STAT 542 / CS 598: Homework 4

Fall 2019, by Regan Chan (ttchan2)

Due: Monday, Oct 14 by 11:59 PM Pacific Time

Contents

Question 1 [70 Points] Tuning Random Forests in Virtual Twins 1
Question 2 [30 Points] Second Step in Virtual Twins 3

Question 1 [70 Points] Tuning Random Forests in Virtual Twins

Personalized medicine draws a lot of attention in medical research. The goal of personalized medicine is to make a tailored decision for each patient, such that his/her clinical outcome can be optimized. Let's consider data modified from the SIDES method. In this dataset, 470 patients and 13 variables are observed. You can download the data from our website. The variables are listed below.

- `Health`: health outcome (larger the better)
- `THERAPY`: 1 for active treatment, 0 for the control treatment
- `TIMFIRST`: Time from first sepsis-organ fail to start drug
- `AGE`: Patient age in years
- `BLLPLAT`: Baseline local platelets
- `blSOFA`: Sum of baseline sofa score (cardiovascular, hematology, hepatorenal, and respiration scores)
- `BLLCREAT`: Base creatinine
- `ORGANNUM`: Number of baseline organ failures
- `PRAPACHE`: Pre-infusion apache-ii score
- `BLGCS`: Base GLASGOW coma scale score
- `BLIL6`: Baseline serum IL-6 concentration
- `BLADL`: Baseline activity of daily living score
- `BLLBILI`: Baseline local bilirubin
- `BEST`: The true best treatment suggested by Doctors. **You should not use this variable when fitting the model!**

For each patient, sepsis was observed during their hospital stay. Hence, they need to choose one of the two treatments (indicated by variable `THERAPY`) to prevent further adverse events. After the treatment, their health outcome (`health`) were measured, with a larger value being the better outcome. However, since treatments were assigned randomly, we are not able to suggest better treatment for a new patient. A strategy called Virtual Twins was proposed by Foster et al. (2011) to tackle this problem. We consider a simpler version of the method. We fit two random forests to model the outcome `health`: one model uses all patients who received treatment 1, and another model for all patients who received treatment 0. Denote these two models as $\widehat{f}_1(x)$ and $\widehat{f}_0(x)$, respectively. When a new patient arrives, we use both models to predict the outcomes and see which model gives a better health status. We will suggest the treatment label associated with the model that gives a larger prediction value. In other words, for a new x^*, we compare $\widehat{f}_1(x^*)$ and $\widehat{f}_0(x^*)$ and suggest the better lable. The goal for this question is to select tuning parameters for random forest such that it will suggest the best treatment for a patient. Perform the following:

- Randomly split the data into 75% for training and 25% for testing.

```
sepsis <- read.csv("Sepsis.csv")
n <- nrow(sepsis)
set.seed(0)

genData <- function() {
  trainIdx <<- sample(1:n, n * 0.75)
```

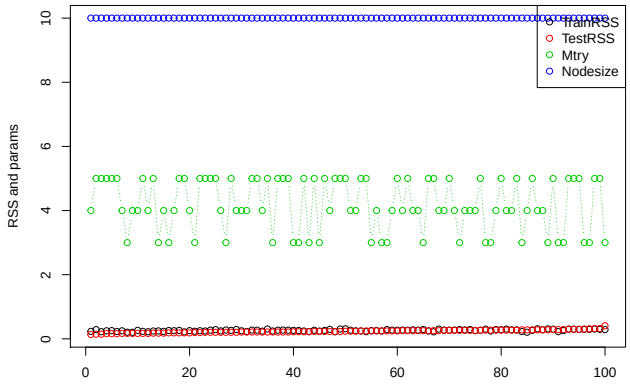
```
  trainSet <<- sepsis[trainIdx, ]
  m <<- ncol(trainSet)
  testSet <<- sepsis[-trainIdx,]
}
```

- For the training data, fit the virtual twins model and then use the testing data to suggest the best treatment.
  - You should not use the variable `BEST` when fitting the models

```
library(randomForest)
set.seed(0)
getRfModel <- function(...) {
  treatment0 <- randomForest(Health ~ .-THERAPY-BEST, data=trainSet[trainSet$THERAPY == 0, ], ...)
  health0 <- predict(treatment0, newdata=trainSet)
  treatment1 <- randomForest(Health ~ .-THERAPY-BEST, data=trainSet[trainSet$THERAPY == 1, ], ...)
  health1 <- predict(treatment1, newdata=trainSet)

  prediction.train <- as.numeric(health1 > health0)
  trainErr <- sum((prediction.train - trainSet$BEST)^2)/nrow(trainSet)

  health0 <- predict(treatment0, newdata=testSet)
  health1 <- predict(treatment1, newdata=testSet)
  prediction.test <- as.numeric(health1 > health0)
  testErr <- sum((prediction.test - testSet$BEST)^2)/nrow(testSet)
  return(list(trainErr=trainErr, testErr=testErr, train=prediction.train, test=prediction.test))
}
getBestErr <- function(...) {
  p <<- m-2  # -1 Because therapy and BEST are removed
  mybest.TrainErr <- Inf
  for(mtry in c(3, 4, 5)) {
    for(nodesize in c(3, 5, 10)) {
      newErr <- getRfModel(mtry=mtry, nodesize=nodesize)
      if(newErr$trainErr < mybest.TrainErr) {
        mybest.TrainErr <- newErr$trainErr
        mybest.TestErr <- newErr$testErr
        mybest.Mtry <- mtry
        mybest.Nodesize <- nodesize
      }
    }
  }
  return(c(mybest.TrainErr, mybest.TestErr, mybest.Mtry, mybest.Nodesize))
}

genAndTest <- function(...) {
  genData()
  getBestErr()
}

library(parallel)
results <- data.frame(t(mcmapply(genAndTest, 1:100, mc.cores=8)))
```

- Pick three different `mtry` values and three different `nodesize`, leave all other tuning parameters as default
- After predicting the best treatment in the testing data, compare it to the truth `BEST`

```
names(results) <- c("TrainRSS", "TestRSS", "Mtry", "Nodesize")
matplot(results[order(results$TestRSS),], type="b", col=1:4, pch=1, ylab="RSS and params")
legend("topright", legend=names(results), col=1:4, pch=1)
```



- Repeat this entire process 100 times and average the prediction errors

```
mean(results$TestRSS)
```

```
## [1] 0.2338983
```

Ans: Average test RSS is 0.231. Because the values are either 1 or 0, each error term is either 1 or 0. The RSS therefore reflects the accuracy in percentage, hence the test accuracy is 76.9%

- Summarize your results, including the model performance and the effect of tuning parameters. Intuitively demonstrate them.

Ans: By default of regression scenario, randomForest picks mtry of p/3 and nodesize of 5. I want to know if these are underestimates or overestimates so I tried values that are centered around the defaults. If I look at just the extremes (where errors are lowest and highest) lowest test error was obtained from using Mtry=p/2 and Nodesize=10, so that's what I will use in the next question

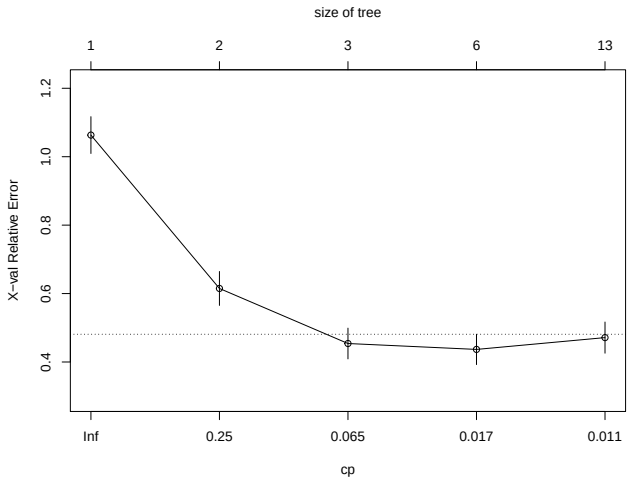
Question 2 [30 Points] Second Step in Virtual Twins

The second step in a virtual twins model is to use a single tree model (CART) to describe the choice of the best treatment. Perform the following: * Based on your optimal tuning parameter, fit the Virtual Twins model described in Question 1. Again, you should not use the `BEST` variable. * For each subject, obtain the predicted best treatment of the training data itself

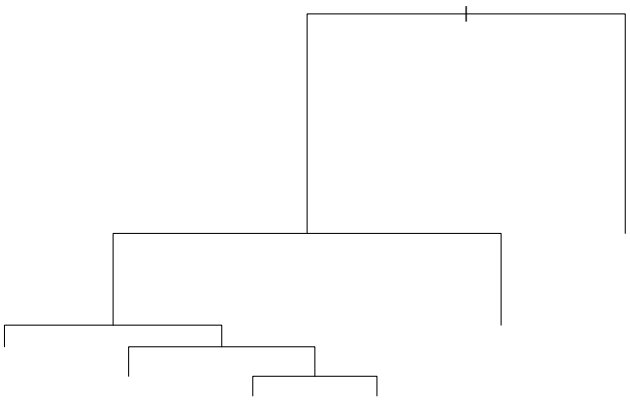
```
library(randomForest)
genData()
set.seed(0)
mymodel <- getRfModel(mtry=13, nodesize=10)
data <- cbind(within(trainSet, rm(BEST), rm(THERAPY)), Y=as.factor(mymodel$train))

library(rpart)
rfModel <- rpart(Y ~ ., data=data)
```

- Treating the label of best treatment as the outcome, and fit a single tree model to predict it. Be careful which variables should be removed from this model fitting.
- Consider tuning the tree model using the cost-complexity tuning.

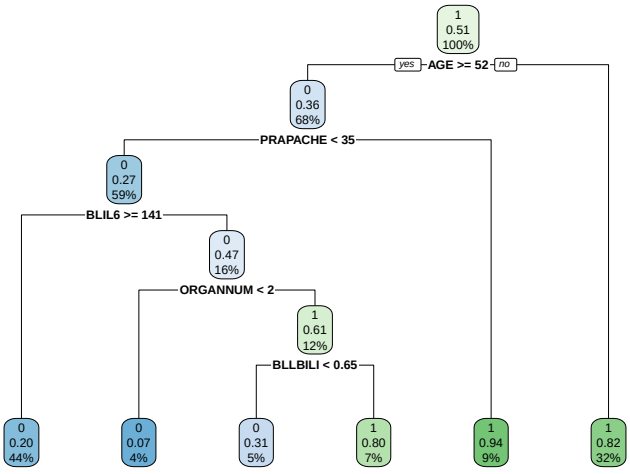
```
library(ggplot2)
library(rpart.plot)
library(tree)
plotcp(rfModel)
```




```
cptable <- data.frame(rfModel$cptable)
opt <- which.min(cptable$xerror)
prunedModel <- prune(rfModel, mean(cptable$CP[(opt-1):opt]))
plot(prunedModel)
```



```
rpart.plot(prunedModel)
```



Ans: According to the CP plot, a max split of 6 yielded lowest cross validated error