Random Forest Capstone

2025-10-23

Lab Report

Research Question

How does the number of trees influence the performance and stability of predicting the medv of the Boston housing dataset.

Hypothesis

Initially, an increase in the number of trees will help model stability and prediction. However, there will get to a point where the increase in the number of trees will no longer improve model stability and prediction accuracy.

Parameters

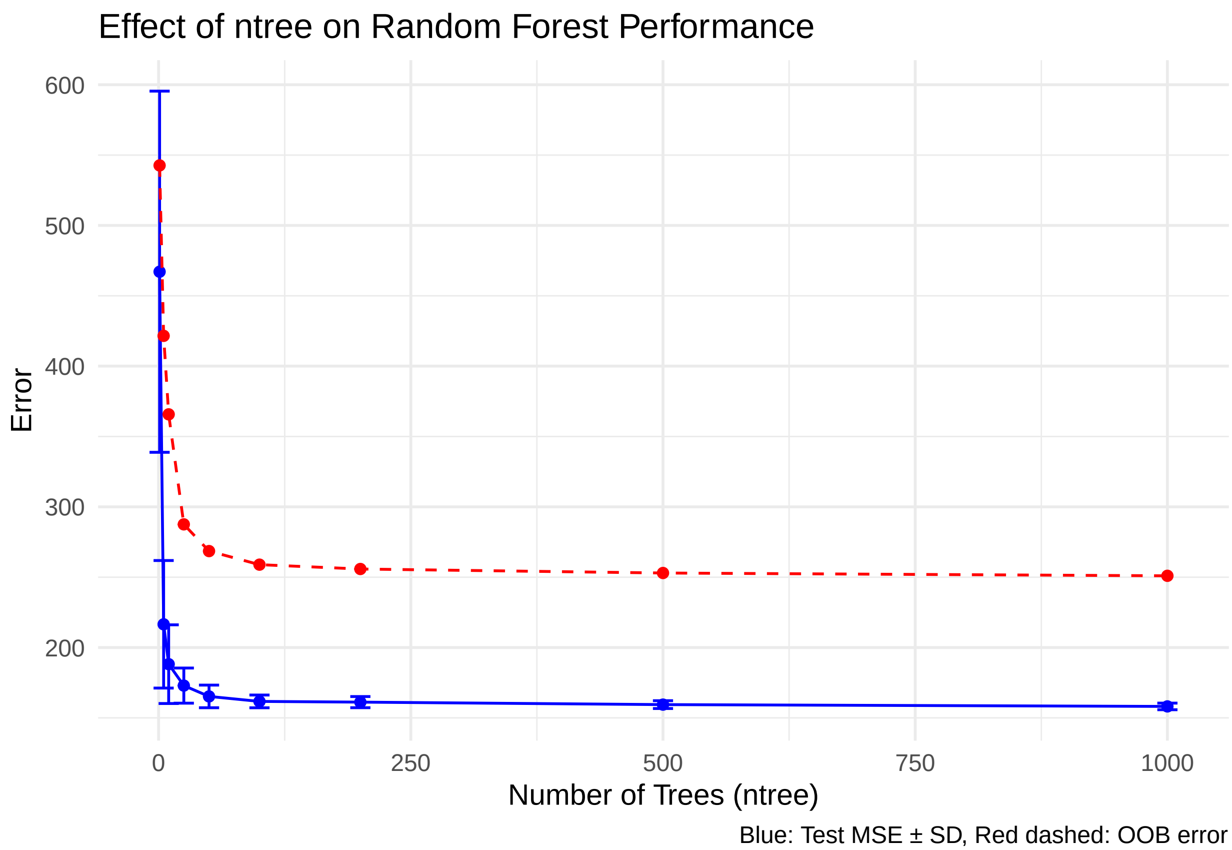
The mtry will be fixed at 6. I will test ntree values of 1,5,10,25,50,100,200, and 500. Lastly, I will use test MSE and OOB error to evaluate the models.

Running the Code

```r
library(MASS)
library(randomForest)
library(tidyverse)

data1 <- read.csv('onlyCompleteData.csv')

set.seed(123)
train_idx <- which(data1$year == 2024)
train <- data1[train_idx, ]
test <- data1[-train_idx, ]

mtry_val <- 6
ntree_values <- c(1,5,10, 25, 50, 100, 200, 500,1000)
n_reps <- 20

results <- data.frame(ntree=integer(),seed=integer(),test_MSE=double(),OOB_error=double())

for (nt in ntree_values) { # goes through all different ntree values
  for (s in 1:n_reps) { # do many times for each ntree
```

```
    set.seed(s)
    rf_model <- randomForest(WRC. ~ ., data=train, mtry=mtry_val, ntree=nt)
    yhat.bag <- predict(rf_model, newdata=test)
    test_mse <- mean((yhat.bag - test$WRC.)^2)
    results <- rbind(results, data.frame(ntree=nt,test_MSE=test_mse,OOB_error=rf_model$mse[nt]))
  }
}

results_all <- results %>% group_by(ntree) %>%summarize(mean_test_MSE = mean(test_MSE),sd_test_MSE = sd
            mean_OOB_error = mean(OOB_error),sd_OOB_error = sd(OOB_error))
```

```
ggplot(results_all, aes(x=ntree)) + geom_errorbar(aes(ymin=mean_test_MSE-sd_test_MSE, ymax=mean_test_MSE
                width=20, color="blue") +geom_line(aes(y=mean_test_MSE), color="blue") + geom_point(aes
  geom_line(aes(y=mean_OOB_error), color="red",linetype="dashed") +
  geom_point(aes(y=mean_OOB_error), color="red") +
  labs(title="Effect of ntree on Random Forest Performance",x="Number of Trees (ntree)",y="Error",captio
  theme_minimal()
```




```
results_all
```

```
## # A tibble: 9 x 5
##   ntree mean_test_MSE sd_test_MSE mean_OOB_error sd_OOB_error
##   <dbl>         <dbl>       <dbl>          <dbl>        <dbl>
## 1     1          467.        128.           543.         169.
```

```
## 2     5           217.       45.3            422.       101.
## 3    10           188.       28.0            366.        68.6
## 4    25           173.       12.5            288.        32.7
## 5    50           165.        8.03           269.        18.3
## 6   100           162.        4.53           259.        11.8
## 7   200           161.        3.96           256.         9.38
## 8   500           159.        2.79           253.         5.44
## 9  1000           158.        2.35           251.         3.89
```

```r
# Packages
library(MASS)
library(randomForest)
library(ggplot2)


set.seed(1)
N <- nrow(data1)
train <- which(data1$year == 2024)

boston.train <- data1[train, ]
boston.test  <- data1[-train, ]

# Response vectors
y.test <- boston.test$WRC.

set.seed(1)
bag.boston <- randomForest(WRC. ~ ., data = data1,
                           subset = train, mtry = 12,
                           importance = TRUE)
bag.boston
```

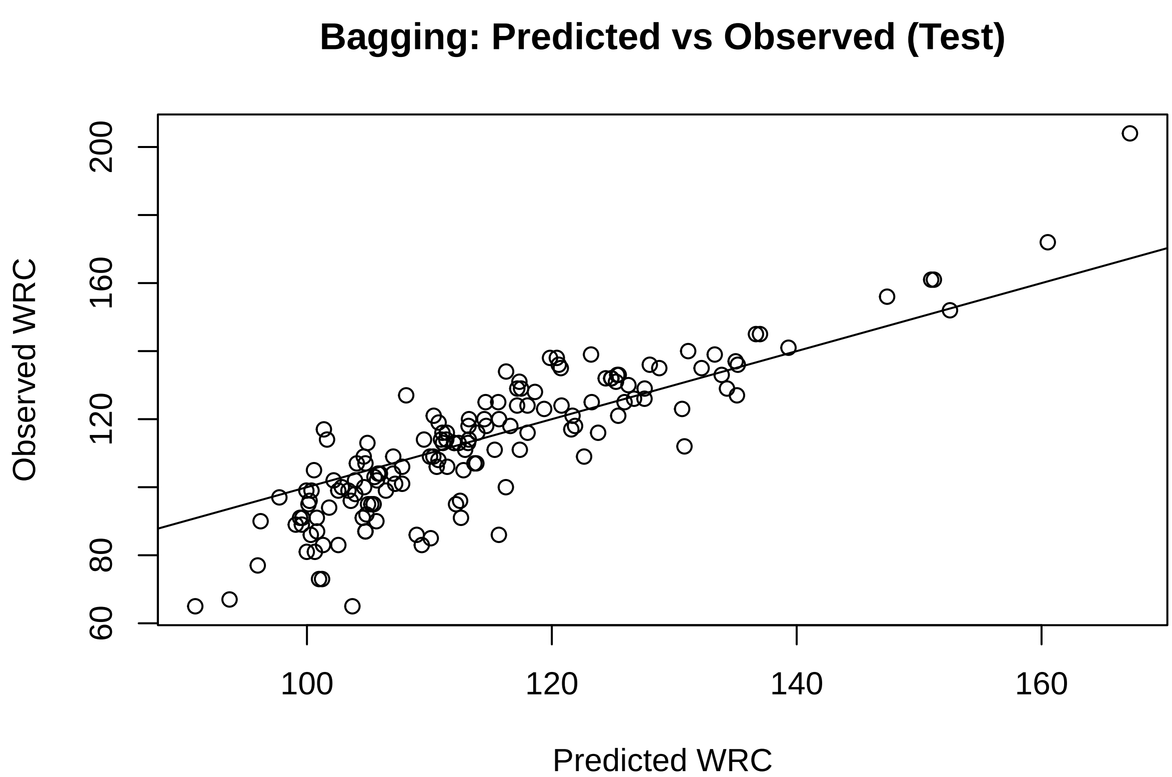
```
##
## Call:
##  randomForest(formula = WRC. ~ ., data = data1, mtry = 12, importance = TRUE,      subset = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 12
##
##           Mean of squared residuals: 221.0222
##                     % Var explained: 54.19
```

```r
yhat.bag <- predict(bag.boston, newdata = boston.test)

# Test MSE
mse_bag <- mean((yhat.bag - y.test)^2)
mse_bag
```

```
## [1] 134.0204
```

```r
# Plot predicted vs observed
plot(yhat.bag, y.test,
     xlab = "Predicted WRC",
```

```r
     ylab = "Observed WRC",
     main = "Bagging: Predicted vs Observed (Test)")
abline(0, 1)
```




```r
set.seed(1)
rf.boston <- randomForest(WRC. ~ ., data = data1,
                          subset = train, mtry = 6,
                          importance = TRUE)

yhat.rf <- predict(rf.boston, newdata = boston.test)
mse_rf <- mean((yhat.rf - y.test)^2)
c(mse_bag = mse_bag, mse_rf = mse_rf)
```

```
##  mse_bag   mse_rf
## 134.0204 159.2697
```

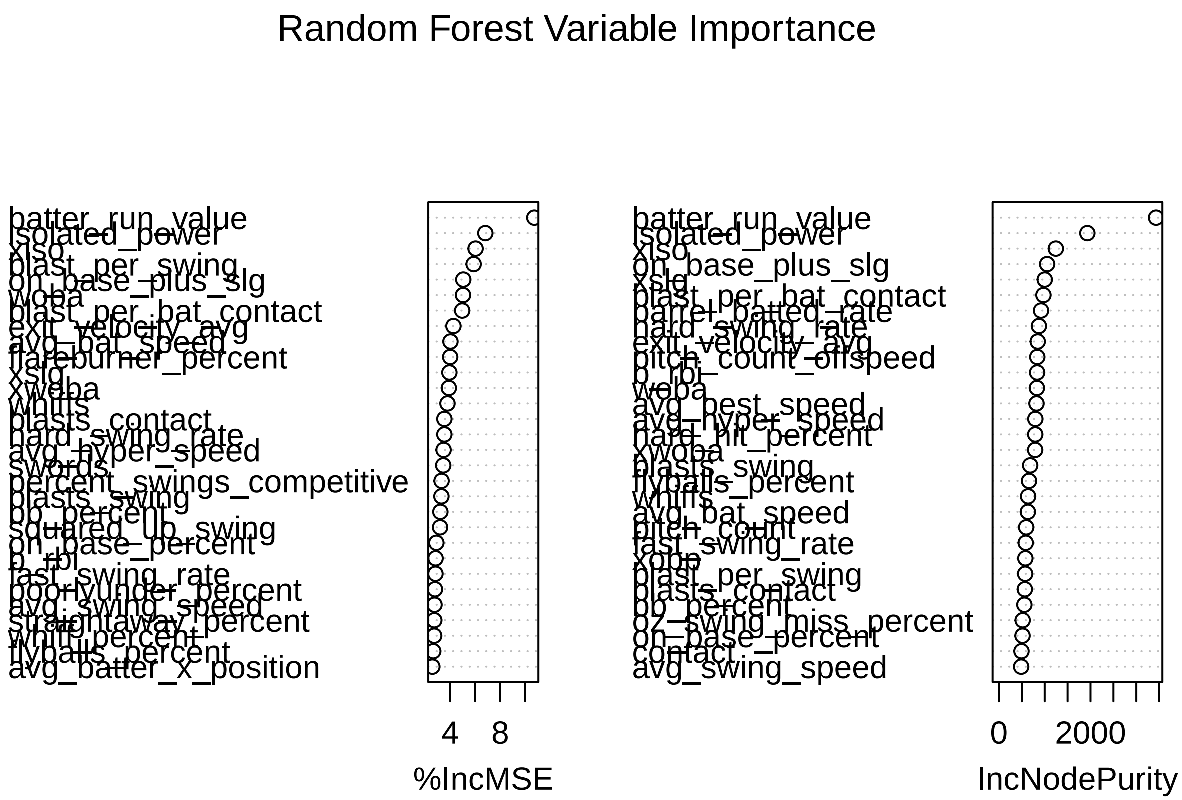
```r
importance(rf.boston)
```

```
##                                   %IncMSE IncNodePurity
## player.name                   -1.21185342     216.50609
## player_id                     -0.30553609     226.55649
## year                           0.00000000       0.00000
## player_age                     1.74792113     135.22045
## ab                             1.17875490     206.56669
```

```
## pa                        1.69838655    269.41539
## k_percent                 0.80992747    192.86564
## bb_percent                3.21192654    556.06730
## on_base_percent           2.89202548    516.19294
## on_base_plus_slg          5.04462262   1053.57004
## isolated_power            6.81081463   1931.56885
## b_rbi                     2.82744477    834.99098
## b_walkoff                 1.26048008    140.44093
## xba                      -0.51729076    425.81863
## xslg                      3.93312178   1000.98856
## woba                      5.02477462    829.18291
## xwoba                     3.88816293    791.67386
## xobp                      2.30668816    577.83978
## xiso                      6.02964898   1243.01092
## xbadiff                   0.73517105    312.45529
## xslgdiff                  0.42664003    292.91007
## wobadiff                 -0.94450764    217.97177
## avg_swing_speed           2.74306040    486.04815
## fast_swing_rate           2.81913814    585.71252
## blasts_contact            3.53222158    565.80070
## blasts_swing              3.29505958    683.15174
## squared_up_contact        2.39975209    232.06789
## squared_up_swing          3.18928958    231.82940
## avg_swing_length          1.50021747    192.94863
## swords                    3.44099923    288.19758
## attack_angle              1.99362675    268.20347
## attack_direction          1.34155860    202.65625
## ideal_angle_rate          2.06216931    224.76117
## vertical_swing_path       0.70847874    259.33681
## exit_velocity_avg         4.25793167    847.89712
## launch_angle_avg          1.79932686    295.53707
## sweet_spot_percent        0.85316595    316.08547
## barrel_batted_rate        1.97956923    919.99498
## solidcontact_percent     -0.88349140    164.21332
## flareburner_percent       3.99789065    450.67755
## poorlyunder_percent       2.77915379    247.40717
## poorlytopped_percent      0.23080368    383.08976
## poorlyweak_percent       -0.38193192    157.06612
## hard_hit_percent          2.07312143    793.38980
## avg_best_speed            2.41659070    820.93979
## avg_hyper_speed           3.46863482    795.82119
## z_swing_percent           1.77475360    362.66774
## z_swing_miss_percent      2.35960402    238.40913
## oz_swing_percent         -1.08675927    247.37704
## oz_swing_miss_percent     1.25835325    518.77773
## oz_contact_percent        1.71405565    398.03187
## out_zone_percent          1.42987001    139.03692
## meatball_swing_percent    1.67837611    269.61012
## meatball_percent          1.19470776    182.47170
## pitch_count_offspeed      2.23663731    838.43464
## pitch_count_fastball     -1.79706317    127.57378
## pitch_count_breaking      1.05284893    163.69261
## pitch_count               2.46452098    596.56557
## iz_contact_percent        0.54929864    216.73504
```

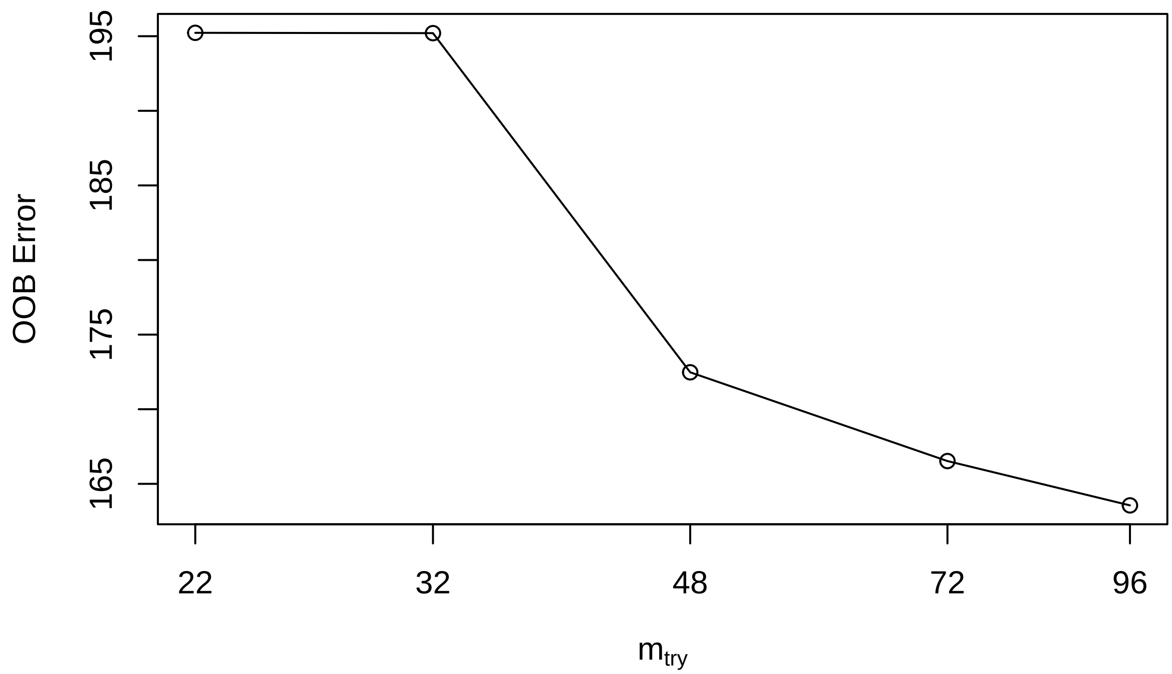
```
## in_zone_percent              1.59377562     206.50840
## edge_percent                 2.06393680     213.05086
## whiff_percent                2.71800957     321.16411
## swing_percent               -0.12221427     275.69043
## pull_percent                 1.02503646     249.48178
## straightaway_percent         2.72795619     308.68278
## opposite_percent             1.36699743     294.21578
## f_strike_percent             2.20202676     351.00407
## groundballs_percent          1.24962944     253.00633
## flyballs_percent             2.64526444     657.86891
## linedrives_percent           1.52491512     298.75428
## popups_percent               1.85530089     293.27783
## id                           0.30156765     253.07660
## bat_side                     1.35145563      45.05291
## side                         0.09171379      96.44633
## avg_batter_y_position       -1.59132425     196.00190
## avg_batter_x_position        2.57211059     229.47311
## avg_foot_sep                 0.98493318     219.21926
## avg_stance_angle             1.93464984     174.49810
## avg_intercept_y_vs_batter   -0.24335539     330.50249
## avg_intercept_y_vs_plate     2.21125623     395.32236
## swings_competitive           2.48255486     474.59602
## percent_swings_competitive   3.30823675     225.95232
## contact                      2.50672994     493.23295
## avg_bat_speed                4.01999102     632.26844
## hard_swing_rate              3.53081356     873.33628
## squared_up_per_bat_contact   0.85693576     317.32502
## squared_up_per_swing         2.50255060     232.48020
## blast_per_bat_contact        4.95885391     972.50777
## blast_per_swing              5.87368452     573.52993
## swing_length                 2.28510311     338.02837
## swords_tracking             -0.19449430     228.40300
## batter_run_value            10.72785547    3431.76524
## whiffs                       3.78217461     638.15367
## whiff_per_swing             -1.15228190     211.63526
## batted_ball_events           1.82664782     306.43399
## batted_ball_event_per_swing  0.06043781     278.21095
```

```r
varImpPlot(rf.boston, main = "Random Forest Variable Importance")
```




```r
set.seed(2)
tune.out <- tuneRF(x = boston.train[, setdiff(names(data1), "WRC.")],
                   y = boston.train$WRC.,
                   stepFactor = 1.5,
                   improve = 0.01,
                   ntreeTry = 500,
                   trace = TRUE,
                   plot = TRUE)
```

```
## mtry = 32  OOB error = 195.2063
## Searching left ...
## mtry = 22    OOB error = 195.2271
## -0.0001064772 0.01
## Searching right ...
## mtry = 48    OOB error = 172.4761
## 0.1164419 0.01
## mtry = 72    OOB error = 166.5269
## 0.03449291 0.01
## mtry = 96    OOB error = 163.5571
## 0.01783359 0.01
```



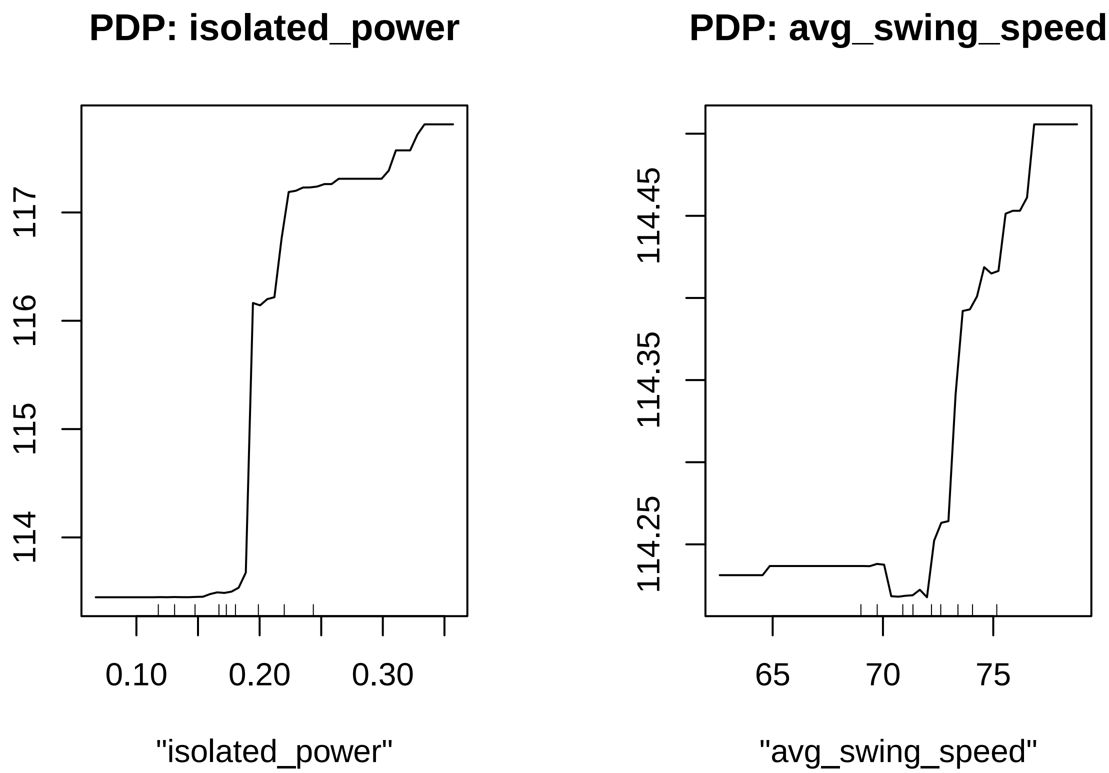

```r
# Best mtry from tuneRF (lower OOB error is better)
best.mtry <- tune.out[which.min(tune.out[, "OOBError"]), "mtry"]
best.mtry
```

```
## [1] 96
```

```r
set.seed(3)
rf.tuned <- randomForest(WRC. ~ ., data = data1,
                         subset = train, mtry = best.mtry,
                         ntree = 500, importance = TRUE)

mse_rf_tuned <- mean( (predict(rf.tuned, boston.test) - y.test)^2 )
c(mse_bag = mse_bag, mse_rf = mse_rf, mse_rf_tuned = mse_rf_tuned)
```

```
##      mse_bag       mse_rf mse_rf_tuned
##     134.0204     159.2697     123.5992
```

```r
# Partial dependence plots give the marginal effect of a feature
par(mfrow = c(1, 2))
partialPlot(rf.tuned, pred.data = boston.test, x.var = "isolated_power",
            main = "PDP: isolated_power")
partialPlot(rf.tuned, pred.data = boston.test, x.var = "avg_swing_speed",
            main = "PDP: avg_swing_speed")
```


PDP: isolated_power
117
116
115
114
0.10
0.20
0.30
"isolated_power"
PDP: avg_swing_speed
114.45
114.35
114.25
65
70
75
"avg_swing_speed"