Problem6

Rodolfo Viana

05-07-2015

Durante os últimos anos a Universidade Federal de Campina Grande observa um número alto de evasão por parte dos alunos. Tentando entender os motivos dessa evasão analisamos uma amostra contendo dados importante. A nossa amostra tem os seguintes atributos:

1. MATRICULA: identificador do aluno
2. PERIODO: identificador do período letivo da universidade (ano.semestre)
3. COD_CURSO: identificador do curso
4. CURSO: nome do curso. Cada curso tem seu COD_CURSO
5. CODIGO: identificador da disciplina que o aluno cursou no periodo
6. DISCIPLINA: nome da disciplina referente que o aluno cursou no periodo.
7. CODIGO: Cada disciplina tem seu
8. CREDITOS: numero de créditos referente a disciplina
9. DEPARTAMENTO: departamento que ofertou a disciplina
10. MEDIA: média do aluno na disciplina (0 a 10). Alunos reprovados por falta numa disciplina recebem 0 e alunos que trancaram a disciplina recebem NA.
11. STATUS: Aprovado, Reprovado Por Falta, Trancado ou Reprovado. Se refere ao estado final do aluno na disciplina
12. PERIODO_INGRESSO: período letivo da universidade em que o aluno ingressou no curso.
13. PERIODO_RELATIVO: número de períodos que o aluno está matriculado na universidade. “1” refere-se ao aluno em seu primeiro periodo, “5” refere-se ao aluno no quinto período.
14. COD_EVASAO: identificador de evasão do aluno. “0” significa que o aluno continuou ativo na universidade no período seguinte e “1” significa que o aluno desistiu do curso nesse período e não voltou a se matricular no seguinte.

O nosso objetivo é construir um modelo de classificação que nos diga se o aluno irá evadir ou não. Para o nosso primeiro modelo vamos classificar apenas para os alunos que tem período relativo 1.

```
library(plyr)
library(dplyr)

arquivo <- read.csv("~/Projetos/DataAnalysis/Assignment5/training_sem_acento.csv")

#Transformação para factor
arquivo$COD_EVASAO <- as.factor(arquivo$COD_EVASAO)
arquivo$DEPARTAMENTO <- as.factor(arquivo$DEPARTAMENTO)
arquivo$COD_CURSO <- as.factor(arquivo$COD_CURSO)
arquivo$MEDIA <- as.character(arquivo$MEDIA)
arquivo$MEDIA[is.na(arquivo$MEDIA)] <- -10
arquivo$MEDIA <- as.numeric(arquivo$MEDIA)

#Missing
levels(arquivo$DEPARTAMENTO)[1] = NA
levels(arquivo$DISCIPLINA)[1] = NA
levels(arquivo$SITUACAO)[1] = NA
```

Antes de criar o modelo é importante dividir o arquivo original em treino e teste (75% treinamento, 25% teste), para assim verificar o F-measure e saber se um modelo criado é melhor do que o modelo anterior.

```
#Primeiro periodo
set.seed(12345)
primeiro_periodo <- filter(arquivo, PERIODO_RELATIVO == 1)
primeiro_periodo <- primeiro_periodo[order(runif(nrow(primeiro_periodo))), ]

#Divisao de treino e teste
treino <- primeiro_periodo[1:round(1*nrow(primeiro_periodo)), ]
test <- primeiro_periodo[round(0.75*nrow(primeiro_periodo)):nrow(primeiro_periodo), ]
```

Podemos notar que a proporção entre evasão e não evasão se manteve parecida após a divisão de treino e teste.

```
prop.table(table(primeiro_periodo$COD_EVASAO))
```

```
##
##         0         1
## 0.8936799 0.1063201
```

```
prop.table(table(treino$COD_EVASAO))
```

```
##
##         0         1
## 0.8936799 0.1063201
```

```
prop.table(table(test$COD_EVASAO))
```

```
##
##         0         1
## 0.8972542 0.1027458
```

Devemos agora decidir qual classificador iremos utilizar, **SVM, kNN, árvores/florestas aleatórias**. Para ajudar na nossa escolha utilizamos a biblioteca caret. Foram utilizados os mesmo atributos do que foi entregue no problema 5. (período letivo da universidade, código da disciplina cursada, departamento e a situação)

```
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```
library(e1071)
library("C50")

treino_labels = treino[, 14] ## Classes das instâncias de treino
test_labels = test[, 14] ## Classes das instâncias de teste
treino = treino[-14] ## Exclui variável alvo

#Transformação para numeric
treino$PERIODO <- as.numeric(treino$PERIODO)
treino$DISCIPLINA <- as.numeric(treino$DISCIPLINA)
treino$DEPARTAMENTO <- as.numeric(treino$DEPARTAMENTO)
treino$SITUACAO <- as.numeric(treino$SITUACAO)
treino$MEDIA <- as.numeric(treino$MEDIA)

best_tree_model = train(treino[c(5,7,9,11)], treino_labels, method="C5.0", preP
rocess=c("range"))
```

```
## Warning in predict.C5.0(modelFit, newdata, trial = submodels$trials[j]):
## 'trials' should be <= 9 for this object. Predictions generated using 9
## trials
```

```
## Warning in predict.C5.0(modelFit, newdata, trial = submodels$trials[j]):
## 'trials' should be <= 9 for this object. Predictions generated using 9
## trials
```

```
accuracy_tree = max(best_tree_model$resample$Accuracy)

best_tree_model
```

```
## C5.0
##
## 13544 samples
##     4 predictors
##     2 classes: '0', '1'
##
## Pre-processing: re-scaling to [0, 1]
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 13544, 13544, 13544, 13544, 13544, 13544, ...
##
## Resampling results across tuning parameters:
##
##   model  winnow  trials  Accuracy   Kappa      Accuracy SD  Kappa SD
##   rules  FALSE    1      0.9249334  0.5683741  0.002538223  0.02033295
##   rules  FALSE   10      0.9261435  0.5503379  0.002906645  0.02111526
##   rules  FALSE   20      0.9266046  0.5529125  0.003345438  0.01890111
##   rules   TRUE    1      0.9251268  0.5687646  0.002539434  0.02056292
##   rules   TRUE   10      0.9262402  0.5516451  0.002887797  0.02013165
##   rules   TRUE   20      0.9266530  0.5531434  0.003351928  0.01872461
##   tree   FALSE    1      0.9259350  0.5692082  0.002767998  0.02046351
##   tree   FALSE   10      0.9257560  0.5737972  0.003133273  0.01688445
##   tree   FALSE   20      0.9271098  0.5771135  0.002580767  0.01489415
##   tree    TRUE    1      0.9259672  0.5693008  0.002781422  0.02053226
##   tree    TRUE   10      0.9257882  0.5737213  0.003123309  0.01695299
##   tree    TRUE   20      0.9271179  0.5765929  0.002576985  0.01525101
##
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were trials = 20, model = tree
##  and winnow = TRUE.
```

Podemos observar que utilizando árvore/floresta o caret encontrou a melhor solução como sendo utilizando trials = 20, model = rules and winnow = TRUE.

Vamos agora utilizar o caret para encontrar a melhor solução utilizando o kNN como classificador.

```
best_knn_model <- train(treino[c(5,7,9,11)], treino_labels,
                 method = "knn",
                 preProcess = c("range"))

accuracy_knn = max(best_knn_model$resample$Accuracy)
best_knn_model
```

```
## k-Nearest Neighbors
##
## 13544 samples
##     4 predictors
##     2 classes: '0', '1'
##
## Pre-processing: re-scaling to [0, 1]
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 13544, 13544, 13544, 13544, 13544, 13544, ...
##
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa      Accuracy SD  Kappa SD
##   5  0.9206156  0.5374783  0.003752228  0.01922826
##   7  0.9215765  0.5424148  0.003905803  0.01809928
##   9  0.9224408  0.5471349  0.003537719  0.01646279
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was k = 9.
```

Podemos observar que utilizando kNN o caret encontrou a melhor solução como sendo utilizando k = 9.

```
#Vamos agora utilizar o caret para encontrar a melhor solução para o SVM como c
lassificardor.
#library("kernlab")
#best_svm_model <- train(treino[c(5,7,9,11)], treino_labels,
#                method = "svmRadial",
#                preProcess = c("range"))

#accuracy_svn = max(best_svm_model$resample$Accuracy)
#best_svm_model
```

Agora para ajudar na escolha do melhor classificador para esse problema vamos observar a acurácia dos dois classificadores:

```
accuracy_tree
```

```
## [1] 0.9309859
```

```
accuracy_knn
```

```
## [1] 0.9284433
```

Podemos observar que a floresta obteve valor mais alto. Por esse motivo escolhemos esse classificador com os parâmetros trials = 20, model = rules and winnow = TRUE para realizar a submissão no kaggle.