Task 3: Random Forest Implementation

ED Cullen

2025-10-03

Table of Contents

[Overview 1](#_Toc210403163)

[1. Setup and Data Loading 1](#_Toc210403164)

[2. Baseline Model 2](#_Toc210403165)

[3. Hyperparameter Tuning 2](#_Toc210403166)

[3.1 Tuning ntree 2](#_Toc210403167)

[3.2 Tuning mtry 3](#_Toc210403168)

[3.3 Tuning nodesize 4](#_Toc210403169)

[4. Final Model 5](#_Toc210403170)

[5. Parameter Documentation 6](#_Toc210403171)

[ntree (Number of Trees) 6](#_Toc210403172)

[mtry (Variables per Split) 6](#_Toc210403173)

[nodesize (Minimum Node Size) 6](#_Toc210403174)

[Session Information 6](#_Toc210403175)

# Overview

Task 3: Implement Random Forest regression with systematic hyperparameter tuning.

# 1. Setup and Data Loading

library(randomForest)  
library(tidyverse)

# Auto-detect environment and set paths  
current\_dir <- basename(getwd())  
if (current\_dir == "Task\_03") {  
 train\_path <- "../../Data/04\_Split/train\_data.csv"  
 test\_path <- "../../Data/04\_Split/test\_data.csv"  
 outputs\_path <- "outputs"  
} else {  
 train\_path <- "02\_Project/Data/04\_Split/train\_data.csv"  
 test\_path <- "02\_Project/Data/04\_Split/test\_data.csv"  
 outputs\_path <- "02\_Project/Milestone\_3/Task\_03/outputs"  
}  
  
if (!dir.exists(outputs\_path)) {  
 dir.create(outputs\_path, recursive = TRUE)  
}

train\_data <- read.csv(train\_path)  
test\_data <- read.csv(test\_path)

# 2. Baseline Model

# Train baseline model with default parameters  
baseline\_rf <- randomForest(  
 value\_log\_scaled ~ .,  
 data = train\_data,  
 ntree = 500,  
 importance = TRUE  
)  
  
print(baseline\_rf)

##   
## Call:  
## randomForest(formula = value\_log\_scaled ~ ., data = train\_data, ntree = 500, importance = TRUE)   
## Type of random forest: regression  
## Number of trees: 500  
## No. of variables tried at each split: 9  
##   
## Mean of squared residuals: 0.005654711  
## % Var explained: 99.45

# 3. Hyperparameter Tuning

## 3.1 Tuning ntree

ntree\_values <- c(500, 750, 1000, 1250, 1500, 2000)  
ntree\_results <- data.frame(ntree = integer(), OOB\_Error = numeric(), Test\_RMSE = numeric())  
  
for (n\_trees in ntree\_values) {  
 rf\_model <- randomForest(  
 value\_log\_scaled ~ .,  
 data = train\_data,  
 ntree = n\_trees,  
 mtry = floor(sqrt(ncol(train\_data) - 1))  
 )  
  
 oob\_error <- tail(rf\_model$mse, 1)  
 pred <- predict(rf\_model, newdata = test\_data)  
 test\_rmse <- sqrt(mean((test\_data$value\_log\_scaled - pred)^2))  
  
 ntree\_results <- rbind(ntree\_results, data.frame(  
 ntree = n\_trees,  
 OOB\_Error = oob\_error,  
 Test\_RMSE = test\_rmse  
 ))  
}  
  
print(ntree\_results)

## ntree OOB\_Error Test\_RMSE  
## 1 500 0.03098836 0.1563969  
## 2 750 0.03060590 0.1581281  
## 3 1000 0.03003316 0.1575704  
## 4 1250 0.03027587 0.1585536  
## 5 1500 0.03028832 0.1568796  
## 6 2000 0.03136728 0.1590984

optimal\_ntree <- ntree\_results$ntree[which.min(ntree\_results$Test\_RMSE)]  
cat("Optimal ntree:", optimal\_ntree, "\n")

## Optimal ntree: 500

## 3.2 Tuning mtry

num\_features <- ncol(train\_data) - 1  
mtry\_values <- unique(floor(seq(sqrt(num\_features), num\_features/3, length.out = 6)))  
mtry\_results <- data.frame(mtry = integer(), OOB\_Error = numeric(), Test\_RMSE = numeric())  
  
for (m in mtry\_values) {  
 rf\_model <- randomForest(  
 value\_log\_scaled ~ .,  
 data = train\_data,  
 ntree = optimal\_ntree,  
 mtry = m  
 )  
  
 oob\_error <- tail(rf\_model$mse, 1)  
 pred <- predict(rf\_model, newdata = test\_data)  
 test\_rmse <- sqrt(mean((test\_data$value\_log\_scaled - pred)^2))  
  
 mtry\_results <- rbind(mtry\_results, data.frame(  
 mtry = m,  
 OOB\_Error = oob\_error,  
 Test\_RMSE = test\_rmse  
 ))  
}  
  
print(mtry\_results)

## mtry OOB\_Error Test\_RMSE  
## 1 5 0.030406207 0.15715877  
## 2 6 0.020744146 0.12264146  
## 3 7 0.012819105 0.09453908  
## 4 8 0.007623756 0.07348939  
## 5 9 0.005999400 0.06170241

optimal\_mtry <- mtry\_results$mtry[which.min(mtry\_results$Test\_RMSE)]  
cat("Optimal mtry:", optimal\_mtry, "\n")

## Optimal mtry: 9

## 3.3 Tuning nodesize

nodesize\_values <- c(1, 2, 3, 5, 7, 10)  
nodesize\_results <- data.frame(nodesize = integer(), OOB\_Error = numeric(), Test\_RMSE = numeric())  
  
for (ns in nodesize\_values) {  
 rf\_model <- randomForest(  
 value\_log\_scaled ~ .,  
 data = train\_data,  
 ntree = optimal\_ntree,  
 mtry = optimal\_mtry,  
 nodesize = ns  
 )  
  
 oob\_error <- tail(rf\_model$mse, 1)  
 pred <- predict(rf\_model, newdata = test\_data)  
 test\_rmse <- sqrt(mean((test\_data$value\_log\_scaled - pred)^2))  
  
 nodesize\_results <- rbind(nodesize\_results, data.frame(  
 nodesize = ns,  
 OOB\_Error = oob\_error,  
 Test\_RMSE = test\_rmse  
 ))  
}  
  
print(nodesize\_results)

## nodesize OOB\_Error Test\_RMSE  
## 1 1 0.004980290 0.05526238  
## 2 2 0.005199664 0.05791136  
## 3 3 0.004804617 0.05570661  
## 4 5 0.005745212 0.05928084  
## 5 7 0.006454809 0.06427381  
## 6 10 0.007235824 0.06973080

optimal\_nodesize <- nodesize\_results$nodesize[which.min(nodesize\_results$Test\_RMSE)]  
cat("Optimal nodesize:", optimal\_nodesize, "\n")

## Optimal nodesize: 1

# 4. Final Model

# Train final model with optimized parameters  
final\_rf <- randomForest(  
 value\_log\_scaled ~ .,  
 data = train\_data,  
 ntree = optimal\_ntree,  
 mtry = optimal\_mtry,  
 nodesize = optimal\_nodesize,  
 importance = TRUE,  
 keep.forest = TRUE  
)  
  
print(final\_rf)

##   
## Call:  
## randomForest(formula = value\_log\_scaled ~ ., data = train\_data, ntree = optimal\_ntree, mtry = optimal\_mtry, nodesize = optimal\_nodesize, importance = TRUE, keep.forest = TRUE)   
## Type of random forest: regression  
## Number of trees: 500  
## No. of variables tried at each split: 9  
##   
## Mean of squared residuals: 0.004988555  
## % Var explained: 99.51

# Save model  
saveRDS(final\_rf, file.path(outputs\_path, "final\_random\_forest\_model.rds"))  
  
# Save tuning results  
write.csv(ntree\_results, file.path(outputs\_path, "ntree\_tuning.csv"), row.names = FALSE)  
write.csv(mtry\_results, file.path(outputs\_path, "mtry\_tuning.csv"), row.names = FALSE)  
write.csv(nodesize\_results, file.path(outputs\_path, "nodesize\_tuning.csv"), row.names = FALSE)

# 5. Parameter Documentation

## ntree (Number of Trees)

* **Tested:** 500-2000
* **Selected:** 500
* **Rationale:** Minimizes test RMSE while ensuring OOB error convergence

## mtry (Variables per Split)

* **Tested:** 5 to 9
* **Selected:** 9
* **Rationale:** Balances tree diversity with prediction accuracy

## nodesize (Minimum Node Size)

* **Tested:** 1-10
* **Selected:** 1
* **Rationale:** Prevents overfitting while maintaining model complexity