**R Lab for Tree Based Algorithm**

**1. Required Libraries**

library(readr)

library(tidyverse)

library(caret)

library(tree)

library(rpart)

library(randomForest)

library(purrr)

**2. Load and Preprocess Data**

# Load data

business\_data <- read\_csv("business\_data.csv")

# Convert all columns to numeric explicitly

business\_data <- business\_data %>%

mutate(across(everything(), ~ .x %>% as.numeric()))

# Train-Test split

set.seed(123)

train\_idx <- createDataPartition(business\_data$Revenue, p = 0.75, list = FALSE)

train <- business\_data[train\_idx, ]

test <- business\_data[-train\_idx, ]

**3. Baseline Model: Mean Predictor**

mse\_mean <- mean((mean(train$Revenue) - test$Revenue)^2)

print(paste("Baseline MSE (mean predictor):", round(mse\_mean, 2)))

**4. Regression Tree Model (with Pruning)**

# Fit full tree

reg\_tree <- tree(Revenue ~ ., data = train)

cv\_rt <- cv.tree(reg\_tree)

# Choose best prune size

idx\_order <- order(cv\_rt$dev)

best\_size <- cv\_rt$size[idx\_order[1]]

second\_best <- cv\_rt$size[idx\_order[2]]

prune\_size <- if (best\_size == 1) second\_best else best\_size

# Prune and evaluate

if (prune\_size > 1) {

pruned\_tree <- prune.tree(reg\_tree, best = prune\_size)

yhat\_tree <- predict(pruned\_tree, newdata = test)

} else {

pruned\_tree <- reg\_tree

yhat\_tree <- predict(reg\_tree, newdata = test)

warning("Best prune size is 1; using unpruned tree.")

}

mse\_tree <- mean((yhat\_tree - test$Revenue)^2)

print(paste("Tree Model MSE:", round(mse\_tree, 2)))

# Plot trees

par(mfrow = c(1, 2))

plot(reg\_tree); text(reg\_tree, pretty = 0)

plot(pruned\_tree); text(pruned\_tree, pretty = 0)

par(mfrow = c(1, 1))

**5. Helper Function for MSE Reporting**

report\_mse <- function(name, y\_true, y\_pred) {

mse <- mean((y\_true - y\_pred)^2)

message(sprintf("%-25s MSE = %.3f", name, mse))

invisible(mse)

}

**6. Bagging with Cross-Validation for number of trees**

p <- ncol(train) - 1

b\_grid <- c(10, 50, 100, 200)

k <- 10

folds <- createFolds(train$Revenue, k = k)

# Cross-validation loop

bag\_cv <- tibble(b = b\_grid) %>%

mutate(cv\_rmse = map\_dbl(b, function(bn) {

mean(map\_dbl(folds, function(idx) {

mod <- randomForest(Revenue ~ ., data = train[-idx, ], mtry = p, ntree = bn)

sqrt(mean((train[idx, ]$Revenue - predict(mod, train[idx, ]))^2))

}))

}))

print(bag\_cv)

# Best ntree

best\_b <- bag\_cv %>% slice\_min(cv\_rmse) %>% pull(b)

message(sprintf("Best Bagging ntree = %d", best\_b))

# Plot

ggplot(bag\_cv, aes(x = b, y = cv\_rmse)) +

geom\_line() + geom\_point() +

labs(title = sprintf("%d-fold CV: Bagging RMSE vs #Trees", k),

x = "# Trees", y = "CV RMSE")

**7. Bagging: Tune number of trees using OOB error**

mtry\_grid <- seq(1, p)

folds <- createFolds(train$Revenue, k = 5)

cv\_mtry <- tibble(mtry = mtry\_grid) %>%

mutate(cv\_mse = map\_dbl(mtry, function(m) {

mean(map\_dbl(folds, function(idx) {

mod <- randomForest(Revenue ~ ., data = train[-idx, ], mtry = m, ntree = best\_b)

mean((train[idx, ]$Revenue - predict(mod, train[idx, ]))^2)

}))

}))

best\_mtry <- cv\_mtry %>% slice\_min(cv\_mse) %>% pull(mtry)

message("Best Bagging mtry = ", best\_mtry)

**8. Final Bagging Model**

final\_bag <- randomForest(

Revenue ~ ., data = train,

mtry = best\_mtry, ntree = best\_b, importance = TRUE

)

# OOB convergence

ggplot(tibble(tree = 1:best\_b, oob\_mse = final\_bag$mse),

aes(tree, oob\_mse)) +

geom\_line() + labs(title = "Bagging: OOB MSE vs Trees")

# Variable importance

varImpPlot(final\_bag, main = "Bagging: %IncMSE Importance")

print(head(importance(final\_bag) %>%

as.data.frame() %>%

rownames\_to\_column("Variable") %>%

arrange(desc(`%IncMSE`)), 5))

# Test MSE

report\_mse("Final Bagging Model", test$Revenue, predict(final\_bag, test))

**9. Random Forest: Tune number of predictors**

rf\_grid <- expand.grid(mtry = 1:p)

ctrl\_oob <- trainControl(method = "oob")

rf\_caret <- train(

Revenue ~ ., data = train, method = "rf",

metric = "RMSE", tuneGrid = rf\_grid, trControl = ctrl\_oob,

ntree = 200

)

best\_mtry\_rf <- rf\_caret$bestTune$mtry

plot(rf\_caret, main = "RF OOB RMSE vs mtry")

**10. Random Forest: Tune number of trees via CV**

ntree\_grid <- c(50, 100, 150, 200, 250, 500, 1000)

cv\_results <- map\_dfr(ntree\_grid, function(nt) {

rf\_mod <- train(

Revenue ~ ., data = train, method = "rf",

trControl = trainControl(method = "cv", number = 10),

tuneGrid = data.frame(mtry = best\_mtry\_rf),

ntree = nt

)

res <- rf\_mod$results %>% filter(mtry == best\_mtry\_rf)

tibble(ntree = nt, mtry = best\_mtry\_rf, RMSE = res$RMSE)

})

ggplot(cv\_results, aes(ntree, RMSE)) +

geom\_line() + geom\_point() +

labs(title = "RF: CV RMSE vs #Trees", x = "# Trees", y = "CV RMSE")

**11. Final Random Forest Model**

final\_rf <- randomForest(

Revenue ~ ., data = train,

mtry = best\_mtry\_rf,

ntree = cv\_results$ntree[which.min(cv\_results$RMSE)],

importance = TRUE

)

# OOB convergence

ggplot(tibble(tree = 1:final\_rf$ntree, oob\_mse = final\_rf$mse),

aes(tree, oob\_mse)) +

geom\_line() + labs(title = "RF: OOB MSE vs Trees")

# Variable importance

varImpPlot(final\_rf, main = "RF: %IncMSE Importance")

print(head(importance(final\_rf) %>%

as.data.frame() %>%

rownames\_to\_column("Variable") %>%

arrange(desc(`%IncMSE`)), 5))

# Test MSE

report\_mse("Final RF Model", test$Revenue, predict(final\_rf, test))