Assignment 6 Response

Timothy Lee

14/11/2019

#Assignment 6

In this assignment you will be looking at data from a MOOC. It contains the following per-student variables:

certified (yes/no) - Whether or not a student paid for the course
forum.posts (numeric) - How many forum posts a student made throughout the course
grade (numeric) - A student's average grade for the course exam
assignment (numeric) - A student's average grade for the course assignments

##Part I

#Packages

```r
library(rpart)
library(dplyr)
```

```
##
## 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
```

```r
library(readr)
```

#Data

```r
#Upload the data sets MOOC1.csv and MOOC2.csv
M1 <- read.csv("MOOC1.csv", header = TRUE)

M2 <- read.csv("MOOC2.csv", header = TRUE)
```

#Decision tree

```r
#Using the rpart package generate a classification tree predicting certified from the other variables i

c.tree1 <- rpart(certified ~ grade +
                   assignment,
                 data = M1)

#Check the results from the classifcation tree using the printcp() command
printcp(c.tree1)
```

```
##
## Classification tree:
## rpart(formula = certified ~ grade + assignment, data = M1)
##
## Variables actually used in tree construction:
## [1] assignment grade
##
## Root node error: 275/1000 = 0.275
##
## n= 1000
##
##         CP nsplit rel error   xerror      xstd
## 1 0.923636      0  1.000000 1.000000 0.0513455
## 2 0.058182      1  0.076364 0.076364 0.0164880
## 3 0.010000      2  0.018182 0.018182 0.0081108
```

```
#Plot your tree

post(c.tree1, file = "tree1.ps", title = "MOOC1") #This creates a pdf image of the tree
```

##Part II

#The heading "xerror" in the printcp table stands for "cross validation error", it is the error rate of assigning students to certified/uncertified of the model averaged over 10-fold cross validation. CP stands for "Complexity Parameter" and represents the cost to error for adding a node to the tree. Notice it decreases as we add more nodes to the tree which implies that more nodes make better predictions. However, more nodes also mean that we may be making the model less generalizable, this is known as "overfitting".

#If we are worried about overfitting we can remove nodes form our tree using the prune() command, setting cp to the CP value from the table that corresponds to the number of nodes we want the tree to terminate at. Let's set it to two nodes.

```
#Set cp to the level at which you want the tree to end
c.tree2 <- prune(c.tree1, cp = 0.058182)

#Visualize this tree and compare it to the one you generated earlier
post(c.tree2, file = "tree2.ps", title = "MOOC2") #This creates a pdf image of the tree
```

It's the same tree, but it stops splitting after the first split, instead of splitting one more time

#Now use both the original tree and the pruned tree to make predictions about the the students in the second data set. Which tree has a lower error rate?

```
M2$predict1 <- predict(c.tree1, M2, type = "class")

M2$predict2 <- predict(c.tree2, M2, type = "class")

#Tree 1 confusion matrix
tree1_conf_mat <- table(M2$certified, M2$predict1)
tree1_conf_mat
```

```
##
##         no  yes
##   no  2056   24
##   yes 7790  130
```

```
#Tree 2 confusion matrix
tree2_conf_mat <- table(M2$certified, M2$predict2)
tree2_conf_mat
```

```
##
##         no  yes
##   no   896 1184
##   yes 3453 4467
```

```
#Formatting
cat("\n 1 - Accuracy = \n")
```

```
##
##  1 - Accuracy =
```

```
#Calculate error as 1 - accuracy
tree1_error <- 1 - ((tree1_conf_mat[1,1] + tree1_conf_mat[2,2])/length(M2$certified))
tree1_error
```

```
## [1] 0.7814
```

```
tree2_error <- 1 - ((tree2_conf_mat[1,1] + tree2_conf_mat[2,2])/length(M2$certified))
tree2_error
```

```
## [1] 0.4637
```

Tree 2 has a higher accuracy/lower error rate

##Part III

Choose a data file from the (University of Michigan Open Data Set)[https://github.com/bkoester/PLA/tree/master/data]. Choose an outcome variable that you would like to predict. Build two models that predict that outcome from the other variables. The first model should use raw variables, the second should feature select or feature extract variables from the data. Which model is better according to the cross validation metrics?

```
#Get Student record table
UMich_student.course <- read_csv("https://raw.githubusercontent.com/bkoester/PLA/master/student.course.


x <- filter(UMich_student.course, DIV == "O")
```

Codebook

- ANONID - Anonymous ID
- SUBJECT - Academic Subject
- CATALOG_NBR - Course code? E.g. ACC 272
- GRD_PTS_PER_UNIT - Grade Points Per Unit
- GPAO - GPA in other classes
- DIV - ACADEMIC DIVISION (E - Engineering, H - Humanities, O - Undergraduate Research, P - Business/Accounting/Finance, S - Science, SS - Social Science)

- ANON_INSTR_ID - ANON INSTRUCTOR ID
- TERM - Term number (see term.table.txt on the website)

Will choose GPAO as outcome variable

```
#First Model
#Use all available variables to predict GPAO
c.tree_model_1 <- rpart(GPAO ~ CATALOG_NBR + SUBJECT + GRD_PTS_PER_UNIT + DIV + ANON_INSTR_ID + TERM,
                        data = UMich_student.course)

#Check the results from the classifcation tree using the printcp() command
printcp(c.tree_model_1)
```

```
##
## Regression tree:
## rpart(formula = GPAO ~ CATALOG_NBR + SUBJECT + GRD_PTS_PER_UNIT +
##     DIV + ANON_INSTR_ID + TERM, data = UMich_student.course)
##
## Variables actually used in tree construction:
## [1] GRD_PTS_PER_UNIT SUBJECT
##
## Root node error: 393398/1327065 = 0.29644
##
## n= 1327065
##
##         CP nsplit rel error  xerror      xstd
## 1 0.205534      0   1.00000 1.00000 0.0035243
## 2 0.042608      1   0.79447 0.79447 0.0034599
## 3 0.030628      2   0.75186 0.75186 0.0034921
## 4 0.021177      3   0.72123 0.72124 0.0032739
## 5 0.012353      4   0.70005 0.70007 0.0032687
## 6 0.011524      5   0.68770 0.68778 0.0032625
## 7 0.010000      6   0.67618 0.67629 0.0032569
```

```
#Plot your tree
post(c.tree_model_1, file = "tree3.ps", title = "Model 1") #This creates a pdf image of the tree
```

```
#Second Model
#Extract two variables to predict GPAO - Subject and term
#If Grade inflation is a thing, later terms should have higher GPAO
#If some subjects are easier to get As in, subject should predict GPAO (presuming that students taking
#in that area)
c.tree_model_2 <- rpart(GPAO ~ SUBJECT + TERM,
                        data = UMich_student.course)

#Check the results from the classifcation tree using the printcp() command
printcp(c.tree_model_2)
```

```
##
## Regression tree:
## rpart(formula = GPAO ~ SUBJECT + TERM, data = UMich_student.course)
##
```

```
## Variables actually used in tree construction:
## [1] SUBJECT
##
## Root node error: 393398/1327065 = 0.29644
##
## n= 1327065
##
##          CP nsplit rel error  xerror      xstd
## 1 0.012413      0   1.00000 1.00000 0.0035243
## 2 0.010000      1   0.98759 0.98761 0.0035252
```

```
#Plot your tree
post(c.tree_model_2, file = "tree4.ps", title = "Model 2") #This creates a pdf image of the tree
```

Two models were used to predict Student's GPA in all other classes. The first model used all other variables. The second model used two variables - subject a student is enrolled in and the term.

The first model generates 6 splits, with a 10-fold CV error of 0.67629 and standard error of 0.0032569. I decided not to prune the model given that this is the lowest error rate reported among all splits, and there was no substantial difference between the CV standard error in this level of the tree relative to other levels of the tree.

The second model generates one split, with a 10-fold CV error of 0.9876 and standard error of 0.0035252. Since there was only one split with a negligible increase in standard error and a minimal decrease in 10-fold CV error, the tree at the level of 1 split was retained.

Overall, the first model generates a better tree as it has a lower cross-validation error.

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.