



# Árboles de decisión

Laboratorio de Datos 1°C 2022

# Problema

Diagram illustrating the data structure for the problem, showing Features and Target columns.

**Features** (Cielo, Humedad) and **Target** (Tenis?) are shown. The data is organized into **Instancias** (Instances).

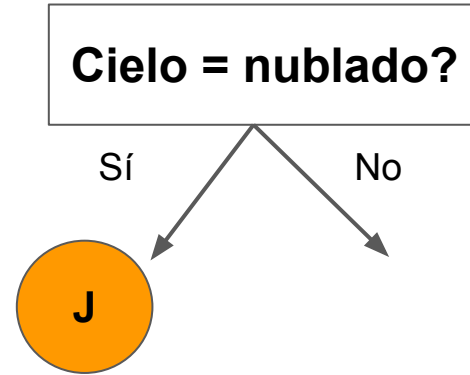
	Cielo	Humedad	Tenis?
Instancias	Sol	Alta	No
	Sol	Alta	No
	Nublado	Alta	Sí
	Sol	Alta	No
	Sol	Normal	Sí
	Nublado	Alta	Sí
	Nublado	Normal	Sí

Queremos predecir si una persona va a jugar al tenis en base a datos del clima y situaciones anteriores.



¿Cómo empezamos? Buscamos asociaciones entre los atributos y lo que queremos predecir.

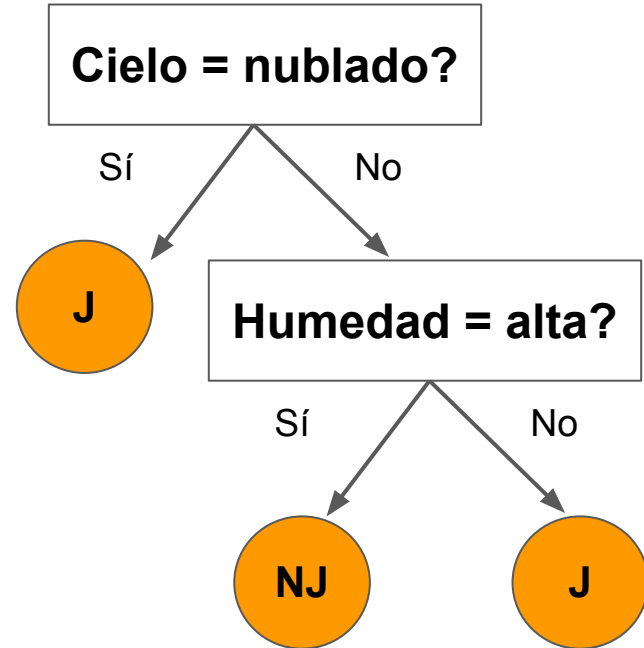
Cielo	Humedad	Tenis?
Sol	Alta	No
Sol	Alta	No
Nublado	Alta	Sí
Sol	Alta	No
Sol	Normal	Sí
Nublado	Alta	Sí
Nublado	Normal	Sí



J: juega; NJ: no juega

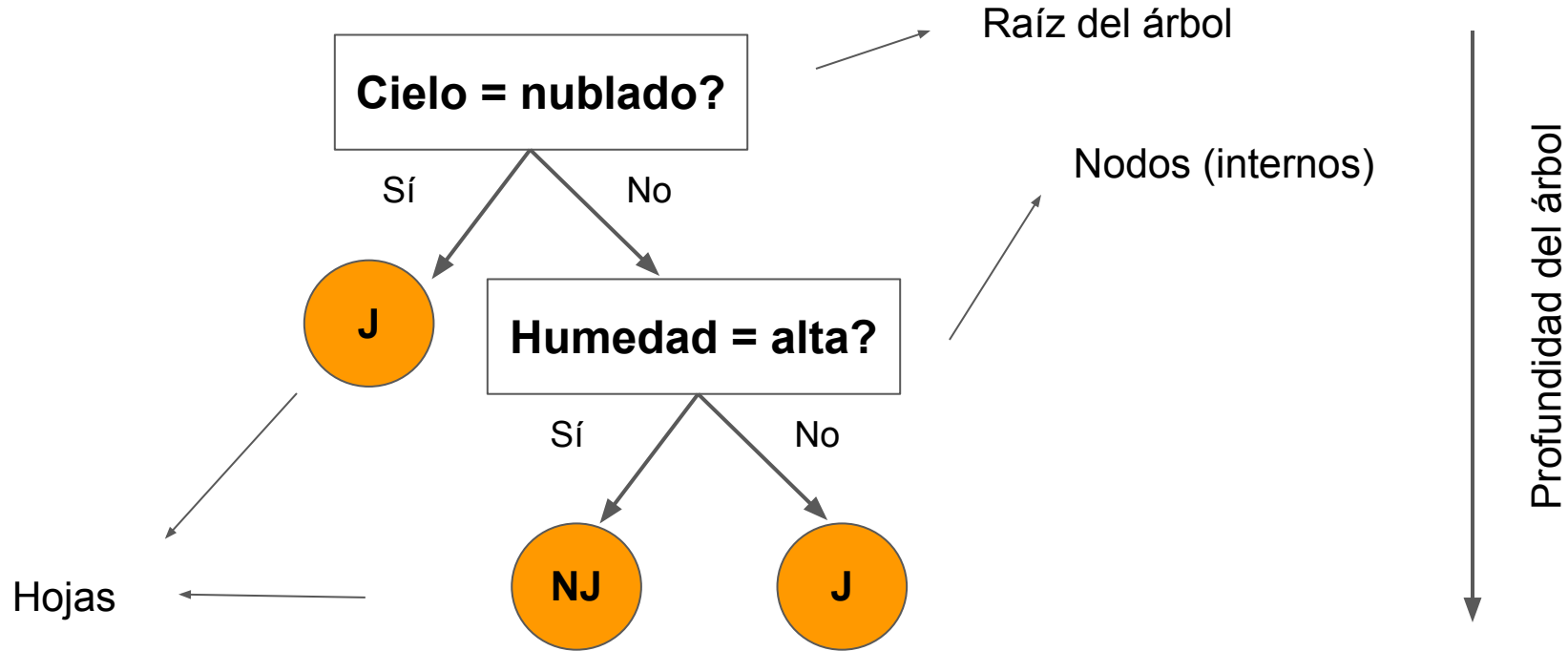
¿Cómo empezamos? Buscamos asociaciones entre los atributos y lo que queremos predecir.

Cielo	Humedad	Tenis?
Sol	Alta	No
Sol	Alta	No
Nublado	Alta	Sí
Sol	Alta	No
Sol	Normal	Sí
Nublado	Alta	Sí
Nublado	Normal	Sí



J: juega; NJ: no juega

# Felicitaciones! Construyó su primer árbol de decisión



# La cosa se puede complicar un cacho...

Base de datos multidimensional y con variables de distintas categorías (numérica, ordinal, nominal):

	M2	AMBIENTES	ANTIGUEDAD	BAÑOS	LATITUD	LONGITUD	COMUNA	DOLARES	log10_DOLARES
0	81	3	4	1	-34.581078	-58.449433	COMUNA 13	225000	5.352183
1	69	3	20	1	-34.623129	-58.439338	COMUNA 06	140000	5.146128
2	75	3	20	1	-34.604972	-58.421278	COMUNA 05	154000	5.187521
3	42	2	40	1	-34.604725	-58.399524	COMUNA 03	75000	4.875061
4	90	3	1	1	-34.623390	-58.504401	COMUNA 10	149900	5.175802

Podemos querer predecir precio de inmuebles en base a sus características o a qué zona pertenece: podemos usar los árboles como para regresión como para clasificación.

**¿Por cuál feature arrancamos?**

Problema desarrollado en el colab...

# ¿Cómo construimos un árbol de decisión?

- Seleccionamos un feature muy informativo y le preguntamos si cumple o no una dada condición.
- De aquí se desprenden dos caminos: si se cumple o no la condición de arriba.
- Para cada “hijo” volvemos a seleccionar un feature informativo y una condición para que cumpla.
- Podemos seguir así hasta eventualmente llegar a un árbol con tantas hojas como datos en mi dataset.

¿Qué es un feature informativo? ¿Cómo elegimos la condición que le preguntamos si cumple o no?

# ¿Qué es un feature informativo? ¿Cómo elegimos la condición que le preguntamos si cumple o no?

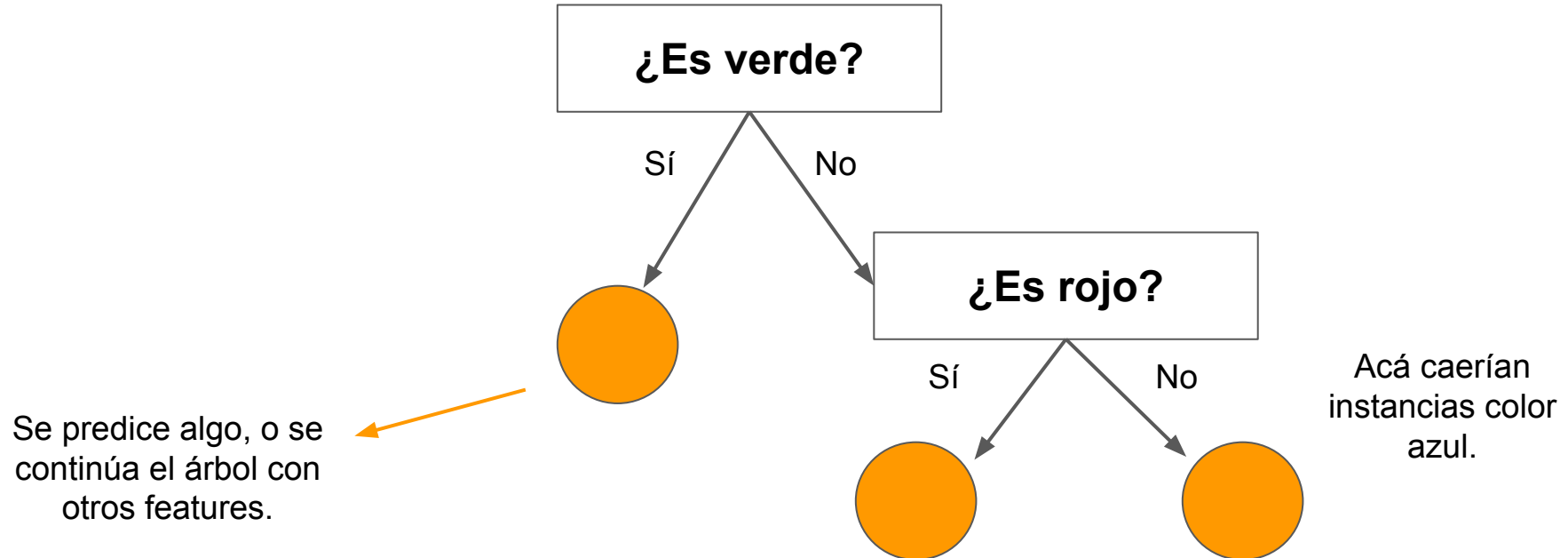
Depende de si es un problema de regresión o clasificación:

- **Clasificación:** buscamos que un nodo me separe los datos en grupos lo más “puros” posibles, es decir, que no haya tanta mezcla de las clases.
- **Regresión:** buscamos nodos que separen los datos en grupos donde el error de predicción sea bajo.



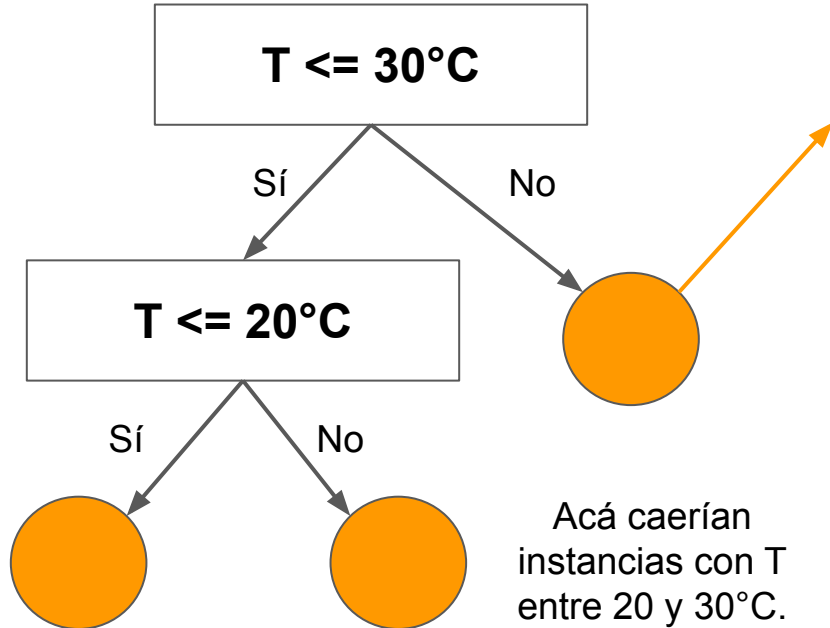
# Antes... ¿cómo tratamos variables categóricas?

Supongamos una variable categórica: rojo, azul, verde. De los nodos se desprende si se cumple o no la variable categórica.



# ¿Cómo tratamos variables numéricas?

Supongamos una variable numérica como la temperatura  $T$ . Podemos elegir cortes en cualquier parte del rango de la variable.

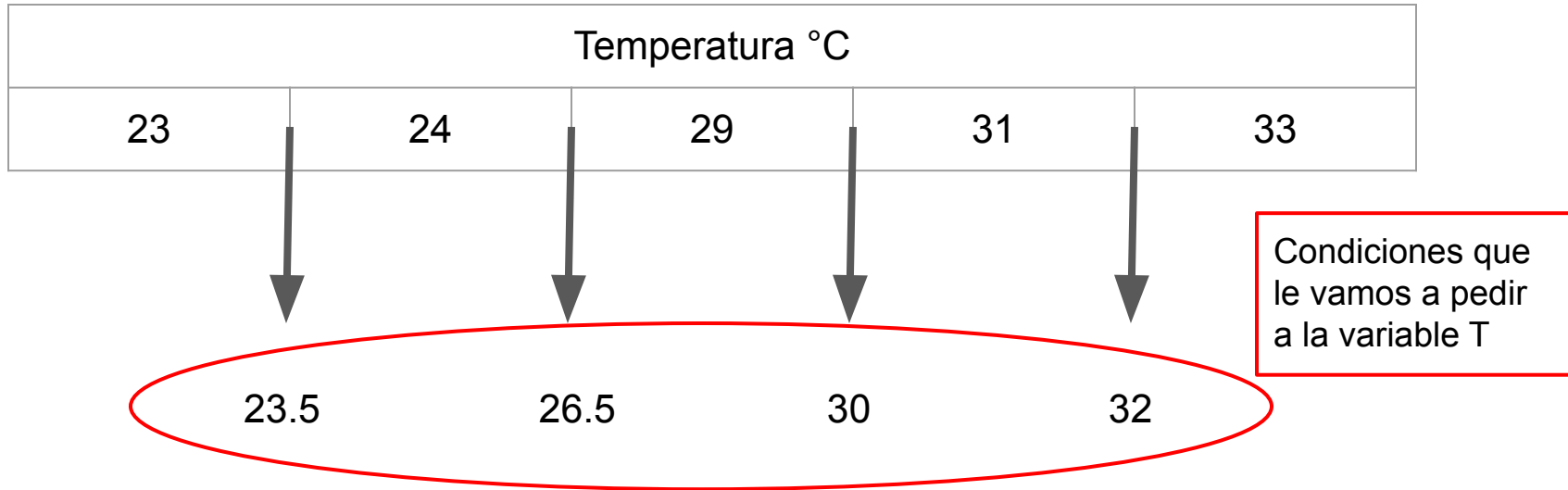


Se predice algo, o se continúa el árbol con otros features y bien subdividiendo con la misma variable.

Acá caerían instancias con  $T$  entre 20 y  $30^\circ\text{C}$ .

# ¿Cómo tratamos variables numéricas?

Cuando la variable es numérica, sólo consideramos valores de corte que efectivamente separen nuestros datos:



# Problemas de clasificación

Tenemos una variable categórica que queremos predecir en base a un conjunto de features. ¿Qué feature y condición elijo?

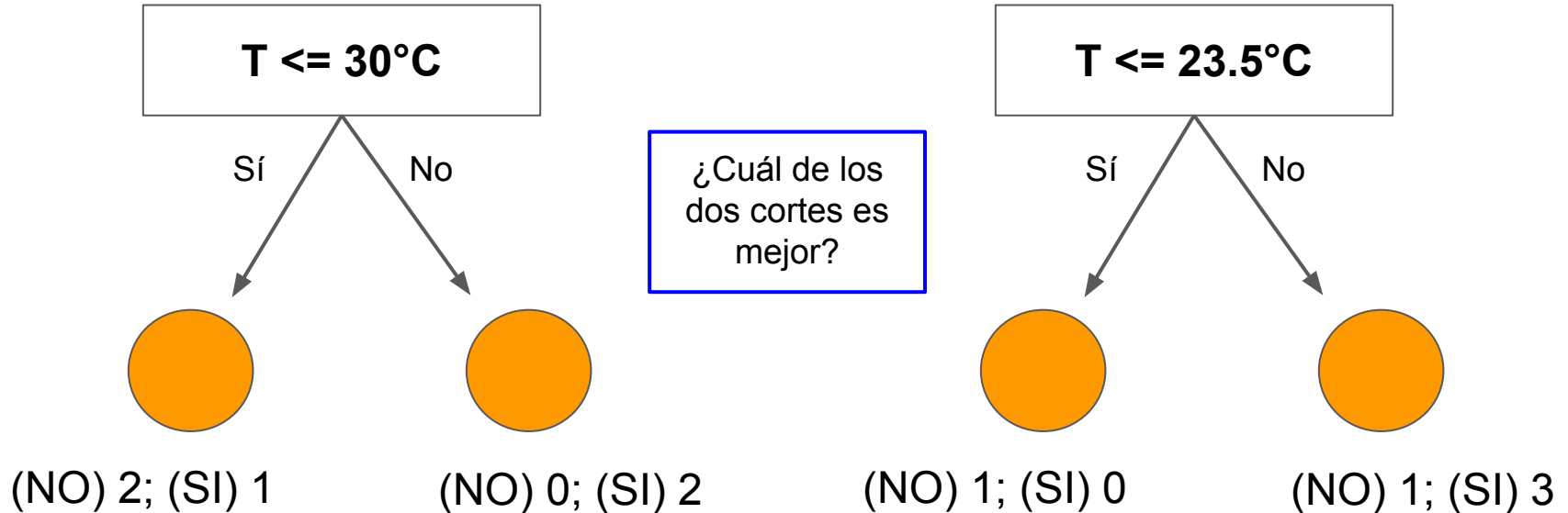
Temperatura °C	Llovió
31	SI
24	NO
29	SI
33	SI
23	NO



Puedo elegir cualquiera de las 4 condiciones de la diapositiva anterior


# Problemas de clasificación

Tenemos una variable categórica que queremos predecir en base a un conjunto de features. ¿Qué feature y condición elijo?



# Medidas de impureza

Proporción de los datos que están en la hoja  $m$  y pertenecen a la clase  $k$ .



Medidas de impureza dentro de cada hoja:

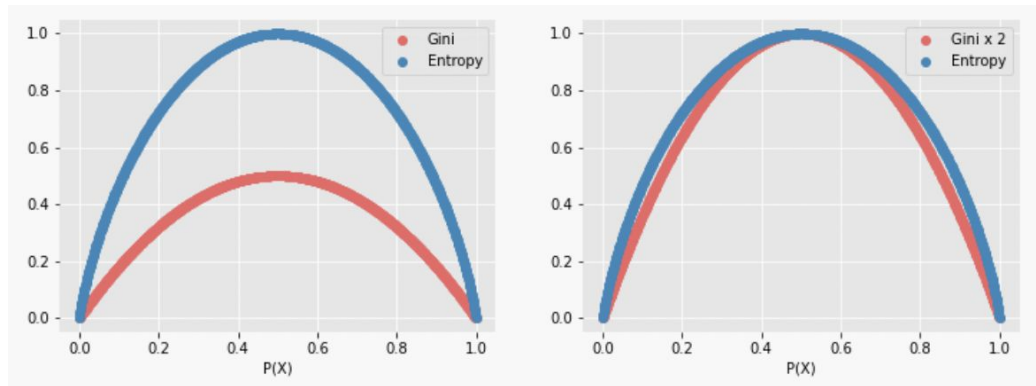
Coeficiente de Gini: 
$$G = \sum_{k=1}^K \hat{p}_{mk}(1 - \hat{p}_{mk})$$

Entropía: 
$$D = - \sum_{k=1}^K \hat{p}_{mk} \log \hat{p}_{mk}$$

Si todos los datos dentro de una hoja pertenecen a la misma clase,  $G = D = 0$ : la hoja tiene impureza 0.

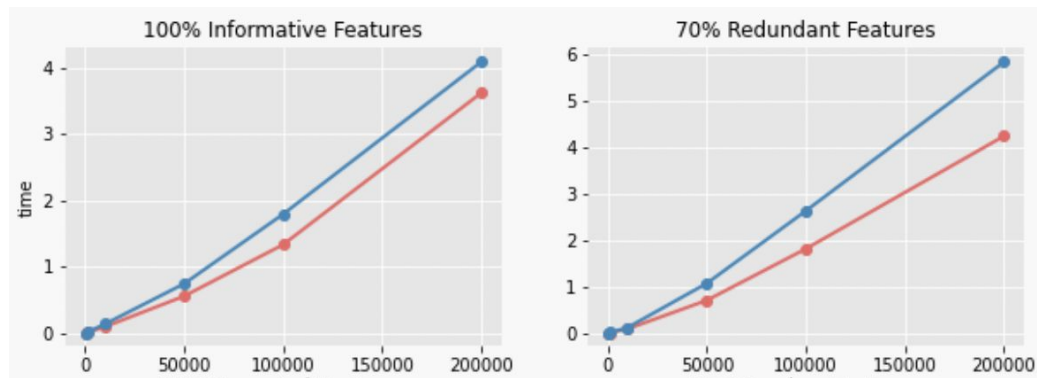
Se define la impureza de un árbol por el **promedio pesado de las impurezas de cada hoja**, pesado por la fracción de datos en cada hoja.

# Medidas de impureza



No hay muchas diferencias entre Gini y entropía cuando se re-escalean a  $[0,1]$ .

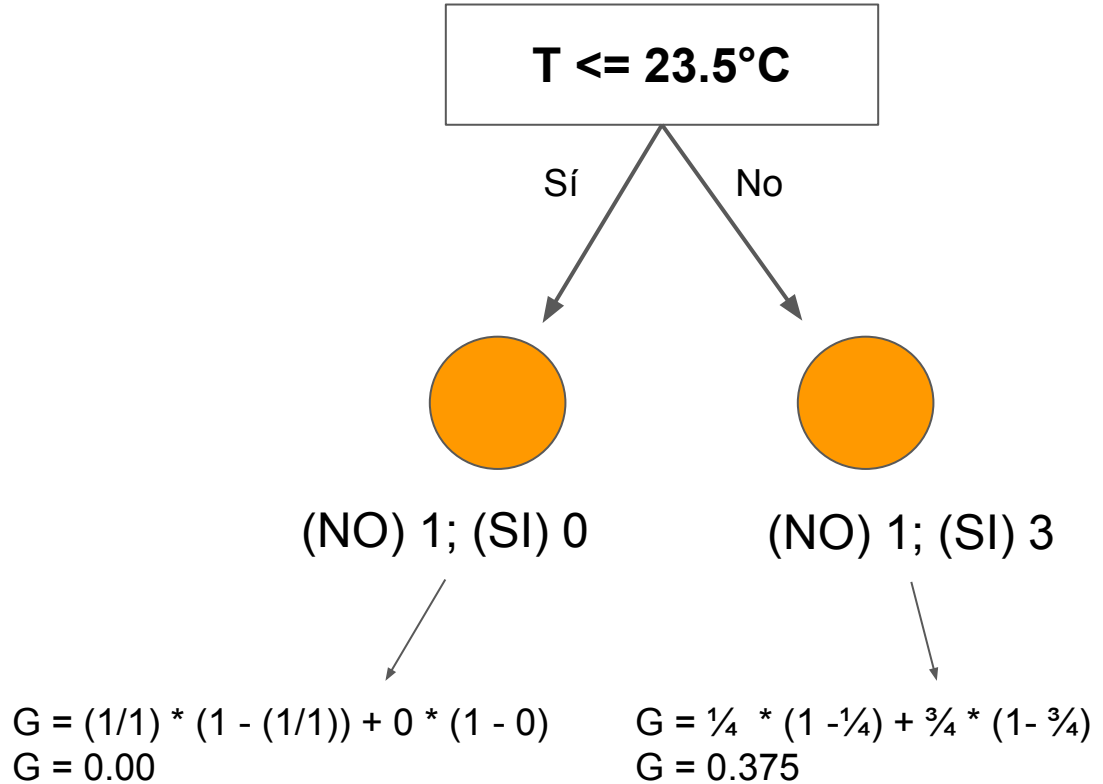
Pero el cálculo de entropía en general es más lento (logaritmos)



# Medidas de impureza

Ejemplo:

Temperatura °C	Llovió
31	SI
24	NO
29	SI
33	SI
23	NO



$$G \text{ del árbol} = \frac{1}{5} * 0 + \frac{4}{5} * 0.375 = 0.3$$

Fracción de los datos en la rama de la derecha

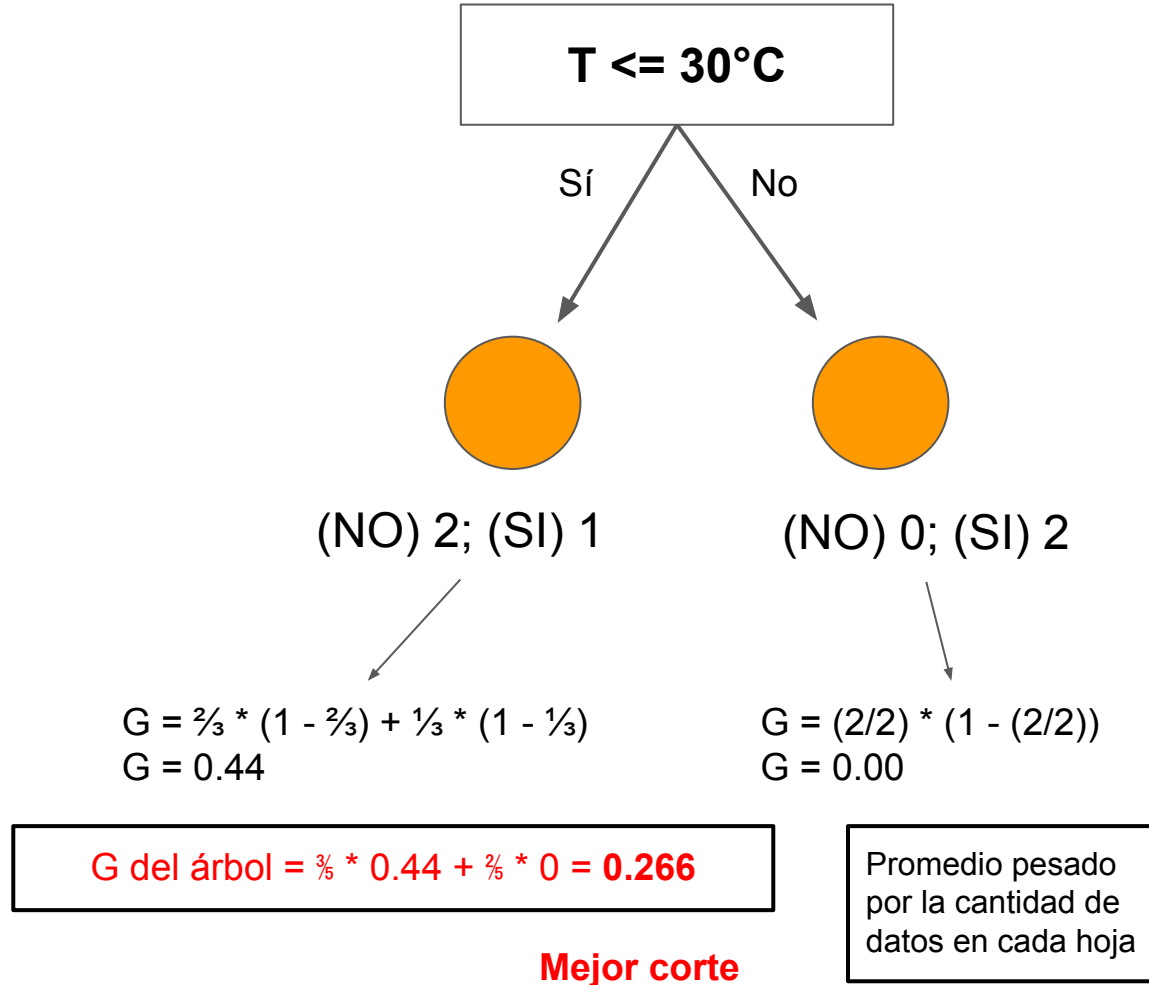
Promedio pesado  
por la cantidad de  
datos en cada hoja



# Medidas de impureza

Ejemplo:

Temperatura °C	Llovió
31	SI
24	NO
29	SI
33	SI
23	NO

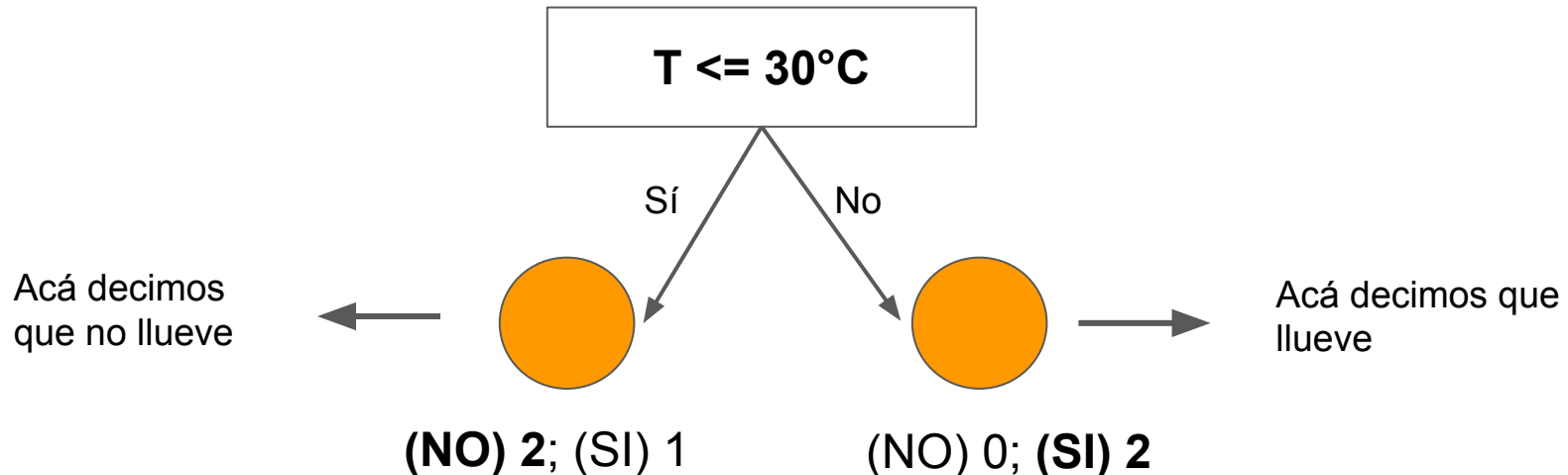


# Problemas de clasificación. Algoritmo

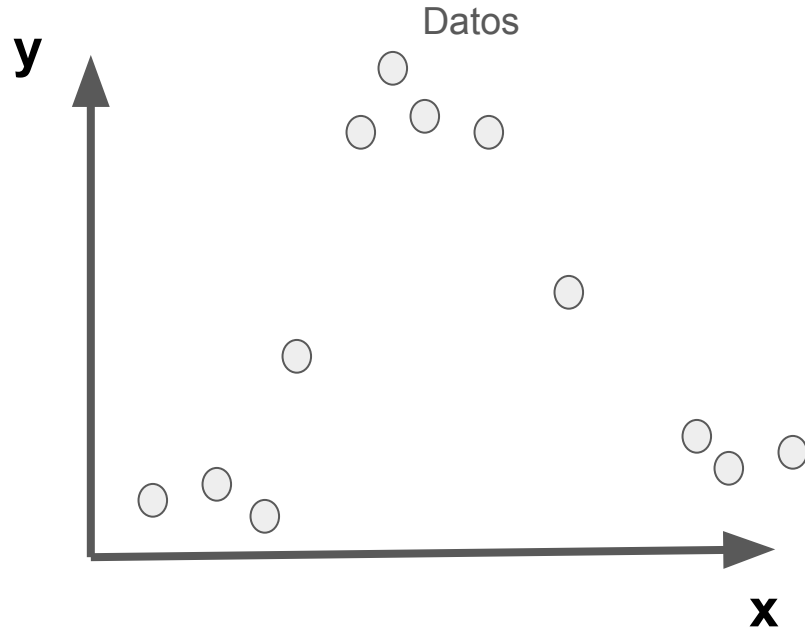
- Buscamos el feature y la condición que minimice la impureza del árbol y lo fijamos como raíz.
- Para cada uno de los dos nodos que se desprenden de la raíz buscamos el feature y la condición que me disminuya la impureza en ese subconjunto.
- Así siguiendo hasta que cada dato quede dentro de una hoja pura, o bien hasta que se cumpla algún criterio de convergencia (por ejemplo, hacer crecer el árbol hasta cierta profundidad).

# Problemas de clasificación. ¿Cómo predecimos?

La categoría más frecuente dentro de cada hoja (también podríamos dar la probabilidad de que sea de una dada clase en base a la fracción de instancias de cada clase que caigan dentro de cada hoja).

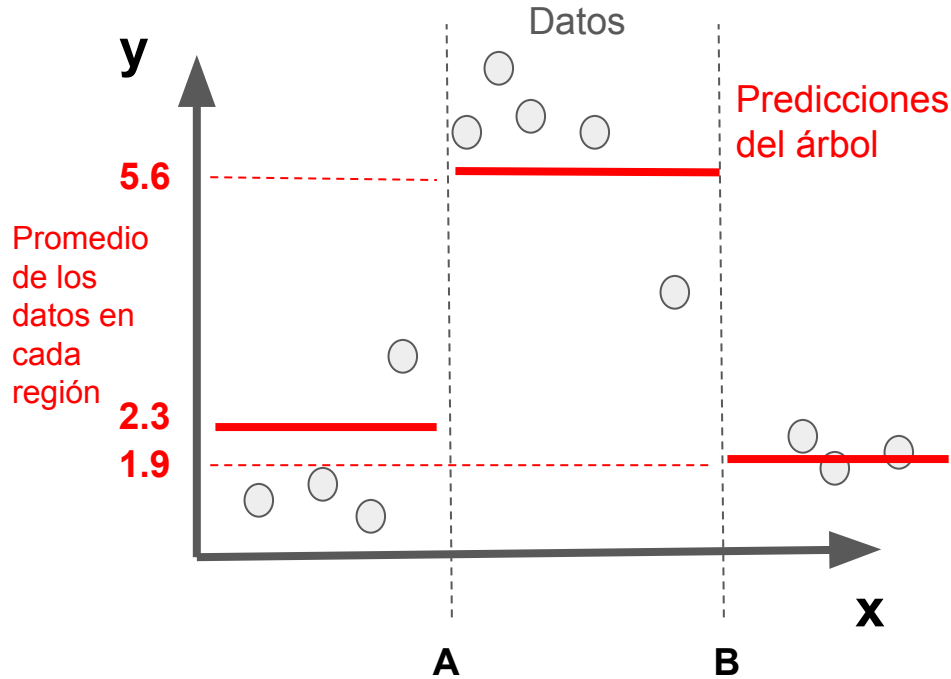


# Problemas de regresión

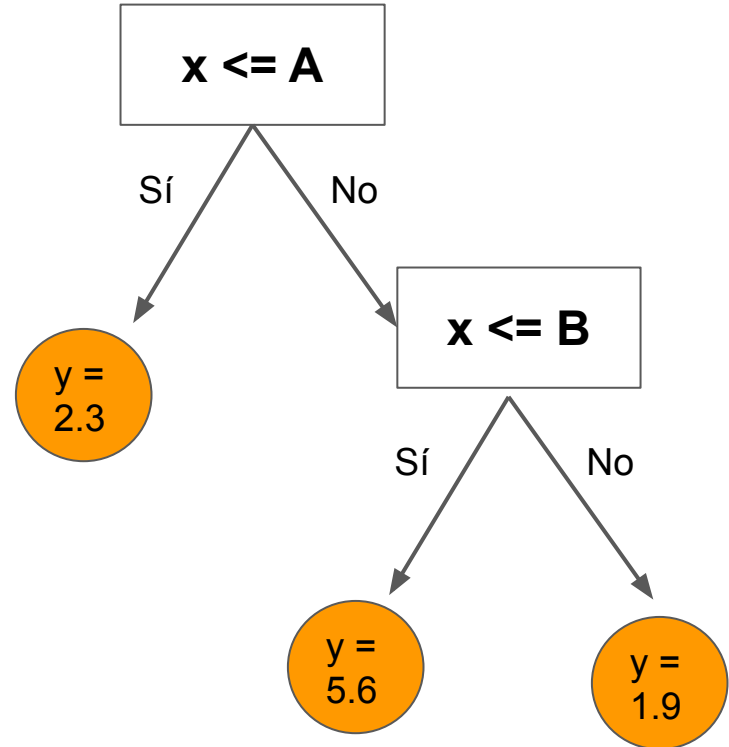


Tenemos una variable numérica y queremos predecir cómo se comporta en base a un conjunto de features. ¿Qué feature y condición elijo?

# Problemas de regresión

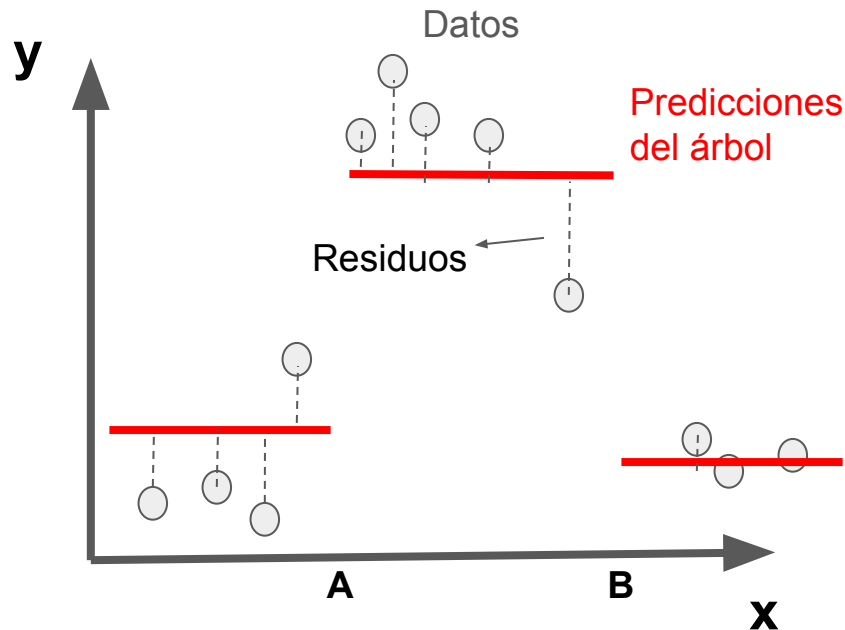


Con un árbol describimos una función por “escalones”



Damos como predicción el valor promedio de los datos en cada hoja

# Problemas de regresión



¿Cómo sabemos dónde cortar?  
(En el problema, cómo elegimos A y B?)

Buscamos los cortes que minimicen la suma del cuadrado de los residuos:

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

Suma sobre todas las hojas

Predicción para cada hoja (promedio de las instancias dentro).

# Problemas de regresión. Algoritmo

- Buscamos el feature y la condición que minimice la suma del cuadrado de los residuos.
- Para cada uno de los dos nodos que se desprenden de la raíz buscamos el feature y la condición que me disminuya la suma de los cuadrados de los residuos en ese subconjunto.
- Así siguiendo hasta que cada dato quede dentro de una hoja pura, o bien hasta que se cumpla algún criterio de convergencia (por ejemplo, hacer crecer el árbol hasta cierta profundidad).
- Damos como **predicción el promedio de las valores** dentro de cada hoja.

# Ventajas de los árboles de decisión

- Fáciles de interpretar: se asemeja bastante a la forma en la que enfrentamos un problema, más que nada de clasificación.
- No hay que preocuparse por diferencias de escala en datos numéricos
- No hay que hacer one-hot-encoding de features categóricas
- Manejan datos faltantes de una forma natural
- Permite incluir todo tipo de variable: categórica, ordinal, numérica.
- Puede usarse para problemas multiclase y regresión

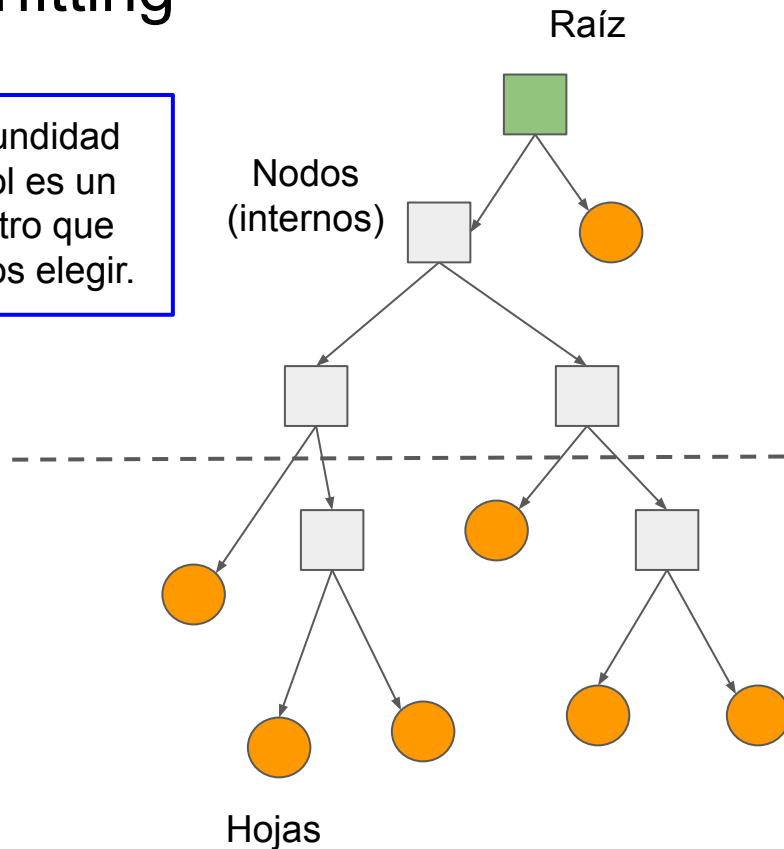


# Desventajas de los árboles de decisión

- Muchas veces no son buenos modelos, tienen baja performance.
- Árboles de mucha profundidad tienden a hacer overfitting (puedo irme tan profundo hasta que cada dato esté en hojas puras o con error igual a 0).
- Baja performance si justo arrancamos con un feature muy ruidoso
- Sesgos hacia clases más dominantes (datasets no balanceados)

# Overfitting

La profundidad del árbol es un parámetro que podemos elegir.



Underfitting: más sesgo,  
menos varianza.

Por algún lado está la  
profundidad ideal.

Overfitting: menos sesgo,  
más varianza.

# Overfitting. Algunas ideas para evitarlo

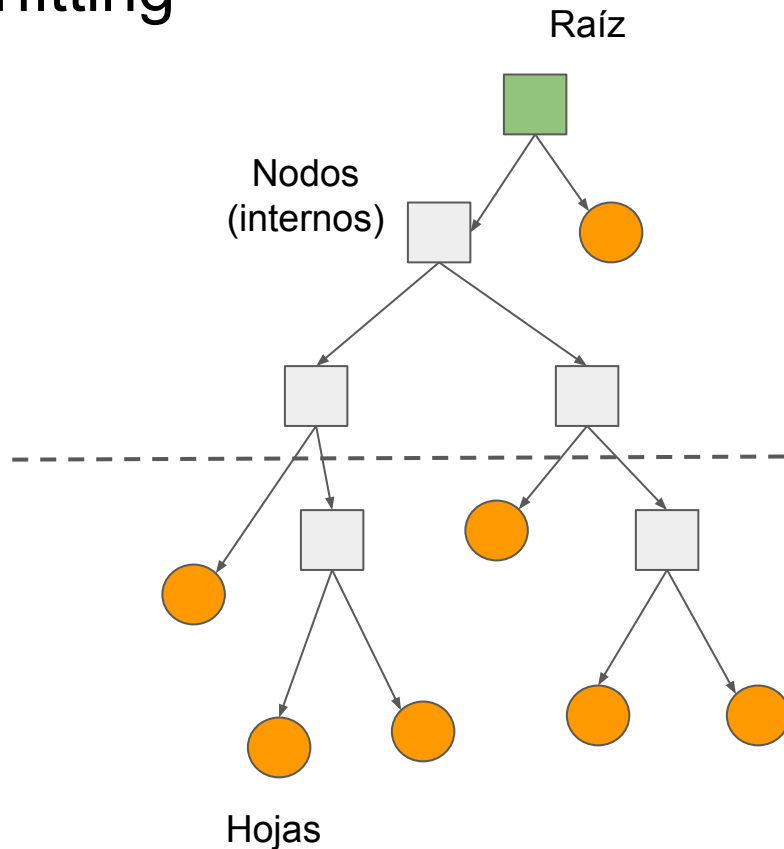
- Fijar la profundidad del árbol.
- Fijar la cantidad de hojas (para armar una cantidad fija de grupos de datos).
- Fijar la mínima cantidad de datos que están contenidos dentro de cada hoja (para hacer, por ejemplo, promedios más robustos).
- **Regularización** = *cost complexity pruning*. Penaliza árboles con muchas hojas al buscar minimizar la siguiente función:

$$\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

Diagram illustrating the cost function for regularized tree fitting:

- The term  $\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2$  is labeled "Suma de los residuos al cuadrado" (Sum of squared residuals).
- The term  $\alpha |T|$  is labeled "Constante de regularización" (Regularization constant).
- The variable  $|T|$  is labeled "Número de hojas" (Number of leaves).

# Overfitting

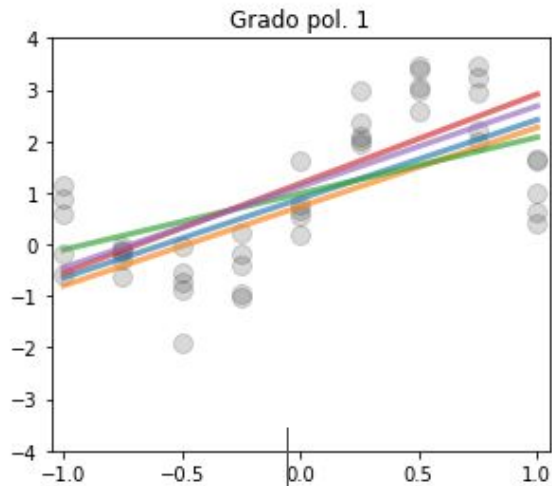


Underfitting: más sesgo,  
menos varianza.

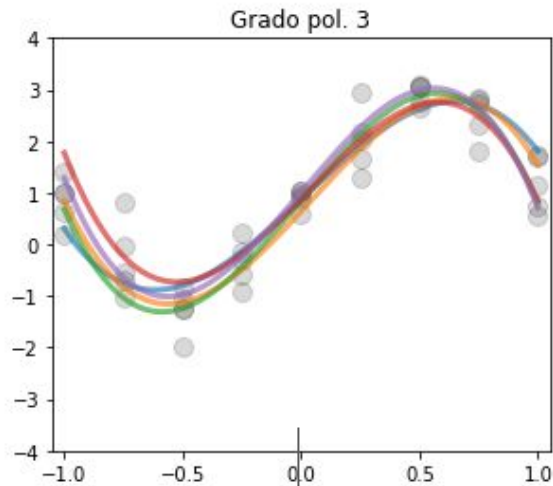
Por algún lado está la  
profundidad ideal.

## Se puede solucionar con ensambles de árboles

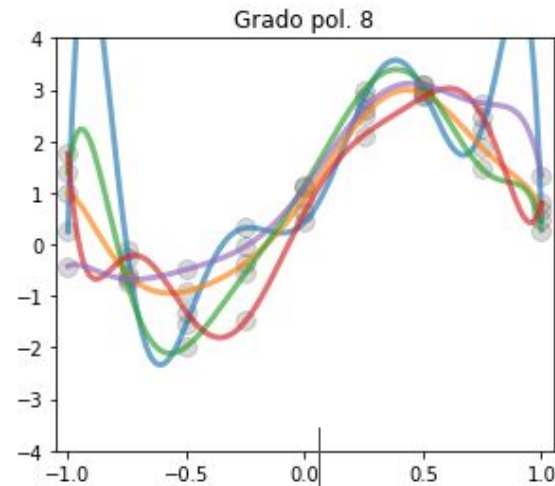
Overfitting: menos sesgo,  
más varianza.



Alto sesgo  
Modelos estables  
(establemente malos)



Punto justo



Alta varianza  
Modelos individuales malos  
**Promedios convergen a  
un mejor modelo**

Construir un modelo promedio no tiene mucho sentido si uno busca un modelo paramétrico sencillo de los datos.

**Pero si adoptamos un enfoque “resultadista” (solo importa la performance en un hold out set) es una alternativa viable**

# Conjunto (ensamble) de clasificadores. Idea

**Teorema central del límite:** sean  $X_1, X_2, X_3, \dots, X_N$ , v.a. independientes e idénticamente distribuidas con media  $\mu$  y varianza  $\sigma^2$ , entonces para  $N$  grande, el promedio está normalmente distribuido.

$$\bar{X} = \frac{1}{N} \sum_i X_i \longrightarrow \bar{X} \sim \text{Norm}\left(\mu, \frac{\sigma}{\sqrt{N}}\right)$$

**Importante!**

**Promediar** diferentes  
observaciones independientes  
**reduce la varianza.**

# Conjunto (ensamble) de clasificadores. Idea

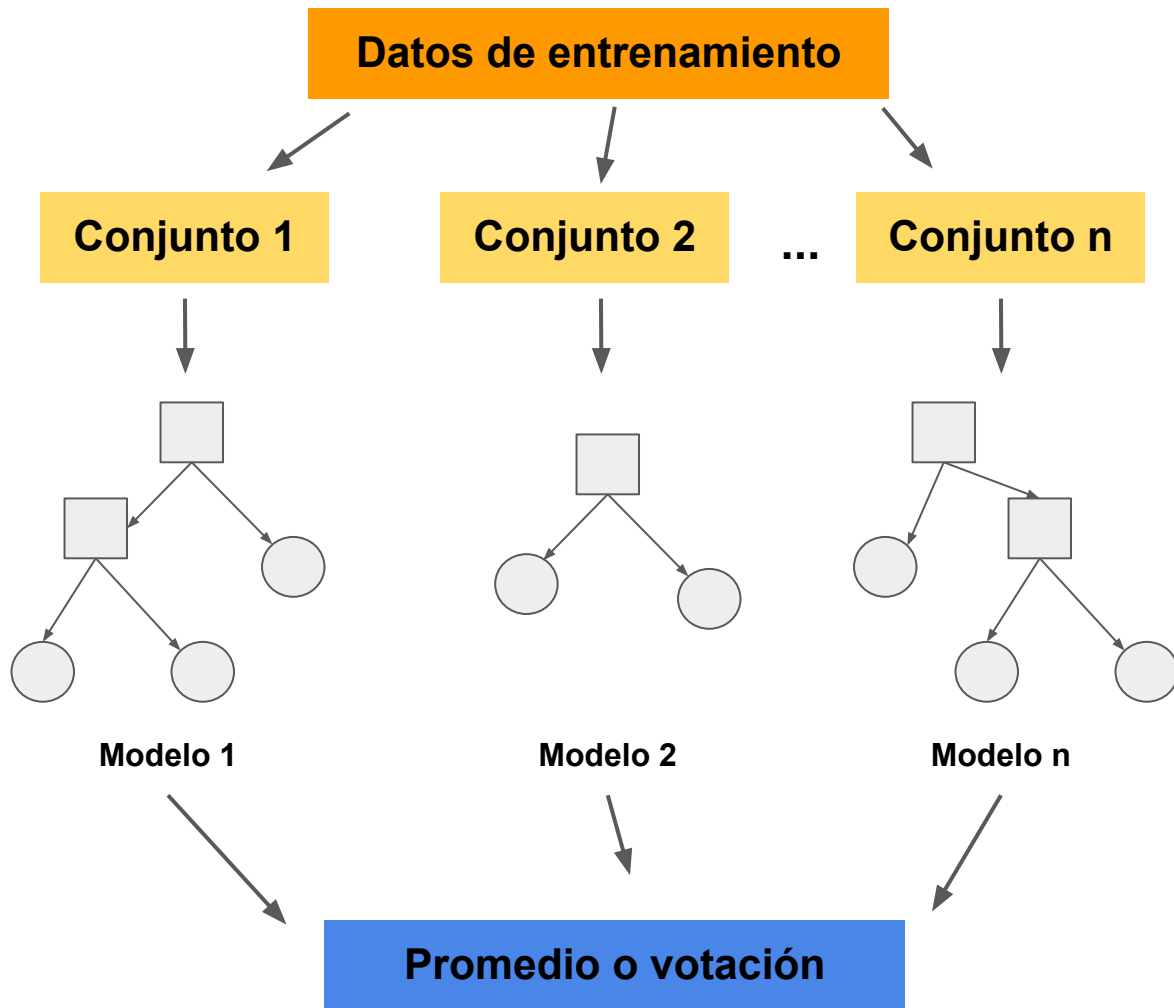
- Entrenar varios modelos distintos que sobre-ajusten (bajo sesgo y mucha varianza). Cada uno de ellos me dan un resultado.
- Promediar varios modelos **reduce la varianza**:
  - Si el problema es de regresión, el resultado final es simplemente el promedio.
  - Si el problema es de clasificación, puedo elegir la clase más frecuente entre todos los modelos (votación).
  - Si el modelo devuelve probabilidades, puedo hacer una votación ponderada.

# Bagging

(Bootstrap Aggregating)

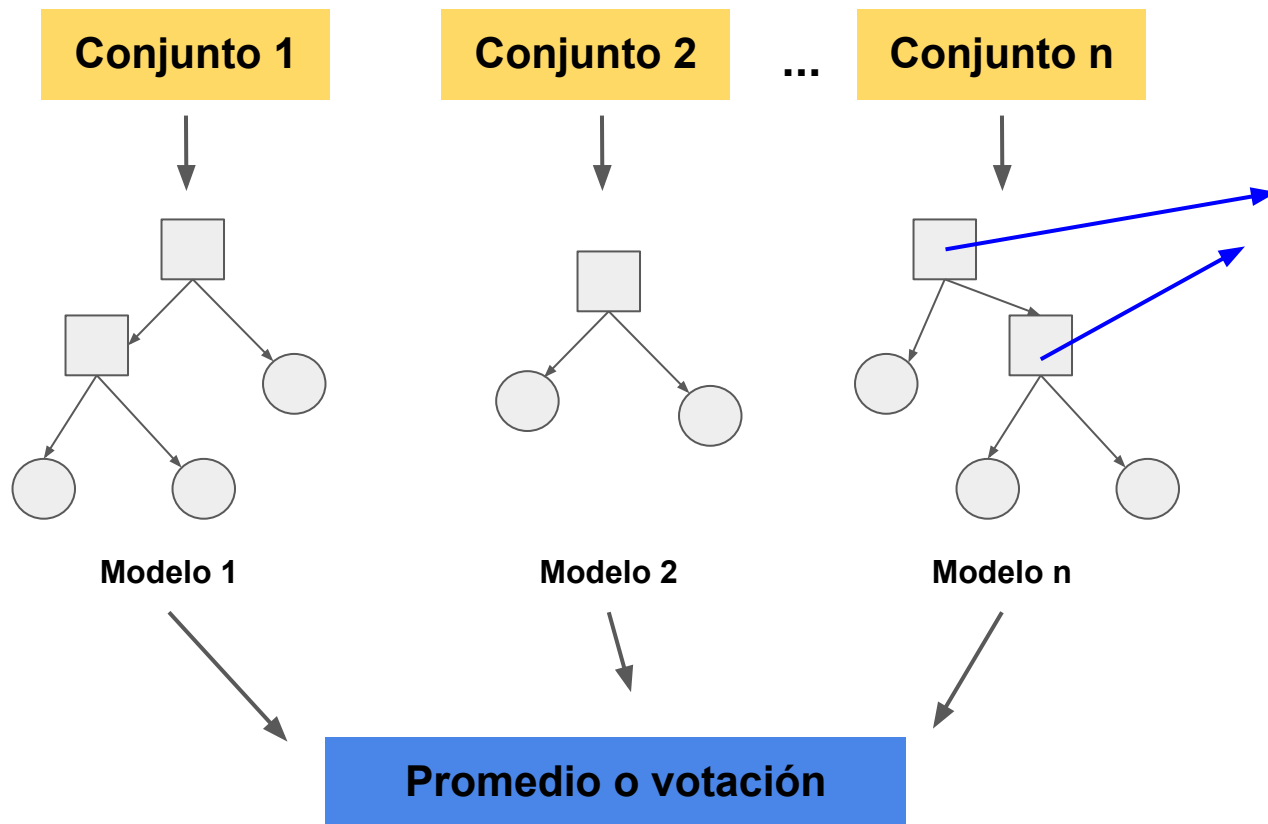
**Idea:** entrenar varios árboles sobre conjuntos de datos tomados con **muestras con reemplazo** (bootstrapping) de los datos de entrenamiento.

**Problema:** si hay una variable muy predictora, los árboles van a ser muy parecidos entre sí (estarían muy correlacionados).





# Random Forest

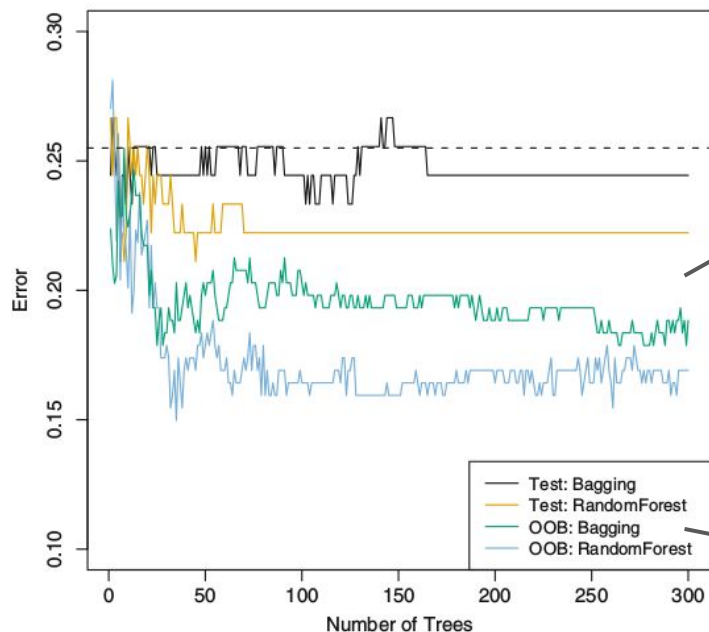


Típicamente,  $m \sim$  raíz de la cantidad de features

**Idea:** cada nodo es el mejor feature de un subconjunto de  $m$  features elegidos al azar (evita predictores fuertes).

**Resultado:** árboles descorrelacionados que promediados dan una buena estimación.

# Bagging y Random Forest. Algunas características



Introduction to Statistical Learning. p.318

La cantidad de árboles usados en la estimación no es un parámetro crítico: **más árboles no lleva al overfitting.**

Por lo tanto, este no es un hiperparámetro del modelo, muchos está bien (en general, tomamos ~100).

## OOB = Out of Bag

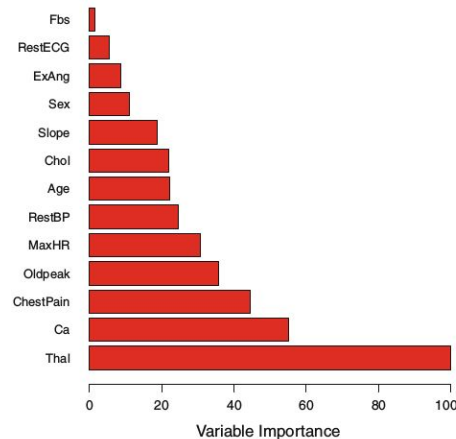
Al hacer bootstrapping estamos dejando datos afuera. Podemos usar estos para testear qué tan bien le va al modelo.

# Bagging y Random Forest. Algunas características

**Ventajas:** la combinación de diferentes modelos es siempre mucho mejor que un único modelo, podemos esperar una performance mucho más alta.

**Desventajas:** perdemos interpretabilidad de cómo el ensamble llega al resultado final.

**Feature importance:** podemos medir en promedio qué tanto una variable reduce el error o la impureza. Esto nos da una idea de qué variable es informativa y cuál no.



# Resumen: regresión y clasificación con árboles

- La idea es encontrar condiciones que separen los datos en grupos donde: haya algunas clases dominantes (clasificación) o el error respecto del promedio sea bajo (problemas de regresión).
- Los árboles de decisión pueden crecer tanto a punto de overfittear, por lo tanto es bueno tener en cuenta todas las técnicas para prevenir esto.
- Mejor que un único árbol es un bosque! Random Forest es un algoritmo mucho más poderoso que los árboles de decisión. El problema es que perdemos interpretabilidad.

# Referencias

- El capítulo 8 de An Introduction to Statistical Learning (Hastie, Tibshirani, ...) es todo sobre árboles. Se recomienda fuertemente leerlo.
- No se vayan de acá sin StatQuest!  
<https://www.youtube.com/watch?v=7VeUPuFGJHk> (el link es sobre árboles de decisión, chequear todos los relacionados con lo que vimos en esta clase!).

# Scikit-learn

## Clasificación:

### `sklearn.ensemble.RandomForestClassifier`

```
class sklearn.ensemble. RandomForestClassifier(n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None)
```

[\[source\]](#)

**Los árboles simples tienen prácticamente los mismos argumentos.**

# Scikit-learn

## Regresión:

### `sklearn.ensemble.RandomForestRegressor`

```
class sklearn.ensemble.RandomForestRegressor(n_estimators=100, *, criterion='mse', max_depth=None,  
min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto',  
max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False,  
n_jobs=None, random_state=None, verbose=0, warm_start=False, ccp_alpha=0.0, max_samples=None) \[source\]
```

**Los árboles simples tienen prácticamente los mismos argumentos.**