CheckPoint3

Rodolfo Viana

11-07-2015

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. Vamos classificar apenas para os alunos que tem período relativo 5.

```
library(plyr)
library(dplyr)

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

#Transformação para factor
arquivo$COD_EVASAO <- as.factor(arquivo$COD_EVASAO)
arquivo$COD_CURSO <- as.factor(arquivo$COD_CURSO)
arquivo$CODIGO <- as.factor(arquivo$CODIGO)
arquivo$CREDITOS <- as.factor(arquivo$CREDITOS)
arquivo$MEDIA[is.na(arquivo$MEDIA)] <- -1
```

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)
arquivo <- filter(arquivo, PERIODO_RELATIVO == 5)
arquivo <- arquivo[order(runif(nrow(arquivo))), ]

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

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

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

```
##
##          0          1
## 0.94101331 0.05898669
```

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

```
##
##          0          1
## 0.94206114 0.05793886
```

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

```
##
##          0          1
## 0.93791574 0.06208426
```

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 <= 1 for this object. Predictions generated using 1
## trials
```

```
## Warning in predict.C5.0(modelFit, newdata, trial = submodels$trials[j]):
## 'trials' should be <= 1 for this object. Predictions generated using 1
## trials
```

```
## Warning in predict.C5.0(modelFit, newdata, trial = submodels$trials[j]):
## 'trials' should be <= 1 for this object. Predictions generated using 1
## trials
```

```
## Warning in predict.C5.0(modelFit, newdata, trial = submodels$trials[j]):
## 'trials' should be <= 1 for this object. Predictions generated using 1
## trials
```

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

best_tree_model
```

```
## C5.0
##
## 4056 samples
##    4 predictors
##    2 classes: '0', '1'
##
## Pre-processing: re-scaling to [0, 1]
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 4056, 4056, 4056, 4056, 4056, 4056, ...
##
## Resampling results across tuning parameters:
##
##   model  winnow  trials  Accuracy   Kappa      Accuracy SD  Kappa SD
##   rules  FALSE    1      0.9426848  0.2131663  0.004937709  0.05349996
##   rules  FALSE   10      0.9433501  0.2185739  0.005511165  0.06101470
##   rules  FALSE   20      0.9439582  0.2158950  0.004597965  0.05134013
##   rules   TRUE    1      0.9426848  0.2131663  0.004937709  0.05349996
##   rules   TRUE   10      0.9433501  0.2185739  0.005511165  0.06101470
##   rules   TRUE   20      0.9439582  0.2158950  0.004597965  0.05134013
##   tree   FALSE    1      0.9419256  0.2612173  0.005103789  0.05303757
##   tree   FALSE   10      0.9426087  0.2110566  0.005745310  0.05915326
##   tree   FALSE   20      0.9437203  0.2268574  0.005323514  0.06196789
##   tree    TRUE    1      0.9419256  0.2612173  0.005103789  0.05303757
##   tree    TRUE   10      0.9426087  0.2110566  0.005745310  0.05915326
##   tree    TRUE   20      0.9437203  0.2268574  0.005323514  0.06196789
##
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were trials = 20, model = rules
##  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
##
## 4056 samples
##    4 predictors
##    2 classes: '0', '1'
##
## Pre-processing: re-scaling to [0, 1]
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 4056, 4056, 4056, 4056, 4056, 4056, ...
##
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa      Accuracy SD  Kappa SD
##   5  0.9295698  0.1988719  0.006757194  0.05493821
##   7  0.9351671  0.2092614  0.006571791  0.06096505
##   9  0.9377185  0.2136137  0.006436711  0.05163939
##
## 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] 0.9515449
```

```
accuracy_knn
```

```
## [1] 0.9497061
```

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.

Agora que já temos o nosso modelo ideial vamos criar novos atributos para melhorar o nosso classificador:

1. Média geral do período do aluno
2. Quantidade de cadeiras aprovadas
3. Quantidade de reprovações
4. Quantidade de reprovações por falta

```
#Criando novos atributos para o treino
treino_group <- group_by(treino, MATRICULA)
media <- summarise(treino_group, mean(MEDIA))
names(media) <- c("MATRICULA", "MEDIATOTAL")

reprovacao <- summarise(group_by(filter(treino, SITUACAO=="Reprovado"), MATRICU
LA), n())
names(reprovacao) <- c("MATRICULA", "REPROVACAO")

aprovado <- summarise(group_by(filter(treino, SITUACAO=="Aprovado"), MATRICUL
A), n())
names(aprovado) <- c("MATRICULA", "APROVADO")

target <- c("Reprovado por Falta", NA)
reprovado_falta <- summarise(group_by(filter(treino, SITUACAO %in% target), MAT
RICULA), n())
names(reprovado_falta) <- c("MATRICULA", "REPROVADOFALTA")

target <- c("Reprovado por Falta", "Reprovado", "Trancado")
n_aprovado <- summarise(group_by(filter(treino, SITUACAO %in% target), MATRICUL
A), n())
names(n_aprovado) <- c("MATRICULA", "NAPROVADO")


#Merge dos novos atributos
treino <- merge(treino, media, by = "MATRICULA", all = TRUE)
treino <- merge(treino, reprovacao, by = "MATRICULA", all = TRUE)
treino <- merge(treino, aprovado, by = "MATRICULA", all = TRUE)
treino <- merge(treino, reprovado_falta, by = "MATRICULA", all = TRUE)
treino <- merge(treino, n_aprovado, by = "MATRICULA", all = TRUE)

#Transformando NA em 0
treino[is.na(treino)] <- 0

#Criando novos atributos para o teste
test_group <- group_by(test, MATRICULA)
media <- summarise(test_group, mean(MEDIA))
names(media) <- c("MATRICULA", "MEDIATOTAL")

reprovacao <- summarise(group_by(filter(test, SITUACAO=="Reprovado"), MATRICUL
A), n())
names(reprovacao) <- c("MATRICULA", "REPROVACAO")

aprovado <- summarise(group_by(filter(test, SITUACAO=="Aprovado"), MATRICULA),
n())
names(aprovado) <- c("MATRICULA", "APROVADO")
```

```
target <- c("Reprovado por Falta", NA)
reprovado_falta <- summarise(group_by(filter(test, SITUACAO %in% target), MATRI
CULA), n())
names(reprovado_falta) <- c("MATRICULA", "REPROVADOFALTA")

target <- c("Reprovado por Falta", "Reprovado", "Trancado")
n_aprovado <- summarise(group_by(filter(test, SITUACAO %in% target), MATRICUL
A), n())
names(n_aprovado) <- c("MATRICULA", "NAPROVADO")


test <- merge(test, media, by = "MATRICULA", all = TRUE)
test <- merge(test, reprovacao, by = "MATRICULA", all = TRUE)
test <- merge(test, aprovado, by = "MATRICULA", all = TRUE)
test <- merge(test, reprovado_falta, by = "MATRICULA", all = TRUE)
test <- merge(test, n_aprovado, by = "MATRICULA", all = TRUE)

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

test[is.na(test)] <- 0
```

Agora com as 4 novas colunas criadas temos mais dados para analisar e ajudar no classificador.

```
model <- C5.0(treino[,c(3,5,7,9,10,11,14,15,16,17,18)], treino_labels, trials =
20, model = rules, winnow = TRUE)

model
```

```
##
## Call:
## C5.0.default(x = treino[, c(3, 5, 7, 9, 10, 11, 14, 15, 16, 17, 18)], y
##  = treino_labels, trials = 20, model = rules, winnow = TRUE)
##
## Classification Tree
## Number of samples: 4056
## Number of predictors: 11
##
## Number of boosting iterations: 20 requested;  1 used due to early stopping
##
## Non-standard options: attempt to group attributes
```

```
summary(model)
```

```
##
## Call:
## C5.0.default(x = treino[, c(3, 5, 7, 9, 10, 11, 14, 15, 16, 17, 18)], y
##  = treino_labels, trials = 20, model = rules, winnow = TRUE)
##
##
## C5.0 [Release 2.07 GPL Edition]      Sat Jul 11 19:34:15 2015
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 4056 cases (12 attributes) from undefined.data
##
## -----  Trial 0:  -----
##
## Decision tree:
##  0 (4056/235)
##
## -----  Trial 1:  -----
##
## Decision tree:
##  0 (4056/1131.5)
##
## *** boosting reduced to 1 trial since last classifier is very inaccurate
##
## *** boosting abandoned (too few classifiers)
##
##
## Evaluation on training data (4056 cases):
##
##      Decision Tree
##    ----------------
##    Size      Errors
##
##       1  235( 5.8%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##    3821          (a): class 0
##     235          (b): class 1
##
##
## Time: 0.1 secs
```

```
pred <- predict(model, test[,c(3,5,7,9,10,11,15,16,17,18,19)])

true_eva <- test$COD_EVASAO == 1
table(pred, true_eva)
```

```
##     true_eva
## pred FALSE TRUE
##    0  1269   84
##    1     0    0
```

Podemos notar que não criamos um bom classificador, pois ele classificou todas as saidas como sendo 0. Temos esse erro pois estamos com dados desbalanceados. No nosso arquivo de treino 94% dos dados são de não evasão e 6% dos dados são evasão.

Por essa razão resolvemos por, nos dados de treino, balancear os dados.

```
treino <- cbind(treino, treino_labels)

treino_positivo = filter(treino, treino_labels == 1)
treino_negativo = filter(treino, treino_labels == 0)

treino_negativo <- treino_negativo[order(runif(nrow(treino_positivo))), ]

novo_treino <- rbind(treino_positivo,treino_negativo)
novo_treino <- novo_treino[order(runif(nrow(novo_treino))), ]
prop.table(table(novo_treino$treino_labels))
```

```
##
##   0   1
## 0.5 0.5
```

Vamos criar um novo modelo agora com os dados balanceados:

```
model <- C5.0(novo_treino[,c(3,5,7,9,10,11,14,15,16,17,18)], novo_treino$trein
o_labels, trials = 20, model = rules, winnow = TRUE)

model
```

```
##
## Call:
## C5.0.default(x = novo_treino[, c(3, 5, 7, 9, 10, 11, 14, 15, 16, 17,
##  18)], y = novo_treino$treino_labels, trials = 20, model = rules, winnow
##  = TRUE)
##
## Classification Tree
## Number of samples: 470
## Number of predictors: 11
##
## Number of boosting iterations: 20 requested;  1 used due to early stopping
##
## Non-standard options: attempt to group attributes
```

```
summary(model)
```

```
##
```

```
## Call:
## C5.0.default(x = novo_treino[, c(3, 5, 7, 9, 10, 11, 14, 15, 16, 17,
##  18)], y = novo_treino$treino_labels, trials = 20, model = rules, winnow
##  = TRUE)
##
##
## C5.0 [Release 2.07 GPL Edition]      Sat Jul 11 19:34:16 2015
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 470 cases (12 attributes) from undefined.data
##
## -----  Trial 0:  -----
##
## Decision tree:
##
## SITUACAO > 3: 0 (30/6)
## SITUACAO <= 3:
## :...SITUACAO > 2: 1 (48/13)
##     SITUACAO <= 2:
##     :...DEPARTAMENTO > 11: 0 (163/70)
##         DEPARTAMENTO <= 11:
##         :...COD_CURSO = 14123100:
##             :...PERIODO <= 2012.1: 1 (96/32)
##             :   PERIODO > 2012.1:
##             :   :...DEPARTAMENTO <= 7: 1 (5/1)
##             :       DEPARTAMENTO > 7: 0 (53/21)
##             COD_CURSO = 12204100:
##             :...PERIODO <= 2011.2: 0 (28/8)
##                 PERIODO > 2011.2:
##                 :...PERIODO <= 2012.1: 1 (8)
##                     PERIODO > 2012.1:
##                     :...MEDIATOTAL > 8.716666: 1 (4)
##                         MEDIATOTAL <= 8.716666:
##                         :...MEDIATOTAL <= 6.475: 1 (3)
##                             MEDIATOTAL > 6.475: 0 (32/12)
##
## -----  Trial 1:  -----
##
## Decision tree:
##
## SITUACAO > 3: 0 (28.5/7.3)
## SITUACAO <= 3:
## :...SITUACAO > 2: 1 (46.8/15.9)
##     SITUACAO <= 2:
##     :...COD_CURSO = 12204100:
##         :...PERIODO <= 2011.2: 0 (27.4/9.8)
##         :   PERIODO > 2011.2: 1 (45.5/17.7)
##         COD_CURSO = 14123100:
##         :...MEDIATOTAL > 6.8: 1 (95.7/31.7)
##             MEDIATOTAL <= 6.8:
##             :...SITUACAO <= 1: 0 (152.9/69.3)
##                 SITUACAO > 1:
```

```
##                 :...DEPARTAMENTO > 15: 1 (4.9)
##                     DEPARTAMENTO <= 15:
##                     :...DISCIPLINA <= 26: 0 (23.3/6.3)
##                         DISCIPLINA > 26: 1 (44.9/18.3)
##
## -----  Trial 2:  -----
##
## Decision tree:
##  1 (470/224)
##
## *** boosting reduced to 2 trials since last classifier is very inaccurate
##
## *** boosting abandoned (too few classifiers)
##
##
## Evaluation on training data (470 cases):
##
##      Decision Tree
##    ----------------
##    Size      Errors
##
##      11  163(34.7%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##     189    46    (a): class 0
##     117   118    (b): class 1
##
##
##  Attribute usage:
##
##  100.00% SITUACAO
##   83.40% DEPARTAMENTO
##   48.72% COD_CURSO
##   48.72% PERIODO
##    8.30% MEDIATOTAL
##
##
## Time: 0.0 secs
```

```
pred <- predict(model, test[,c(3,5,7,9,10,11,15,16,17,18,19)])

true_eva <- test$COD_EVASAO == 1
table(pred, true_eva)
```

```
##     true_eva
## pred FALSE TRUE
##    0   723   43
##    1   546   41
```

Temos agora um f-measure com o valor de:

```
precision <- 41/41+43
recall <- 41/(41+546)

fmeasure2 <- 2*precision*recall/(precision+recall)
fmeasure2
```

```
## [1] 0.139472
```

Se mostrando melhor do que o modelo anterior, porém com valor de f-measure ainda muito baixo.