UNIDAD 4. Arboles de Decisión: Cart; Clasificación C5.0

CART (Classification and Regression Trees)

CART (árboles de clasificación y regresión, Breiman et al. 1984) es un método que construye árboles binarios con los casos de la muestra, en los que cada nodo se divide en dos ramas o subconjuntos de la muestra. Los nodos terminales, las hojas del árbol, permiten predecir una clase de pertenencia o un número real, por lo que se puede utilizar como método de clasificación o de regresión. En el caso de la regresión, las divisiones buscan minimizar el cuadrado del residuo o error de predicción (criterio de mínimos cuadrados), y la predicción está basada en la media ponderada para cada nodo del árbol. Se trata de dividir en cada paso los datos en dos partes homogéneas, de modo que los casos dentro de cada parte sean similares entre sí, y distintos de los casos de la otra parte.

Cuando la variable explicativa es numérica, los valores pueden ser ordenados para encontrar el punto de corte óptimo que divide a los valores en dos grupos (menor o igual que el punto de corte, y mayor respectivamente), y lo mismo ocurre cuando la variable explicativa es binaria u ordinal. Sin embargo para un predictor cualitativo nominal con más de dos niveles la elección de un nivel o valor de corte no define dos grupos de forma natural, no sabemos cuales son los niveles que forman cada grupo, ya que no están ordenados. En esos casos se utilizan variables artificiales (dummy) binarias para cada nivel del factor nominal, que son empleadas de forma separada en el modelo en lugar del factor.

Cuando CART se utiliza como método de clasificación, la homogeneidad de los grupos formados en la división del árbol se traduce en que dentro de cada grupo predominen los elementos pertenecientes a una sola clase, lo que se conoce como "pureza" del nodo. Se han propuesto dos medidas del grado de pureza: el índice de Gini, y la entropía cruzada, también llamada devianza o cantidad de información. En cada paso de la construcción del árbol se mejora la pureza global minimizando el valor del estadístico de ajuste.

Para dos clases, el índice de Gini para cada grupo formado se define como:

$$G = p_1(1-p_1) + p_2(1-p_2)$$

siendo p_1 y p_2 las probabilidades de ambas clases. Como ambos valores suman la unidad $p_1=1-p_2$, y el índice se puede expresar como $G = 2p_1 p_2$ cuyo valor es cero cuando alguna de las probabilidades es cero, es decir cuando el grupo es puro (homogéneo) y solo contiene elementos de una clase. El valor máximo se obtiene cuando $p_1= p_2 = 0,5$ indicando el mayor grado de impureza. El índice de Gini para dos grupos se calcula como la media de ambos ponderada con el número de casos.

La entropía cruzada se define como

$$I = - [p_1 \log_2(p_1) + p_2 \log_2(p_2)]$$

cuyo valor se sustituye por cero cuando alguna de las probabilidades es nula. El valor de I, al igual que el índice de Gini, es más pequeño cuanto mayor es la pureza de los grupos.

El árbol de clasificación completo tiene un claro problema de sobreajuste: en general clasifica muy bien los elementos de la muestra, pero la precisión puede disminuir drásticamente cuando se predicen casos nuevos, datos no utilizados para la construcción del árbol. Para reducir ese problema se utiliza un procedimiento de pruning o "poda" del árbol que penaliza la complejidad y lo reduce, eliminando elementos superfluos o criterios irrelevantes. Cuanto mayor es el tamaño del árbol mayor es también la precisión o pureza conseguida (con los datos de la muestra), ya que ese es el criterio utilizado para su construcción, pero también es mayor el sobreajuste. Se trata por lo tanto de reducir el tamaño manteniendo una precisión aceptable, buscando un equilibrio razonable entre tamaño y precisión. En general un árbol pequeño es más sencillo, más fácil de interpretar, y tiene menos problema de sobreajuste.

Para podar el árbol basta con sumar a la función a minimizar, es decir al estadístico de ajuste que mide la pureza de los nodos, un parámetro adicional "c", parámetro de complejidad, en cada paso, de modo que si el aumento de pureza conseguido con la nueva partición no supera el valor c no se realizará la división (ya que no mejora el índice), y se prefiere por lo tanto el árbol existente, puesto que la mayor complejidad del árbol no está compensada con el pequeño beneficio conseguido.

Si c es grande el árbol resultante será más pequeño, ya que el aumento de tamaño está más penalizado. Para decidir el valor más adecuado de c se puede utilizar algún procedimiento de validación cruzada que explore distintos valores de penalización.

Breiman propone alternativamente la regla del error estándar (one-standard-error rule), con la cual se elige el árbol más pequeño dentro del intervalo de un error estándar desde el suma de cuadrados mínima.

Para aplicar CART con R se emplea la función rpart, del paquete rpart (que debemos instalar la primera vez). El argumento principal es una fórmula en la que se expresa la variable dependiente "y" –generalmente un factor de clasificación- en función de las variables predictoras "x", pudiendo indicarse opcionalmente con el argumento data el conjunto al que pertenecen las variables.

rpart(y ~ x, data)

Utilizaremos para el ejemplo el conjunto ya conocido "deudas", con una muestra de 100 clientes de un banco, algunos de los cuales han presentado impagos, y variables adicionales como edad, nivel educativo, o antigüedad en el empleo. El objetivo es predecir el impago.

```
setwd("C:/CURSO DM")
load("deudas.RData")
names(deudas)
```

```
> names(deudas)
Edad Formacion Empleo Residencia Ingreso Deud_ing Deud_tarj
Deud_otr Impago
```

Construimos los conjuntos de entrenamiento y validación:

```
set.seed(12345)
muestra <- sample(1:nrow(deudas), 80)
entrenamiento <- deudas[muestra, ]
prueba <- deudas[-muestra, ]

library(rpart)        #  carga el paquete rpart de R, que debe ser instalado previamente
modelo <- rpart(Impago ~ . , data = entrenamiento)
resultados <- predict(object = modelo, newdata = prueba, type = "class")
t <- table(resultados, prueba$Impago)
t  ;  100 * sum(diag(t)) / sum(t)
```

```
      resultados No Si
             No 14  0
             Si  0  6
[1] 100%
```

El 100% de los casos de la muestra de validación ha sido identificado correctamente. Sabiendo que no existe problema de sobreajuste podemos utilizar todos los casos de la muestra disponible para construir el clasificador:

```
# con toda la muestra
modelo <- rpart(Impago ~ . , data = deudas)
resultados <- predict(object = modelo, newdata = deudas, type = "class")
t <- table(resultados, deudas$Impago)
t  ;  100 * sum(diag(t)) / sum(t)
```

```
resultados No Si
        No 68  3
        Si  0 29
[1] 97%
```

Podemos obtener detalles del modelo elaborado ejecutando el nombre del objeto "modelo". Escribirá para cada nodo la variable utilizada y el punto de corte, el número de casos de cada clase, y la proporción o probabilidad correspondiente:

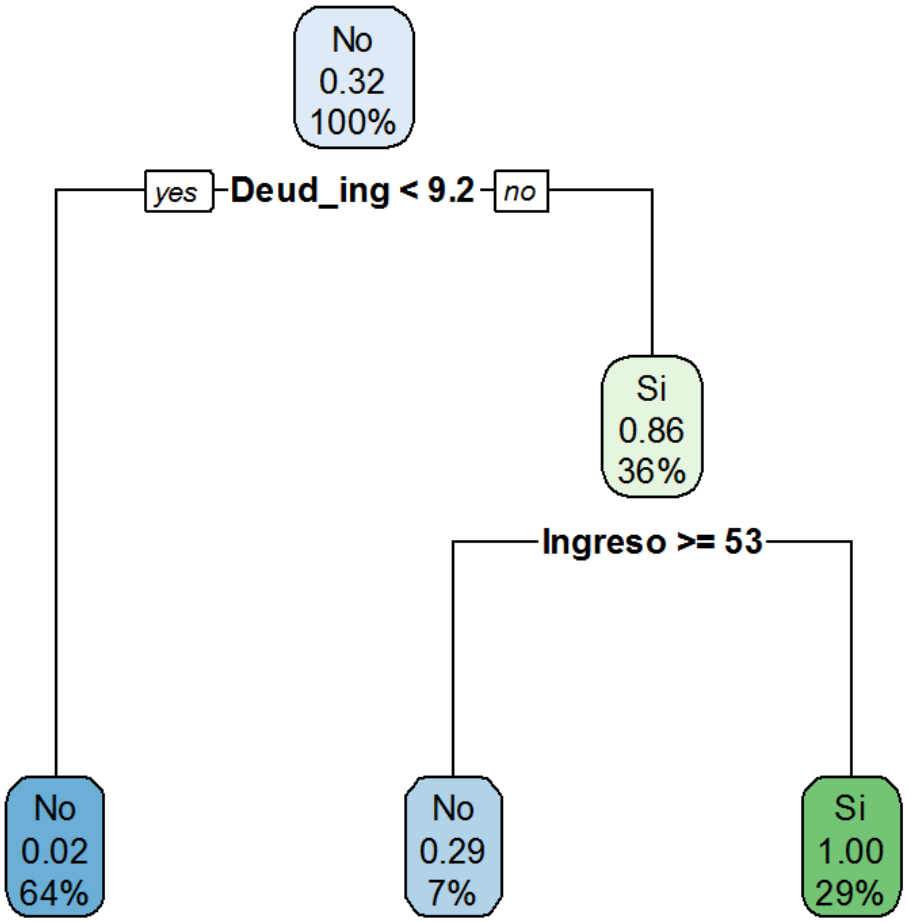
```
modelo     # escribe información del modelo construido
```

```
> modelo
n= 100

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

1) root 100 32 No (0.6800000 0.3200000)
  2) Deud_ing< 9.2 64  1 No (0.9843750 0.0156250) *
  3) Deud_ing>=9.2 36  5 Si (0.1388889 0.8611111)
    6) Ingreso>=53 7  2 No (0.7142857 0.2857143) *
    7) Ingreso< 53 29  0 Si (0.0000000 1.0000000) *
```

En el nodo 1 (raiz) hay 100 casos, de los cuales 32 (32%) son impagos. En el segundo se utiliza la variable relación entre deudas e ingresos, con el punto de corte 9,2: en ese nodo hay 64 casos de los cuales solamente uno (0,0156%) es impago.

Podemos también representar gráficamente el árbol construido, incluyendo en el gráfico la información relevante de los resultados anteriores. Para ello emplearemos la función rpart.plot, del paquete del mismo nombre, que debemos instalar y cargar:

```
library(rpart.plot)
rpart.plot(modelo)
```



Los nodos finales –hojas- del árbol muestran la clase predominante (impago NO/SI), la proporción de casos de impago = SI dentro del nodo, y el porcentaje de casos del nodo sobre el total de la muestra.

Se pueden configurar algunos elementos del algoritmo mediante la función rpart.control; los valores por defecto son adecuados para la mayoría de las aplicaciones y por lo tanto no será necesario modificarlos:

rpart.control(minsplit = 20, minbucket = round(minsplit/3), cp = 0.01, maxcompete = 4, maxsurrogate = 5, usesurrogate = 2, xval = 10, surrogatestyle = 0, maxdepth = 30)

minsplit: Número mínimo. Un nodo con menos casos no será dividido.

minbucket: número mínimo de casos en una hoja o nodo terminal. Si solamente se indica uno de estos dos primeros argumentos, se pone automáticamente el primero igual al triple del segundo.

cp: parámetro de complejidad. Cualquier partición que no mejore el estadístico de ajuste en la proporción definida por cp es ignorada.

maxcompete: numero de particiones alternativas que se conservan en el resultado. Sirve para averiguar no solo la variable optima con su punto de corte elegida en cada caso, sino la segunda, tercera, etc. que fueron rechazadas.

maxsurrogate: número de particiones suplentes conservadas, para cuando hay valores perdidos.

usesurrogate: uso de las particiones suplentes. 0 indica que no se utilizarán, 1 que se emplearán, en el orden obtenido, si la opción principal (o siguientes) tiene un valor perdido que impide utilizarla; 2 indica que si todos los valores son missing se asignará el caso a la clase mayoritaria (opción recomendada por Breiman).

xval: número de validaciones cruzadas

surrogatestyle: si es 0 se utiliza el número total de aciertos para elegir la partición suplente; si es 1 se utiliza el porcentaje, excluyendo los valores perdidos.

maxdepth: profundidad máxima del árbol o número de nodos en el camino desde el nodo raiz (0 es el nodo raíz)

C5.0

Los árboles de clasificación, de los cuales C5.0 es uno de los métodos más utilizados, permiten construir modelos para predecir o identificar la clase o tipo al que pertenece un elemento a partir de los valores de diversas variables de entrada o explicativas. La variable dependiente es siempre una variable cualitativa, con varios niveles que definen los tipos, y las variables independientes o explicativas son cuantitativas.

Se busca una clasificación formada por grupos homogéneos, formados si es posible por elementos de un único tipo. Es necesario establecer un criterio que permita valorar esa homogeneidad, y C5.0 utiliza el criterio de entropía o cantidad de información I.

$$I = - \sum_i [p_i \log_2(p_i)]$$

Para cada grupo construido se considera la frecuencia de elementos de cada uno de los k tipos o clases, p_i (probabilidad estimada de que un elemento del grupo pertenezca a la clase o tipo i). Si todos pertenecen al mismo tipo -caso de máxima homogeneidad- la entropía vale cero. En el extremo opuesto, si todas las frecuencias son iguales -máxima heterogeneidad- la entropía vale uno cuando existen solamente dos clases (en general $\log_2 k$). Es más pequeña cuanto más homogéneo sea el grupo.

El algoritmo, de forma resumida, es el siguiente:

1) Se inicia con todos los elementos agrupados en un único conjunto, que será dividido en dos partes. Para cada predictor o variable independiente se busca un punto de corte, para lo cual se ordenan sus valores de menor a mayor, tomando como punto de corte el valor intermedio entre cada dos consecutivos, y eligiendo entre todos ellos aquel que minimiza la entropía. Entre todos los predictores se elige también el que minimiza la entropía del grupo resultante y finalmente se divide el conjunto total en dos subconjuntos utilizando ese predictor y valor de corte.

2) Con cada una de las dos partes obtenidas se repite el paso 1, y así sucesivamente con cada grupo obtenido hasta que ya no se puede subdividir, bien porque todos los elementos pertenecen a la misma clase, porque no es posible reducir la entropía, o porque se cumple alguno de los criterios que pueden prefijarse como regla de parada (como por ejemplo un número máximo de iteraciones, un número mínimo de elementos en las hojas del árbol, o un mínimo en la ganancia de entropía).

C5.0 incorpora en el algoritmo el refuerzo adaptativo, generando varios clasificadores (árboles de decisión), en número predeterminado en lugar de uno solo. Para obtener cada uno de los clasificadores se ponderan de forma diferente los casos bien y mal clasificados en la anterior. Para clasificar un caso nuevo, se obtienen las predicciones de todos los clasificadores, y la clase con mayor frecuencia es la predicción final.

Los árboles de decisión y los conjuntos de reglas construidos por C5.0 generalmente no usan todos los atributos. La capacidad de elegir entre los predictores es una ventaja

importante de las técnicas de modelado basadas en árboles, especialmente cuando el número de predictores es muy elevado. El mecanismo que utiliza C5.0 para hacer esto se denomina winnowing –aventar- que hace referencia al proceso utilizado antiguamente para separar el trigo (los atributos útiles) de la paja (los superfluos). Los predictores con menor contribución a la reducción de la tasa de error de clasificación son excluidos en el proceso previo, antes de construir el árbol.

Los puntos de corte para un predictor cuantitativo dividen el espacio muestral en dos partes. Si el corte es por ejemplo 1, un valor 0,999 es diferente de 1,001 y la decisión puede ser incluirlo en una clase o en otra dependiendo de ese valor. En algunos casos este cambio brusco puede ser poco apropiado, y es más conveniente que la decisión cambie de forma más suave cerca del valor de corte. Para ello se utilizan puntos de corte "blandos" (soft thresholds), consistentes en un intervalo en lugar de un corte único. Si el valor de la variable está fuera de los límites se decide directamente, pero si está dentro del intervalo central se asigna un peso relativo que depende de la distancia al punto de corte central. Los límites del intervalo los determina C5.0 analizando la sensibilidad aparente de la clasificación a cambios pequeños en el valor de corte. Estos cortes blandos afectan únicamente al criterio de asignación, no a la interpretación del conjunto de reglas.

Para aplicar este método con R utilizaremos la función C5.0 del paquete C50, que tendremos que instalar la primera vez. En la función es necesario indicar la fórmula que expresa la variable de clasificación "y", variable dependiente necesariamente cualitativa, en función de "x", el conjunto de predictores, que tienen que ser cuantitativos:

C5.0(y ~ x, data)

El argumento opcional data permite indicar a qué conjunto pertenecen las variables de la fórmula.

Utilizaremos para el ejemplo el conjunto de datos "vinos", ya conocido, con 54 vinos monovarietales y 6 ácidos orgánicos. El objetivo es identificar la variedad de uva a partir de las 6 concentraciones.

Leemos los datos

```
setwd("C:/CURSO DM")
load("vinos.RData")     #  lee el conjunto "vinos"

names(vinos)            # listado de los nombres de variables
summary(vinos)          # estadísticos descriptivos de todas las variables
```

```
> names(vinos)

[1] "var" "gal" "tar" "mal" "shi" "cit" "suc"
```

> summary(vinos)

```
 var       gal         tar          mal         shi          cit         suc        
A:35  Min.  :0.000 Min.  :0.000 Min.  :0.000 Min.  :0.000 Min.  :0.000 Min.  :0.000
G:19  1st Q :0.455 1st Q :1.555 1st Q :1.054 1st Q :0.019 1st Q :0.101 1st Q :0.235
      Median:0.569 Median:1.988 Median:2.503 Median:0.028 Median:0.141 Median:0.311
      Mean  :0.578 Mean  :1.936 Mean  :2.193 Mean  :0.025 Mean  :0.151 Mean  :0.350
      3rd Q :0.684 3rd Q :2.394 3rd Q :3.291 3rd Q :0.033 3rd Q :0.177 3rd Q :0.466
      Max.  :1.098 Max.  :3.90  Max.  :4.891 Max.  :0.060 Max.  :0.370 Max.  :0.763
```

Construimos los conjuntos de entrenamiento (40 casos elegidos aleatoriamente) y de validación (los 14 restantes):

```
set.seed(123)
muestra <- sample(1:nrow(vinos), 40)   # 40 números de caso elegidos al azar
entrenamiento <- vinos[muestra, ]
prueba <- vinos[-muestra, ]
```

Aplicamos la función C5.0 al conjunto de entrenamiento:

```
library(C50)
modelo <- C5.0(var ~ gal+tar+mal+shi+cit+suc, data = entrenamiento)
```

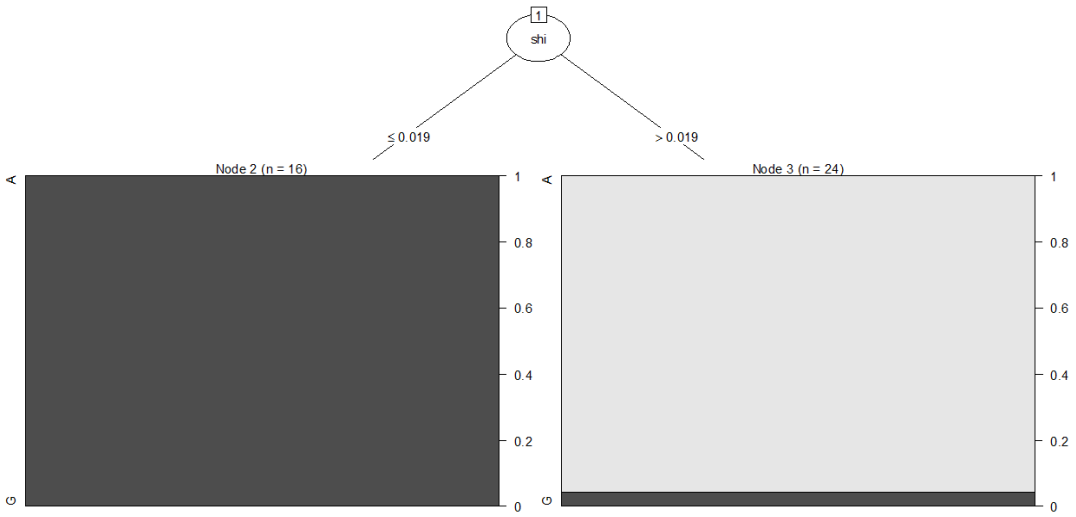
summary(modelo) # Información sobre el modelo

```
> summary(modelo) # Información sobre el modelo

Call:
C5.0.formula(formula = var ~ gal+tar+mal+shi+cit+suc, data = entrenamiento)


Decision tree:

shi <= 0.019: G (16)
shi > 0.019: A (24/1)


Evaluation on training data (40 cases):

          Decision Tree
        ----------------
        Size      Errors
           2    1( 2.5%)   <<


         (a)   (b)    <-classified as
        ----  ----
          23          (a): class A
           1    16    (b): class G

Attribute usage: 100.00%  shi
Time: 0.0 secs
```

El modelo solo ha necesitado una variable, la concentración de ácido shiquímico, con el punto de corte 0,019; por debajo de ese valor predice Godello, y por encima Albariño.

Podemos representar gráficamente la misma información con el arbol, que en este ejemplo contiene solamente una partición con dos nodos finales:

plot(modelo) # Gráfico



Veamos la matriz de confusión:

resultados.entrenamiento <- predict(modelo, newdata = entrenamiento, type = "class")
table(resultados.entrenamiento, entrenamiento$var)

```
> table(resultados.entrenamiento, entrenamiento$var)
resultados.entrenamiento  A  G
                       A 23  1
                       G  0 16
```

Solamente uno de los 40 elementos ha sido clasificado erroneamente. Veamos la predicción en el conjunto de prueba, y el grado de acierto en esa validación:

resultados.prueba <- predict(modelo, newdata = prueba, type = "class")
table(resultados.prueba, prueba$var)

```
> table(resultados.prueba, prueba$var)
resultados.prueba  A  G
                A 12  0
                G  0  2
```

Todos los vinos han sido identificados correctamente. Podemos construir el modelo con la muestra completa:

modelo <- C5.0(var ~ gal+tar+mal+shi+cit+suc, data = vinos)
resultados <- predict(modelo, vinos, type = "class")

table(resultados, vinos$var)

```
> table(resultados, vinos$var)

resultados  A  G
         A 35  1
         G  0 18
```