STA 5936: Homework 2

Due January 23rd, 11:59pm

In this problem we use the abalone dataset available on Canvas. The dataset is about predicting the age of the abalone from its physical measurements. Use the first 7 variables as predictors and the 8-th as the response. Obtain all results using 10-fold cross-validation, which is computed as follows:

1. Generate a random permutation of the data. Use this random permutation to split the data into 10 disjoint subsets of almost equal size (417 or 418 observations each).
2. For each fold i ∈ {1, …, 10}, train the model on all data except subset i and test it on subset i, obtaining test error εi.
3. Obtain the final test error as the average of the 10 test errors εi obtained above. Here the errors εi could be the MSE, or the R2. Report results for the following models:
   1. Null model. Report the average train and test MSE of the null model that always predicts training ̄y (average training y). (1 point)
   2. OLS regression computed analytically by solving the following normal equations: (1 N XT X + λIp)β = 1 N XT Y where λ ∈ {0, 10−5, 10−4, 10−3, 10−2, 10−1} and N is the number of observa- tions in X. Report in a table the average training and test R2 and MSE, as well as their standard deviations obtained from the 10 folds. On the same graph, plot the average average training and test MSE vs λ as two separate curves. Also plot the average value of the logarithm of the determinant of 1 N XT X + λIp (average obtained from the 10 folds) vs λ. (3 points)
      * t
   3. Regression tree of maximum depth 1, 2, …. up to 7, for a total of 7 regression trees. On the same plot, plot the average training and test R2 vs the tree depth as two separate curves. On another plot, plot the average training and test MSE vs the tree depth, and show the null model MSE from a) as a horizontal line. (2 points)
      * t
   4. Random forest regression with 10, 30, 100 and 300 trees. Report the average training, OOB, and test R2 and MSE and their standard deviations in each case. On the same plot, plot the average training, OOB and test R2 vs the number of trees as three separate curves. How does the average OOB R2 compare to the test R2? (2 points)
      * t

df <- read.csv("C:/Users/Bryce/Downloads/STA 5635/HW2/abalone.csv", header = FALSE, sep = ",")

set.seed(0)  
  
n <- nrow(df)  
k <- 10  
  
RandPerm <- sample(n)  
  
# Split data  
foldSize <- rep(floor(n/k), k)  
  
foldSize[1:(n %% k)] <- foldSize[1:(n %% k)] + 1  
  
folds <- split(RandPerm, rep(1:k, foldSize))  
  
foldedData <- lapply(folds, function(indices) df[indices, ])  
  
# Check folds  
sapply(foldedData, nrow)

## 1 2 3 4 5 6 7 8 9 10   
## 418 418 418 418 418 418 418 417 417 417

library(Metrics)

## Warning: package 'Metrics' was built under R version 4.4.2

# Model Training for the Null Model  
trainMSE <- numeric(10)  
testMSE <- numeric(10)  
  
for (i in 1:10) {  
   
 testIndices <- folds[[i]]  
 trainIndices <- setdiff(1:nrow(df), testIndices)  
   
 trainData <- df[trainIndices, ]  
 testData <- df[testIndices, ]  
   
 yTrain <- trainData$V8   
 yTest <- testData$V8   
 yTrainMean <- mean(yTrain)  
   
 trainMSE[i] <- mse(yTrain, rep(yTrainMean, length(yTrain)))   
 testMSE[i] <- mse(yTest, rep(yTrainMean, length(yTest)))  
}  
  
averageTrainMSE <- mean(trainMSE)  
averageTestMSE <- mean(testMSE)  
  
cat("Average Training MSE (Null Model):", averageTrainMSE, "\n")

## Average Training MSE (Null Model): 10.39242

cat("Average Test MSE (Null Model):", averageTestMSE, "\n")

## Average Test MSE (Null Model): 10.40003

library(caret)

## Warning: package 'caret' was built under R version 4.4.2

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.4.2

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following objects are masked from 'package:Metrics':  
##   
## precision, recall

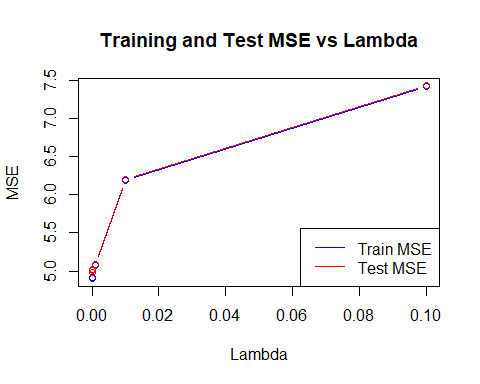
library(Matrix)

## Warning: package 'Matrix' was built under R version 4.4.2

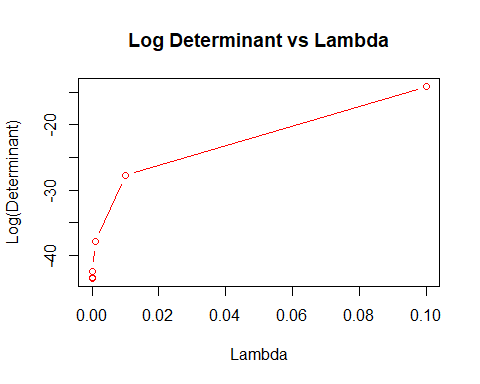
lambdas <- c(0, 10^(-5), 10^(-4), 10^(-3), 10^(-2), 10^(-1))  
  
trainMSE <- testMSE <- trainR2 <- testR2 <- logDet <- matrix(NA, nrow = length(lambdas), ncol = k)  
  
# cv and fitting  
for (lambdaIDX in 1:length(lambdas)) {  
 lambda <- lambdas[lambdaIDX]  
   
 for (foldIDX in 1:k) {  
   
 testData <- foldedData[[foldIDX]]  
 trainData <- do.call(rbind, foldedData[-foldIDX])  
   
 xTrain <- cbind(1, as.matrix(trainData[, -ncol(df)]))  
 yTrain <- trainData[, ncol(df)]  
 xTest <- cbind(1, as.matrix(testData[, -ncol(df)]))  
 yTest <- testData[, ncol(df)]  
   
 # Normal equation  
 XTX <- t(xTrain) %\*% xTrain / nrow(xTrain)  
 XTXlambda <- XTX + lambda \* diag(ncol(xTrain))  
 XTY <- t(xTrain) %\*% yTrain / nrow(xTrain)  
   
 beta <- solve(XTXlambda, XTY)  
   
 # Train pred and perf  
 yTrainPred <- xTrain %\*% beta  
 trainMSE[lambdaIDX, foldIDX] <- mse(yTrain, yTrainPred)  
 trainR2[lambdaIDX, foldIDX] <- R2(yTrain, yTrainPred)  
   
 # Test pred and perf  
 yTestPred <- xTest %\*% beta  
 testMSE[lambdaIDX, foldIDX] <- mse(yTest, yTestPred)  
 testR2[lambdaIDX, foldIDX] <- R2(yTest, yTestPred)  
   
 # Log det  
 logDet[lambdaIDX, foldIDX] <- determinant(XTXlambda, logarithm = TRUE)$modulus  
 }  
}  
  
# avg and std dev  
trainMSEavg <- apply(trainMSE, 1, mean)  
testMSEavg <- apply(testMSE, 1, mean)  
trainMSEsd <- apply(trainMSE, 1, sd)  
testMSEsd <- apply(testMSE, 1, sd)  
  
trainR2avg <- apply(trainR2, 1, mean)  
testR2avg <- apply(testR2, 1, mean)  
  
logDetavg <- apply(logDet, 1, mean)  
  
results <- data.frame(  
 Lambda = lambdas,  
 TrainR2avg = trainR2avg,  
 TestR2avg = testR2avg,  
 TrainMSEavg = trainMSEavg,  
 TestMSEavg = testMSEavg,  
 TrainMSEsd = trainMSEsd,  
 TestMSEsd = testMSEsd,  
 LogDetavg = logDetavg  
)  
  
# Print results  
print(results)

## Lambda TrainR2avg TestR2avg TrainMSEavg TestMSEavg TrainMSEsd TestMSEsd  
## 1 0e+00 0.5280324 0.5216713 4.904990 5.016222 0.07380305 0.8108113  
## 2 1e-05 0.5280203 0.5221116 4.905133 5.009041 0.07376744 0.7887751  
## 3 1e-04 0.5273338 0.5241585 4.913411 4.977585 0.07182748 0.6703977  
## 4 1e-03 0.5154296 0.5162472 5.071642 5.091793 0.06024019 0.5649924  
## 5 1e-02 0.4095029 0.4115330 6.193869 6.206721 0.06974351 0.7529775  
## 6 1e-01 0.3123559 0.3149953 7.420396 7.429738 0.09432220 0.9853778  
## LogDetavg  
## 1 -43.53125  
## 2 -43.40539  
## 3 -42.45541  
## 4 -37.90852  
## 5 -27.73346  
## 6 -14.12231

plot(lambdas, trainMSEavg, type = "b", col = "blue", xlab = "Lambda", ylab = "MSE", main = "Training and Test MSE vs Lambda")  
lines(lambdas, testMSEavg, type = "b", col = "red")  
legend("bottomright", legend = c("Train MSE", "Test MSE"), col = c("blue", "red"), lty = 1)



plot(lambdas, logDetavg, type = "b", col = "red", xlab = "Lambda", ylab = "Log(Determinant)", main = "Log Determinant vs Lambda")



library(rpart)  
library(dplyr)

## Warning: package 'dplyr' was built under R version 4.4.2

##   
## Attaching package: 'dplyr'

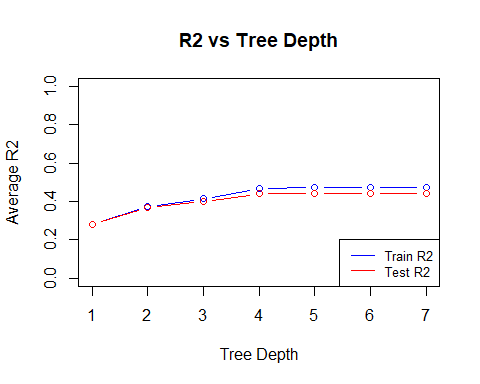
## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

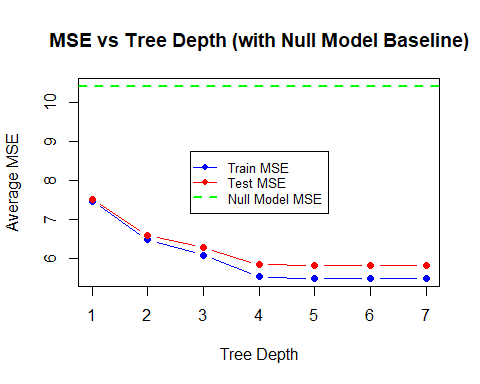
k <- length(foldedData)   
  
trainMSE <- matrix(NA, nrow = length(1:7), ncol = k)  
testMSE <- matrix(NA, nrow = length(1:7), ncol = k)  
trainR2 <- matrix(NA, nrow = length(1:7), ncol = k)  
testR2 <- matrix(NA, nrow = length(1:7), ncol = k)  
  
  
for (depth in 1:7) {  
   
 for (foldIDX in 1:k) {  
   
 testData <- foldedData[[foldIDX]]  
 trainData <- do.call(rbind, foldedData[-foldIDX])  
   
 # Tree  
 tree\_model <- rpart(  
 V8 ~ .,   
 data = trainData,  
 method = "anova",  
 control = rpart.control(maxdepth = depth)  
 )  
   
 # Train preds  
 yTrainPred <- predict(tree\_model, trainData)  
 yTrain <- trainData$V8  
 trainMSE[depth, foldIDX] <- mse(yTrain, yTrainPred)  
 trainR2[depth, foldIDX] <- R2(yTrain, yTrainPred)  
   
 # Test preds  
 yTestPred <- predict(tree\_model, testData)  
 yTest <- testData$V8  
 testMSE[depth, foldIDX] <- mse(yTest, yTestPred)  
 testR2[depth, foldIDX] <- R2(yTest, yTestPred)  
 }  
}  
  
trainMSEavg <- apply(trainMSE, 1, mean)  
testMSEavg <- apply(testMSE, 1, mean)  
trainR2avg <- apply(trainR2, 1, mean)  
testR2avg <- apply(testR2, 1, mean)  
  
results <- data.frame(  
 Depth = 1:7,  
 TrainMSEavg = trainMSEavg,  
 TestMSEavg = testMSEavg,  
 TrainR2avg = trainR2avg,  
 TestR2avg = testR2avg)   
  
print(results)

## Depth TrainMSEavg TestMSEavg TrainR2avg TestR2avg  
## 1 1 7.458863 7.514351 0.2822763 0.2788266  
## 2 2 6.487685 6.579579 0.3757303 0.3696510  
## 3 3 6.088498 6.273050 0.4141116 0.3984290  
## 4 4 5.529737 5.837267 0.4678603 0.4399638  
## 5 5 5.491621 5.832157 0.4715848 0.4400065  
## 6 6 5.491621 5.832157 0.4715848 0.4400065  
## 7 7 5.491621 5.832157 0.4715848 0.4400065

plot(1:7, trainR2avg, type = "b", col = "blue", ylim = c(0, 1),  
 xlab = "Tree Depth", ylab = "Average R2", main = "R2 vs Tree Depth")  
  
lines(1:7, testR2avg, type = "b", col = "red")  
  
legend("bottomright", legend = c("Train R2", "Test R2"), col = c("blue", "red"), lty = 1, cex = 0.8)



plot(1:7, trainMSEavg, type = "b", col = "blue", pch = 16, ylim = range(c(trainMSEavg, testMSEavg, averageTestMSE)),  
 xlab = "Tree Depth", ylab = "Average MSE", main = "MSE vs Tree Depth (with Null Model Baseline)")  
  
lines(1:7, testMSEavg, type = "b", col = "red", pch = 16)  
  
abline(h = averageTestMSE, col = "green", lty = 2, lwd = 2)  
  
legend("center", legend = c("Train MSE", "Test MSE", "Null Model MSE"),  
 col = c("blue", "red", "green"), lty = c(1, 1, 2), pch = c(16, 16, NA), lwd = c(1, 1, 2), cex = 0.8)



library(randomForest)

## Warning: package 'randomForest' was built under R version 4.4.2

## randomForest 4.7-1.2

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

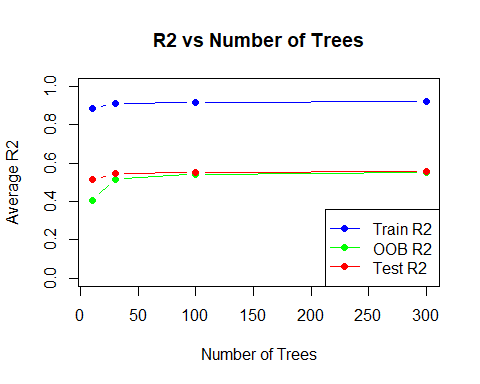
## The following object is masked from 'package:dplyr':  
##   
## combine

## The following object is masked from 'package:ggplot2':  
##   
## margin

treeCount <- c(10, 30, 100, 300)  
trainR2 <- testR2 <- oobR2 <- trainMSE <- testMSE <- oobMSE <- matrix(NA, nrow = length(treeCount), ncol = k)  
  
for (nTrees in treeCount) {  
 for (i in 1:k) {  
   
 testIndices <- folds[[i]]  
 trainIndices <- setdiff(1:nrow(df), testIndices)  
   
 trainData <- df[trainIndices, ]  
 testData <- df[testIndices, ]  
   
 rfModel <- randomForest(V8 ~ ., data = trainData, ntree = nTrees)  
   
 # Train preds  
 yTrainPred <- predict(rfModel, newdata = trainData)  
 trainR2[nTrees == treeCount, i] <- R2(trainData$V8, yTrainPred)  
 trainMSE[nTrees == treeCount, i] <- mse(trainData$V8, yTrainPred)  
   
 # OOB  
 oobR2[nTrees == treeCount, i] <- rfModel$rsq[length(rfModel$rsq)]   
 oobMSE[nTrees == treeCount, i] <- mean((rfModel$y - rfModel$predicted)^2)   
   
 # test preds  
 yTestPred <- predict(rfModel, newdata = testData)  
 testR2[nTrees == treeCount, i] <- R2(testData$V8, yTestPred)  
 testMSE[nTrees == treeCount, i] <- mse(testData$V8, yTestPred)  
 }  
}  
  
# avg and std dev  
trainR2avg <- apply(trainR2, 1, mean)  
testR2avg <- apply(testR2, 1, mean)  
oobR2avg <- apply(oobR2, 1, mean)  
  
trainMSEavg <- apply(trainMSE, 1, mean)  
testMSEavg <- apply(testMSE, 1, mean)  
oobMSEavg <- apply(oobMSE, 1, mean)  
  
trainR2sd <- apply(trainR2, 1, sd)  
testR2sd <- apply(testR2, 1, sd)  
oobR2sd <- apply(oobR2, 1, sd)  
  
trainMSEsd <- apply(trainMSE, 1, sd)  
testMSEsd <- apply(testMSE, 1, sd)  
oobMSEsd <- apply(oobMSE, 1, sd)  
  
results <- data.frame(  
 Trees = treeCount,  
 TrainR2avg = trainR2avg,  
 OOBR2avg = oobR2avg,  
 TestR2avg = testR2avg,  
 TrainMSEavg = trainMSEavg,  
 OOBMSEavg = oobMSEavg,  
 TestMSEavg = testMSEavg,  
 TrainR2sd = trainR2sd,  
 OOBR2sd = oobR2sd,  
 TestR2sd = testR2sd,  
 TrainMSEsd = trainMSEsd,  
 OOBMSEsd = oobMSEsd,  
 TestMSEsd = testMSEsd  
)  
  
print(results)

## Trees TrainR2avg OOBR2avg TestR2avg TrainMSEavg OOBMSEavg TestMSEavg  
## 1 10 0.8865525 0.4072991 0.5156062 1.286288 NA 5.052605  
## 2 30 0.9094320 0.5160888 0.5452158 1.101971 5.028956 4.731017  
## 3 100 0.9182881 0.5424960 0.5519620 1.033360 4.754497 4.663469  
## 4 300 0.9203430 0.5502045 0.5552729 1.017506 4.674341 4.626219  
## TrainR2sd OOBR2sd TestR2sd TrainMSEsd OOBMSEsd TestMSEsd  
## 1 0.0028843811 0.011475763 0.02301231 0.03691528 NA 0.3970279  
## 2 0.0012358099 0.004527119 0.02394146 0.01343241 0.06515333 0.3484426  
## 3 0.0011129689 0.004065406 0.01747700 0.01256756 0.05775123 0.3859143  
## 4 0.0005341447 0.002480689 0.02029529 0.01098546 0.04103449 0.3608691

plot(treeCount, trainR2avg, type = "b", col = "blue", ylim = c(0, 1), pch = 16,  
 xlab = "Number of Trees", ylab = "Average R2", main = "R2 vs Number of Trees")  
lines(treeCount, oobR2avg, type = "b", col = "green", pch = 16)  
lines(treeCount, testR2avg, type = "b", col = "red", pch = 16)  
legend("bottomright", legend = c("Train R2", "OOB R2", "Test R2"), col = c("blue", "green", "red"), lty = 1, pch = 16)



ymin <- min(c(trainMSEavg, oobMSEavg, testMSEavg), na.rm = TRUE)  
ymax <- max(c(trainMSEavg, oobMSEavg, testMSEavg), na.rm = TRUE)  
  
plot(treeCount, trainMSEavg, type = "b", col = "blue", pch = 16,  
 xlab = "Number of Trees", ylab = "Average MSE", main = "MSE vs Number of Trees",  
 ylim = c(ymin, ymax))   
lines(treeCount, oobMSEavg, type = "b", col = "green", pch = 16)  
lines(treeCount, testMSEavg, type = "b", col = "red", pch = 16)  
legend("center", legend = c("Train MSE", "OOB MSE", "Test MSE"), col = c("blue", "green", "red"), lty = 1, pch = 16)

