Evaluación MLI: Ejercicio 3 (Árboles de clasificación)

Inmaculada Perea Fernández

Abril 2017

Completar la construcción de un árbol de clasificación correspondiente al fichero de instrucciones "Ejemplo-LABrpart_default.r", dentro del material correspondiente a Árboles de Clasificación y Regresión

Carga de librerías necesarias

```
if (!require('rpart')) install.packages('rpart'); library('rpart')
if (!require('rpart.plot')) install.packages('rpart.plot'); library('rpart.plot')
if (!require('ROCR')) install.packages('ROCR'); library('ROCR')
if (!require('partykit')) install.packages('partykit'); library('partykit')
```

1 Obtención e inspección del conjunto de datos

1.1 Carga de los datos

El conjunto de datos *Default* consta de 673 observaciones y 4 variables:

- *default*: (No/Yes) el cliente presenta números rojos en la tarjeta de crédito
- *student*: (No/Yes)
- *balance*: saldo medio tras el pago mensual
- *income*: ingresos

```
data=read.table(file="Default.txt",header=TRUE)
dim(data)
```

```
## [1] 673   4
```

```
str(data)
```

```
## 'data.frame':    673 obs. of  4 variables:
##  $ default: Factor w/ 2 levels "No","Yes": 1 1 1 1 2 1 2 1 2 1 ...
##  $ student: Factor w/ 2 levels "No","Yes": 2 1 2 1 1 2 1 1 2 1 ...
##  $ balance: num  700 1095 256 1717 2064 ...
##  $ income : num  15905 26465 15628 51057 37373 ...
```

```
head(data)
```

```
##   default student   balance   income
## 1      No     Yes  700.3352 15905.21
## 2      No      No 1095.0727 26464.63
## 3      No     Yes  256.3257 15627.66
## 4      No      No 1716.5954 51056.87
## 5     Yes      No 2063.5719 37372.76
## 6      No     Yes  824.6166 10062.58
```

```
summary(data)
```

```
##  default   student      balance           income
##  No :340   No :429   Min.   :   0.0   Min.   : 4755
##  Yes:333   Yes:244   1st Qu.: 799.7   1st Qu.:19539
##                      Median :1357.4   Median :32630
```

```
##                     Mean   :1280.7   Mean   :32230
##                     3rd Qu.:1790.7   3rd Qu.:43096
##                     Max.   :2654.3   Max.   :66466
```

1.2 Estudio de valores perdidos

```
sum(is.na(data))
```

```
## [1] 0
```

No existen valores perdidos

1.3 División en entrenamiento y test

```
set.seed(123456789)
n=nrow(data)

index_train=sample(1:n, floor(0.7*n))

default_train=data[index_train,]
default_test=data[-index_train,]

dim(default_train)
```

```
## [1] 471   4
```

```
dim(default_test)
```

```
## [1] 202   4
```

2 Construcción del modelo

2.1 Matriz de costes

El banco prefiere evitar tarjetas "deudoras". Se va a considerar una matriz de costes. El coste de clasificar *NO* como *YES* es 5 veces superior a clasificar YES como NO

```
L=matrix(c(0,1,5,0),2,2)
rownames(L)=colnames(L)=levels(data$default)
L
```

```
##     No Yes
## No   0   5
## Yes  1   0
```

2.2 Construcción del árbol

Construir un árbol de clasificación considerando los costes definidos en la matriz L y aplicando el procedimiento de recorte *1-ES*.

```
default.rpart<- rpart(default~ .,
                      data=default_train,
                      method = "class",
                      cp=0.001,
                      parms=list(loss=L))
default.rpart
```

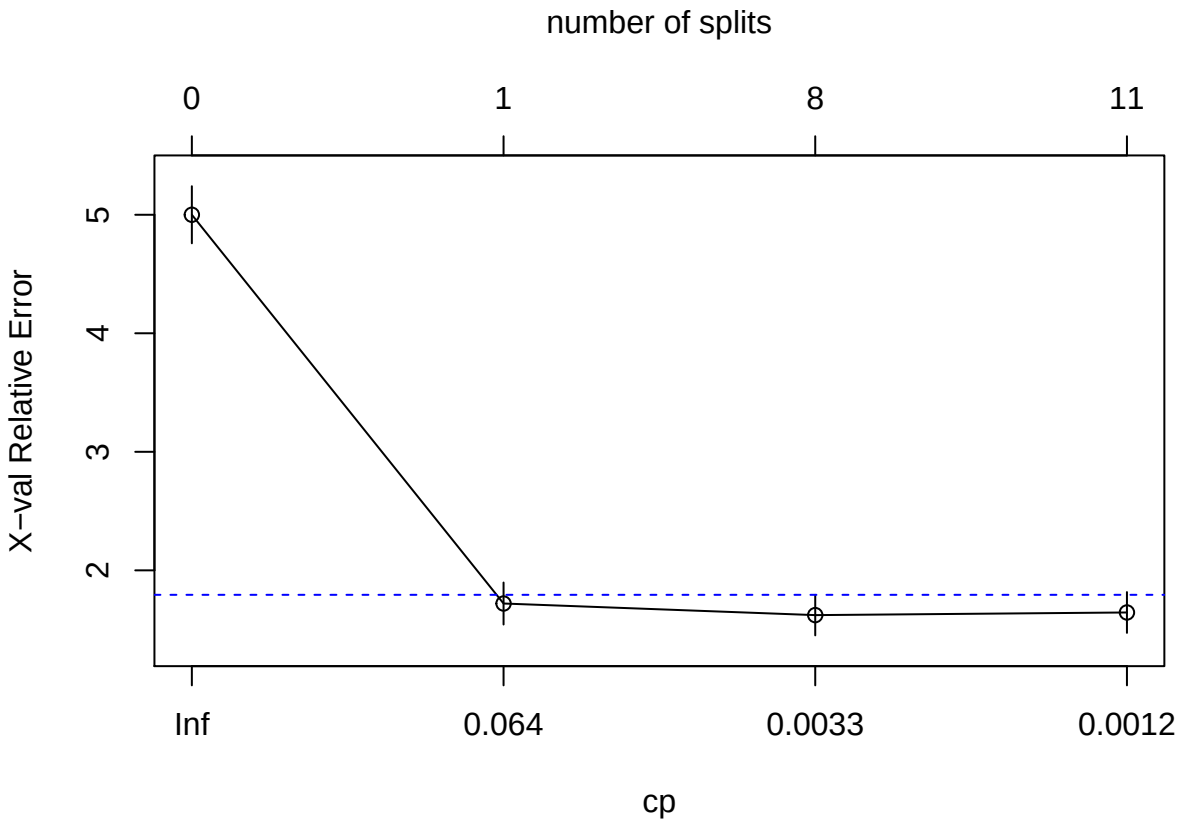
```
## n= 471
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 471 225 No (0.522292994 0.477707006)
##    2) balance< 1640.867 318  77 No (0.757861635 0.242138365)
##      4) balance< 1277.116 221  17 No (0.923076923 0.076923077) *
##      5) balance>=1277.116 97  60 No (0.381443299 0.618556701)
##       10) balance< 1452.841 52  27 No (0.480769231 0.519230769)
##         20) balance>=1415.291 10   3 No (0.700000000 0.300000000) *
##         21) balance< 1415.291 42  24 No (0.428571429 0.571428571)
##           42) income>=38952.73 16   7 No (0.562500000 0.437500000) *
##           43) income< 38952.73 26  17 No (0.346153846 0.653846154)
##             86) income< 35578.94 19  10 No (0.473684211 0.526315789) *
##             87) income>=35578.94 7   0 Yes (0.000000000 1.000000000) *
##       11) balance>=1452.841 45  33 No (0.266666667 0.733333333)
##         22) income>=19369.42 34  23 No (0.323529412 0.676470588)
##           44) income< 46044.47 26  16 No (0.384615385 0.615384615) *
##           45) income>=46044.47 8   5 Yes (0.125000000 0.875000000) *
##         23) income< 19369.42 11   5 Yes (0.090909091 0.909090909) *
##    3) balance>=1640.867 153  25 Yes (0.032679739 0.967320261)
##      6) balance< 1739.44 23  10 Yes (0.086956522 0.913043478) *
##      7) balance>=1739.44 130  15 Yes (0.023076923 0.976923077)
##       14) balance>=2133.605 28  10 Yes (0.071428571 0.928571429)
##         28) income< 19852.84 11   9 No (0.181818182 0.818181818) *
##         29) income>=19852.84 17   0 Yes (0.000000000 1.000000000) *
##       15) balance< 2133.605 102   5 Yes (0.009803922 0.990196078) *
```

Tabla con las estimaciones VC

```
cptabla<- default.rpart$cptable
cptabla
```

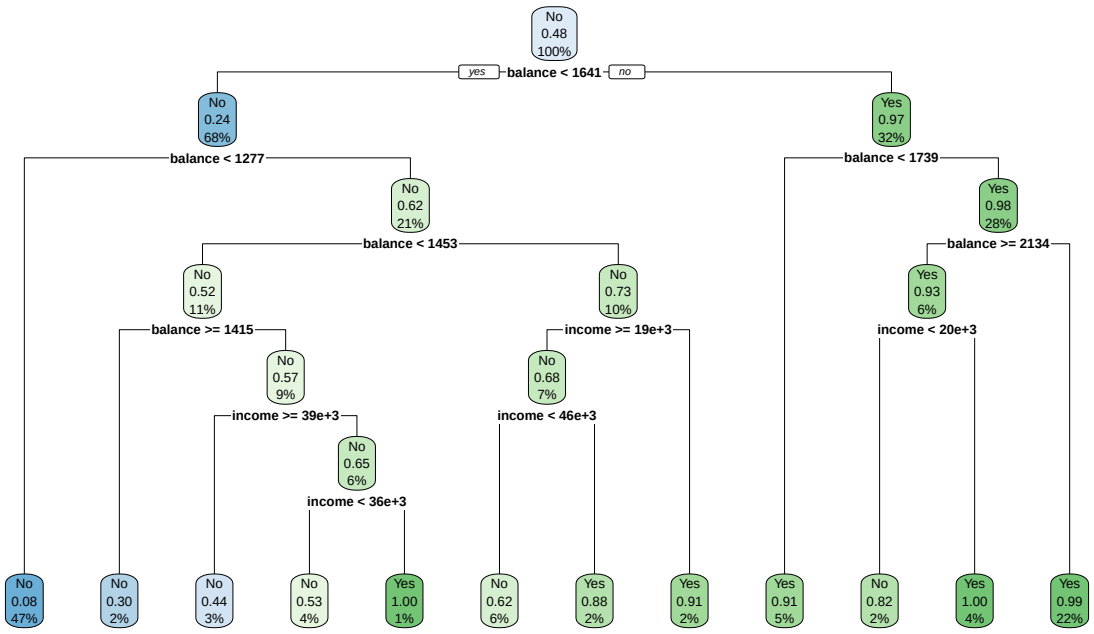
```
##            CP nsplit rel error   xerror      xstd
## 1 0.546666667      0 1.0000000 5.000000 0.2408995
## 2 0.007407407      1 0.4533333 1.720000 0.1771667
## 3 0.001481481      8 0.3911111 1.622222 0.1711050
## 4 0.001000000     11 0.3866667 1.644444 0.1720948
```

```
plotcp(default.rpart, lty=2, upper="splits", col="blue")
```



Representación gráfica

```
rpart.plot(default.rpart, main="CART sobre Default")
```



2.3 Recorte regla 1-ES

Cálculo del punto de corte con la regla 1-ES

```
CP1ES<- min(cptabla[,4])+cptabla[which.min(cptabla[,4]),5]
cat("CP 1-ES= ", round(CP1ES, 3),"\n")
```

```
## CP 1-ES=  1.793
```

```
cprecorte<- cptabla[cptabla[,4]<CP1ES,][1,1]
cat("CP Recorte= ", round(cprecorte, 3),"\n")
```

```
## CP Recorte=  0.007
```

Recorte

Aplicamos la función *prune.rpart* para hacer la poda del arbol construido en el punto de corte calculado con anterioridad

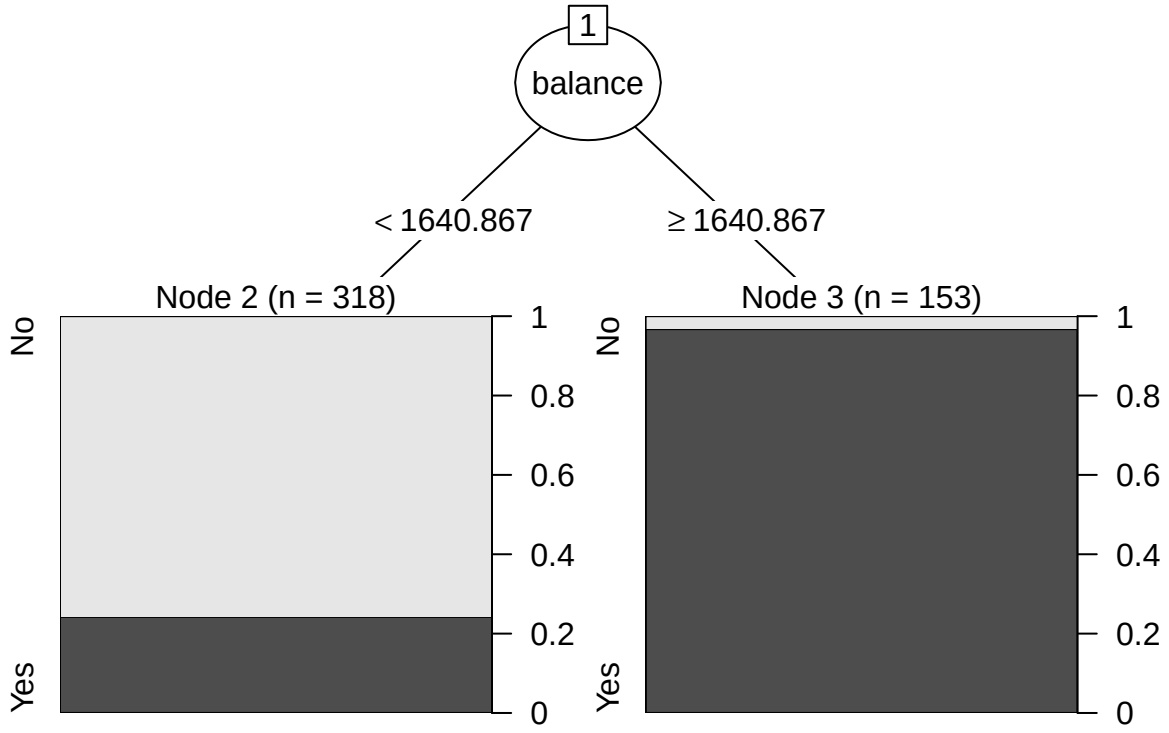
```
default.rpart.1es<-prune.rpart(default.rpart, cp=cprecorte)
default.rpart.1es
```

```
## n= 471
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 471 225 No (0.52229299 0.47770701)
```

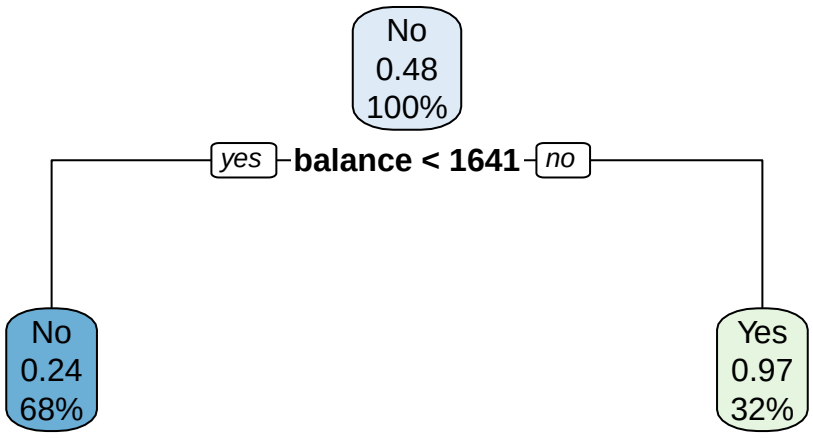
```
##   2) balance< 1640.867 318  77 No (0.75786164 0.24213836) *
##   3) balance>=1640.867 153  25 Yes (0.03267974 0.96732026) *
```

Representación gráfica

```
default.1es.asparty <- as.party(default.rpart.1es)
plot(default.1es.asparty)
```




```
rpart.plot(default.rpart.1es, main="CART sobre Default (árbol recortado)")
```



3 Medida de ajuste

3.1 Evaluar el modelo (acierto, sensitividad, especificidad)

Comparemos ambos árboles

Árbol sin recortar

```
ct<-table(default_test$default,
          predict(default.rpart, default_test, type="class"),
          dnn=c("C. REAL", "C. PRONOSTICADA"))
ct
```

```
##         C. PRONOSTICADA
## C. REAL No Yes
##     No  84  10
##     Yes 34  74
```

```
# Porcentaje correcto por grupos
100*diag(prop.table(ct, 1))
```

```
##       No      Yes
## 89.36170 68.51852
```

```
acierto=100*sum(diag(prop.table(ct)))
sens=ct[2,2]/(ct[2,2] + ct[2,1])
spec=ct[1,1]/(ct[1,1] + ct[1,2])
```

Árbol recortado

```
ct.1es<-table(default_test$default,
              predict(default.rpart.1es, default_test, type="class"),
              dnn=c("C. REAL", "C. PRONOSTICADA"))
ct.1es
```

```
##        C. PRONOSTICADA
## C. REAL No Yes
##     No  87   7
##     Yes 42  66
```

```
# Porcentaje correcto por grupos
100*diag(prop.table(ct.1es, 1))
```

```
##       No      Yes
## 92.55319 61.11111
```

```
acierto.1es=100*sum(diag(prop.table(ct.1es)))
sens.1es=ct.1es[2,2]/(ct.1es[2,2] + ct.1es[2,1])
spec.1es=ct.1es[1,1]/(ct.1es[1,1] + ct.1es[1,2])
```

A continuación construiremos una tabla comparativas para el ambos árboles (con y sin recorte)

```
arb=c(acierto, sens, spec)
arb.1es=c(acierto.1es, sens.1es, spec.1es)


tabla_resumen = data.frame (round(rbind(arb, arb.1es), 3),
                            row.names=c("Árbol sin recorte",
                                        "Árbol con recorte 1-ES"))

print(knitr::kable(tabla_resumen, format = "pandoc",
                   col.names = c("Acierto", "Sensitividad", "Especificidad"),
                   align='c'))
```

	Acierto	Sensitividad	Especificidad
Árbol sin recorte	78.218	0.685	0.894
Árbol con recorte 1-ES	75.743	0.611	0.926

Ambos árboles presentan un acierto, especificidad y sensitividad alta. No existe apenas diferencia en el acierto total y sin embargo el arbol recortado es más sencillo e interpretable. Además, en el árbol recortado se consigue disminuir los errores de clasificar *NO* como *YES*, que es una de las preocupaciones del banco que ha encargado el estudio.

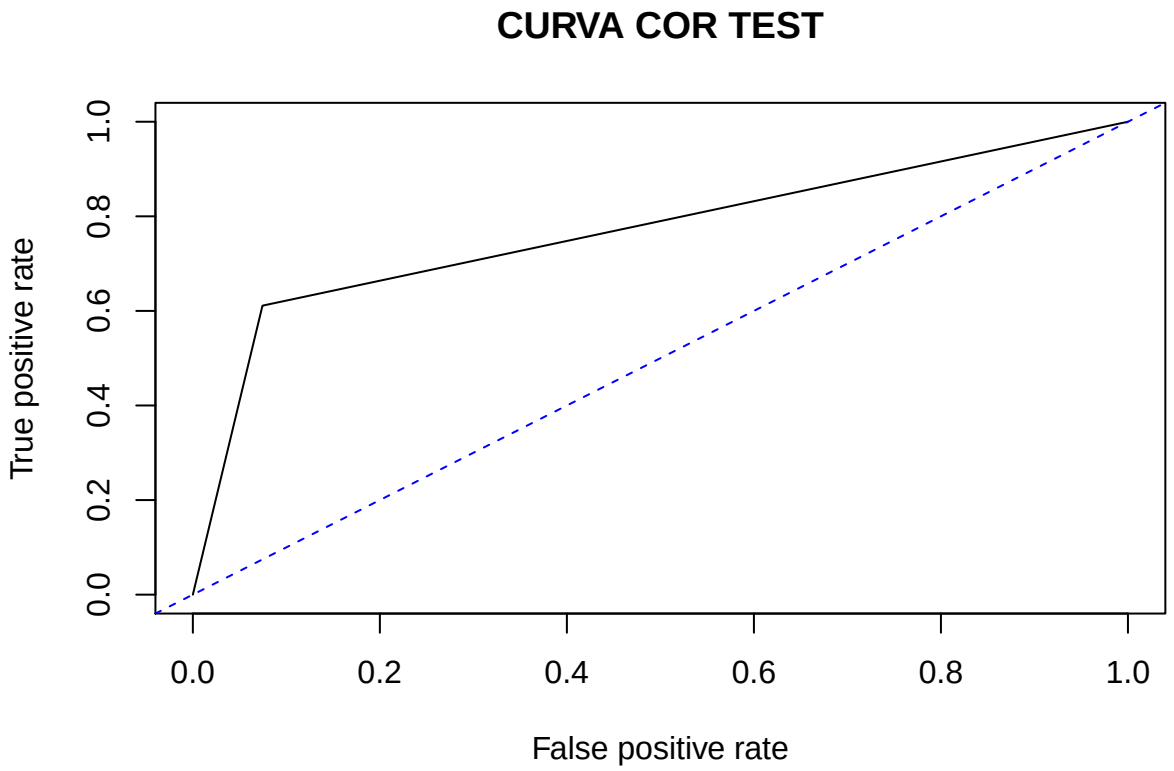
Por tanto, nos quedamos con el árbol recortado que es bastante satisfactorio para los datos.

A contincuación el resto de cálculos los realizaremos sobre el árbol recortado.

3.2 Área bajo la curva operativa característica

```
probabi<- predict(default.rpart.1es, default_test, type="prob")[,2]
prediobj<-prediction(probabi, default_test$default)
```

```r
plot(performance(prediobj, "tpr","fpr"), main="CURVA COR TEST")
abline(a=0, b=1, col="blue", lty=2)
```




```r
auc<- as.numeric(performance(prediobj,"auc")@y.values)
cat("AUC test= ",auc ,"\n")
```

```
## AUC test=  0.7683215
```

El *AUC* tambien es alto, como cabía esperar después del acierto total obtenido.

3.3 Indicador EMC (Expected Misclassification Cost)

p[NO]p[YES/NO]coste[YES/NO]+p[YES]p[NO/YES]coste[NO/YES]

```r
pNo=(ct.1es[1,1]+ct.1es[1,2])/sum(ct.1es)
PYes_No=ct.1es[1,2]/sum(ct.1es)
PYes=(ct.1es[2,1]+ct.1es[2,2])/sum(ct.1es)
PNo_Yes=ct.1es[2,1]/sum(ct.1es)
EMC=pNo * PYes_No * L[1,2] + PYes * PNo_Yes* L[2,1]
cat("EMC= ", round(EMC, 3), "\n")
```

```
## EMC=  0.192
```