Module6L2

Neha Parulekar

March 11, 2016

# Additional packages needed

install.packages("ggplot2");  
install.packages("C50");  
install.packages("gmodels");  
install.packages("rpart");  
install.packages("rattle");  
install.packages("RColorBrewer");  
install.packages("tree");  
install.packages("party");

require("ggplot2");

## Loading required package: ggplot2

require("C50");

## Loading required package: C50

require("gmodels");

## Loading required package: gmodels

require("rpart");

## Loading required package: rpart

require("RColorBrewer");

## Loading required package: RColorBrewer

require("tree");

## Loading required package: tree

require("party");

## Loading required package: party

## Loading required package: grid

## Loading required package: mvtnorm

## Loading required package: modeltools

## Loading required package: stats4

## Loading required package: strucchange

## Loading required package: zoo

##   
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

## Loading required package: sandwich

## **About the dataset**

The dataset used is taken from UCI Machine learning Repository. The dataset is **Teaching Assistant Evaluation Dataset**. The data consist of evaluations of teaching performance over three regular semesters and two summer semesters of 151 teaching assistant (TA) assignments at the Statistics Department of the University of Wisconsin-Madison. The scores were divided into 3 roughly equal-sized categories ("low", "medium", and "high") to form the class variable.

Attribute Information. 1. Whether of not the TA is a native English speaker (binary); 1=English speaker, 2=non-English speaker 2. Course instructor (categorical, 25 categories) 3. Course (categorical, 26 categories) 4. Summer or regular semester (binary) 1=Summer, 2=Regular 5. Class size (numerical) 6. Class attribute (categorical) 1=Low, 2=Medium, 3=High

# Loading the dataset   
  
Data\_Url <-"https://archive.ics.uci.edu/ml/machine-learning-databases/tae/tae.data"  
TAData <- read.csv(url(Data\_Url), header = FALSE, sep = ",")  
head(TAData)

## V1 V2 V3 V4 V5 V6  
## 1 1 23 3 1 19 3  
## 2 2 15 3 1 17 3  
## 3 1 23 3 2 49 3  
## 4 1 5 2 2 33 3  
## 5 2 7 11 2 55 3  
## 6 2 23 3 1 20 3

##Step 1: Decision Trees -----------------------------------  
  
# Whether of not the TA is a native English speaker: 1=English speaker, 2=non-English speaker  
  
##Step 2: Exploring and preparing the data -----------------------  
str(TAData)

## 'data.frame': 151 obs. of 6 variables:  
## $ V1: int 1 2 1 1 2 2 2 2 1 2 ...  
## $ V2: int 23 15 23 5 7 23 9 10 22 15 ...  
## $ V3: int 3 3 3 2 11 3 5 3 3 3 ...  
## $ V4: int 1 1 2 2 2 1 2 2 1 1 ...  
## $ V5: int 19 17 49 33 55 20 19 27 58 20 ...  
## $ V6: int 3 3 3 3 3 3 3 3 3 3 ...

# look at the class  
table(TAData$V1)

##   
## 1 2   
## 29 122

# create a random sample for training and test data  
set.seed(99999)  
TA\_rand <- TAData[order(runif(151)),]  
  
# compare the original order and TA\_rand(random order)  
summary(TAData$V6)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.00 1.00 2.00 2.02 3.00 3.00

summary(TA\_rand$V6)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.00 1.00 2.00 2.02 3.00 3.00

head(TAData$V6)

## [1] 3 3 3 3 3 3

head(TA\_rand$V6)

## [1] 2 3 1 3 3 2

# split the data frames  
TAData\_train <- TA\_rand[1:100,]  
TAData\_test <- TA\_rand[101:151,]  
  
# converting int to factors  
TAData\_train$V1 <- as.factor(TAData\_train$V1)  
TAData\_test$V1 <- as.factor(TAData\_test$V1)  
  
# check proportion of class variable  
prop.table(table(TAData\_train$V1))

##   
## 1 2   
## 0.16 0.84

prop.table(table(TAData\_test$V1))

##   
## 1 2   
## 0.254902 0.745098

## Step 3: Training a model on the data ---------------------  
  
model <- C5.0(TAData\_train[-1],TAData\_train$V1)  
  
# display simple facts about the tree  
model

##   
## Call:  
## C5.0.default(x = TAData\_train[-1], y = TAData\_train$V1)  
##   
## Classification Tree  
## Number of samples: 100   
## Number of predictors: 5   
##   
## Tree size: 3   
##   
## Non-standard options: attempt to group attributes

# display detailed information about the tree  
summary(model)

##   
## Call:  
## C5.0.default(x = TAData\_train[-1], y = TAData\_train$V1)  
##   
##   
## C5.0 [Release 2.07 GPL Edition] Thu Mar 17 08:14:16 2016  
## -------------------------------  
##   
## Class specified by attribute `outcome'  
##   
## Read 100 cases (6 attributes) from undefined.data  
##   
## Decision tree:  
##   
## V2 <= 21: 2 (78/6)  
## V2 > 21:  
## :...V6 <= 2: 2 (13/3)  
## V6 > 2: 1 (9/2)  
##   
##   
## Evaluation on training data (100 cases):  
##   
## Decision Tree   
## ----------------   
## Size Errors   
##   
## 3 11(11.0%) <<  
##   
##   
## (a) (b) <-classified as  
## ---- ----  
## 7 9 (a): class 1  
## 2 82 (b): class 2  
##   
##   
## Attribute usage:  
##   
## 100.00% V2  
## 22.00% V6  
##   
##   
## Time: 0.0 secs

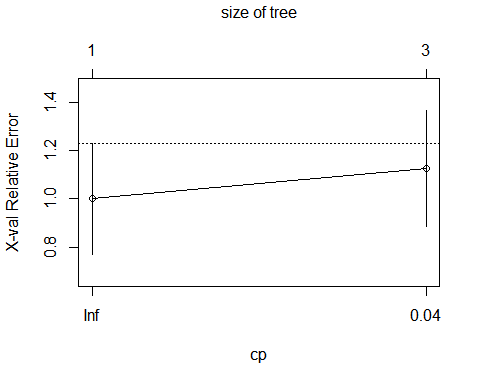
## Step 4: Evaluating model performance -------------------  
  
# create a factor vector of predictions(model) on test data  
TAData\_pred <- predict(model, TAData\_test)  
  
# cross tabulation of predicted versus actual classes  
  
CrossTable(TAData\_test$V1, TAData\_pred, prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE, dnn = c('actual type', 'predicted type'))

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Table Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 51   
##   
##   
## | predicted type   
## actual type | 1 | 2 | Row Total |   
## -------------|-----------|-----------|-----------|  
## 1 | 3 | 10 | 13 |   
## | 0.059 | 0.196 | |   
## -------------|-----------|-----------|-----------|  
## 2 | 1 | 37 | 38 |   
## | 0.020 | 0.725 | |   
## -------------|-----------|-----------|-----------|  
## Column Total | 4 | 47 | 51 |   
## -------------|-----------|-----------|-----------|  
##   
##

formula <- V1 ~ V2 + V3 + V4 + V5 + V6  
  
fit = rpart(formula, method = "class", data = TAData\_train)  
  
# display the results  
printcp(fit)

##   
## Classification tree:  
## rpart(formula = formula, data = TAData\_train, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] V2 V6  
##   
## Root node error: 16/100 = 0.16  
##   
## n= 100   
##   
## CP nsplit rel error xerror xstd  
## 1 0.15625 0 1.0000 1.000 0.22913  
## 2 0.01000 2 0.6875 1.125 0.24012

# visualize cross-validation results  
plotcp(fit)



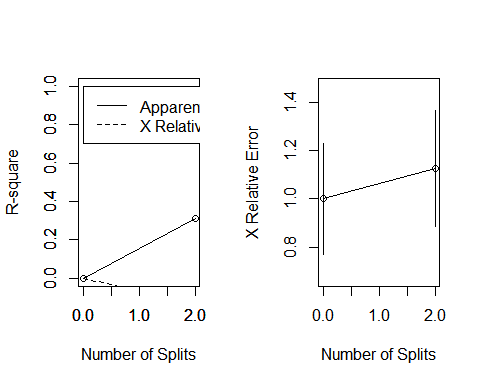
# detailed summary of splits  
summary(fit)

## Call:  
## rpart(formula = formula, data = TAData\_train, method = "class")  
## n= 100   
##   
## CP nsplit rel error xerror xstd  
## 1 0.15625 0 1.0000 1.000 0.2291288  
## 2 0.01000 2 0.6875 1.125 0.2401172  
##   
## Variable importance  
## V2 V6 V4 V5   
## 50 30 13 7   
##   
## Node number 1: 100 observations, complexity param=0.15625  
## predicted class=2 expected loss=0.16 P(node) =1  
## class counts: 16 84  
## probabilities: 0.160 0.840   
## left son=2 (22 obs) right son=3 (78 obs)  
## Primary splits:  
## V2 < 21.5 to the right, improve=4.893986, (0 missing)  
## V6 < 2.5 to the right, improve=1.853262, (0 missing)  
## V4 < 1.5 to the left, improve=1.507763, (0 missing)  
## V5 < 48.5 to the right, improve=1.280000, (0 missing)  
## V3 < 3.5 to the left, improve=1.280000, (0 missing)  
##   
## Node number 2: 22 observations, complexity param=0.15625  
## predicted class=2 expected loss=0.4545455 P(node) =0.22  
## class counts: 10 12  
## probabilities: 0.455 0.545   
## left son=4 (9 obs) right son=5 (13 obs)  
## Primary splits:  
## V6 < 2.5 to the right, improve=3.1825950, (0 missing)  
## V5 < 33 to the right, improve=0.7305195, (0 missing)  
## V2 < 22.5 to the right, improve=0.4475524, (0 missing)  
## Surrogate splits:  
## V4 < 1.5 to the left, agree=0.773, adj=0.444, (0 split)  
## V5 < 43.5 to the right, agree=0.682, adj=0.222, (0 split)  
## V2 < 22.5 to the right, agree=0.636, adj=0.111, (0 split)  
##   
## Node number 3: 78 observations  
## predicted class=2 expected loss=0.07692308 P(node) =0.78  
## class counts: 6 72  
## probabilities: 0.077 0.923   
##   
## Node number 4: 9 observations  
## predicted class=1 expected loss=0.2222222 P(node) =0.09  
## class counts: 7 2  
## probabilities: 0.778 0.222   
##   
## Node number 5: 13 observations  
## predicted class=2 expected loss=0.2307692 P(node) =0.13  
## class counts: 3 10  
## probabilities: 0.231 0.769

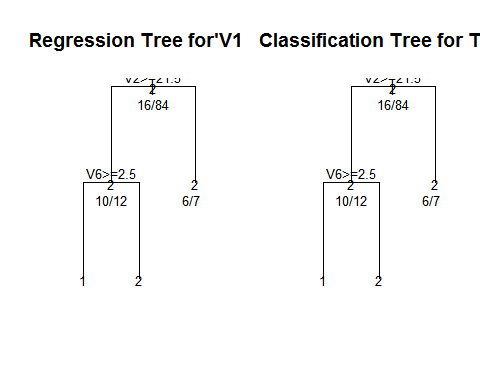
# create additional plots   
  
# two plots on one page  
par(mfrow = c(1,2))   
# visualize cross-validation results   
rsq.rpart(fit)

##   
## Classification tree:  
## rpart(formula = formula, data = TAData\_train, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] V2 V6  
##   
## Root node error: 16/100 = 0.16  
##   
## n= 100   
##   
## CP nsplit rel error xerror xstd  
## 1 0.15625 0 1.0000 1.000 0.22913  
## 2 0.01000 2 0.6875 1.125 0.24012

## Warning in rsq.rpart(fit): may not be applicable for this method



# plot tree   
plot(fit, uniform = TRUE, main = "Regression Tree for'V1'")  
text(fit, use.n = TRUE, all = TRUE, cex = .8)  
  
### ---------------- plot tree  
  
plot(fit, uniform = T, main = "Classification Tree for TA ")  
text(fit, use.n = TRUE, all = TRUE, cex = .8)

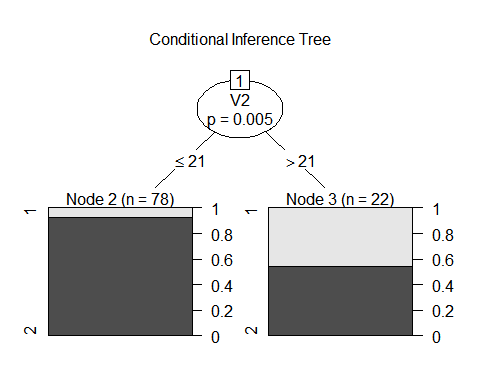


##----------------- TREE package  
  
tr = tree(formula, data = TAData\_train)  
summary(tr)

##   
## Classification tree:  
## tree(formula = formula, data = TAData\_train)  
## Variables actually used in tree construction:  
## [1] "V2" "V5" "V6"  
## Number of terminal nodes: 10   
## Residual mean deviance: 0.5423 = 48.81 / 90   
## Misclassification error rate: 0.11 = 11 / 100

plot(tr); text(tr)  
  
##-------------------Party package  
  
ct = ctree(formula, data = TAData\_train)  
plot(ct, main ="Conditional Inference Tree")  
  
# Estimated class probabilities  
tr.pred = predict(ct, newdata = TAData\_train, type = "prob")  
  
#Table of prediction errors  
table(predict(ct), TAData\_train$V1)

##   
## 1 2  
## 1 0 0  
## 2 16 84



\***Generate a Decision Tree with your data. You canuse any method/package you wish. Answer the following questions:**

***Does the size of the data set make a difference?*** I think the size of the dataset makes a big difference.I think increase in the size of the dataset would reduce the error. But this should be done keeping in mind not to overfit the tree as sometime even though the accurary becomes constant, we have a increase in tree subsets with incrase in data size. As we can see with this data it would have had helped if the dataset contained more attributes of binary classification as that would have give more tree subsets which would have helped in better accuracy of the dataset.

\***Do the rules make sense? If so why did the algorithm generate good rules? If not, why not?** Since the dataset does not have many binary attributes the rules makes sense as The decision tree can be linearized into decision rules, where the outcome is the contents of the leaf node, and the conditions along the path form a conjunction in the if clause. Thus helping in finding accurate reults.

\***Does scaling, normalization or leaving the data unscaled make a difference?** Decision tree does not require scaling though unless we would like to compare with other supervised learning methods.