Árboles de regresión y clasificación

Introducción y ejemplo

(archivo arboles v2.0.R)

Recordemos el ejemplo de determinar si la persona tiene una enfermedad coronaria (Chd=1 SI, Chd=0 NO), a partir de un conjunto de variables independientes.

y	x1	x2	A	…
Chd	**Obesity**	**Tobacco**	**Famhist**	**…**
1	25.3	12	Present	…
1	28.8	0.01	Absent	…
0	29.14	0.08	Present	…
…	…	…	…	…

Los árboles tratan de hallar **puntos de corte** en las variables independientes que lleven a grupos de individuos con comportamiento homogéneo (% de unos, % de ceros) respecto a la variable respuesta y diferente entre los grupos.

En el ejemplo de árbol que aparece más abajo se han creado 3 hojas finales (se ha dividido la población en tres segmentos o grupos):

Tobacco<0.49 (1: 14% , 0:86%), contiene el 32% de todas las observaciones

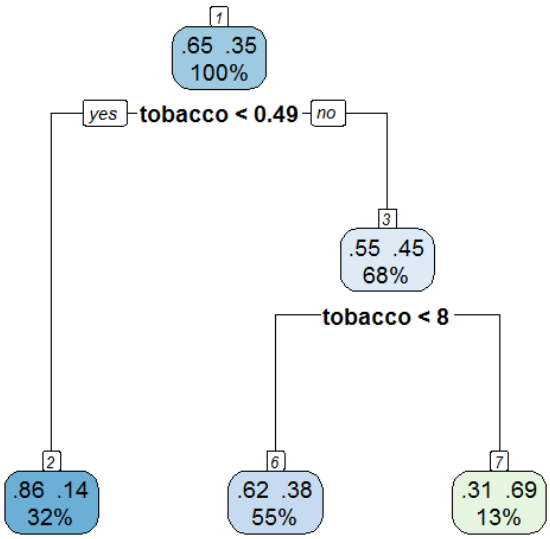
(nodo propenso a chd=0)

0.49<=Tobacco<8(1: 38% , 0:62%), contiene el 55% de todas las observaciones

(nodo normal propenso a chd=0)

Tobacco>=8 (1: 69% , 0:31%), contiene el 13% de todas las observaciones

(nodo más propenso a chd=1)



El árbol se comporta como un **análisis cluster** (creación de grupos homogéneos y diferentes entre sí) y a la vez como un **método predictivo** (a los individuos con tobacco>8 les podemos asignar chd=1 y a los demás 0).

Por ello este tipo de técnicas cuando se orientan a crear grupos se denominan **técnicas de segmentación**. Cuando se quieren crear grupos poblacionales con respecto a una variable dependiente, se utilizan estas técnicas.

Si se añaden más variables independientes el proceso es el mismo, pero haciendo grupos a partir de todas las variables independientes e implícitamente, de interacciones entre ellas.

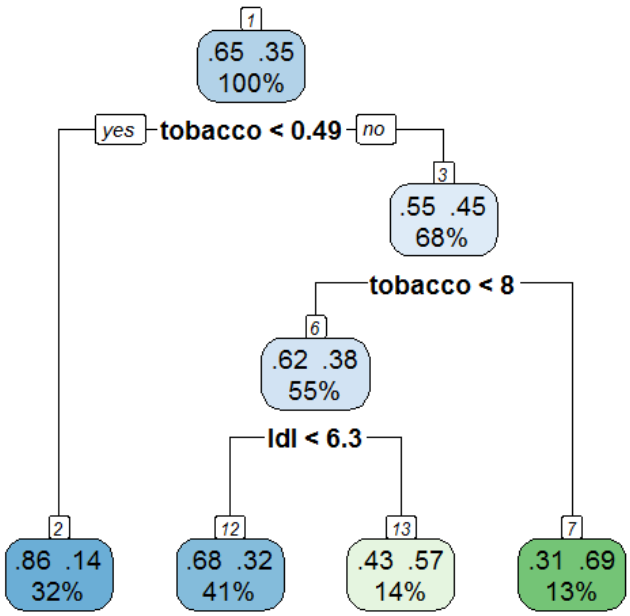
Añadiendo la variable ldl al modelo se obtienen cuatro grupos-hojas (al añadir ldl descubrimos que dentro del grupo intermedio de Tobacco se puede discriminar por ldl):

Tobacco<0.49

+0.49<=Tobacco<8 y ldl<6.3

0.49<=Tobacco<8 y ldl>=6.3

Tobacco>8



Medidas y proceso para la construcción del árbol

El proceso de creación del árbol es laborioso por la **complicada casuística** y combinatoria que se va generando cada vez que es necesario dividir un nodo.

Históricamente la construcción de árboles ha ido mejorando, creándose nuevos algoritmos cada vez:

CHAID (Chi square Automatic Interaction Detector)

CRT o CART (Classification and Regression Trees)

ID3 (Interactive Dichotomizer)

C4.5

etc.

Cada uno de estos algoritmos aporta algunos trucos y opciones para solucionar los problemas de árboles.

En este curso se utilizará el paquete de R **rpart**, que se basa en el algoritmo **CART** de Breiman.

La construcción de árboles debe de tener en cuenta los siguientes aspectos (y muchos más):

- Criterios para elegir qué variable independiente y nodo (si el árbol ya está avanzado) va a ser la base para la siguiente división
- Criterios para elegir el(los) punto(s) de corte óptimo dentro de cada variable independiente o nodo
- Criterios para elegir grupos de corte para variables independientes nominales con más de una categoría
- Número de observaciones mínima para construir un nodo
- Criterios de parada-fin del algoritmo
- Límites al número de nodos, divisiones, etc.
- Criterios de tratamiento de missings
- Si se van a utilizar datos de validación para controlar el proceso de construcción o no

Criterios para elegir el punto de corte óptimo dentro de cada variable independiente o nodo

Por simplicidad, consideramos la posibilidad de que cada nodo solo se divide en dos partes (binary split) en cada paso.

Variable dependiente cualitativa, variable independiente o nodo cualitativa u ordinal

1) Un criterio utilizado es el **Chi cuadrado (versión CHAID)**: se selecciona el corte o agrupación de categorías de la variable independiente con mayor valor del estadístico asociado cruzando la variable dependiente con la independiente. Normalmente se aplica un contraste de significatividad.

$$\chi^2 = \sum_{i=1}^{m}\sum_{j=1}^{k}\left(\frac{\left(n_{ij} - \frac{n_{i.}n_{.j}}{N}\right)^2}{\frac{n_{i.}n_{.j}}{N}}\right)$$

	chd=0	chd=1	
nodo 2	128	20	148
nodo3	174	140	314
	302	160	**462**

Nodos 2,3:

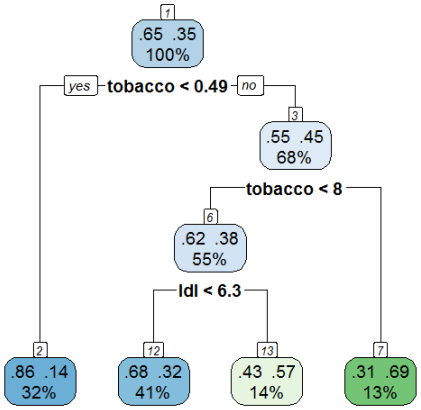
$\chi 2=(128-148*302/462)^2 +\ldots+(140-314*160/462)^2$

2) Otra posibilidad es utilizar el **índice de Gini** (cuanto más pequeño mejor, pues indica mayor distancia entre clases, llamada también "impurity"):

$$I(Gini) = \left(1 - \sum_{i=1}^{k} \left(\frac{n_i}{n}\right)^2\right)$$

Se selecciona la división de la variable independiente que mejora más el indice de Gini respecto de no utilizarla (el índice de Gini sin utilizar la variable independiente):

$$Max \quad \left(I(Gini_{nodo\ padre}) - \sum_{b} I(Gini_{rama_b})\, p(b)\right)$$



Gini nodo 1 (padre):

$1-(.65)^2-(.35)^2=0.455$

Gini nodo 2:

$1-(.86)^2-(.14)^2=0.2408$

Gini nodo 3:

$1-(.55)^2-(.45)^2=0.495$

Cálculo de la mejora :

(cuanto más bajo Gini mejor):

Gini(padre)-Gini2*.32-Gini3*.68=

=0.041344

El Gini padre es 0.04 mayor que realizando la división, por lo tanto "compensa" realizarla.

3) La tercera posibilidad para variables cualitativas es utilizar el concepto de **Entropía** o medida de información. Mide la ganancia de información en las divisiones . Cuanto menor, mejor, más información.

$$I(Entropia) = -\sum_{i=1}^{k} p_i \log_2(p_i)$$

Se selecciona la división que mejora el índice de entropía respecto de la entropía base del nodo padre:

$$Max \quad \left(I(Entropia_{nodo\ padre}) - \sum_{b} I(Entropia_{rama_b})\, p(b)\right)$$

Entropía nodo 1 (padre):

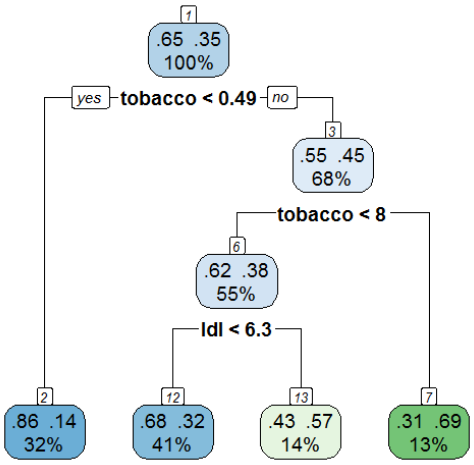
-(.65)*log(.65)- (.34)*log(.34)=.646

Entropía nodo 2:

-(.86)*log(.86)- (.14)*log(.14)=0.40

Entropía nodo 3:

-(.55)*log(.55)- (.45)*log(.45)=0.688



Cálculo de la mejora :

(cuanto más baja la entropía mejor):

Entropía(padre)- Entropía 2*.32- Entropía3*.68=

=0.04928

La entropía padre es 0.049 mayor que realizando la división, por lo tanto "compensa" realizarla.

Variable dependiente continua, variable independiente o nodo cualitativa

1) En este caso el criterio más utilizado es el estadístico **F de Snedecor** (usado en ANOVA):

$$F = \frac{\frac{\sum_b (\bar{y}_b - \bar{y})^2 n_b}{b-1}}{\frac{\sum_b \sum_{j=1}^{b} (y_{j,b} - \bar{y}_b)^2}{n-b}} \Rightarrow pvalor = prob\left(F_{b-1,n-b} > F\right)$$

Se toma la división que es más significativa (aquella que hace variar más la media de la variable dependiente en los dos grupos, que es la que tiene el valor de F más alto).

b=2,n=1030

numerador F=

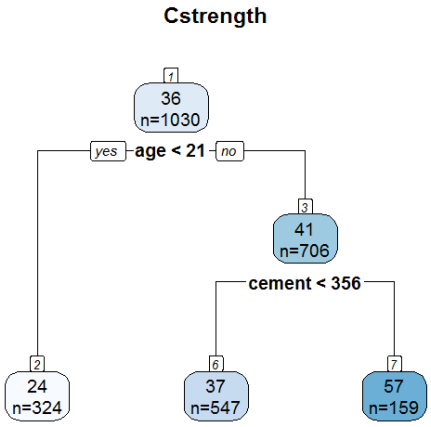
$(24-36)^2*324+(41-36)^2*706=71243$

denominador F=210

(para calcularlo se calcula la varianza de y en cada nodo

2 y 3)

F=71243/210=**339**



2) El segundo criterio utilizado es el de la **varianza**. Se calcula la variabilidad de la variable dependiente en cada grupo y se suma (literalmente la varianza sería dividiendo por n pero no se hace así).

$$Varianza(nodo) \approx \sum_{j=1}^{n_{nodo}} \left(y_j - \overline{y}_{(nodo)} \right)^2$$

A continuación se elige la división que construye grupos más homogéneos internamente y diferentes entre sí.

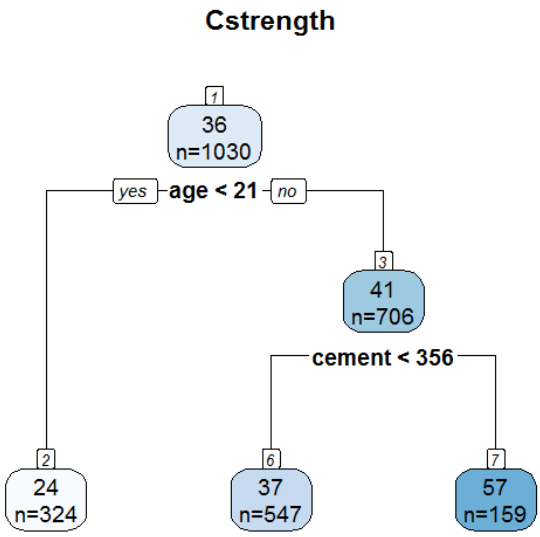
$Max(Var(nodo\ padre) - \sum_b Var(rama_b))$

SST=varianza nodo padre=287091

SSnodo2=varianza nodo 2=49896

SSnodo3=varianza nodo 3=165910

SST-SSnodo2-SSnodo3=71285



Variable dependiente continua, variable independiente o nodo continua

Los nodos continuos han de ser tratados como categóricos para realizar la división, por lo tanto el punto de corte se buscará entre un conjunto de puntos de corte candidatos seleccionados a través de métodos iterativos y se calcularán las mismas medidas ProbF o Varianza vistas anteriormente, donde la variable independiente continua, dividida en dos grupos, hace el papel de categórica.

Variable dependiente cualitativa, variable independiente o nodo continua

Se categoriza la variable continua por métodos iterativos y se utilizan los criterios Chi cuadrado, Gini o Entropía.

Criterios para seleccionar la variable independiente o nodo que va a crear la siguiente división.

Se comparan las medidas de performance calculadas para establecer las divisiones en cada variable independiente o nodo anterior y se escoge la mejor variable.

En este estado, puede haber restricciones, si se cumplen no se considera la división:

- Número de observaciones en la categoría construida por división
- Profundidad máxima del árbol (número máximo de generaciones de nodos).

Criterios para el manejo de missings

Hay varias posibilidades:

a) Asignar las observaciones con missing a la rama más grande

b) Asignar los missings a la rama con menor error calculado en las observaciones sin missings

c) Usar surrogates ("sustituciones"): imputar los missings teniendo en cuenta las observaciones con no missing en la misma rama.

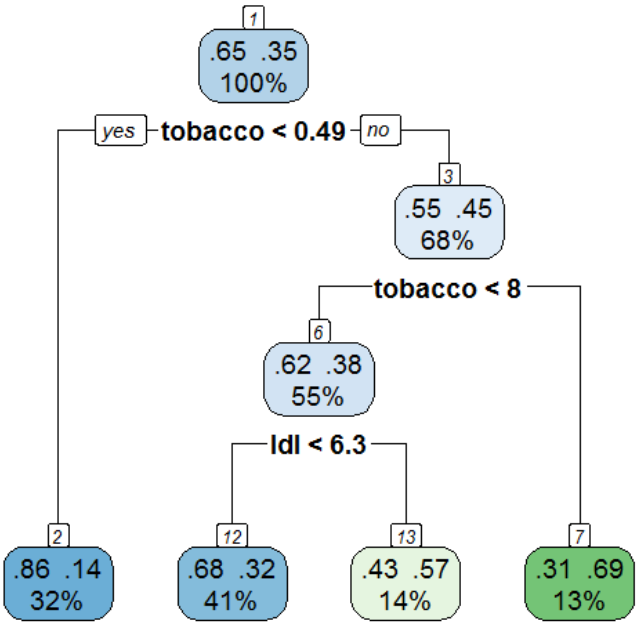
Ejemplo. Supongamos que en la primera división del gráfico (nodos 2 y 3) hay una observación missing en tobacco. Según los criterios propuestos:

a) La observación iría al nodo 3 más numeroso

b) La observación probablemente iría al nodo 2

c) Se considerarían los nodos 2 y 3 como una variable binaria A dependiente.

Se buscaría la mejor variable (diferente de Tobacco), llamada "surrogate" para predecir esa variable A y se aplicaría la predicción con un nodo simple binario con esa variable.Se predeciría a qué nodo (2 o 3) iría la observación missing, y a ese nodo se asignaría.



Árbol final, subárboles y pruning

El algoritmo finaliza cuando se cumple alguno de los criterios de parada:

- No hay suficientes observaciones en las hojas finales para considerar su división
- La profundidad máxima (parámetro prefijado) ha sido alcanzada
- En ningún nodo se puede mejorar el criterio de división, por ejemplo índice de Gini (o F para variables dependientes continuas)

El modelo final de árbol puede estar sobreajustado si los datos son complejos. Tras obtener el árbol final, puede actuarse como en los métodos cluster: escoger la solución=subárbol que parezca más estable. **Pruning** significa podar y es el acto de quedarse con un subárbol. El algoritmo original CRT o CART propone utilizar datos de validación para la selección final del subárbol.

En la implementación en el paquete rpart de R, se utiliza una constante de complejidad para orientarse en el podado de árbol, que es opcional.

Utilización del paquete rpart de R

archivo arboles v 2.0.R

Los criterios utilizados en rpart son :

- Para variables o nodos dependientes categóricas:**gini o entropía ("information")**
- Para variables dependientes continuas: el criterio de la **varianza**

Para variable dependiente binaria es mejor definirla previamente como factor en el mismo paquete o en el data frame.

En este caso se puede optar por utilizar "gini" o "information" (entropía). Por defecto es gini.

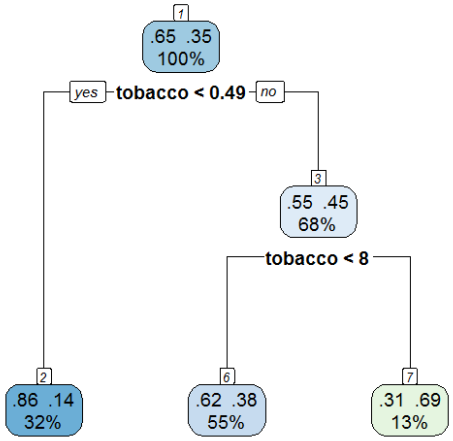
minbucket es el mínimo número de observaciones en los nodos finales. Para evitar sobreajustes, es bueno que no sea demasiado pequeño.

```
library(rpart)
library(rpart.plot)
library(rattle)

# *********************************
# EJEMPLO VARIABLE DEPENDIENTE BINARIA
# En este caso hay que nombrarla como factor, y/o pasarla antes a
#  factor, y poner method="class"
# *********************************

load("saheart.Rda")
dput(names(saheart))

saheart$chd<-ifelse(saheart$chd==1,"Yes","No")

arbol1 <- rpart(factor(chd) ~ tobacco, data = saheart,
 minbucket =30,method = "class",parms=list(split="gini"))

summary(arbol1)
```

```
Node number 1: 462 observations,    complexity param=0.075
  predicted class=No   expected loss=0.3463203  P(node) =1
    class counts:   302   160
   probabilities: 0.654 0.346
  left son=2 (148 obs) right son=3 (314 obs)
  Primary splits:
      tobacco < 0.49 to the left,  improve=19.42366, (0 missing)
Node number 2: 148 observations
  predicted class=No   expected loss=0.1351351  P(node) =0.3203463
    class counts:   128    20
   probabilities: 0.865 0.135
Node number 3: 314 observations,    complexity param=0.075
  predicted class=No   expected loss=0.4458599  P(node) =0.6796537
    class counts:   174   140
   probabilities: 0.554 0.446
  left son=6 (252 obs) right son=7 (62 obs)
  Primary splits:
      tobacco < 8.04 to the left,  improve=9.479, (0 missing)
Node number 6: 252 observations
  predicted class=No   expected loss=0.3849206  P(node) =0.5454545
    class counts:   155    97
   probabilities: 0.615 0.385
Node number 7: 62 observations
  predicted class=Yes  expected loss=0.3064516  P(node) =0.1341991
    class counts:    19    43
   probabilities: 0.306 0.694
```

El control de gráficos es importante para trabajar con árboles. Vemos algunas opciones aquí.

```
# **********************************
# GRÁFICOS
# **********************************
# extra=101:
# En cada nodo, Número de observaciones de cada clase
# y % de observaciones sobre el total
rpart.plot(arbol1,extra=101)
# extra=105:
# En cada nodo, % de observaciones de cada clase
# y % de observaciones sobre el total
rpart.plot(arbol1,extra=105)# extra=101:
# extra=5:
#  En cada nodo, solo % de observaciones de cada clase
rpart.plot(arbol1,extra=5)
# extra=5:
#  En cada nodo, solo Número de observaciones de cada clase
rpart.plot(arbol1,extra=1)
# SI QUIERO EL NÚMERO DE NODO: nn=TRUE
rpart.plot(arbol1,extra=1,nn=TRUE)
# EL MÁS COMPLETO
rpart.plot(arbol1,extra=105,nn=TRUE)

# Tamaño de letra (importante)
# The default tweak is 1, meaning no adjustment.
# Use say tweak=1.2 to make the text 20% larger.
rpart.plot(arbol1,extra=1,tweak=0.7)
# Grabar en archivo gráfico
tiff(file="arbol1.tiff")
rpart.plot(arbol1,extra=1,tweak=1.2)
dev.off()
```



Obtención de las reglas de decisión en formato literal

Se pueden obtener las reglas de decisión del árbol en un formato más accesible con la función asRules del paquete rattle

```
# ********************************
# REGLAS DE DECISIÓN DEL ÁRBOL
# Obtener las reglas en texto: función asRules del paquete rattle
# ********************************

asRules(arbol1)
```

```
 Rule number: 7 [factor(chd)=Yes cover=62 (13%) prob=0.69]
   tobacco>=0.49
   tobacco>=8.04

 Rule number: 6 [factor(chd)=No cover=252 (55%) prob=0.38]
   tobacco>=0.49
   tobacco< 8.04

 Rule number: 2 [factor(chd)=No cover=148 (32%) prob=0.14]
   tobacco< 0.49
```

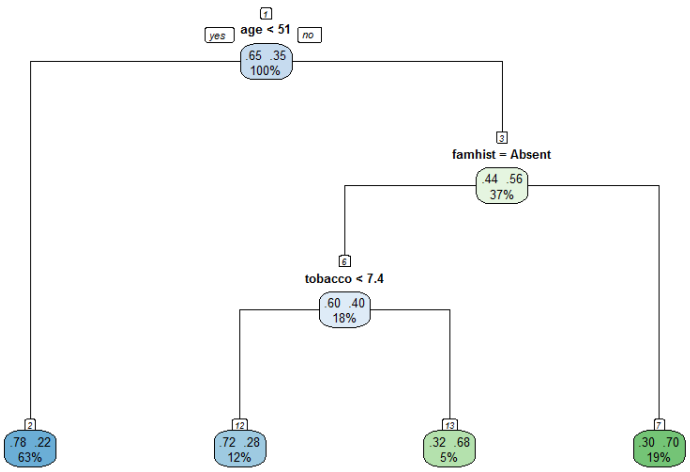
Importancia De Variables

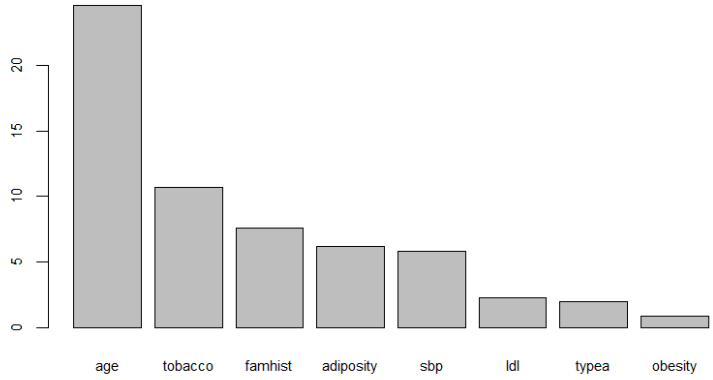
El árbol no usa todas las variables, va seleccionando las mejores (método de selección embedded) . Además, hay variables que usa más que otras y/o cuya influencia, en términos de mejora del Gini (o F) es mayor o menor. Esto permite elaborar un índice de importancia , basado en la suma de la mejora en cada división de las que ha participado. Las variables no usadas, en muchos paquetes estadísticos tendrán importancia 0. En rpart, si se consideran como surrogates ya las considera y pueden tener importancia>0. Por ello, si no hay muchos missings, es mejor poner maxsurrogate=0 para que solo nos presente la importancia de las variables que efectivamente participen en el modelo.

```
# ********************************
# IMPORTANCIA DE VARIABLES
# ********************************
arbol2 <- rpart(factor(chd) ~ ., data = saheart,minbucket =25,
method = "class")

# la función par(cex=) antes de los gráficos
# cambia el tamaño del texto

par(cex=0.7)
arbol2$variable.importance
```

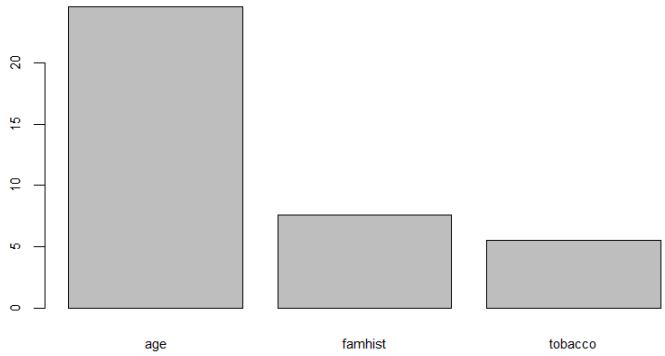
```
      age   tobacco    famhist  adiposity       sbp       ldl     typea   obesity
24.5885557 10.7144252  7.5981849  6.1471389  5.8029985  2.2555833  1.9855319  0.8702848
```

```
barplot(arbol2$variable.importance)

par(cex=1)
rpart.plot(arbol2,extra=105,tweak=0.7,type=1,nn=TRUE)
asRules(arbol2)
```






```
arbol2 <- rpart(factor(chd) ~ ., data = saheart,minbucket =25,
method = "class", maxsurrogate=0)

par(cex=1.2)
arbol2$variable.importance
barplot(arbol2$variable.importance)
```



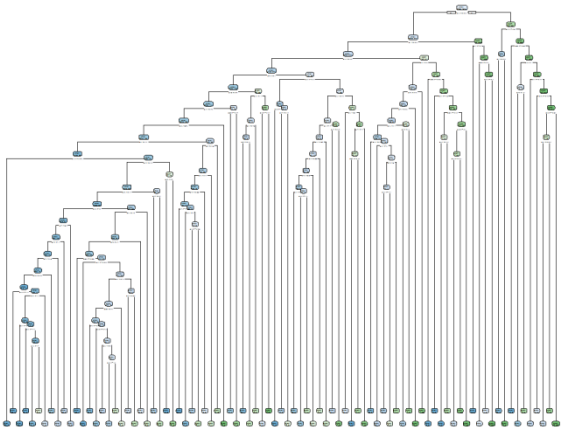
Como se ve, si no se pone nada presenta variables que no participan en el modelo (gráfico superior) pero si se pone maxsurrogate=0 solo presenta la importancia de las variables que participan en el modelo, age, famhist y typea. La más importante para predecir chd es age, que como vemos origina dos divisiones en el árbol con buena ganancia en gini o entropía.

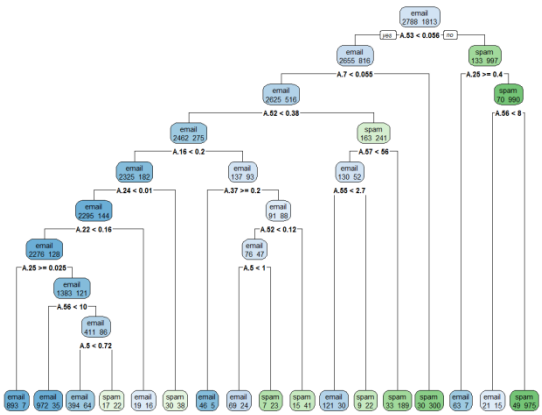
Complejidad del árbol

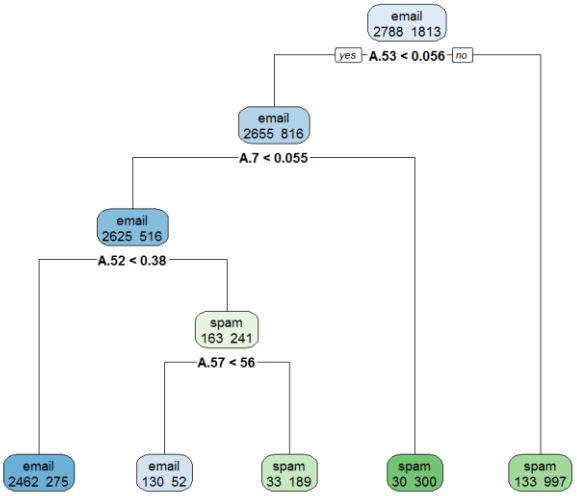
Un árbol demasiado complejo es inestable, y puede variar mucho al añadir observaciones nuevas (alta varianza). Un árbol demasiado sencillo puede tener poca potencia predictiva (alto sesgo) o bajo valor explicativo. Para controlar la complejidad del árbol usaremos un parámetro que suele existir en casi todos los paquetes que usan árboles, que es el número de observaciones mínimas en cada nodo final (minbucket). Si ese número lo ponemos bajo, permitimos árboles más complejos. Si es alto, los árboles serán más sencillos.

En el paquete rpart debemos poner el parámetro cp=0, que es un parámetro de complejidad que fijamos a cero para obviarlo pues no es extensible a otros paquetes y programas que usan árboles.

```
# **********************************
# TUNEANDO LA COMPLEJIDAD DEL ÁRBOL
# **********************************
#  Número de observaciones máximo en nodo final: minbucket

load("spam.Rda")

# Cambiando minbucket
arbol2 <- rpart(factor(spam) ~ .,data = spam,minbucket =5,cp=0)
rpart.plot(arbol2,extra=1)
arbol2 <- rpart(factor(spam) ~ .,data = spam,minbucket =30,cp=0)
rpart.plot(arbol2,extra=1)
arbol2 <- rpart(factor(spam) ~ .,data = spam,minbucket =100,cp=0)
rpart.plot(arbol2,extra=1)
```





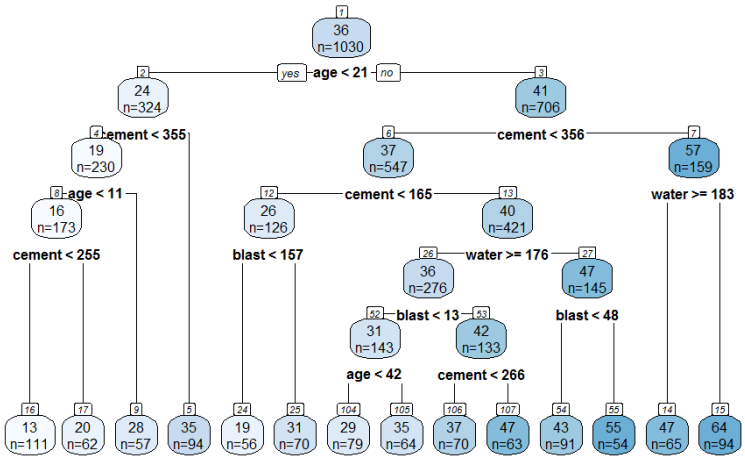


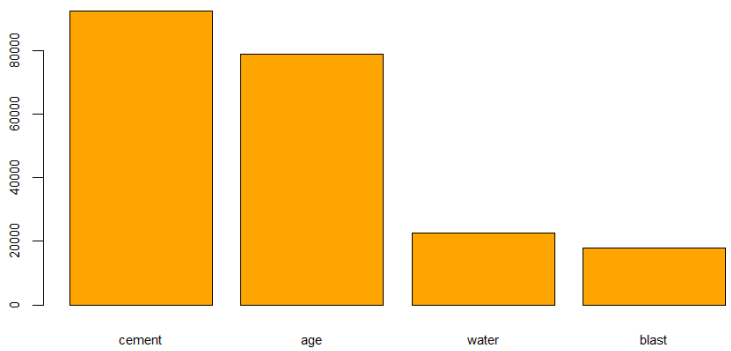
Ejemplo con variable dependiente continua

Archivo ejemplo continua arbol.R

Cuando la variable dependiente es continua buscamos el árbol que mejor discrimine la media de la variable dependiente (usamos method=”anova” en rpart).

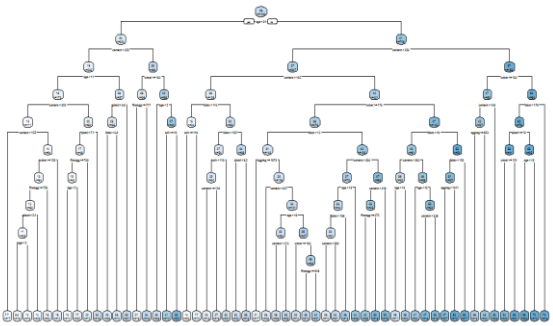
```
# *********************************
# EJEMPLO VARIABLE DEPENDIENTE CONTINUA
# PONER method="anova" en rpart
# *********************************

load("compress.Rda")

arbol3 <- rpart(cstrength~ .,
data = compress,minbucket=40,cp=0,method = "anova")

summary(arbol3)

# *********************************
# GRÁFICOS, REGLAS, IMPORTANCIA
# *********************************

rpart.plot(arbol3,extra=1,tweak=1.1,nn=TRUE)

asRules(arbol3)

arbol3$variable.importance
par(cex=0.7)
barplot(arbol3$variable.importance,col="orange")
```

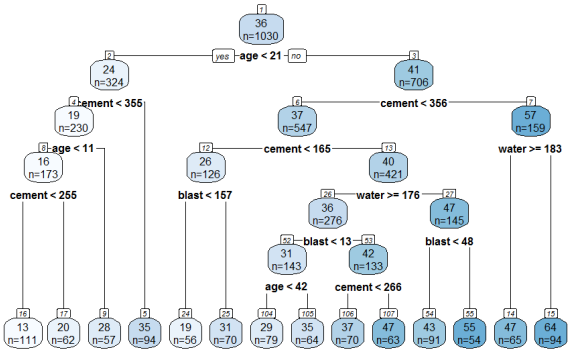


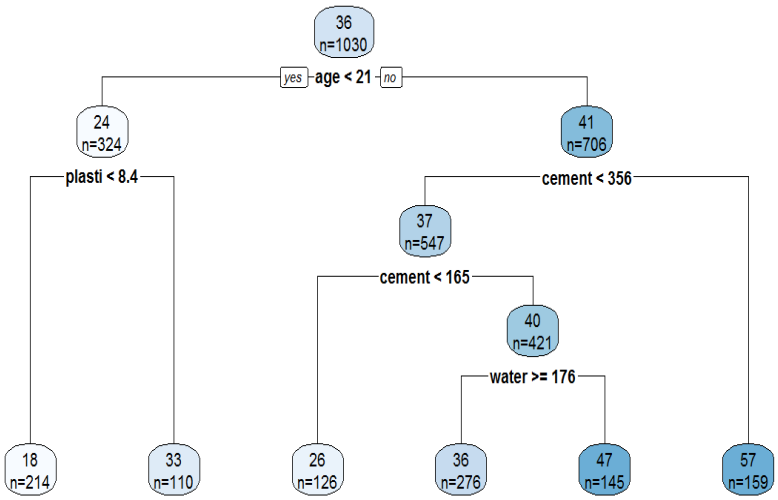


Con variable dependiente continua se controla la complejidad del árbol del mismo modo que con variable dependiente binaria. En ambos casos, se trata de un proceso artesanal que tiene en cuenta también la interpretabilidad, interés, calidad predictiva y coherencia del árbol.

```
# **********************************
# TUNEANDO LA COMPLEJIDAD DEL ÁRBOL
# **********************************

# Cambiando minbucket
arbol3 <- rpart(cstrength~ .,data = compress,minbucket =5,cp=0)
rpart.plot(arbol3,extra=1)

arbol3 <- rpart(cstrength~ .,data = compress,minbucket =30,cp=0)
rpart.plot(arbol3,extra=1)

arbol3 <- rpart(cstrength~ .,data = compress,minbucket =100,cp=0)
rpart.plot(arbol3,extra=1)
```







Tuneado y evaluación predictiva con caret. Variable binaria

Antes de probar el árbol comprobamos la logística y tuneamos la red para comparar después.

```
# ****************************************************
# TUNEADO Y EVALUACIÓN DE LA EFICACIA PREDICTIVA CON CARET
# ****************************************************
# EJEMPLO VARIABLE DEPENDIENTE BINARIA

load("saheartbis.Rda")

# Validación cruzada simple
control<-trainControl(method = "cv",number=4,
classProbs=TRUE,savePredictions = "all")

# CON LOGÍSTICA

logi<-
train(factor(chd)~age+tobacco+ldl+adiposity+typea+famhist.Absent,data=
saheartbis,
method="glm",trControl=control)
summary(logi)
logi
sal<-logi$pred
salconfu<-confusionMatrix(sal$pred,sal$obs)
salconfu
curvaroc<-roc(response=sal$obs,predictor=sal$Yes)
auc<-curvaroc$auc
auc
plot(roc(response=sal$obs,predictor=sal$Yes))
```

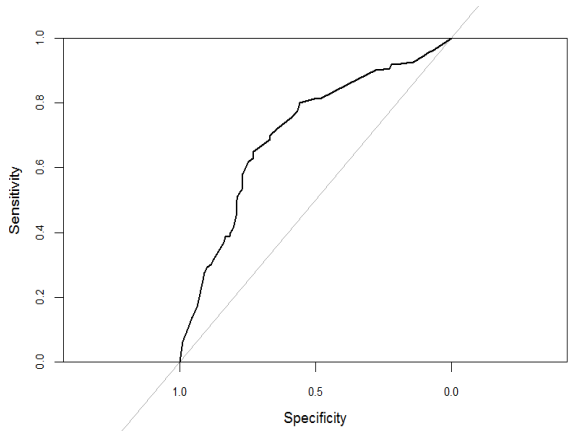
```
            Sensitivity : 0.8675
            Specificity : 0.4375
         Pos Pred Value : 0.7443
         Neg Pred Value : 0.6364
             Prevalence : 0.6537
         Detection Rate : 0.5671
   Detection Prevalence : 0.7619
      Balanced Accuracy : 0.6525
Area under the curve: 0.7474
```

```
# CON  RED
nnetgrid <-  expand.grid(size=c(5),decay=c(0.1),bag=FALSE)
red1<- train(chd~age+tobacco+ldl+adiposity+typea+famhist.Absent,
data=saheartbis,method="avNNet",linout = FALSE,maxit=100,repeats=5,
 trControl=control,tuneGrid=nnetgrid)
summary(red1)
sal<-red1$pred
salconfu<-confusionMatrix(sal$pred,sal$obs)
salconfu
curvaroc<-roc(response=sal$obs,predictor=sal$Yes)
auc<-curvaroc$auc
auc
plot(roc(response=sal$obs,predictor=sal$Yes))
```

```
            Sensitivity : 0.8775
            Specificity : 0.4375
         Pos Pred Value : 0.7465
         Neg Pred Value : 0.6542
             Prevalence : 0.6537
         Detection Rate : 0.5736
   Detection Prevalence : 0.7684
      Balanced Accuracy : 0.6575


Area under the curve: 0.746
```

```
# CON ARBOL: con rpart se puede tunear el cp, pero no lo hacemos por
´no tener extensión a otros paquetes y modelos de árboles
# En su lugar hacemos pruebas con diferentes valores de minbucket

# UNA SOLA PRUEBA CON MINBUCKET=30

arbolgrid <-  expand.grid(cp=c(0))

arbolcaret<-
train(factor(chd)~age+tobacco+ldl+adiposity+typea+famhist.Absent,
data=saheartbis,
method="rpart",minbucket=30,trControl=control,tuneGrid=arbolgrid)

arbolcaret

sal<-arbolcaret$pred

salconfu<-confusionMatrix(sal$pred,sal$obs)
salconfu

curvaroc<-roc(response=sal$obs,predictor=sal$Yes)
auc<-curvaroc$auc
auc
plot(roc(response=sal$obs,predictor=sal$Yes))
```

```
            Sensitivity : 0.8318
            Specificity : 0.4163
         Pos Pred Value : 0.7290
         Neg Pred Value : 0.5673
             Prevalence : 0.6537
         Detection Rate : 0.5437
   Detection Prevalence : 0.7459
      Balanced Accuracy : 0.6240

       'Positive' Class : No

Area under the curve: 0.6916
```



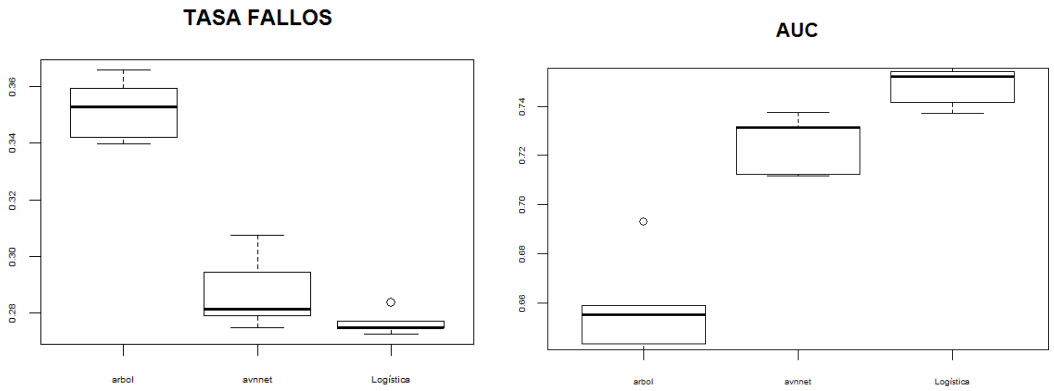
```
# TUNEADO VARIANDO EL VALOR DE MINBUCKET EN UN BUCLE (ya que en caret
no se puede tunear minbucket en el grid)

for (minbu in seq(from=5, to=60, by=5))
{
print(minbu)
cat("/n")

arbolgrid <-  expand.grid(cp=c(0))

arbolcaret<-
train(factor(chd)~age+tobacco+ldl+adiposity+typea+famhist.Absent,
data=saheartbis,
method="rpart",minbucket=30,trControl=control,tuneGrid=arbolgrid)

# arbolcaret

sal<-arbolcaret$pred

salconfu<-confusionMatrix(sal$pred,sal$obs)
print(salconfu)

curvaroc<-roc(response=sal$obs,predictor=sal$Yes)
auc<-curvaroc$auc
print(auc)
# plot(roc(response=sal$obs,predictor=sal$Yes))
}
```

En las sucesivas salidas parece que minbucket=10 tiene mejor Accuracy(0.69) y AUC (0.71)

Ejemplo validación cruzada repetida para variable dependiente binaria

```
load ("saheartbis.Rda")
source ("cruzadas avnnet y log binaria.R")
source ("cruzada arbolbin.R")

medias1<-cruzadalogistica(data=saheartbis,
 vardep="chd",listconti=c("age", "tobacco","ldl",  "adiposity",
"typea", "famhist.Absent"), listclass=c(""),
grupos=4,sinicio=1234,repe=5)

 medias1$modelo="Logística"

medias2<-cruzadaavnnetbin(data=saheartbis,
 vardep="chd",listconti=c("age", "tobacco","ldl",  "adiposity",
"typea", "famhist.Absent"),
 listclass=c(""),grupos=4,sinicio=1234,repe=5,
  size=c(5),decay=c(0.1),repeticiones=5,itera=200)
  medias2$modelo="avnnet"
```

```
  medias3<-cruzadaarbolbin(data=saheartbis,
 vardep="chd",listconti=c("age", "tobacco","ldl",  "adiposity",
"typea", "famhist.Absent"),
 listclass=c(""),grupos=4,sinicio=1234,repe=5,
  cp=c(0),minbucket =10)
  medias3$modelo="arbol"
   union1<-rbind(medias1,medias2,medias3)
par(cex.axis=0.5)
boxplot(data=union1,tasa~modelo,main="TASA FALLOS")
boxplot(data=union1,auc~modelo,main="AUC")
```



Se observa que no se consigue nada con el árbol, desde el punto de vista predictivo es mejor la logística en este caso.

Tuneado y validación cruzada repetida con variable continua

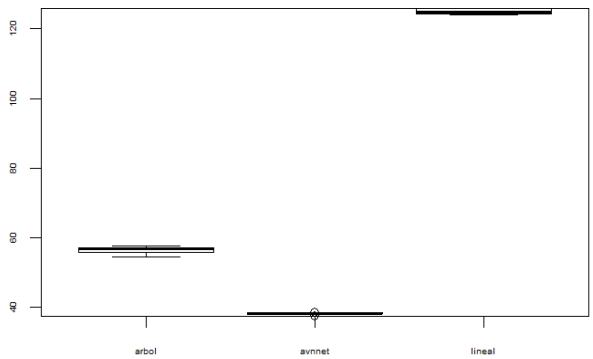
```
load("compressbien.Rda")

control<-trainControl(method = "cv",number=4,savePredictions = "all")

# Regresión
reg1<- train(cstrength~cement+plasti+age+blast+water+ash,
 data=compressbien, method="lm",trControl=control)

reg1

RMSE      Rsquared  MAE
10.48584  0.60631   8.321747

# Red

nnetgrid <-  expand.grid(size=c(15),decay=c(0.01),bag=F)

rednnet<- train(cstrength~cement+plasti+age+blast+water+ash,
 data=compressbien, method="avNNet",linout = TRUE,maxit=100,
 trControl=control,repeats=5,tuneGrid=nnetgrid)

rednnet

RMSE      Rsquared   MAE
5.428117  0.8940643  4.054742
```

```
# Árbol con rpart: parece que minbucket=45 está bien

arbolgrid <-  expand.grid(cp=c(0))

for (minbu in seq(from=5, to=60, by=5))
{

arbolcaret<- train(cstrength~cement+plasti+age+blast+water+ash,
 data=compressbien,method="rpart",minbucket=minbu,
 trControl=control,tuneGrid=arbolgrid)
print(minbu)
print(arbolcaret)
}
```

```
  RMSE      Rsquared   MAE
  7.251849  0.8137638  5.380279
```

Probamos ahora los gráficos de cajas elaborados a travñes de validación cruzada repetida con esos modelos:

```
source("cruzada arbol continua.R")
source("cruzadas avnnet y lin.R")

data<-compressbien
medias1<-cruzadaavnnet(data=data,
vardep="cstrength",listconti=c("age","water","cement","blast"),
listclass=c(""),grupos=4,sinicio=1234,repe=5,
size=c(15),decay=c(0.01),repeticiones=5,itera=100)
medias1$modelo="avnnet"
medias2<-cruzadalin(data=data,
vardep="cstrength",listconti=c("age","water","cement","blast"),
listclass=c(""),grupos=4,sinicio=1234,repe=5)
medias2$modelo="lineal"
medias3<-cruzadaarbol(data=data,
vardep="cstrength",listconti=c("age","water","cement","blast"),
listclass=c(""),grupos=4,sinicio=1234,repe=5,cp=0,minbucket=45)
medias3$modelo="arbol"

union1<-rbind(medias1,medias2,medias3)

par(cex.axis=0.5)
boxplot(data=union1,error~modelo)
```



Al ser la relación no lineal el árbol se adapta mejor que la regresión, pero en este caso peor que la red neuronal.

Grandes diferencias de los árboles frente a otros métodos predictivos

- Las transformaciones monótonas (crecientes o decrecientes) sobre las variables continuas no tienen efecto, el árbol solo tiene en cuenta **el orden** entre observaciones.

- A veces no es necesario crear dummies con las variables categóricas, aunque puede ser conveniente porque muchos paquetes de árboles en R no admiten factores con demasiados niveles. El orden interno (alfanumérico por ejemplo) en las variables categóricas puede tener influencia según que programa-paquete se use.

- Los missing están incorporados al proceso y normalmente no necesitan un tratamiento exhaustivo previo (solo lo básico, borrar observaciones y variables con demasiados missings si el hecho de ser missing no es informativo).

- La selección de variables está incorporada al proceso (método embedded). Aunque siempre es beneficioso realizar un estudio previo y preselección.

- Es el mejor método para descubrir interacciones entre variables categóricas (de hecho CHAID, históricamente el primer método de árboles, significa Chi-Square Automatic Interaction Detector).

Ventajas de los árboles

- Gran potencia descriptiva, se comprende muy bien el resultado. Resultados a menudo simples.

- Sobre todo en clasificación pero también en regresión, se descubren interacciones y reglas muy difíciles de encontrar con otros métodos. Estas reglas se pueden utilizar como variables dummy para utilizar en otros métodos predictivos. En la práctica, si realizamos un estudio sencillo y descriptivo de búsqueda de relaciones con regresión logística, es conveniente siempre complementarlo con la utilización de árboles, pues descubren relaciones opacas para la regresión logística.

- Las relaciones no lineales no afectan tanto al comportamiento de los árboles como a otros métodos.

- No hay asunciones teóricas sobre los datos

- Aportan medidas de importancia de las variables

- Manera propia y eficiente de tratar los missings, incorporada al proceso

- Incorporación automática de interacciones, detectan relaciones por regiones que ningún otro método puede encontrar

Desventajas de los árboles

- Poca fiabilidad y mala generalización: cada hoja es un parámetro y esto provoca modelos sobreajustados e inestables para la predicción. Añadir una variable nueva o un nuevo conjunto de observaciones puede alterar mucho el árbol.

- Complejidad en la construcción del árbol y casuística: dos plataformas (programas) diferentes dan dos árboles diferentes

- Poca eficacia predictiva: toscos en los valores de predicción.

TABLE 10.1. *Some characteristics of different learning methods. Key:* 🟢*= good,* 🟡*=fair, and* 🔴*=poor.*

Characteristic	Neural nets	SVM	Trees	MARS	k-NN, kernels
Natural handling of data of "mixed" type	🔴	🔴	🟢	🟢	🔴
Handling of missing values	🔴	🔴	🟢	🟢	🟢
Robustness to outliers in input space	🔴	🔴	🟢	🔴	🟢
Insensitive to monotone transformations of inputs	🔴	🔴	🟢	🔴	🔴
Computational scalability (large N)	🔴	🔴	🟢	🟢	🔴
Ability to deal with irrelevant inputs	🔴	🔴	🟢	🟢	🔴
Ability to extract linear combinations of features	🟢	🟢	🔴	🔴	🟡
Interpretability	🔴	🔴	🟡	🟢	🔴
Predictive power	🟢	🟢	🔴	🟡	🟢

Tabla extraída del libro Elements of Statistical Learning, Hastie, Tibshirani

Se observa en la tabla orientativa cómo los árboles tienen muchas virtudes pero poco poder predictivo. Afortunadamente, se ha conseguido mantener las ventajas de los árboles mejorando el poder predictivo mediante la combinación de muchos árboles, lo que será la base de las técnicas Bagging, Random Forest y Gradient Boosting.

Bibliografía básica

Libros disponibles en PDF

Hastie, Tibshirani: The Elements of Statistical Learning (PDF)

(En la web hay más información)

http://statweb.stanford.edu/~tibs/ElemStatLearn/

Hastie, Tibshirani: An Introduction to Statistical Learning with Applications in R (PDF)

(básicamente el mismo que el anterior, pero para R)

http://www-bcf.usc.edu/~gareth/ISL/data.html/

Paquetes R

https://cran.r-project.org/web/packages/rpart/index.html