ÁRBOLES DE DECISIÓN Y DE REGRESIÓN

Los árboles de decisión o regresión están formados por un conjunto de criterios anidados según una estructura en forma de árbol, en la que las ramas representan las reglas de decisión. Los elementos que intervienen en este problemas son un conjunto de variables aleatorias $(X_1, \ldots, X_p)$ que se van a utilizar para predecir el comportamiento de otra variable Y.

Hay dos tipos árboles dependiendo de la naturaleza de la variable a predecir:

1. Árboles de clasificación: se emplean cuando Y es una variable cualitativa.
2. Árboles de regresión: ídem cuantitativa.

Los árboles se construyen en etapas sucesivas, mediante la división del conjunto inicial de datos en subconjuntos excluyentes. Cada división depende de una sola variable y puede ser de dos tipos:

- Si la variable elegida es cuantitativa: $X_j \leq c_j$
- Si la variable elegida es cualitativa: $X_j \in A_j$

Teniendo en cuenta cuántas ramas se pueden formar a partir de un nodo, se distinguen dos tipos de árboles:

- Los basados en el procedimiento CART: Cada nodo sólo puede dar lugar a dos ramas. Este procedmiento puede trabajar con variables discretas y continuas simultáneamente.
- Los basados en el procedimiento CHAID: Se pueden formar más de dos ramas de un mismo nodo. Sólo emplea variables discretas; por lo tanto, cuando hay variables continuas es necesaria discretizarlas.

Elementos básicos en la contrucción de un árbol

En la construcción de un árbol hay tres elementos fundamentales, que son los siguientes:

1. El método que se utiliza para decidir qué nodo se debe dividir
2. El criterio de parada en la división de un nodo.
3. La asignación de un nodo a una clase o una predicción.

La notación que se emplea para referirse a los distintos tipos de nodos es la siguiente:

- t, nodo simple; T, el conjunto de todos los nodos de un árbol.
- t_p para un nodo padre.
- t_L y t_R para los nodos hijos.

A los nodos finales se los llama hojas.

Medidas de mezla

Cuando se divide un nodo es importante poder medir la ganancia que esto representa en la predicción.

- En los árboles de decisión esto se hace midiendo el grado de mezcla ('impureza') de categorías.
- En los árboles de regresión se mide el error de la predicciones.

Para cuantificar el grado de mezcla ('impureza') en un árbol de clasificación se debe emplear una función que cumpla los siguientes criterios:

$$\Phi : \quad (p_1, \ldots, p_k) \longrightarrow \mathbb{R}$$

1. Alcance su máximo en el punto $(\frac{1}{k}, \ldots, \frac{1}{k})$.
2. Alcance su mínimo en los puntos del tipo $(0, \ldots, 0, 1, 0, \ldots, 0)$.
3. La función sea invariante frente a permutaciones de $(p_1, \ldots, p_k)$

La impureza en un nodo t se define por $i(t) = \Phi\left(p(1 \mid t), \ldots, p(k \mid t)\right)$, donde $p(j|t)$ es la probabilidad de que un individuo del nodo t pertenezca a la clase j.

El interés que tiene la división de un nodo se va a cuantificar midiendo la reducción en la mezcla de clases:

$$\triangle i(s,t) = i(t) - p_R\, i(t_R) - p_l\, i(t_L)$$

donde p_R y p_L representan la proporción de casos del nodo padre asignados al nodo hijo derecho y nodo hijo izquierdo, respectivamente. Obviamente p_R y p_L son probabilidades condicionadas que verifican $p_R + p_l = 1$

La aportación de un nodo a la impureza total de un árbol es $I(t) = p(t)\, i(t)$. La mezcla o impureza de un árbol t se define como $I(T) = \sum_{t \in T} p(t) i(t)$, donde $p(t)$ es la proporción de individuos en el nodo t.

Entre las medidas más comunes de mezcla o impureza se pueden citar:

1. Entropía: $-\sum_{j=1}^{K} p_j \log(p_j)$, con la condición $0 \log(0) = 0$
2. Gini: $\sum_{j=1}^{K} p_j\,(1 - p_j) = 1 - \sum_{j=1}^{K} p_j^2$
3. Criterio *twoing*: $\frac{p_L\, p_R}{4} \left(\sum_{j=1}^{K} \left|p(j \mid t_L) - p(j \mid t_R)\right|\right)^2$

En general los resultados de un árbol de decisión dependen poco del criterio de impureza empleado.

Probabilidades de pertenencia a nodos y clases

El tamaño total de la muestra se representa por N y el número de individuos de cada clase por N_j, con $j = 1, \ldots, K$.

El número de individuos que pertenecen al nodo t es $N(t)$ y se cumple que $N(t) = \sum_{j=1}^{K} N_j(t)$.

Además $N_j(t_L) + N_j(t_R) = N_j(t)$.

Las probabilidades a priori de cada clase vienen dadas por $\pi = (\pi_1, \ldots, \pi_K)$. Estas probabilidades se pueden conocer previamente o se pueden estimar de la muestra mediante $\pi_j = \frac{N_j}{N}$.

La probabilidad de pertencer al nodo t y la clase j es $p(t,j) = \pi_j\, p(t \mid j) = \pi_j \frac{N_j(t)}{N_j}$.

La probabilidad de pertencer al nodo t es $p(t) = \sum j = 1^K p(t,j)$.

La probabilidad de pertencer a la clase j condicionada al nodo t es $p(j \mid t) = \frac{p(t,j)}{p(t)}$.

Para asignar un nodo hoja a una clase se calcula arg $\max_j p(j \mid t)$, es decir, en cada nodo se mira la clase que tiene una mayor frecuencia.

Probabilidades de error

Para estimar la probabilidad de error en la predicción se emplea el criterio de *resustitución*, que consiste en calcular la probabilidad de error condicionada al nodo-hoja t mediante la expresión:

$$r(t) = 1 - \max_j p(j \mid t)$$

La probabilidad de pertenecer al nodo-hoja t y cometer un error es:

$$R(t) = p(t)\, r(t)$$

La probabilidad global de error de un árbol T se estima por :

$$R(T) = \sum_{t \in T} R(t)$$

Criterios de parada en la división de un nodo

Hay muchos criterios para no seguir diviendo un nodo. Entre ellos se pueden citar:

- El número de sujetos del nodo está por debajo de cierto umbral o su pureza es del 100 % (superior a una cantidad prefijada)
- La mejora en el grado de mezcla o *impureza* no alcanza un valor mínimo. (Es un criterio poco eficiente.)
- Lo más usual es permitir un árbol muy largo y *podar* posteriormente las ramas que no aporten una mejora suficiente en la mejora de la probabilidad de error.

Criterios de poda

Los criterios de poda de un árbol suelen tener en cuenta los siguientes elementos:

- La probabilidad directa de errror en la clasificación. Este criterio puede tener problemas con la aplicación del árbol a nuevos conjuntos de datos.
- La probabilidad de error estimada mediante técnicas de validación cruzada.
- Teniendo en cuenta el coste de complejidad, que tiene en cuenta la tasa de error y la complejidad del árbol.

R

Vamos a ilustrar los anteriores conceptos con un ejemplo en R. La distribución básica de R incluye un paquete llamado `rpart` que implementa el algoritmo CART. Carguemos el paquete:

```
> library (rpart)
```

El paquete contiene varios datos de ejemplo. Para un árbol de clasificación usaremos el conjunto `kyphosis`. Contiene datos de 81 individuos (niños sometidos a cirugía de la columna) y 4 variables:

Kyphosis: atributo dicótomo que indica si el niño tenía cifosis (`present`) o no (`absent`) después de la operación.

Age: edad en meses.

Number: número de vértebras involucradas.

Start: número de la primera (la más alta) vértebra operada.

Realicemos el análisis:

```
> árbol <- rpart (Kyphosis ~ ., kyphosis)
> árbol # equivalente a print (árbol)

n= 81

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 81 17 absent (0.79012346 0.20987654)
   2) Start>=8.5 62  6 absent (0.90322581 0.09677419)
```

```
    4) Start>=14.5 29  0 absent (1.00000000 0.00000000) *
    5) Start< 14.5 33  6 absent (0.81818182 0.18181818)
     10) Age< 55 12  0 absent (1.00000000 0.00000000) *
     11) Age>=55 21  6 absent (0.71428571 0.28571429)
       22) Age>=111 14  2 absent (0.85714286 0.14285714) *
       23) Age< 111 7  3 present (0.42857143 0.57142857) *
  3) Start< 8.5 19  8 present (0.42105263 0.57894737) *
```

La información impresa por omisión incluye el tamaño muestral (n) y, por cada nodo,

node: el número de nodo; si el nodo padre es x, los nodos hijos son $2x$ y $2x + 1$.

split: la condición que cumplen los miembros del nodo; `root` es el nodo raíz, es decir, el árbol completo.

n: el número de miembros del nodo.

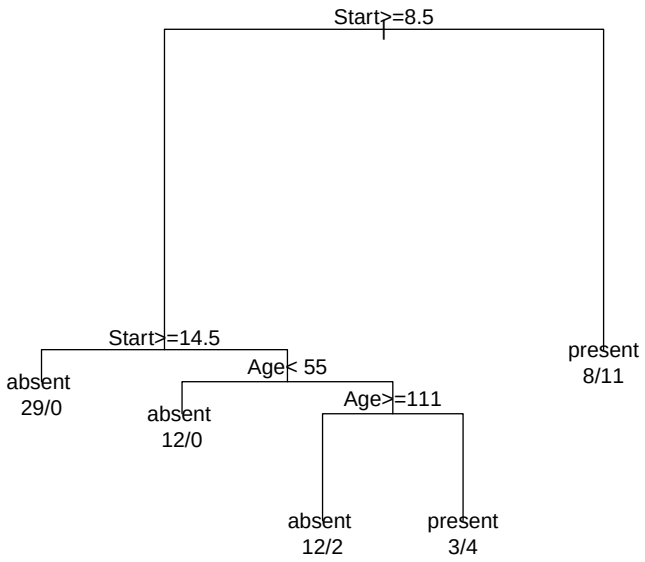
loss: el número de individuos mal clasificados en ese nodo.

yval: la predicción para los miembros del nodo, es decir, la categoría más frecuente en el nodo (en nuestro caso, `absent` o `present`).

(yprob): vector de frecuencias relativas de las categorías de la variable respuesta en el nodo; en nuestro caso, frecuencias de `absent` y `present` respectivamente.

Veamos una representación gráfica:

```
> par (xpd = NA) # para que no corte el texto al borde
> plot (árbol)
> text (árbol, use.n = TRUE)
```



En cada ramificación, la rama de la izquierda se sigue si la condición es verdadera. Se muestran la predicción para cada nodo y las frecuencias absolutas de las categorías de la variable respuesta en cada nodo.

¿Cómo decide `rpart` cuándo no subdividir las hojas? Mediante los siguientes criterios de `rpart.contol`:

minsplit: número mínimo de observaciones en un nodo para que se intente la subdivisión (por omisión, 20).

minbucket: número mínimo de observaciones en las hojas (por omisión, `round(minsplit/3)`).

cp: penalización por el incremento en complejidad, es decir, en el número de hojas (por omisión, `cp=0.01`). Sea $R(T)$ la tasa de clasificación errónea (riesgo) en el árbol T; sea $|T|$ su número de hojas; el costo de un árbol se calcula como $R_{\mathrm{cp}}(T) = R(T) + \mathrm{cp} \cdot |T| \cdot R(T_\infty)$, donde T_∞ es el árbol sin divisiones. Para cada valor de `cp` el ordenador calcula el árbol con costo mínimo.

Ejemplos de utilización:

```
> rpart (Kyphosis ~ ., kyphosis, control = rpart.control (minbucket = 2))

n= 81

node), split, n, loss, yval, (yprob)
      * denotes terminal node

  1) root 81 17 absent (0.79012346 0.20987654)
    2) Start>=8.5 62  6 absent (0.90322581 0.09677419)
      4) Start>=14.5 29  0 absent (1.00000000 0.00000000) *
      5) Start< 14.5 33  6 absent (0.81818182 0.18181818)
       10) Age< 55 12  0 absent (1.00000000 0.00000000) *
       11) Age>=55 21  6 absent (0.71428571 0.28571429)
         22) Age>=98 16  2 absent (0.87500000 0.12500000) *
         23) Age< 98 5  1 present (0.20000000 0.80000000) *
    3) Start< 8.5 19  8 present (0.42105263 0.57894737)
      6) Age< 11.5 2  0 absent (1.00000000 0.00000000) *
      7) Age>=11.5 17  6 present (0.35294118 0.64705882)
       14) Start< 5.5 12  6 absent (0.50000000 0.50000000)
         28) Age>=130.5 2  0 absent (1.00000000 0.00000000) *
         29) Age< 130.5 10  4 present (0.40000000 0.60000000)
           58) Age< 93 6  2 absent (0.66666667 0.33333333)
            116) Number< 4.5 3  0 absent (1.00000000 0.00000000) *
            117) Number>=4.5 3  1 present (0.33333333 0.66666667) *
           59) Age>=93 4  0 present (0.00000000 1.00000000) *
       15) Start>=5.5 5  0 present (0.00000000 1.00000000) *

> rpart (Kyphosis ~ ., kyphosis, control = rpart.control (cp = 0.1))

n= 81

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 81 17 absent (0.79012346 0.20987654)
  2) Start>=8.5 62  6 absent (0.90322581 0.09677419) *
  3) Start< 8.5 19  8 present (0.42105263 0.57894737) *
```

Parece que el valor 0,01 para `cp` funciona bien a menudo. Si queremos refinar más ese parámetro, podemos echar un vistazo a una tabla que indica cuántos nodos conseguimos para cada intervalo de `cp`:

```
> printcp (árbol)

Classification tree:
rpart(formula = Kyphosis ~ ., data = kyphosis)

Variables actually used in tree construction:
[1] Age   Start

Root node error: 17/81 = 0.20988

n= 81

        CP nsplit rel error xerror    xstd
1 0.176471      0   1.00000 1.0000 0.21559
2 0.019608      1   0.82353 1.1765 0.22829
3 0.010000      4   0.76471 1.1765 0.22829
```

La tabla contiene las siguientes columnas:

CP: cota inferior de CP para el intervalo correspondiente. Por ejemplo, si CP es mayor que 0,176471 entonces el árbol no se divide. Por omisión, el último CP es 0,01.

nsplit: número de divisiones; es igual a $|T| - 1$.

rel error: tasa de errores, dividida por la tasa de error del árbol sin divisiones.

xerror: tasa media de error por validación cruzada, dividida por la tasa de error del árbol sin divisiones.

xstd: desviación típica de la tasa de error por cruzvalidación, dividida por la tasa de error del árbol sin divisiones.

Por omisión, se hacen $s = 10$ cruzvalidaciones, en cada una de las cuales se dejan fuera n/s individuos al azar, se ajusta un árbol con el resto y se mide la tasa de error de clasificación sobre los individuos apartados. Para hacer $s = 100$ cruzvalidaciones, por ejemplo, para obtener resultados más estables:

```
> árbol100 <- rpart (Kyphosis ~ ., kyphosis,
+                    control = rpart.control (xval = 100))
> printcp (árbol100)
```

```
Classification tree:
rpart(formula = Kyphosis ~ ., data = kyphosis, control = rpart.control(xval = 100))

Variables actually used in tree construction:
[1] Age   Start

Root node error: 17/81 = 0.20988

n= 81

        CP nsplit rel error  xerror    xstd
1 0.176471      0   1.00000 1.00000 0.21559
2 0.019608      1   0.82353 0.82353 0.20018
3 0.010000      4   0.76471 0.94118 0.21078
```

Es razonable quedarse con el árbol que da menos error de cruzvalidación:

```
> árbol.mejor <- prune (árbol,
+                       cp = árbol$cptable [which.min (árbol$cptable[,"xerror"]),
+                                           "CP"])
```

Otra posibilidad para probar la capacidad de clasificación de un árbol es generar explícitamente un conjunto de individuos de entrenamiento, y aplicar el árbol ajustado al resto:

```
> entrena <- sample (nrow (kyphosis), 50)
> árbol1 <- rpart (Kyphosis ~ ., kyphosis, subset=entrena)
> predict (árbol1, kyphosis[-entrena,])

      absent    present
7  0.9393939 0.06060606
8  0.9393939 0.06060606
9  0.9393939 0.06060606
16 0.9393939 0.06060606
21 0.9393939 0.06060606
23 0.9393939 0.06060606
25 0.4705882 0.52941176
29 0.9393939 0.06060606
30 0.9393939 0.06060606
31 0.9393939 0.06060606
32 0.9393939 0.06060606
```

```
33 0.9393939 0.06060606
34 0.9393939 0.06060606
40 0.9393939 0.06060606
42 0.9393939 0.06060606
46 0.9393939 0.06060606
51 0.9393939 0.06060606
52 0.9393939 0.06060606
55 0.9393939 0.06060606
56 0.9393939 0.06060606
57 0.9393939 0.06060606
59 0.9393939 0.06060606
62 0.4705882 0.52941176
66 0.9393939 0.06060606
69 0.9393939 0.06060606
70 0.9393939 0.06060606
74 0.9393939 0.06060606
76 0.9393939 0.06060606
77 0.9393939 0.06060606
78 0.9393939 0.06060606
79 0.9393939 0.06060606
```

```
> predict (árbol1, kyphosis[-entrena,], type="class")
```

```
      7       8       9      16      21      23      25      29      30      31
 absent  absent  absent  absent  absent  absent present  absent  absent  absent
     32      33      34      40      42      46      51      52      55      56
 absent  absent  absent  absent  absent  absent  absent  absent  absent  absent
     57      59      62      66      69      70      74      76      77      78
 absent  absent present  absent  absent  absent  absent  absent  absent  absent
     79
 absent
Levels: absent present
```

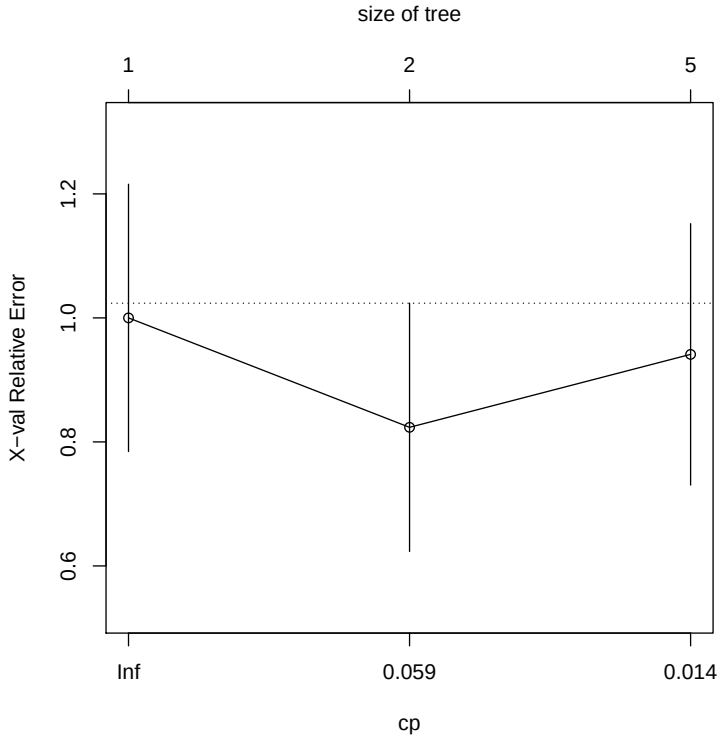
```
> ## tasa de clasificación correcta:
> mean (predict (árbol1, kyphosis[-entrena,], type="class")
+       ==
+       kyphosis$Kyphosis[-entrena])
```

```
[1] 0.8709677
```

Un criterio gráfico para escoger el árbol óptimo es escoger el árbol con menor número de hojas entre los que tienen una tasa de error por debajo

de una desviación típica por encima de la mínima tasa de error. Un gráfico ayuda:

```
> plotcp (árbol100)
```



Se cogería el árbol más a la izquierda entre aquellos por debajo de la horizontal.

Acabaremos este ejemplo explorando más en profundidad qué contiene una estructura de árbol en R:

```
> str (árbol)
```

```
List of 12
 $ frame    :'data.frame':        9 obs. of  9 variables:
  ..$ var       : Factor w/ 4 levels "<leaf>","Age",..: 4 4 1 2 1 2 1 1 1
  ..$ n         : int [1:9] 81 62 29 33 12 21 14 7 19
  ..$ wt        : num [1:9] 81 62 29 33 12 21 14 7 19
  ..$ dev       : num [1:9] 17 6 0 6 0 6 2 3 8
  ..$ yval      : num [1:9] 1 1 1 1 1 1 1 2 2
  ..$ complexity: num [1:9] 0.1765 0.0196 0.01 0.0196 0.01 ...
```

```
 ..$ ncompete  : int [1:9] 2 2 0 2 0 2 0 0 0
 ..$ nsurrogate: int [1:9] 1 2 0 2 0 0 0 0 0
 ..$ yval2     : num [1:9, 1:5] 1 1 1 1 1 1 1 2 2 64 ...
$ where    : Named int [1:81] 9 7 9 9 3 3 3 3 3 8 ...
 ..- attr(*, "names")= chr [1:81] "1" "2" "3" "4" ...
$ call     : language rpart(formula = Kyphosis ~ ., data = kyphosis)
$ terms    :Classes 'terms', 'formula' length 3 Kyphosis ~ Age + Number + Start
 .. ..- attr(*, "variables")= language list(Kyphosis, Age, Number, Start)
 .. ..- attr(*, "factors")= int [1:4, 1:3] 0 1 0 0 0 0 1 0 0 0 ...
 .. .. ..- attr(*, "dimnames")=List of 2
 .. .. .. ..$ : chr [1:4] "Kyphosis" "Age" "Number" "Start"
 .. .. .. ..$ : chr [1:3] "Age" "Number" "Start"
 .. ..- attr(*, "term.labels")= chr [1:3] "Age" "Number" "Start"
 .. ..- attr(*, "order")= int [1:3] 1 1 1
 .. ..- attr(*, "intercept")= int 1
 .. ..- attr(*, "response")= int 1
 .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
 .. ..- attr(*, "predvars")= language list(Kyphosis, Age, Number, Start)
 .. ..- attr(*, "dataClasses")= Named chr [1:4] "factor" "numeric" "numeric" "numeri
 .. .. ..- attr(*, "names")= chr [1:4] "Kyphosis" "Age" "Number" "Start"
$ cptable  : num [1:3, 1:5] 0.1765 0.0196 0.01 0 1 ...
 ..- attr(*, "dimnames")=List of 2
 .. ..$ : chr [1:3] "1" "2" "3"
 .. ..$ : chr [1:5] "CP" "nsplit" "rel error" "xerror" ...
$ splits   : num [1:17, 1:5] 81 81 81 0 62 62 62 0 0 33 ...
 ..- attr(*, "dimnames")=List of 2
 .. ..$ : chr [1:17] "Start" "Number" "Age" "Number" ...
 .. ..$ : chr [1:5] "count" "ncat" "improve" "index" ...
$ method   : chr "class"
$ parms    :List of 3
 ..$ prior: num [1:2(1d)] 0.79 0.21
 .. ..- attr(*, "dimnames")=List of 1
 .. .. ..$ : chr [1:2] "1" "2"
 ..$ loss : num [1:2, 1:2] 0 1 1 0
 ..$ split: num 1
$ control  :List of 9
 ..$ minsplit      : num 20
 ..$ minbucket     : num 7
 ..$ cp            : num 0.01
 ..$ maxcompete    : num 4
```

```
  ..$ maxsurrogate  : num 5
  ..$ usesurrogate  : num 2
  ..$ surrogatestyle: num 0
  ..$ maxdepth      : num 30
  ..$ xval          : num 10
 $ functions:List of 3
  ..$ summary:function (yval, dev, wt, ylevel, digits)
  ..$ print  :function (yval, ylevel, digits)
  ..$ text   :function (yval, dev, wt, ylevel, digits, n, use.n)
 $ y        : int [1:81] 1 1 2 1 1 1 1 1 1 2 ...
 $ ordered  : Named logi [1:3] FALSE FALSE FALSE
  ..- attr(*, "names")= chr [1:3] "Age" "Number" "Start"
 - attr(*, "ylevels")= chr [1:2] "absent" "present"
 - attr(*, "class")= chr "rpart"
```

```
> names (árbol)

 [1] "frame"     "where"     "call"      "terms"     "cptable"   "splits"
 [7] "method"    "parms"     "control"   "functions" "y"         "ordered"
```

```
> árbol$frame

      var  n wt dev yval complexity ncompete nsurrogate     yval2.1     yval2.2
1   Start 81 81  17    1 0.17647059        2          1  1.00000000 64.00000000
2   Start 62 62   6    1 0.01960784        2          2  1.00000000 56.00000000
4  <leaf> 29 29   0    1 0.01000000        0          0  1.00000000 29.00000000
5     Age 33 33   6    1 0.01960784        2          2  1.00000000 27.00000000
10 <leaf> 12 12   0    1 0.01000000        0          0  1.00000000 12.00000000
11    Age 21 21   6    1 0.01960784        2          0  1.00000000 15.00000000
22 <leaf> 14 14   2    1 0.01000000        0          0  1.00000000 12.00000000
23 <leaf>  7  7   3    2 0.01000000        0          0  2.00000000  3.00000000
3  <leaf> 19 19   8    2 0.01000000        0          0  2.00000000  8.00000000
       yval2.3     yval2.4     yval2.5
1  17.00000000  0.79012346  0.20987654
2   6.00000000  0.90322581  0.09677419
4   0.00000000  1.00000000  0.00000000
5   6.00000000  0.81818182  0.18181818
10  0.00000000  1.00000000  0.00000000
11  6.00000000  0.71428571  0.28571429
22  2.00000000  0.85714286  0.14285714
```

```
23  4.00000000  0.42857143  0.57142857
3  11.00000000  0.42105263  0.57894737
```

> árbol$splits

```
        count ncat    improve index       adj
Start      81    1 6.76232996   8.5 0.0000000
Number     81   -1 2.86679493   5.5 0.0000000
Age        81   -1 2.25021152  39.5 0.0000000
Number      0   -1 0.80246914   6.5 0.1578947
Start      62    1 1.02052786  14.5 0.0000000
Age        62   -1 0.68486352  55.0 0.0000000
Number     62   -1 0.29753321   4.5 0.0000000
Number      0   -1 0.64516129   3.5 0.2413793
Age         0   -1 0.59677419  16.0 0.1379310
Age        33   -1 1.24675325  55.0 0.0000000
Start      33    1 0.28877005  12.5 0.0000000
Number     33    1 0.17532468   3.5 0.0000000
Start       0   -1 0.75757576   9.5 0.3333333
Number      0    1 0.69696970   5.5 0.1666667
Age        21    1 1.71428571 111.0 0.0000000
Start      21    1 0.79365079  12.5 0.0000000
Number     21    1 0.07142857   3.5 0.0000000
```

> árbol$where

```
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
 9  7  9  9  3  3  3  3  3  8  8  3  9  5  3  3  3  7  3  5  3  9  8  9  9  5
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52
 9  8  3  3  3  7  7  3  7  3  5  9  5  8  9  5  9  9  3  7  3  7  9  7  8  3
53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78
 9  3  3  3  5  9  5  8  9  9  9  3  3  5  3  7  5  3  7  7  3  7  3  3  7  5
79 80 81
 7  9  5
```

`árbol$frame` es un dataframe cuyas filas representan los nodos del árbol; sus columnas:

var: variable del criterio de división, o `<leaf>` para un nodo hoja.

n: tamaño del nodo.

wt: peso; por omisión, igual al tamaño.

dev: número de individuos mal clasificados.

yval: código de la categoría mayoritaria en el nodo.

complexity: valor de CP a partir del cual aparece ese nodo.

ncompete: número de otros criterios tenidos en cuenta a la hora de dividir.

nsurrogate: número de criterios adicionales, útiles en caso de datos faltantes.

yvalX.Y: categoría predicha para ese nodo y frecuencias absolutas y relativas de cada categoría en ese nodo.

`árbol$where` da los nodos en que se clasificaría cada individuo de la muestra usada para construir el árbol.

`árbol$splits` contiene las divisiones del árbol (incluyendo *competes* y *surrogates*). `ncat` indica el sentido de la comparación. `improve` da la mejora que se consigue (véase la descripción de `summary(árbol)` en el documento Ejemplo en el campus virtual).

Regresión

En el caso de variable respuesta continua, se usan árboles de regresión (método `anova`). En este caso, en vez de el índice de Gini, el criterio de división es maximizar

$$\mathrm{SC}_T - \mathrm{SC}_L - \mathrm{SC}_R$$

donde SC representa la suma de cuadrados $\sum(y_i - \bar{y})^2$, referidas al nodo T y a sus hijos L y R.

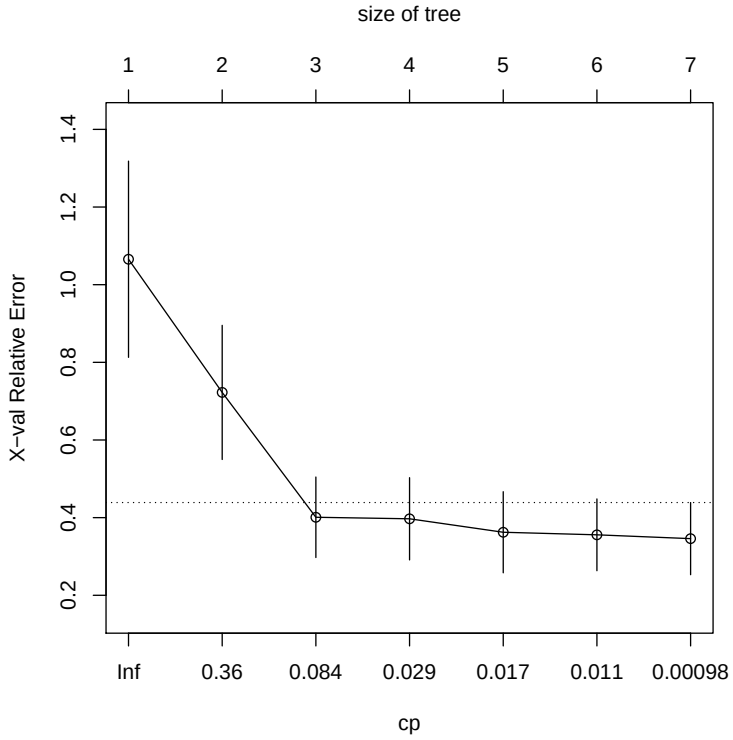
En regresión, en vez de asignar a cada nodo la categoría más frecuente, el nodo se describe mediante su media.

En regresión, el error asociado a un nodo es la varianza de y dentro del nodo. El error de predicción para una nueva observación es $y_{\text{nueva}} - \bar{y}$.

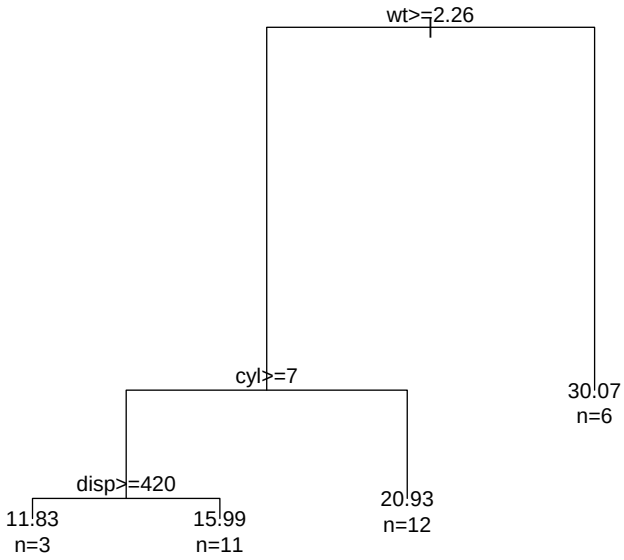
Como ejemplo, usaremos los datos `mtcars`, que contiene información sobre ciertos modelos de coches. Supongamos que tenemos interés en predecir la variable consumo (mpg).

```
> ar <- rpart (mpg ~ ., mtcars,
+              control = rpart.control (cp=0.0001,
+                                       minbucket=3,
```

```
+                                          xval=100))
> printcp (ar)

Regression tree:
rpart(formula = mpg ~ ., data = mtcars, control = rpart.control(cp = 1e-04,
    minbucket = 3, xval = 100))

Variables actually used in tree construction:
[1] cyl  disp hp   qsec wt

Root node error: 1126/32 = 35.189

n= 32

        CP nsplit rel error  xerror     xstd
1 0.652661      0  1.000000 1.06556 0.252682
2 0.194702      1  0.347339 0.72253 0.172759
3 0.036183      2  0.152636 0.40093 0.103680
4 0.023250      3  0.116453 0.39700 0.105910
5 0.011937      4  0.093203 0.36235 0.104493
6 0.009670      5  0.081267 0.35572 0.092455
7 0.000100      6  0.071597 0.34592 0.092915

> plotcp (ar)
```




```
> ar <- prune (ar, cp=0.035)
> par (xpd=NA)
> plot (ar)
> text (ar, use.n=TRUE)
```



Veamos los errores de predicción y comparémoslos con la regresión lineal habitual:

```
> summary (lm (mpg ~ ., mtcars))

Call:
lm(formula = mpg ~ ., data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-3.4506 -1.6044 -0.1196  1.2193  4.6271

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 12.30337   18.71788   0.657   0.5181
cyl         -0.11144    1.04502  -0.107   0.9161
disp         0.01334    0.01786   0.747   0.4635
hp          -0.02148    0.02177  -0.987   0.3350
drat         0.78711    1.63537   0.481   0.6353
```

```
wt          -3.71530    1.89441  -1.961   0.0633
qsec         0.82104    0.73084   1.123   0.2739
vs           0.31776    2.10451   0.151   0.8814
am           2.52023    2.05665   1.225   0.2340
gear         0.65541    1.49326   0.439   0.6652
carb        -0.19942    0.82875  -0.241   0.8122

Residual standard error: 2.65 on 21 degrees of freedom
Multiple R-squared: 0.869,        Adjusted R-squared: 0.8066
F-statistic: 13.93 on 10 and 21 DF,  p-value: 3.793e-07
```

```
> summary (predict(ar) - mtcars$mpg)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -3.833  -1.451  -0.075   0.000   1.433   4.067
```

```
> sd (predict(ar) - mtcars$mpg)
```

```
[1] 2.056709
```

```
> meanvar (ar)
```