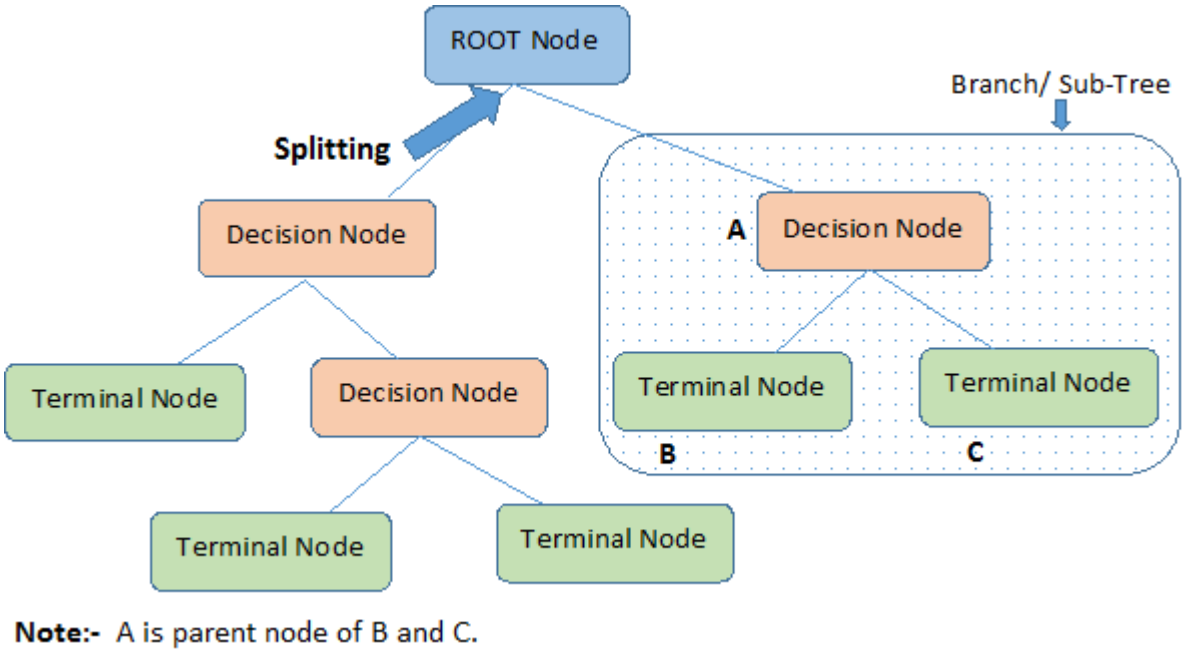
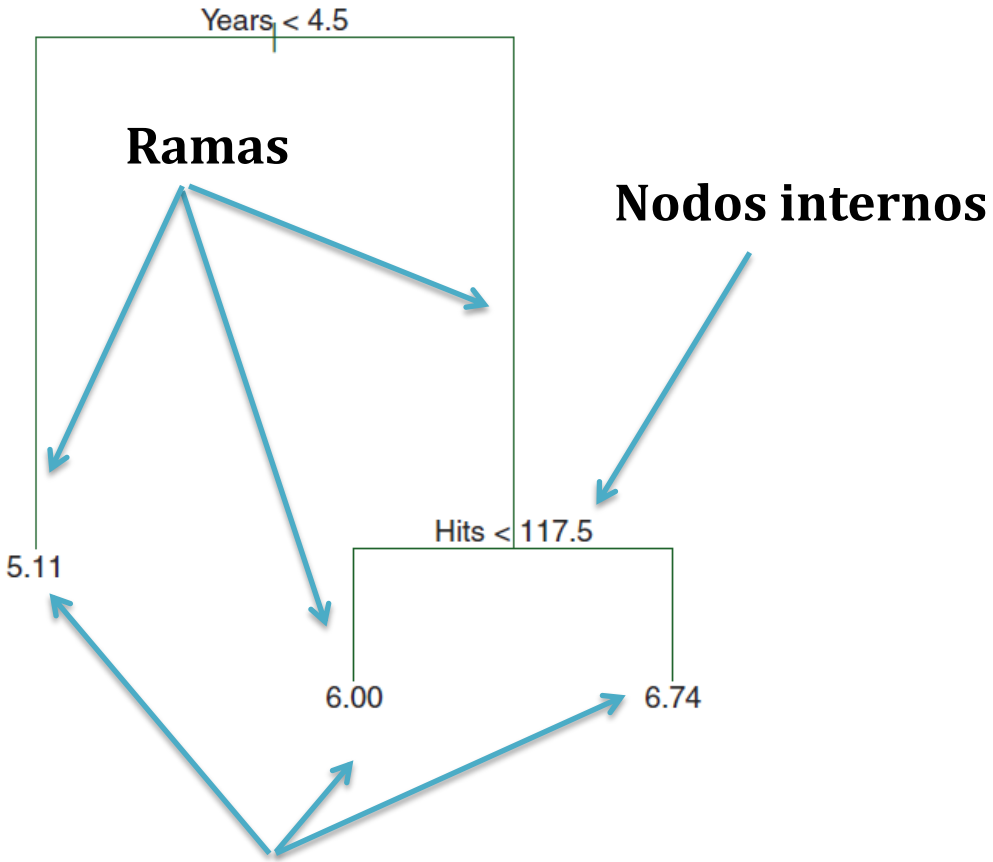
# Árboles de Decisión

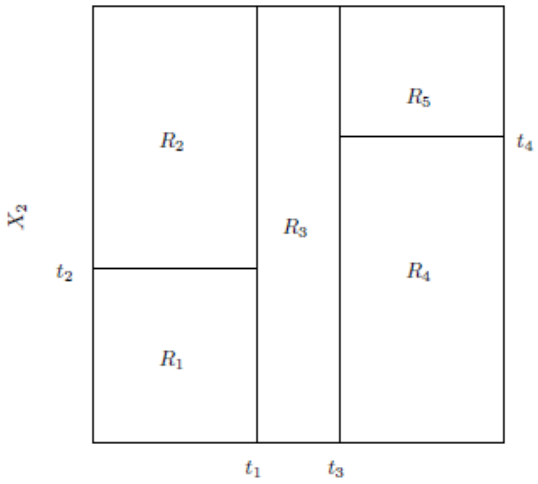
Los modelos de clasificación o regresión de Árboles de Decisión se basan en la estratificación del dominio de las co-variantes predictoras $X$ en un número de regiones mediante el método de división binaria recursiva.

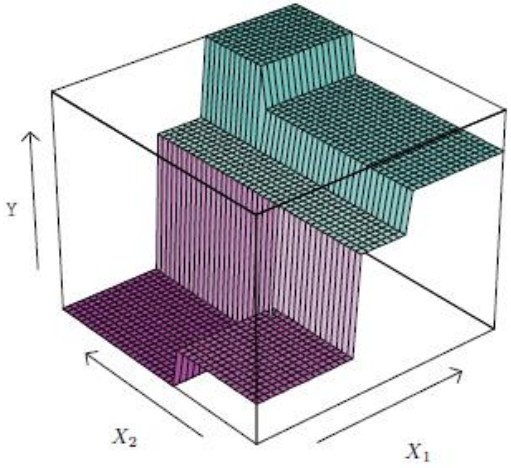



Years < 4.5
Ramas
Nodos internos
Hits < 117.5
5.11
6.00
6.74
Nodos terminales u hojas

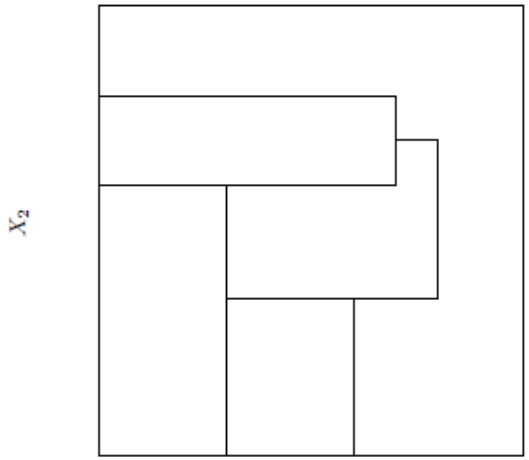
# Segmentación binaria recursiva

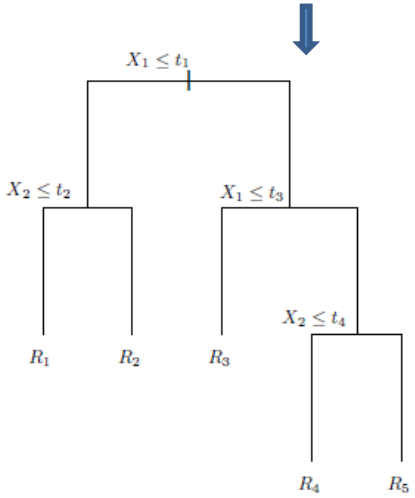
## Es Segmentación binaria recursiva



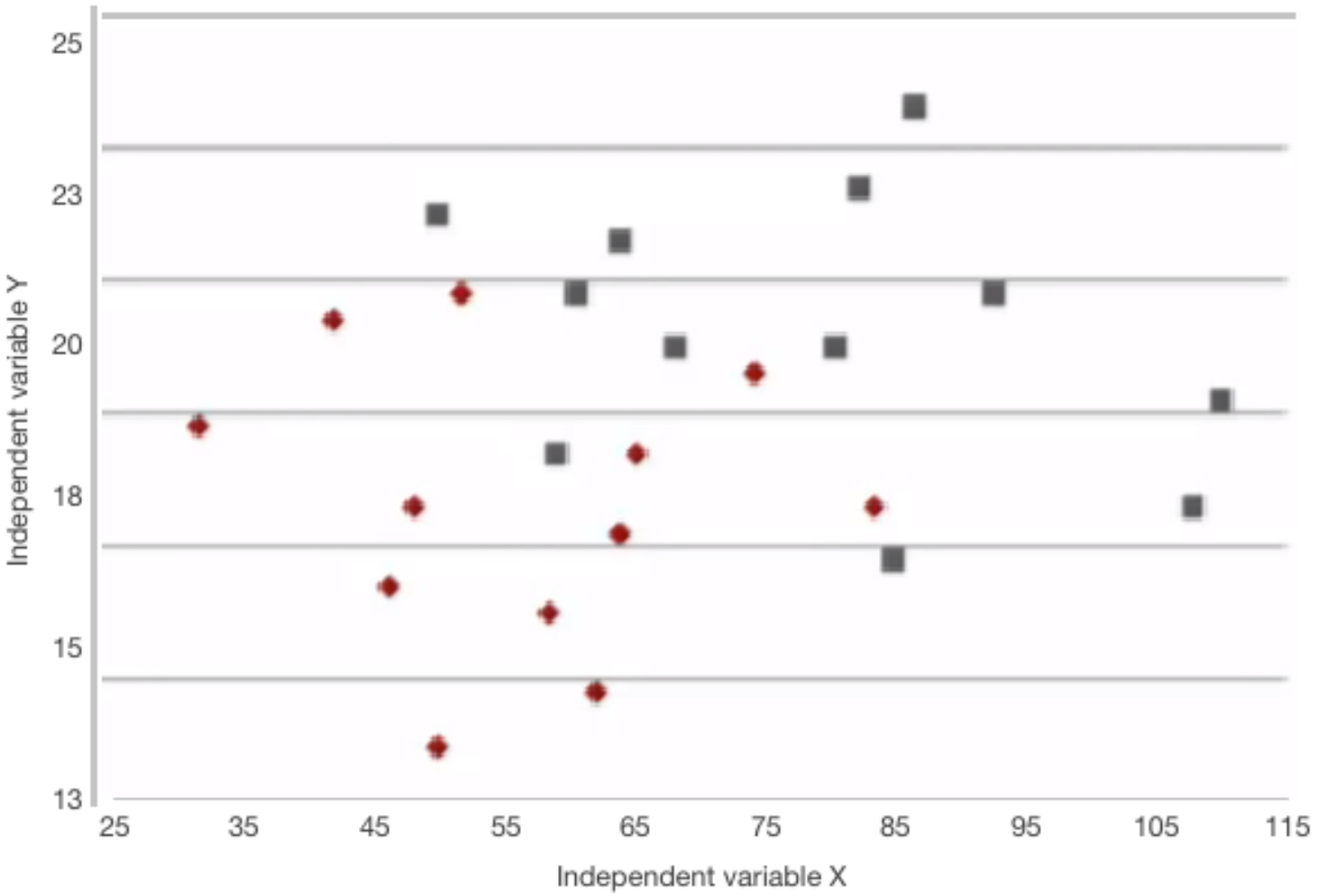



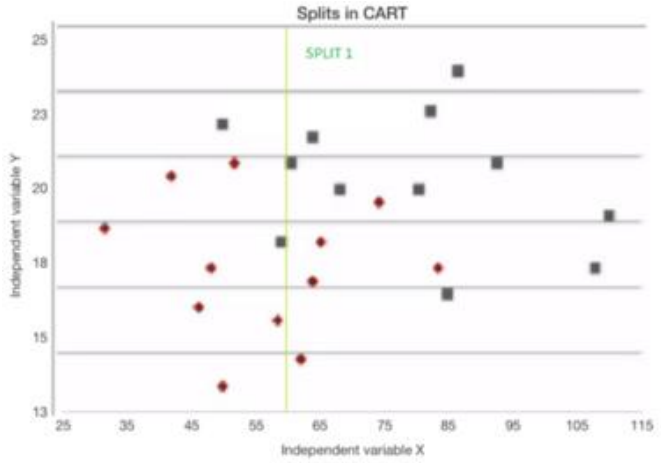

## No Es Segmentación binaria recursiva

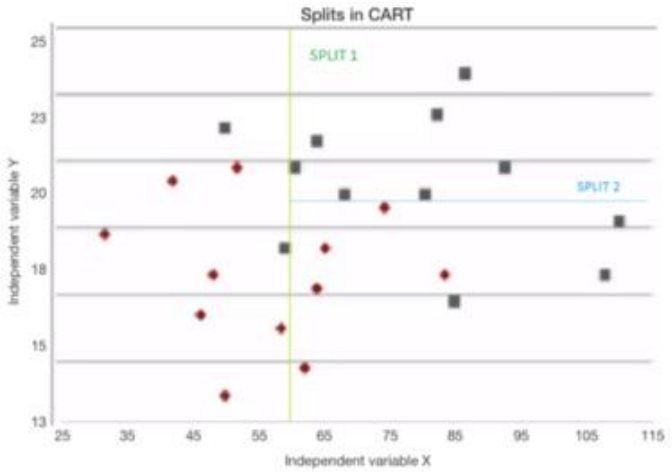


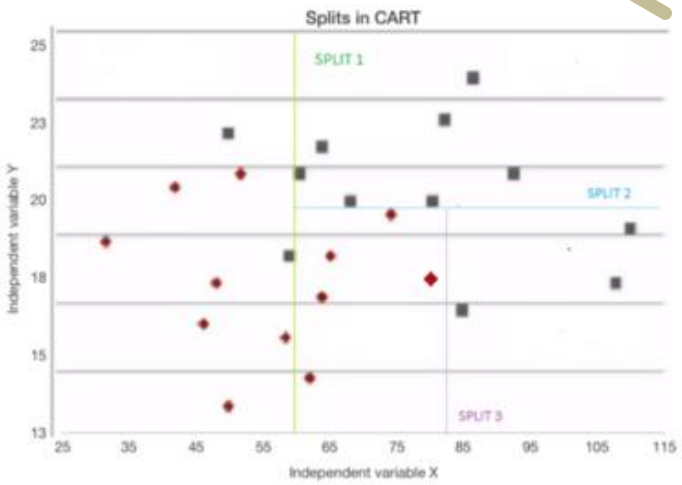



# Entrenamiento del Modelo

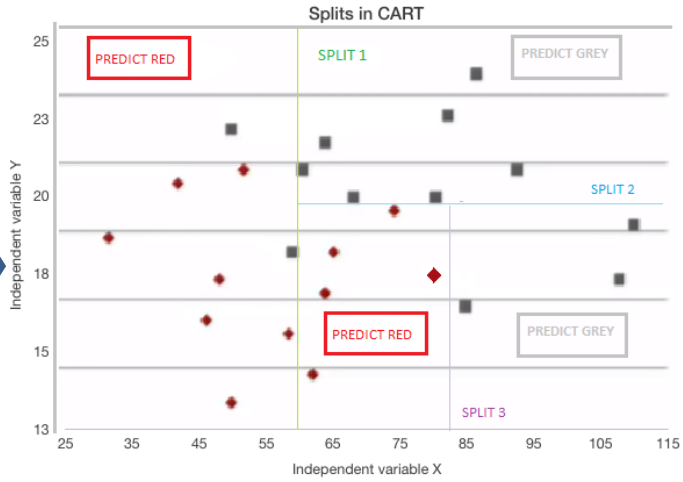


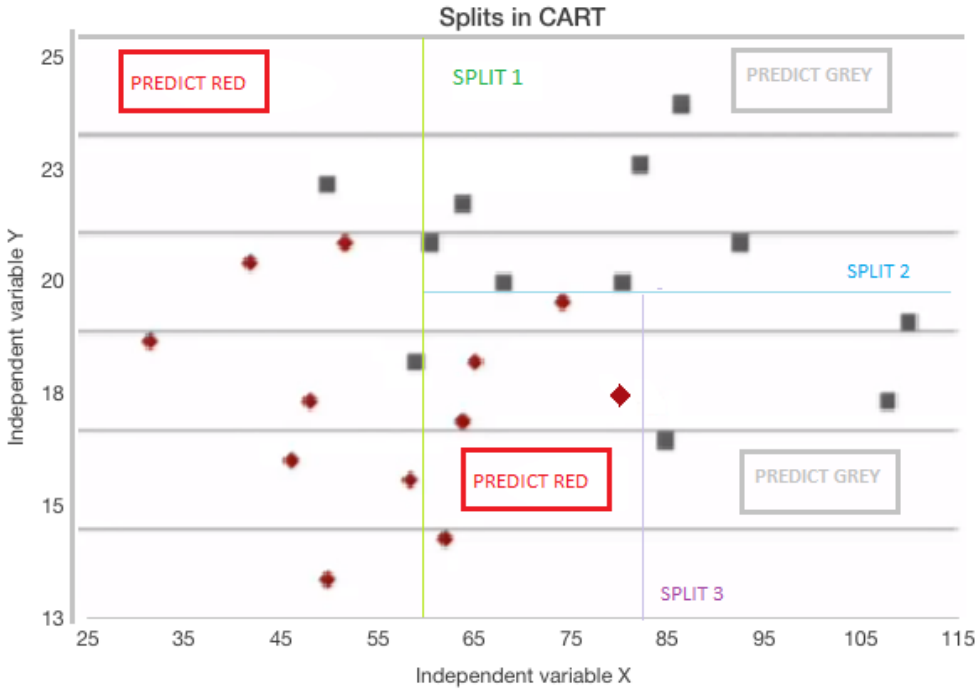

Splits in CART
SPLIT 1
Independent variable Y
Independent variable X

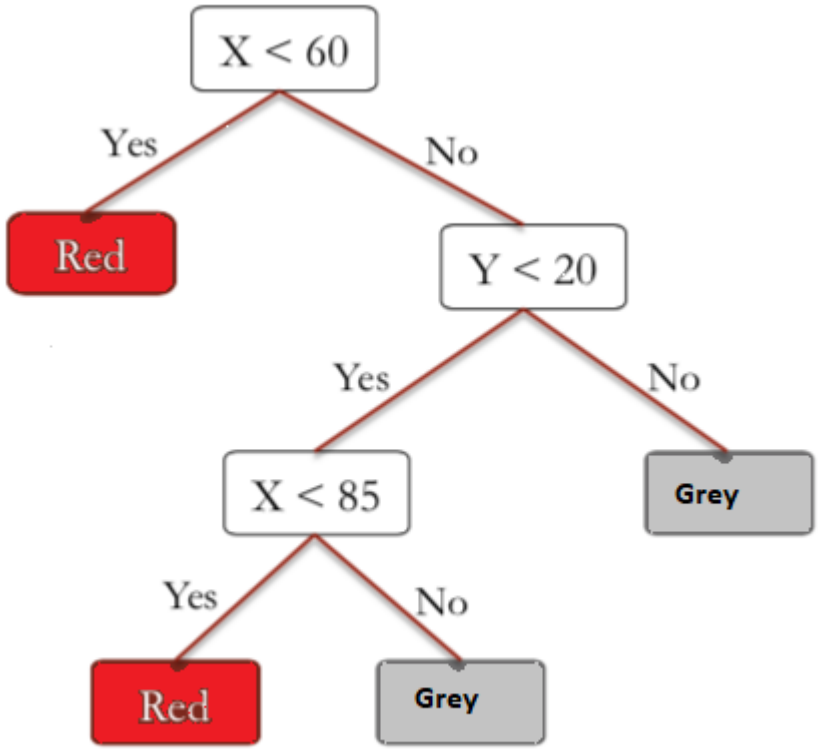

Splits in CART
SPLIT 1
SPLIT 2
Independent variable Y
Independent variable X


Splits in CART
SPLIT 1
SPLIT 2
SPLIT 3
Independent variable Y
Independent variable X

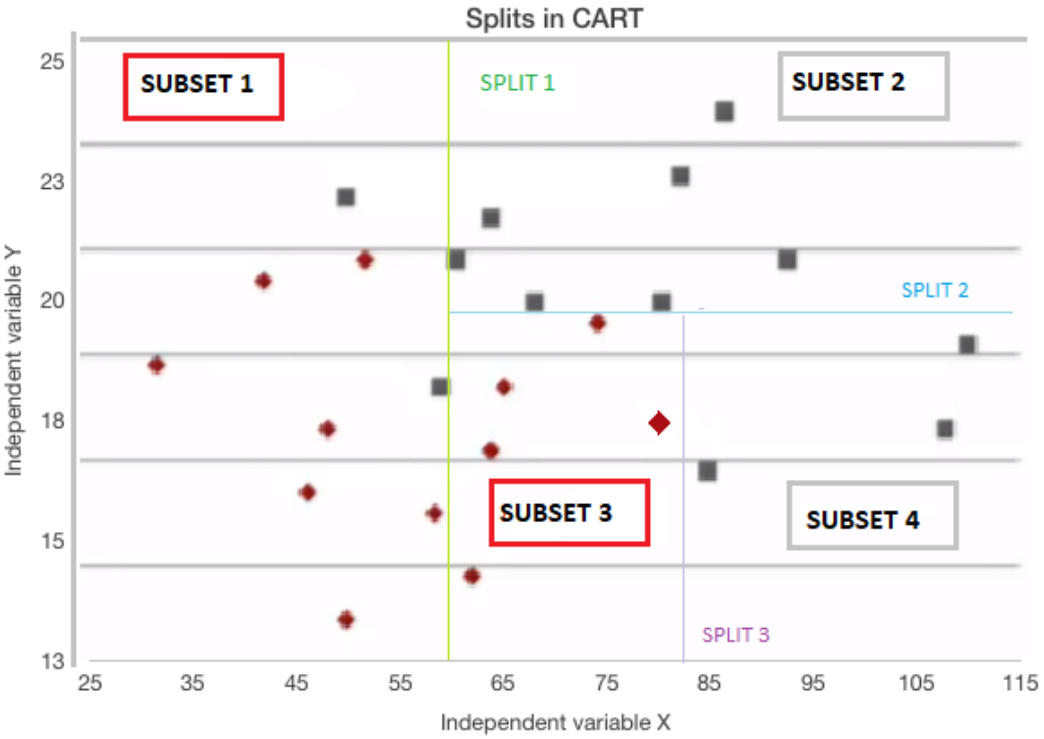
Modelo de AD


Splits in CART
PREDICT RED
SPLIT 1
PREDICT GREY
SPLIT 2
PREDICT RED
PREDICT GREY
SPLIT 3
Independent variable Y
Independent variable X


Splits in CART
PREDICT RED
SPLIT 1
PREDICT GREY
SPLIT 2
PREDICT RED
PREDICT GREY
SPLIT 3
Independent variable Y
Independent variable X
25
23
20
18
15
13
25
35
45
55
65
75
85
95
105
115


X < 60
Yes
No
Red
Y < 20
Yes
No
X < 85
Grey
Yes
No
Red
Grey

# Predicciones utilizando el Modelo




**Predicciones para Nodo 1** :
Rojos = 7, Negros= 2
Porc. de R = 7/(7+2) = 78%
Porc. de N = 22%.

Ahora, al igual que en Regresión Logística y Naive Bayes, podemos usar un valor umbral para clasificar al valor que caiga en ese subconjunto.
Un umbral de 0,5/50%, corresponde a la elección del resultado más frecuente que sería Rojo.

# Armado del Árbol e Decisión

Como es computacionalmente inviable analizar todas las posibles formas de armar las j regiones (fuerza bruta) es que se utiliza un mecanismo llamado ***partición recursiva binaria.*** Es un método HEURÍSTICO *por pasos o etapas.*

Procedimiento: Elegimos una variable $X_i$ y probamos con diferentes punto de corte ***s*** evaluando en cada una de las regiones $\{X|X_j < s\}\ \{X|X_j \geq s\}$ la SCE. El punto de corte que de la menor SCE será el que utilizaremos para delimitar la ramificaciones $R_i$ del árbol

$$R_1(j,s) = \{X|X_j < s\} \quad \text{y} \quad R_2(j,s) = \{X|X_j \geq s\}$$

Y buscaremos minimizar la suma de los SCE que llamaremos **Costo,** de las regiones recién conformadas por la partición binaria:

$$\sum_{i:\ x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\ x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

Siendo $\hat{y}_{R_1}$ la media muestral de las observaciones de la región $R_1(j,s)$ y $\hat{y}_{R_2}$ la media muestral de $R_2(j,s)$.

Este proceso se repite en cada una de las ramificaciones hasta llegar a un pto de parada, que podría ser la cantidad de datos en las hojas o nodos terminales (p.e. mínimo 5 datos)

# Árboles de Decisión en Clasificación

Proporción de la clases *k* en las observaciones del nodo *m*

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$$

Clasificamos la nueva observación del nodo *m* a la clase *k* que cumpla:

$k(m) = \arg\max_k \hat{p}_{mk}$, Si k=2 será: $\max(p, 1-p)$

El error o **costo** al clasificar mediante este modelo será:

$$\frac{1}{N_m} \sum_{i \in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_{mk(m)} = 1 - \max(p, 1-p)$$

PROBLEMA: este modelo de clasificación tiene el problema que no privilegia la *pureza* de los nodos resultantes. Veamos un ejemplo: Tenemos un nodo con la siguiente distribución de clases (400,400). Aplicamos dos modelos diferentes de clasificación, generando estos dos splits :

**S1:** (300,100) y (100,300) y **S2:** (200,400) y (200,0)

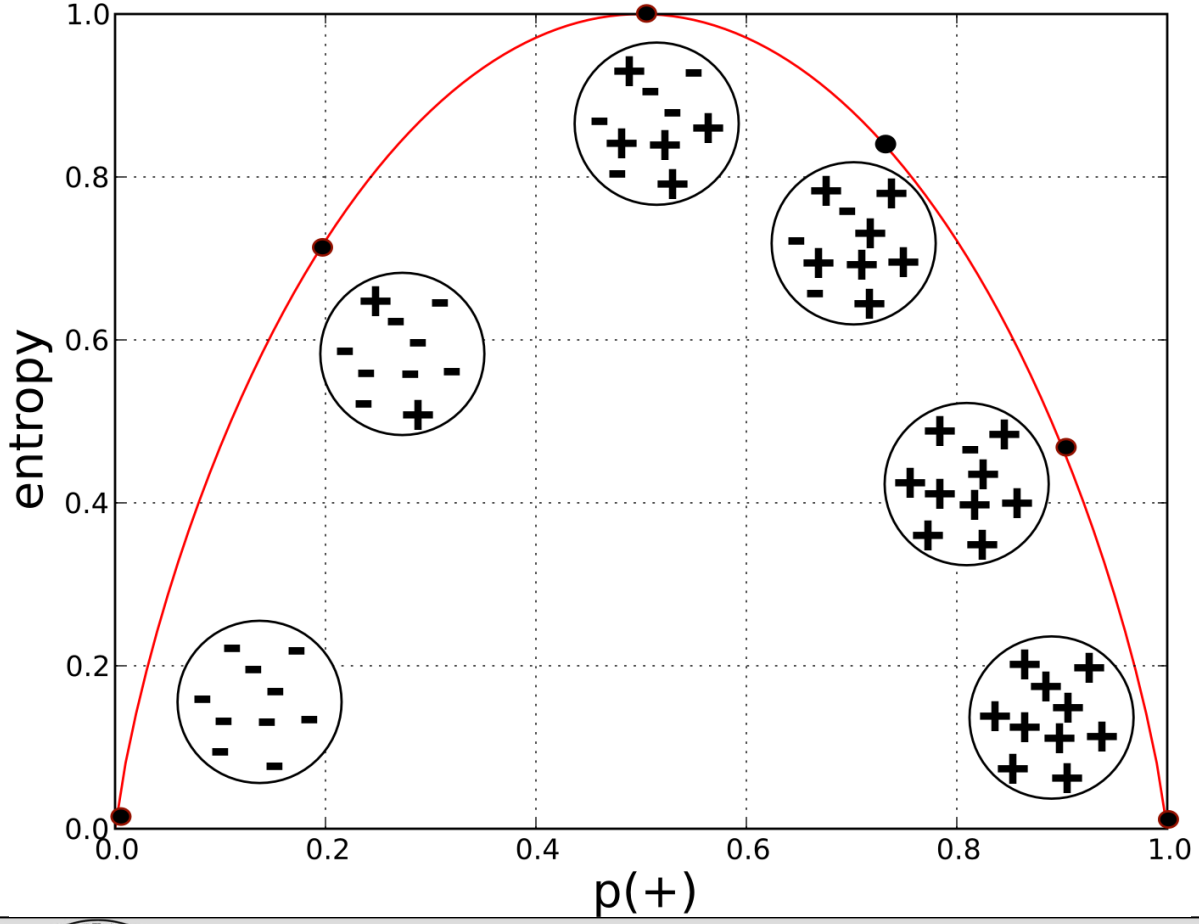
El error de clasificación para cada modelo sería:
E(S1)=0,5*1/400*100 + 0,5*1/400*100 = **0,25**
Ec(S2)=0,75*1/600*200+ 0,25*1/200*0 = **0,25**

**El segundo modelo clasifica mejor (al tener un nodo de pureza 100%) . Debemos encontrar otra forma de medir el ERROR de Clasificación**

# Medida de impureza en los nodos

La entropía mide el grado de desorden o impureza entre los miembros de un grupo.
A partir de paper de Shannon (1948):

$$\textbf{\textit{entropy}} = - p_1 \log (p_1) - p_2 \log (p_2) - \ldots - p_k \log (p_k)$$




*entropy*(*S*) = - 0.7 × $\log_2$ (0.7) - -0.3 × $\log_2$ (0.3)= 0.88

*entropy*(*S*) = - 0.9 × $\log_2$ (0.9) - -0.1 × $\log_2$ (0.1)= 0.47

*entropy*(*S*)= -2*0,5*× $\log_2$ (0,5)=1

*entropy*(*S*) = - 0.2 × $\log_2$ (0.2) - 0.8 × $\log_2$(0.2) =0,72

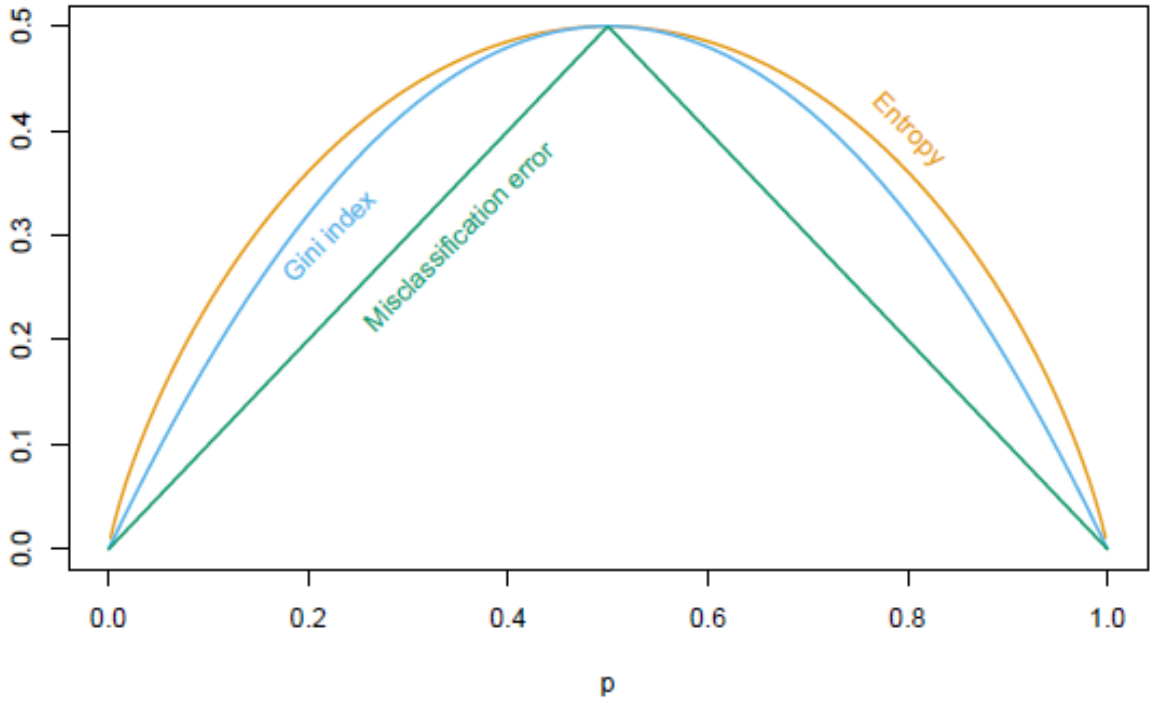
*entropy*(*S*) = - 1 × $\log_2$(1)=0

Si la variable respuesta tiene 2 clases y p es la proporción de la clase 2, entonces:

$$\frac{1}{N_m}\sum_{i\in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_{mk(m)} = 1 - \max(p, 1-p)$$

$$\sum_{k\neq k'} \hat{p}_{mk}\hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1-\hat{p}_{mk}) = 2p(1-p) \qquad \textbf{Gini}$$

$$-\sum_{k=1}^{K} \hat{p}_{mk}\log\hat{p}_{mk} = -p\log p - (1-p)\log(1-p) \qquad \textbf{Entropía}$$




Las tres son similares
pero Gini y Entropía cruzada son diferenciables

→ mejor para aplicar métodos numéricos

→ Aparte Gini y E.C cumplen las premisas de la entropía de la información

**ejemplo**

# Ganancia de Información *(Information gain)*

la ganancia de información mide el cambio en la entropía debido a cualquier cantidad de nueva información agregada. En el contexto de los árboles de decisión, se refiere a la información obtenida al crear nuevos nodos en base a los valores de un cierto atributo. Digamos que el atributo divide al grupo original en k nodos diferentes. Llamemos al nodo original conjunto "padre" y a los nodos originados como resultado de la división en base al atributo, nodos "hijos".

$$IG(padre, hijos) = entropía(padre) - p(hijo_1) \times entropía(hijo_1) - p(hijo_2) \times entropía(hijo_2)$$

Siendo :

$$p(hijo_1) = N(hijo_1) / N(padre)$$
$$p(hijo_2) = N(hijo_2) / N(padre)$$

$$entropía(hijo_1) = -p_1 * log(p_1) - (1-p_1) * log(1-p_1)$$
$$entropía(hijo_2) = -p_2 * log(p_2) - (1-p_2) * log(1-p_2)$$

En A.D, de decisiones la IG mide la disminución de la entropía del árbol debida a la nueva nueva ramificación.

**Ejemplo, k=2, (400,400)**
Error(orig)=1/800*400=0,5
Gini(original)=2*0,5*0,5=0,5
Entropía(original)= -2*0,5*$\log_2$(0,5)=1

$$\frac{1}{N_m}\sum_{i\in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_{mk(m)} = 1 - \max(p, 1-p)$$

$$\sum_{k\neq k'} \hat{p}_{mk}\hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1-\hat{p}_{mk}) = 2p(1-p) \quad \textbf{Gini}$$

$$-\sum_{k=1}^{K} \hat{p}_{mk}\log\hat{p}_{mk} = -p\log p - (1-p)\log(1-p) \quad \textbf{Entropía}$$

Supongamos probamos dos modelos de clasificación, que nos dan los sig. splits :

**S1:** (300,100) y (100,300) y **S2:** (200,400) y (200,0)

Los errores de clasificación para cada uno son:
E(S1)=0,5*1/400*100 + 0,5*1/400*100 = **0,25**
Ec(S2)=0,75*1/600*200+ 0,25*1/200*0 = **0,25**

Son iguales, cuando se debería privilegiar a la segmentación que dio un nodo puro al 100%

Con Gini y entropía da:
**Gini:** Ec(S1)=0,5*(2*0,75*(1-0,75))+0,5*2*0,75*(1-0,75)=**0,375**
Ec(S2)= 0,75*(2*0,666*(1-0,666)) =**0,336**

**Entropía:** Ec(S1)=0,5*(-0,75*$\log_2$0,75-0,25*$\log_2$0,25)+0,5*(-0,75*$\log_2$0,75-0,25*$\log_2$0,25)=**0,81**
Ec(S2)=0,75*(-0,33*$\log_2$0,33-0,66*$\log_2$0,66) =**0,69**

Veamos la ganancia de información (IG) en el ejemplo de recién

$IG(padre, hijos) = entropía(padre) - p(hijo_1) \times entropía(hijo_1) - p(hijo_2) \times entropía(hijo_2)$

**k=2, (400,400)**

**IG (S1) = 1 – 0,81= 0,19**
**IG (S2) = 1 – 0,69= 0,31**

**Mayor pureza**

## Pseudo Código de algoritmo de Árboles de Decisión (de CART, Breiman at.al. 1984)

```
BEGIN:  Assign all training data to the root node
        Define the root node as a terminal node

SPLIT:
New_splits=0
FOR every terminal node in the tree:
   If the terminal node sample size is too small or all instances in the
   node belong to the same target class goto GETNEXT
   Find the attribute that best separates the node into two child nodes
   using an allowable splitting rule
   New_splits+1
GETNEXT:
NEXT
```

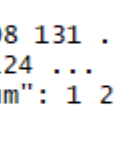
# Árboles de Decisión en R

Ejemplo: Data Set Carseats

| | CompPrice | Income | Advertising | Population | Price | ShelveLoc | Age | Education | Urban | US | High |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 138 | 73 | 11 | 276 | 120 | Bad | 42 | 17 | Yes | Yes | Alta |
| 2 | 111 | 48 | 16 | 260 | 83 | Good | 65 | 10 | Yes | Yes | Alta |
| 3 | 113 | 35 | 10 | 269 | 80 | Medium | 59 | 12 | Yes | Yes | Alta |
| 4 | 117 | 100 | 4 | 466 | 97 | Medium | 55 | 14 | Yes | Yes | Baja |
| 5 | 141 | 64 | 3 | 340 | 128 | Bad | 38 | 13 | Yes | No | Baja |
| 6 | 124 | 113 | 13 | 501 | 72 | Bad | 78 | 16 | No | Yes | Alta |
| 7 | 115 | 105 | 0 | 45 | 108 | Medium | 71 | 15 | Yes | No | Baja |
| 8 | 136 | 81 | 15 | 425 | 120 | Good | 67 | 10 | Yes | Yes | Alta |
| 9 | 132 | 110 | 0 | 108 | 124 | Medium | 76 | 10 | No | No | Baja |
| 10 | 132 | 113 | 0 | 131 | 124 | Medium | 76 | 17 | No | Yes | Baja |
| 11 | 121 | 78 | 9 | 150 | 100 | Bad | 26 | 10 | No | Yes | Alta |
| 12 | 117 | 94 | 4 | 503 | 94 | Good | 50 | 13 | Yes | Yes | Alta |
| 13 | 122 | 35 | 2 | 393 | 136 | Medium | 62 | 18 | Yes | No | Baja |
| 14 | 115 | 28 | 11 | 29 | 86 | Good | 53 | 18 | Yes | Yes | Alta |
| 15 | 107 | 117 | 11 | 148 | 118 | Good | 52 | 18 | Yes | Yes | Alta |
| 16 | 149 | | | | | | | | | | Alta |
| 17 | 118 | | | | | | | | | | Baja |

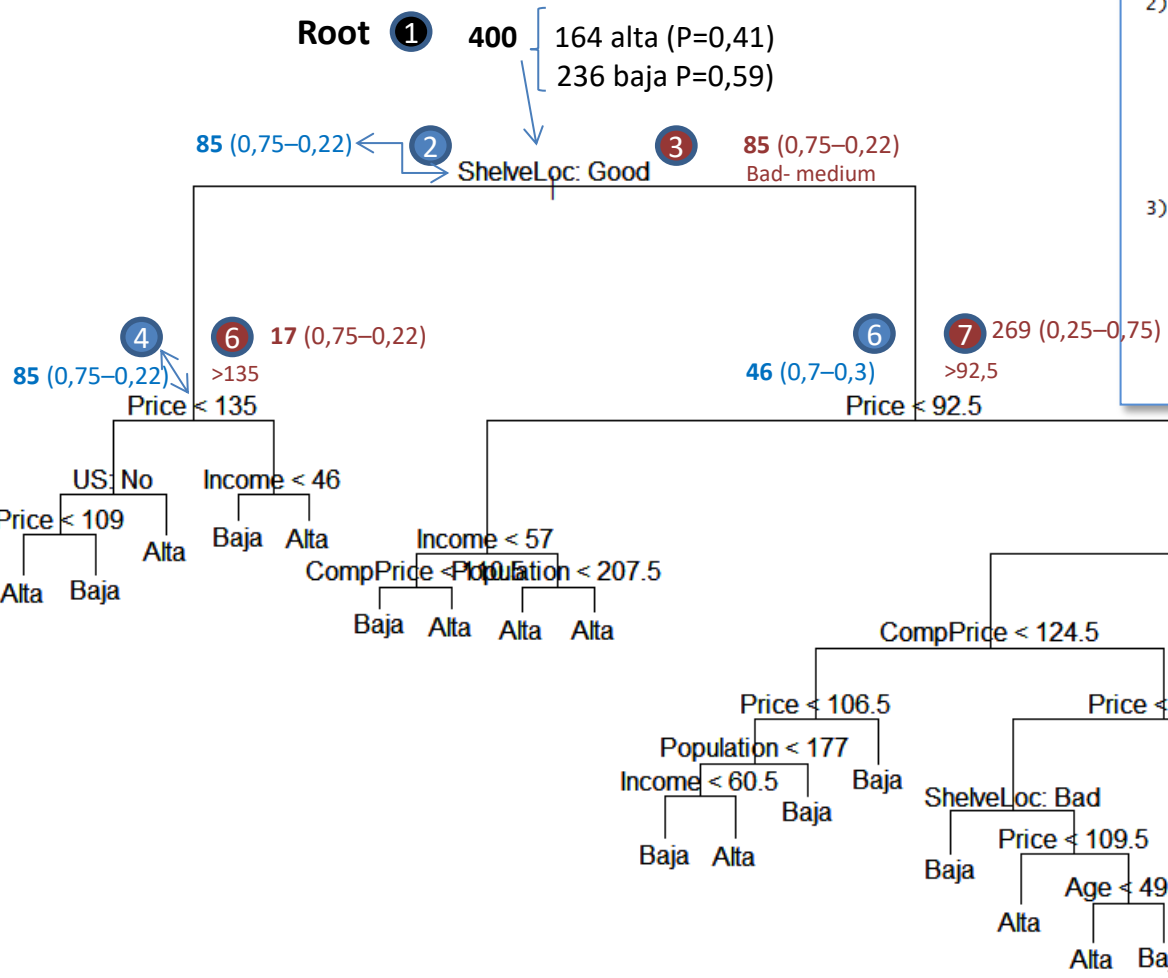
Showing 1 to 17 of 400 entrie

```
'data.frame':   400 obs. of  11 variables:
 $ CompPrice  : num  138 111 113 117 141 124 115 136 132 132 ...
 $ Income     : num  73 48 35 100 64 113 105 81 110 113 ...
 $ Advertising: num  11 16 10 4 3 13 0 15 0 0 ...
 $ Population : num  276 260 269 466 340 501 45 425 108 131 ...
 $ Price      : num  120 83 80 97 128 72 108 120 124 124 ...
 $ ShelveLoc  : Factor w/ 3 levels "Bad","Good","Medium": 1 2 3 3 1 1 3
 $ Age        : num  42 65 59 55 38 78 71 67 76 76 ...
 $ Education  : num  17 10 12 14 13 16 15 10 10 17 ...
 $ Urban      : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 1 1 ...
 $ US         : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 2 1 2 ...
 $ High       : Factor w/ 2 levels "Alta","Baja": 1 1 1 2 2 1 2 1 2 2 .
```



**Clase**

# Gráfico de Árboles de Decisión en R




```
1) root 400 541.500 Baja ( 0.41000 0.59000 )
  2) ShelveLoc: Good 85  90.330 Alta ( 0.77647 0.22353 )
    4) Price < 135 68  49.260 Alta ( 0.88235 0.11765 )
      8) US: No 17  22.070 Alta ( 0.64706 0.35294 )
       16) Price < 109 8   0.000 Alta ( 1.00000 0.00000 ) *
       17) Price > 109 9  11.460 Baja ( 0.33333 0.66667 ) *
      9) US: Yes 51  16.880 Alta ( 0.96078 0.03922 ) *
    5) Price > 135 17  22.070 Baja ( 0.35294 0.64706 )
     10) Income < 46 6   0.000 Baja ( 0.00000 1.00000 ) *
     11) Income > 46 11  15.160 Alta ( 0.54545 0.45455 ) *
  3) ShelveLoc: Bad,Medium 315 390.600 Baja ( 0.31111 0.68889 )
    6) Price < 92.5 46  56.530 Alta ( 0.69565 0.30435 )
     12) Income < 57 10  12.220 Baja ( 0.30000 0.70000 )
       24) CompPrice < 110.5 5   0.000 Baja ( 0.00000 1.00000 ) *
       25) CompPrice > 110.5 5   6.730 Alta ( 0.60000 0.40000 ) *
     13) Income > 57 36  35.470 Alta ( 0.80556 0.19444 )
       26) Population < 207.5 16  21.170 Alta ( 0.62500 0.37500 )
       27) Population > 207.5 20   7.941 Alta ( 0.95000 0.05000 )
    7) Price > 92.5 269 299.800 Baja ( 0.24535 0.75465 )
```

## Procedimiento de Poda merdiante evaluación de Costo Complejidad *(Del Libro de Breiman Friedmann Ohlsen CART - 1984)*

Luego de construir (entrenar) un árbol de decisión sólo indicando la cantidad mínima de observaciones en los nodos terminales (hojas) ahora debemos decidir con qué parte de ese árbol nos quedaremos en orden de incrementar la capacidad de generalización del modelo Sean $T_1, T_2, \ldots, T_k$ los nodos terminales de un árbol de decisión $T$. Definamos:

$|T|$ = número de nodos terminales u hojas del árbol $T$

*Llamaremos*

***Costo o Riesgo*** *(risk)* ***de T*** $= R(T) = \sum_{i=1}^{k} P(T_i) R(T_i)$ **(1)**

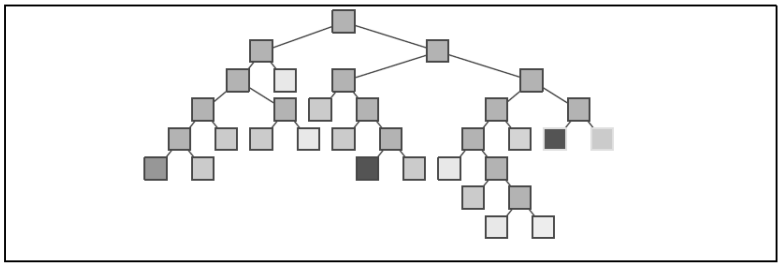
siendo $P(T_i)$ la proporción de instancias en el nodo *i* y $R(Ti)$ la proporción de instancias mal clasificados en el nodo *i*

## Penalización por Costo- Complejidad - Determinación del α

Ahora llamemos ***Cp*** (Parámetro de Complejidad por sus siglas en inglés) al parámetro que mide el "costo de adicionar" una nueva ramificación al modelo, quedándonos la siguiente medida de Costo Complejidad :

$$R_\alpha(T) = R(T) + \alpha|T| \qquad \textbf{(2)}$$

El proceso de poda va removiendo iterativamente los nodos terminales más débiles del árbol (que son los que más errores tienen) absorbiendo los dos hijos su nodo padre.



EL algoritmo de pruning en cada poda calcula el costro-Complejidad (formula 2) en los datos de entrenamiento, y devuelve el sub-árbol podado con el $R\alpha(T)$ *mínimo..*

# PSEUDOALGORITMO DE PODA

DEFINICIONES

*#info de nodo*

r(i) = porcentaje de instancias mal clasificadas en nodo i
p(i) = proporción de instancias en nodo 8
i_left = nodo hijo de la izquierda del nodo padre i
i_right = nodo hijo de la derecha del nodo padre i

*#tree info*

|T| = número de nodos del árbol T
alfa = parámetro de complejidad
R(i) = r(i)*p(i)
Risk(T) = SUM_i [ R(i)]
Risk_alfa(T) = Risk(T) + alfa*|T|

INICIO:

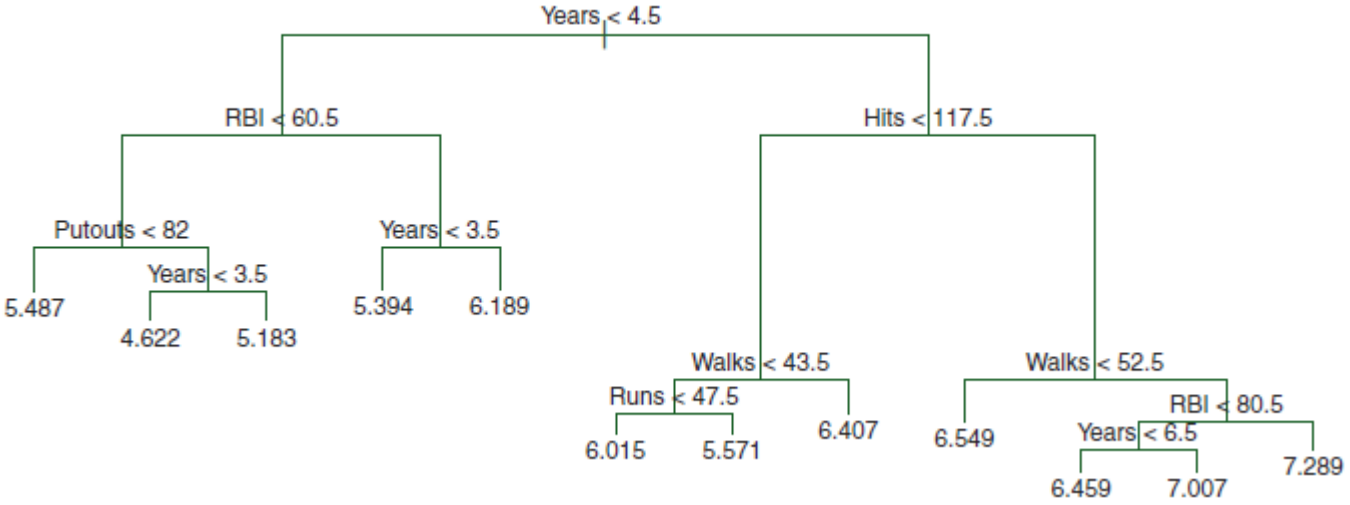
Tmax = árbol_tamaño_max
Risk_alfa_min = Risk(Tmax) + alfa*|Tmax|
Current_Tree = Tmax

PRUNE: **Do While** |Current_tree| > 1
**For** all parents i of two terminal nodes in Current_Tree
**Remove** node(s) i for which R(i) - R(i_left) - R(i_right) is **minimum**
Current_Tree = Current_Tree - 1
**If** {Risk_alfa(Current_Tree) < Risk_alfa_min) } **then**
Risk_alfa_min= Risk_alfa(Current_Tree)

## Algoritmo de poda

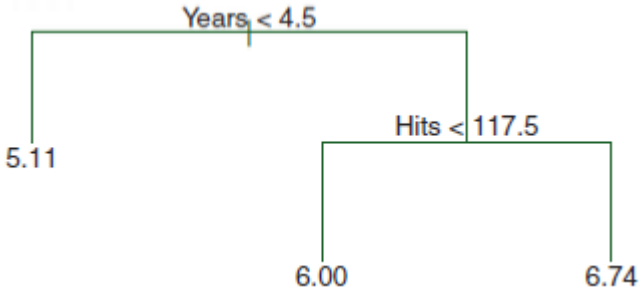
Para seleccionar el α óptimo debemos seguir los siguientes pasos:

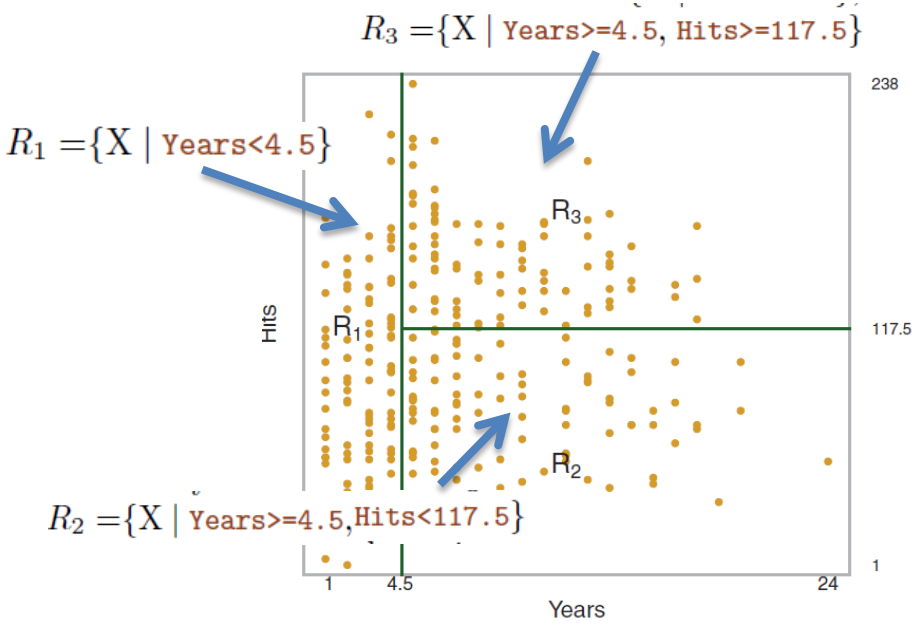
1. Utilizamos el método de partición binaria recursiva para hacer crecer el árbol, parando cuando cada uno de los nodos terminales contengan una cierta cantidad mínima de observaciones
2. Aplicamos poda por costo-complejidad al árbol anterior y obtenemos una secuencia de sub-arboles de decisión en función de diferentes α
3. Hacemos validación cruzada para elegir el α .
    a. Repetimos los pasos 1 y 2 sobre cada set de entrenamiento de CV (o sea todos menos el k-ésimo sub-set)
    b. Evaluamos el error de predicción en el k-ésimo subset .
    c. Luego promediamos los errores del pto b para todos los k-ésimo subsets , quedándonos como el error representativo del sub-árbol podado por el parámetro de regularización α.
    d. Elegimos el α con el menor error de predicción del pto c.
4. Con el valor de α recién hallado volvemos al árbol correspondiente a ese α del punto 2, que es el que va a ser nuestro modelo de clasificación.

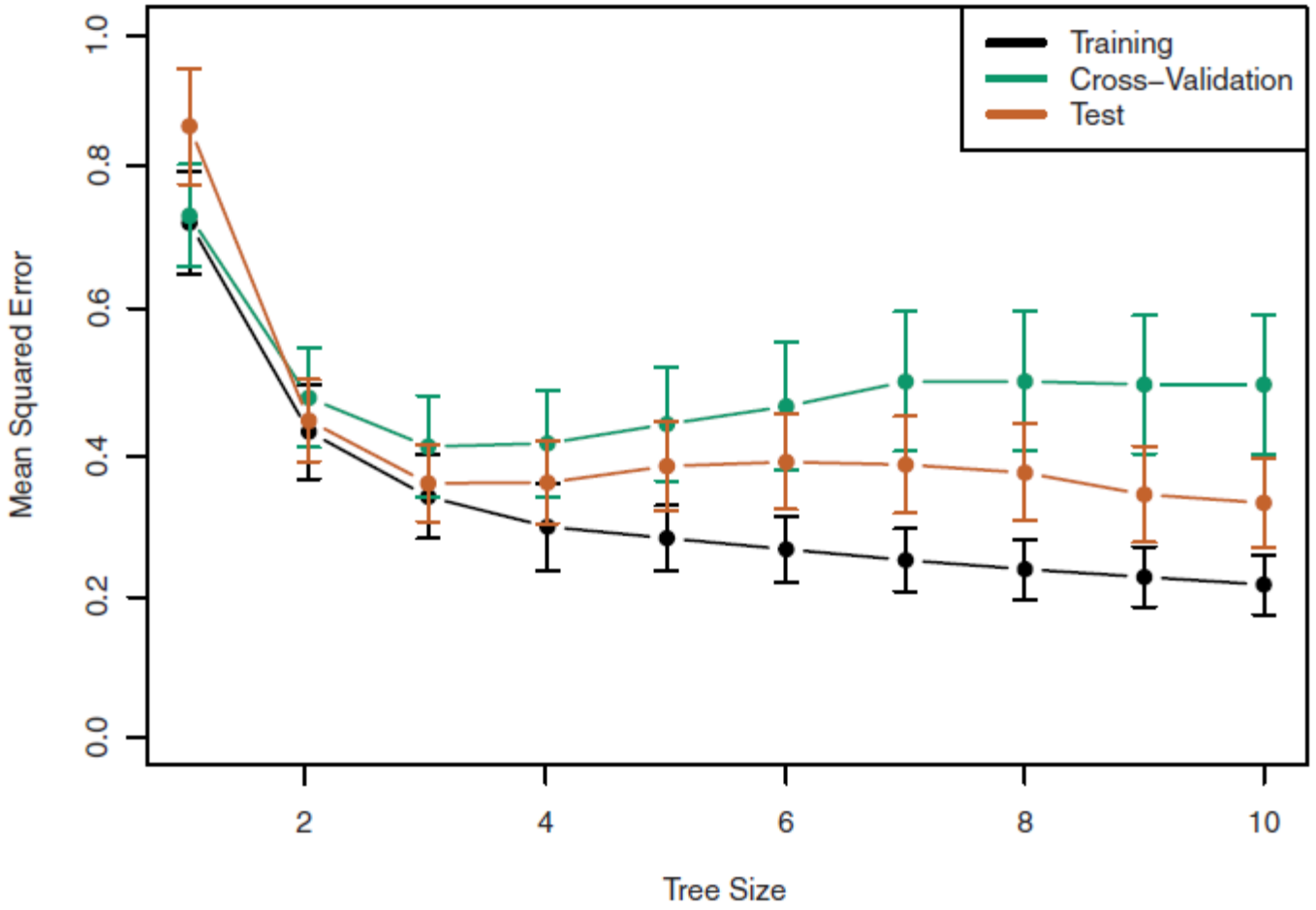



**9 covariantes**

**2 covariantes**




Training
Cross–Validation
Test
Mean Squared Error
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10
Tree Size