STAT 542 / CS 598: Homework 4

Fall 2019, by Paul Nel (paulnel2)

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

Contents

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

Set up of data sets

```
set.seed(101)
require(randomForest)
mydata = read.csv('Sepsis.csv')
n = nrow(mydata)
train.id = sample(1:n, round(n*0.75,0))
train.data = mydata[train.id,-15]
test.data = mydata[-train.id,-15]
```

Prediction and comparison functions

```
get_prediction <- function(therapy, mtry, nodesize){
  model.data = train.data[train.data[,"THERAPY"] == therapy,]
  rforest = randomForest(model.data$Health ~ .-THERAPY , data=model.data, mtry=mtry, nodesize=nodesize)
  return (predict(rforest, test.data[,-2]))
}

get_treatment <- function(mtry, nodesize){
  pred.0 = get_prediction(0, mtry, nodesize)
  pred.1 = get_prediction(1, mtry, nodesize)
  return (pred.1>pred.0)
}

get_accuracy <- function(prediction, truth) {
  mean(prediction == truth)
}
```

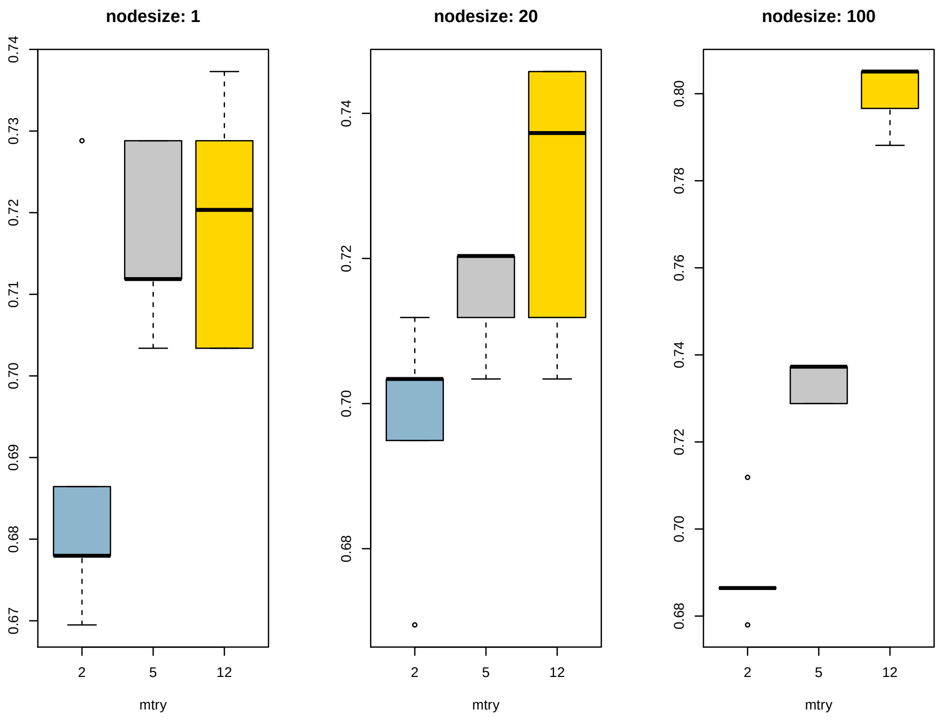
Iteration loop

```
reps = 5
mtries = c(2,5,12)
nodesizes = c(1,20,100)
zeros = rep(0, length(mtries)*length(nodesizes)*reps)
accuracies.tot =array(zeros, dim=c(reps,length(mtries),length(nodesizes)))
for (i in 1:reps){
  accuracies = matrix(rep(0, length(mtries)*length(nodesizes)),ncol=length(mtries))
  for (j in 1:length(mtries)){
    for (k in 1:length(nodesizes)){
      prediction = get_treatment(mtry=mtries[j], nodesize=nodesizes[k])
      accuracies.tot[i,k,j] = get_accuracy(prediction, mydata[-train.id, "BEST"])
    }
  }
}
```

This leads to the following results with rows representing different `nodesize` and the coloumns different `mtry` values.

Table 1: Model performance for combinations of mtry and nodesize

	2	5	12
1	0.6881356	0.7169492	0.7186441
20	0.6966102	0.7152542	0.7288136
100	0.6898305	0.7338983	0.8000000



Intuition

Normally when one has highly correlated data, it helps to reduce the number of predictors (`mtry`) available for selection at each split. When all predictors are used (12 in this case if we ignore `THERAPY`) this process is similar to simple bagging. The result of the above analysis would suggest that the predictors are largely uncorrelated since the accuracy improves as we allow more predictors into the selection at each split, up to 12.

The parameter `nodesize` determines the minimum size of nodes, in other words it determines the depth of each tree. For small values we can expect very deep trees. In this example I vary this from a minimum of `1` (largest possible tree) to `100`. This parameter has a significant impact for larger values of `mtry` suggesting that smaller trees results in better performance on the test set.

The paramters chosen will be `mtry = 13` and `nodesize = 100`

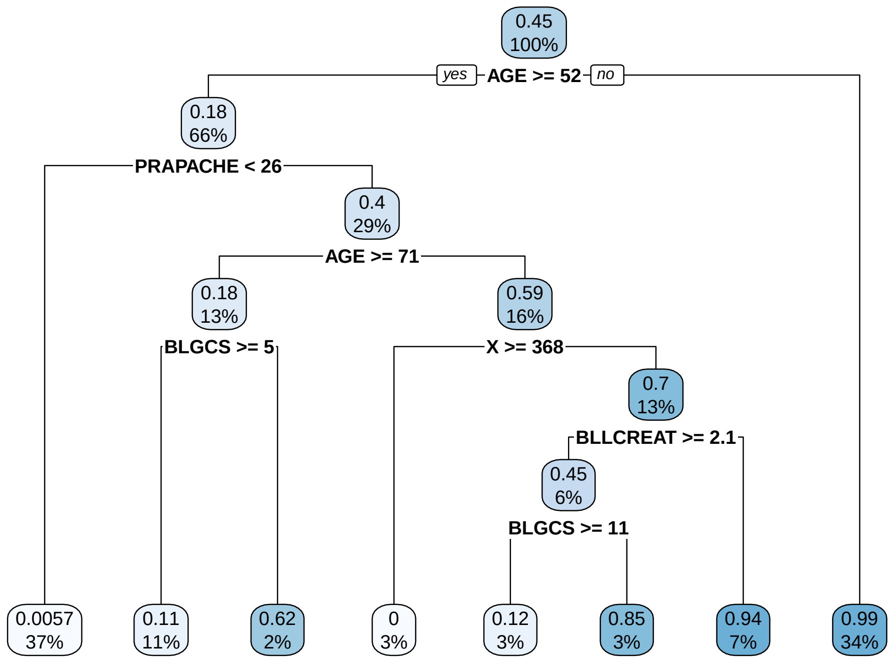
Question 2 [30 Points] Second Step in Virtual Twins

Generate predictions on all data based on the best parameters from Question 1.

```r
all_data = mydata[,-15]

get_all_prediction <- function(therapy, mtry, nodesize){
  model.data = all_data[all_data[,"THERAPY"] == therapy,]
  rforest = randomForest(model.data$Health ~ .-THERAPY , data=model.data, mtry=mtry, nodesize=nodesize)
  return (predict(rforest, all_data[,-2]))
}

get_all_treatment <- function(mtry, nodesize){
  pred.0 = get_all_prediction(0, mtry, nodesize)
  pred.1 = get_all_prediction(1, mtry, nodesize)
  return (ifelse(pred.1>pred.0, 1,0))
}

all_data['PRED'] = get_all_treatment(mtry=12, nodesize=100)
```

Build a single regression tree using the predicted values `all_data['PRED]` as the proposed treatment.

```r
require(rpart)
require(rpart.plot)
train.all_data = all_data[,-c(2,3)]
rtree = rpart(train.all_data$PRED~., data=train.all_data)
par(mfrow=c(1,1))
rpart.plot(rtree)
```


0.45
100%
yes
AGE >= 52
no
0.18
66%
PRAPACHE < 26
0.4
29%
AGE >= 71
0.18
13%
BLGCS >= 5
0.59
16%
X >= 368
0.7
13%
BLLCREAT >= 2.1
0.45
6%
BLGCS >= 11
0.0057
37%
0.11
11%
0.62
2%
0
3%
0.12
3%
0.85
3%
0.94
7%
0.99
34%