Exercise CART Class 08-01-2020

Daniel Ferreira Zanchetta and Lais Silva Almeida Zanchetta

1. Suponga el siguiente árbol simple T con sólo dos nodos (hojas) terminales. En el nodo raíz se tiene 100 individuos que se dividen en dos nodos hijos de 60 y 40 individuos cada uno. La variable de respuesta indica la compra (No o Si) de un cierto producto. Calcule la reducción de impureza que se obtiene al pasar del nodo padre a los dos nodos hijos.

```
#Nodo 1:
pnodo1 <- (1/60)^2 + (59/60)^2

#Nodo 2:
pnodo2 <- (21/40)^2 + (19/40)^2

#Gini index padre
it0 <- round((pnodo1*(60/100) + pnodo2*(40/100)),3)

#Gini index nodo hijo 1
it1 <- round(1-(1/60)^2-(59/60)^2,3)

#Gini index nodo hijo 2
it2 <- round(1-(21/40)^2-(19/40)^2,3)

#Drecrement of impurity
deltait <- it0 - ((60/100)*it1) - ((40/100)*it2)
deltait
```

```
## [1] 0.5616
```

2. Con el mismo árbol precedente, calcule su coste de mal clasificación R(T).

```
#Para hacer el calculo del coste de cada nodo hijo (rt1 y rt2 abajo),
hemos utilizado el valor correspondiente al SI para realización de la
compra.
rt1 <- 1-(1/60)
rt1
```

```
## [1] 0.9833333
```

```
rt2 <- 1-(19/40)
rt2
```

```
## [1] 0.525
```

3. Retome los datos del problema churn. Se trata ahora de obtener un árbol de decisión que nos permita efectuar predicciones sobre la probabilidad de baja de los clientes. Cargue en R la Liberia rpart y obtenga un árbol máximo (cp=0.0001) con crossvalidación (xval=10).

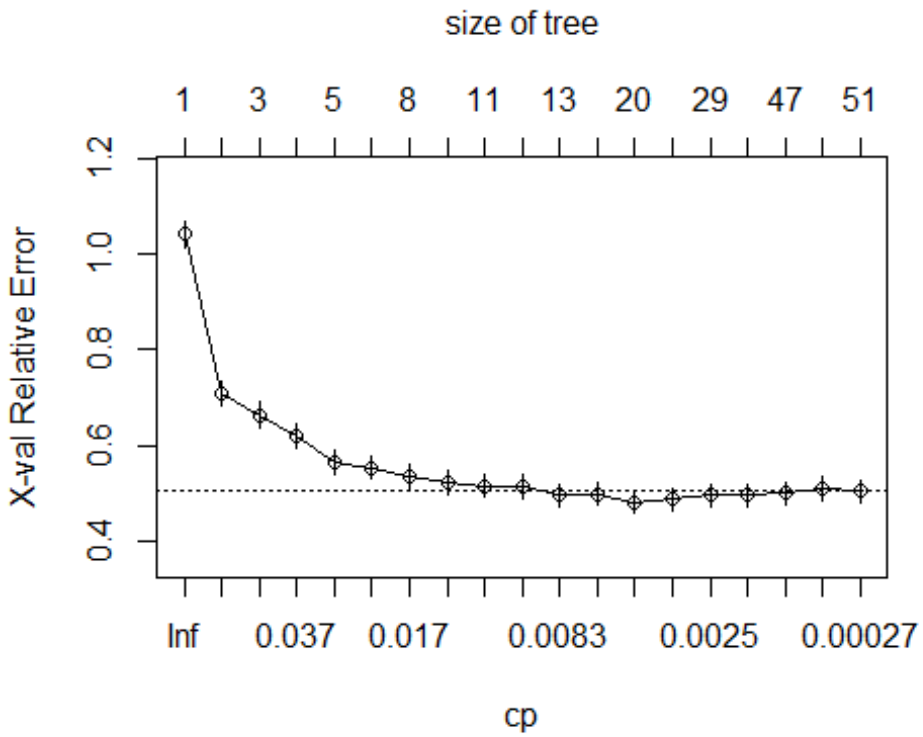
```
#library(mice)
library(rpart)
library(dplyr)

##
## 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

library(tidyr)
library(mice)

## Warning: package 'mice' was built under R version 3.6.2

## Loading required package: lattice

##
## Attaching package: 'mice'

## The following object is masked from 'package:tidyr':
##
##     complete

## The following objects are masked from 'package:base':
##
##     cbind, rbind

setwd("C:/Users/Daniel/Documents/Certificados & Faculdade/UPC Master Big
Data/Data Analytics/Aula 8 - 08-01/exer_cart")
churn <- read.table(file = "churn.txt",header = TRUE,sep = "")
churn$antig[churn$antig==99] <- NA
churn <- mice::complete(mice(churn, m=1))

##
##  iter imp variable
##    1   1  antig  Nomina  Pension  Debito_aff  VISA  MCard
##    2   1  antig  Nomina  Pension  Debito_aff  VISA  MCard
```

```
##   3   1  antig  Nomina  Pension  Debito_aff  VISA  MCard
##   4   1  antig  Nomina  Pension  Debito_aff  VISA  MCard
##   5   1  antig  Nomina  Pension  Debito_aff  VISA  MCard

churn_tidy <- churn %>%
  separate(Baja, into = c("Baja_Rem","Baja"), sep = " ", extra = "merge",
fill = "left") %>%
  separate(edatcat, into = c("edatcat_Rem",
"edatcat","edatcat_Rem2","edatcat_Rem3"), sep = "([\\ \\.\\.])", extra =
"merge", fill = "right") %>%
  separate(Nomina, into = c("Nomina_Rem", "Nomina"), sep = " ", extra =
"merge", fill = "left") %>%
  separate(Pension, into = c("Pension_Rem", "Pension"), sep = " ", extra
= "merge", fill = "left") %>%
  separate(Debito_normal, into =
c("Debito_normal_Rem","Debito_normal_Rem2", "Debito_normal"), sep = "([\\
\\ ])", extra = "merge", fill = "left") %>%
  separate(Debito_aff, into = c("Debito_aff_Rem","Debito_aff_Rem2",
"Debito_aff"), sep = "([\\ \\. ])", extra = "merge", fill = "left") %>%
  separate(VISA, into = c("VISA_Rem", "VISA"), sep = " ", extra =
"merge", fill = "left") %>%
  separate(VISA_aff, into = c("VISA_aff_Rem","VISA_aff_Rem2",
"VISA_aff"), sep = "([\\ \\. ])", extra = "merge", fill = "left") %>%
  separate(MCard, into = c("MCard_Rem", "MCard"), sep = " ", extra =
"merge", fill = "left") %>%
  separate(Amex, into = c("Amex_Rem", "Amex"), sep = " ", extra =
"merge", fill = "left") %>%
  separate(dif_resid, into = c("dif_resid_Rem","dif_resid_Rem2",
"dif_resid"), sep = "([\\ \\. ])", extra = "merge", fill = "left") %>%
  transform(sexo = ifelse(.$sexo == "No informado", "MUJER", "HOMBRE"))
%>%
  select(-c("Baja_Rem",
"edatcat_Rem","edatcat_Rem2","edatcat_Rem3","Nomina_Rem","Pension_Rem","D
ebito_normal_Rem","Debito_normal_Rem2","Debito_aff_Rem","Debito_aff_Rem2"
,"VISA_Rem","VISA_aff_Rem","VISA_aff_Rem2","MCard_Rem","Amex_Rem","dif_re
sid_Rem","dif_resid_Rem2"))

#Comandos para la generación de training data, con 2/3 (67% aprox) de
observaciones random
n <- nrow(churn_tidy)
set.seed(7)
trainingdata <- sample(1:n,round(0.67*n))

#Comandos para la generación del arbol CART
set.seed(27)
arbbaja <- rpart(Baja~.,data = churn_tidy[trainingdata,],control =
rpart.control(cp=0.0001,xval = 10))
printcp(arbbaja)
```

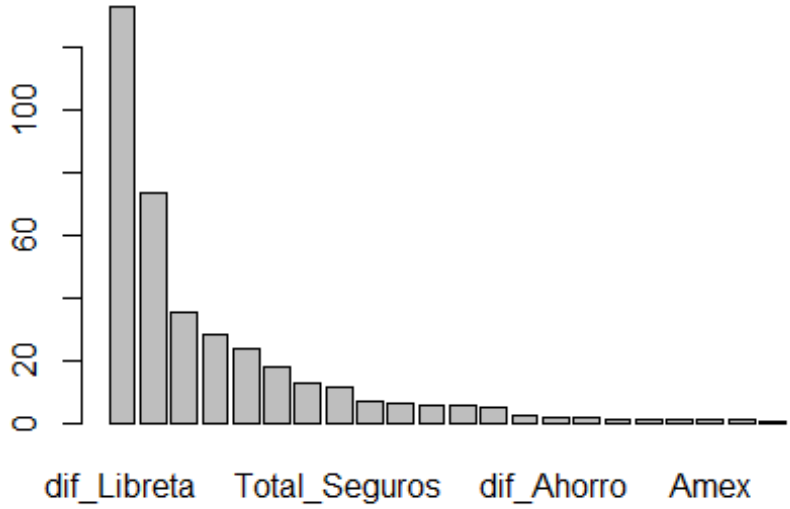
```
##
## Classification tree:
## rpart(formula = Baja ~ ., data = churn_tidy[trainingdata, ],
##     control = rpart.control(cp = 1e-04, xval = 10))
##
## Variables actually used in tree construction:
##  [1] antig            Debito_aff       dif_CC           dif_Hipoteca
##  [5] dif_Libreta      dif_Plazo        dif_Seguros      edatcat
##  [9] oper_caj_Libreta oper_ven_Libreta sexo             Total_activo
## [13] Total_Inversion  Total_Plazo      Total_Seguros    Total_Vista
## [17] VISA
##
## Root node error: 663/1340 = 0.49478
##
## n= 1340
##
##            CP nsplit rel error  xerror     xstd
## 1  0.31070890      0   1.00000 1.04223 0.027593
## 2  0.11010558      1   0.68929 0.70890 0.026348
## 3  0.04374057      2   0.57919 0.66365 0.025929
## 4  0.03167421      3   0.53544 0.61840 0.025443
## 5  0.01960784      4   0.50377 0.56410 0.024766
## 6  0.01809955      5   0.48416 0.55354 0.024622
## 7  0.01659125      7   0.44796 0.53394 0.024343
## 8  0.01282051      8   0.43137 0.52187 0.024164
## 9  0.01055807     10   0.40573 0.51584 0.024072
## 10 0.00904977     11   0.39517 0.51433 0.024049
## 11 0.00754148     12   0.38612 0.49623 0.023763
## 12 0.00452489     14   0.37104 0.49774 0.023788
## 13 0.00377074     19   0.34691 0.48115 0.023515
## 14 0.00251383     23   0.32881 0.48718 0.023615
## 15 0.00245098     28   0.31373 0.49623 0.023763
## 16 0.00226244     38   0.28808 0.49623 0.023763
## 17 0.00150830     46   0.26244 0.49925 0.023812
## 18 0.00075415     48   0.25943 0.50980 0.023979
## 19 0.00010000     50   0.25792 0.50377 0.023884
```

4. Determine ahora el árbol óptimo y su valor del complexity parameter (cp). Diga cuales son las variables más importantes en la definición del árbol óptimo.

```
plotcp(arbbaja)
```



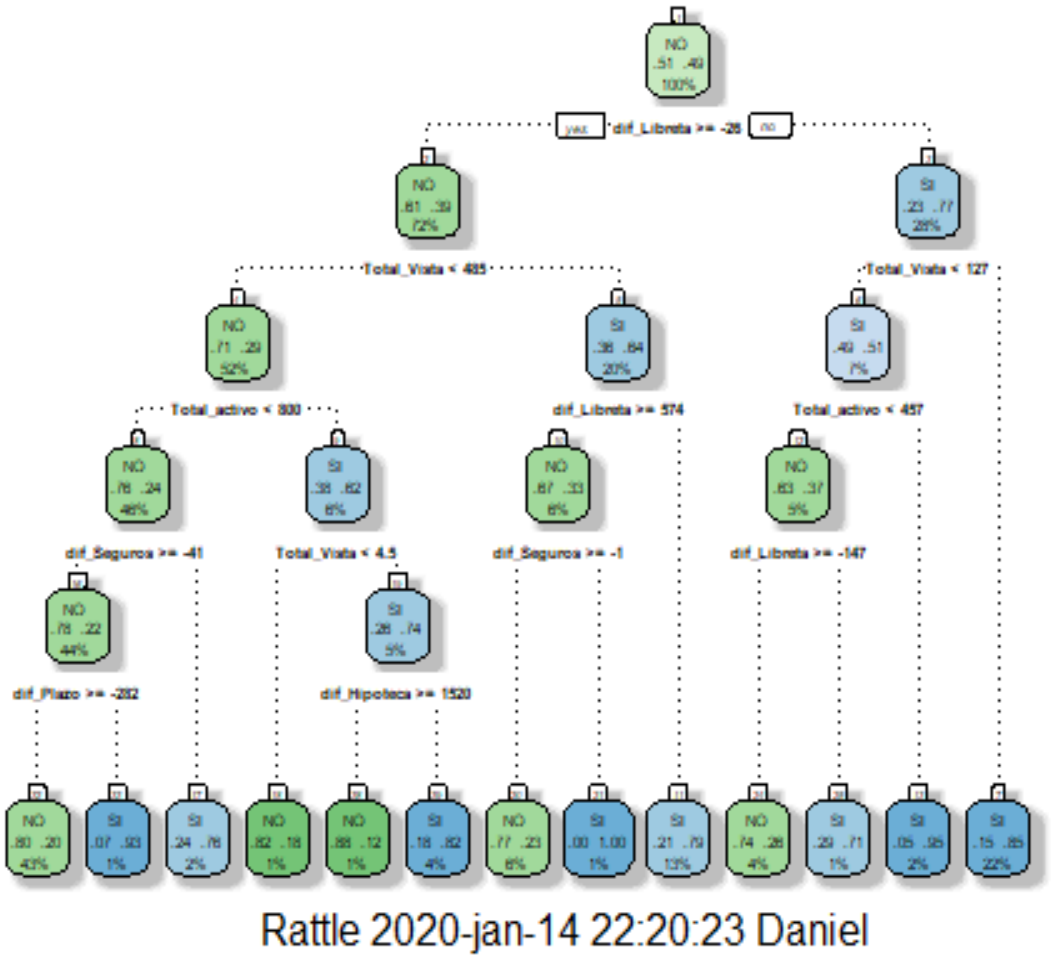

```r
arbbaja$cptable <- as.data.frame(arbbaja$cptable)
#Descubrir el index del arbol con menos calidad
indMenorXError <- which.min(arbbaja$cptable$xerror)

#Grabar el registro en una nueva variable
xerror <- arbbaja$cptable$xerror[indMenorXError]
#Coger su respectiva standard variation
sd <- arbbaja$cptable$xstd[indMenorXError]

#A través de comando loop, descubrir el primer valor que es más pequeño
que el XError + 1 standard deviation
i=1
while(arbbaja$cptable$xerror[i] > (xerror+sd))
  i = i + 1

#Valor de Complexity Parameter
optimalTreeCP <- arbbaja$cptable$CP[i]
optimalTreeCP

## [1] 0.007541478

#Considerando el valor de CP que hemos atribuido anteriormente, tenemos
que descubrir las variables mas importantes
p1 <- prune(arbbaja,cp = optimalTreeCP)
barplot(p1$variable.importance)
```



R.: El arbol optimo que hemos obtenido es de numero 11 de la cptable con complexity parameter de 0.007541478. Las variables que han sido mas importantes en la definición del arbol optimo han sido Total_vista (que no se ve a través del grafico), dif_Libreta.

5. Represente gráficamente el árbol óptimo y liste sus reglas de decisión.

```
#Ambos plots representan graficamente el arbol optimo. En este ejercicio
hemos elegido utilizar el fancyRpartPlot del paquete rattle
#library(rpart.plot)
#rpart.plot(p1)

library(rattle)

## Warning: package 'rattle' was built under R version 3.6.2

## Rattle: A free graphical interface for data science with R.
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

fancyRpartPlot(p1)
```




```
asRules(p1)

##
##  Rule number: 21 [Baja=SI cover=12 (1%) prob=1.00]
##    dif_Libreta>=-25.54
##    Total_Vista>=484.5
##    dif_Libreta>=574.2
##    dif_Seguros< -1
##
##  Rule number: 13 [Baja=SI cover=21 (2%) prob=0.95]
##    dif_Libreta< -25.54
##    Total_Vista< 126.5
##    Total_activo>=457
##
##  Rule number: 33 [Baja=SI cover=14 (1%) prob=0.93]
##    dif_Libreta>=-25.54
##    Total_Vista< 484.5
##    Total_activo< 800
##    dif_Seguros>=-40.64
```

```
##    dif_Plazo< -282.5
##
##  Rule number: 7 [Baja=SI cover=292 (22%) prob=0.85]
##    dif_Libreta< -25.54
##    Total_Vista>=126.5
##
##  Rule number: 39 [Baja=SI cover=60 (4%) prob=0.82]
##    dif_Libreta>=-25.54
##    Total_Vista< 484.5
##    Total_activo>=800
##    Total_Vista>=4.5
##    dif_Hipoteca< 1520
##
##  Rule number: 11 [Baja=SI cover=178 (13%) prob=0.79]
##    dif_Libreta>=-25.54
##    Total_Vista>=484.5
##    dif_Libreta< 574.2
##
##  Rule number: 17 [Baja=SI cover=25 (2%) prob=0.76]
##    dif_Libreta>=-25.54
##    Total_Vista< 484.5
##    Total_activo< 800
##    dif_Seguros< -40.64
##
##  Rule number: 25 [Baja=SI cover=17 (1%) prob=0.71]
##    dif_Libreta< -25.54
##    Total_Vista< 126.5
##    Total_activo< 457
##    dif_Libreta< -146.9
##
##  Rule number: 24 [Baja=NO cover=50 (4%) prob=0.26]
##    dif_Libreta< -25.54
##    Total_Vista< 126.5
##    Total_activo< 457
##    dif_Libreta>=-146.9
##
##  Rule number: 20 [Baja=NO cover=75 (6%) prob=0.23]
##    dif_Libreta>=-25.54
##    Total_Vista>=484.5
##    dif_Libreta>=574.2
##    dif_Seguros>=-1
##
##  Rule number: 32 [Baja=NO cover=571 (43%) prob=0.20]
##    dif_Libreta>=-25.54
##    Total_Vista< 484.5
##    Total_activo< 800
##    dif_Seguros>=-40.64
##    dif_Plazo>=-282.5
##
##  Rule number: 18 [Baja=NO cover=17 (1%) prob=0.18]
```

```
##    dif_Libreta>=-25.54
##    Total_Vista< 484.5
##    Total_activo>=800
##    Total_Vista< 4.5
##
##  Rule number: 38 [Baja=NO cover=8 (1%) prob=0.12]
##    dif_Libreta>=-25.54
##    Total_Vista< 484.5
##    Total_activo>=800
##    Total_Vista>=4.5
##    dif_Hipoteca>=1520
```

6. Las probabilidades de baja no están por fortuna equidistribuidas, sino que la probabilidad de baja es muy inferior (un 5%). Exporte a Excel la tabla de resultados por hoja y pondere estos resultados de acuerdo con las probabilidades a priori mencionadas. Obsérvese que en este caso no utilizamos una muestra test de validación del árbol obtenido (en general deberíamos obtener la predicción del árbol en una muestra independiente (test) y validar la calidad del árbol con los resultados obtenidos en esta muestra test).

```
leaf <- subset(p1$frame, var=="<leaf>",select=c(n,yval2))
numLeaf <- row.names(leaf)
leaf <- data.frame(leaf$n, leaf$yval2)
names(leaf) <-
c("n_train","class_train","n1_train","n2_train","p1_train","p2_train","pr
obnode_train")
row.names(leaf) <- numLeaf
leaf <- leaf[order(-leaf$p2_train),]

leaf$cum_n1 <- cumsum(leaf$n1_train)/sum(leaf$n1_train)
leaf$cum_n2 <- cumsum(leaf$n2_train)/sum(leaf$n2_train)
leaf$dif_cum <- leaf$cum_n2 - leaf$cum_n1

print(leaf)
```

```
##    n_train class_train n1_train n2_train   p1_train  p2_train
## 21      12           2        0       12 0.00000000 1.0000000
## 13      21           2        1       20 0.04761905 0.9523810
## 33      14           2        1       13 0.07142857 0.9285714
## 7      292           2       44      248 0.15068493 0.8493151
## 39      60           2       11       49 0.18333333 0.8166667
## 11     178           2       38      140 0.21348315 0.7865169
## 17      25           2        6       19 0.24000000 0.7600000
## 25      17           2        5       12 0.29411765 0.7058824
## 24      50           1       37       13 0.74000000 0.2600000
```

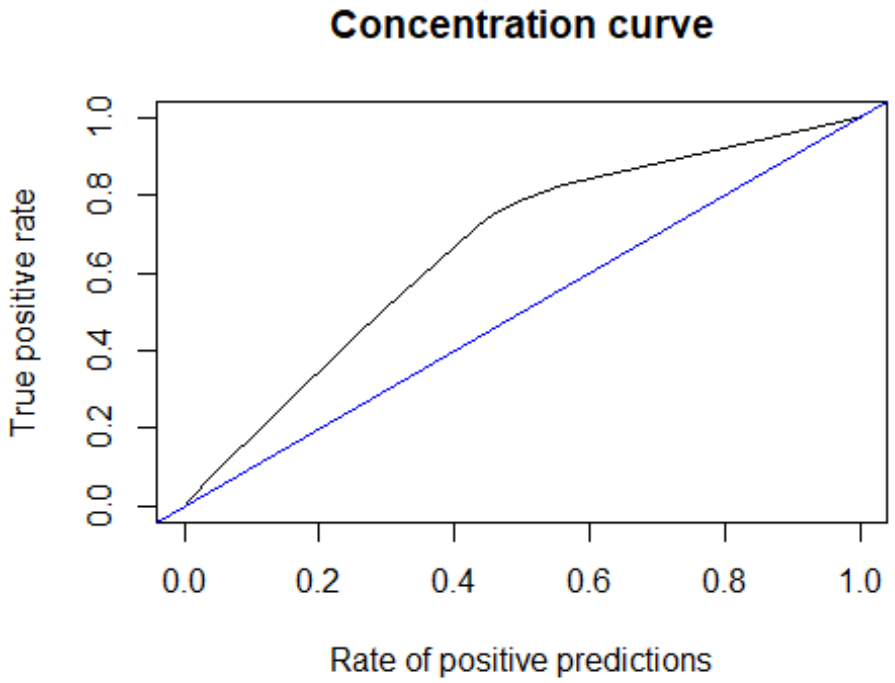
```
## 20      75           1       58       17 0.77333333 0.2266667
## 32     571           1      455      116 0.79684764 0.2031524
## 18      17           1       14        3 0.82352941 0.1764706
## 38       8           1        7        1 0.87500000 0.1250000
##    probnode_train      cum_n1     cum_n2     dif_cum
## 21    0.008955224 0.000000000 0.01809955 0.018099548
## 13    0.015671642 0.001477105 0.04826546 0.046788355
## 33    0.010447761 0.002954210 0.06787330 0.064919093
## 7     0.217910448 0.067946824 0.44193062 0.373983794
## 39    0.044776119 0.084194978 0.51583710 0.431642126
## 11    0.132835821 0.140324963 0.72699849 0.586673529
## 17    0.018656716 0.149187592 0.75565611 0.606468516
## 25    0.012686567 0.156573117 0.77375566 0.617182539
## 24    0.037313433 0.211225997 0.79336350 0.582137502
## 20    0.055970149 0.296898080 0.81900452 0.522106445
## 32    0.426119403 0.968980798 0.99396682 0.024986020
## 18    0.012686567 0.989660266 0.99849170 0.008831438
## 38    0.005970149 1.000000000 1.00000000 0.000000000
```

```
tab_results = data.frame(matrix(NA, nrow=nrow(leaf), ncol=4))
row.names(tab_results) = row.names(leaf)
tab_results[,1] = leaf$n_train + leaf$n_train
tab_results[,2] = leaf$n1_train + leaf$n1_train
tab_results[,3] = leaf$n2_train + leaf$n2_train
tab_results[,4] = tab_results[,3]/tab_results[,1]
names(tab_results) = c("n","n1","n2","p2")
tab_results = tab_results[order(-tab_results$p2),]
#print(leaf)

Sys.setenv(JAVA_HOME='C:\\Program Files\\Java\\jdk1.8.0_231\\jre')
library(xlsx)
write.xlsx(tab_results, "tab_results.xlsx")
```

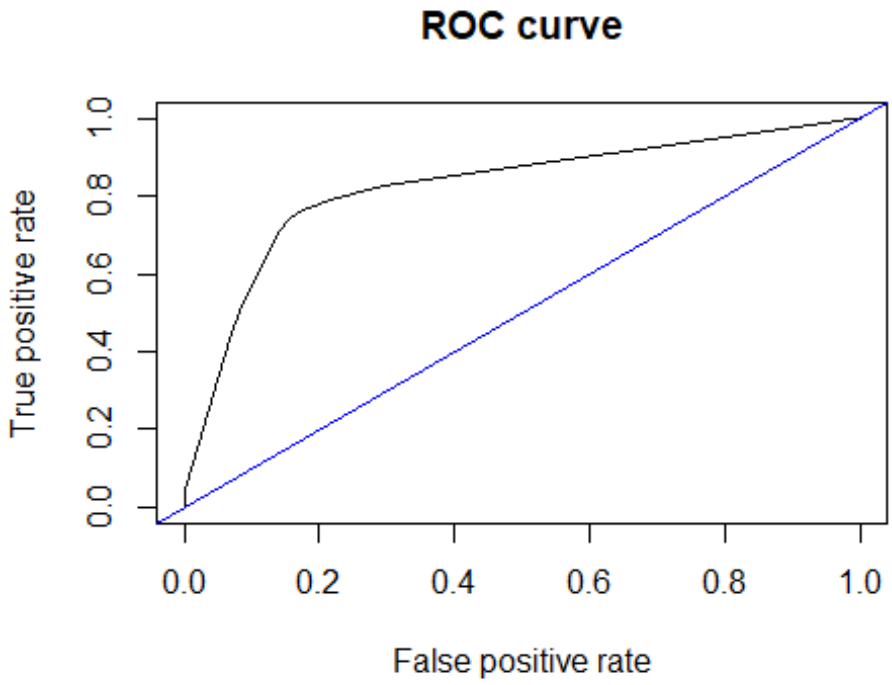
7. Obtenga gráficamente las curvas de concentración y ROC correspondientes.

```
pred_learn <- as.data.frame(predict(p1,
data=churn_tidy[trainingdata,],type="prob"))
pred_test  <- as.data.frame(predict(p1, newdata=churn_tidy[-
trainingdata,],type="prob"))

library(ROCR)

## Warning: package 'ROCR' was built under R version 3.6.2

## Loading required package: gplots

##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess

pred <- prediction(c(pred_learn$SI,pred_test$SI),
c(churn_tidy$Baja[trainingdata],churn_tidy$Baja[-trainingdata]))
con <- performance(pred,measure="tpr",x.measure="rpp")
plot(con, main="Concentration curve")
abline(0,1,col="blue")
```




```
roc <- performance(pred,measure="tpr",x.measure="fpr")
plot(roc, main="ROC curve")
abline(0,1,col="blue")
```




```
auc.tmp <- performance(pred,"auc")
auc <- as.numeric(auc.tmp@y.values)

#Valor del area entre la curva ROC y la linea diagonal
auc
```

```
## [1] 0.823641
```

8. Decida un umbral de decisión para la predicción de “baja” y obtenga el “error_rate”, la precisión en la predicción positiva, la precisión en la predicción negativa, el promedio de ambas precisiones y el Recall asociado al umbral escogido.

Resp.: Los resultados pueden ser vistos en el archivo adjunto “Exercise-CART-08-01-2020.xlsx”