Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Sistemas Urbanos Inteligentes

Árboles y Ensambles

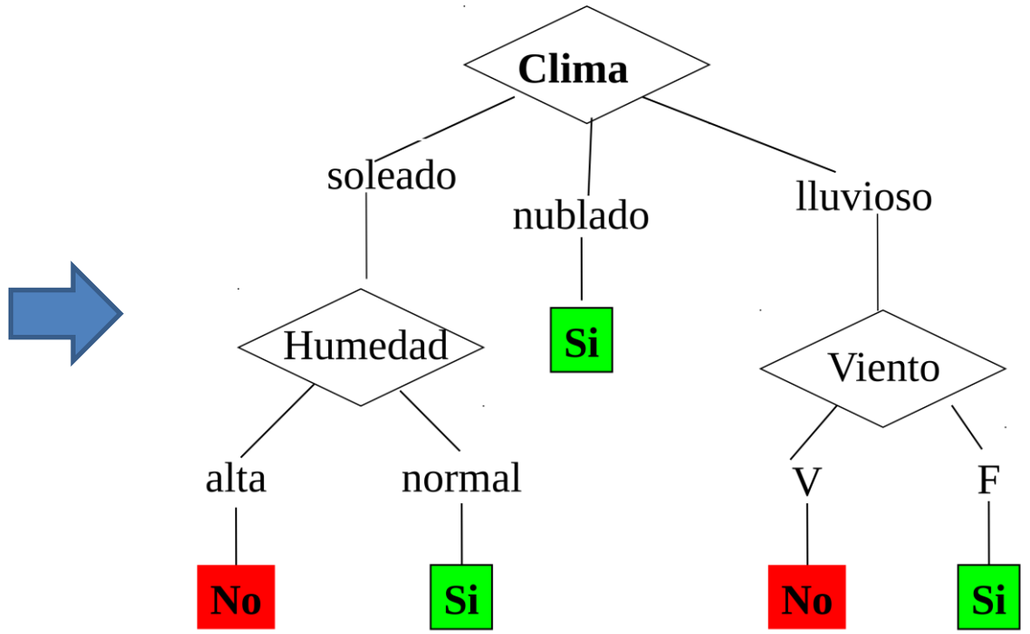
Hans Löbel
Dpto. Ingeniería de Transporte y Logística
Dpto. Ciencia de la Computación

¿Cómo solucionamos el siguiente problema de clasificación?

Clima	Temperatura	Humedad	Viento	Jugar?
soleado	alta	alta	F	No
soleado	alta	alta	V	No
nublado	alta	alta	F	Si
lluvioso	Agradable	alta	F	Si
lluvioso	frio	normal	F	Si
lluvioso	frio	normal	V	No
nublado	frio	normal	V	Si
soleado	Agradable	alta	F	No
soleado	frio	normal	F	Si
lluvioso	Agradable	normal	F	Si
soleado	Agradable	normal	V	Si
nublado	Agradable	alta	V	Si
nublado	alta	normal	F	Si
lluvioso	Agradable	alta	V	No

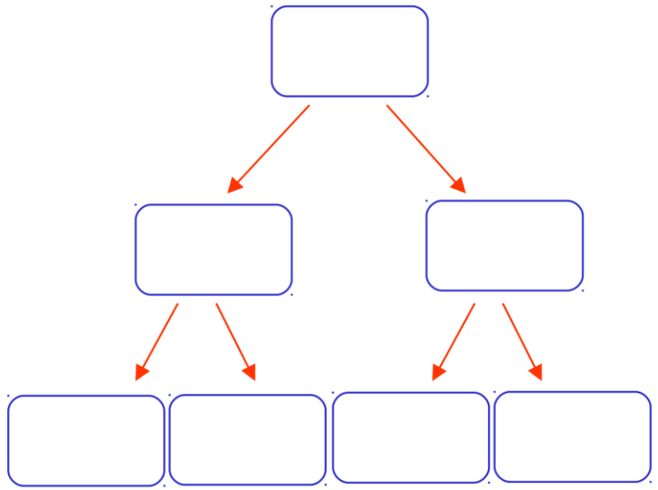
¿Cómo solucionamos el siguiente problema de clasificación?

Clima	Temperatura	Humedad	Viento	Jugar?
soleado	alta	alta	F	No
soleado	alta	alta	V	No
nublado	alta	alta	F	Si
lluvioso	Agradable	alta	F	Si
lluvioso	frio	normal	F	Si
lluvioso	frio	normal	V	No
nublado	frio	normal	V	Si
soleado	Agradable	alta	F	No
soleado	frio	normal	F	Si
lluvioso	Agradable	normal	F	Si
soleado	Agradable	normal	V	Si
nublado	Agradable	alta	V	Si
nublado	alta	normal	F	Si
lluvioso	Agradable	alta	V	No



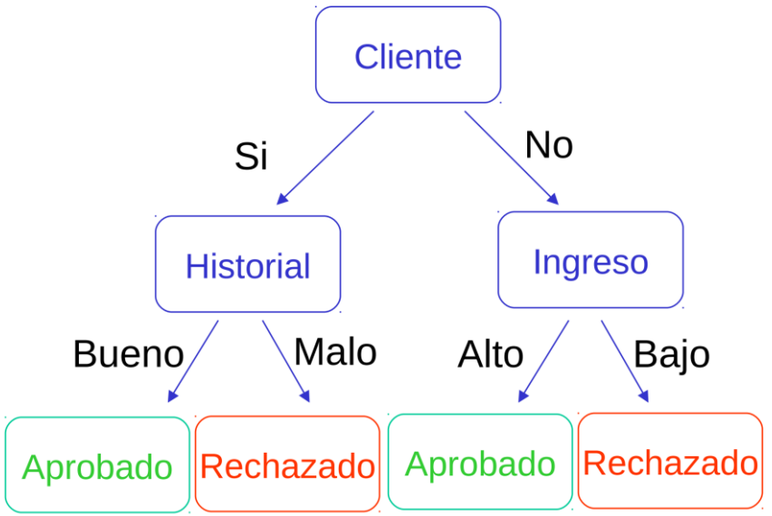
Árboles de decisión pueden solucionar el caso anterior

- Técnica (principalmente) de aprendizaje supervisado.
- Pueden realizar clasificación y regresión.
- Pueden usarse sobre distintos tipos de variables (binaria, categórica, numérica) sin mayor esfuerzo.

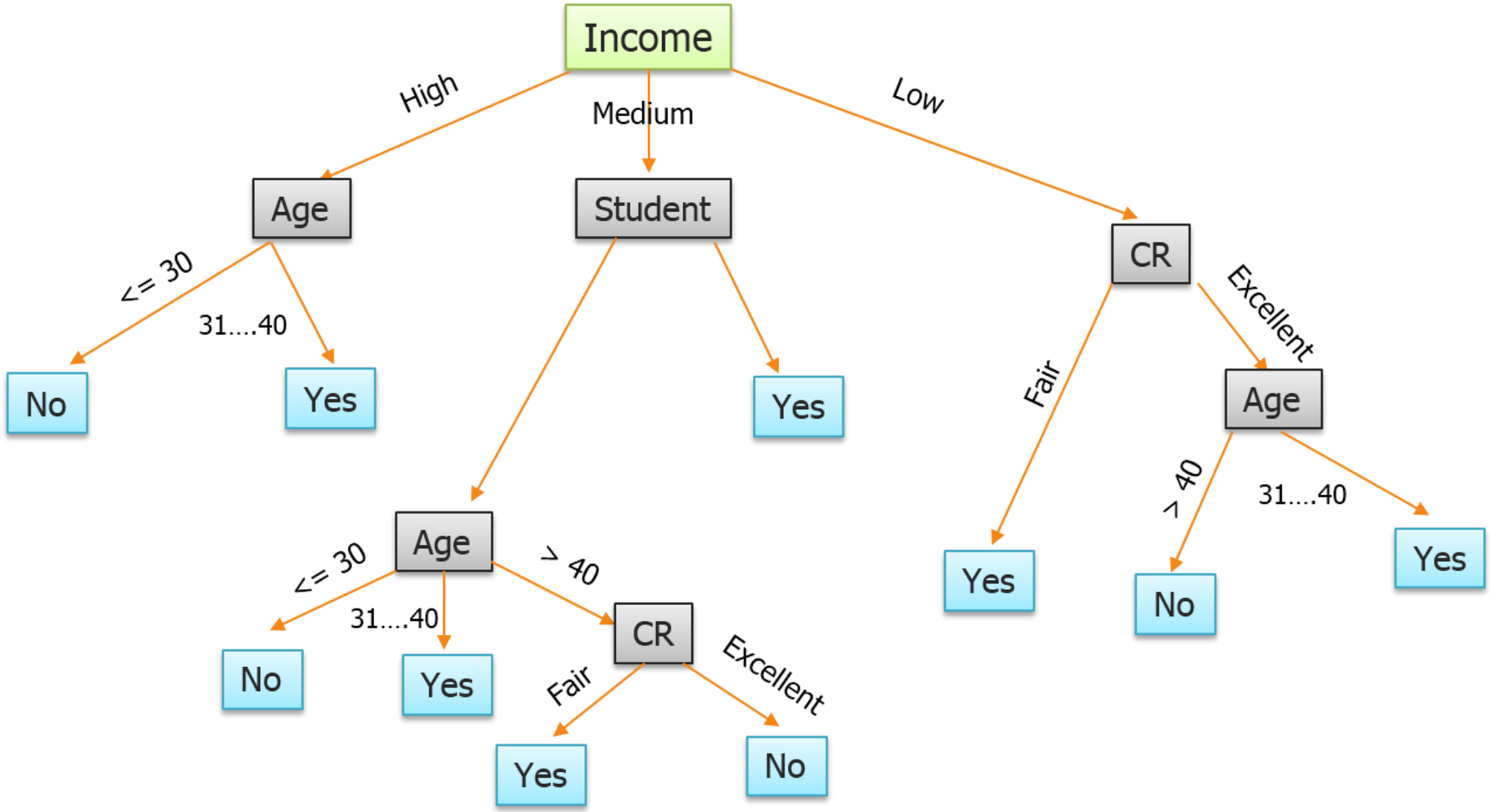


Árboles de decisión son ampliamente utilizados en la práctica

- Cada nodo interno representa un atributo y cada nodo hoja representa una categoría.
- En cada nodo interno, se realiza un test en base a los valores del atributo.
- Aristas representan el resultado del test.
- Para clasificar un registro, se debe pasar desde la raíz hasta alguna hoja.

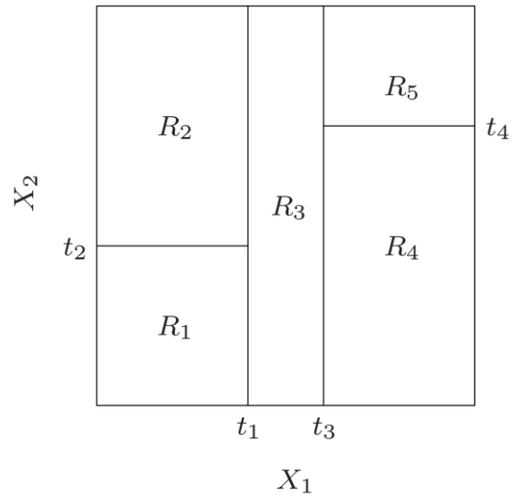


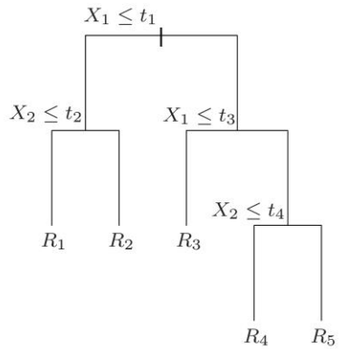
¿Cómo construimos un árbol de decisión?

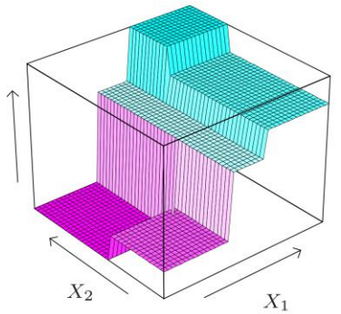


Podemos usar un algoritmo recursivo para construir un árbol

1. ¿Pertenecen todos los registros a la misma clase?
   - Retornar marcando el nodo hoja con la clase respectiva.
2. ¿Tienen todos los registros el mismo valor para todos los atributos que determinan su clase?
   - Retornar marcando nodo hoja con la clase más común.
3. De lo contrario:
   i. Seleccionar el mejor atributo.
   ii. Usar ese atributo como nodo raíz.
   iii. Dividir el set de entrenamiento de acuerdo a este atributo y para cada rama resultante continuar la construcción del árbol en forma recursiva.

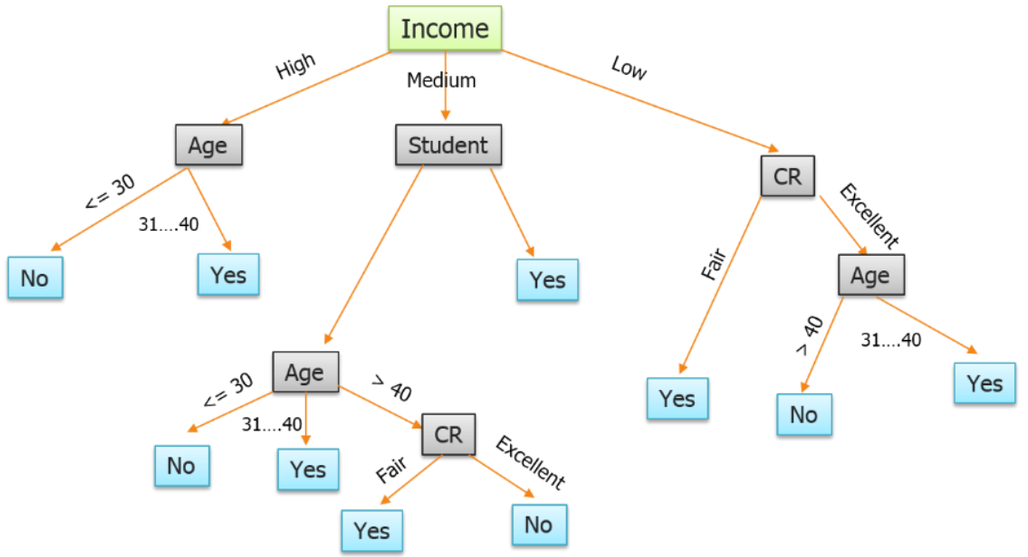






Podemos destilar este paso en dos preguntas más específicas

1. ¿Cómo defino cuál es el mejor atributo?
2. ¿En que parte del dominio pongo el umbral de separación para los atributos numéricos?

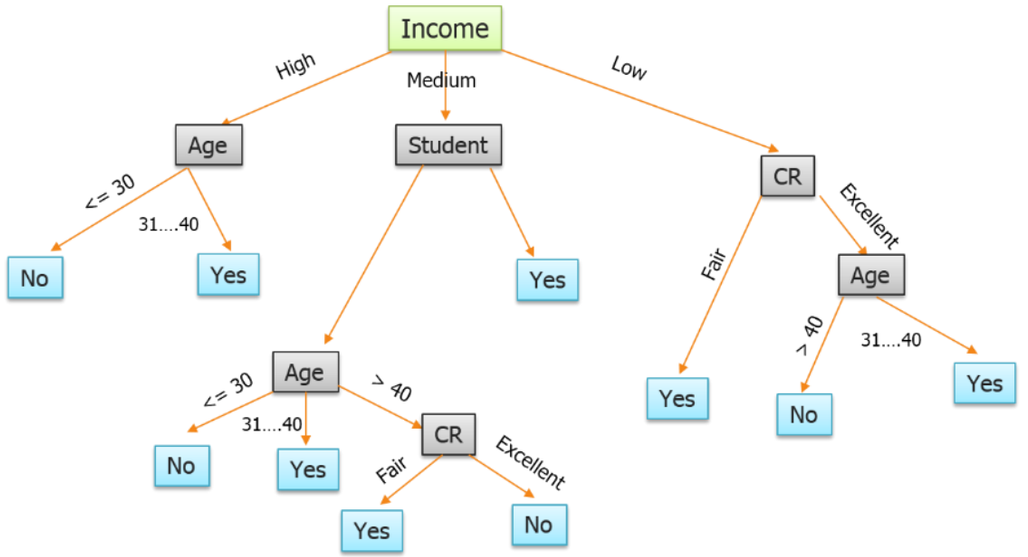


Todo depende de la tarea que busquemos solucionar

- Si objetivo es clasificar, es razonable que el mejor atributo sea el que mejor separe las clases.
- Dos maneras típicas de medir esto son:
  - Gini Index: desigualdad (inequidad) sobre distintas categorías.
  - Information Entropy: uniformidad (entropía) de una distribución / bits promedios necesarios para codificar información sobre ocurrencia de eventos.

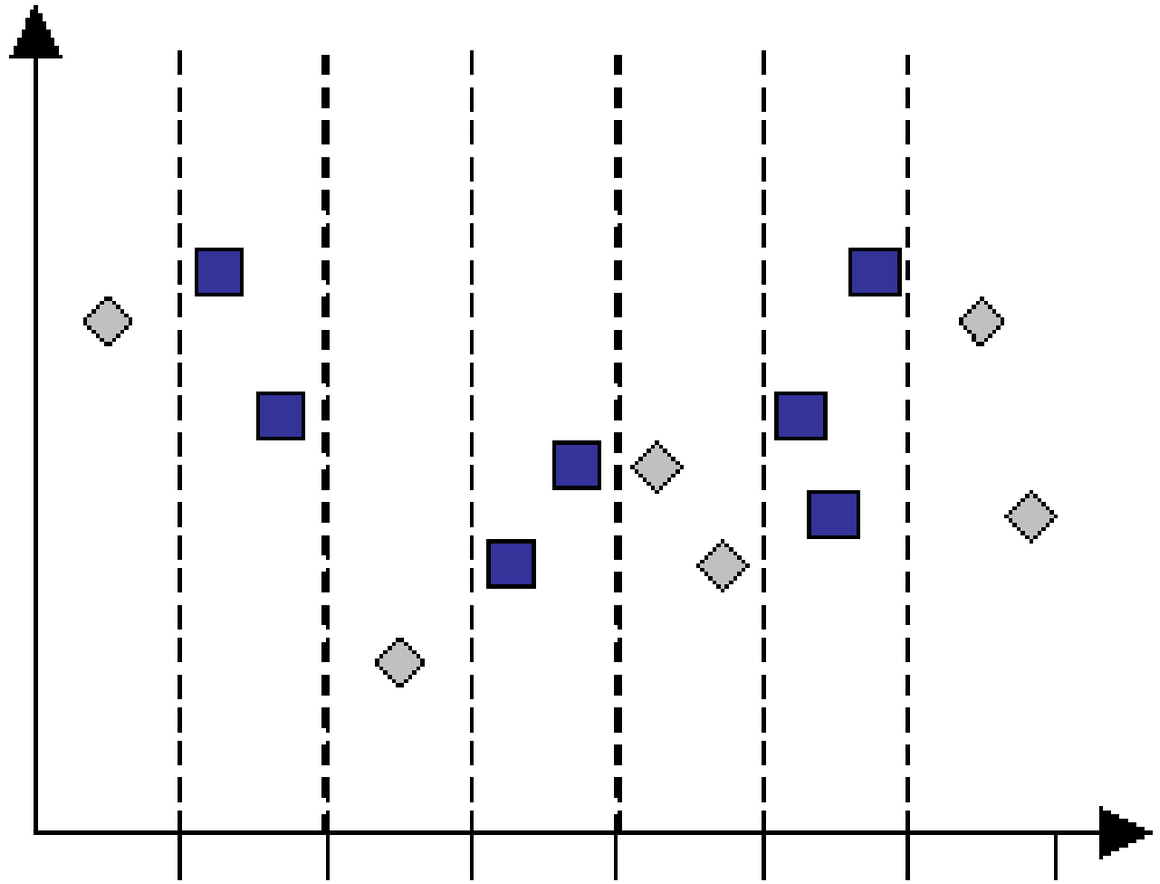
Podemos destilar este paso en dos preguntas más específicas

1. ¿Cómo defino cuál es el mejor atributo?
2. ¿En que parte del dominio pongo el umbral de separación para los atributos numéricos?



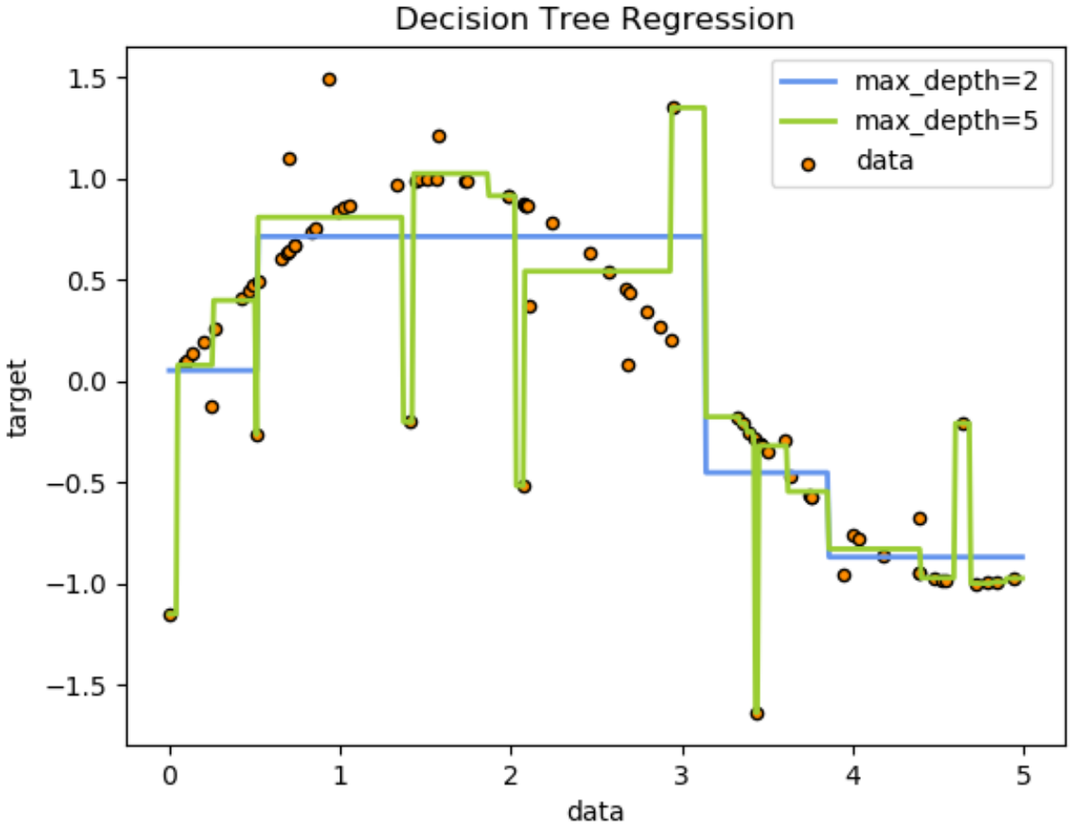
Nuevamente tenemos dos opciones directas

- Fuerza bruta
- Ordenar por dimensión, y evaluar split en cada cambio de categoría.



Una última consideración es el uso de árboles para hacer regresión

- Cada nodo hoja retorna la media de los ejemplos que contiene.
- Para elegir el atributo, se calcula la disminución de la varianza (VR).
- Construcción recursiva sigue la misma idea que para clasificación.



Volvamos un poco a *overfitting* y complejidad

¿Qué tipo de árboles prefiere construir el algoritmo que analizamos?

(sesgo inductivo / bias del algoritmo)

Árboles pocos profundos (mientras más arriba mejora la clasificación, mejor)

¿Tiene esto algo que ver con el sobreajuste?

Occam’s Razor[1] (o la navaja de Occam, claramente una traducción poco afortunada) be my guide

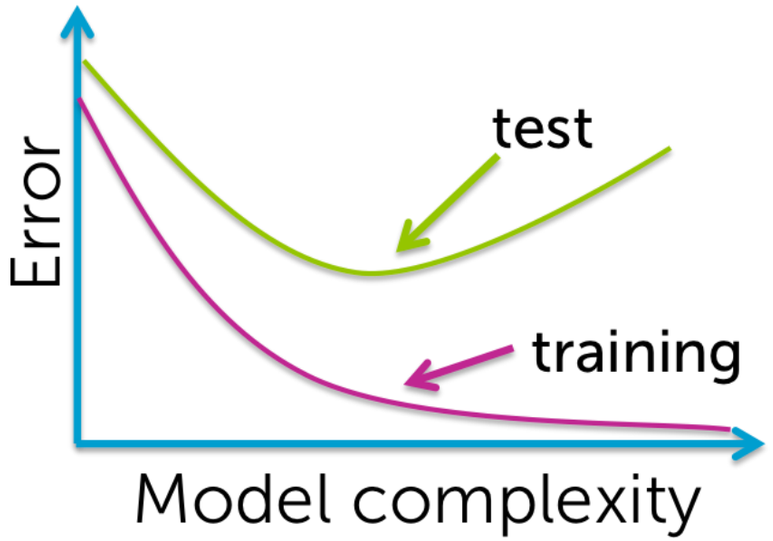
En igualdad de condiciones, la explicación más sencilla suele ser la más probable [2]

[1] Supuestamente fue enunciado mientras se afeitaba.
[2] Esto es un principio filosófico/metodológico, no una ley de la naturaleza.

Overfitting es un problema importante para los árboles de decisión

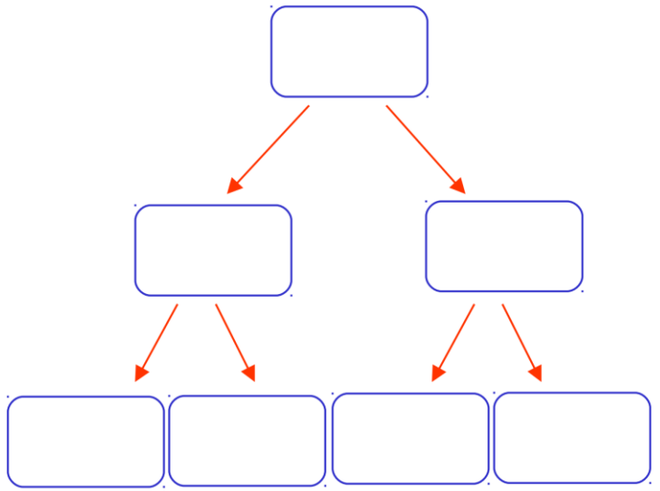
Existen varias técnicas que pueden ayudar reduciendo el overfitting.

- Detener construcción del árbol en base a un set de validación.
- Detener construcción cuando registros restantes no son estadísticamente significativos.
- Construir un árbol completo y luego podar ramas.
- Penalizar complejidad en métrica de selección del siguiente atributo.



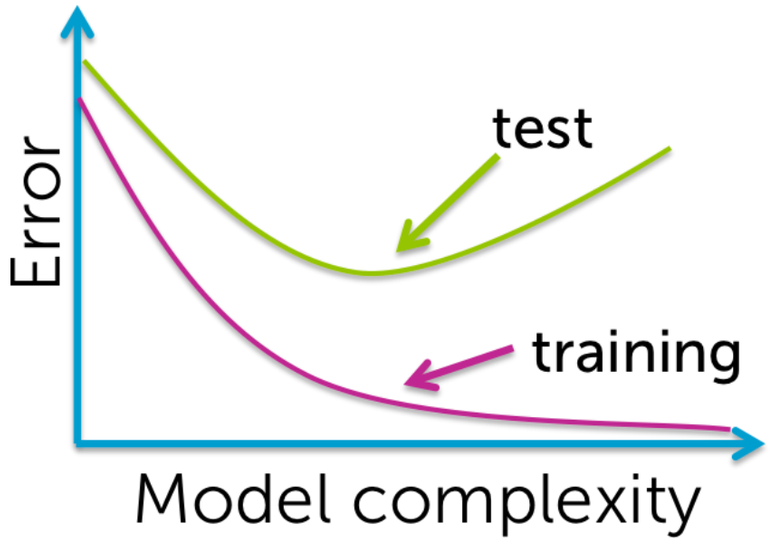
Árboles de decisión son ampliamente utilizados en la práctica

- Gran ventaja radica en la simplicidad y facilidad de interpretación.
- Pueden sufrir de serios problemas de sobreajuste.



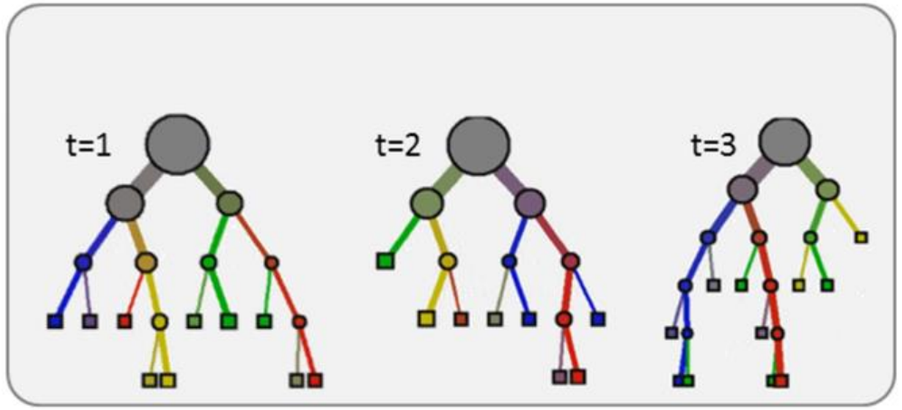
¿Podemos hacer algo más para evitar el overfitting en los árboles?

- ¿De donde proviene este problema?
  - Muestra es progresivamente reducida al bajar en el árbol.
  - Si hay muchos atributos, existe alta probabilidad de elegir alguno “inútil” en cuanto a generalización, pero bueno para entrenar.



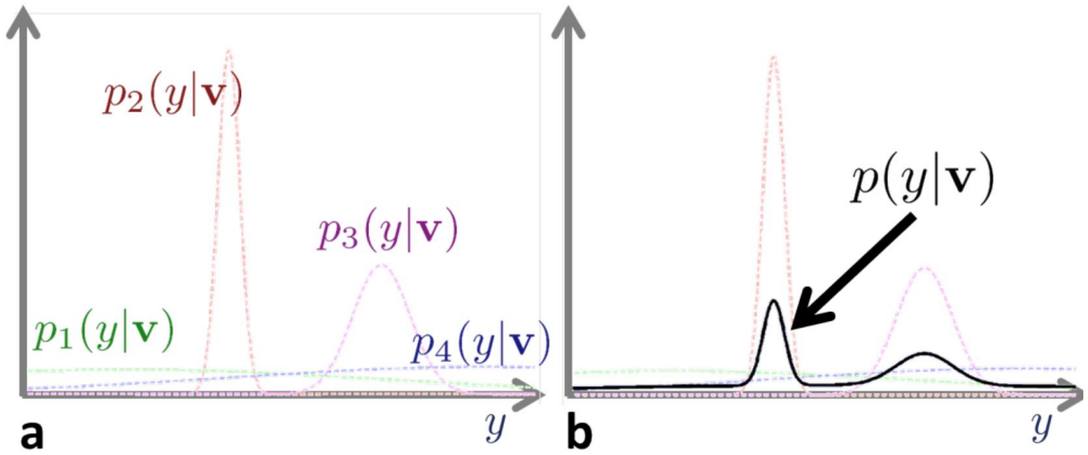
Ensambles de modelos

- Si queremos estar seguros de algo, le preguntamos a más de un modelo.
- Si le preguntamos a suficientes, es probable que el promedio sea cercano a la verdad.
- ¿Qué condición deben cumplir los modelos para que esto funcione?



Ensambles de modelos no correlacionados

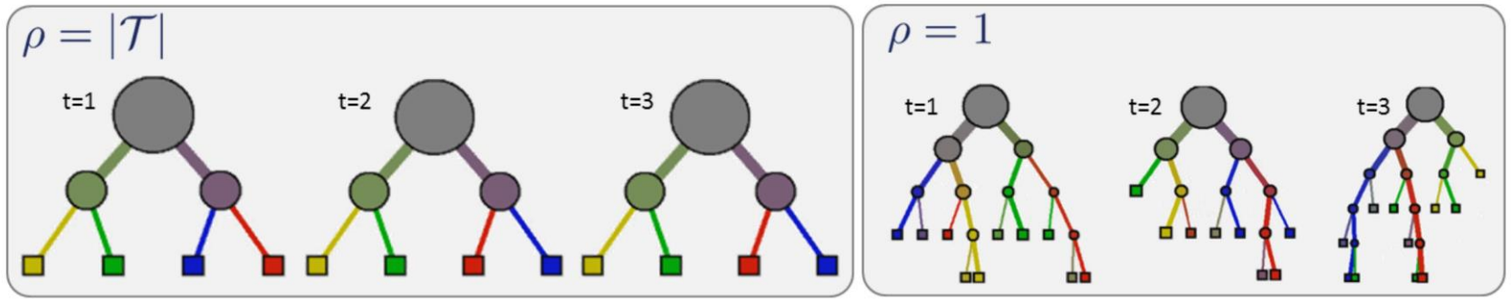
- Si el patrón de error (acierto) es distinto para todos, es altamente probable que en promedio, la respuesta del ensamble sea correcta.
- Para que esto ocurra, todos los modelos del ensamble deben ser débiles, es decir, ser un poco mejores que dejar la decisión al azar.



¿Cómo logramos esto con los árboles?

La solución está en las muestras aleatorias

- Si cada modelo ve sólo una parte de los datos (elegida aleatoriamente), la correlación entre ellos disminuye.



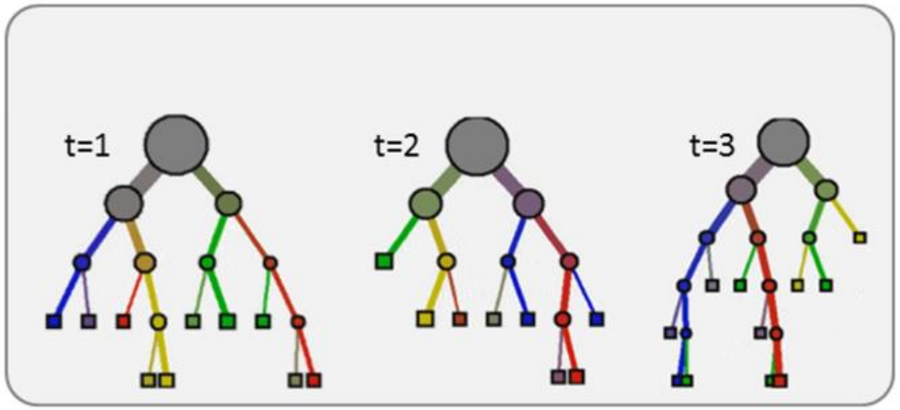
a) Low randomness, high tree correlation

b) High randomness, low tree correlation

¿Qué pasa si hay una variable muy buena?

La solución está nuevamente en las muestras aleatorias, pero ahora de atributos

- Cada vez que se hace un *split*, se elige un subconjunto aleatorio de los atributos.
- Luego, para cada uno de ellos, se calcula la ganancia de información.
- Además, se limita la profundidad del árbol.
- Esto trae además el beneficio de reducir la complejidad.
- Por lo general, se utiliza una muestra proporcional a la raíz del número de atributos.



La introducción de todas estas estrategias “aleatorizantes”, transforma a los árboles en *random forests*

Algorithm 15.1 *Random Forest for Regression or Classification.*

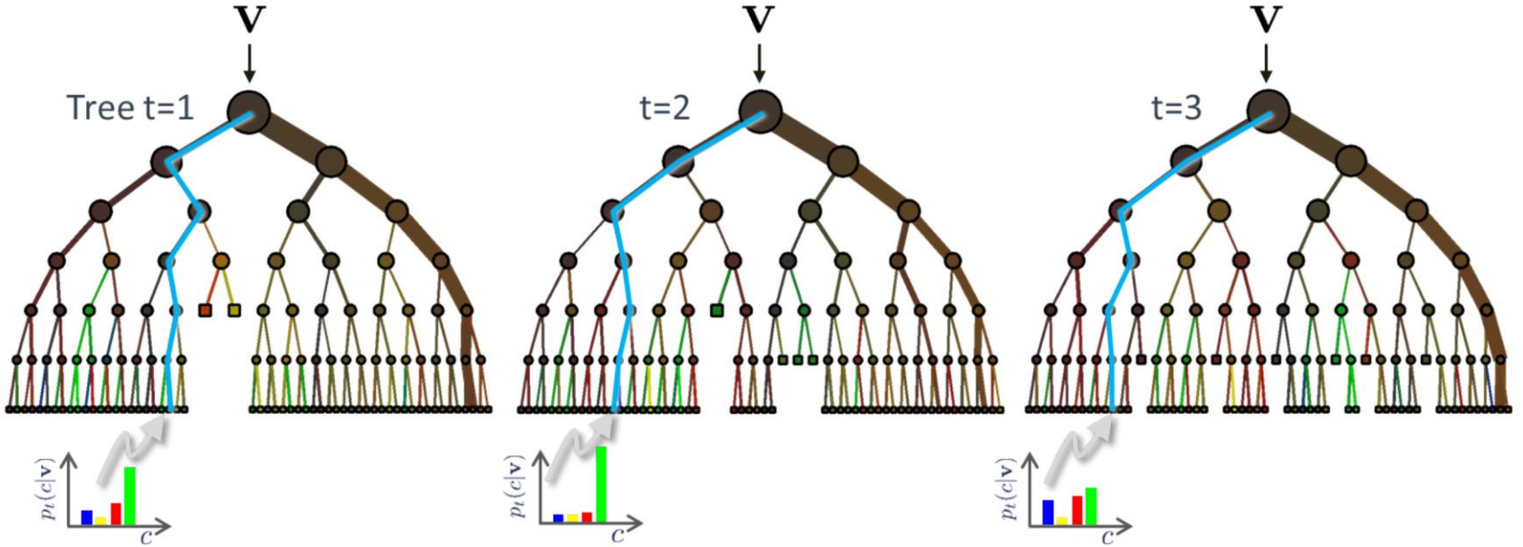
1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

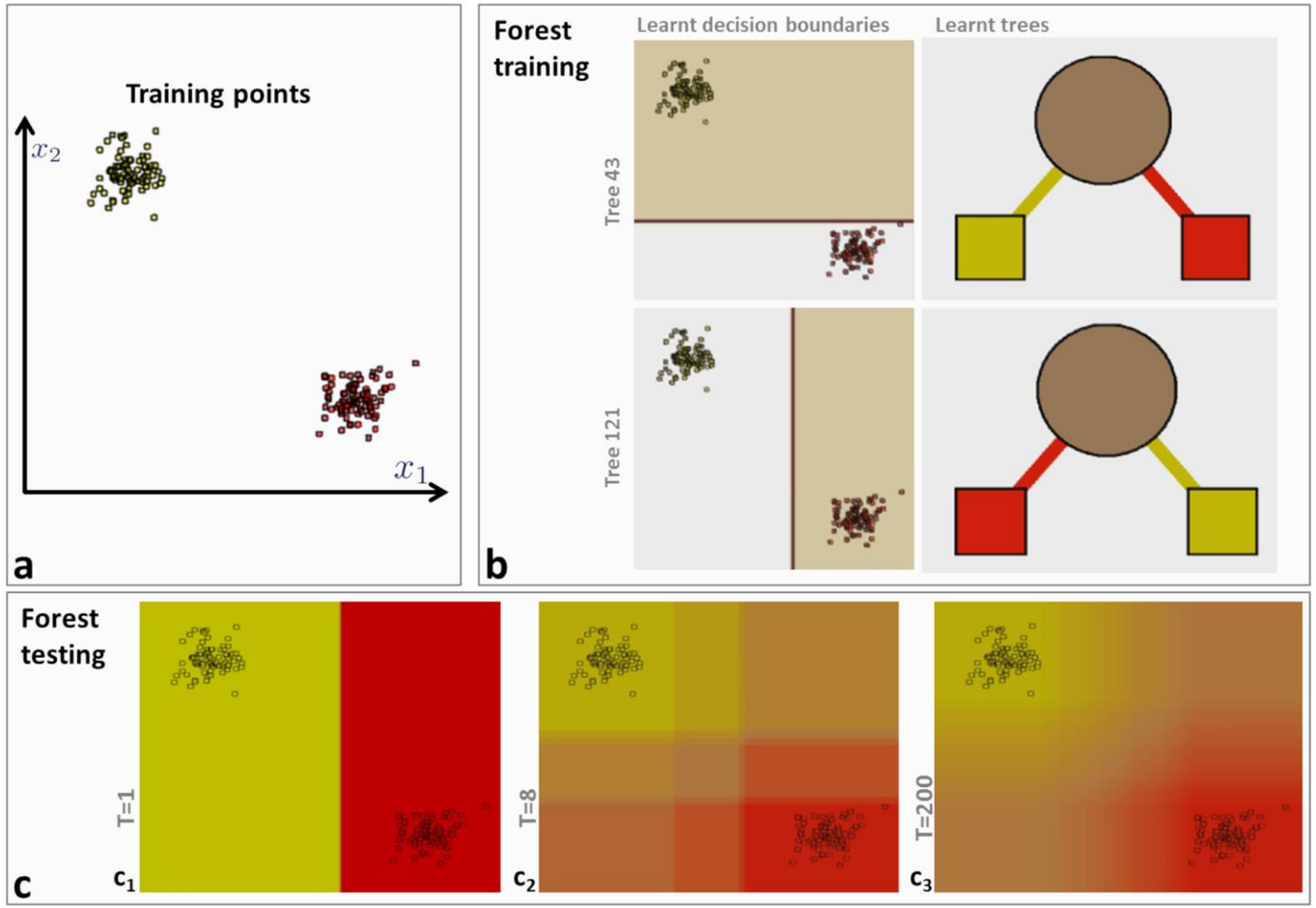
To make a prediction at a new point x:

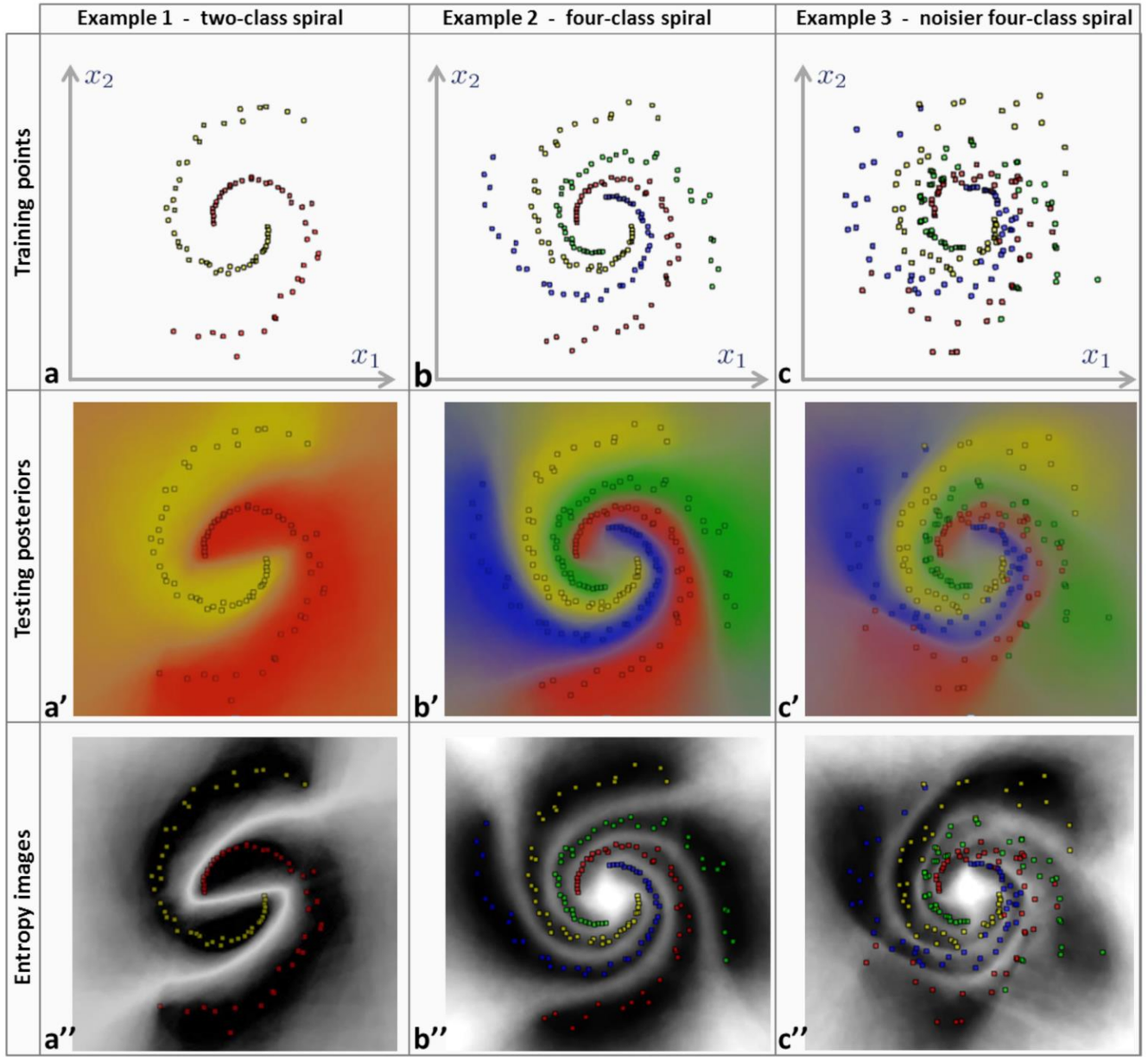
Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

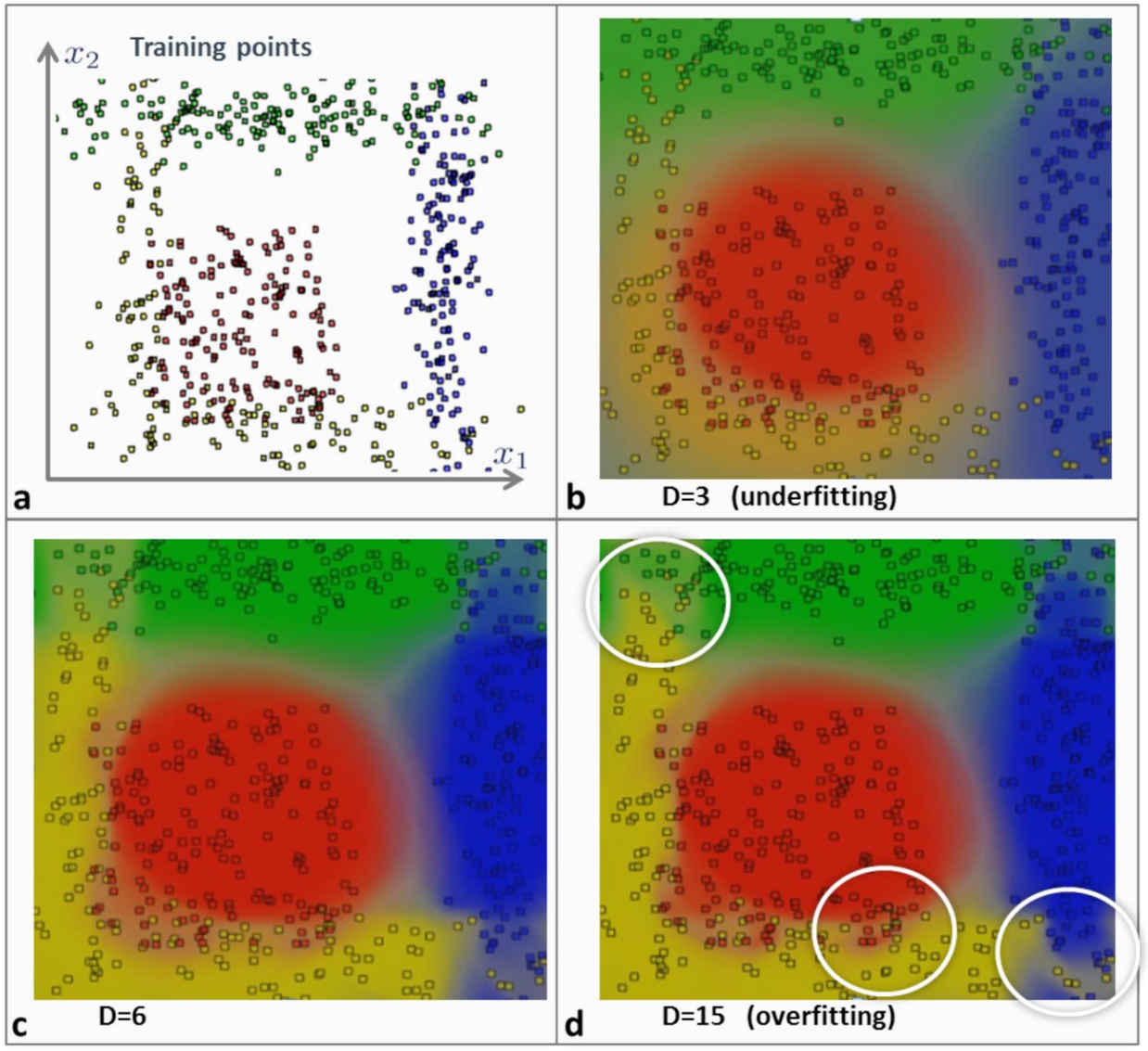
Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

La introducción de todas estas estrategias “aleatorizantes”, transforma a los árboles en *random forests*



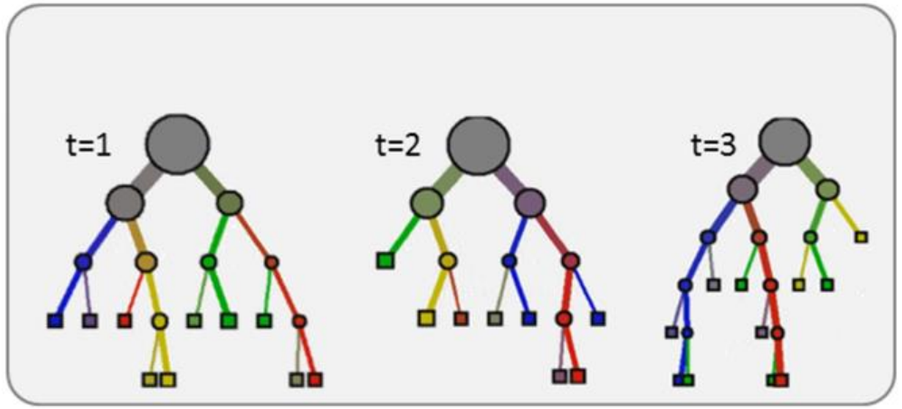

Training points
x_2
x_1
a
Forest training
Learnt decision boundaries
Learnt trees
Tree 43
Tree 121
b
Forest testing
T=1
T=8
T=200
c_1
c_2
c_3
c


Example 1 - two-class spiral
Example 2 - four-class spiral
Example 3 - noisier four-class spiral
Training points
Testing posteriors
Entropy images
x_2
x_1
a
b
c
a'
b'
c'
a''
b''
c''


Training points
x_2
x_1
a
b
D=3 (underfitting)
c
D=6
d
D=15 (overfitting)

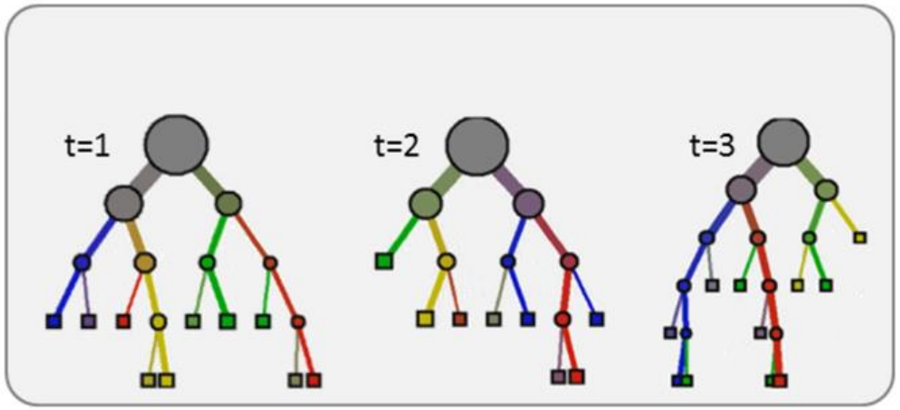
Algunos consideraciones relacionadas con la evaluación

- Es posible medir el rendimiento usando ejemplos no vistos por lo árboles (*out-of-bag error*)
- Para medir la importancia de una variable, basta con permutar su valor entre los ejemplos (¿por qué?).



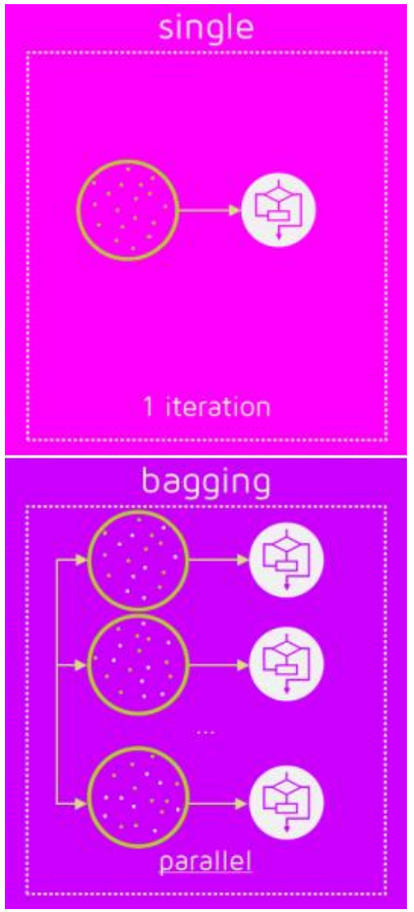
Random Forests son ampliamente utilizados en la práctica

- Al igual que los árboles de decisión, son modelos fácilmente interpretables.
- Rendimiento es altamente competitivo en problemas con datos *estructurados*.
- Gracias a aleatorización en su construcción, son altamente resistentes al *overfitting*.



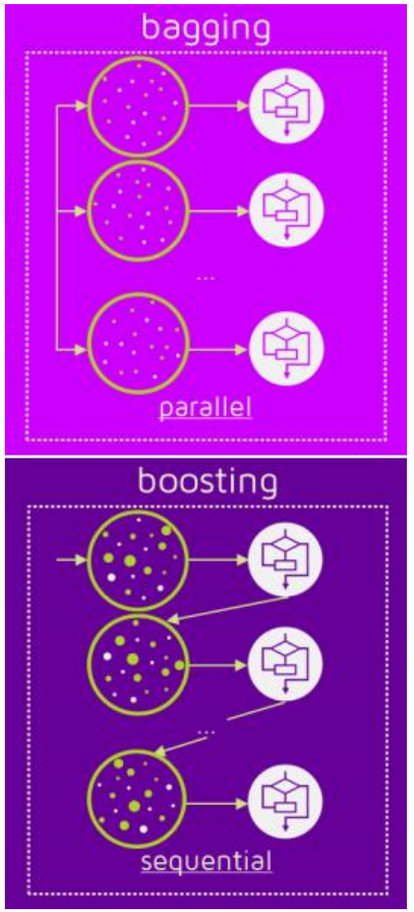
Random Forests se basan en un concepto llamado *bagging*

- *Bagging* busca ensamblar múltiples modelos, donde la baja correlación entre ellos permite mejorar el rendimiento.
- Una manera distinta de hacer ensambles es intentar que exista una correlación explícita entre los modelos, pero enfocada en la reducción del error.
- Este esquema de ensamble se conoce como *boosting*.



Boosting explora la construcción iterativa de ensambles

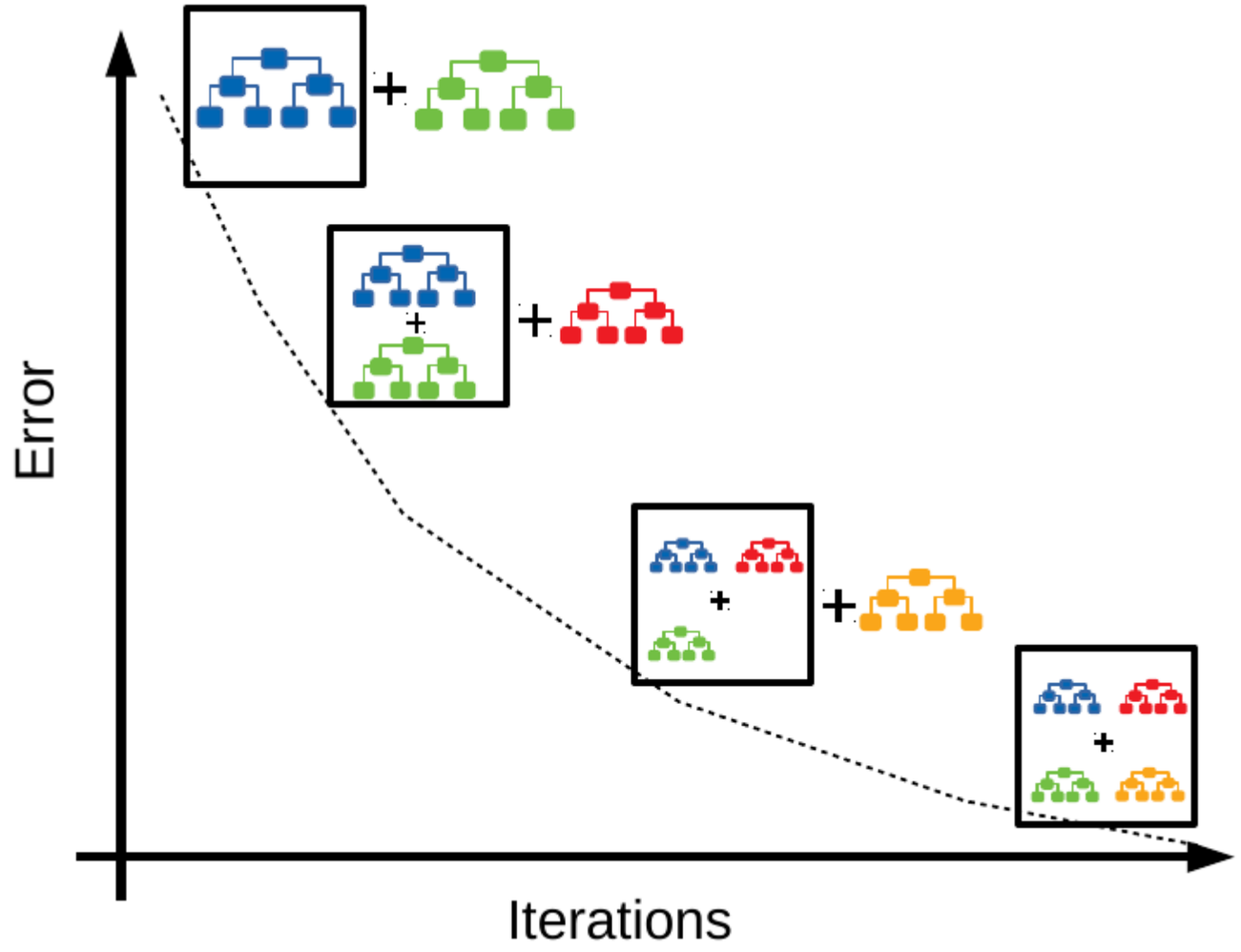
- Progresivamente, cada modelo débil busca corregir el error del ensamble.
- Esto significa que ve una “versión modificada” de la señal de supervisión.
- Una vez entrenado, este modelo débil es agregado al ensamble y se continua el proceso.
- El mecanismo de *boosting* más exitoso en la actualidad es *Gradient Boosting*

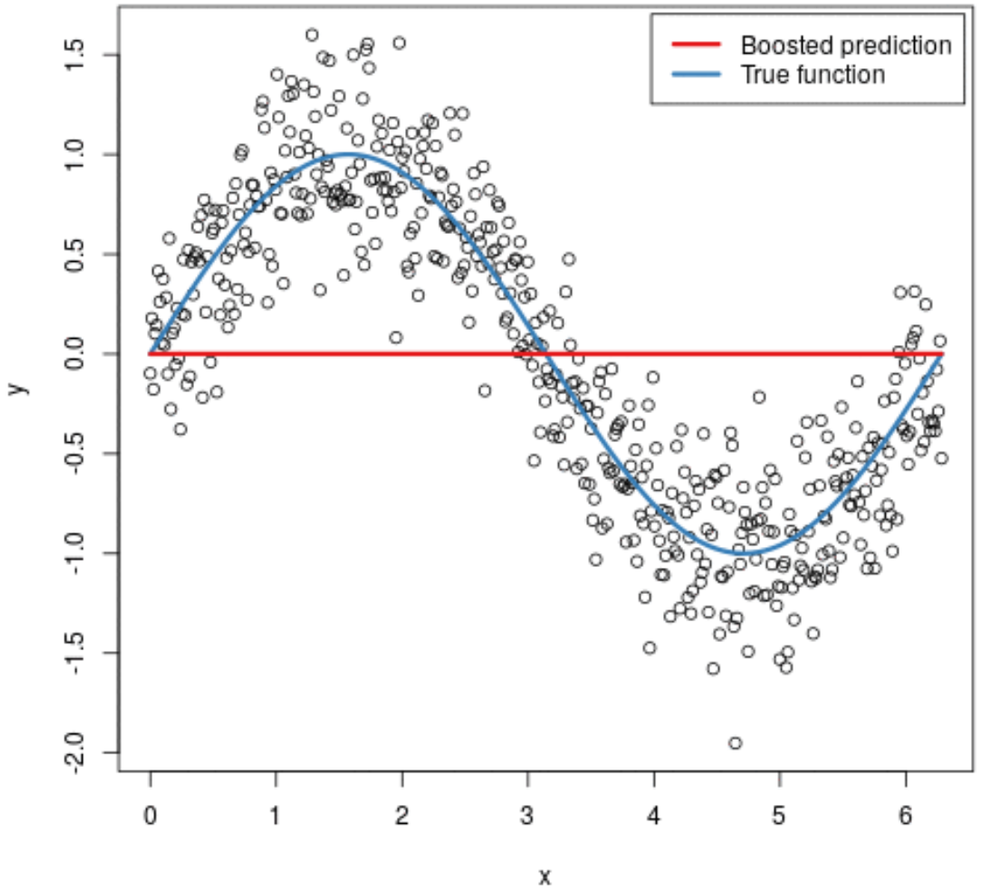


¿Cómo funciona este *Gradient Boosting*?

1. Entrenar un clasificador/regresor: $F_1(x) = h_0(x) \simeq y$
2. Entrenar un nuevo clasificador/regresor, pero ahora usando el residuo de F_1: $h_1(x) \simeq y - F_1(x)$
3. Sumar este nuevo clasificador/regresor al ensamble: $F_2(x) = F_1(x) + h_1(x)$
4. Entrenar un nuevo clasificador/regresor con los residuos de F_2: $h_2(x) \simeq y - F_2(x)$
5. Sumar este nuevo clasificador/regresor al ensamble: $F_3(x) = F_2(x) + h_2(x)$
6. Continuar este proceso, hasta cumplir algún criterio de fin. El clasificador/regresor final estará dado por:

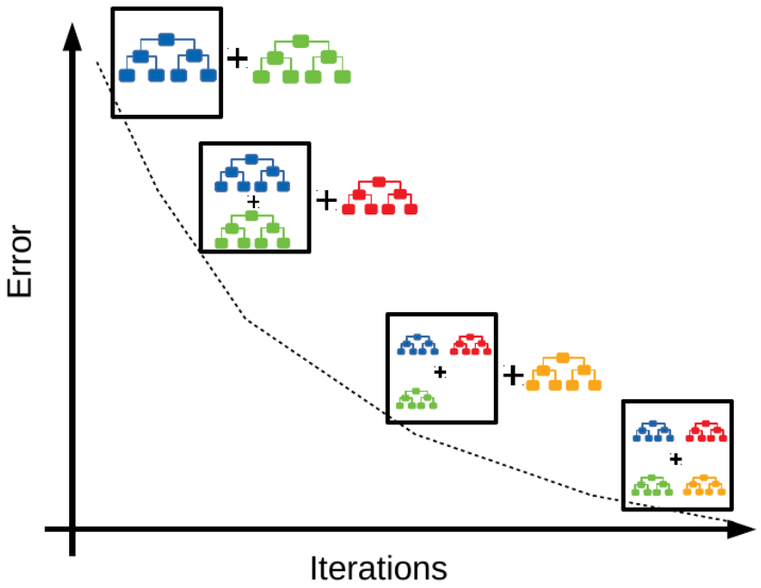
$$f(x) = \sum_{i=1}^{N} h_i(x)$$


Error
Iterations


Boosted prediction
True function
y
x
1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
-2.0
0
1
2
3
4
5
6

¿Cómo funciona este *Gradient Boosting*?

- En la práctica, el proceso anterior funciona muy bien (con ciertos arreglos).
- Variedades modernas, como XGBoost o LightGBM, son ideales para procesar grandes volúmenes de datos estructurados (tabulados), con rendimiento superior a las redes neuronales profundas tradicionales.



Lecturas de profundización (100% optativas)

- "*Clearing air around "Boosting*", artículo en Towards Data Science: https://towardsdatascience.com/clearing-air-around-boosting-28452bb63f9e
- "*The Evolution of Boosting Algorithms From Machine Learning to Statistical Modelling*", de Andreas Mayr et al.: https://arxiv.org/abs/1403.1452

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Sistemas Urbanos Inteligentes

Árboles y Ensambles

Hans Löbel
Dpto. Ingeniería de Transporte y Logística
Dpto. Ciencia de la Computación