HW09

Zixin Ouyang

11/19/2017

Exercise 1

```
library(mlbench)
set.seed(42)
sim_trn = mlbench.spirals(n = 2500, cycles = 1.5, sd = 0.125)
sim_trn = data.frame(sim_trn$x, class = as.factor(sim_trn$classes))
sim_tst = mlbench.spirals(n = 10000, cycles = 1.5, sd = 0.125)
sim_tst = data.frame(sim_tst$x, class = as.factor(sim_tst$classes))
```

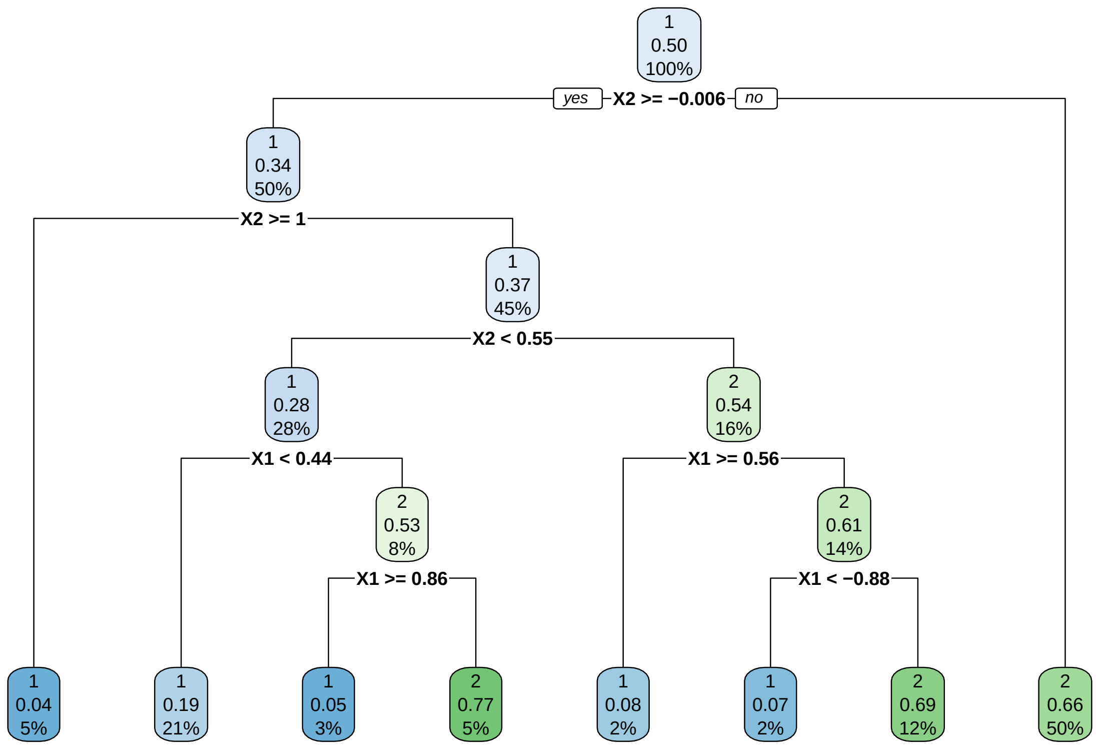
```
uin = 659017838
set.seed(uin)
```

```
library(caret)
cv_5 = trainControl(method = "cv", number = 5)
```

```
glm_cv_time = system.time({
  sim_glm_cv  = train(
    class ~ .,
    data = sim_trn,
    trControl = cv_5,
    method = "glm")
})

tree_cv_time = system.time({
  sim_tree_cv = train(
    class ~ .,
    data = sim_trn,
    trControl = cv_5,
    method = "rpart")
})
```

```
library(rpart.plot)
rpart.plot(sim_tree_cv$finalModel)
```




```
rf_grid = expand.grid(mtry = c(1, 2))
```

```
library(randomForest)
```

```
rf_cv_time = system.time({
  sim_rf_cv  = train(
    class ~ .,
    data = sim_trn,
    trControl = cv_5,
      method = "rf",
    tuneGrid=rf_grid)
})
```

```
rf_oob_time = system.time({
  sim_rf_oob  = train(
    class ~ .,
    data = sim_trn,
    trControl = trainControl(method = "oob"),
    method = "rf",
    tuneGrid=rf_grid)
})
```

```
best_tune = c(NA, sim_tree_cv$bestTune$cp, sim_rf_oob$bestTune$mtry, sim_rf_cv$bestTune$mtry)
```

```
resampled_acc = c(max(sim_glm_cv$results$Accuracy),
                  max(sim_tree_cv$results$Accuracy),
                  max(sim_rf_cv$results$Accuracy),
                  max(sim_rf_oob$results$Accuracy))
```

```
calc_acc = function(actual, predicted) {
  mean(actual == predicted)
}
```

```r
glm_cv_tst_acc = calc_acc(predicted = predict(sim_glm_cv, sim_tst),
                          actual    = sim_tst$class)

tree_cv_tst_acc = calc_acc(predicted = predict(sim_tree_cv, sim_tst),
                           actual    = sim_tst$class)

rf_cv_tst_acc = calc_acc(predicted = predict(sim_rf_cv, sim_tst),
                         actual    = sim_tst$class)

rf_oob_tst_acc = calc_acc(predicted = predict(sim_rf_oob, sim_tst),
                          actual    = sim_tst$class)

test_acc = c(glm_cv_tst_acc, tree_cv_tst_acc, rf_cv_tst_acc, rf_oob_tst_acc)
```

Model	Chosen tuning parameter	Elapsed tuning time	Resampled Accuracy	Test Accuracy
Logistic with CV	none	1.487	0.6572	0.6606
Tree with CV	0.0196	1.572	0.7328	0.7233
RF with CV	1	10.389	0.8492	0.8509
RF with OOB	1	3.139	0.8548	0.8519

Exercise 2

```r
library(ISLR)
Hitters = na.omit(Hitters)
```

```r
uin = 659017838
set.seed(uin)
hit_idx = createDataPartition(Hitters$Salary, p = 0.6, list = FALSE)
hit_trn = Hitters[hit_idx,]
hit_tst = Hitters[-hit_idx,]
```

```r
gbm_grid = expand.grid(interaction.depth = c(1, 2),
                       n.trees = c(500, 1000, 1500),
                       shrinkage = c(0.001, 0.01, 0.1),
                       n.minobsinnode = 10)
```

```r
gbm_mod = train(
  Salary ~ .,
  data = hit_trn,
  trControl = trainControl(method = "cv", number = 5),
  method = "gbm",
  tuneGrid = gbm_grid,
  verbose = FALSE
)
```

```r
rf_grid = expand.grid(mtry = 1:(ncol(hit_trn) - 1))
rf_mod  = train(
    Salary ~ .,
    data = hit_trn,
    trControl = trainControl(method = "oob"),
```

```r
    method = "rf",
    tuneGrid = rf_grid)
```

```r
bag_mod = train(
    Salary ~ .,
    data = hit_trn,
    trControl = trainControl(method = "oob"),
    method = "rf",
    tuneGrid = data.frame(mtry = (ncol(hit_trn) - 1)))
```

```r
models = list(gbm_mod, rf_mod, bag_mod)
```

```r
rmse = function(actual, predicted) {
  sqrt(mean((actual - predicted) ^ 2))
}
```

```r
get_rmse = function(model, data, response) {
  rmse(actual = data[, response],
       predicted = predict(model, data))
}
```

```r
test_rmse = sapply(models, get_rmse, data = hit_tst, response = "Salary")
```

```r
get_best_result = function(caret_fit) {
  best = which(rownames(caret_fit$results) == rownames(caret_fit$bestTune))
  best_result = caret_fit$results[best, ]
  rownames(best_result) = NULL
  best_result
}
```

Model	Resampled RMSE	Test RMSE
gbm_mod	296.0714716	314.1562942
rf_mod	279.2478206	336.9923979
bag_mod	290.85675	334.3302984

Exercise 3

```r
rf_log_mod  = train(
    log(Salary) ~ .,
    data = hit_trn,
    trControl = trainControl(method = "oob"),
    method = "rf")
```

```r
trans_predicted=exp(predict(rf_log_mod, newdata = hit_tst))
trans_rmse=rmse(actual = hit_tst$Salary,predicted = trans_predicted)
```

Model	Test RMSE
rf_mod	336.9923979
rf_log_mod	331.8389838

Exercise 4

Timing

```r
rf_cv_time["elapsed"] / rf_oob_time["elapsed"]
```

```
##  elapsed
## 3.309653
```

(a) The speed-up for OOB is about four times that of 5-fold CV, instead of the five times that would have been expected. There appears to be some additional overhead in using OOB.
(b) The tuned values of mtry for both random forests tuned are 1. So they choose the same model.
(c) Logistic: Performs the worst. This is expected as clearly a non-linear decision boundary is needed. Single Tree: Better than logistic, but not the best seen here. We see above that this is not a very deep tree. It will have non-linear boundaries, but since it uses binary splits, they will be rectangular regions. Random Forest: First note that both essentially fit the same model. (The exact forests will be different due to randomization.) By using many trees (500) the boundaries will become less rectangular than the single tree, and will better match the spiral data in the data.

Salary

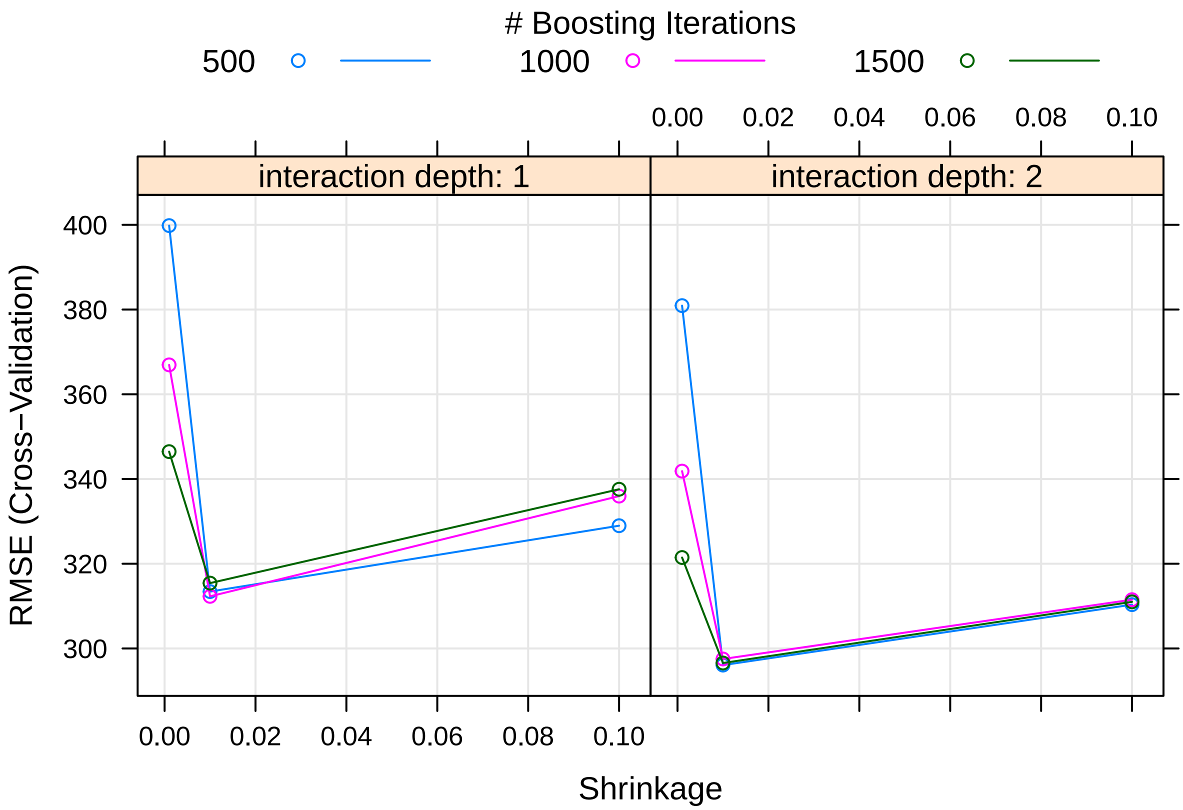
(d)

```r
rf_mod$bestTune
```

```
##   mtry
## 3    3
```

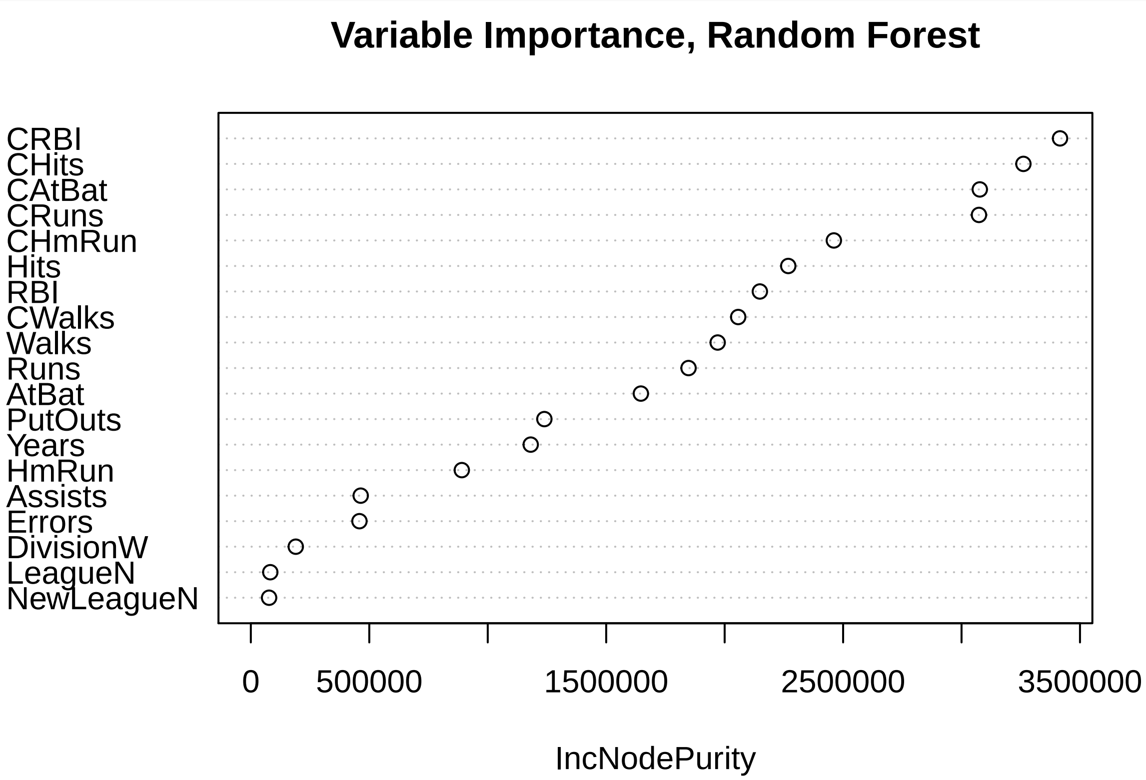
(e)

```r
plot(gbm_mod)
```



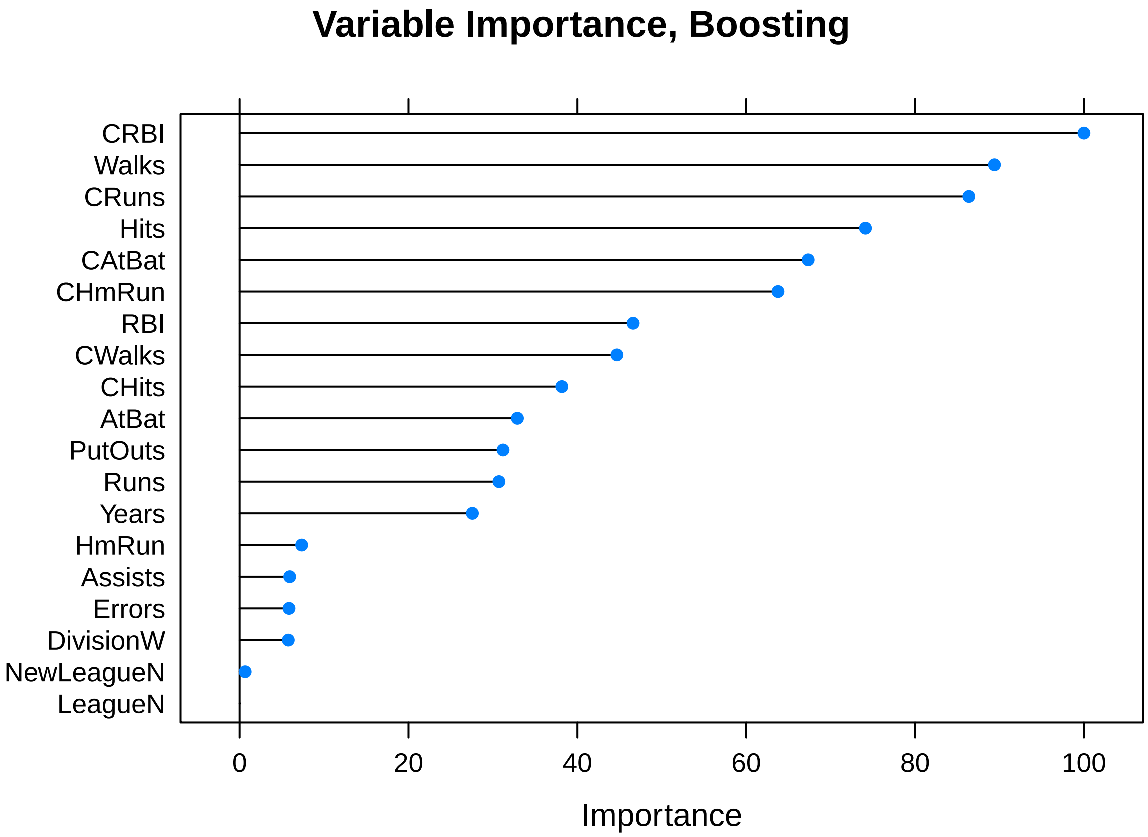
(f)

```r
varImpPlot(rf_mod$finalModel, main = "Variable Importance, Random Forest")
```



(g)

```r
plot(varImp(gbm_mod), main = "Variable Importance, Boosting")
```



(h) According to the random forest, the three most important predictors are CRBI, CHits,CAtBat.
(i) According to the boosted model, the three most important predictors are CRBI, Walks,CRuns.

Transformation

(j) I think the transformation was unnecessary because the test RMSE did not change a lot.