Introducción a CART's

Germán Rosati

IDAES/UNSAM - CONICET - PIMSA

29 de Octubre de 2020

¿Qué es un árbol de decisión?

- Técnica para regresión y clasificación
- Dividir el espacio de predictores en regiones o espacios simples
- Simples, interpretables y claros
- Baja performance predictiva

- Vamos a construir el árbol de decisión para poder saber si podemos o no jugar al golf.

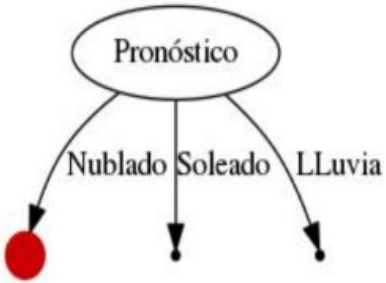
- El set de entrenamiento es la tabla de la derecha.

Pronóstico	Temperatura	Humedad	Viento	Jugar
Soleado	Alta	Alta	Débil	No
Soleado	Alta	Alta	Fuerte	No
Nublado	Alta	Alta	Débil	Si
LLuvia	Media	Alta	Débil	Si
LLuvia	Baja	Normal	Débil	Si
LLuvia	Baja	Normal	Fuerte	No
Nublado	Baja	Normal	Fuerte	Si
Soleado	Media	Alta	Débil	No
Soleado	Baja	Normal	Débil	Si
LLuvia	Media	Normal	Débil	Si
Soleado	Media	Normal	Fuerte	Si
Nublado	Media	Alta	Fuerte	Si
Nublado	Alta	Normal	Débil	Si
LLuvia	Media	Alta	Fuerte	No

- En primer lugar debemos verificar si todos los registros pertenecen a la misma clase, ya que en ese caso deberíamos construir un nodo hoja con esa clase como etiqueta. Como no es el caso, vamos a particionar nuestro set según la variable Pronóstico.

- Por cada partición creamos una arista y un nodo hijo.

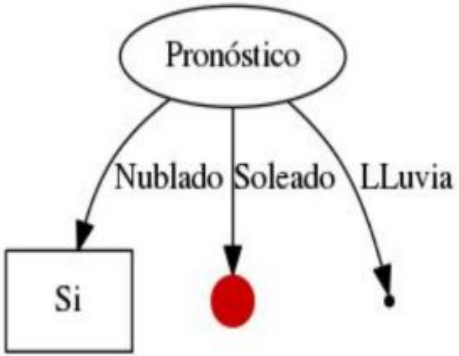
- Ahora aplicamos recursivamente el algoritmo. En primer lugar analizamos la partición correspondiente a Pronóstico=Nublado.



Pronóstico	Temperatura	Humedad	Viento	Jugar
Nublado	Alta	Alta	Débil	Si
Nublado	Baja	Normal	Fuerte	Si
Nublado	Media	Alta	Fuerte	Si
Nublado	Alta	Normal	Débil	Si

- Al analizar a que clase pertenecen los registros, vamos que todos corresponden a "Si" por lo tanto este será un nodo hoja con la etiqueta "Si".

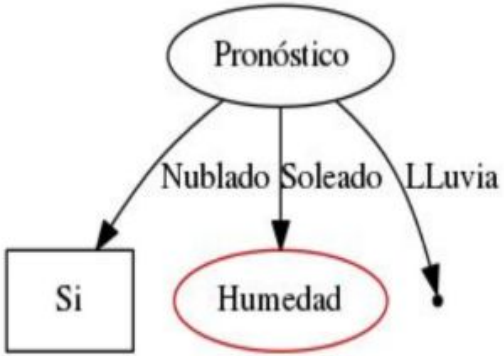
- Ahora analicemos la partición correspondiente a Pronóstico=Soleado.



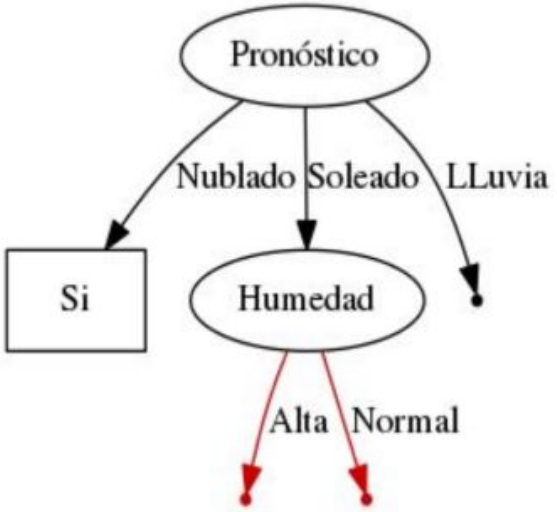
Pronóstico	Temperatura	Humedad	Viento	Jugar
Soleado	Alta	Alta	Débil	No
Soleado	Alta	Alta	Fuerte	No
Soleado	Media	Alta	Débil	No
Soleado	Baja	Normal	Débil	Si
Soleado	Media	Normal	Fuerte	Si

- Como podemos ver, los registros pertenecen a distintas clases, por esta razón tendremos que sub-particionar. Podemos optar por particionar según las siguientes variables: {Temperatura, Humedad y Viento}.
- ¿Cual conviene utilizar?

- Al analizar los datos, resulta evidente que la forma más simple es particionar por Humedad, ya que si seleccionásemos Viento o Temperatura, deberíamos hacer una división inferior más.



- Ahora que tenemos seleccionado el criterio, creamos las particiones.

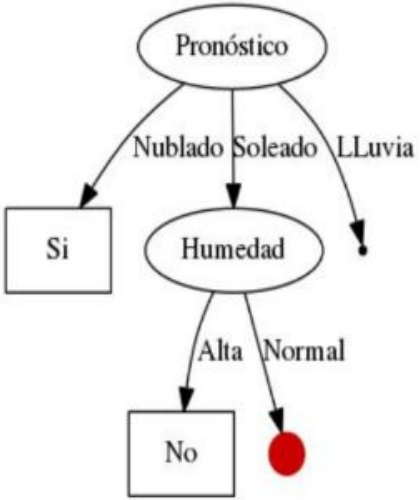


- Ahora debemos aplicar recursivamente el algoritmo. Para la sub-partición Humedad=Alta tenemos este caso:

Pronóstico	Temperatura	Humedad	Viento	Jugar
Soleado	Alta	Alta	Débil	No
Soleado	Alta	Alta	Fuerte	No
Soleado	Media	Alta	Débil	No

- Como podemos ver que todos los registros pertenecen a la clase "No", sabemos que será un nodo hoja con la etiqueta "No"

- Para la sub-partición Humedad=Normal tenemos este caso:

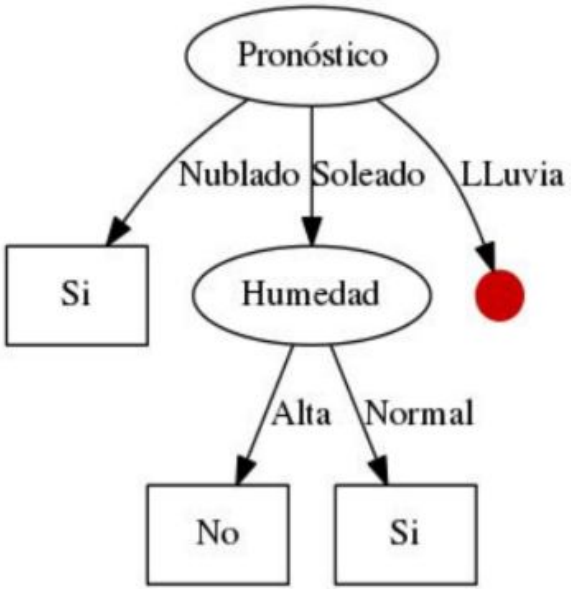


Pronóstico	Temperatura	Humedad	Viento	Jugar
Soleado	Baja	Normal	Débil	Si
Soleado	Media	Normal	Fuerte	Si

- Como podemos ver que todos los registros pertenecen a la clase "Si", sabemos que será un nodo hoja con la etiqueta "Si"

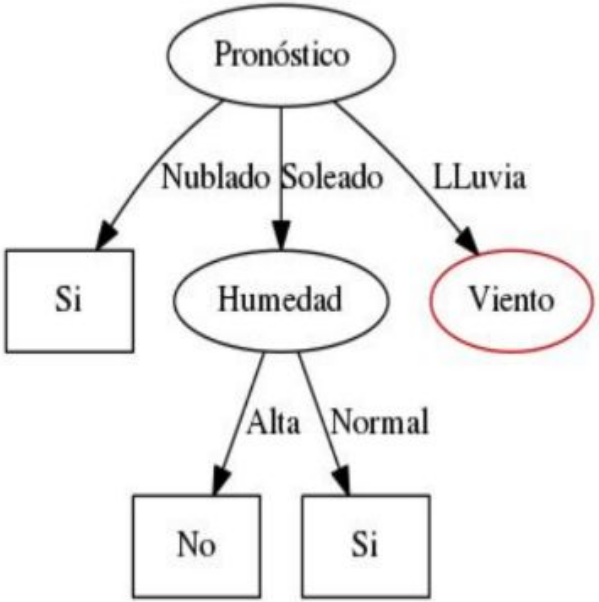
- Ahora resta analizar la partición correspondiente a Pronóstico=LLuvia.

Pronóstico	Temperatura	Humedad	Viento	Jugar
LLuvia	Media	Alta	Débil	Si
LLuvia	Baja	Normal	Débil	Si
LLuvia	Baja	Normal	Fuerte	No
LLuvia	Media	Normal	Débil	Si
LLuvia	Media	Alta	Fuerte	No

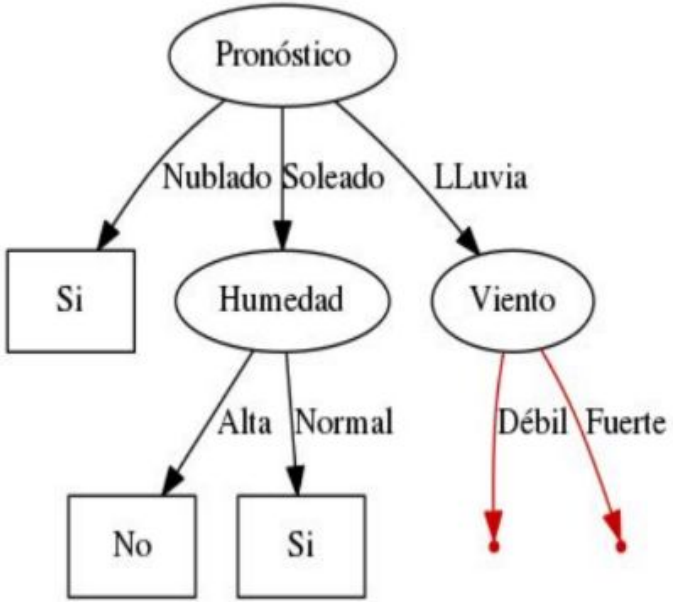


- Nuevamente vemos que los registros pertenecen a clases distintas y tendremos que sub-particionar. Podemos optar por particionar según las siguientes variables: {Temperatura, Humedad y Viento}.
- ¿Cual conviene utilizar?

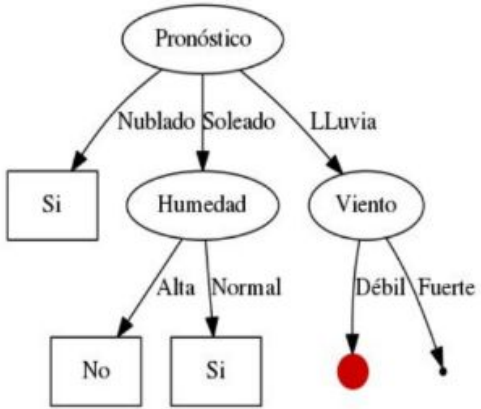
- En este caso resulta más conveniente utilizar como criterio de partición Viento.



- Ahora que tenemos seleccionado el criterio, creamos las particiones.

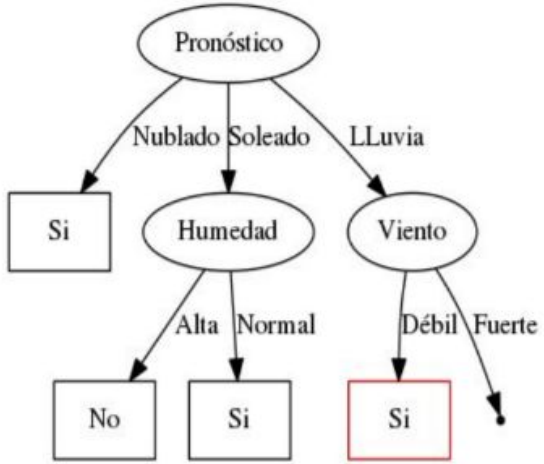


- Aplicamos nuevamente el algoritmo de forma recursiva. Para la sub-partición Viento=Débil tenemos este caso:

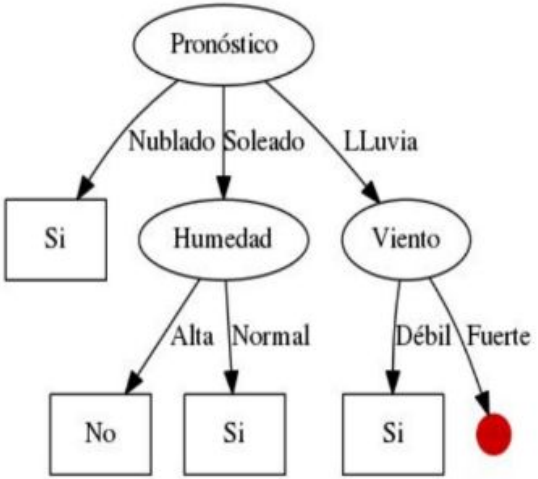


Pronóstico	Temperatura	Humedad	Viento	Jugar
LLuvia	Media	Alta	Débil	Si
LLuvia	Baja	Normal	Débil	Si
LLuvia	Media	Normal	Débil	Si

- Como podemos ver que todos los registros pertenecen a la clase "Si", sabemos que será un nodo hoja con la etiqueta "Si"

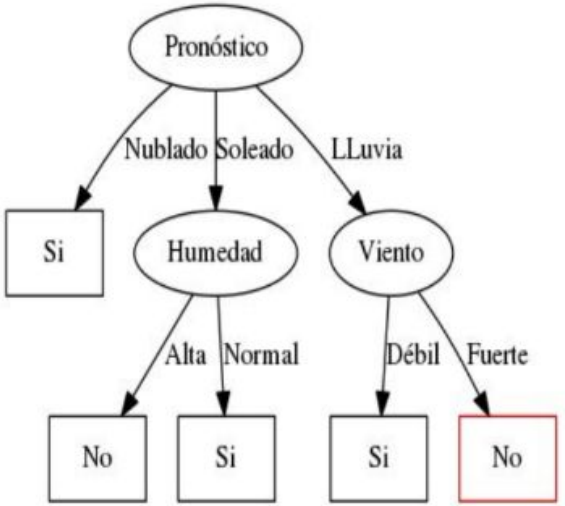


- Para la sub-partición Viento=Fuerte tenemos este caso:



Pronóstico	Temperatura	Humedad	Viento	Jugar
LLuvia	Baja	Normal	Fuerte	No
LLuvia	Media	Alta	Fuerte	No

- Como podemos ver que todos los registros pertenecen a la clase "No", sabemos que será un nodo hoja con la etiqueta "No"



Árboles de decisión

- El anterior era un caso claro de clasificación
- Variable dependiente cualitativa
- Comencemos por los árboles de clasificación

Algoritmo general

Sea D_t el conjunto Tr-Set en un nodo t

- Si D_t contiene registros que pertenecen todos a la misma clase y_t, luego t es un nodo hoja rotulado como y_t
- Si D_t es un conjunto vacío, luego t es un nodo hoja rotulado por la clase default, y_d
- Si D_t contiene registros que pertenecen a más de una clase, usar un **test de atributo para separar los datos** en subconjuntos más pequeños.
- Recursivamente aplicar el procedimiento a cada subconjunto

Dos preguntas:

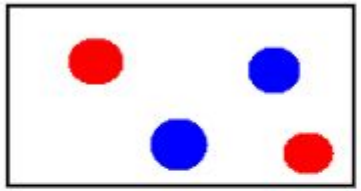
- ¿Cómo separar un nodo? => “splitting”
- ¿Cuándo dejar de hacer crecer un árbol? => “stopping”

Splitting - problemas

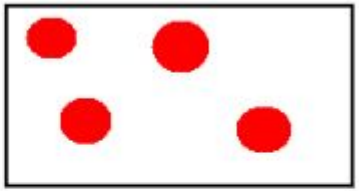
- Splits binarios
- Splits pluricotómicos
- Variables cuantitativas: ¿discretizar o no?
- Variables cualitativas: ¿dicotomizar o no?
- ¿Cómo elegir entre particiones candidatas?
  - Medidas de impureza del nodo

Funciones de pérdida

- Queremos hacer una clasificación según el color de las figuras



Impureza Máxima



Impureza Mínima

Funciones de pérdida

- Definamos p(i|t) como la probabilidad de la clase i en el nodo t (por ejemplo, la fracción de registros con la etiqueta i en el nodo t)

- Para un problema de clasificación binaria (0/1), la distribución máxima de impureza, donde ambas clases están presentes de igual manera, viene dada por la distribución:

$$p(0|t) = p(1|t) = 0.5$$

- La mínima de impureza (o máxima pureza) se obtiene cuando está presente sólo una clase, es decir:

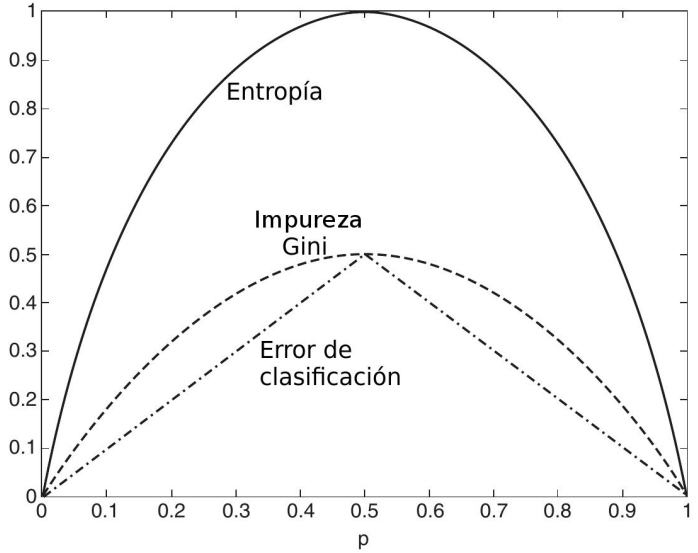
$$p(0|t) = 1 - p(1|t) = 0 \vee 1$$

Funciones de pérdida más comunes

$$\text{Entropía}(t) = -\sum_{i=0}^{c-1} p(i|t) log_2 p(i|t)$$

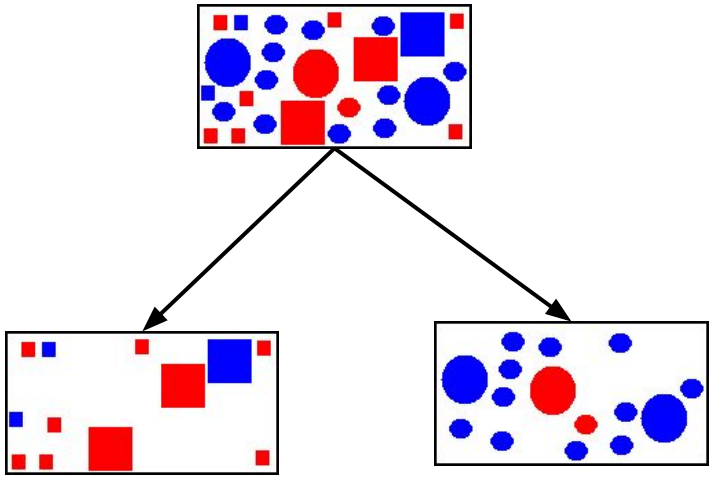
$$\text{Impureza Gini}(t) = 1 - \sum_{i=0}^{c-1} [p(i|t)]^2$$

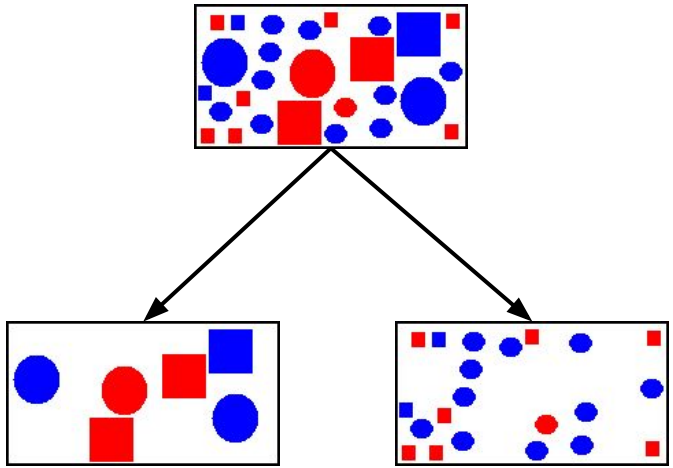
$$\text{Error de clasificación}(t) = 1 - \max_i [p(i|t)]$$



Funciones de pérdida

Siguiendo con el ejemplo de la clasificación según el color de las figuras. Si podemos seleccionar la forma y el tamaño, ¿que partición conviene realizar?





Ganancia (o pérdida) de impureza

- Las medidas de impureza, pero por sí solas, no son suficientes para decirnos cómo funcionará la división. Todavía tenemos que mirar la impureza antes y después de la división. Podemos hacer esta comparación usando la ganancia:

$$\Delta = I(\text{padre}) - \sum_{j \in \text{hijos}} \frac{N_j}{N} I(\text{hijo}_j)$$

- Donde I es la medida de impureza, Nj es el número de registros en el nodo hijo j y N es el número de registros en el nodo padre.
- Cuando I es la entropía, esta cantidad se llama ganancia de información.

Ganancia (o pérdida) de impureza

- En general, una condición de prueba con un alto número de resultados puede conducir a sobreajuste (por ejemplo: una división con un solo resultado por registro).
    - Restringir el algoritmo a únicamente divisiones binarias (CART).
    - Utilizar un criterio de división que penalice explícitamente el número de resultados (C4.5)

Un último ejemplo...

PARTICION 1

P (Nodo padre)	
C1	6
C2	6
n	12
GINI	0,5

K1 (hijo 1)	
C1	5
C2	2
n_{k1}	7
$P(C1\|n_{k1})^2$	0,51
$P(C2\|n_{k1})^2$	0,08
SUMA	0,59
$GINI_{k1}$	0,41
$n_{k1}/n*GINI_{k1}$	0,24

K2 (hijo 2)	
C1	1
C2	4
n_{k2}	5
$P(C1\|n_{k2})^2$	0,04
$P(C2\|n_{k2})^2$	0,64
SUMA	0,68
$GINI_{k2}$	0,32
$n_{k2}/n*GINI_{k2}$	0,13333333

$GINI_{part1}$	0,371

PARTICION 2

P (Nodo padre)	
C1	6
C2	6
n	12
GINI	0,5

K1 (hijo 1)	
C1	7
C2	1
n_{k1}	8
$P(C1\|n_{k2})^2$	0,77
$P(C2\|n_{k2})^2$	0,02
SUMA	0,78
$GINI_{k1}$	0,22
$n_{k1}/n*GINI_{k1}$	0,15

K2 (hijo 2)	
C1	1
C2	3
n_{k2}	4
$P(C1\|n_{k2})^2$	0,0625
$P(C2\|n_{k2})^2$	0,5625
SUMA	0,625
$GINI_{k2}$	0,375
$n_{k2}/n*GINI_{k2}$	0,125

$GINI_{part2}$	0,271

Stopping - overfitting

- Podemos detenernos cuando todos los registros pertenecen a la misma clase, o cuando todos los registros tienen los mismos atributos. Esto es correcto en principio, pero probablemente conduciría a un sobreajuste.

- Prepoda: establecer un umbral mínimo en la ganancia, y detenerse cuando ninguna división logra una ganancia por encima de este umbral. Esto evita el sobreajuste, pero es difícil de calibrar en la práctica.

Stopping - overfitting

- Post-poda: construir el árbol completo, y luego realizar una poda

- Para podar un árbol, examinamos los nodos desde abajo hacia arriba y simplificamos las ramas del árbol (de acuerdo con algunos criterios).

- Los subárboles complicados pueden ser reemplazados con un solo nodo o con un subárbol más simple (secundario).

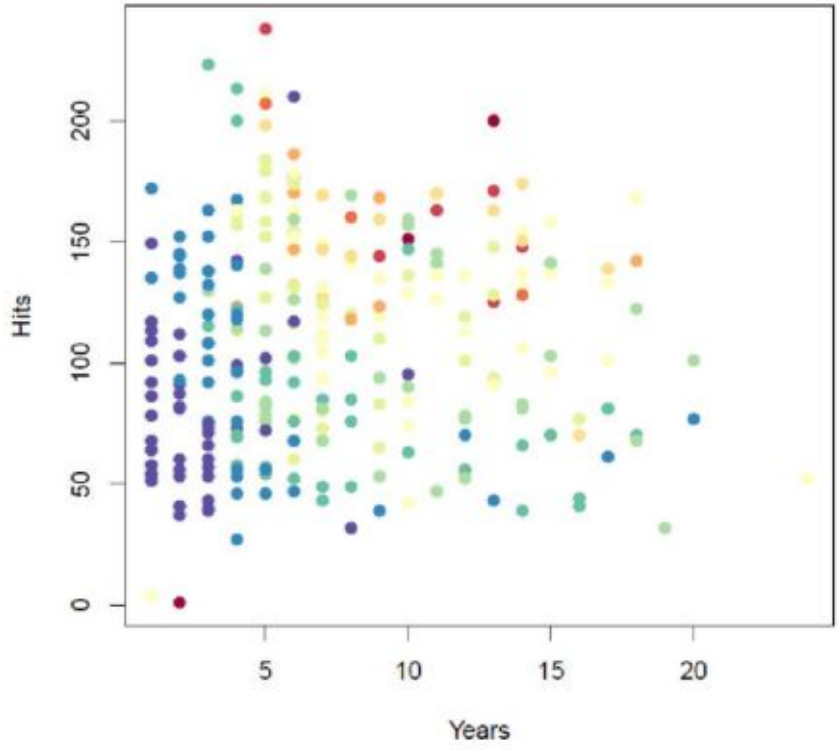
Árboles de regresión

- Variable de resultado es cuantitativa
- La varianza de dicho valor en un nodo nos dá una medida de impureza de dicho valor.
  - Error Cuadrático Medio como medida de impureza y la función a maximizar seguirá siendo la ganancia.

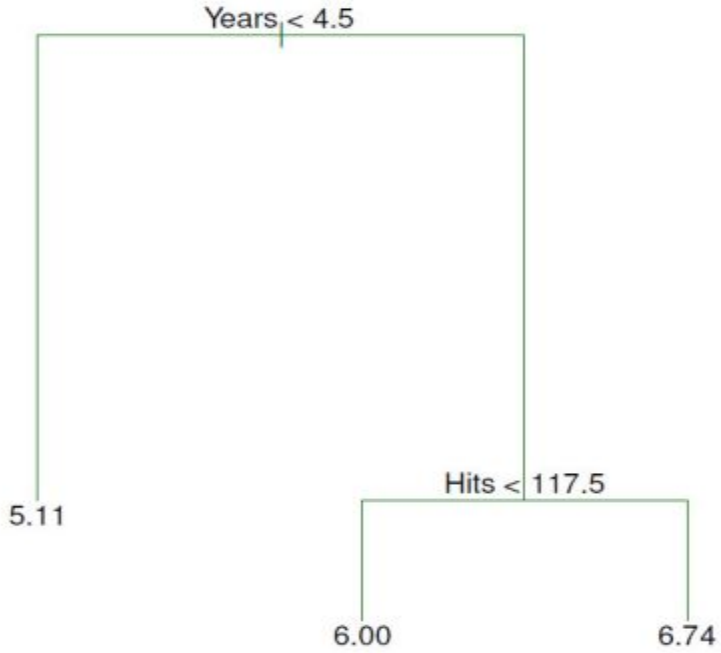
$$\Delta = ECM(\text{padre}) - \sum_{j \in \text{hijos}} \frac{N_j}{N} ECM(\text{hijo}_j)$$

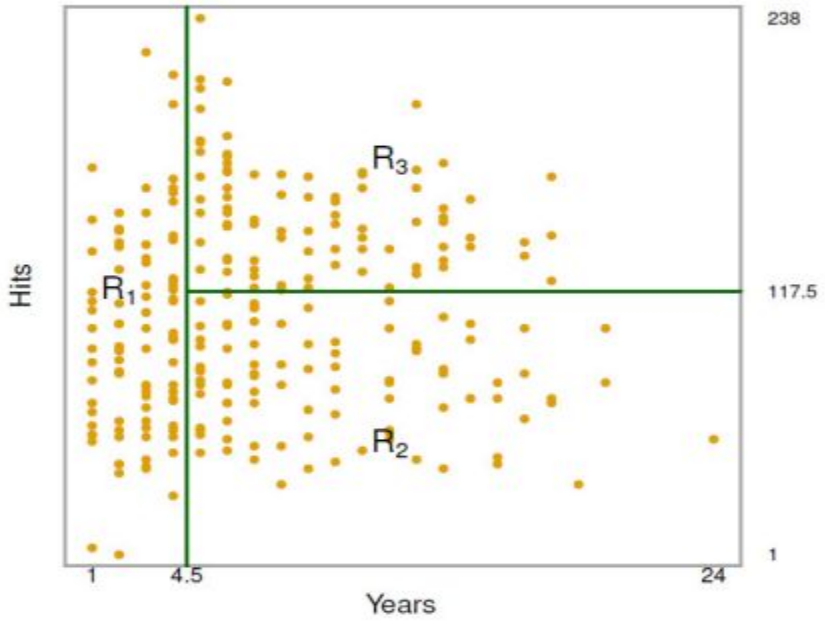
- Objetivo: máxima ganancia, donde ECM es el Error Cuadrático Medio, Nj es el número de registros en el nodo hijo j y N es el número de registros en el nodo padre.

Árboles de regresión



Árboles de regresión





¿Cómo construimos las regiones?

- En teoría las regiones R1,...,RJ podrían tener cualquier forma.
- Elegimos dividir el espacio de predictores en rectángulos o cajas en varias dimensiones por simplicidad y facilidad de interpretación del modelo predictivo resultante. El objetivo es encontrar cajas R1,...,RJ que minimizan la suma de residuos al cuadrado (RSS) dada por

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2,$$

¿Cómo construimos las regiones?

- Problema: no es factible computacionalmente considerar todas las posibles particiones del espacio de atributos en J cajas.
- Enfoque de arriba hacia abajo “greedy” que es conocido como recursive binary splitting.
- **Recursive binary splitting** comienza en la parte de arriba del árbol (donde todas las observaciones pertenecen a una sola región) y sucesivamente particiona el espacio de predictores.
- **Greedy** porque en cada paso de la construcción del árbol se busca la mejor división en ese punto en particular en lugar de mirar hacia adelante y elegir una división que llevaría a un mejor árbol en un paso futuro.

Síntesis

- CART's: modelos interpretables y simples
- Inestables (cambian mucho si el dataset cambia un poco)
- Criterios de "splitting"
- Criterios de "stopping"
- Medidas de impureza
- Funciones de optimización