

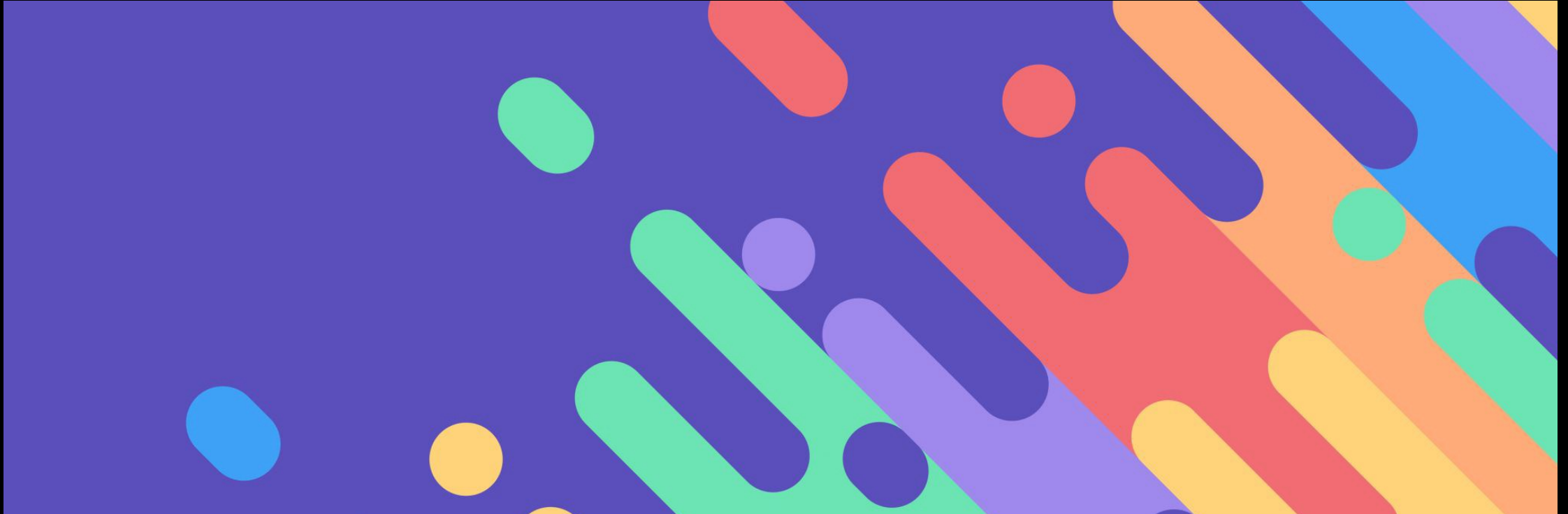
# ÁRBOLES DE DECISIÓN



Aprendizaje  
Automático

CEIoT - FIUBA

Dr. Ing. Facundo Adrián  
Lucianna



---

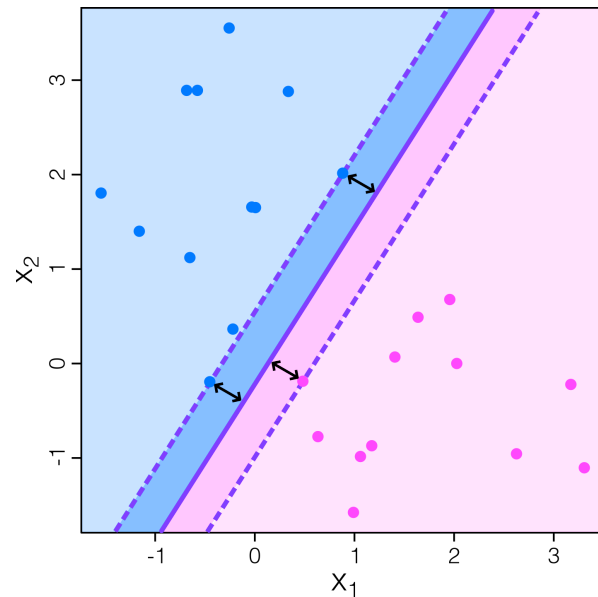
LO QUE VIMOS LA CLASE ANTERIOR...

---

# MAXIMAL MARGIN CLASSIFIER

En general si nuestra data es linealmente separable, puede existir un numero infinito de hiperplanos que van a funcionar

*Por lo que necesitamos algún criterio de selección.* El caso que aquí estamos en busca del hiperplano que más lejos se encuentra de los datos de entrenamiento.



El objetivo es buscar el hiperplano que mas grande posee este margen y, el algoritmo que hace esto es el **Maximal Margin Classifier**.

Podemos pensar que el clasificador busca el máximo **grosor** de recta que puede pasar entre las clases

---

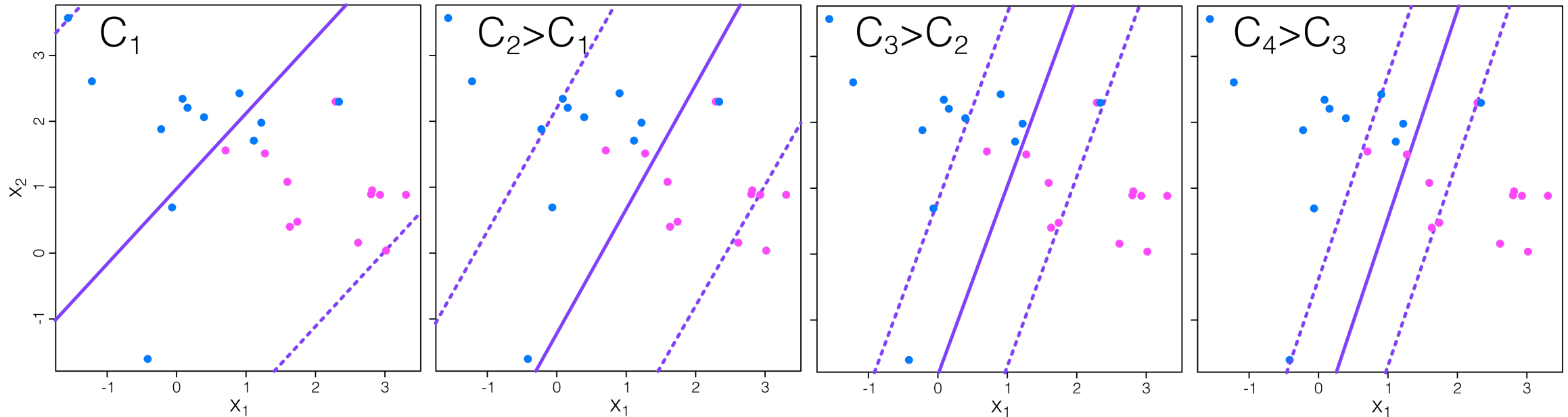
# CLASIFICADOR DE VECTOR DE SOPORTES

Por lo que vimos, si queremos seguir usando un hiperplano, debemos relajar las exigencias:

- Mayor robustez a observaciones individuales.
- Mejor clasificación de la **mayoría** (no todas) de las observaciones de entrenamiento.

Es decir, podría valer la pena clasificar **erróneamente algunas observaciones** de entrenamiento para poder clasificar mejor las observaciones restantes.

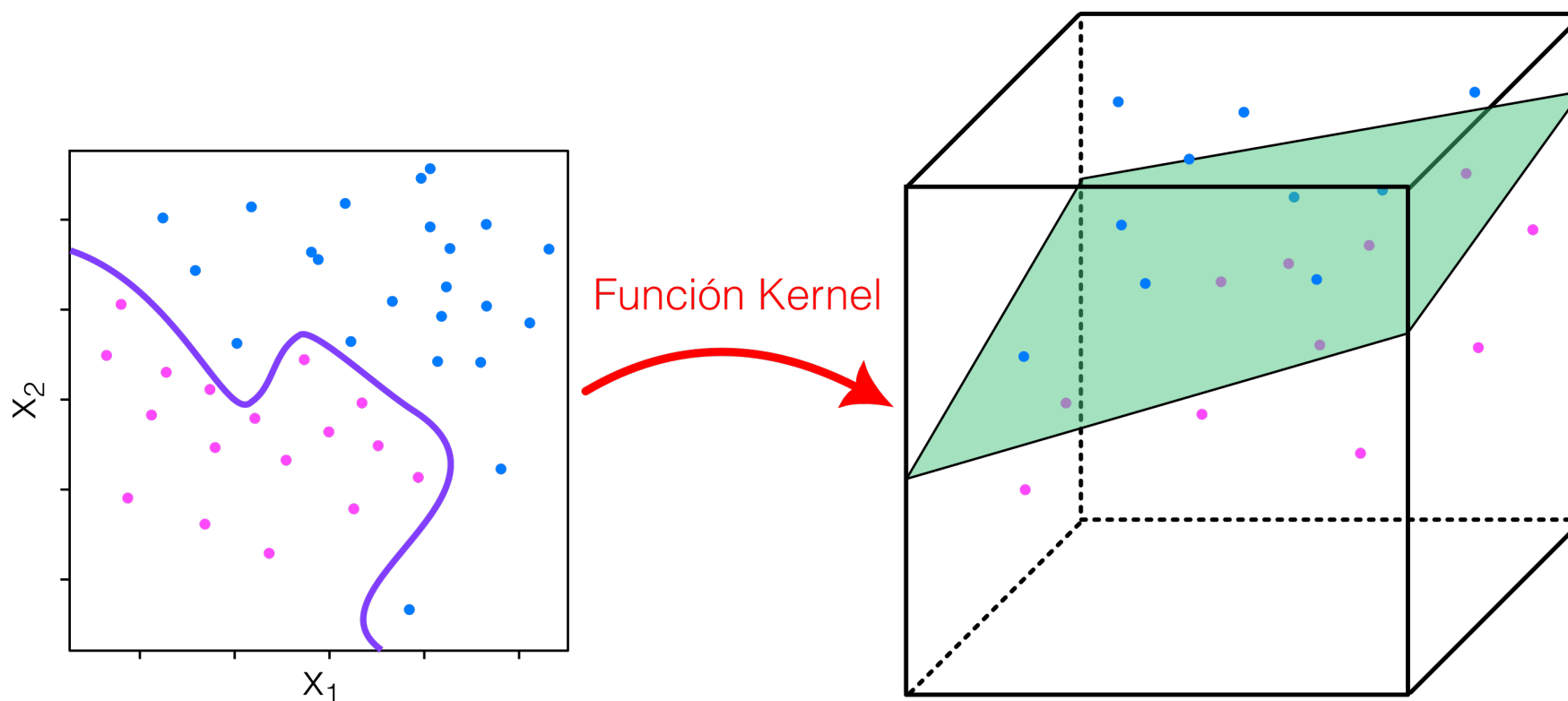
# CLASIFICADOR DE VECTOR DE SOPORTES



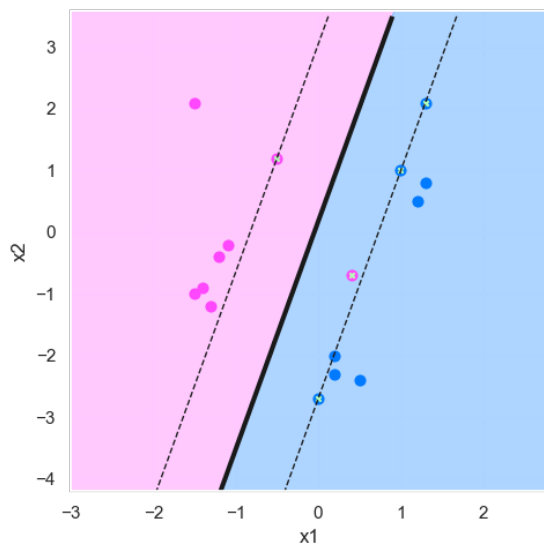
El valor de  $C$  nos da un trade-off (compensación) entre **sesgo** y **varianza**.

- Si  $C$  es chico, entonces el margen es grande y hay muchos vectores de soporte. Este clasificador tiene poca varianza, pero potencialmente poca exactitud.
- Si  $C$  es grande, hay menos vectores de soporte, y, por consiguiente, más varianza a expensa de una potencial mejor exactitud.

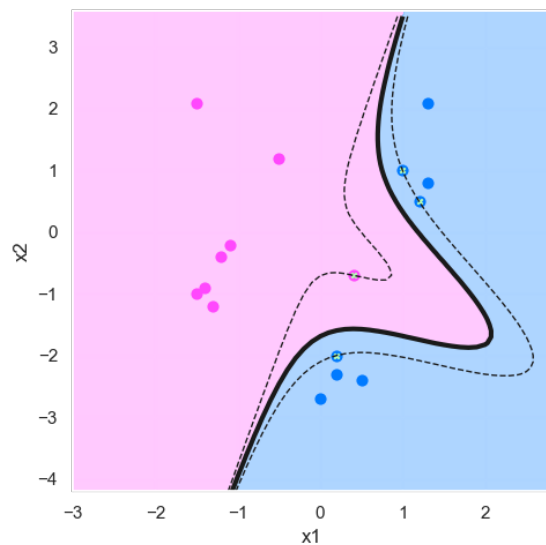
# SUPPORT VECTOR MACHINE



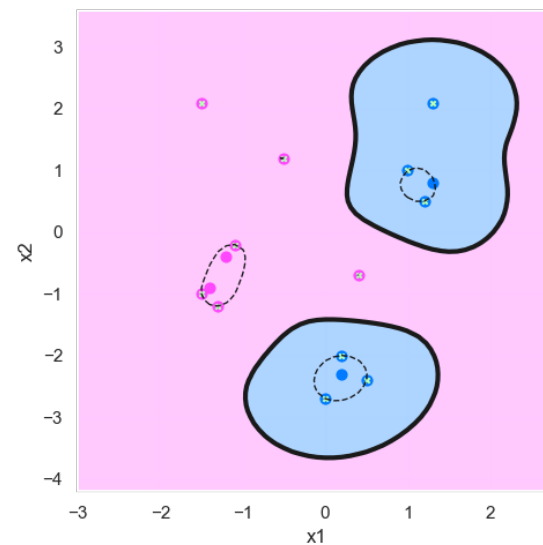
# SUPPORT VECTOR MACHINE



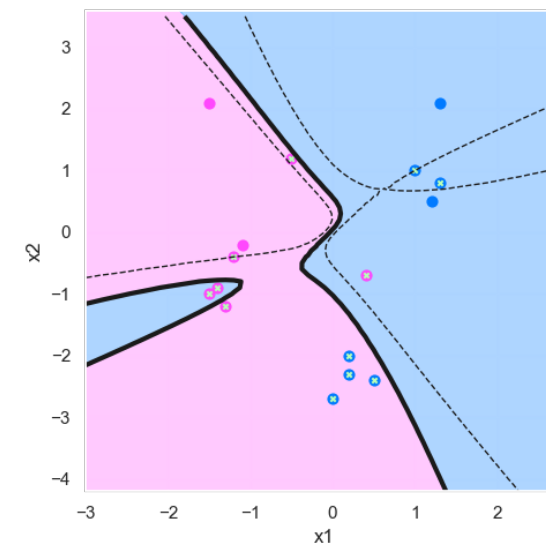
Lineales



Polinomial



Radial



Sigmoidea



---

# ÁRBOLES DE DECISIÓN



---

# ÁRBOLES DE DECISIÓN

Los árboles de clasificación y regresión, conocidos como CART (Classification and Regression Trees), son una poderosa técnica de aprendizaje automático que se utiliza ampliamente para resolver problemas tanto de clasificación como de regresión.

Los árboles CART son modelos de decisión que utilizan una estructura de árbol para realizar predicciones basadas en reglas **lógicas sencillas y fáciles de interpretar**.

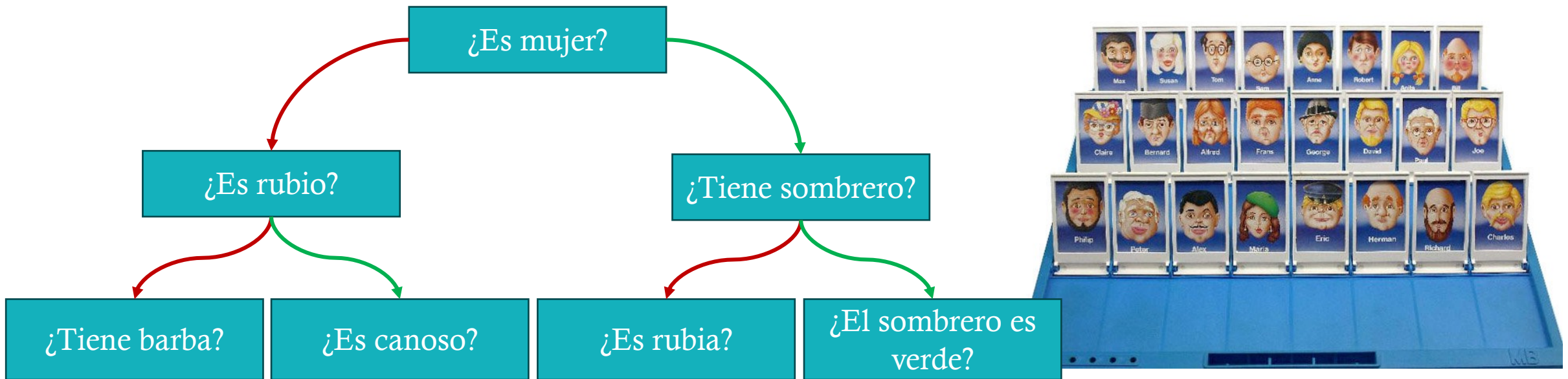
Árboles de clasificación

Árboles de regresión

# ÁRBOLES DE DECISIÓN

Los árboles de decisión son formas extremadamente intuitivas de clasificar objetos: simplemente hace una serie de preguntas diseñadas para acercarse en la clasificación.

Por ejemplo, si estás jugando al juego quien es quien, eligiendo preguntas binarias podés determinar quién es la persona que tiene tu contrincante...

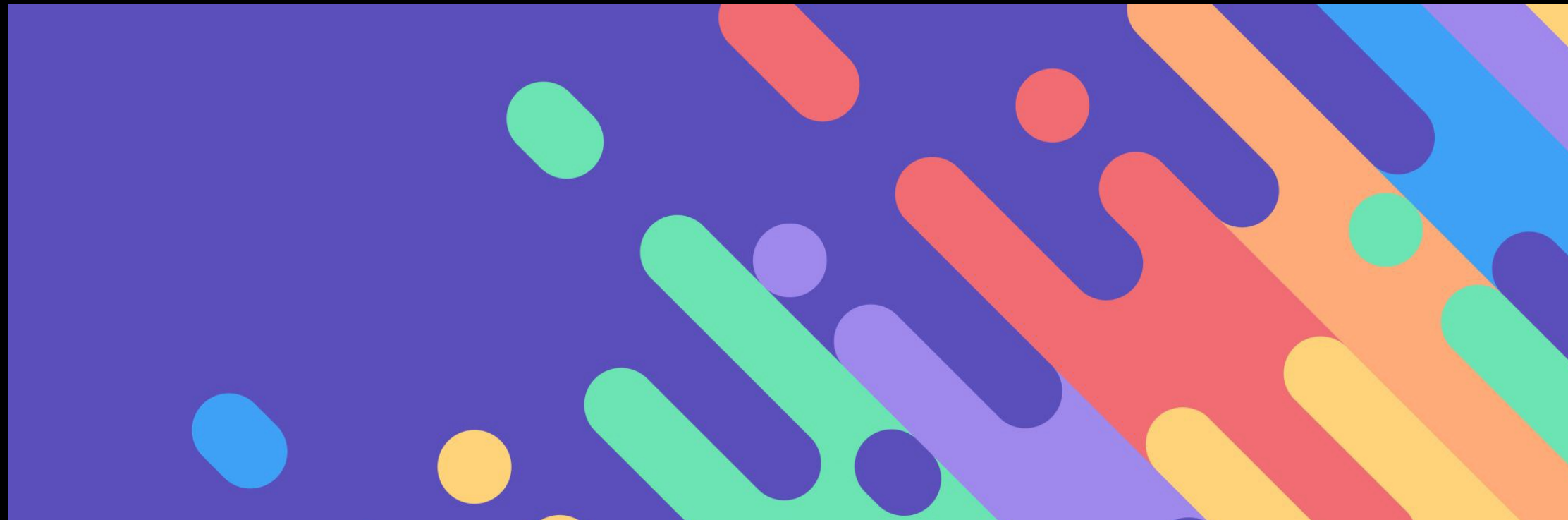


---

# ÁRBOLES DE DECISIÓN

La división binaria hace que esto sea extremadamente eficiente: en un árbol bien construido, cada pregunta reducirá el número de opciones a aproximadamente la mitad, reduciendo muy rápidamente las opciones incluso entre una gran cantidad de clases.

El truco, por supuesto, está en decidir qué preguntas hacer en cada paso.



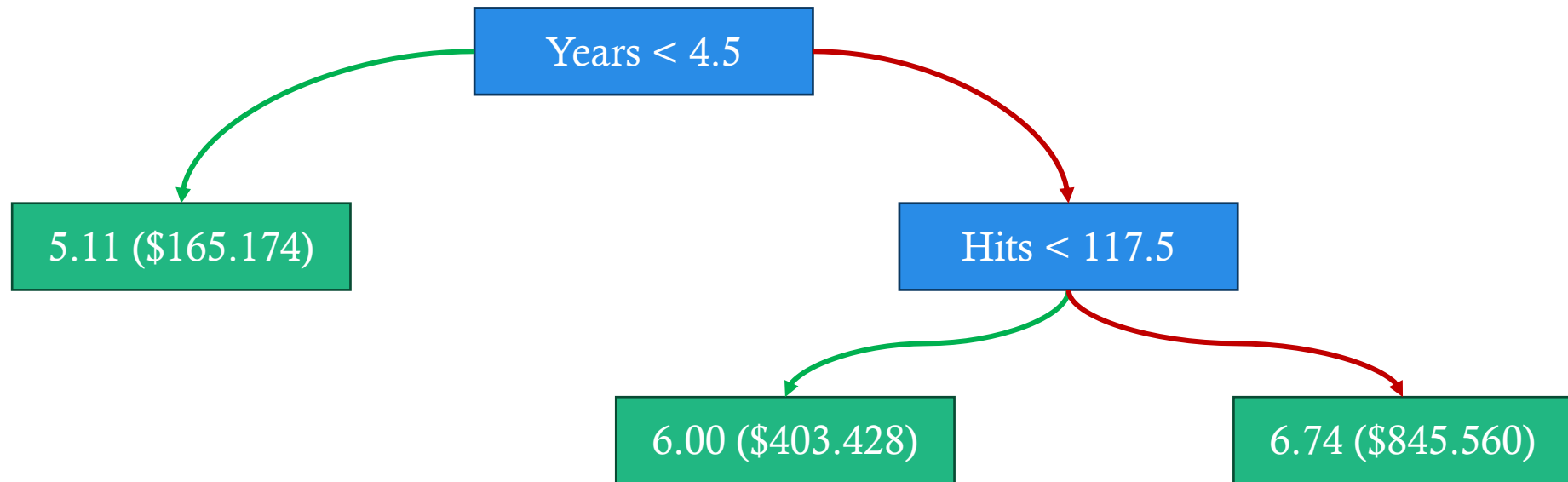
---

# ÁRBOL DE REGRESIÓN

---

# ÁRBOLES DE REGRESIÓN

Empecemos con un ejemplo sencillo, usando el dataset Hitters (Dato de salarios de jugadores de Beisbol de 1987 y estadísticas deportivas de 1986).



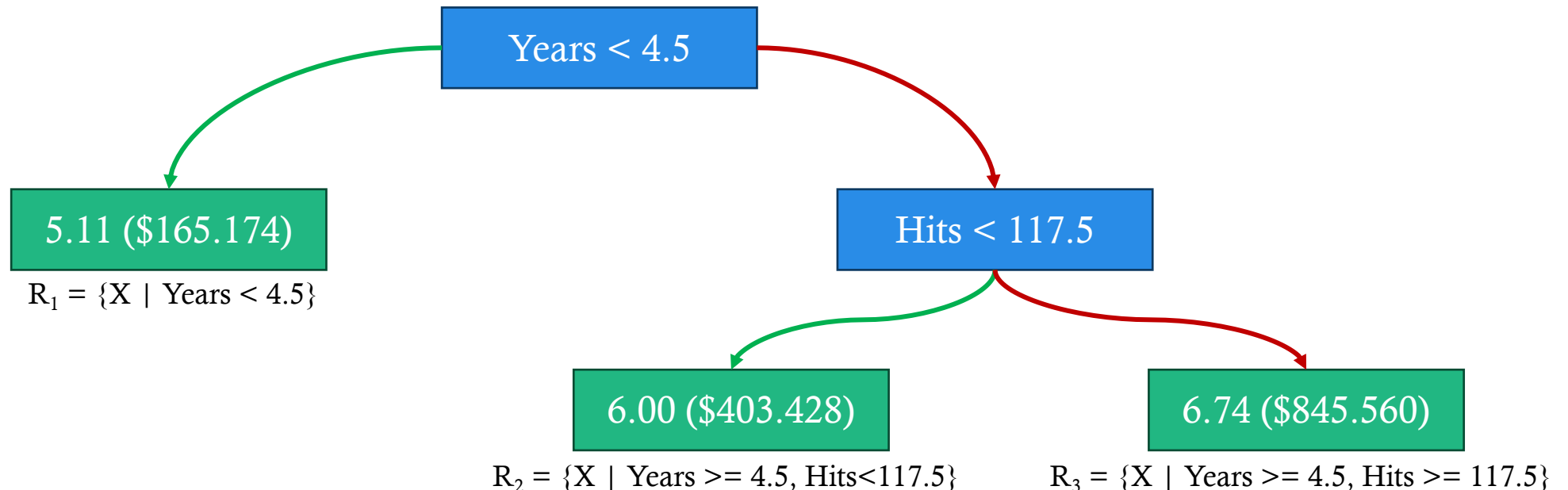
*Se predice el logaritmo del salario (tiene una distribución más de campana).*

***Years** es años jugando en las ligas. **Hits** es el número de hits que realizó el año pasado.*

---

# ÁRBOLES DE REGRESIÓN

Empecemos con un ejemplo sencillo, usando el dataset Hitters (Dato de salarios de jugadores de Beisbol de 1987 y estadísticas deportivas de 1986).

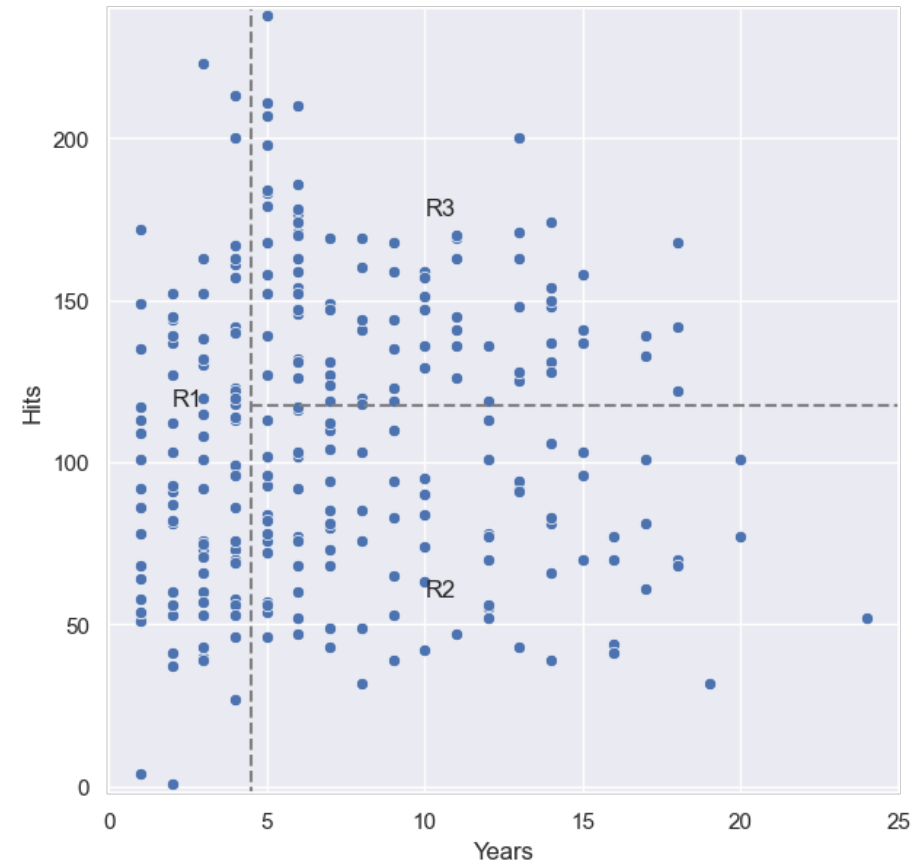


*Se predice el logaritmo del salario (tiene una distribución más de campana).*

***Years** es años jugando en las ligas. **Hits** es el número de hits que realizó el año pasado.*

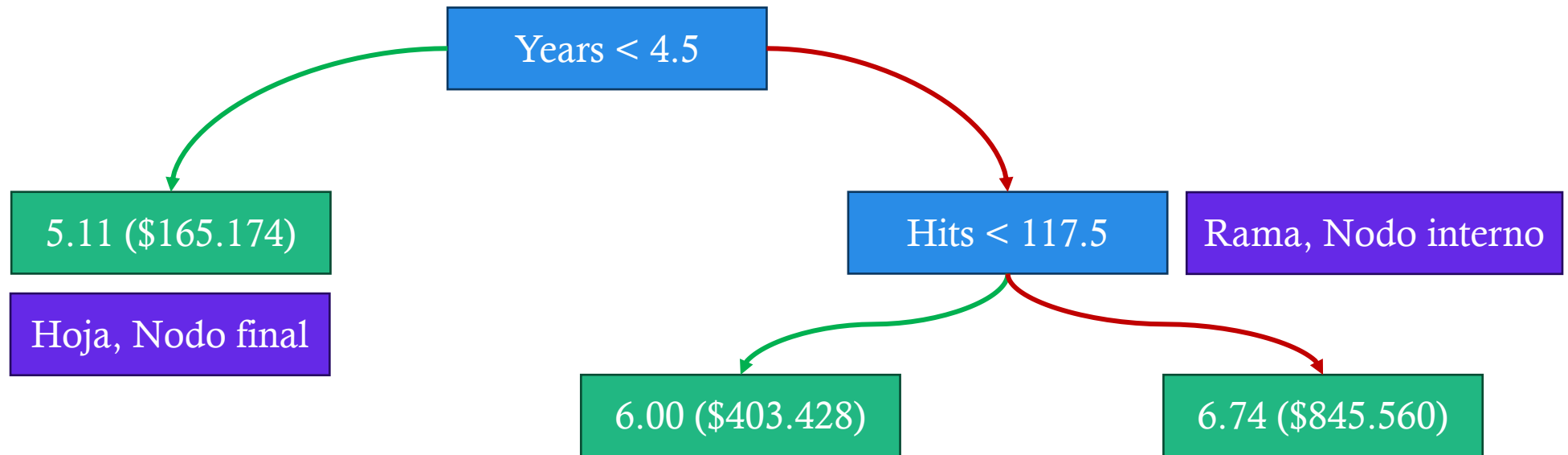
---

# ÁRBOLES DE REGRESIÓN



# ÁRBOLES DE REGRESIÓN

Empecemos con un ejemplo sencillo, usando el dataset Hitters (Dato de salarios de jugadores de Beisbol de 1987 y estadísticas deportivas de 1986).



*Este árbol de regresión es una sobre simplificación del verdadero valor de regresión entre **Salary**, **Years** e **Hits**. Sin embargo, tiene sus ventajas porque es más fácil entender y tienen mejor representación gráfica.*



---

# ÁRBOLES DE REGRESIÓN

Como construimos el proceso de construcción del árbol de regresión:

1. Dividimos el espacio de observaciones, que son el set de los valores posibles  $X_1, x_2, \dots, X_p$ , en  $J$  regiones distintas y que no se solapan  $R_1, R_2, \dots, R_J$ .
2. Para cada observación que cae en una región  $R_j$ , hacemos la misma predicción, la cual es simplemente la media de la respuesta de los valores de entrenamiento que están en  $R_j$ .

Podemos usar otra métrica de medición de posición central.

---

# ÁRBOLES DE REGRESIÓN

¿Cómo dividimos el espacio de observaciones?

En teoría, el espacio lo podríamos dividir en cualquier tipo de regiones, pero se elige espacios rectangulares para simplificar el modelo.

El objetivo es encontrar cajas  $R_1, \dots, R_J$  que minimice la suma al cuadrado de los residuos, dado por:

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

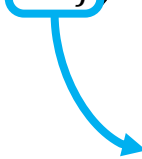
---

# ÁRBOLES DE REGRESIÓN

¿Cómo dividimos el espacio de observaciones?

En teoría, el espacio lo podríamos dividir en cualquier tipo de regiones, pero se elige espacios rectangulares para simplificar el modelo.

El objetivo es encontrar cajas  $R_1, \dots, R_J$  que minimice la suma al cuadrado de los residuos, dado por:

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


*Es la media de  $y$  en la región  $R_j$*

---

# ÁRBOLES DE REGRESIÓN

¿Cómo dividimos el espacio de observaciones?

Es imposible buscar todas las combinaciones posibles de valores para encontrar la minimizar a RSS.

Tenemos que usar algún algoritmo de optimización. Para ello tomamos un algoritmo **top-down greedy** que es conocido como **Recursive binary splitting**.

- **Top-down:** Arrancamos desde el tronco del árbol y vamos bajando.
- **Greedy:** En cada paso, se busca la mejor bifurcación en ese paso particular.

---

# ÁRBOLES DE REGRESIÓN

## Recursive binary splitting

Se elije un  $\mathbf{X}_j$  y el punto de corte  $s$  de tal forma que bifurca el espacio de features en dos regiones  $\{X|X_j < s\}$  y  $\{X|X_j \geq s\}$  que lleve a la mayor reducción de RSS.

Es decir, para cada valor de  $j$  y cada valor de  $s$ :

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

Y buscamos el valor de  $j$  y  $s$  que minimice esta ecuación:

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

---

# ÁRBOLES DE REGRESIÓN

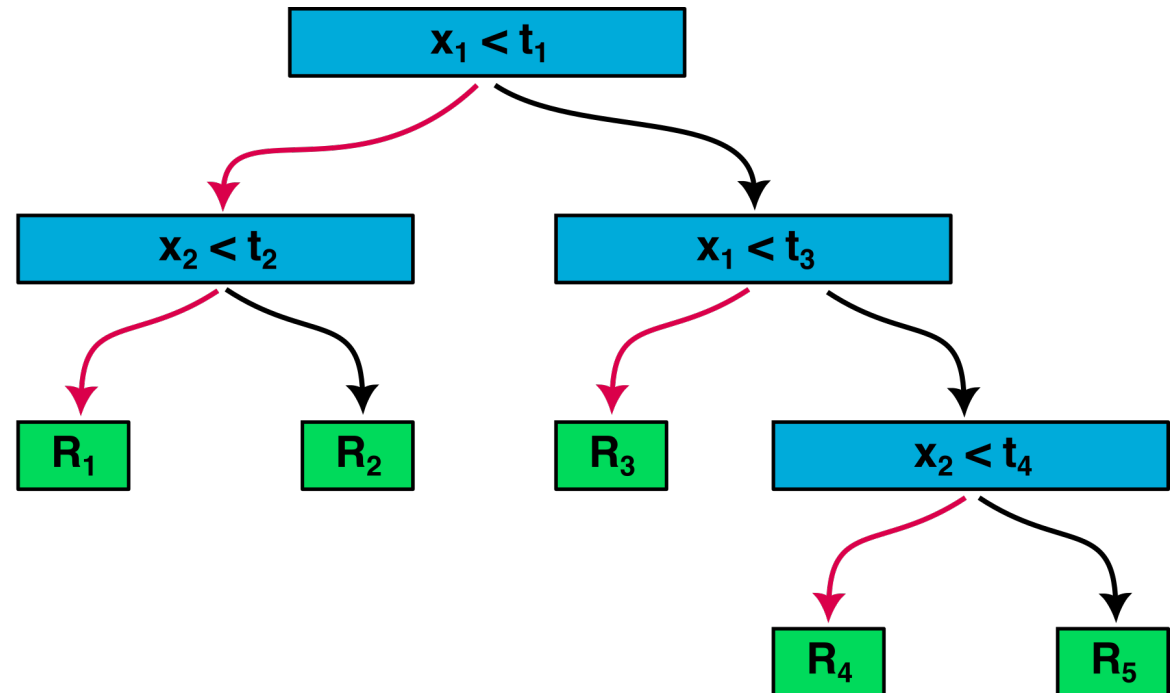
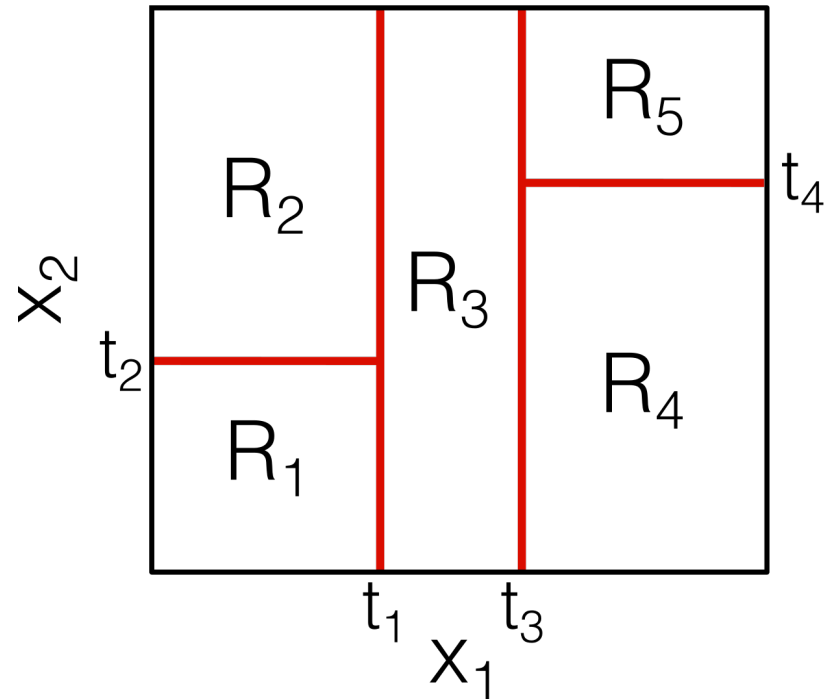
## Recursive binary splitting

Y recursivamente repetimos esto para los segmentos que se generan, pero ahora tomando a las regiones formadas y aplicando este proceso, partimos en nuevas regiones.

Este proceso continúa hasta que llegamos a un **criterio de corte**.

# ÁRBOLES DE REGRESIÓN

## Recursive binary splitting



---

# ÁRBOLES DE REGRESIÓN

## Recursive binary splitting

El proceso que se describió puede producir buenas predicciones del set de entrenamiento, pero muy fácilmente puede generar **overfitting**, haciendo que se desempeñe muy mal en el set de evaluación.

*El caso más extremo es un árbol con una hoja por cada punto del set de entrenamiento.*

Esto se debe a que el árbol es muy complejo. Un árbol más pequeño con menor regiones puede llevar a menos **varianza** y mejor interpretación a expensa de un poco de **sesgo**.

*Esto lo podemos resolver usando búsqueda de hiper-parámetros mediante validación cruzada, limitando la profundidad del árbol, cantidad mínima de muestras por hojas, etc.*





---

# ÁRBOL DE CLASIFICACIÓN

---

# ÁRBOL DE CLASIFICACIÓN

Un árbol de clasificación es muy similar a uno de regresión, pero ahora se usa para predecir una **variable cualitativa**.

En el caso de regresión, al llegar la hoja, obteníamos el valor con el promedio de los valores en la hoja. Ahora, obtenemos la clase en base a la clase que más ocurre en las muestras que están en la hoja.

Al interpretar los resultados de un **árbol de clasificación**, a menudo estamos interesados no sólo en la predicción de clase correspondiente a una región de nodo terminal particular, sino también en las **proporciones de clase entre las observaciones de entrenamiento** que caen en esa región.

---

# ÁRBOL DE CLASIFICACIÓN

La forma más simple en que se crea un árbol de clasificación es muy parecida al árbol de regresión con la estrategia **top-down** y **greedy**, pero no contamos con el error cuadrático.

Una primera métrica que podemos usar es la tasa de error de clasificación:

$$E = 1 - \max_k(\hat{p}_{mk})$$

Es la fracción de las observaciones de entrenamiento en esa región que no pertenecen a la clase más común.

$\hat{p}_{mk}$  representa la proporción de las observaciones de entrenamiento en la región **m** que son de la clase **k**.

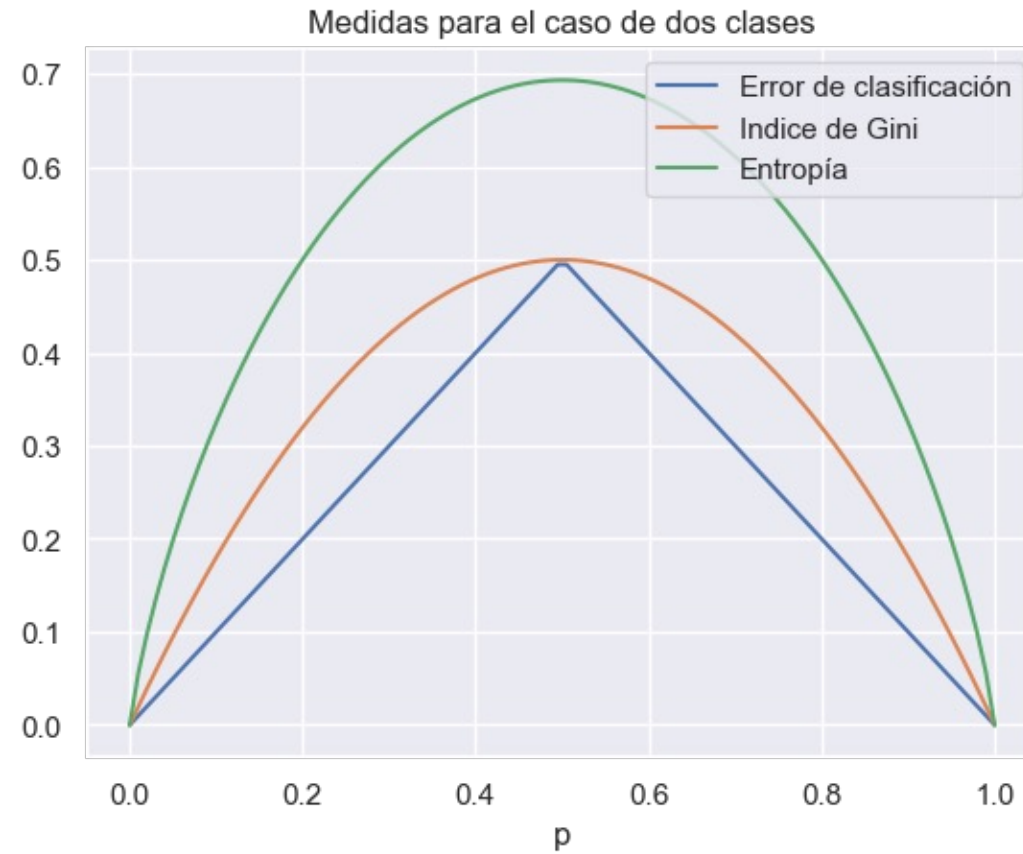
Ojo, el error de clasificación **no es lo suficientemente sensible para hacer crecer a los árboles** y, en la práctica, son preferibles otras dos medidas.

---

# ÁRBOL DE CLASIFICACIÓN







- **Índice de Gini:** Es una medida de la desigualdad usada inicialmente para medir la desigualdad de los países.
- **Entropía:** Mide cuanta información transmite cuando un árbol separa una rama. Si en una hoja, todas las observaciones de entrenamiento son de una sola clase, la entropía es cero. En cambio, si las clases están desperdigadas de forma uniforme entre la clase, la entropía es grande.

# ÁRBOL DE CLASIFICACIÓN



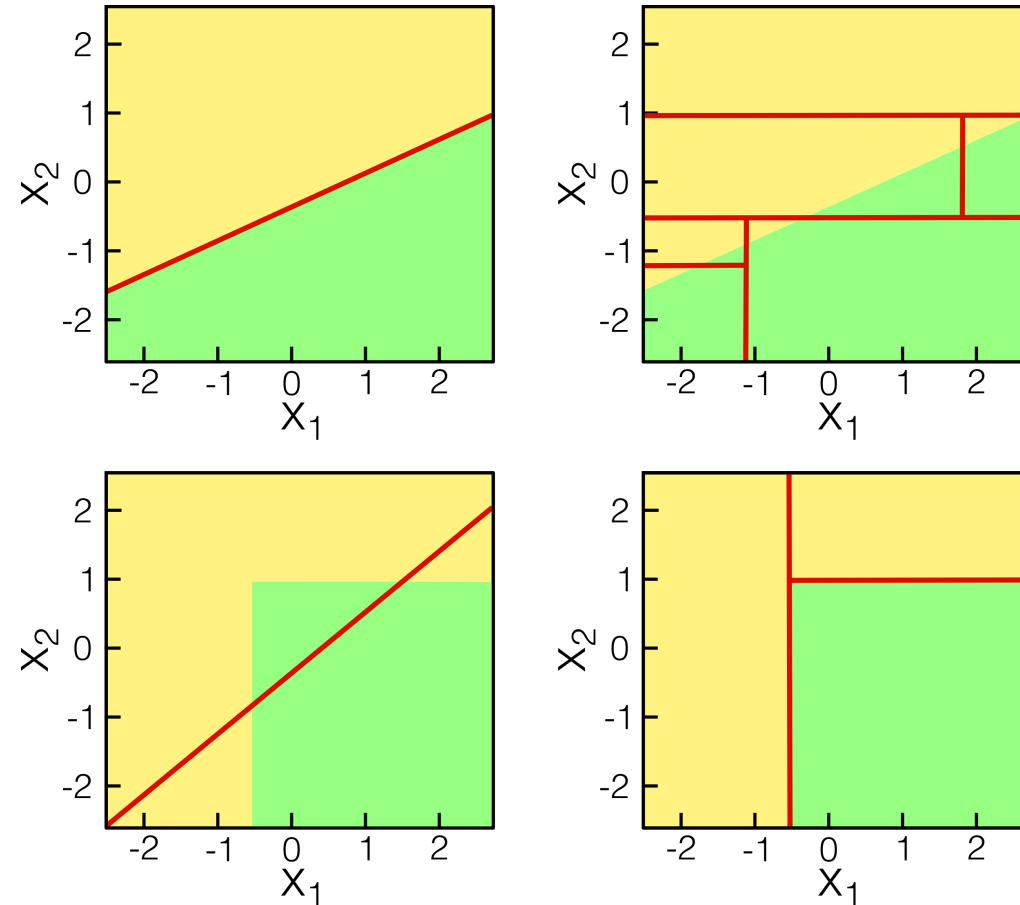
---

# VENTAJAS Y DESVENTAJAS DE LOS ÁRBOLES

-  Son fáciles de explicar a las personas, más inclusive que a la regresión lineal.
-  Se hipotetiza que el árbol de decisión se acerca más a la forma que un humano piensa.
-  Se puede representar gráficamente.
-  Los árboles puede manejar fácilmente variables cualitativas sin necesidad de crear variables dummy.
-  No tienen el mismo nivel de exactitud de predicción que otros modelos
-  No son robustos, pequeños cambios en los datos pueden cambiar grandes cambios en la predicción.

---

# VENTAJAS Y DESVENTAJAS DE LOS ÁRBOLES





---

# BOSQUES ALEATORIOS



---

# BOSQUES ALEATORIOS

Es común que los árboles sobre-ajusten, dado que tan exacto tienden a adaptarse a los datos de entrenamiento,.

Una forma de evitar esto es mediante **bosques aleatorios**, en el cual se construyen múltiples arboles de decisión y combinar sus salidas.

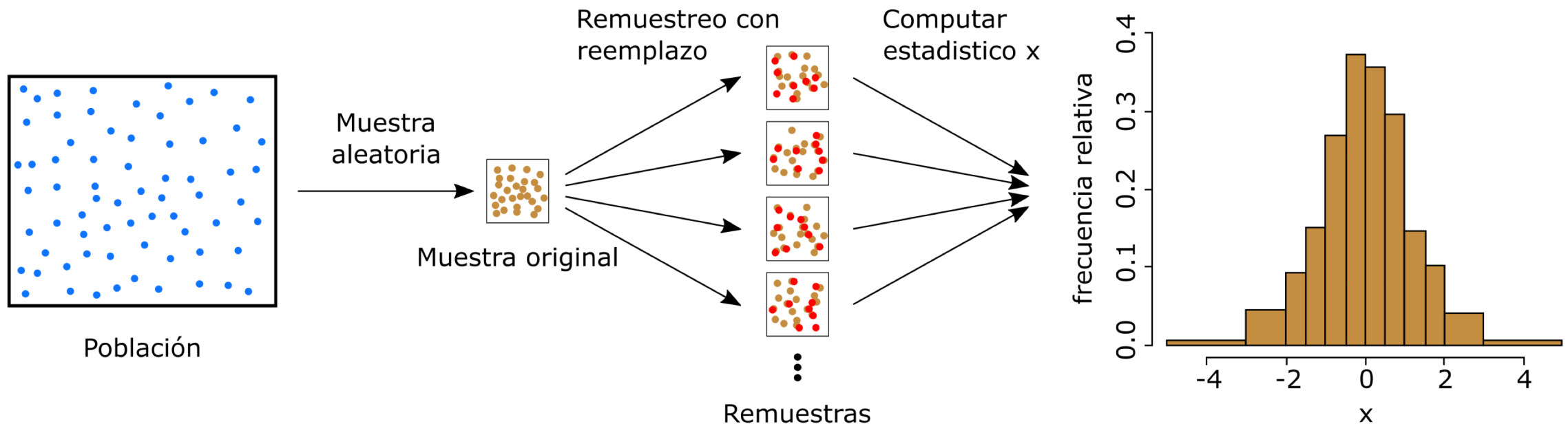
Si son de **clasificación**, estos árboles votan la clase.

Si son de **regresión**, se promedia las predicciones.

# BOSQUES ALEATORIOS

¿Como se construyen aleatoriamente estos árboles?

Una forma es usando **bootstrapping**



---

# BOSQUES ALEATORIOS

¿Como se construyen aleatoriamente estos árboles?

## Bagging

En el caso de los árboles, en vez de entrenar a los árboles con todas las observaciones del set de entrenamiento, se arma un nuevo set para cada árbol, usando **bootstrapping**.

Dado que cada árbol es entrenado diferente, va a ser diferente de los demás árboles.

En general estos árboles son profundos, es decir gran **varianza**, pero poco **sesgo**. Al promediar las salidas, en regresión, reducimos la varianza.

Bagging mejora sustancialmente los resultados cuando se lleva a cientos o miles de árboles.

---

# BOSQUES ALEATORIOS

¿Como se construyen aleatoriamente estos árboles?

## Bosques aleatorios

Los bosques aleatorios son una mejora de los obtenidos mediante **bagging**.

Los bosques aleatorios hacen lo mismo que **bagging**, pero además se usa una cantidad aleatoria de atributos. El valor de cuantos atributos a usar se elige aproximadamente la raíz cuadrada de la cantidad de atributos totales.

Por ejemplo, si tenemos  $p=13$  atributos, se usarán  $m=4$ , y en cada árbol será una combinación al azar diferente.

---

# BOSQUES ALEATORIOS

El razonamiento de esto es:

Supongamos que hay un atributo que es muy fuerte predictor, junto a otros atributos moderadamente fuertes. Si aplicamos **Bagging**, todos estos árboles siempre tenderán a usar el atributo fuerte en la bifurcación inicial. Por consiguiente, todos serán parecidos y habrá una fuerte correlación entre árboles, quitando la posibilidad de reducir la varianza.

**Bosques aleatorios**, al separar los atributos, en promedio  $(p-m)/p$  de las particiones no van a considerar al atributo fuerte, quitando la correlación.

---

# BOSQUES ALEATORIOS

Por último, nos queda **Boosting**

La idea es similar a la de **Bagging**, pero en vez de construir arboles aleatoriamente dado por el proceso de **bootstrapping**, los nuevos árboles que se construyen usando la información de árboles anteriores.

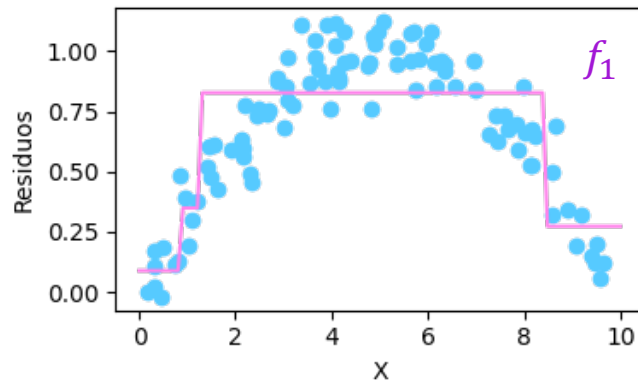
El truco está en que se entrena un árbol, se calcula los residuos, y esos residuos pasan a ser la variable predictora del siguiente árbol. Los árboles elegidos son chicos con unos pocos nodos.

Esto es un proceso de aprendizaje lento que depende de árboles anteriores. Un proceso lento de aprendizaje estadísticamente lleva a buenos entrenamiento.

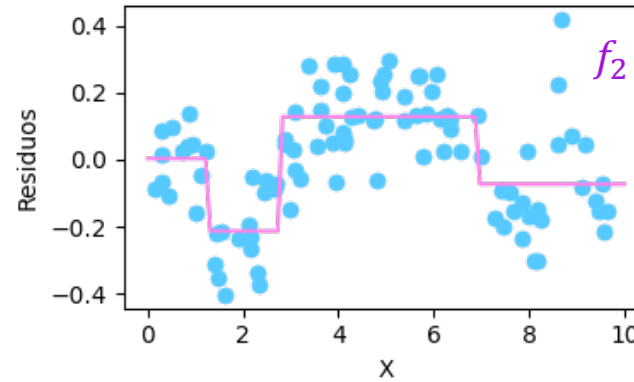
Una implementación famosa es **XGBoost**.

# BOSQUES ALEATORIOS

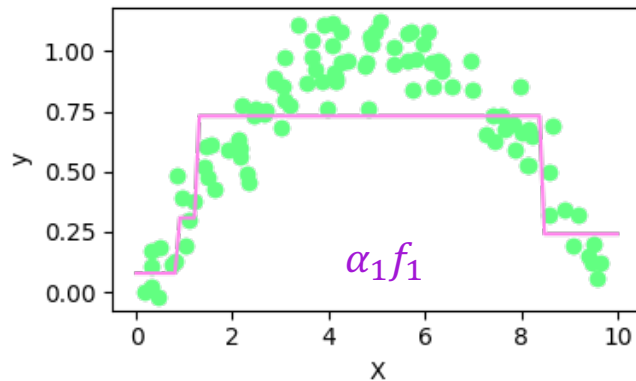
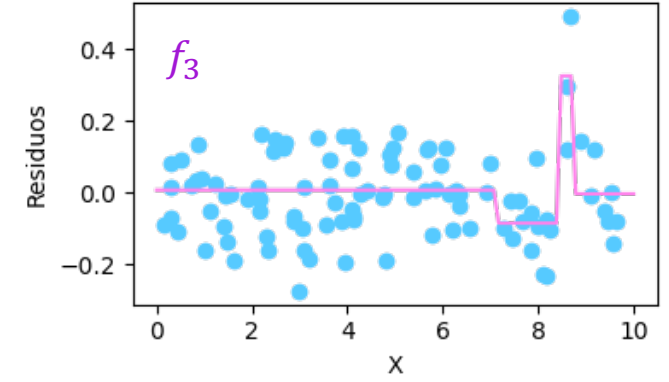
Iteración 1



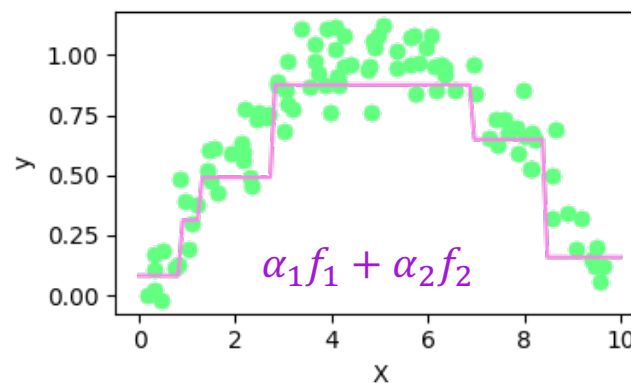
Iteración 2



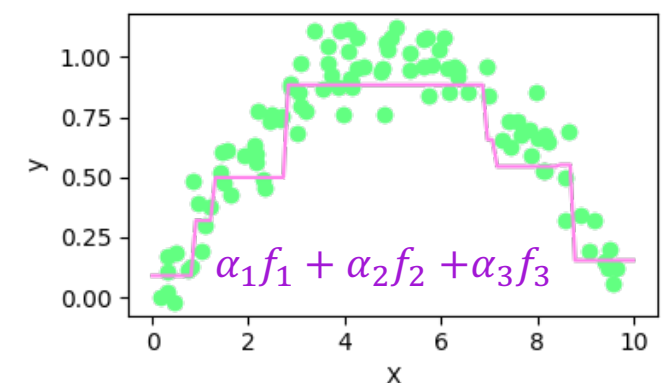
Iteración 3



Salida de ensamble:  $F = \alpha_1 f_1$



Salida de ensamble:  $F = \alpha_1 f_1 + \alpha_2 f_2$



Salida de ensamble:  $F = \alpha_1 f_1 + \alpha_2 f_2 + \alpha_3 f_3$