Classwork 3: Hyperparameter Tuning and Ensemble Methods

Machine Learning with Tree-Based Models in R

Prof. Asc. Endri Raco, Ph.D.

November 2025

Learning Objectives

By completing this classwork, you will:

- Implement systematic hyperparameter tuning using grid search
- Build bagged tree models using bootstrap aggregation
- Construct random forest models with feature randomization
- Compare ensemble methods against single decision trees
- Select optimal hyperparameters using cross-validation performance
- Visualize tuning results to understand parameter impacts

Time Allocation: 30 minutes

Tuning Grid (8 min) | Bagging (7 min) | Random Forests (10 min) | Comparison (5 min)

Dataset: Diabetes Classification Revisited

We return to the diabetes prediction task, now applying advanced ensemble techniques to improve upon our baseline classification tree from Classwork 1.

Task 1: Hyperparameter Tuning Setup (8 minutes)

Implement a systematic grid search to find optimal tree hyperparameters.

```r
# Load required libraries
library(tidymodels)
library(dplyr)

# Load and split diabetes data
data(diabetes, package = "modeldata")
set.seed(2025)

diabetes_split <- initial_split(diabetes, prop = 0.75, strata = outcome)
diabetes_train <- training(diabetes_split)
diabetes_test <- testing(diabetes_split)

# Task 1a: Create 5-fold cross-validation folds from training data
# Task 1b: Create a tunable decision tree specification
# Use tune() to mark tree_depth and min_n for tuning
tree_tune_spec <- decision_tree(
  mode = "classification",
  engine = "rpart",
  tree_depth = tune(),
  min_n = tune()
)

# Task 1c: Create a tuning grid
# Use grid_regular() with levels = 5 for both parameters


# Task 1d: Create a workflow with the tunable model


# Task 1e: Perform grid search tuning
# Use tune_grid() with your workflow, CV folds, and grid


# Task 1f: Visualize tuning results
# Use autoplot() to see how parameters affect performance
```

Question 1.1: What is the purpose of hyperparameter tuning? Why can't we simply use the default parameter values?

Your Answer:

Question 1.2: Examine your tuning visualization. Which hyperparameter (tree_depth or min_n) appears to have a stronger impact on model performance? What pattern do you observe?

Your Answer:

Question 1.3: What are the optimal hyperparameter values according to your grid search? Report the tree_depth and min_n that maximize AUC.

Your Answer:

Task 2: Bootstrap Aggregation (Bagging) (7 minutes)

Implement bagging to reduce variance through ensemble averaging.

```r
# Task 2a: Create a bagged tree specification
# Use bag_tree() with mode = "classification" and engine = "rpart"
bagging_spec <- bag_tree(
  mode = "classification",
  engine = "rpart"
) %>%
  set_args(times = 25)  # Use 25 bootstrap samples

# Task 2b: Fit the bagged model on training data
# Use all predictors: outcome ~ .


# Task 2c: Generate probability predictions on test set
# Task 2d: Calculate AUC for the bagged model
```

Question 2.1: Explain how bagging works. Why does creating multiple bootstrap samples and averaging their predictions reduce overfitting?

Your Answer:

Question 2.2: How does bagging differ from simply training one deep tree? What is the key mechanism that makes bagging effective?

Your Answer:

Task 3: Random Forest Implementation (10 minutes)

Build a random forest that combines bagging with feature randomization.

```r
# Task 3a: Create a random forest specification
# Use rand_forest() with mode = "classification" and engine = "ranger"
rf_spec <- rand_forest(
  mode = "classification",
  engine = "ranger",
  mtry = tune(),      # Number of features to sample
  trees = 100,        # Number of trees
  min_n = tune()      # Minimum node size
)

# Task 3b: Create a tuning grid for random forest
# mtry should range from 2 to 6, min_n from 5 to 20


# Task 3c: Create a workflow and tune the random forest


# Task 3d: Select the best random forest model


# Task 3e: Finalize the workflow with best parameters


# Task 3f: Fit the final random forest on full training data


# Task 3g: Generate predictions and calculate AUC on test set
```

Question 3.1: What is the key difference between bagging and random forests? What additional randomization step does random forest introduce?

Your Answer:

Question 3.2: The mtry parameter controls how many features are randomly sampled at each split. What is the optimal mtry value from your tuning? How does this compare to the total number of features available?

Your Answer:

Question 3.3: Why does feature randomization help create more diverse trees in the ensemble?

Your Answer:

Task 4: Model Comparison and Selection (10 minutes)

Compare all three approaches: single tree, bagging, and random forest.

```r
# Task 4a: Fit an optimally-tuned single decision tree for comparison
# Use the best parameters from Task 1
single_tree_spec <- decision_tree(
  mode = "classification",
  engine = "rpart",
  tree_depth = ___,  # Fill in your optimal value
  min_n = ___        # Fill in your optimal value
)

single_tree_fit <- single_tree_spec %>%
  fit(outcome ~ ., data = diabetes_train)

# Task 4b: Generate predictions for all three models on test set


# Task 4c: Calculate AUC for all three models


# Task 4d: Create a comparison table or visualization


# Task 4e: Calculate ROC curves for visual comparison
```

Question 4.1: Fill in the performance comparison table:

Model	Test Set AUC	Rank	Training Time (relative)
Single Decision Tree			Fast
Bagged Trees			Medium
Random Forest			Slower

Question 4.2: Which model performed best on the test set? Was the performance improvement worth the increased computational cost?

Your Answer:

Question 4.3: In what scenarios might you still prefer a single decision tree over random forest, despite lower predictive performance?

Your Answer:

Task 5: Critical Analysis (5 minutes)

Question 5.1: Random forests are often described as "black box" models compared to single decision trees. What does this mean, and why is interpretability reduced in ensemble methods?

Your Answer:

Question 5.2: You observe that your random forest has perfect accuracy (100%) on the training set but only 85% accuracy on the test set. What problem does this indicate, and what hyperparameter adjustments might help?

Your Answer:

Question 5.3: The lecture mentioned that ensemble methods can capture non-linear relationships better than single trees. Provide an intuitive explanation for why averaging multiple trees can approximate complex decision boundaries more effectively than one tree.

Your Answer:

Submission Instructions

1. Complete all code chunks, ensuring proper hyperparameter values are filled in
2. Answer all questions with detailed explanations
3. Verify all models run successfully and produce valid AUC scores
4. Knit this document to PDF
5. Submit the PDF through the course management system

Grading Rubric (100 points total)

Component	Points	Criteria
Task 1: Hyperparameter Tuning	25	Correct grid search implementation, visualization, and parameter selection
Task 2: Bagging	20	Proper bagging implementation and conceptual understanding
Task 3: Random Forest	30	Complete RF tuning workflow with optimal parameter selection
Task 4: Model Comparison	15	Accurate comparison of all three approaches with AUC calculations
Task 5: Critical Analysis	10	Thoughtful analysis of interpretability and ensemble mechanisms
Total	**100**	

Expected Outputs Summary

By the end of this classwork, you should have:

- A hyperparameter tuning visualization showing performance across parameter space
- Three trained models: optimally-tuned single tree, bagged trees, and random forest
- Test set AUC scores for all three models demonstrating ensemble superiority
- ROC curves comparing discrimination ability across approaches
- Written analysis of the bias-variance-interpretability tradeoffs in ensemble methods

---

Key Concepts Reinforced

This classwork reinforces the third quarter of lecture content:

- **Hyperparameter tuning** systematically finds optimal model configurations using cross-validation
- **Bagging** reduces variance by averaging predictions from bootstrap-sampled trees
- **Random forests** combine bagging with feature randomization for decorrelated trees
- **Grid search** explores hyperparameter space to maximize validation performance
- **Ensemble methods** trade interpretability for improved predictive accuracy
- **Model comparison** requires consistent evaluation protocols across different approaches

---

This classwork covers approximately 25% of the lecture content, focusing on hyperparameter tuning, bagging, random forests, and systematic model comparison.