STAT 542 / CS 598: Homework 4

Fall 2019, by John Moran (jfmoran2)

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

Contents

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.
- 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
  - Pick three different `mtry` values and three different `nodesize`, leave all other tuning parameters as default

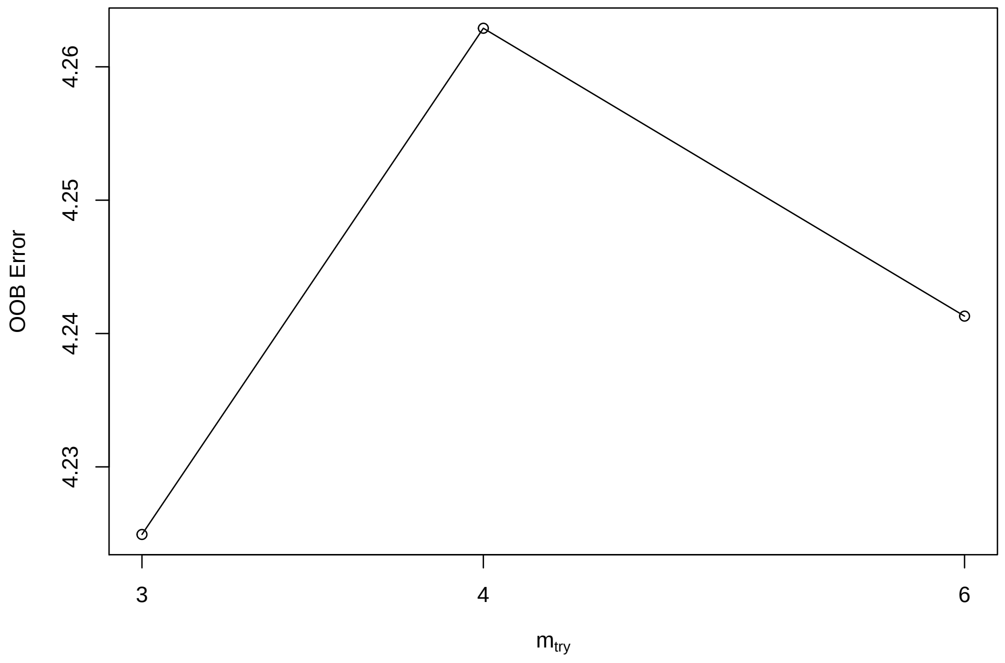
BEGIN STUDENT ANSWER:

The default value for **mtry** for regression is: "number_of_parameters/3", which in this case, p is either 13 if we count "Therapy", or 12 if we don't. In either case, a good guess at **mtry** is 4. However, R provides a function in the randomForest library *tuneRF* which will calculate the best **mtry** parameters based on the data. So I used *tuneRF* to calculate the best **mtry** values in this case.

The default **nodesize** is 5, so since there is no function provided to tune **nodesize**, I used the default value +/- 2 as candidate values.

```r
library(randomForest)
library(knitr)

set.seed(5)

sepsis.data <- read.csv(file="Sepsis.csv",header=TRUE, sep=",")

n.rows <- nrow(sepsis.data)
n.cols <- ncol(sepsis.data)

tune.data <- sepsis.data[, !colnames(sepsis.data) %in% c("BEST", "X")]

# tuning before deciding on values for mtry
tune.mtry <- tuneRF(tune.data[,-1], tune.data[,1], ntreeTry=400,stepFactor=1.5)
```

```
## mtry = 4  OOB error = 4.262892
## Searching left ...
## mtry = 3     OOB error = 4.224933
## 0.008904527 0.05
## Searching right ...
## mtry = 6     OOB error = 4.241304
## 0.005063956 0.05
```



```r
# val is in first column
tune.mtry <- tune.mtry[,1]
```

```r
print("Using tuneRF, the following mtry values were suggested: ")
```

```
## [1] "Using tuneRF, the following mtry values were suggested: "
```

```r
print(as.character(tune.mtry))
```

```
## [1] "3" "4" "6"
```

```r
mtry.vals <- as.numeric(tune.mtry)
nodesize.vals <- c(3,5,7)
```

- After predicting the best treatment in the testing data, compare it to the truth BEST
- Repeat this entire process 100 times and average the prediction errors

```r
iterations <- 100
rowcount <- 1

predictions <- data.frame(matrix(0, nrow=length(mtry.vals), ncol = length(nodesize.vals)))
colnames(predictions) <- paste("nodesize", nodesize.vals, sep="")
rownames(predictions) <- paste("mtry", mtry.vals, sep="")

for (mtry.idx in 1:length(mtry.vals)) {
  for (nodesize.idx in 1:length(nodesize.vals)) {
    for (j in 1:iterations) {

      # 75% of the data for training, 25% for testing
      train.ind <- sample(1:n.rows, size=round(n.rows*0.75))

      train.data <- sepsis.data[train.ind,]
      train.X <- train.data[, !colnames(train.data) %in% c("BEST", "X")]

      #select the test data
      test.data <- sepsis.data[-train.ind,]
      test.X <- test.data[, !colnames(test.data) %in% c("BEST","THERAPY", "X")]
      test.Y <- test.data[, "BEST"]

      therapy.0.ind <- which(train.X$THERAPY == 0)
      therapy.col <- which(colnames(train.X)=="THERAPY")

      train.X.0 <- train.X[therapy.0.ind, -therapy.col]
      train.X.1 <- train.X[-therapy.0.ind, -therapy.col]

      rfModel.0 <- randomForest(Health ~ ., data=train.X.0,
                                mtry=mtry.vals[mtry.idx],
                                nodesize=nodesize.vals[nodesize.idx])

      rfModel.1 <- randomForest(Health ~ ., data=data.frame(train.X.1),
                                mtry=mtry.vals[mtry.idx],
                                nodesize=nodesize.vals[nodesize.idx])

      # predict therapy model based on highest health score
      pred.0 <- predict(rfModel.0, test.X)
      pred.1 <- predict(rfModel.1, test.X)

      best.health <- ifelse(pred.0 > pred.1, 0, 1)
      correct <- sum(test.Y-best.health == 0)
      pred.accuracy <- correct/length(test.Y)

      predictions[mtry.idx, nodesize.idx] <- predictions[mtry.idx, nodesize.idx] + pred.accuracy

    }
  }
}
```

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

```
# divide the sum of predictions by number of iterations
predictions <- predictions/iterations

# get maximum value in matrix
min.indices <- which(predictions == max(predictions), arr.ind = TRUE)
best.mtry <- mtry.vals[min.indices[1]]
best.nodesize <- nodesize.vals[min.indices[2]]

# make ktable here with predictions
kable(predictions, caption="Accuracy predictions for different mtry, nodesize")
```

Table 1: Accuracy predictions for different mtry, nodesize

	nodesize3	nodesize5	nodesize7
mtry3	0.7640678	0.7612712	0.7792373
mtry4	0.7657627	0.7613559	0.7825424
mtry6	0.7738983	0.7762712	0.7853390

```
print(sprintf("best mtry: %d best nodesize: %d", best.mtry, best.nodesize))
```

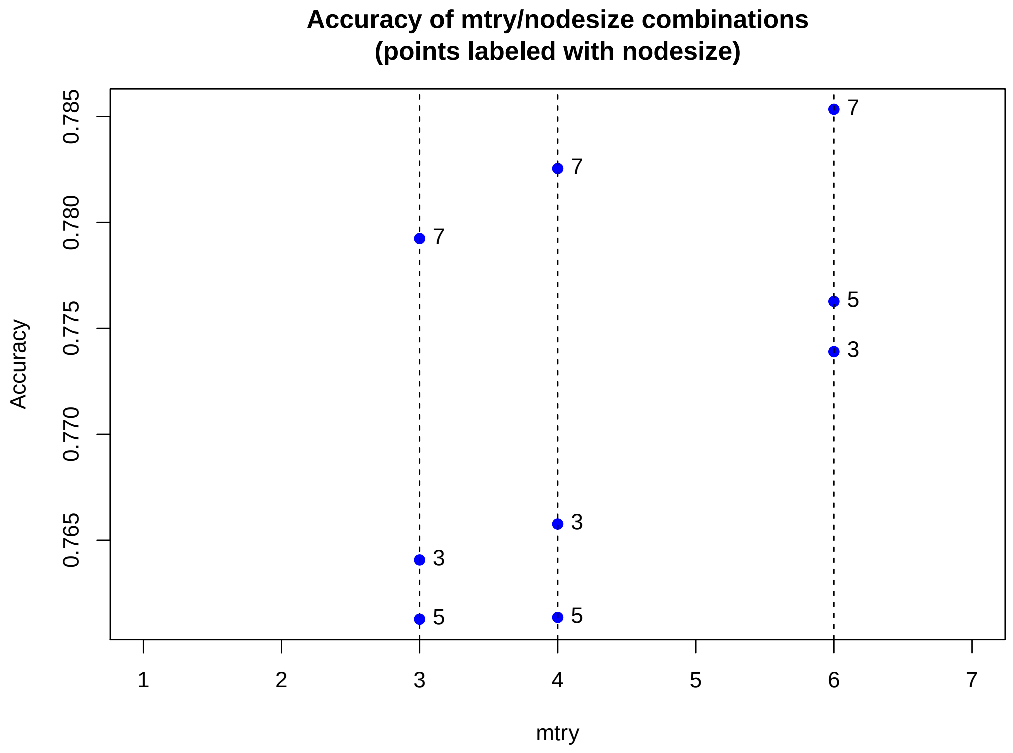
```
## [1] "best mtry: 6 best nodesize: 7"
```

```
print(sprintf("best accuracy: %f", predictions[min.indices[1],min.indices[2]]))
```

```
## [1] "best accuracy: 0.785339"
```

To help visualize the relationships between our values of **mtry** and **nodesize** versus **accuracy**, I produced a scatterplot where X-axis is **mtry**, Y-axis is **accuracy**, and the value of **nodesize** is printed as a label to the right of the data point. In all cases for this random seed and these parameter options, **nodesize** = 7 and **mtry** = 6 produces the highest accuracy. These values will be used in Question 2 as the optimal paarameters to run the Virtual Twin model on the full data set.

```
#unroll the data to plot it
p1 <- matrix(nrow=9, ncol=3)
colnames(p1) <- c("accuracy", "mtry", "nodesize")

p1[,"accuracy"] <- as.vector(as.matrix(t(predictions)))
p1[,"mtry"] <- rep(mtry.vals, each=3)
p1[,"nodesize"] <- rep(nodesize.vals, 3)

plot(accuracy~mtry,
     xlim=c(1,7),
     xlab = 'mtry',
     ylab = 'Accuracy',
     main = 'Accuracy of mtry/nodesize combinations\n(points labeled with nodesize)',
     pch=19,
     col="blue",
     data=p1)

text(accuracy~mtry, labels=nodesize, data=p1, pos=4, col="black")
abline(v=mtry.vals, lty=2)
```



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.

```
# use the whole dataset less the appropriate columns to remove
train.data <- sepsis.data[, !colnames(sepsis.data) %in% c("BEST", "X")]

therapy.0.ind <- which(train.data$THERAPY == 0)
therapy.col <- which(colnames(train.data)=="THERAPY")

train.0 <- train.data[therapy.0.ind, -therapy.col]
train.1 <- train.data[-therapy.0.ind, -therapy.col]

rfModel.0 <- randomForest(Health ~ ., data=train.0,
                          mtry=best.mtry,
                          nodesize=best.nodesize)
```

```r
rfModel.1 <- randomForest(Health ~ ., data=data.frame(train.1),
                          mtry=best.mtry,
                          nodesize=best.nodesize)
```

- For each subject, obtain the predicted best treatment of the training data itself

```r
# predict therapy model based on highest health score
pred.0 <- predict(rfModel.0, train.data)
pred.1 <- predict(rfModel.1, train.data)
best.health <- ifelse(pred.0 > pred.1, 0, 1)

correct <- sum(sepsis.data$BEST-best.health == 0)
pred.accuracy <- correct/length(best.health)
print(sprintf("Twin Model with optimal paramaters, full data set, accuracy = %f", pred.accuracy))
```

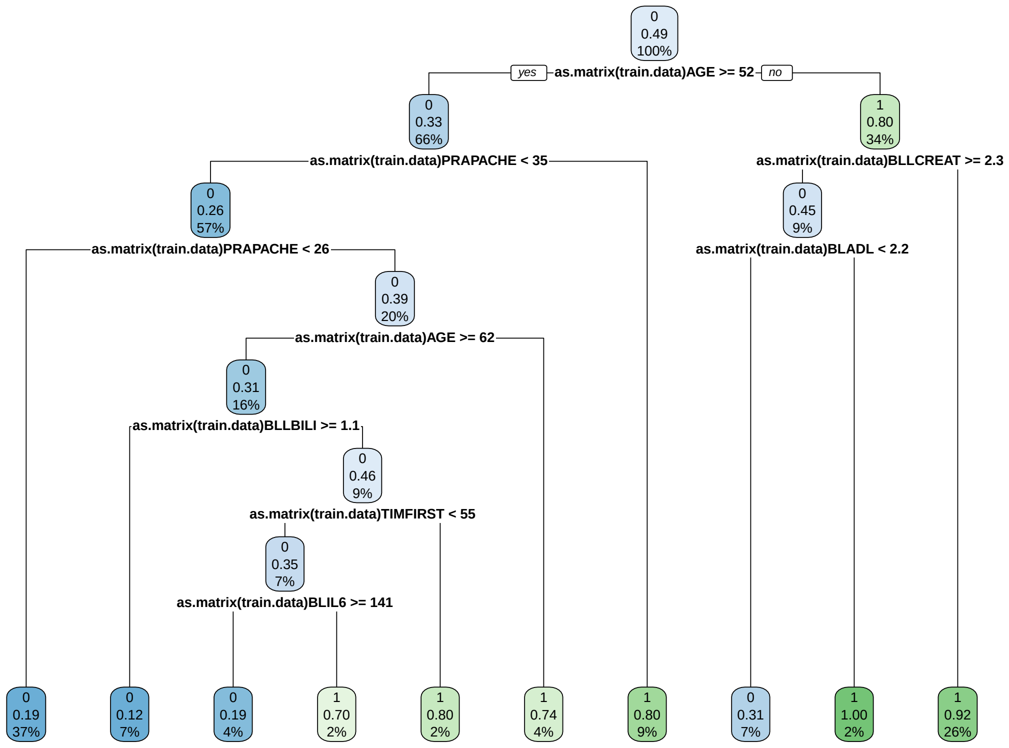
```
## [1] "Twin Model with optimal paramaters, full data set, accuracy = 0.751064"
```

- 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.

```r
library(rpart)
library(rpart.plot)
library(tree)

# also remove HEALTH
train.data <- sepsis.data[, !colnames(sepsis.data) %in% c("BEST", "X", "THERAPY","HEALTH")]

cart.model = rpart(best.health ~ as.matrix(train.data), method="class")
cart.pred <- rpart.predict(cart.model, train.data, type="class")
cart.pred <- as.numeric(as.character(cart.pred))

correct <- sum(sepsis.data$BEST-cart.pred == 0)
cart.pred.accuracy <- correct/length(cart.pred)
print(sprintf("CART Model, full tree, full data set, accuracy = %f", cart.pred.accuracy))
```

```
## [1] "CART Model, full tree, full data set, accuracy = 0.838298"
```

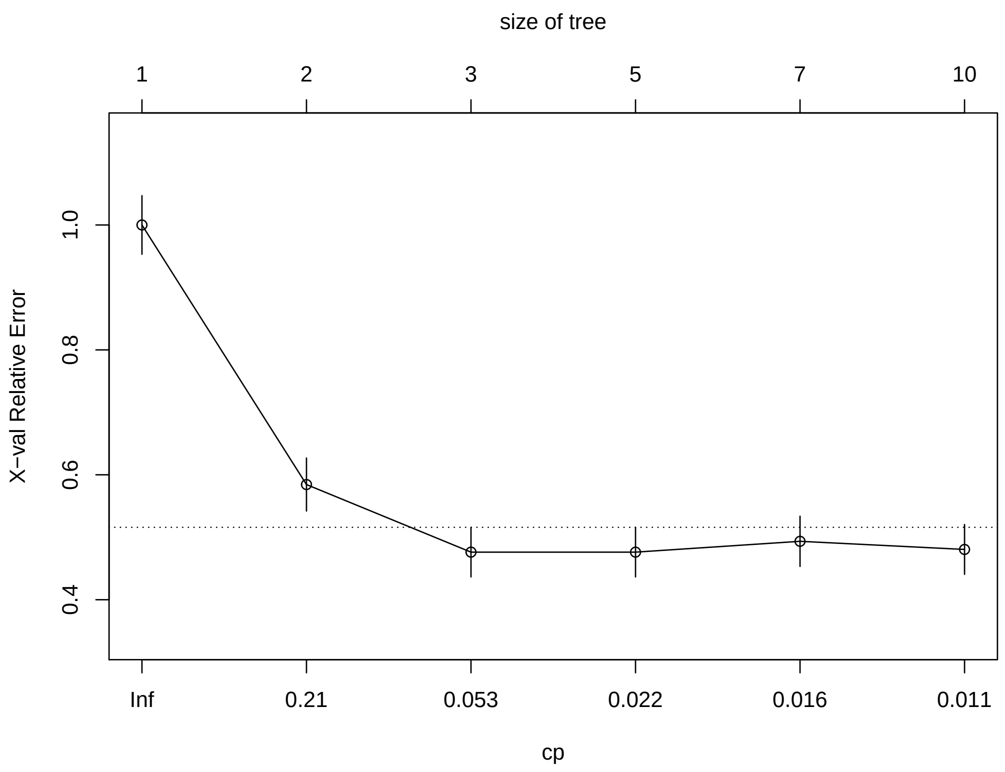
```r
par(mfrow=c(1,1))
rpart.plot(cart.model)
```


0
0.49
100%
yes
as.matrix(train.data)AGE >= 52
no
0
0.33
66%
1
0.80
34%
as.matrix(train.data)PRAPACHE < 35
as.matrix(train.data)BLLCREAT >= 2.3
0
0.26
57%
0
0.45
9%
as.matrix(train.data)PRAPACHE < 26
as.matrix(train.data)BLADL < 2.2
0
0.39
20%
as.matrix(train.data)AGE >= 62
0
0.31
16%
as.matrix(train.data)BLLBILI >= 1.1
0
0.46
9%
as.matrix(train.data)TIMFIRST < 55
0
0.35
7%
as.matrix(train.data)BLIL6 >= 141
0
0.19
37%
0
0.12
7%
0
0.19
4%
1
0.70
2%
1
0.80
2%
1
0.74
4%
1
0.80
9%
0
0.31
7%
1
1.00
2%
1
0.92
26%

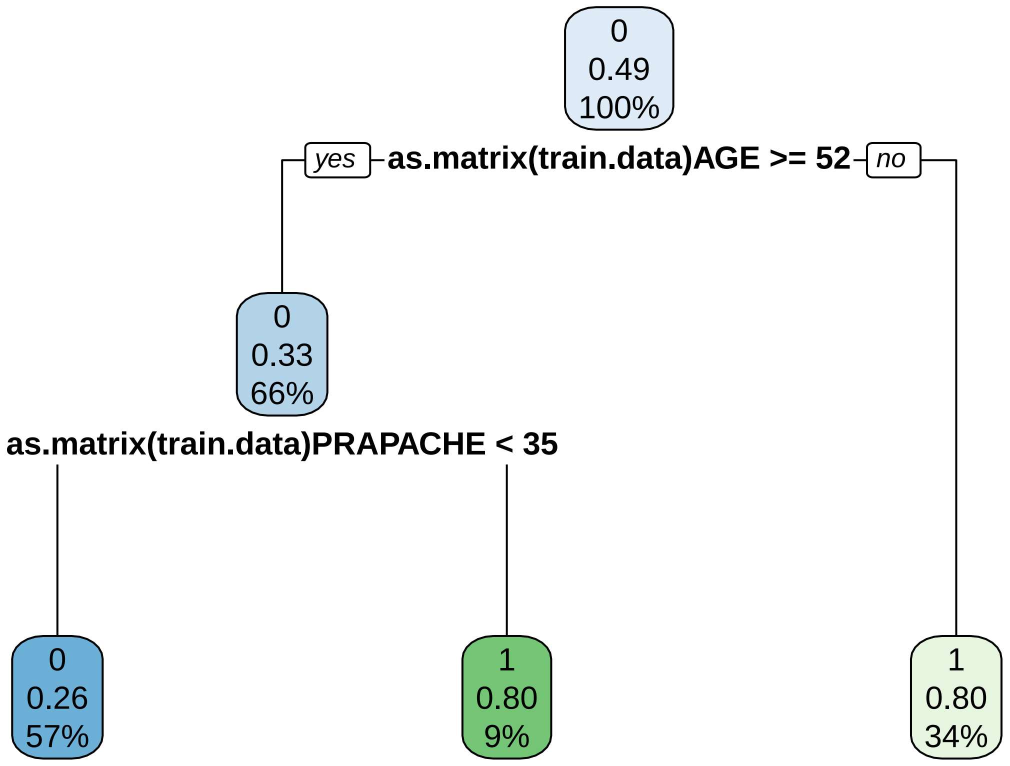
- Consider tuning the tree model using the cost-complexity tuning.

Use the CP table returned by the CART Model of the full tree to find the best value to prune the tree at by looking at the minimum value of **xerror** in the CP table to locate a 1SE CP value to pass to the *prune* function.

```
# Prune the tree
plotcp(cart.model)
```



```r
# Find the row with minimum value of "xerror" in the cptable
opt = which.min(cart.model$cptable[, "xerror"])

# get the two values of CP to average
cp.val1 <- cart.model$cptable[opt, "CP"]
cp.val2 <- cart.model$cptable[opt-1, "CP"]

# pick any CP value between the two, so just average them
good.cp.val <- (cp.val1 + cp.val2)/2

# Prune tree with the CP found above
pruned.model = prune(cart.model, cp = good.cp.val)
rpart.plot(pruned.model)
```


0
0.49
100%
yes
as.matrix(train.data)AGE >= 52
no
0
0.33
66%
as.matrix(train.data)PRAPACHE < 35
0
0.26
57%
1
0.80
9%
1
0.80
34%

```
prune.pred <- rpart.predict(pruned.model, train.data, type="class")
prune.pred <- as.numeric(as.character(prune.pred))

correct <- sum(sepsis.data$BEST-prune.pred == 0)
prune.pred.accuracy <- correct/length(prune.pred)
print(sprintf("CART Model, pruned tree, full data set, accuracy = %f", prune.pred.accuracy))
```

```
## [1] "CART Model, pruned tree, full data set, accuracy = 0.861702"
```

The pruned tree using cost-complexity tuning produces a better result (accuracy = 86.2%) than the full tree (accuracy = 83.8%).