Project 2 notebook

Meenakshi Nagarajan

Oct 5, 2017

# Dataset – Teaching Assistant Evaluation

<http://archive.ics.uci.edu/ml/datasets/Teaching+Assistant+Evaluation>

#Meenakshi Nagarajan  
#nagarajan.12@wright.edu  
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

#Load the data into 'mydata'  
mydata=read.csv(file="/Users/meenakshinagarajan/Desktop/Datamining/Teaching\_Assistant\_Evaluation.csv",head=TRUE,sep=",")  
head(mydata)

## Typeofspeaker CourseInstructor Course TypeofSemester ClassSize  
## 1 1 23 3 1 19  
## 2 2 15 3 1 17  
## 3 1 23 3 2 49  
## 4 1 5 2 2 33  
## 5 2 7 11 2 55  
## 6 2 23 3 1 20  
## ClassAttribute  
## 1 3  
## 2 3  
## 3 3  
## 4 3  
## 5 3  
## 6 3

# Background of data

The dataset used in this study is obtained from the UCI Machine learning repository. (<http://archive.ics.uci.edu/ml/datasets.html>). It consists of teaching performance evaluation of 151 teaching assistant assignments at statistics department of University of Wisconsin-Madison. The output classs attribute is divinded into three categories namely, low (1), medium(2) and high(3).

Dataset Characteristics: Multivariate

Attribute characteristics: Categorical, integer

Date Donated: 1997/ 06/07

Number of instances: 151

Number of Attributes: 5

Missing values: None

# Attributes

Type of speaker: 1 = English speaker 2= Non-English speaker

Course Instructor: It is divided into 25 categories

Course: The course that is being taught is divided into 26 categories

Type of semester : Summer or regular semester. 1= Summer 2= Regular

Class size: Number of students in a class. It is a numerical value.

ClassAttribute: Performance measure of TA. 1 = Low 2= Medium 3= High

# Converting numerics to factors

mydata$ClassAttribute <- factor(mydata$ClassAttribute, levels=sort(unique(mydata$ClassAttribute)))  
mydata$Typeofspeaker<- factor(mydata$Typeofspeaker, levels=sort(unique(mydata$Typeofspeaker)))  
mydata$CourseInstructor<- factor(mydata$CourseInstructor, levels=sort(unique(mydata$CourseInstructor)))  
mydata$Course<- factor(mydata$Course, levels=sort(unique(mydata$Course)))  
mydata$TypeofSemester<- factor(mydata$TypeofSemester, levels=sort(unique(mydata$TypeofSemester)))  
#structure of data  
str(mydata)

## 'data.frame': 151 obs. of 6 variables:  
## $ Typeofspeaker : Factor w/ 2 levels "1","2": 1 2 1 1 2 2 2 2 1 2 ...  
## $ CourseInstructor: Factor w/ 25 levels "1","2","3","4",..: 23 15 23 5 7 23 9 10 22 15 ...  
## $ Course : Factor w/ 26 levels "1","2","3","4",..: 3 3 3 2 11 3 5 3 3 3 ...  
## $ TypeofSemester : Factor w/ 2 levels "1","2": 1 1 2 2 2 1 2 2 1 1 ...  
## $ ClassSize : int 19 17 49 33 55 20 19 27 58 20 ...  
## $ ClassAttribute : Factor w/ 3 levels "1","2","3": 3 3 3 3 3 3 3 3 3 3 ...

# Creating training and testing data

set.seed(100)  
#70-30 split  
trainingData <- sample(1:nrow(mydata), 0.7\*nrow(mydata))  
training <- mydata[trainingData, ]  
test <- mydata[-trainingData, ]

# Build and test a Model using RandomForest approach

library(randomForest)

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

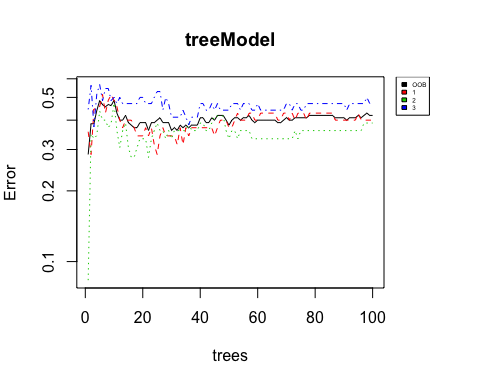
##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':  
##   
## combine

myModel = ClassAttribute ~ .  
treeModel <- randomForest(myModel, data=training, ntree=100, proximity=TRUE,log="y")  
print(treeModel)

##   
## Call:  
## randomForest(formula = myModel, data = training, ntree = 100, proximity = TRUE, log = "y")   
## Type of random forest: classification  
## Number of trees: 100  
## No. of variables tried at each split: 2  
##   
## OOB estimate of error rate: 41.9%  
## Confusion matrix:  
## 1 2 3 class.error  
## 1 21 7 7 0.4000000  
## 2 9 22 5 0.3888889  
## 3 8 8 18 0.4705882

layout(matrix(c(1,2),nrow=1),  
 width=c(4,1))   
#No margin on the right side  
par(mar=c(5,4,4,0))   
  
#Overall error of the model  
plot(treeModel, log="y")  
#No margin on the left side  
par(mar=c(5,0,4,2))  
plot(c(0,1),type="n", axes=F, xlab="", ylab="")  
#Adding legend  
legend("top", colnames(treeModel$err.rate),col=1:4,cex=0.4,fill=1:4)



# Predict on test data

predClass <- predict(treeModel,newdata=test)  
#confusion matrix  
table(predClass, test$ClassAttribute)

##   
## predClass 1 2 3  
## 1 8 6 2  
## 2 2 4 3  
## 3 4 4 13

cat("MisClassification Error Rate:",mean(as.character(predClass) != as.character(test$ClassAttribute)))

## MisClassification Error Rate: 0.4565217

A misclassification error of 45.6% is high. It could be improved by changing the model terms.