

**Fundamentos de la Programación Estadística y
Data Mining en R**

Unidad 4 Árboles de Decisión y Random Forests

Dr. Germán Rosati (Digital House/MTEySS/UNSAM)

Métodos de árboles de decisión



Métodos de árboles de decisión

- Se proponen dividir o estratificar el espacio de predictores en un determinado número de regiones simples.
- Como las diferentes “reglas de partición” que se usan para dividir el espacio de los predictores pueden ser representadas en un árbol, este tipo de técnicas se conocen con el nombre genérico de “métodos basados en árboles de decisión”.

Métodos de árboles de decisión

- Son fáciles de construir y de interpretar
- Generalmente, tienen por sí solos menor capacidad predictiva que otros métodos de AS
- En la próxima unidad vamos a discutir algunos métodos para incrementar la capacidad predictiva de los métodos basados en árboles (random forest, bagging, etc.)
- Los métodos de árboles pueden usarse tanto en problemas de regresión como de clasificación.

Métodos de árboles de decisión

- 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

Métodos de árboles de decisión

- 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.



Métodos de árboles de decisión

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



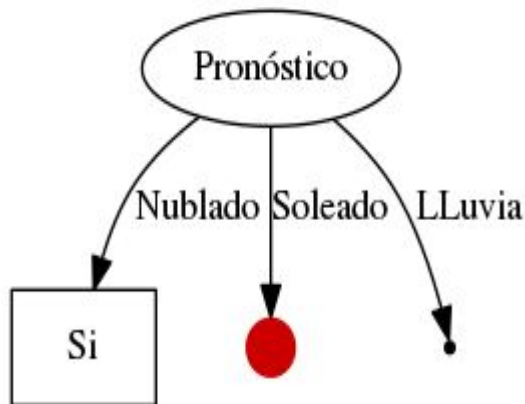
Pronóstico	Temperatura	Humedad	Viento	Jugar
Nublado	Alta	Alta	Débil	Si
Nublado	Baja	Normal	Fuerte	Si
Nublado	Media	Alta	Fuerte	Si
Nublado	Alta	Normal	Débil	Si

- 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".



Métodos de árboles de decisión

- Ahora analicemos la partición correspondiente a Pronóstico=Soleado.

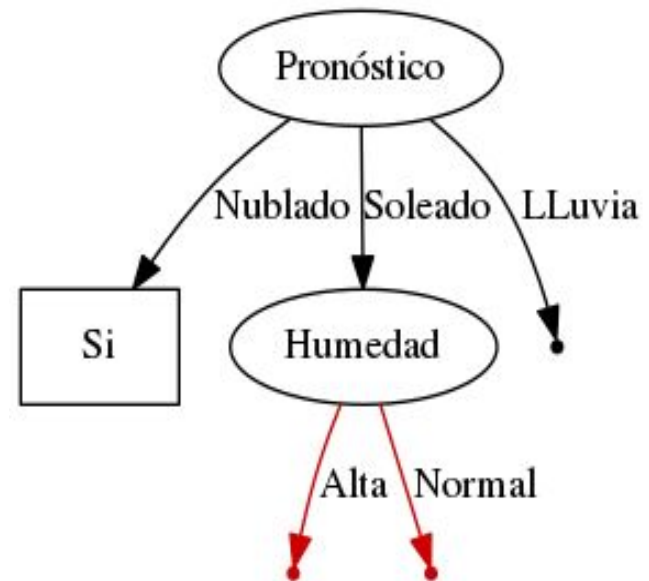
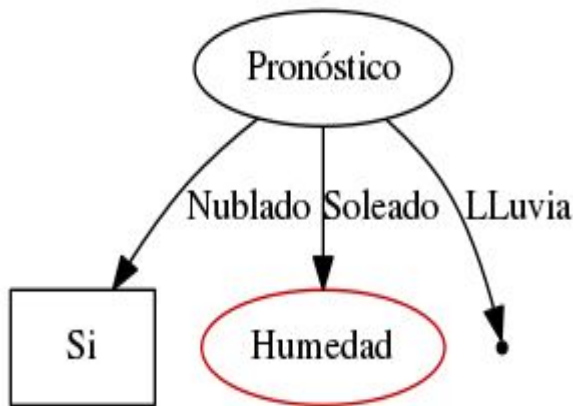


Pronóstico	Temperatura	Humedad	Viento	Jugar
Soleado	Alta	Alta	Débil	No
Soleado	Alta	Alta	Fuerte	No
Soleado	Media	Alta	Débil	No
Soleado	Baja	Normal	Débil	Si
Soleado	Media	Normal	Fuerte	Si

- 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?

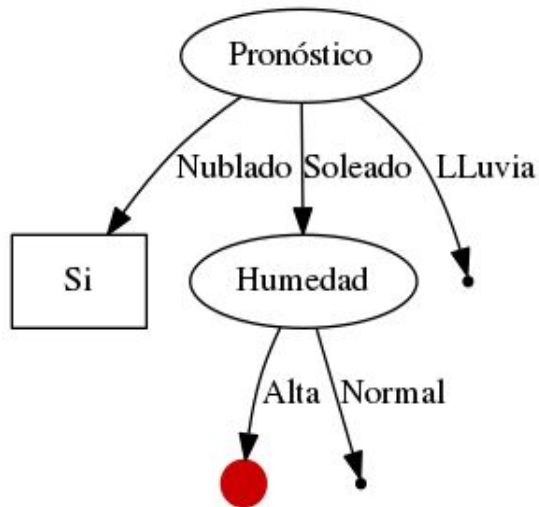
Métodos de árboles de decisión

- 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.



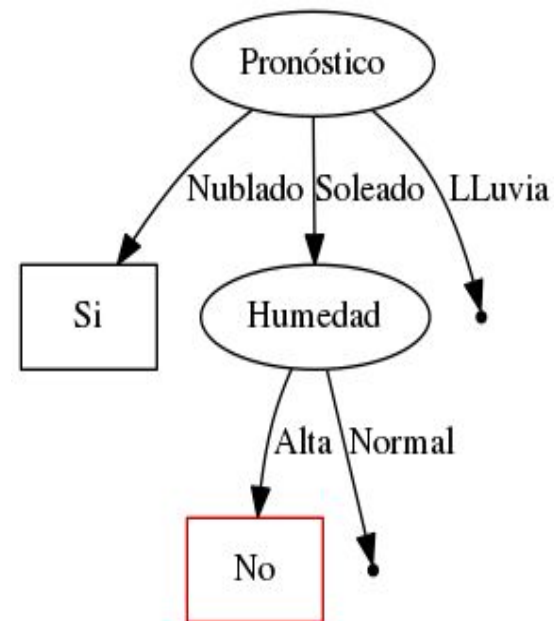
Métodos de árboles de decisión

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



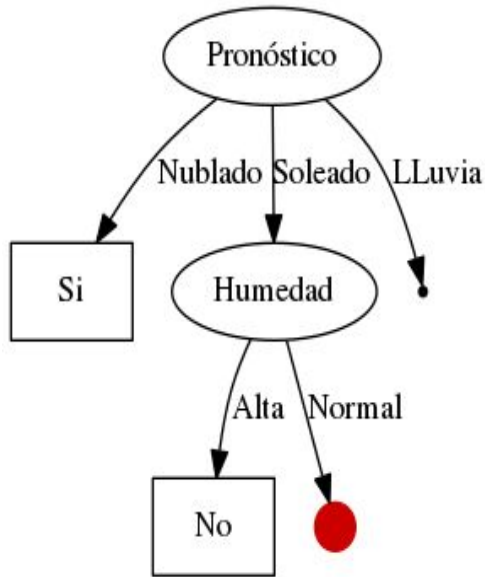
Pronóstico	Temperatura	Humedad	Viento	Jugar
Soleado	Alta	Alta	Débil	No
Soleado	Alta	Alta	Fuerte	No
Soleado	Media	Alta	Débil	No

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



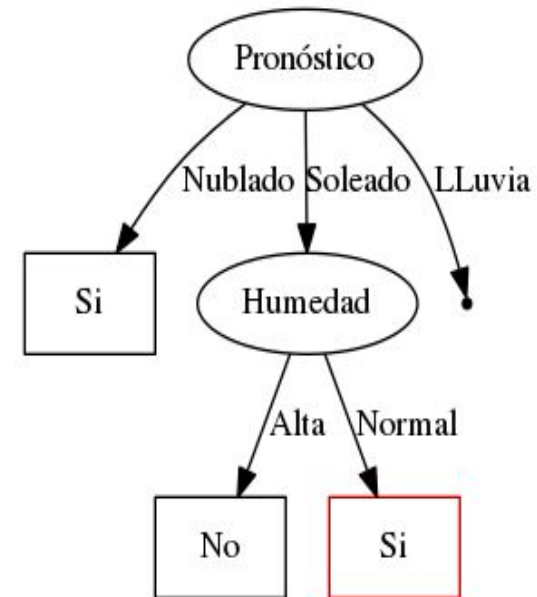
Métodos de árboles de decisión

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



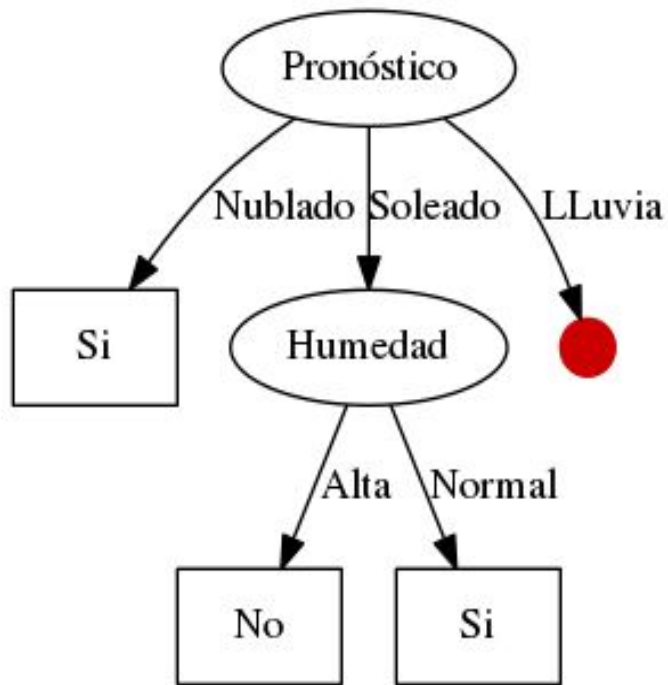
Pronóstico	Temperatura	Humedad	Viento	Jugar
Soleado	Baja	Normal	Débil	Si
Soleado	Media	Normal	Fuerte	Si

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



Métodos de árboles de decisión

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

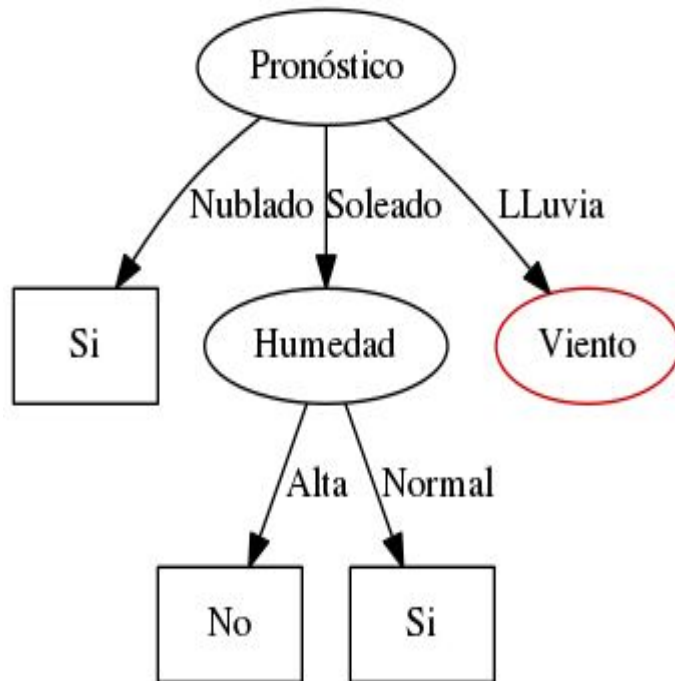


Pronóstico	Temperatura	Humedad	Viento	Jugar
LLuvia	Media	Alta	Débil	Si
LLuvia	Baja	Normal	Débil	Si
LLuvia	Baja	Normal	Fuerte	No
LLuvia	Media	Normal	Débil	Si
LLuvia	Media	Alta	Fuerte	No

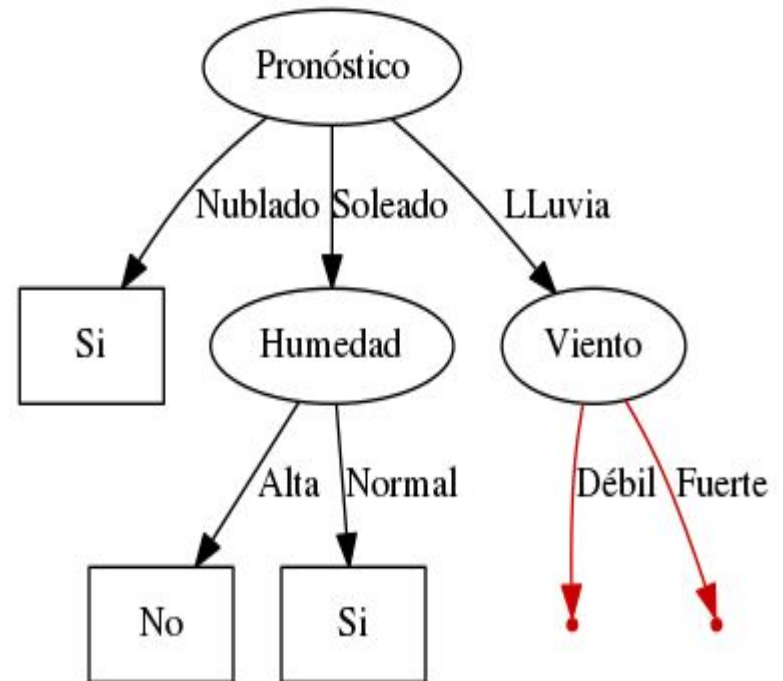
- Nuevamente vemos que los registros pertenecen a clases distintas y tendremos que sub-particionar. Podemos optar por particionar según las siguientes variables: {Temperatura, Humedad y Viento}.
- ¿Cual conviene utilizar?

Métodos de árboles de decisión

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

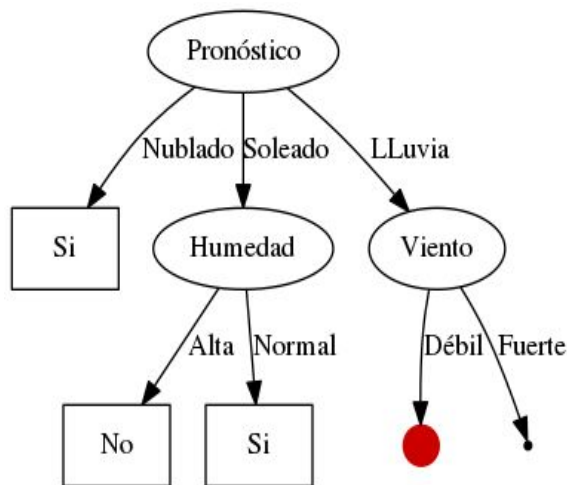


- Ahora que tenemos seleccionado el criterio, creamos las particiones.



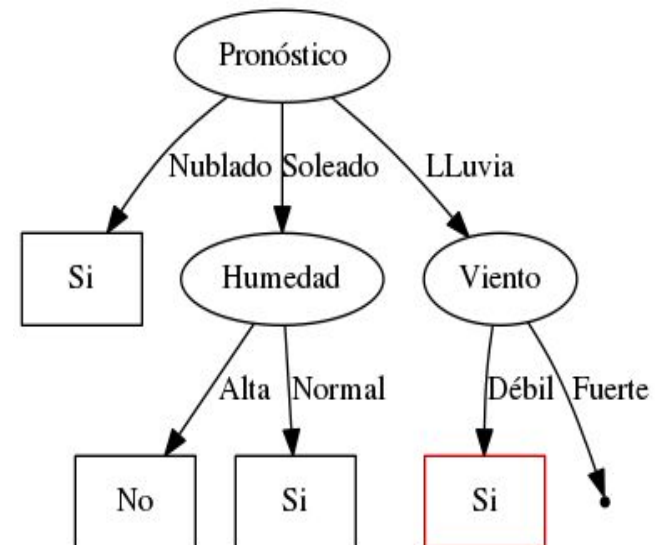
Métodos de árboles de decisión

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



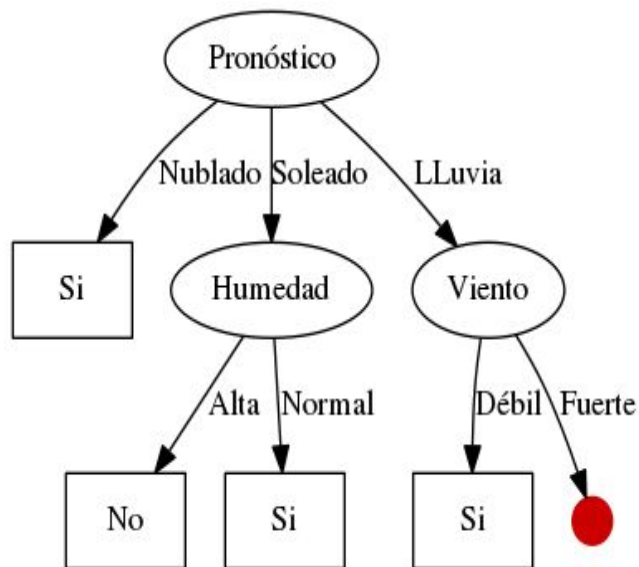
Pronóstico	Temperatura	Humedad	Viento	Jugar
LLuvia	Media	Alta	Débil	Si
LLuvia	Baja	Normal	Débil	Si
LLuvia	Media	Normal	Débil	Si

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



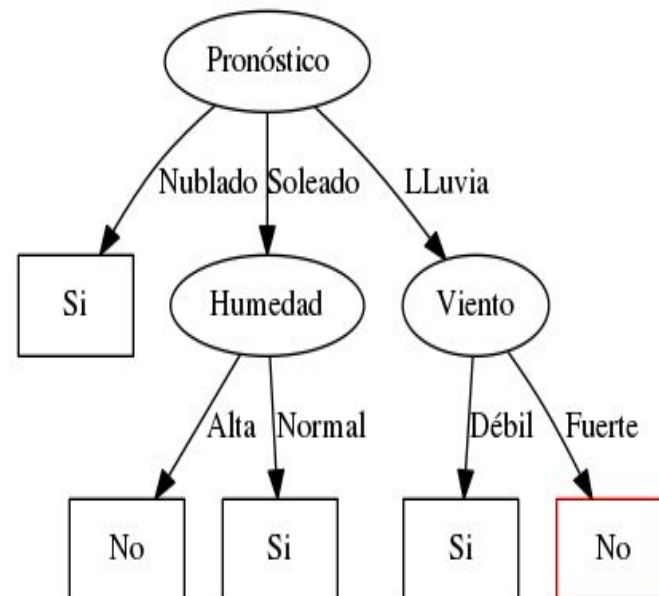
Métodos de árboles de decisión

- 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"



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

Algoritmo general

- Dos cuestiones surgen al momento de construir un árbol:
 - ¿Cuál es el criterio para separar los nodos? (“splitting criteria”)
 - ¿Cuándo dejar de hacer crecer al árbol? (“stopping criteria”)

Splitting en árboles de clasificación

- Dos problemas asociados;
 - ¿Cómo especificar la condición del atributo?
 - Splits binarios (dos valores)
 - Splits multivía (múltiples valores)
 - En variables cualitativas: Decidir si se dicotomizan las variables o si se usa con la cantidad de categorías “original”.
 - En variables cuantitativas:
 - Discretizar (Split multivía), por ejemplo, cuantiles
 - Partición binaria (Split binario): X es mayor a 10000 (Si-No)
 - ¿Cómo determinar el mejor separador (split)?
 - Se necesitan medidas de “impureza” del nodo

Splitting criteria (impureza del nodo)

- Se prefieren nodos terminales homogéneos (en términos de la distribución de la variable dependiente)

Cat.	f
C1:	5
C2:	5

Baja homogeneidad

Cat.	f
C1:	9
C2:	1

Alta homogeneidad

Cat.	f
C1:	9
C2:	0

Homogeneidad perfecta

Splitting en árboles de clasificación

- Medidas de homogeneidad (algunas)
 - Entropía
 - Índice de Gini
 - Error de clasificación

Clasificación - Impureza: Entropía

- Dos tipos de problemas
 - Aprendizaje no supervisado: no hay variable de «respuesta» o «dependiente»
 - Aprendizaje supervisado:
 - Una variable resultado Y (variable dependiente, respuesta, target)
 - Un vector de p variables independientes X (inputs, predictores, etc.)
 - En problemas de regresión Y es cuantitativa
 - En problemas de clasificación Y es cualitativa

Clasificación - Impureza: Error clasific.

- El error de clasificación de un nodo t se computa como:
 - $\text{Error}(t) = 1 - \max p(i|t)$
 - $p(j|t)$ es la frecuencia relativa de la clase p en el nodo t
- Mide los casos mal clasificados de un nodo.
 - Máximo valor: cuando los registros están igualmente distribuidos
 - Mínimo (0): cuando todos los registros pertenecen a una misma clase (máxima información).

Clasificación - Impureza: Gini

- Dado un nodo t
 - $G(t) = 1 - \sum_j [p(j|t)]^2$
 - Donde $p(j|t)$ es la frecuencia relativa de la categoría j en el nodo t
 - Máximo ($1 - 1/nc$) cuando los registros están uniformemente distribuidos entre todas las clases, implicando mínima información interesante
 - Mínimo (0) cuando todos los registros pertenecen a una clase, implicando la información más interesante

Clasificación - Impureza: Gini

C1	0
C2	6

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$\text{Gini} = 1 - P(C1)^2 - P(C2)^2 = 1 - 0 - 1 = 0$$

C1	1
C2	5

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$\text{Gini} = 1 - (1/6)^2 - (5/6)^2 = 0.278$$

C1	2
C2	4

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

$$\text{Gini} = 1 - (2/6)^2 - (4/6)^2 = 0.444$$


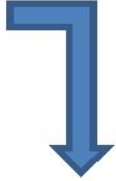
Splitting basado en Gini



- Se usa en algoritmos CART, SLIQ, SPRING (nosotros nos vamos a centrar más adelante en CART)
- Cuando un nodo p es dividido en k particiones (nodos hijos) la calidad del split se computa como

$$GINI_{split} = \sum_{i=1}^k \frac{n_i}{n} GINI(i)$$

- donde
 - n_i =cantidad de registros en nodo i
 - n =cantidad de registros en nodo p

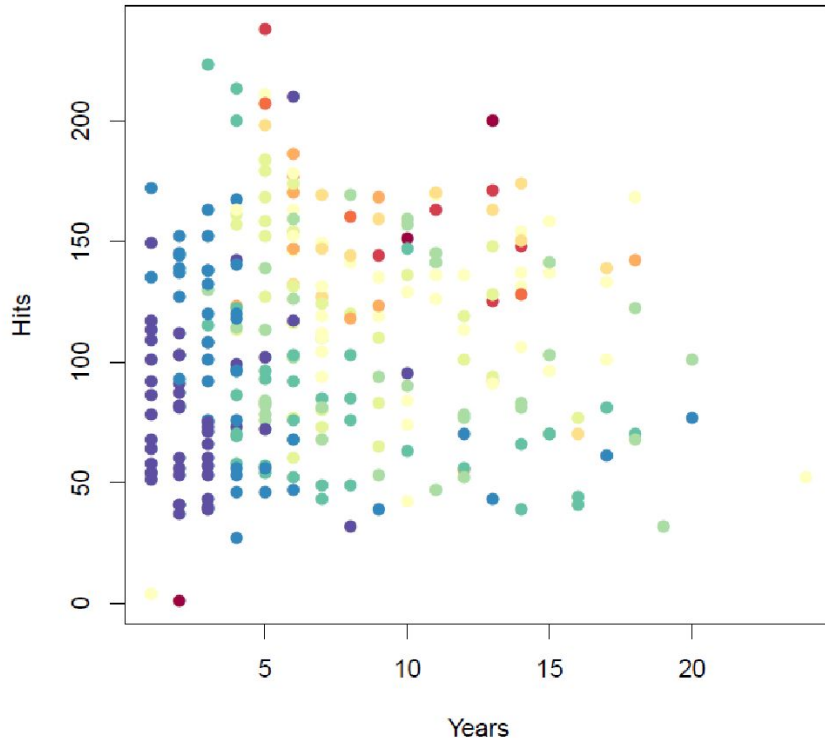
Cálculo splitting basado en Gini

PARTICION 1																							
	<table><tr><th colspan="2">P (Nodo padre)</th></tr><tr><td>C1</td><td>6</td></tr><tr><td>C2</td><td>6</td></tr><tr><td>n</td><td>12</td></tr><tr><td colspan="2">GINI</td><td>0,5</td></tr></table>	P (Nodo padre)		C1	6	C2	6	n	12	GINI		0,5											
	P (Nodo padre)																						
	C1	6																					
	C2	6																					
	n	12																					
GINI		0,5																					
<table><tr><th colspan="2">K1 (hijo 1)</th></tr><tr><td>C1</td><td>5</td></tr><tr><td>C2</td><td>2</td></tr><tr><td>n_{k1}</td><td>7</td></tr><tr><td colspan="2">P(C1 n_{k1})²</td><td>0,51</td></tr><tr><td colspan="2">P(C2 n_{k1})²</td><td>0,08</td></tr><tr><td colspan="2">SUMA</td><td>0,59</td></tr><tr><td colspan="2">GINI_{k1}</td><td>0,41</td></tr><tr><td colspan="2">n_{k1}/n * GINI_{k1}</td><td>0,24</td></tr></table>	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
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																					
<table><tr><th colspan="2">K2 (hijo 2)</th></tr><tr><td>C1</td><td>1</td></tr><tr><td>C2</td><td>4</td></tr><tr><td>n_{k2}</td><td>5</td></tr><tr><td colspan="2">P(C1 n_{k2})²</td><td>0,04</td></tr><tr><td colspan="2">P(C2 n_{k2})²</td><td>0,64</td></tr><tr><td colspan="2">SUMA</td><td>0,68</td></tr><tr><td colspan="2">GINI_{k2}</td><td>0,32</td></tr><tr><td colspan="2">n_{k2}/n * GINI_{k2}</td><td>0,13333333</td></tr></table>	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
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																							
	<table><tr><th colspan="2">P (Nodo padre)</th></tr><tr><td>C1</td><td>6</td></tr><tr><td>C2</td><td>6</td></tr><tr><td>n</td><td>12</td></tr><tr><td colspan="2">GINI</td><td>0,5</td></tr></table>	P (Nodo padre)		C1	6	C2	6	n	12	GINI		0,5											
	P (Nodo padre)																						
	C1	6																					
	C2	6																					
	n	12																					
GINI		0,5																					
<table><tr><th colspan="2">K1 (hijo 1)</th></tr><tr><td>C1</td><td>7</td></tr><tr><td>C2</td><td>1</td></tr><tr><td>n_{k1}</td><td>8</td></tr><tr><td colspan="2">P(C1 n_{k2})²</td><td>0,77</td></tr><tr><td colspan="2">P(C2 n_{k2})²</td><td>0,02</td></tr><tr><td colspan="2">SUMA</td><td>0,78</td></tr><tr><td colspan="2">GINI_{k1}</td><td>0,22</td></tr><tr><td colspan="2">n_{k1}/n * GINI_{k1}</td><td>0,15</td></tr></table>	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
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																					
<table><tr><th colspan="2">K2 (hijo 2)</th></tr><tr><td>C1</td><td>1</td></tr><tr><td>C2</td><td>3</td></tr><tr><td>n_{k2}</td><td>4</td></tr><tr><td colspan="2">P(C1 n_{k2})²</td><td>0,0625</td></tr><tr><td colspan="2">P(C2 n_{k2})²</td><td>0,5625</td></tr><tr><td colspan="2">SUMA</td><td>0,625</td></tr><tr><td colspan="2">GINI_{k2}</td><td>0,375</td></tr><tr><td colspan="2">n_{k2}/n * GINI_{k2}</td><td>0,125</td></tr></table>	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
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																					

Árboles de regresión

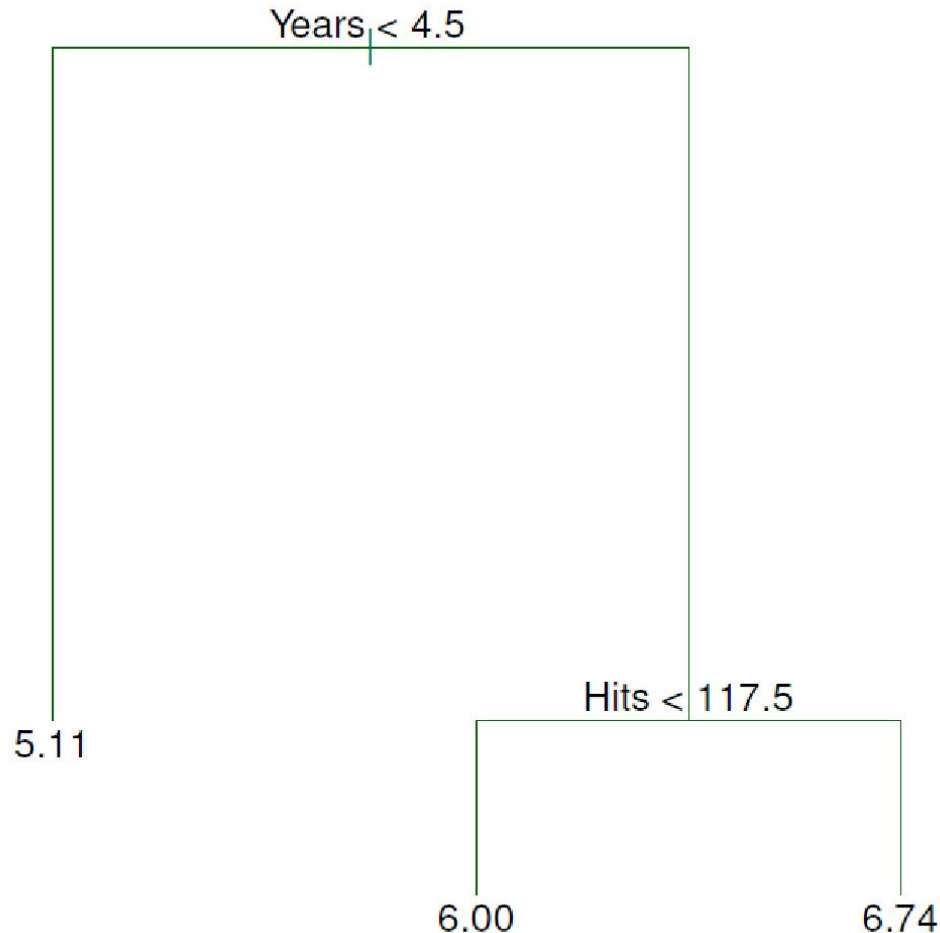
- Base de datos en R «Hitters»



- Salarios codificados (bajos: azul-verde; altos: amarillo rojo).
- La idea es predecir el salario a través de la antigüedad y el rendimiento del jugador (medido en “Hits”).

Árboles de regresión

- Árbol de decisión



CART: Árboles de regresión

- Si la variable dependiente es cuantitativa, entonces, no hay clases de respuesta. Se busca dividir el espacio de predictores (el espacio de todos los valores posibles de X_1, X_2, \dots, X_J) en J regiones diferentes y no «solapadas» (R_1, R_2, \dots, R_J)
- La predicción final de cada región es la media de los valores de la variable dependiente en la región J .
- Entonces, se busca encontrar regiones que logren minimizar la suma de errores cuadrados medios a lo largo de todas las «regiones» o «nodos» del árbol.

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

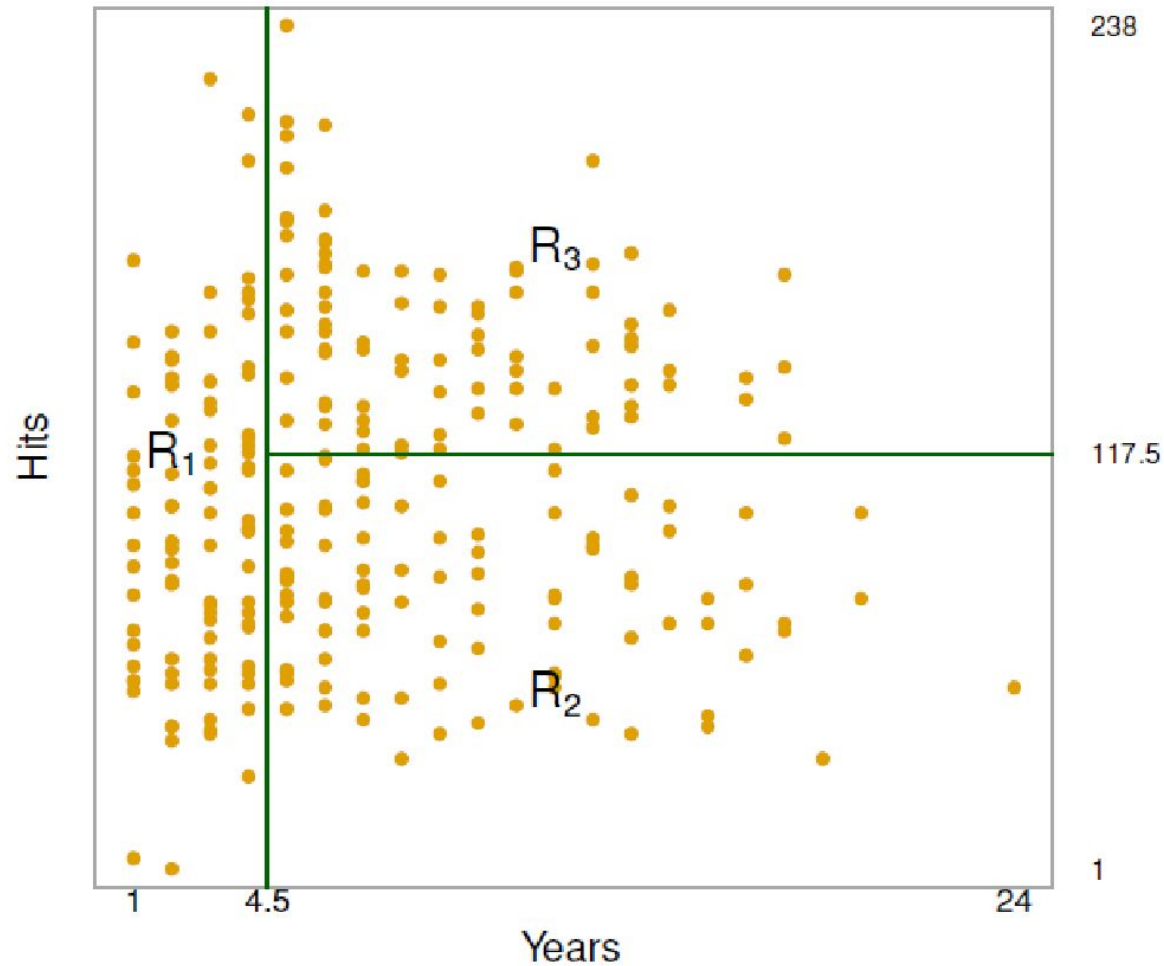
Árboles de regresión

- Para un determinado nodo, la etiqueta indica que la rama “izquierda” que sale de esa división agrupa a los casos cuyos valores son $X_j < t_k$; en cambio, la rama derecha corresponde a $X_j > t_k$. Por ejemplo, la primera división del árbol divide los registros en dos grandes ramas: la izquierda corresponde a los que han estado en la liga por menos de 4,5 años; La derecha a los que han estado 4,5 años o más.
- El árbol tiene dos nodos internos y tres nodos terminales (también llamados “hojas”). El número en cada hoja es la media de los registros que caen allí.

CART: Árboles de regresión

- El árbol segmenta a los jugadores en tres regions del espacio de predicción:
 - $R_1 = \{X | Years < 4.5\};$
 - $R_2 = \{X | Years \geq 4.5, Hits < 117.5\}$
 - $R_3 = \{X | Years \geq 4.5, Hits \geq 117.5\}$

CART: Árboles de regresión



Árboles de regresión

- **YEARS** (años) es la variable más importante en la determinación del salario => jugadores con menos experiencia ganan menos salarios.
- Condicionado a que tenga menos experiencia, el número de **HITS**, que un jugador hizo en el año previo parece jugar un rol menor en el salario.
- Pero entre los jugadores con más de 4 años de experiencia el número de **HITS** afecta al salario en sentido positivo.
- Comparado con un modelo de regresión en el que lo que se ve son coeficientes... la cosa es más clara.

Algoritmo CART (introducción)

- Se particiona el espacio en dos regiones y se modela la respuesta (Y) como la media (para árboles de regresión) o la moda de la región (para árboles de clasificación).
- El algoritmo recorre cada uno de los predictores y cada uno de los valores buscando la partición más eficiente en función de algún criterio de error.
- Selecciona la variable y el punto de corte que logra la mejor *performance* (el menor error).
- A continuación, cada una de estas regiones se particiona nuevamente en dos regiones más y este proceso se repite hasta que se satisfaga algún criterio de corte, definido previamente.

Algoritmo CART (introducción)

- En general, es computacionalmente imposible considerar todas las particiones posibles en J cajas.
- Por eso, en CART se usa un enfoque «top-down» y «greedy» llamado «recursive binary splitting»
- Top-down (arriba-abajo) porque comienza «arriba» el árbol y sucesivamente divide al espacio de predictores; cada split es indicado como dos ramas hacia abajo en el árbol
- Greedy porque en cada paso del proceso de construcción el mejor split se evalúa solamente en esa etapa y no para mejorar etapas posteriores.

Más detalles de CART

- Generalmente consta de tres partes
 - Construir el «máximo árbol»
 - Elegir el tamaño correcto del árbol
 - Clasificar el Te-S usando el árbol construido
- En un nodo t CART va a buscar a lo largo de todos los valores posibles de todas las variables en la matriz X para encontrar el mejor split (binario)

Más detalles de CART

- La «máxima homogeneidad» de cada nodo se define a partir de una «función de impureza». Dado que la impureza del nodo «padre» es constante para cada valor de los splits, la máxima homogeneidad de los nodos derecho e izquierdo es equivalente a la maximización del cambio en función de impureza:
 - $\Delta i(t) = i(t_p) - E[i(t_c)]$

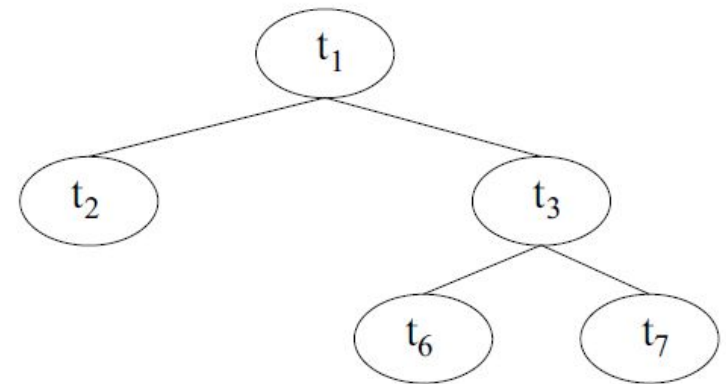
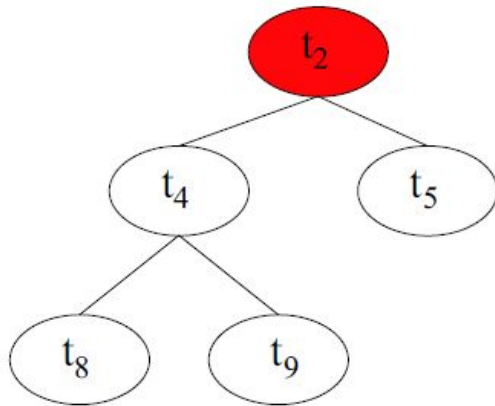
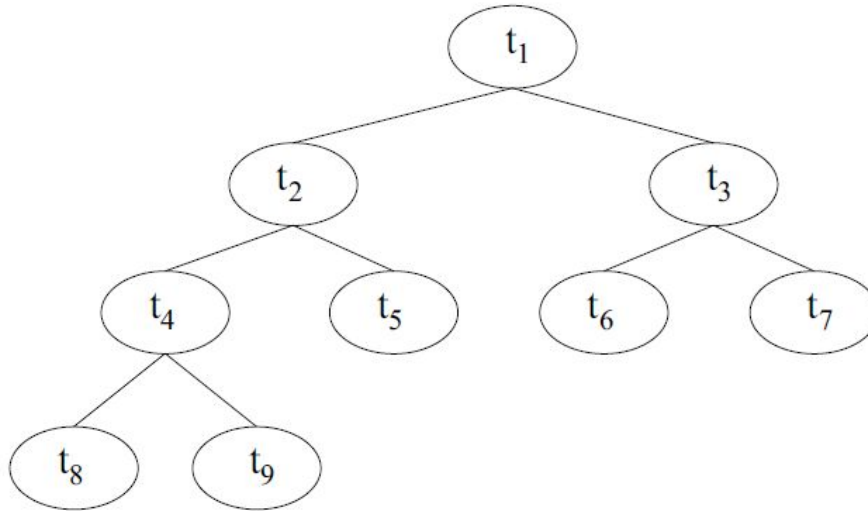
Problemas: overfitting

- Todos los procesos y algoritmos antes mencionados pueden llevar al overfitting... ¿Qué implica eso? ¿Por qué? Un árbol más pequeño (es decir, con menos regiones) puede llevar a un modelo con menor variancia y mejor interpretación con el costo de un pequeño sesgo.
- Alternativa: desarrollar el árbol mientras que el descenso en el RSS/error de clasificación debido a cada split sea mayor a cierto umbral. Esto genera árboles más chicos. Pero puede pasar que un Split “malo” venga antes de uno muy bueno (mucha reducción de RSS/error de clasificación)

CART: podando (prunning) un árbol

- CART usa una mejor opción: generar un árbol muy grande (T_0) para “podarlo” –achicarlo– luego y obtener un “sub-árbol”.
- El problema es que dado un árbol T_0 existen muchos subárboles binarios. Si $|T|$ denota la cantidad de nodos terminales, entonces, hay aproximadamente $1.5028369^{|T|}$ subárboles.
- En lugar de considerar todos los subárboles, usamos *Cost-complexity prunning* (“podado basado en costo y complejidad”).

CART: podando (prunning) un árbol



CART: podando (prunning) un árbol

- En lugar de considerar todos los subárboles, usamos un subconjunto de árboles. Los «mejores» en el sentido que se define a continuación.
- Método *Cost-complexity prunning* (“podado basado en costo y complejidad”).

CART: Podando (prunning) un árbol

- Consideremos un conjunto grande de árboles indexados por un parámetro de “tuneo” no negativo α . Para cada valor de α hay un subárbol $T \subset T_0$ tal que
 - $\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$
- sea lo más pequeño posible. Acá $|T|$ indica el número de nodos hijos del árbol T ; R_m es el rectángulo (la región) correspondiente a m -ésimo nodo; \hat{y}_{R_m} es la media de las observaciones de entrenamiento de D

CART: Podando (prunning) un árbol

- En un árbol de clasificación la función de costo es $C_\alpha(T) = R(T) + \alpha|T|$ donde $R(T)$ es el error de clasificación
- Cuando la cantidad de hojas terminales se incrementa en una unidad, el costo del árbol $C_\alpha(T)$ se incrementa en α . Si fijamos el valor de α , entonces hay un subárbol $T \subset T_0$ que cumple las siguientes condiciones
 - No hay otro T con menor costo dentro de T_0
 - Si hay «empates» entre varios T elegimos el más pequeño

CART: Podando (prunning) un árbol

- α es un parámetro de tuneo y controla el balance entre la complejidad del sub-árbol y su ajuste a los datos.
- Seleccionamos un valor óptimo de $\hat{\alpha}$ usando cross-validation.
- Luego, volvemos al dataset y obtenemos el sub-árbol correspondiente a $\hat{\alpha}$.

Resumen del algoritmo CART

- Usar splitting recursivo binario para estimar un árbol grande en el Tr-S, parando solo cuándo el n de casos por nodo sea menor a un umbral
- Aplicar cost-complexity pruning para obtener una secuencia de sub-árboles “mejores” como función de $\hat{\alpha}$.
- Usar K-fold cross-validation para seleccionar $\hat{\alpha}$. Para grupo K:
 - Repetir pasos 1 y 2 en la $(K-1)/K$ fracciones del training data, excluyendo la k -ésima parte
 - Evaluar el MSE/impureza en la k -ésima parte dejada fuera como test-set, como función de $\hat{\alpha}$
- Promediar los resultados y elegir el $\hat{\alpha}$ que minimice el CV_MSE
- Elegir el sub-árbol del paso 2 que corresponda al valor

Problemas de los árboles de decisión

- Tienden al sobreajuste de los datos
- En general, bajo poder predictivo
- “Inestables”, es decir, son sensibles a pequeños cambios en los datos. Cambiando un solo caso, se puede obtener un árbol totalmente diferente.
- ¿Qué hacer?...

Bagging de árboles de decisión

- Es posible aplicar el algoritmo de bagging para construir un ensamble de árboles de decisión (Breiman 1990)
- Procedimiento
 - Extraer B muestras con reposición del dataset
 - Para cada una de las muestras entrenar un árbol de clasificación
 - Con cada una de los B árboles contruidos hacer una predicción sobre el test set
 - Agregar por «mayoría de votos» las predicciones para generar la predicción final

Bagging: árboles de decisión

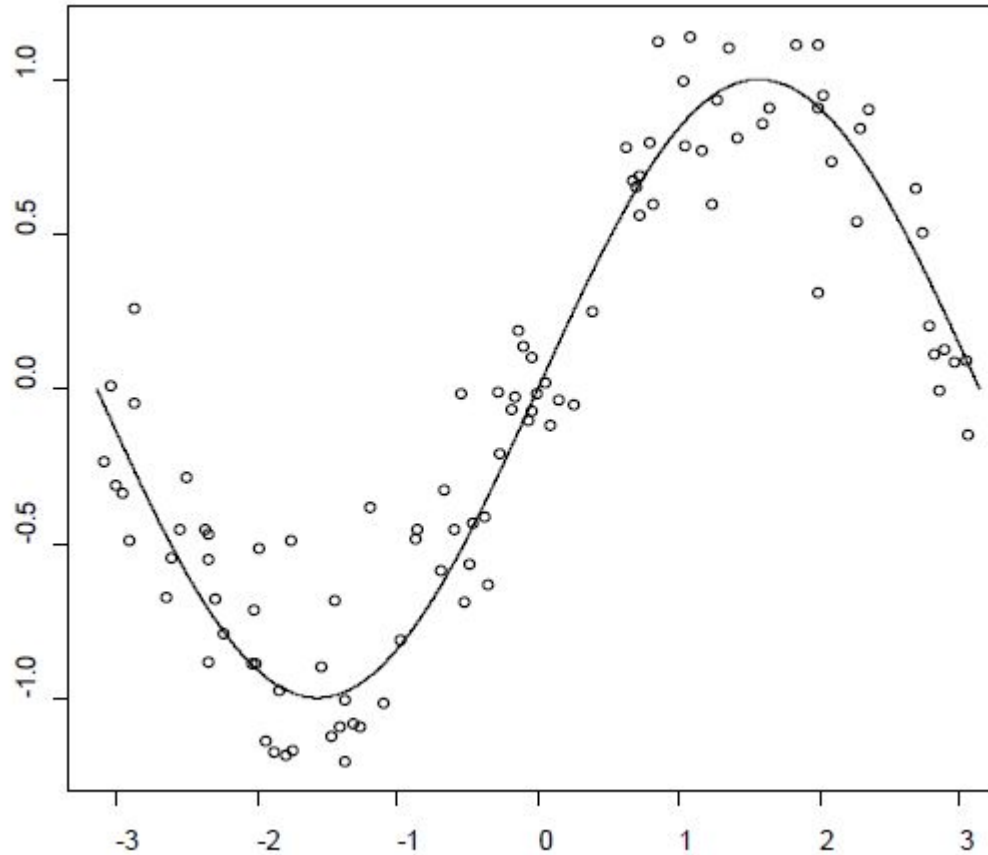
- Dos tipos de problemas
 - Aprendizaje no supervisado: no hay variable de «respuesta» o «dependiente»
 - Aprendizaje supervisado:
 - Una variable resultado Y (variable dependiente, respuesta, target)
 - Un vector de p variables independientes X (inputs, predictores, etc.)
 - En problemas de regresión Y es cuantitativa
 - En problemas de clasificación Y es cualitativa

Bagging: árboles de decisión

- $$Y = f(X) + \epsilon$$
- Objetivo: predecir Y basado en los valores de X , sin observar u y sin saber qué forma asume $f(\cdot)$.
- Éxito: minimizar el “out of sample” error. OJO Econometristas
- Lo que importa es la performance predictiva del modelo

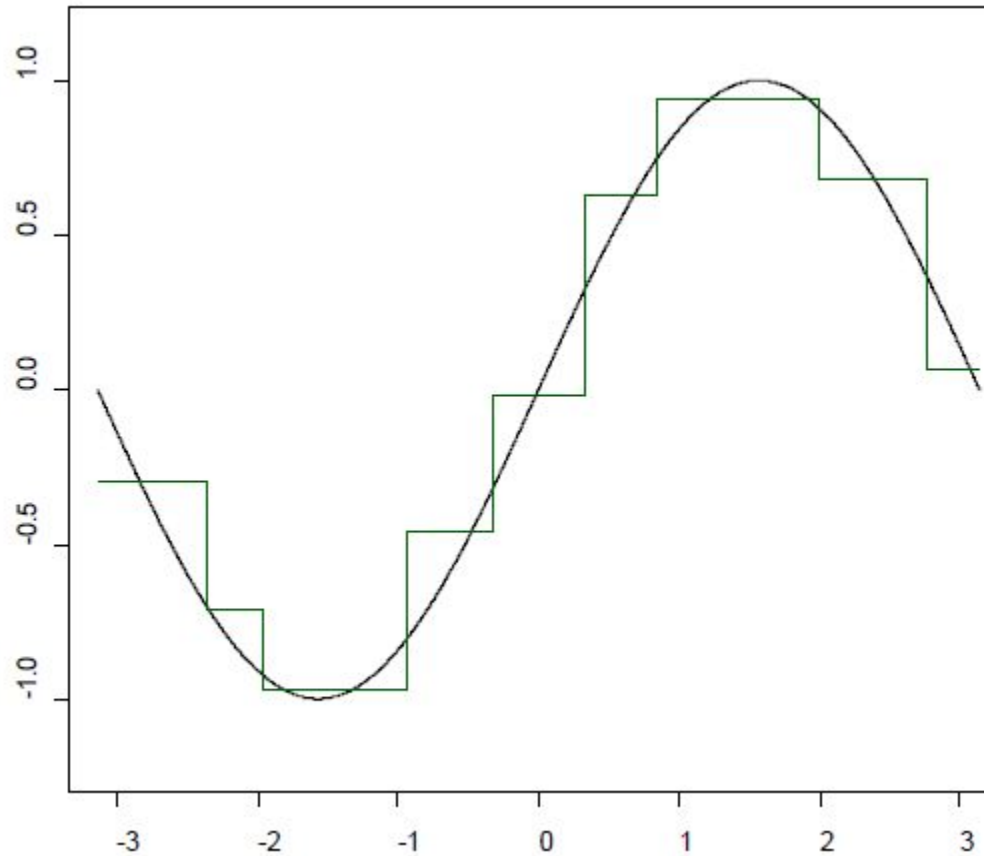
Bagging árboles: ejemplo

- Función generadora



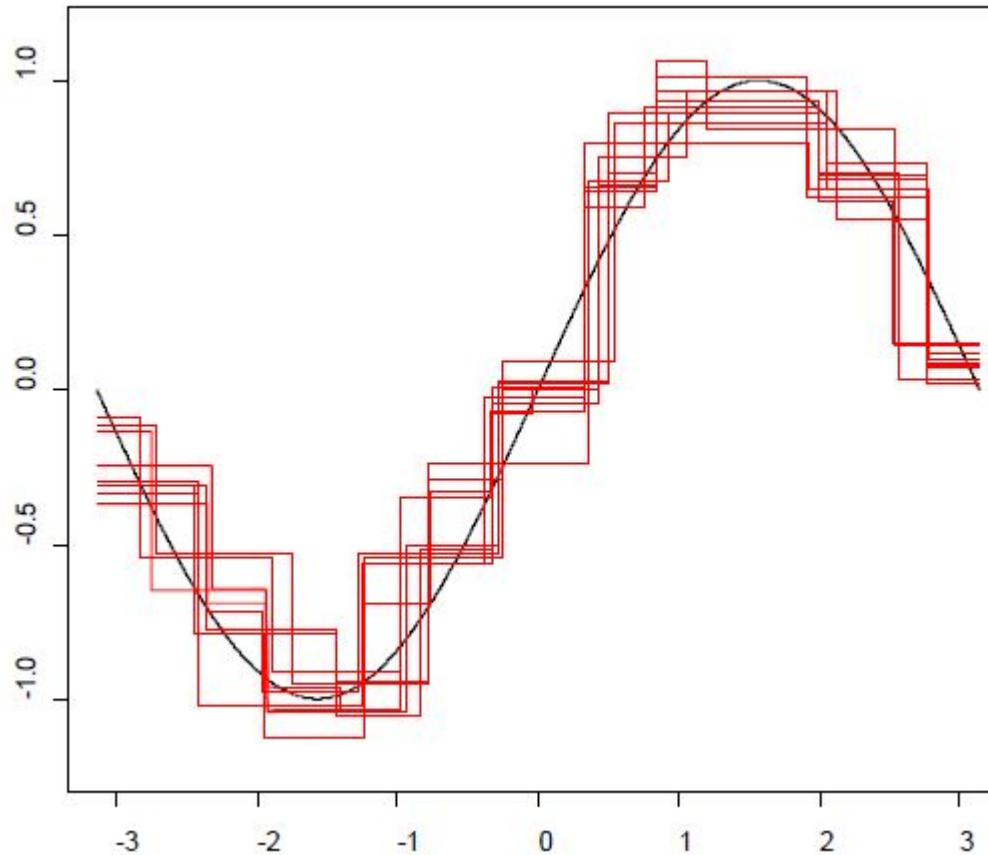
Bagging árboles: ejemplo

- Un solo árbol



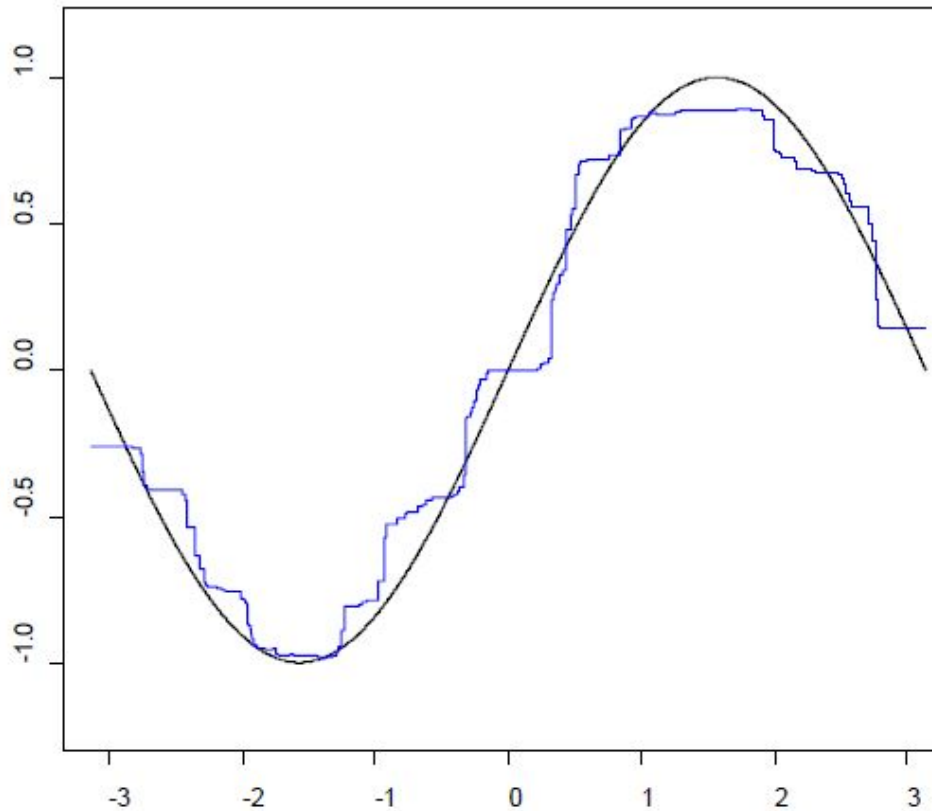
Bagging árboles: ejemplo

- 10 árboles



Bagging árboles: ejemplo

- Promedio de 100 árboles



Bagging árboles: importancia

- ¿Cómo medir la importancia de las variables a través de un modelo bagging de árboles de decisión?
- Problema de un bagging: dificultad en la interpretación.
- Un árbol «simple» era un modelo claramente interpretable y gráfico.
- Si construyo 2000 árboles de decisión... ¿cuál tiene sentido graficar para avanzar en la interpretación de la importancia de los predictores?

Bagging árboles: importancia

- Es necesario construir medidas agregadas que sirvan como resumen de la importancia de cada una de las variables a lo largo de todos los árboles construidos
 - RSS (suma de cuadrados de los residuos) en árboles de regresión: podemos calcular la cantidad en que el RSS se «achica» debido a los splits a lo largo de un predictor dado a lo largo de todos los B árboles. Valores grandes significan un predictor relevante
 - Gini en árboles de clasificación: parecido al anterior. Podemos agregar el valor total en que el índice de Gini es «achicado» por todos los splits en los que se usa un predictor en todos los árboles

Random forest

- Random forest es muy similar a un bagging de árboles de decisión
- La diferencia: además de generar variabilidad sobre los registros, se genera variabilidad sobre los predictores.
- En bagging, genero B predicciones a partir de
 - B remuestras bootstrap del TrS original y de
 - M predictores del TrS original
- De esta forma, en bagging entran el total de los M predictores.

Random forest

- Esto genera que los B árboles estén muy correlacionados. ¿Por qué?
- Random forest logra “descorrelacionar” los árboles a través de este pequeño ajuste: muestrear (permutar) un subconjunto (m) de los M predictores.
- De esta forma, cada vez que el algoritmo evalúa un split de un árbol, solamente considera una muestra aleatoria simple de m E M . Solamente considera las m variables elegidas y selecciona la que mejor split genera.

Random forest

- Supongamos que hay un predictor muy “fuerte” y muchos otros “moderados”.
- Al hacer bagging, probablemente este predictor fuerte entre en todos los árboles (y seguramente, además, en el primer split).
- En consecuencia, todos los árboles van a ser “parecidos”, es decir, correlacionados. Lo mismo va a pasar con las predicciones.
- Al promediar muchos árboles muy correlacionados, no se va a observar una ganancia en la reducción de la variancia del ensamble.

Random forest

- Random forests, lo soluciona obligando a considerar solamente un subset de predictores.
- Entonces, en promedio $(p - m)/p$ de todos los splits no van a considerar ese predictor fuerte y todos los otros predictores van a pesar más. Esto es “descorrelacionar” los árboles.
- En general, usar un valor pequeño de m debería ayudar cuando tenemos muchos predictores muy correlacionados.

Random forest

- Cada vez que el árbol genera un split nuevo, realiza una nueva selección de m variables.
- ¿Cuántas variables hay que seleccionar en cada split? Es decir, ¿cuál es valor de m ?
 - Una opción $m = \sqrt{M}$
 - Otra opción: seleccionarlo a través de cross-validation

Random forest - Interpretación

- ¿Cómo interpretar un clasificador random forest?
 - Importancia de las variables (decrecimiento en el MSE o en el error de clasificación de cada una de las variables a lo largo de todo el ensamble - ver más arriba)
- Otra opción (complementaria a la anterior) es comenzar a pensar en la “dependencia parcial”. Es decir, ¿cuál es la relación entre el predictor y la variable de respuesta?

Random forest – Dependencia parcial

- El error de clasificación de un nodo t se computa como:
 - $\text{Error}(t) = 1 - \max p(i|t)$
 - $p(j|t)$ es la frecuencia relativa de la clase p en el nodo t
- Mide los casos mal clasificados de un nodo.
 - Máximo valor: cuando los registros están igualmente distribuidos
 - Mínimo (0): cuando todos los registros pertenecen a una misma clase (máxima información).

Random forest – Dependencia parcial

- En un modelo de regresión logística
 - $\text{logit}(Y_i) = \ln\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p$
- los efectos marginales de las variables independientes son los P β_p de la regresión. Pero su interpretación es más compleja porque la relación no es lineal. El impacto de cada X sobre Y es función de los valores de X .

Random forest – Dependencia parcial

- Idea básica: obtener una predicción para cada valor único de X_p manteniendo constante el efecto del resto de las X_p .
- Luego, graficar las predicciones para cada valor de X_p . Esto debería brindar una aproximación a la forma funcional de la relación entre X_p e Y
- Dado que random forest puede aproximar casi cualquier relación entre X e Y resulta útil para detectar relaciones no lineales sin necesidad de parametrizarlas previamente

Random forest – Dependencia parcial

- Para valor de la variable de interés se crea un nuevo dataset en el cual a todas las observaciones se asigna el mismo valor en la variable de interés
- Este dataset es “dropped-down” en el bosque y una predicción para cada dataset es obtenida.
- Se promedian las observaciones y se obtiene una predicción final

Random forest – Dependencia parcial

- El resultado es “más” que los efectos marginales.
- Cada predicción se hace no “manteniendo constantes” los otros predictores sino que se usa toda la información disponible del resto de los predictores.
- Esto quiere decir que la relación graficada contiene todas las relaciones entre x_j e y , incluyendo los efectos “medios” de todas las interacciones de x_j con el resto de los predictores \mathbf{X}_j . Por eso, se lo llama “dependencia parcial” y no “dependencia marginal”.

Random forest – Dependencia parcial

- Para un determinado nodo, la etiqueta indica que la rama “izquierda” que sale de esa división agrupa a los casos cuyos valores son $X_j < t_k$; en cambio, la rama derecha corresponde a $X_j > t_k$. Por ejemplo, la primera división del árbol divide los registros en dos grandes ramas: la izquierda corresponde a los que han estado en la liga por menos de 4,5 años; La derecha a los que han estado 4,5 años o más.
- El árbol tiene dos nodos internos y tres nodos terminales (también llamados “hojas”). El número en cada hoja es la media de los registros que caen allí.

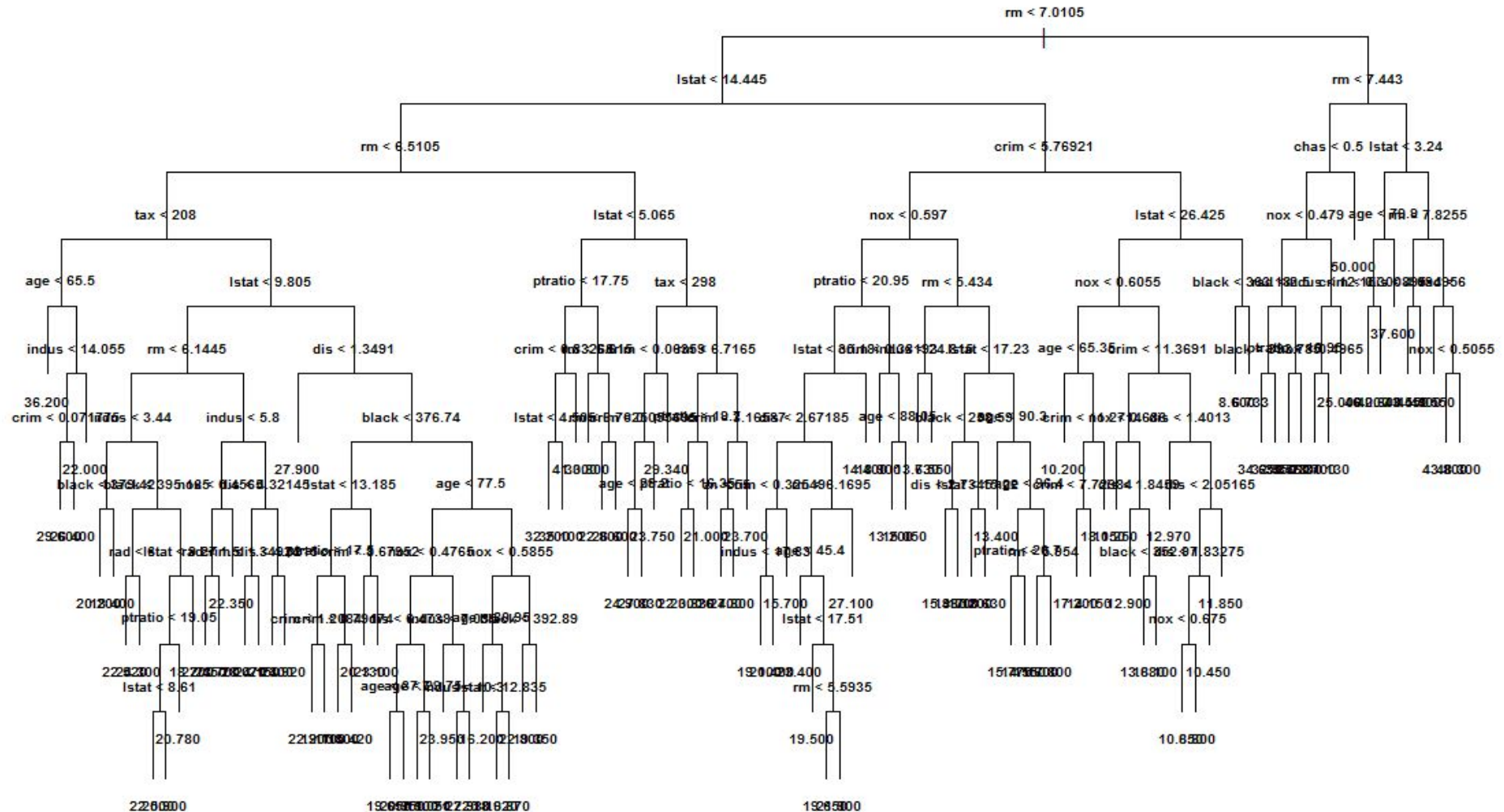
Random forest

- Ventajas según Breiman (2000):
 - Buena capacidad predictiva (comparable a boosting)
 - Robusto relativamente a ruido y a outliers
 - Más rápido que bagging y boosting
 - Da estimaciones de error e importancia de variables
- Desventajas
 - Funciona mejor en clasificación que en regresión

Random Forest - Ejemplo

- Datos: Boston dataset (ISLR package)
- Objetivo: predecir el precio mediano de las viviendas en diversos condados de Boston
 - 1. CRIM: per capita crime rate by town
 - 2. ZN: proportion of residential land zoned for lots over 25,000 sq.ft.
 - 3. INDUS: proportion of non-retail business acres per town
 - 4. CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
 - 5. NOX: nitric oxides concentration (parts per 10 million)
 - 6. RM: average number of rooms per dwelling
 - 7. AGE: proportion of owner-occupied units built prior to 1940
 - 8. DIS: weighted distances to five Boston employment centres
 - 9. RAD: index of accessibility to radial highways
 - 10. TAX: full-value property-tax rate per \$10,000
 - 11. PTRATIO: pupil-teacher ratio by town
 - 12. B: $1000(B_k - 0.63)^2$ where B_k is the proportion of blacks by town
 - 13. LSTAT: % lower status of the population
 - 14. MEDV: Median value of owner-occupied homes in \$1000's

Random Forest – Ej: árbol máximo



Comparación Random Forest y bagg

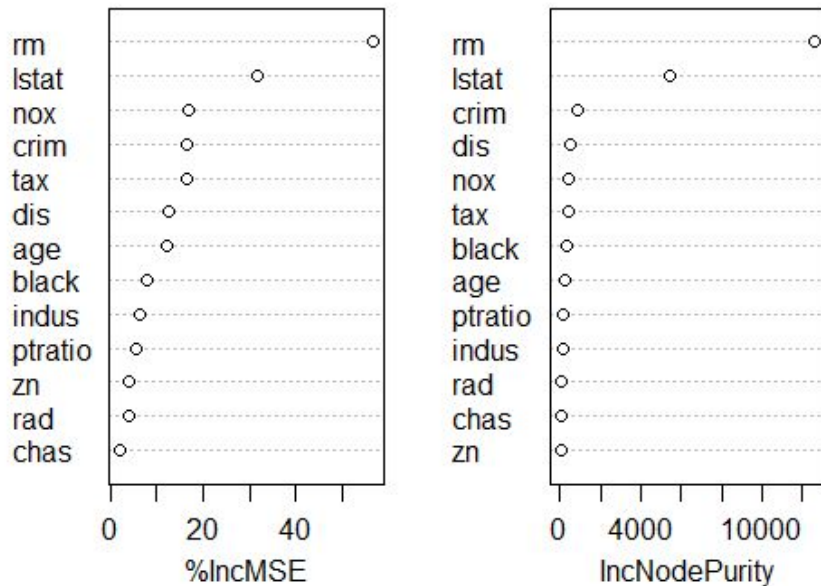
- Se estimaron un modelo bagging y 12 modelos random forest variando el parámetro m_{try} (es decir, cantidad de predictores que se muestrean en cada split).
- En cada modelo se entrenaron 500 árboles
- Se dividió el dataset en 50% como TrS y 50% como TeS

Comparación Random Forest y bagg

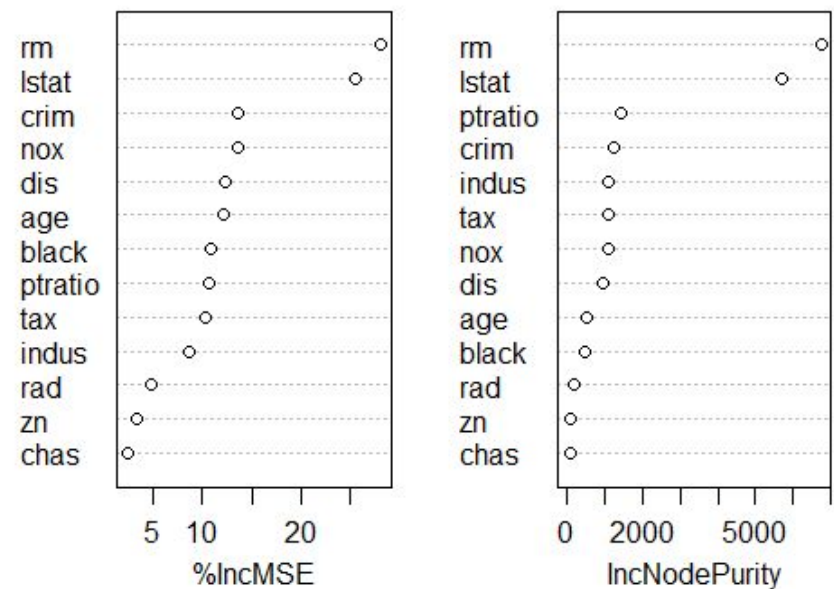
	OOB Error	Test Error
Bagging m=13	11.04	21.84
R. Forest m=1	19.08	27.3
R. Forest m=2	13.3	20.58
R. Forest m=3	11.63	18.89
R. Forest m=4	11.05	18.35
R. Forest m=5	10.64	18.32
R. Forest m=6	10.63	18.00
R. Forest m=7	10.42	18.63
R. Forest m=8	10.38	19.34
R. Forest m=9	10.55	19.56
R. Forest m=10	10.62	20.15
R. Forest m=11	10.84	21.08
R. Forest m=12	10.95	21.7

Random Forest - Ejemplo

Bagging

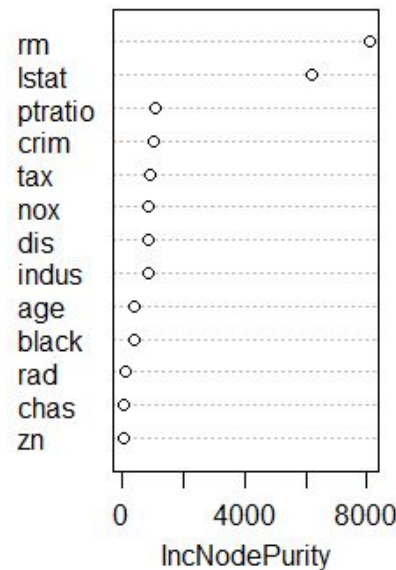
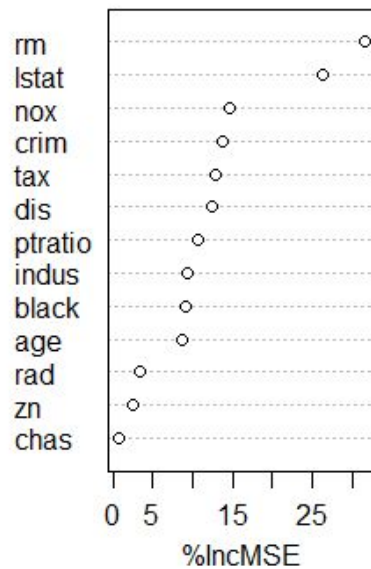


RF m=4

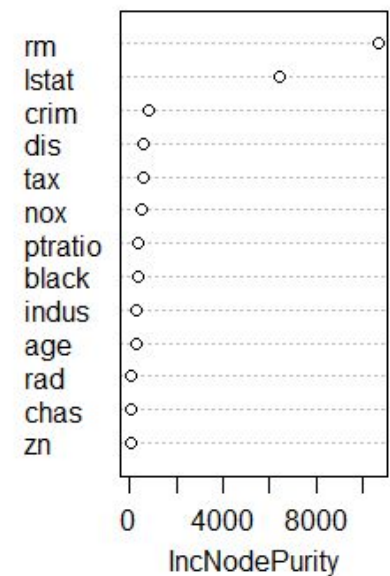
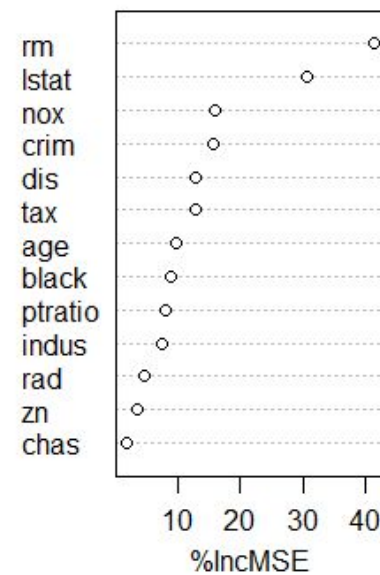


Random Forest - Ejemplo

RF m=6

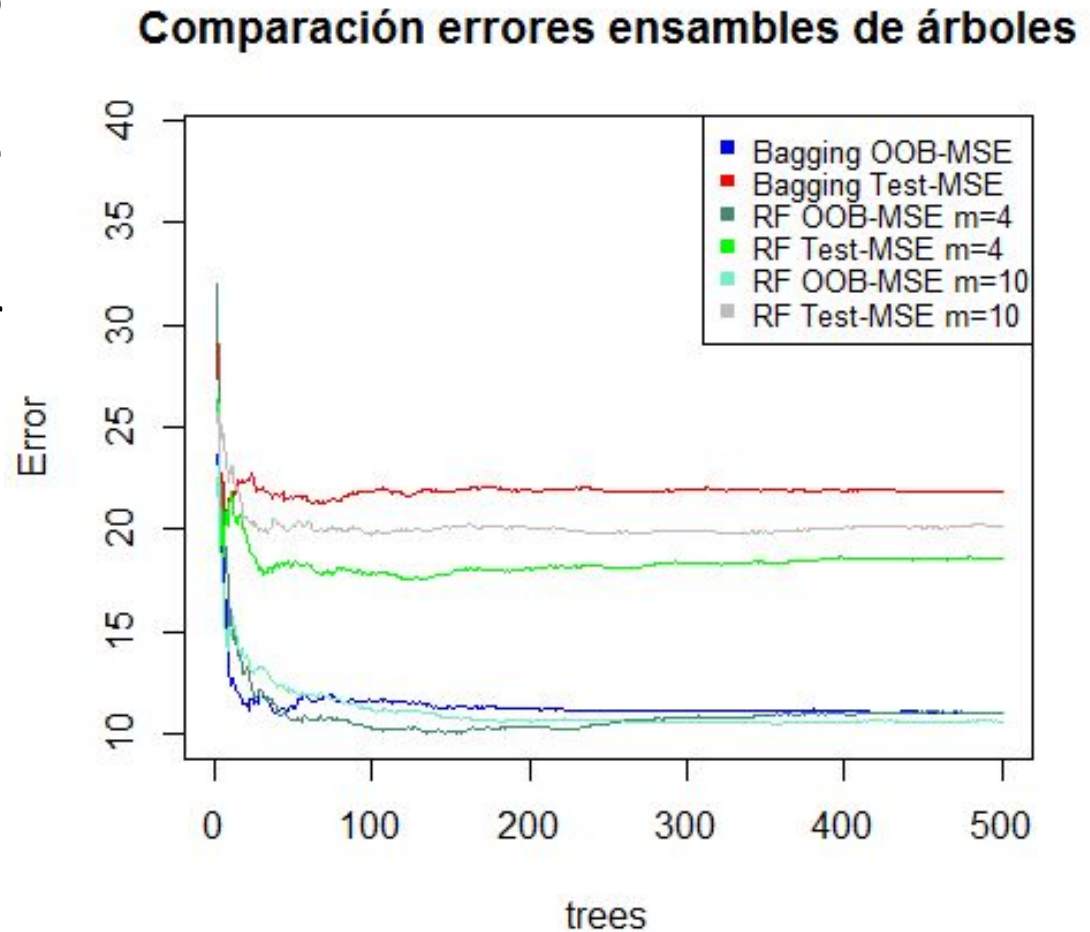


RF m=10



Random Forest - Ejemplo

- A partir de determinado tamaño de árboles el modelo no parece sobreajustarse
- RF es el que menor error en el test set muestra



Random Forest - Ejemplo

Gráficos de dependencia parcial de algunas variables

