Statistical learning: tree based methods

Andrea Mauro

2025-04-18

Introduction

In this project, we analyze a dataset containing clinical measurements from 442 diabetic patients in order to investigate the relationship between several predictors and disease progression after one year (**progr**). The explanatory variables include age, sex, body mass index (**BMI**), blood pressure (**BP**), total cholesterol (**TC**), low-density lipoproteins (**LDL**), high-density lipoproteins (**HDL**), the ratio between total cholesterol and HDL (**TCH**), triglycerides level (**TG**, log-scaled), and blood glucose (**GC**). The regression models used to obtain predictions are: 1. **regression tree**, 2. **random forest**, 3. **boosted regression tree.**

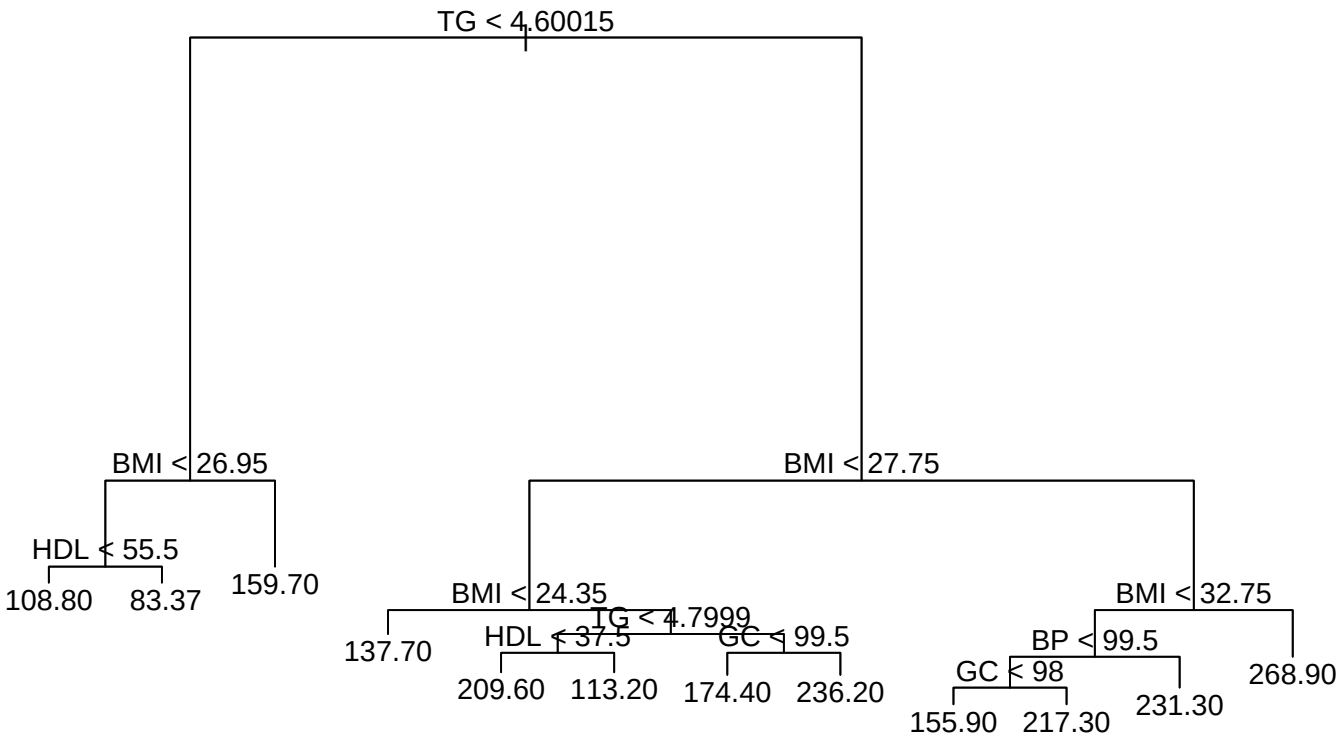
1. Regression tree

A regression tree was fitted to the full dataset by modeling the disease progression (`progr`) as a function of all available clinical predictors.

```
mf <- model.frame(progr ~ ., data = df)
tree_progr <- tree(progr ~ ., data = mf)
```

The `summary()` and the `plot()` of the of the regression tree model as follows:

```
Regression tree:
tree(formula = progr ~ ., data = mf)
Variables actually used in tree construction:
[1] "TG"  "BMI" "HDL" "GC"  "BP"
Number of terminal nodes:  12
Residual mean deviance:  2674 = 1150000 / 430
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-140.900  -35.830   -4.805    0.000   33.540  154.100
```



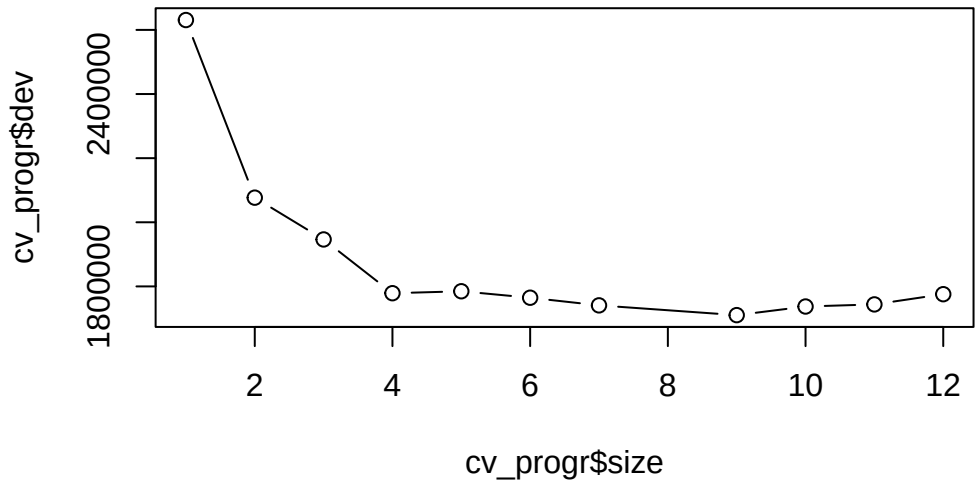
As said before a regression tree was first fitted to the entire dataset. The model selected five variables for splitting: triglycerides level (**TG**), body mass index (**BMI**), high-density lipoproteins (**HDL**), blood glucose (**GC**), and blood pressure (**BP**). The resulting tree contained **12** terminal nodes. The residual mean deviance was **2674**, indicating a moderate level of variability in the prediction errors

Pruning via cross-validation

A cross-validation procedure was performed to select the optimal size of the regression tree, using `cv.tree()`. A fixed seed (`set.seed(123)`) was set before pruning to ensure reproducibility.

```
set.seed(123)
cv_progr <- cv.tree(tree_progr)
```

As shown below, the deviance was plotted against the number of terminal nodes to guide pruning decisions



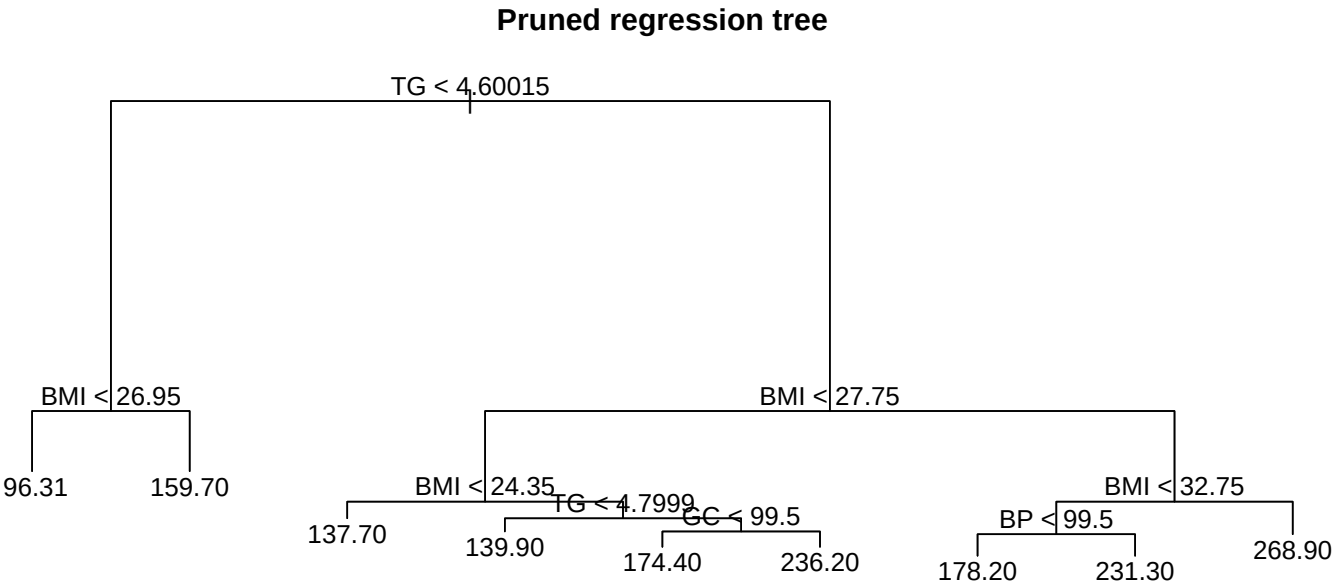
Tree-pruning

Based on the cross-validation results, the tree was pruned to the size minimizing the deviance, which in this case is **9**.

```
opt_size <- cv_progr$size[which.min(cv_progr$dev)]
pruned_progr <- prune.tree(tree_progr, best = opt_size)
```

Below we show the `summary()` and the `plot()` of the pruned-tree.

```
Regression tree:
snip.tree(tree = tree_progr, nodes = c(4L, 28L, 26L))
Variables actually used in tree construction:
[1] "TG"  "BMI" "GC"  "BP"
Number of terminal nodes:  9
Residual mean deviance:  2864 = 1240000 / 433
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-140.900  -37.310   -6.325    0.000   34.150  156.700
```



After pruning, the decision tree reduced its complexity from **12** to **9** terminal nodes. The set of variables used slightly changed, with **HDL** no longer appearing among the splits. Although the residual mean deviance increased from **2674** to **2864**, the pruned tree is simpler and potentially more generalizable. This slight loss in fit is acceptable to prevent overfitting and improve model interpretability.

2. Random forest

To fit the random forest model, we first selected the optimal number of predictors (`mtry`) to consider at each split. A grid search over all possible values of `mtry` was performed, evaluating models based on the out-of-bag (**OOB**) error. The value of `mtry` minimizing the **OOB** error was then selected for the final model fitting. Also in this case the results are obtained setting a seed(`seed(1)`) before implementing the model.

```r
set.seed(1)
n_pred <- ncol(df) - 1
# Grid with all possible mtry values
mtry_grid <- 1:n_pred
oob_errors <- numeric(length(mtry_grid))

# Testing each single values of the grid
for (i in seq_along(mtry_grid)) {
    rf_model <- randomForest(progr ~ ., data = df, ntree = 500,
        mtry = mtry_grid[i], importance = TRUE)
    oob_errors[i] <- min(rf_model$mse)
}

# Finding the best mtry value based on oob error
best_mtry_idx <- which.min(oob_errors)
best_mtry <- mtry_grid[best_mtry_idx]
```

```
Best mtry value: 2
```

```
Corresponding OOB value: 3201.44
```

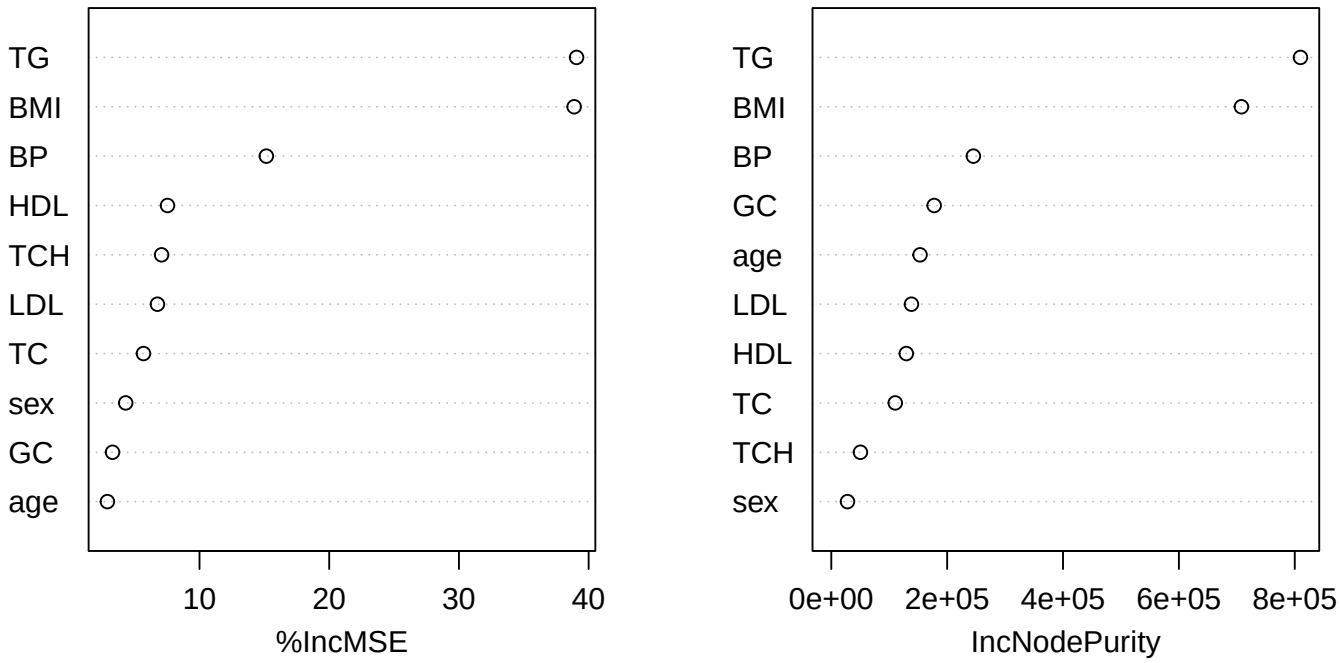
Model interpretation

Below we show the most important variables that contributed to the performance of the model.

```r
importance(rf_model)
```

```
      %IncMSE IncNodePurity
age  2.898188     153203.59
sex  4.310007      27974.86
BMI 38.884242     707919.41
BP  15.153027     245268.15
TC   5.680486     110412.44
LDL  6.755481     138354.66
HDL  7.533833     129520.50
TCH  7.076487      50341.45
TG  39.070101     809862.48
GC   3.294379     177606.34
```

```r
varImpPlot(rf_model)
```



The importance of each predictor was assessed using two measures: the mean decrease in MSE (%IncMSE) and the total decrease in node impurity (IncNodePurity). According to both metrics, **triglycerides (TG)** and **body mass index (BMI)**emerged as the most influential variables in predicting disease progression, followed by **blood pressure (BP)** and **blood glucose (GC)**. Other variables, including cholesterol-related measures (TC, LDL, HDL, TCH), and demographic features (age, sex), showed comparatively lower importance. These results suggest that metabolic and cardiovascular risk factors play a major role in the progression of diabetes-related complications over one year.

3. Boosted regression tree

To optimize the performance of the boosted regression trees, a grid search over different interaction depths (from 1 to 5) was performed. For each depth value, a 5-fold cross-validation was applied to identify the optimal number of boosting iterations (n.trees). The best model configuration was selected based on the minimum cross-validated error. For reproducibility reasons a seed was implemented.

```
set.seed(123)
depth_grid <- c(1, 2, 3, 4, 5)  #interaction.depth values to test

# Creating a list to save errors and optimal number of
# trees
results <- lapply(depth_grid, function(d) {
    model <- gbm(progr ~ ., data = df, distribution = "gaussian",
        n.trees = 1000, interaction.depth = d, shrinkage = 0.01,
        cv.folds = 5, verbose = FALSE)
    best_trees <- gbm.perf(model, method = "cv", plot.it = FALSE)
    list(error = min(model$cv.error), trees = best_trees)
})
```

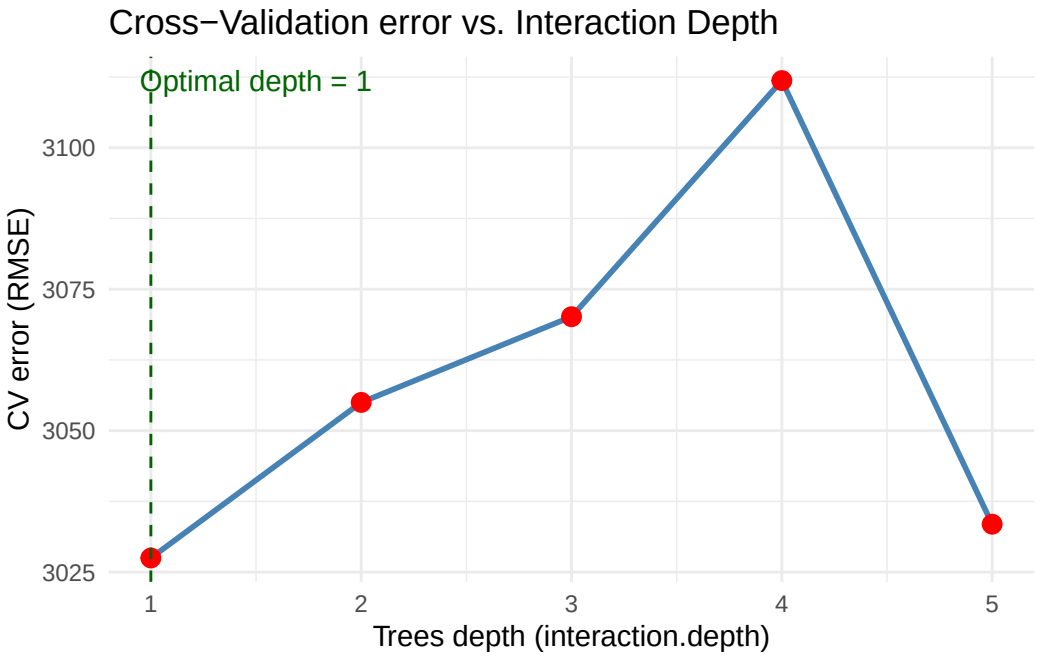
```r
# optimal depth
errors <- sapply(results, function(x) x$error)
best_depth <- depth_grid[which.min(errors)]

# optimal number of trees for the best depth
best_trees <- results[[which.min(errors)]]$trees
```

```
Best interaction.depth: 1
```

```
Best trees number: 889
```

The cross-validation procedure identified an optimal interaction depth of 1 and 889 boosting iterations. This indicates that relatively simple trees are sufficient to capture the main predictive patterns in the data. The choice of a shallow depth suggests that additive effects dominate the relationship between predictors and the response, while deeper interactions are less relevant in this setting

For a better representation is shown the cross-validation error as a function of the interaction depth

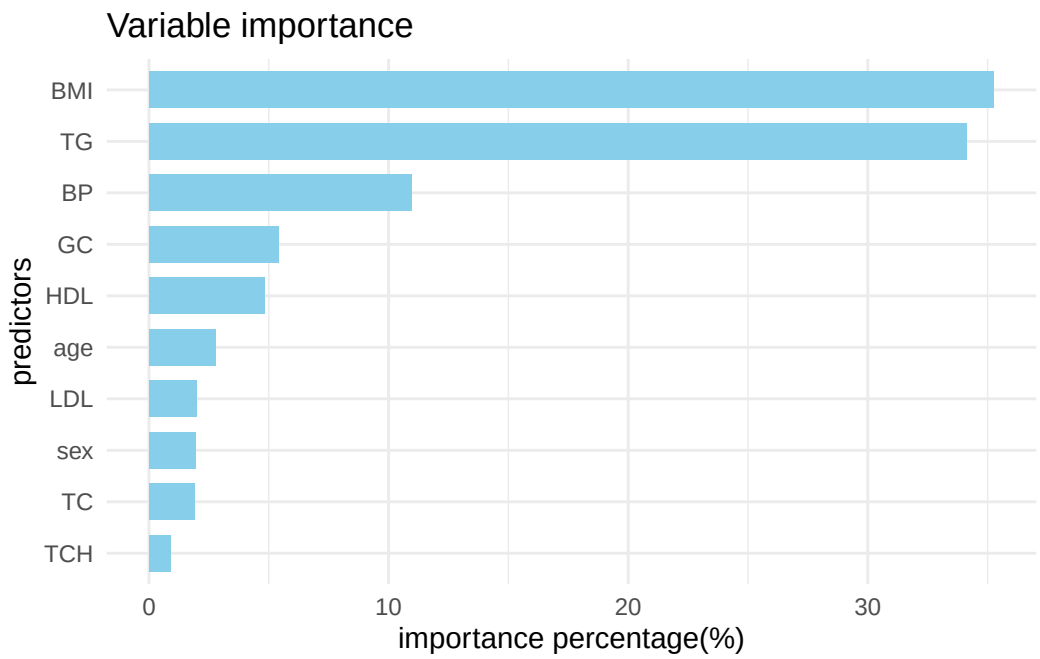


Final boosted model

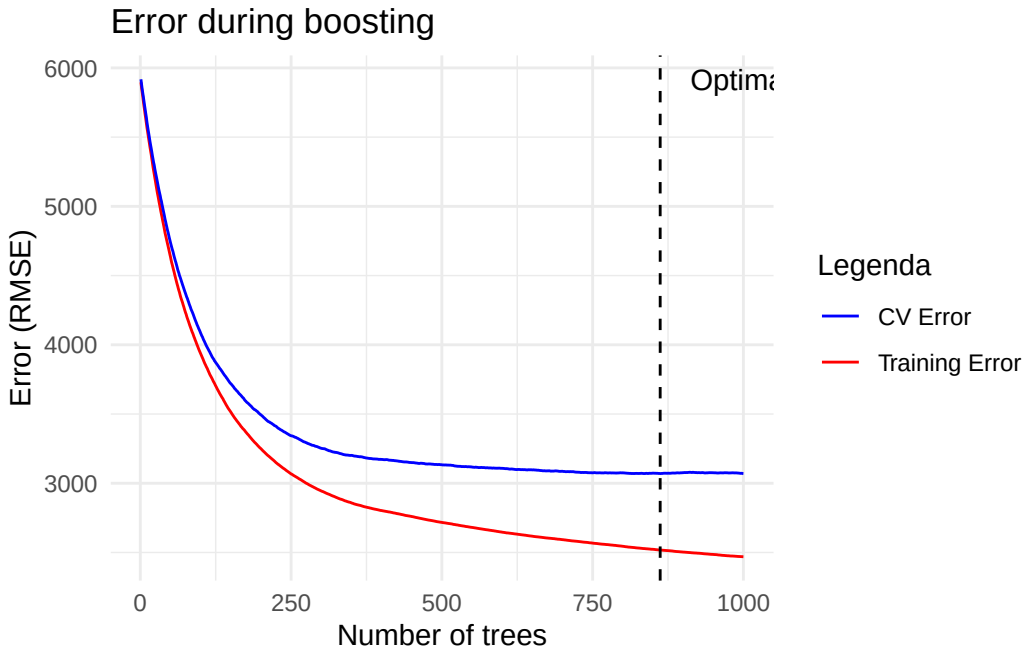
Based on the optimal interaction depth identified through cross-validation, we retrained the final boosting model and evaluated the relative importance of the predictors. The variable importance plot reveals that triglycerides (**TG**) and body mass index (**BMI**) are the most influential features in predicting disease progression, followed by glucose concentration (**GC**) and blood pressure (**BP**)

```r
set.seed(1)
# re-training the model with the optimal depth
final_model <- gbm(progr ~ ., data = df, distribution = "gaussian",
    n.trees = 1000, interaction.depth = best_depth, shrinkage = 0.01,
    cv.folds = 5, verbose = FALSE)
# variable importance
importance <- summary(final_model, plotit = FALSE)
```

```r
# Plot
ggplot(importance, aes(x = reorder(var, rel.inf), y = rel.inf)) +
    geom_col(fill = "skyblue", width = 0.7) + coord_flip() +
    labs(title = "Variable importance", x = "predictors", y = "importance percentage(%)") +
    theme_minimal()
```



The evolution of the training and cross-validation errors as a function of the number of boosting iterations is displayed. While the training error steadily decreases, the cross-validation error reaches its minimum at around 889 trees, after which it starts to increase slightly, suggesting potential overfitting beyond this point. Therefore, 889 iterations are selected as the optimal stopping point to balance model complexity and predictive performance.



Models comparison

In this section, we perform a model comparison using 5-fold cross-validation. The objective is to evaluate and compare the predictive performance of three different models: a pruned decision tree, a random forest, and a boosting model. For each model, we use the training data to build the model, using the optimized parameters found in the previous analysis (such as the tree size for the decision tree, the mtry parameter for the random forest, and the interaction depth for boosting), and evaluate the model's performance using Mean Squared Error (MSE) on the test set.

```
set.seed(123)
n <- nrow(df)
K <- 5
folds <- sample(rep(1:K, length.out = n))

# vectors to memorize errors
mse_tree <- numeric(K)
mse_rf <- numeric(K)
mse_boost <- numeric(K)

# optimal parameters alredy defined (mtry, depth, n.trees)
mtry_forest <- 2
best_n_trees <- best_trees

for (k in 1:K) {
    test_idx <- which(folds == k)
    train_idx <- setdiff(1:n, test_idx)

    # training and test splitting
    train_data <- df[train_idx, ]
    test_data <- df[test_idx, ]

    # 1. Decision Tree with Pruning
    tree_model <- tree(progr ~ ., data = train_data)
    cv_tree <- cv.tree(tree_model)
    opt_size <- cv_tree$size[which.min(cv_tree$dev)]
    pruned_tree <- prune.tree(tree_model, best = opt_size)  # Pruning
    preds_tree <- predict(pruned_tree, newdata = test_data)  # Prediction
    mse_tree[k] <- mean((test_data$progr - preds_tree)^2)

    # 2. Random Forest (with mtry alredy optimized)
    rf_model <- randomForest(progr ~ ., data = train_data, ntree = 500,
        mtry = mtry_forest, importance = TRUE)
    preds_rf <- predict(rf_model, newdata = test_data)
    mse_rf[k] <- mean((test_data$progr - preds_rf)^2)

    # 3. Boosting (with interaction.depth and n.trees
    # optimizee)
    boost_model <- gbm(progr ~ ., data = train_data, distribution = "gaussian",
        n.trees = best_n_trees, interaction.depth = best_depth,
        shrinkage = 0.01, verbose = FALSE)
    preds_boost <- predict(boost_model, newdata = test_data,
        n.trees = best_n_trees)
    mse_boost[k] <- mean((test_data$progr - preds_boost)^2)
```

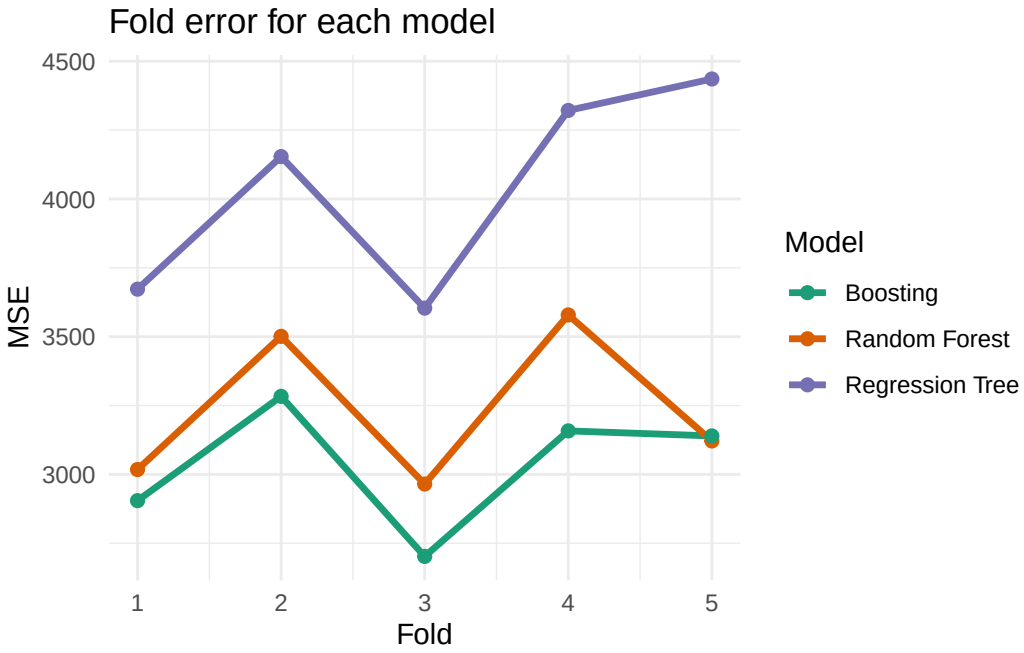
```r
}

# MSE for each model
mean_mse_tree <- mean(mse_tree)
mean_mse_rf <- mean(mse_rf)
mean_mse_boost <- mean(mse_boost)
```

```
Mean error Regression Tree (MSE): 4037.383
```

```
Mean error Random Forest (MSE): 3236.747
```

```
Mean error Boosting (MSE): 3037.45
```

The following plot illustrates the MSE across folds for Regression Tree, Random Forest, and Boosting models. Boosting consistently achieves the lowest error, outperforming both Random Forest and Decision Tree. Random Forest shows better predictive performance than Decision Tree, but still lags behind Boosting. Overall, ensemble methods, especially Boosting, provide a clear improvement over a single pruned tree.



After performing the 5-fold cross-validation, the results show that **boosting** achieved the lowest Mean Squared Error (MSE), making it the most accurate model among the three. The random forest model also performed well, with a lower MSE than the regression tree, indicating that the random forest's ensemble approach helps reduce variance and improves generalization. In contrast, the pruned decision tree exhibited a higher MSE, suggesting that, while it is simpler and interpretable, it is prone to overfitting when compared to the more complex models. These findings highlight the advantages of ensemble methods, especially boosting, in improving predictive accuracy over a single decision tree model.