Assignment 5 - Decision Trees Response

Timothy Lee

November 9, 2016

For this assignment we will be using data from the Assistments Intelligent Tutoring system. This system gives students hints based on how they perform on math problems.

#Install & call libraries

```
#Install needed packages
#I've done this, so eval = FALSE so it doesn't run each time
install.packages("party")
install.packages("rpart")
```

```
#Import rpart
library(rpart)

#Import party!
library(party)
```

Part I

```
D1 <- read.csv("intelligent_tutor.csv")
```

Codebook * id - student id * prior_prob_count - The number of problems a student has done in the system prior to the surrent session
* score - The score the student achieved in the current session
* hints - The number of hints the student requested in the current session
* hint.y - Whether or not the student asked for hints in the current session
* complete - Whether or not the student completed the current session
* action - The action suggested by the system to a teacher about a given student based on their performance

##Classification Tree - CART First we will build a classification tree to predict which students ask a teacher for help, which start a new session, or which give up, based on whether or not the student completed a session `(D1$complete)` and whether or not they asked for hints `(D1$hint.y)`.

```
#Build the classification tree
c.tree <- rpart(action ~ hint.y + complete, method = "class", data = D1) #Notice the standard R notion

#Look at the error of this tree
printcp(c.tree)
```

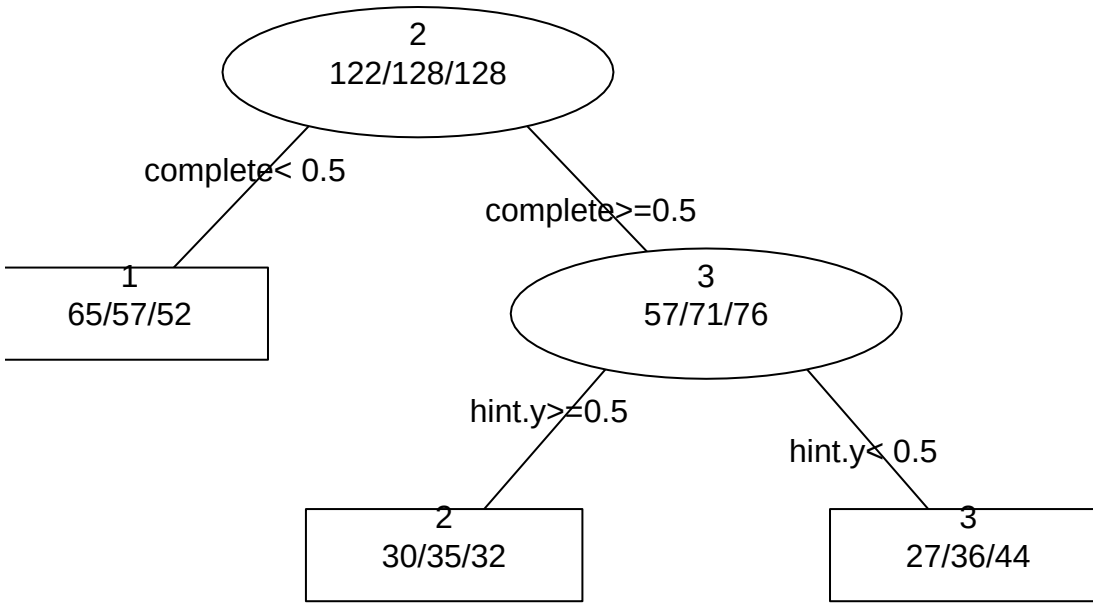
```
##
## Classification tree:
## rpart(formula = action ~ hint.y + complete, data = D1, method = "class")
##
## Variables actually used in tree construction:
## [1] complete hint.y
##
```

```
## Root node error: 250/378 = 0.66138
##
## n= 378
##
##      CP nsplit rel error xerror     xstd
## 1 0.052      0     1.000  1.128 0.033851
## 2 0.012      1     0.948  1.012 0.036587
## 3 0.010      2     0.936  0.988 0.037008
```

```
#Plot the tree
#This saves it to a file called tree.ps
#post(c.tree, filename = "tree.ps", title = "Session Completion Action: 1 - Ask teacher, 2 - Start new

#This prints our tree to the Rmd file, which is what we want
post(c.tree, filename = "", title = "Session Completion Action: 1 - Ask teacher, 2 - Start new session,
```

sion Completion Action: 1 − Ask teacher, 2 − Start new session, 3 − Giv



Printcp output gives error rates at each split:

- Root node error - the number of misclassifications/total n in the top-level (root) node, before any splits have occurred
- CP - looks like the marginal decrease in `rel error` with each split
- nsplit - number of splits in the tree
- rel error - When multiplied by the root node error, gives the resubstitution error rate (error rate computed against the training sample, so probably crap)
- xerror - When multiplied by the root node error, gives the cross-validated error rate using 10-fold CV.
- xstd - ??? - probably K-fold error rate standard error?

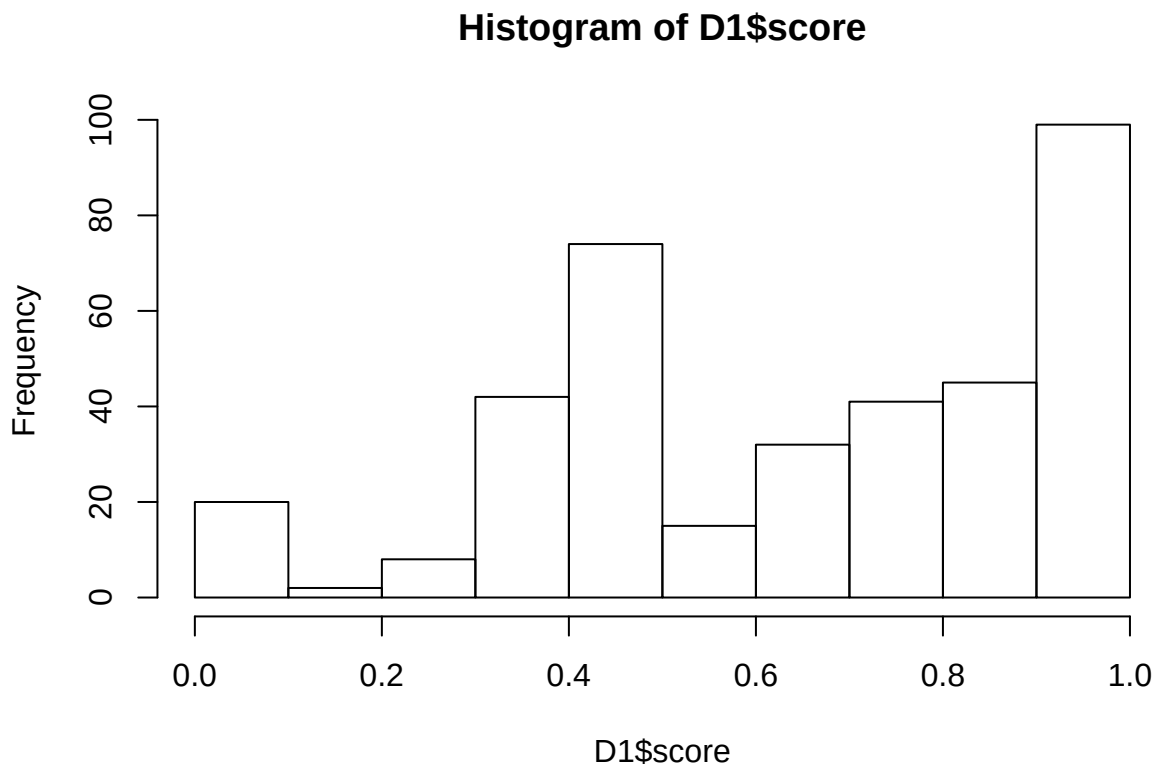
Part II

#Regression Tree

We want to see if we can build a decision tree to help teachers decide which students to follow up with, based on students' performance in Assistments. We will create three groups ("teacher should intervene", "teacher should monitor student progress" and "no action") based on students' previous use of the system and how many hints they use. To do this we will be building a decision tree using the "party" package. The party package builds decision trees based on a set of statistical stopping rules.

#Visualize our outcome variable "score"

```
hist(D1$score)
```



#Create a categorical outcome variable based on student score to advise the teacher using an "ifelse" statement

```
#Use nested ifelse statements to create 3 categories
#no_action if the student gets a score above 0.7
#monitor if student is between 0.3 and 0.7
#intervene if student is 0.3 or below

D1$advice <- ifelse(D1$score > 0.3,
                    ifelse(D1$score > 0.7,
                           "no_action",
                           "monitor"),
                    "intervene"
                    )
```

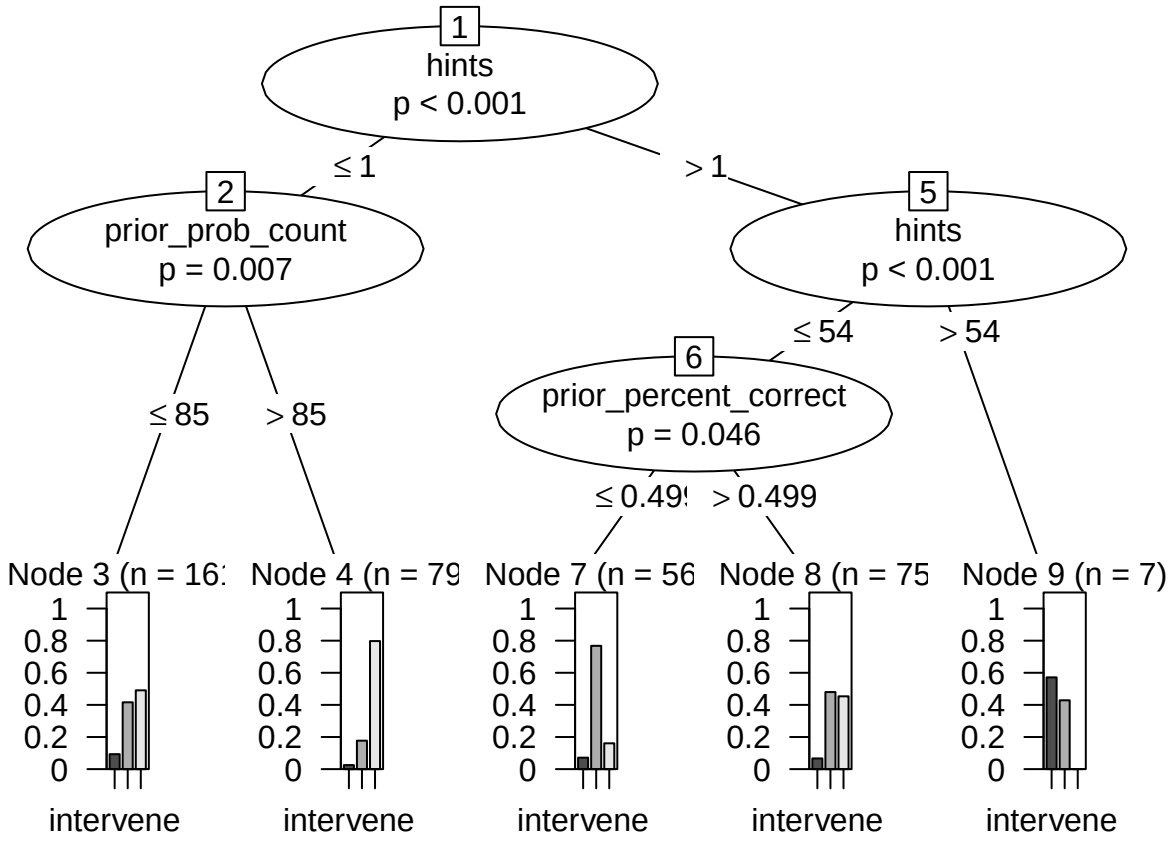
#Build a decision tree that predicts "advice" based on how many problems students have answered before, the percentage of those problems they got correct and how many hints they required

```
#Create a conditional inference tree using ctree
#Recursive partitioning with continuous and categorical data

score_ctree <- ctree(factor(advice) ~ prior_prob_count +
                                      prior_percent_correct +
                                      hints,
                     data = D1)
```

#Plot tree

```
plot(score_ctree)
```



Please interpret the tree, which two behaviors do you think the teacher should most closely pay attention to?

The output does not show properly - bar on the left is monitor, middle is intervene, and right is no action. This can be seen if you open the plot in a new window and maximise it.

Teachers should definitely pay attention to hints. Hints are relevant at two split levels. It also serves as an initial split. Those who ask for any hints (right of the tree) are far more likely to be in the intervene and monitor groups, while those who ask for no hints are much more likely to require no action. Those who ask for many hints probably require more intervention than those who ask for less as well.

Teachers should also pay attention to prior prob count correct. Relative to prior percent correct, differences in prior prob count are decisive for more students (240 vs 131), suggest greater magnitude increases in the

number of students where teachers need to intervene rather than take no action, and similar magnitude increases in the proportion that they should monitor rather than take no action

#Test Tree Upload the data "intelligent_tutor_new.csv". This is a data set of a differnt sample of students doing the same problems in the same system. We can use the tree we built for the previous data set to try to predict the "advice" we should give the teacher about these new students.

```
#Upload new data

D2 <- read.csv("intelligent_tutor_new.csv")

#Generate predicted advice using the predict() command for new students based on tree generated from ol
D2$prediction <- predict(score_ctree, D2)
```

Part III

Compare the predicted advice with the actual advice that these students recieved. What is the difference between the observed and predicted results?

```
#Compute the actual advice that the Test set students received
#Weirdly all the score data for the test set == 1, so we should expect everyone in the test set to requ
D2$advice <- ifelse(D2$score > 0.3,
                    ifelse(D2$score > 0.7,
                           "no_action",
                           "monitor"),
                    "intervene"
                    )

#Compute difference between observed and predicted results
conf_mat <- table(D2$advice, D2$prediction)
conf_mat
```

```
##
##             intervene monitor no_action
##   no_action         2      68       130
```

```
#Calculate accuracy based on take no action being the 'positive' result - did the model correctly ident
accuracy <- conf_mat[1,3]/sum(conf_mat[1, ])
accuracy
```

```
## [1] 0.65
```

With an accuracy rate of 65% and a high false negative rate of 35%, one might argue that the model is likely inappropriate for use with the intelligent tutoring system. In a dataset where everyone should have no action taken, the model suggests that the teacher should monitor or intervene with 35% of the students who do not need the help and are coping very well - a waste of effort. It is possible to argue that the model is overfit to the training data of students with mixed scores, such that it could not predict situations where everyone does well with sufficient accuracy.

However, one might argue that the test set is not a representative dataset, since it reflects an unlikely scenario where everyone gets perfect scores regardless of variation in their use of hints, prior experience, etc. Following this argument, the test set presents an unfair evaluative context on the model, trained on data that is likely more representative of future data that the model will be deployed on.

To Submit Your Assignment

Please submit your assignment by first "knitting" your RMarkdown document into an html file and then commit, push and pull request both the RMarkdown file and the html file.