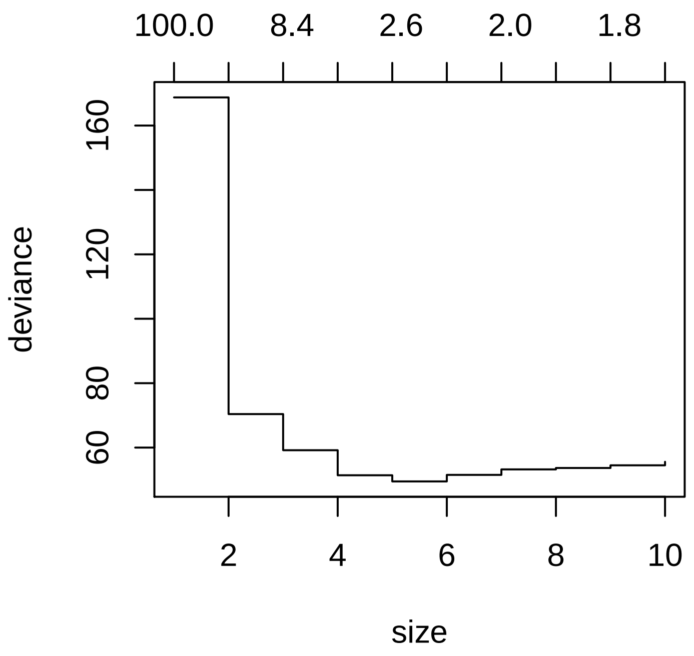
```
plot(tree_mod)
text(tree_mod, cex = 0.5)
```



6. Apply the `cv.tree()` function to the training set in order to determine the optimal tree size. Produce a plot with tree size on the x-axis and cross-validated classification mean squared error on the y-axis.

The plot shows that the deviance plummeted when the tree size was increased from 1 to 2. The least deviance appeared when tree size is 5.

```
set.seed(42)
cv_model <- cv.tree(tree_mod)
plot(cv_model)
```



7. Which tree size corresponds to the lowest cross-validated MSE?

As already mentioned in tasked 6, size of **5** leads to lowest MSE.

```
cv_model$size[which(cv_model$dev == min(cv_model$dev))]
```

```
[1] 5
```

8. 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.

Given the best size in task 7, the pruned tree shown below contains only five terminal nodes.

```r
tree_pruned <- prune.tree(tree_mod, best = 5)
tree_pruned
```

```
node), split, n, deviance, yval
      * denotes terminal node

 1) root 200 166.400 5.940
   2) CAtBat < 1322 70  23.300 4.971
     4) CRuns < 92.5 43  12.600 4.696 *
     5) CRuns > 92.5 27   2.261 5.409 *
   3) CAtBat > 1322 130  42.040 6.462
     6) Walks < 48.5 80  20.260 6.232
      12) Hits < 98.5 41   9.833 6.019 *
      13) Hits > 98.5 39   6.622 6.456 *
     7) Walks > 48.5 50  10.810 6.829 *
```

9. Compare the training MSE between the pruned and un-pruned trees. Which is higher?

The MSE of pruned tree is 0.2106276, which is higher than the unpruned tree's MSE (0.1581972). The result is foreseeable, since the pruned tree could not keep growing even though it could for better performance in training dataset.

```r
pred_train_pruned <- predict(tree_pruned, train_dat)
mse_train_pruned <- mean((pred_train_pruned - train_dat$Salary) ^ 2)
mse_train_pruned
```

```
[1] 0.2106276
```

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

With regard to test dataset, unpruned tree still outperforms pruned tree (0.3116231 vs. 0.3983201). The result, however, does not imply that the pruned tree must be inferior to the unpruned one in that the unpruned tree suffers from the risk of being overfitting. Hence, if dissimilar data appears, the pruned tree may perform better than the unpruned tree.

```r
pred_test <- predict(tree_mod, test_dat)
mse_test <- mean((pred_test - test_dat$Salary) ^ 2)
mse_test
```

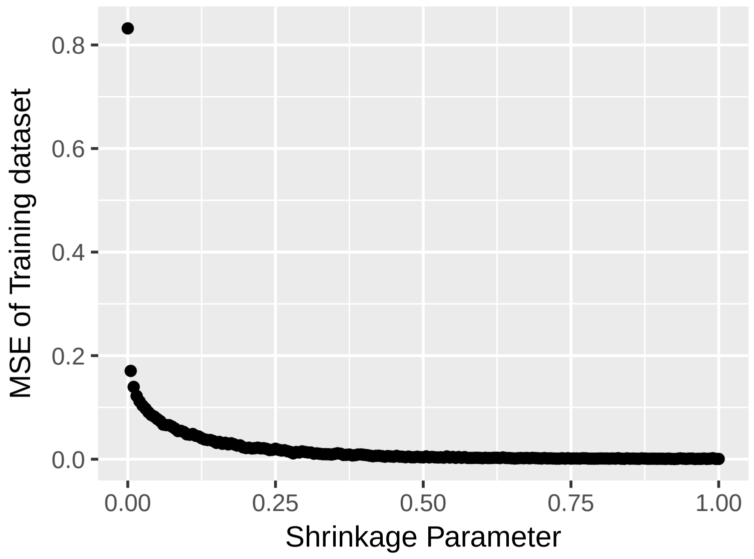
```
[1] 0.3116231
```

```r
pred_test_pruned <- predict(tree_pruned, test_dat)
mse_test_pruned <- mean((pred_test_pruned - test_dat$Salary) ^ 2)
mse_test_pruned
```

```
[1] 0.3983201
```

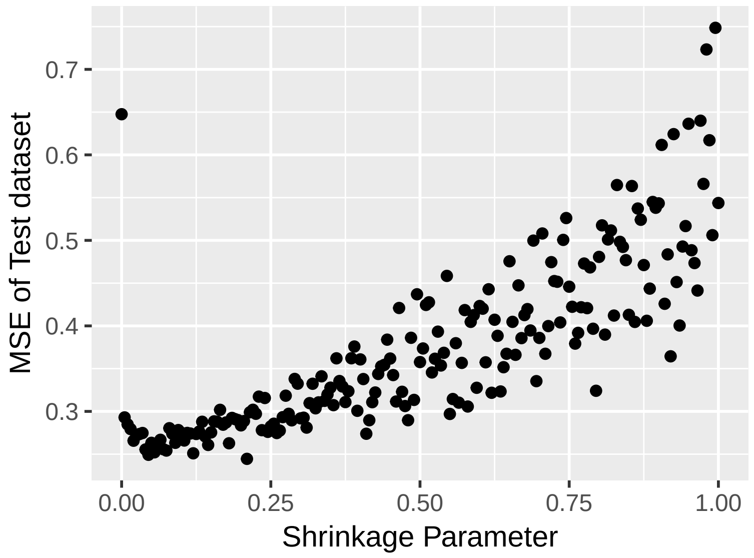
11. Perform boosting on the training set with 1,000 trees for a range of values of the shrinkage parameter lambda. Produce a plot with different shrinkage values on the x-axis and the corresponding training set MSE on the y-axis.

As can be seen from the plot, the MSE of boosting models decreased to almost zero when the shrinkage parameter varied from 0 to 1.



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

The MSE of test dataset, on the other hand, did not decrease as the shrinkage parameter became larger. Instead, the plot reveals an upward trend of the MSE when shrinkage parameter increased. This result is a typical situation of being overfitting, namely, a model obtains almost perfect performance in training dataset, but performs disappointingly when applied to test dataset.



13. Which variables appear to be the most important predictors in the boosted model?

The most influential predictor in the boosted tree is the 'CAtBat' variable, which is also the first splitting node of the decision tree in task 3 to 5.

```
# shrinkage of 0.225 allows us obtain the least MSE in test dataset (showed in Task 12)
gbm_mod <- gbm(Salary ~ .,distribution = "gaussian",
               data = train_dat, n.trees = 1000, shrinkage = 0.225)
kable(summary(gbm_mod, plotit = FALSE)[1:5, ])
```

	var	rel.inf
CAtBat	CAtBat	15.440660
CRBI	CRBI	10.344093
CRuns	CRuns	9.311075
PutOuts	PutOuts	8.849853
Walks	Walks	7.382570

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

After applying bagging (provided in **ipred** package), we can see that the bagging tree's MSE of test dataset is 0.2969247, which is better than the single tree's MSE in Task 10 (0.3116231).

```
set.seed(42)
bagging_mod <- bagging(Salary ~ ., data = train_dat)
pred_test_bagging <- predict(bagging_mod, test_dat)

mse_test_bagging <- mean((pred_test_bagging - test_dat$Salary) ^ 2)
mse_test_bagging
```

```
[1] 0.2969247
```

15. Now apply random forest to the training set. What is the test set MSE for this approach? Which variables appear to be the most important predictors in the random forest model?

The random forest approach can be accomplished by `randomForest` in the **randomForest** package. The test dataset MSE is 0.2173247. In addition, for this model, the most important predictor is the 'CAtBat' variable.

```
set.seed(42)
rf_mod <- randomForest(Salary ~ ., data = train_dat)

pred_test_rf <- predict(rf_mod, test_dat)
mse_test_rf <- mean((pred_test_rf - test_dat$Salary) ^ 2)
mse_test_rf
```

```
[1] 0.2173247
```

```
d <- data.frame(Features = rownames(rf_mod$importance),
                Importance = rf_mod$importance[, 1])
```

```r
kable(d[order(d$Importance, decreasing = TRUE)[1:5], ])
```

	Features	Importance
CAtBat	CAtBat	35.11249
CHits	CHits	27.89276
CRuns	CRuns	22.33031
CRBI	CRBI	15.51534
CWalks	CWalks	14.91341