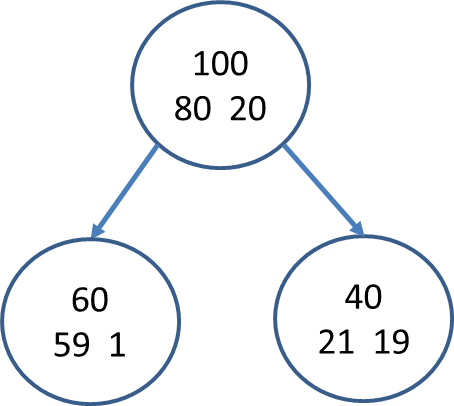
**SESIÓN 1: ÁRBOLES DE DECISIÓN**

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.  
Teniendo en cuenta que la raíz is No, el nodo hoja izquierdo es No(60), el derecho Si(40), y lo mismo con los valores, No = 80,59,21 y Si = 20,1,19, procedemos a hacer los cálculos.

Impureza nodo 1 = (80/100)\*(1-(80/100)) + (20/100)\*(1-(20/100)) = 0.32

Impureza nodo 2 = (59/60)\*(1-(59/60)) + (1/60)\*(1-(1/60)) = 0.032

Impureza nodo 3 = (21/40)\*(1-(21/40)) + (19/40)\*(1-(19/40)) = 0.498

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

la mal clasificación (o missclassification o r(t)) de cada nodo se calcula con r(t)=1−Max(p(j/t)) así que:

r1 <- 1-(20/100)

r2 <- 1-(1/60)

r3 <- 1-(21/40)

Y la fórmula de R(t) nos dice que es igual a:

Rt <- (((0.6)\*r2 + (0.4)\*r3)/r1)\*100

Lo que es igual a 97.5

1. 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).

> d <- read.csv(file="churn.txt", sep=" ")

> m<-rpart(Baja ~ ., data=d, control=rpart.control( cp = 0.0001, xval=10))

> printcp(m)

Classification tree:

rpart(formula = Baja ~ ., data = d, control = rpart.control(cp = 1e-04,

xval = 10))

Variables actually used in tree construction:

[1] Debito\_aff Nomina Pension Total\_Plazo Total\_Seguros Total\_Vista

[7] Total\_activo antig dif\_CC dif\_Hipoteca dif\_Largo\_plazo dif\_Libreta

[13] dif\_Planes\_pension dif\_Plazo dif\_Prest\_personales dif\_Seguros edatcat oper\_ven\_Libreta

[19] sexo

Root node error: 1000/2000 = 0.5

n= 2000

CP nsplit rel error xerror xstd

1 0.35300000 0 1.000 1.086 0.022278

2 0.07400000 1 0.647 0.693 0.021281

3 0.02800000 2 0.573 0.591 0.020405

4 0.02450000 3 0.545 0.594 0.020435

5 0.01800000 5 0.496 0.540 0.019854

6 0.01600000 6 0.478 0.521 0.019629

7 0.01500000 7 0.462 0.500 0.019365

8 0.01400000 8 0.447 0.500 0.019365

9 0.01100000 9 0.433 0.482 0.019127

10 0.01000000 10 0.422 0.480 0.019100

11 0.00900000 13 0.392 0.469 0.018948

12 0.00800000 14 0.383 0.468 0.018934

13 0.00700000 15 0.375 0.455 0.018748

14 0.00650000 18 0.354 0.453 0.018719

15 0.00500000 20 0.341 0.448 0.018645

16 0.00450000 23 0.326 0.446 0.018616

17 0.00300000 27 0.308 0.427 0.018326

18 0.00266667 31 0.293 0.415 0.018135

19 0.00250000 34 0.285 0.413 0.018103

20 0.00200000 37 0.276 0.412 0.018087

21 0.00175000 42 0.266 0.412 0.018087

22 0.00166667 46 0.259 0.412 0.018087

23 0.00150000 49 0.254 0.408 0.018021

24 0.00100000 51 0.251 0.412 0.018087

25 0.00050000 59 0.243 0.427 0.018326

26 0.00033333 61 0.242 0.432 0.018403

27 0.00010000 67 0.240 0.435 0.018450

1. 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.

> m$cptable = as.data.frame(m$cptable)

> ind = which.min(m$cptable$xerror)

> xerr <- m$cptable$xerror[ind]

> xstd <- m$cptable$xstd[ind]

> i=1

> while (m$cptable$xerror[i] > xerr+xstd) i = i+1

> alfa = m$cptable$CP[i]

> m1 <- prune(m,cp=alfa)

> m1

n= 2000

node), split, n, loss, yval, (yprob)

\* denotes terminal node

1) root 2000 1000 Baja NO (0.50000000 0.50000000)

2) Total\_Vista< 327.5 1191 419 Baja NO (0.64819479 0.35180521)

4) dif\_Libreta>=-22.045 911 242 Baja NO (0.73435785 0.26564215)

8) Total\_activo< 793 793 169 Baja NO (0.78688525 0.21311475)

...

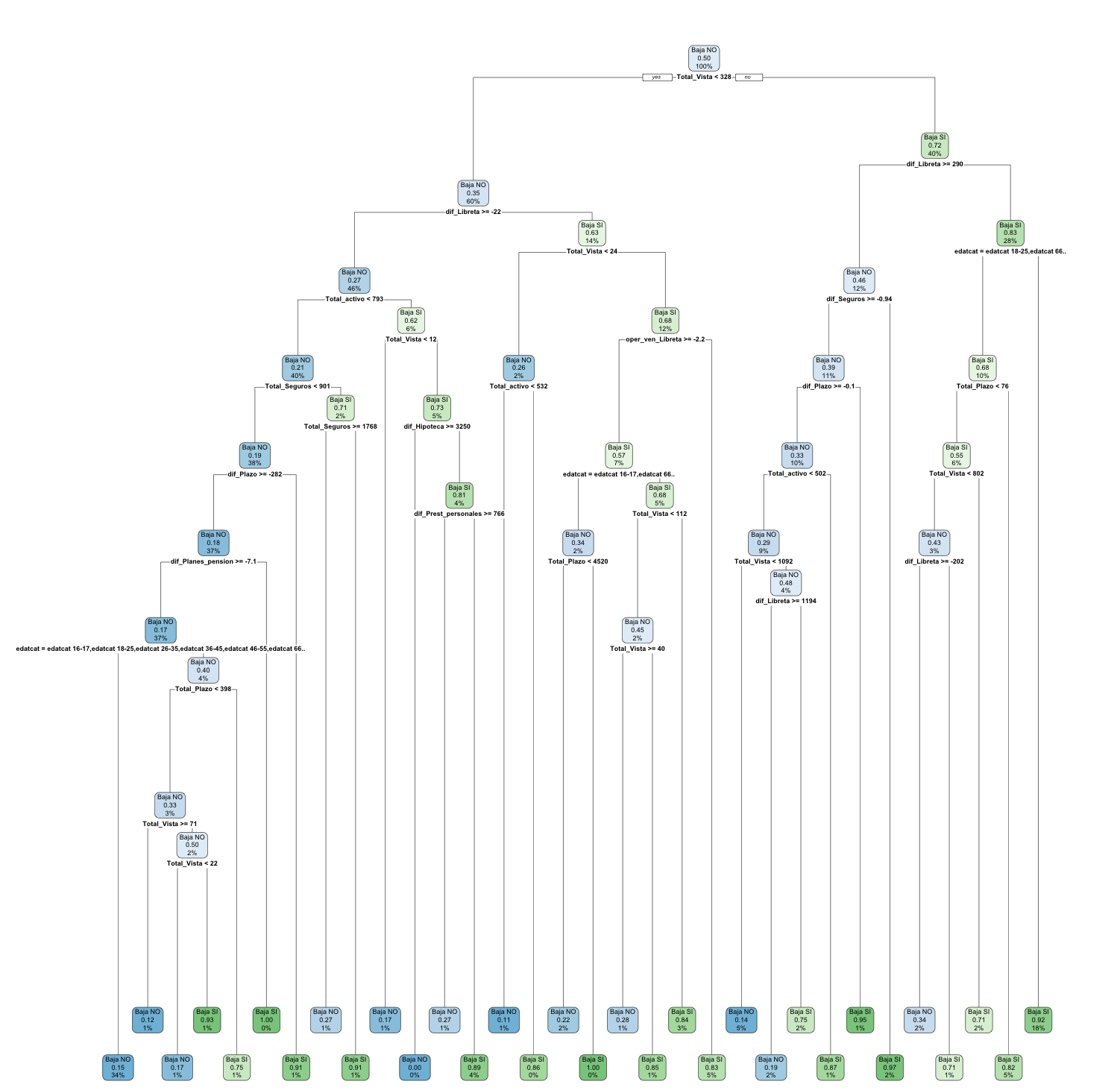
Según m1, la variable con más importancia es Total\_Vista, seguida de dif\_Libreta y Total\_activo. El análisis da un conjunto de variables más grande, pero estas son las primeras y por las que se hacen las divisiones principales desde la raíz del árbol.

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

> install.packages("rpart.plot")

> library("rpart.plot")

> rpart.plot(m1)



library("rattle")

asRules(m1)

Rule number: 89 [Baja=Baja SI cover=7 (0%) prob=1.00]

Total\_Vista< 327.5

dif\_Libreta< -22.05

Total\_Vista>=23.5

oper\_ven\_Libreta>=-2.25

edatcat=edatcat 16-17,edatcat 66..

Total\_Plazo>=4520

Rule number: 65 [Baja=Baja SI cover=8 (0%) prob=1.00]

Total\_Vista< 327.5

dif\_Libreta>=-22.05

Total\_activo< 793

Total\_Seguros< 901

dif\_Plazo>=-282.5

dif\_Planes\_pension< -7.08

Rule number: 13 [Baja=Baja SI cover=32 (2%) prob=0.97]

Total\_Vista>=327.5

dif\_Libreta>=289.9

dif\_Seguros< -0.935

Rule number: 25 [Baja=Baja SI cover=20 (1%) prob=0.95]

Total\_Vista>=327.5

dif\_Libreta>=289.9

dif\_Seguros>=-0.935

dif\_Plazo< -0.1

Rule number: 1035 [Baja=Baja SI cover=14 (1%) prob=0.93]

Total\_Vista< 327.5

dif\_Libreta>=-22.05

Total\_activo< 793

Total\_Seguros< 901

dif\_Plazo>=-282.5

dif\_Planes\_pension>=-7.08

edatcat=edatcat 56-65

Total\_Plazo< 398

Total\_Vista< 71

Total\_Vista>=21.5

...

Y así es como se pueden imprimir todas las reglas. Faltan muchas más que me dejé por copiar.

En ellas se puede ver como cual es el tipo del nodo/hoja, de que pasos se ha llegado hasta ahí y cual es la

1. 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).

> m1.leaf=subset(m1$frame, var=="<leaf>",select=c(n,yval2))

> num\_leaf = row.names(m1.leaf)

> m1.leaf=data.frame(m1.leaf$n,m1.leaf$yval2)

> names(m1.leaf) = c("n\_train","class\_train","n1\_train","n2\_train","p1\_train","p2\_train","probnode\_train")

> row.names(m1.leaf) = num\_leaf

> m1.leaf=m1.leaf[order(-m1.leaf$p2\_train),] # ordering by decreasing > positive probabilities

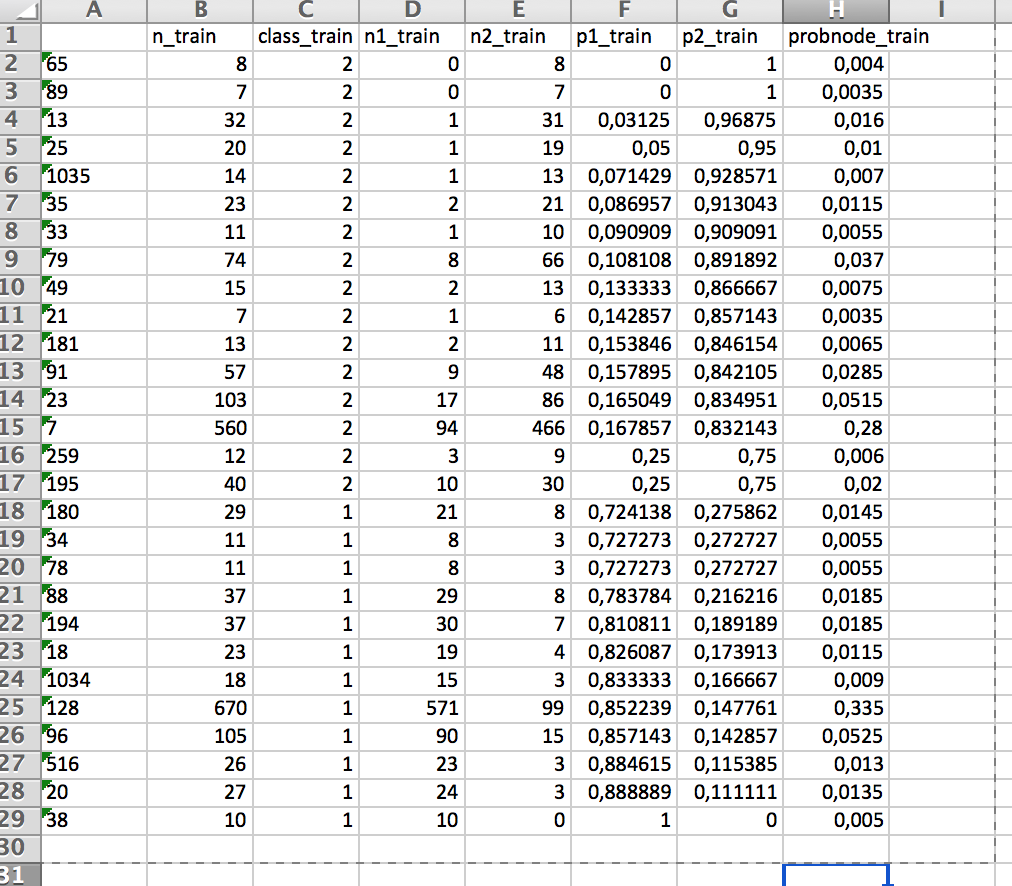
> install.packages('xlsx')

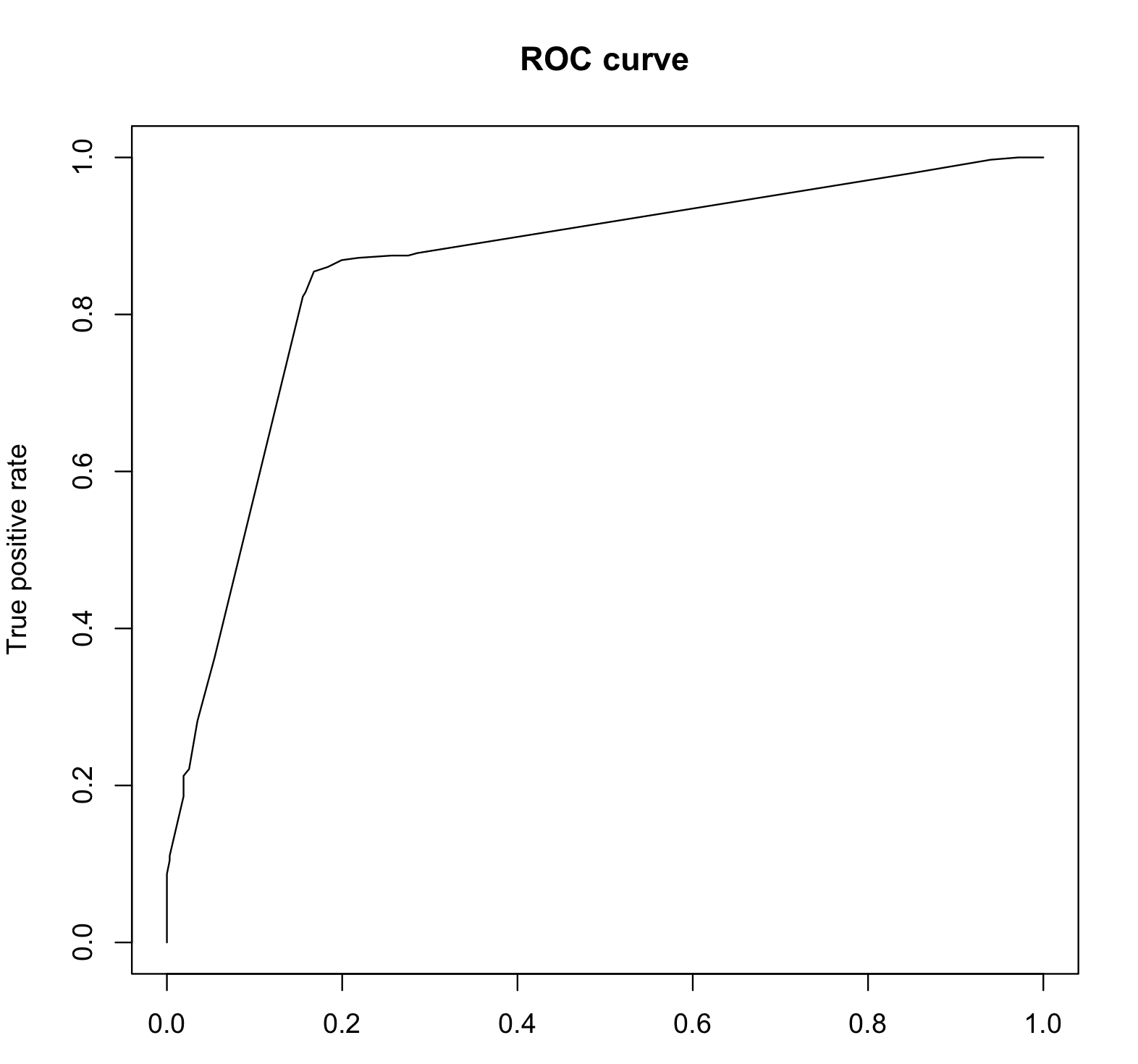
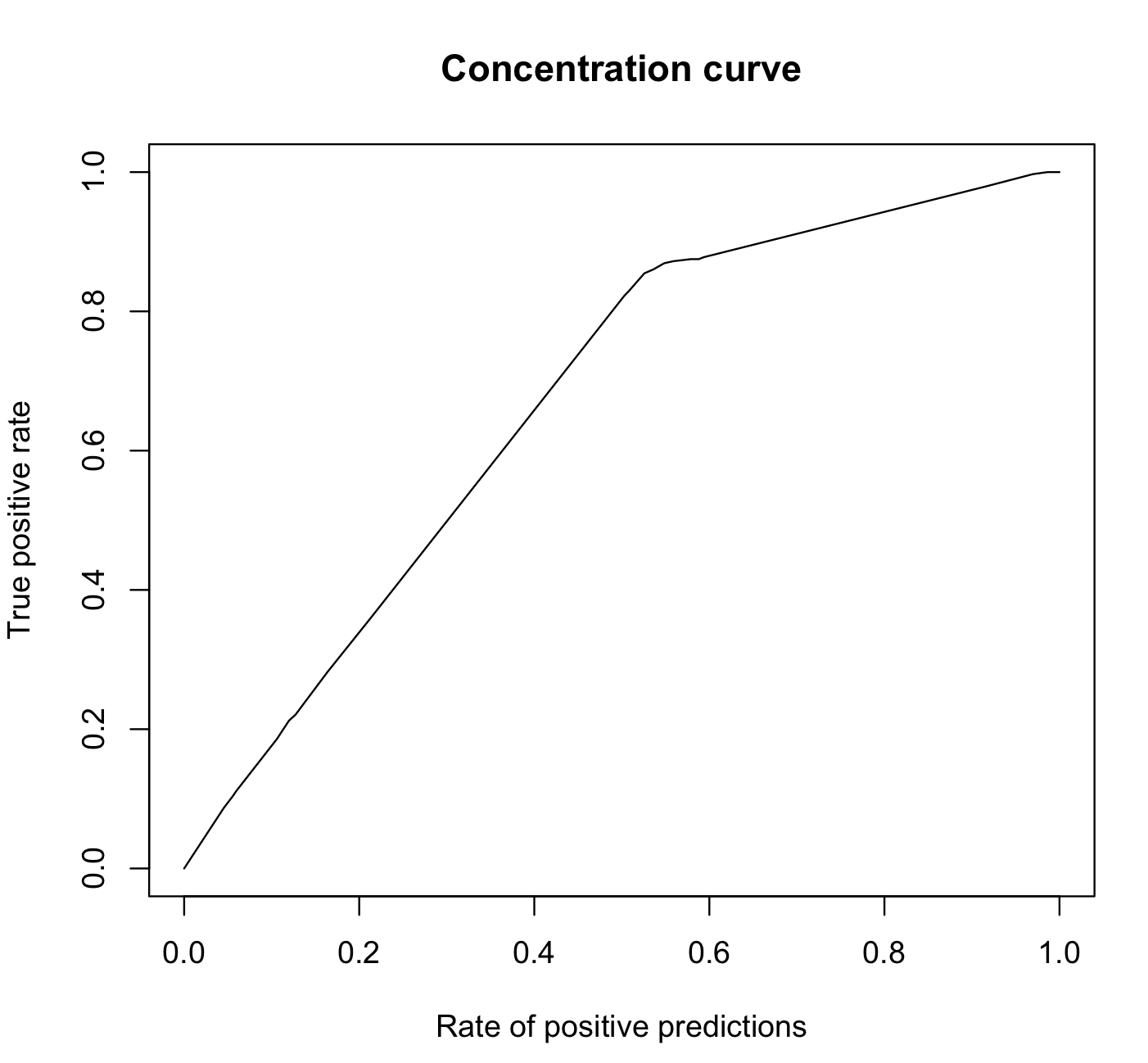
> install.packages('rJava')

> install.packages('xlsxjars')

> library(xlsx)

write.xlsx(m1.leaf, "resultats.xlsx")



1. Obtenga gráficamente las curvas de concentración y ROC correspondientes.  
   > pred\_test = as.data.frame(predict(m1, newdata=d[- learn,],type="prob"))  
   > pred <- prediction(pred\_test$"Baja SI", d$Baja[-learn])  
   > roc <- performance(pred,measure="tpr",x.measure="fpr")  
   > plot(roc, main="ROC curve")  
     
   > con <- performance(pred,measure="tpr",x.measure="rpp")  
   > plot(con, main="Concentration curve")  
   
2. 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.