Unidad 4 - Módulo 1

Pablo Anicich

23 de julio de 2016

Ejercicio árbol 1 resuelto

Lo que primero que hago es setear el directorio de trabajo para leer el archivo Arbol1.csv:

```
setwd("C:/Documentos/Diplomatura_en_BI_UTN/Modulo_1/Unidad_4/");
arbol1 <- read.csv("Arbol1.csv",
                   header=TRUE, sep=";", na.strings=c("NA"));
```

Para familiarme con los datos, extraigo la info sobre la estructura de los datos en el dataframe "arbol1".

```
str(arbol1)
```

```
## 'data.frame':    10000 obs. of  7 variables:
##  $ Atributo.1: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Atributo.2: int  0 0 1 0 1 0 0 0 0 0 ...
##  $ Atributo.3: int  1 0 0 1 0 0 0 1 0 0 ...
##  $ Atributo.4: int  1 1 0 0 1 1 0 0 1 1 ...
##  $ Atributo.5: int  0 0 1 0 0 1 1 1 0 0 ...
##  $ Atributo.6: int  1 1 0 1 1 1 1 1 1 1 ...
##  $ Resultado : int  0 0 0 0 0 0 0 0 0 0 ...
```

Son 7 variables con 10000 observaciones. Todas las variables son enteras. Cargo el paquete "rpart" para trabajar con árboles:

```
library(rpart)
```

Para calcular el primer árbol, se pretende explicar la variable "Resultado" en términos de las variables Atributo.i (con $1 \leq i \leq 6$):

```
fit.arbol1 <- rpart(Resultado~Atributo.1+Atributo.2+Atributo.3+Atributo.4
                    +Atributo.5+Atributo.6, method="class", data=arbol1)
```

Chequeo la información disponible sobre "fit.arbol1":

- Usando **printcp**:

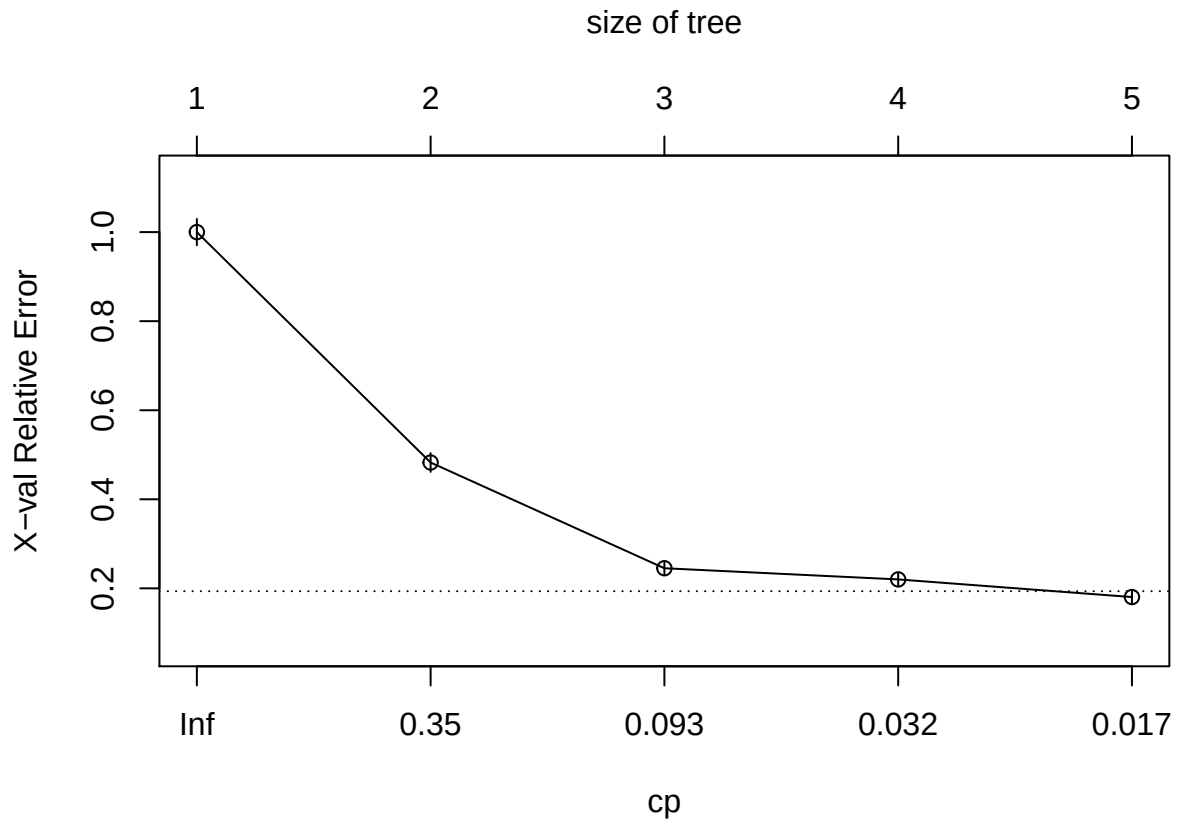
```
printcp(fit.arbol1)
```

```
##
## Classification tree:
## rpart(formula = Resultado ~ Atributo.1 + Atributo.2 + Atributo.3 +
##     Atributo.4 + Atributo.5 + Atributo.6, data = arbol1, method = "class")
##
## Variables actually used in tree construction:
```

```
## [1] Atributo.1 Atributo.2 Atributo.5 Atributo.6
##
## Root node error: 1036/10000 = 0.1036
##
## n= 10000
##
##         CP nsplit rel error  xerror     xstd
## 1 0.517375      0   1.00000 1.00000 0.029415
## 2 0.237452      1   0.48263 0.48263 0.021037
## 3 0.036680      2   0.24517 0.24517 0.015187
## 4 0.027992      3   0.20849 0.22008 0.014408
## 5 0.010000      4   0.18050 0.18050 0.013076
```

- Usando **plotcp**:

```
plotcp(fit.arbol1)
```



* Usando **summary**:

```
summary(fit.arbol1)
```

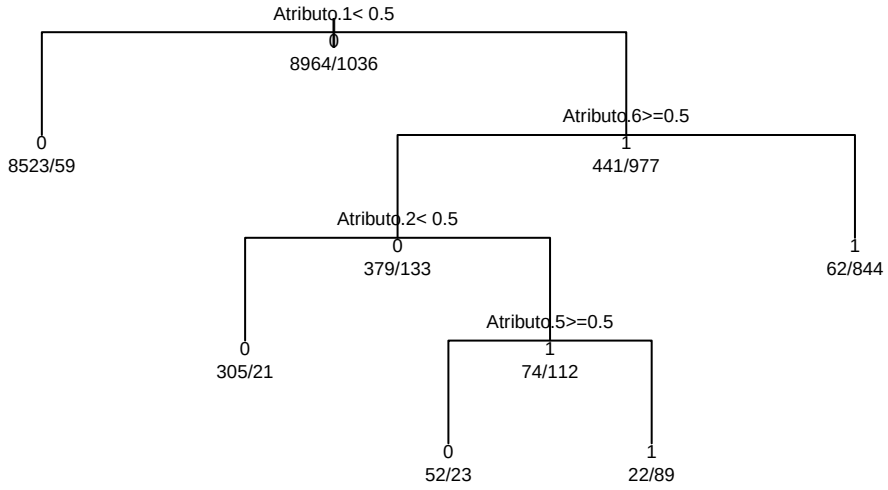
```
## Call:
## rpart(formula = Resultado ~ Atributo.1 + Atributo.2 + Atributo.3 +
##     Atributo.4 + Atributo.5 + Atributo.6, data = arbol1, method = "class")
##   n= 10000
```

```
##
##           CP nsplit rel error    xerror       xstd
## 1 0.51737452      0 1.0000000 1.0000000 0.02941515
## 2 0.23745174      1 0.4826255 0.4826255 0.02103716
## 3 0.03667954      2 0.2451737 0.2451737 0.01518694
## 4 0.02799228      3 0.2084942 0.2200772 0.01440786
## 5 0.01000000      4 0.1805019 0.1805019 0.01307561
##
## Variable importance
## Atributo.1 Atributo.6 Atributo.2 Atributo.5
##         70         18          8          4
##
## Node number 1: 10000 observations,    complexity param=0.5173745
##   predicted class=0  expected loss=0.1036  P(node) =1
##     class counts:  8964  1036
##    probabilities: 0.896 0.104
##   left son=2 (8582 obs) right son=3 (1418 obs)
##   Primary splits:
##       Atributo.1 < 0.5 to the left,  improve=1132.45500, (0 missing)
##       Atributo.6 < 0.5 to the right, improve= 496.41010, (0 missing)
##       Atributo.2 < 0.5 to the left,  improve= 333.96960, (0 missing)
##       Atributo.5 < 0.5 to the right, improve= 204.51190, (0 missing)
##       Atributo.3 < 0.5 to the left,  improve=  31.78468, (0 missing)
##
## Node number 2: 8582 observations
##   predicted class=0  expected loss=0.006874854  P(node) =0.8582
##     class counts:  8523    59
##    probabilities: 0.993 0.007
##
## Node number 3: 1418 observations,    complexity param=0.2374517
##   predicted class=1  expected loss=0.3110014  P(node) =0.1418
##     class counts:   441   977
##    probabilities: 0.311 0.689
##   left son=6 (512 obs) right son=7 (906 obs)
##   Primary splits:
##       Atributo.6 < 0.5 to the right, improve=295.28010, (0 missing)
##       Atributo.2 < 0.5 to the left,  improve=212.39930, (0 missing)
##       Atributo.5 < 0.5 to the right, improve=125.42360, (0 missing)
##       Atributo.3 < 0.5 to the left,  improve= 24.00487, (0 missing)
##       Atributo.4 < 0.5 to the right, improve= 22.29086, (0 missing)
##   Surrogate splits:
##       Atributo.2 < 0.5 to the left,  agree=0.719, adj=0.221, (0 split)
##       Atributo.5 < 0.5 to the right, agree=0.675, adj=0.100, (0 split)
##
## Node number 6: 512 observations,    complexity param=0.03667954
##   predicted class=0  expected loss=0.2597656  P(node) =0.0512
##     class counts:   379   133
##    probabilities: 0.740 0.260
##   left son=12 (326 obs) right son=13 (186 obs)
##   Primary splits:
##       Atributo.2 < 0.5 to the left,  improve=68.489590, (0 missing)
##       Atributo.5 < 0.5 to the right, improve=38.665370, (0 missing)
##       Atributo.4 < 0.5 to the right, improve=11.429670, (0 missing)
##       Atributo.3 < 0.5 to the left,  improve= 5.652017, (0 missing)
```

```
##   Surrogate splits:
##       Atributo.5 < 0.5 to the right, agree=0.67, adj=0.091, (0 split)
##
## Node number 7: 906 observations
##   predicted class=1  expected loss=0.06843267  P(node) =0.0906
##     class counts:    62   844
##    probabilities: 0.068 0.932
##
## Node number 12: 326 observations
##   predicted class=0  expected loss=0.06441718  P(node) =0.0326
##     class counts:   305    21
##    probabilities: 0.936 0.064
##
## Node number 13: 186 observations,    complexity param=0.02799228
##   predicted class=1  expected loss=0.3978495  P(node) =0.0186
##     class counts:    74   112
##    probabilities: 0.398 0.602
##   left son=26 (75 obs) right son=27 (111 obs)
##   Primary splits:
##       Atributo.5 < 0.5 to the right, improve=21.945670, (0 missing)
##       Atributo.4 < 0.5 to the right, improve= 8.103729, (0 missing)
##       Atributo.3 < 0.5 to the left,  improve= 2.256487, (0 missing)
##   Surrogate splits:
##       Atributo.4 < 0.5 to the right, agree=0.613, adj=0.04, (0 split)
##
## Node number 26: 75 observations
##   predicted class=0  expected loss=0.3066667  P(node) =0.0075
##     class counts:    52    23
##    probabilities: 0.693 0.307
##
## Node number 27: 111 observations
##   predicted class=1  expected loss=0.1981982  P(node) =0.0111
##     class counts:    22    89
##    probabilities: 0.198 0.802
```

- Haciendo un plot del árbol de decisión:

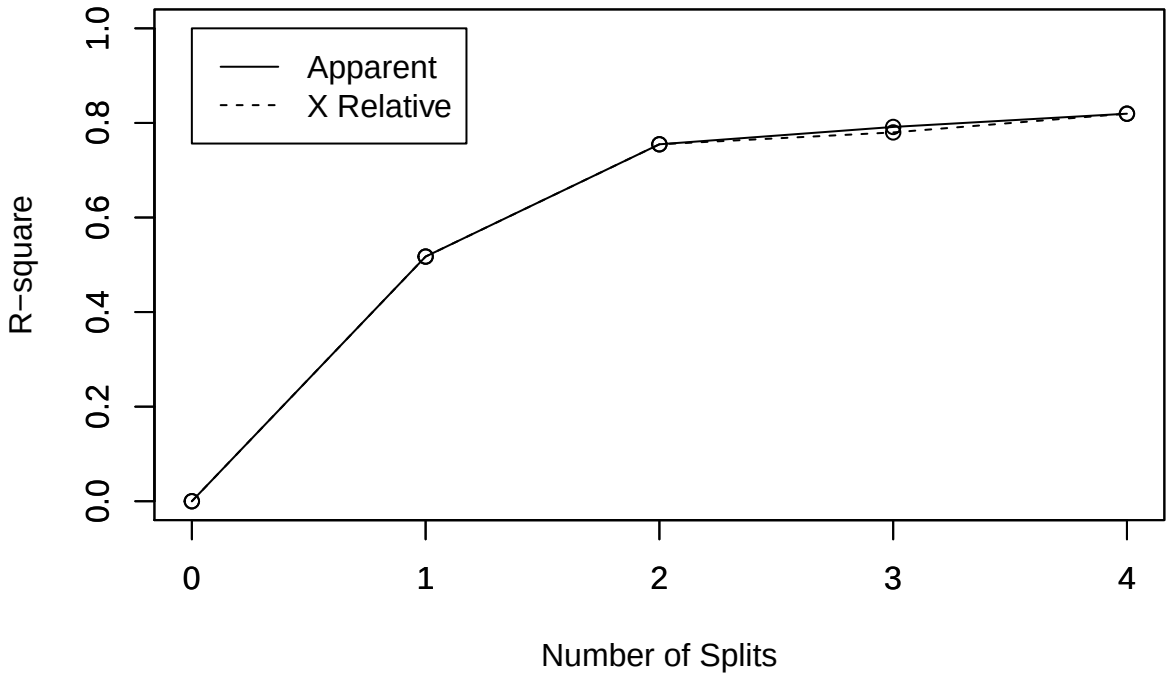
```
plot(fit.arbol1, uniform=TRUE, cex=0.75, margin=0.075)
text(fit.arbol1, use.n=TRUE, all=TRUE, cex=.6)
```

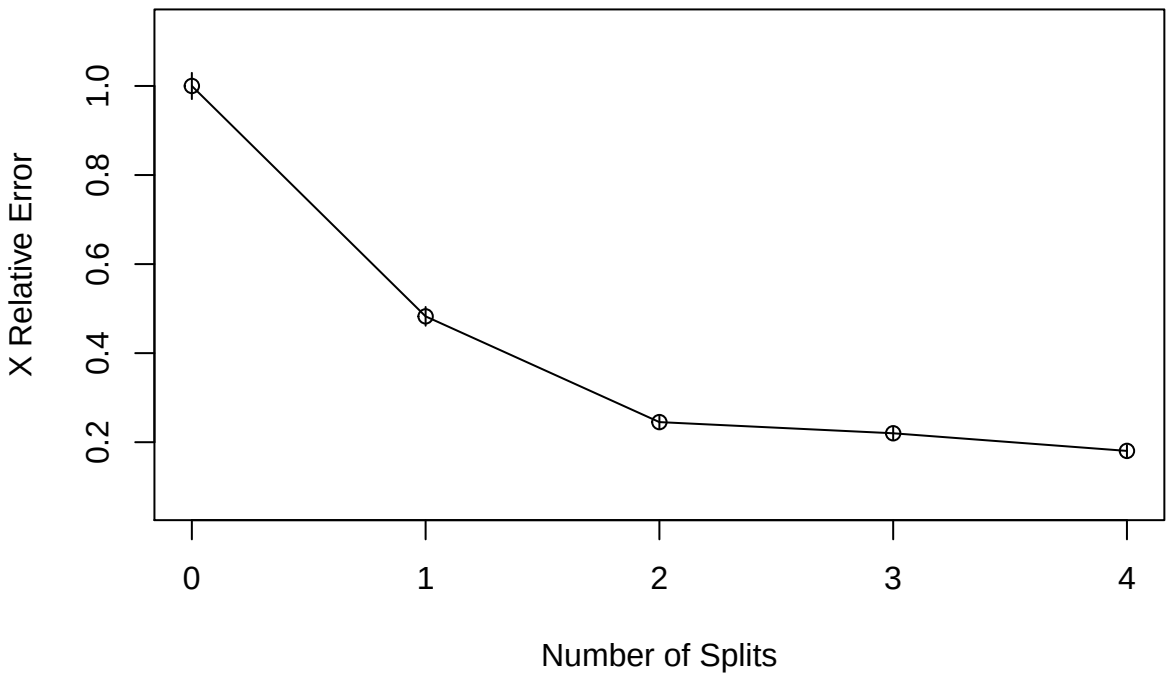


* Usando **rsq.rpart**:

```
rsq.rpart(fit.arbol1)
```

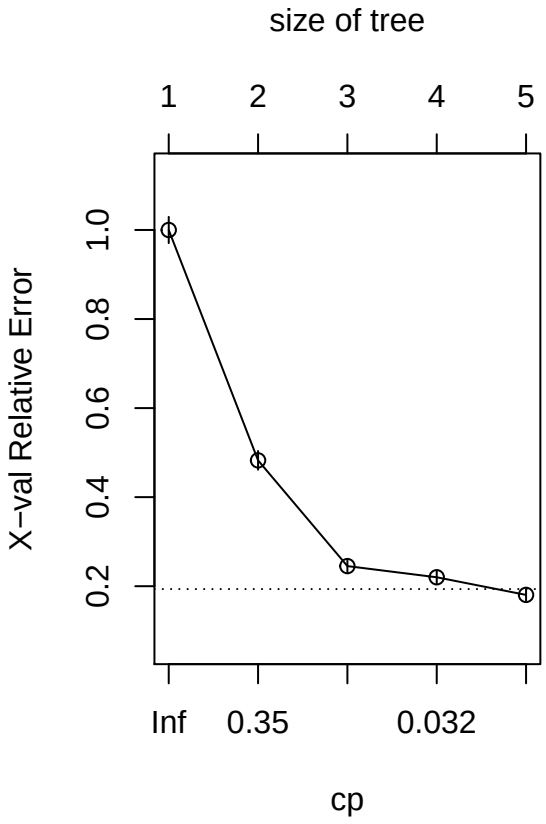
```
##
## Classification tree:
## rpart(formula = Resultado ~ Atributo.1 + Atributo.2 + Atributo.3 +
##     Atributo.4 + Atributo.5 + Atributo.6, data = arbol1, method = "class")
##
## Variables actually used in tree construction:
## [1] Atributo.1 Atributo.2 Atributo.5 Atributo.6
##
## Root node error: 1036/10000 = 0.1036
##
## n= 10000
##
##         CP nsplit rel error  xerror     xstd
## 1 0.517375      0   1.00000 1.00000 0.029415
## 2 0.237452      1   0.48263 0.48263 0.021037
## 3 0.036680      2   0.24517 0.24517 0.015187
## 4 0.027992      3   0.20849 0.22008 0.014408
## 5 0.010000      4   0.18050 0.18050 0.013076

## Warning in rsq.rpart(fit.arbol1): may not be applicable for this method
```

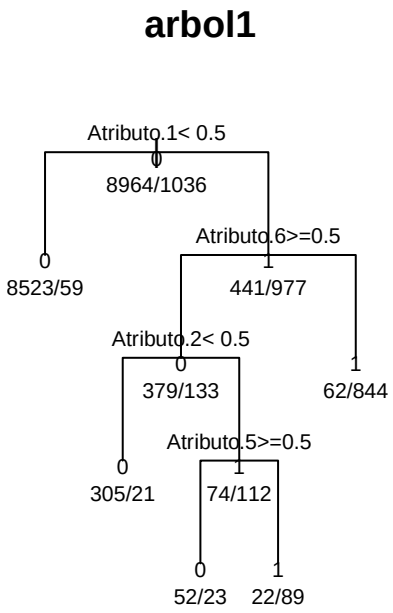

Apparent
X Relative
R−square
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
Number of Splits



Una manera algo más sofisticada de presentar gráficos es dividiendo la pantalla disponible para gráficos usando la función "par" con el parámetro "mfrow". En el caso que sigue se consiguen dos columnas y una fila de gráficos, todos independientes:

```
par(mfrow = c(1,2))
plotcp(fit.arbol1)
plot(fit.arbol1, uniform = TRUE, main = "arbol1", margin=0.075)
text(fit.arbol1, use.n = T, all = T, cex = .7)
```






```
par(mfrow = c(1,1))
```

El último comando resetea al modo normal de operación: un gráfico por pantalla.

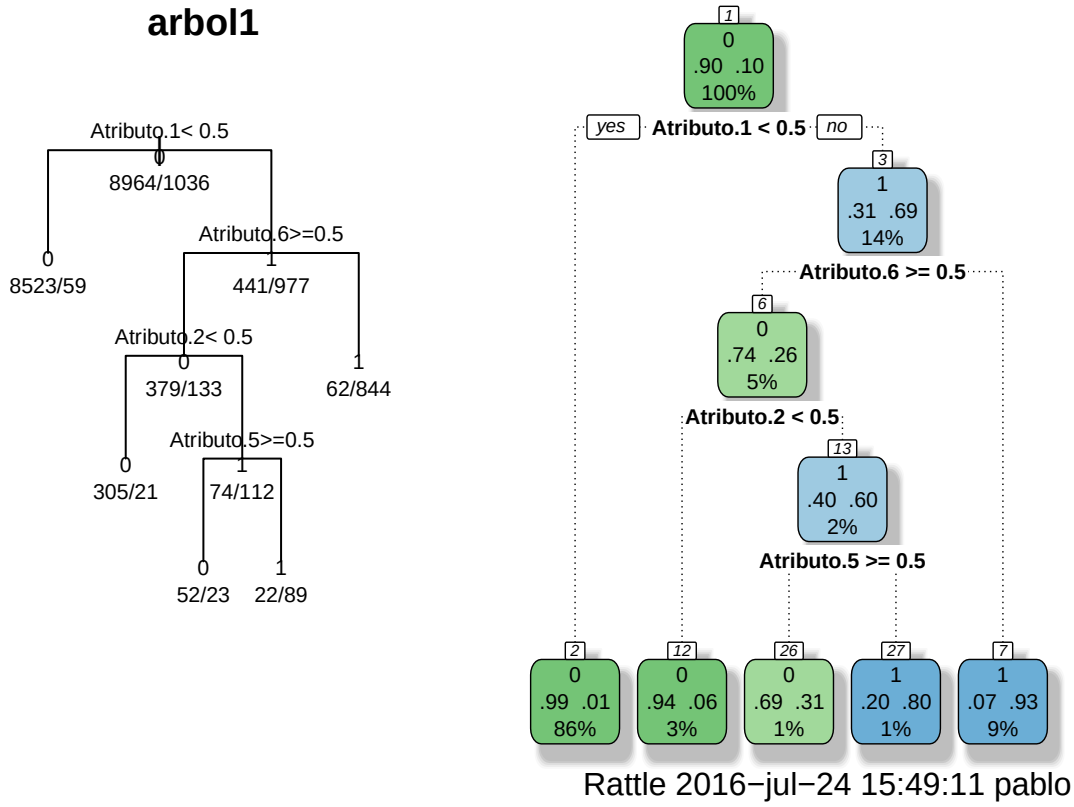
```
suppressWarnings(library(rattle))
```

```
## Rattle: A free graphical interface for data mining with R.
## Versión 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Escriba 'rattle()' para agitar, sacudir y  rotar sus datos.
```

```
suppressWarnings(library(rpart.plot))
suppressWarnings(library(RColorBrewer))
```

Cargadas los paquetes “rattle”, “rpart.plot”, “RColorBrewer”, podemos tratar mejores representaciones del árbol1. Comencemos por tratar con “fancyRpartPlot”:

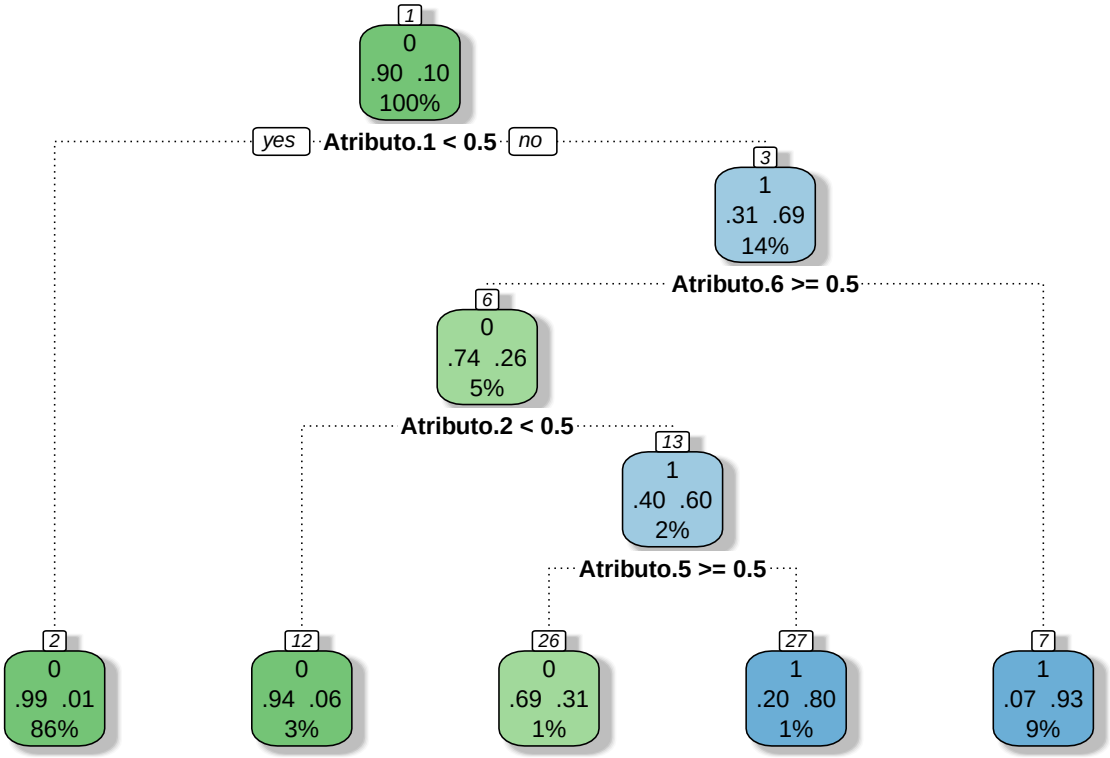
```
par(mfrow = c(1,2))
plot(fit.arbol1, uniform = TRUE, main = "arbol1", margin=0.075)
text(fit.arbol1, use.n = T, all = T, cex = .7)
fancyRpartPlot(fit.arbol1)
```




```
par(mfrow = c(1,1))
```

Ahora que veo que fancyRpartPlot es más claro, lo hago solo para mayor análisis:

```
fancyRpartPlot(fit.arbol1)
```



Podemos ver que el poder predictivo de los atributos es: **1 -> 6 -> 2 -> 5**. Por ejemplo para Atributo.1 < 0.5 se tiene valor "yes", o sea "0" o "FALSE", mientras que para **Atributo.1 > 0.5, se tiene el valor "no", o sea "1" o "TRUE"**. La lectura del árbol se sigue de igual modo con Atributo.6, Atributo.2, y Atributo.5. Atributo.3 y Atributo.4 no ayudan a predecir por lo que serían irrelevantes para el problema.