



DEPARTAMENTO  
DE COMPUTACION

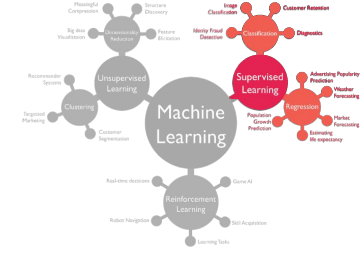
Facultad de Ciencias Exactas y Naturales - UBA

# Aprendizaje Automático

## **Clase 2:**

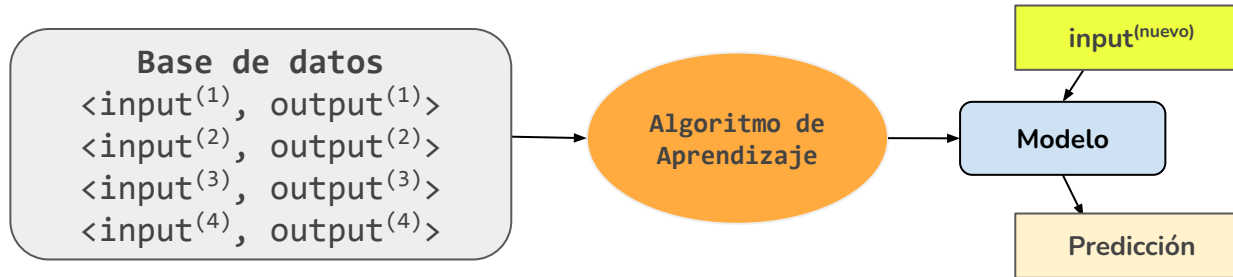
Árboles de decisión

# Repaso



# Aprendizaje Supervisado

- Dados una serie de pares  $\{\text{input}^{(i)}, \text{output}^{(i)}\}^n$  provenientes de una función desconocida  $f(\text{input}) = \text{output}$ .
- Se construye un **modelo** que permita crear un output a partir de un input que **nunca vio antes** sin la ayuda de decisiones hardcodedas por humanos.



- Un algoritmo "aprende" **un mapeo** que relaciona **input**  $\rightarrow$  **output**.
- ¿Cómo? Siguiendo **patrones** a partir de los ejemplos vistos.

# Generalización

Para intentar formalizar un poco estos conceptos, podemos utilizar las siguientes definiciones basadas en [1]

Se desea aprender una función objetivo  $\mathbf{f}^*$  **desconocida** mediante un conjunto finito de datos de entrenamiento  $\mathbf{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^n$ , una muestra de la distribución real (y de nuevo.. desconocida) de los datos.

Decimos que  $\mathbf{h}_{L,D}$ , un modelo construido usando el algoritmo  $L$  sobre  $\mathbf{D}$  **subajusta** si:

$$(\exists h' : H) Err_{train}(h_{L,D}) \geq Err_{train}(h') \wedge Err_{true}(h_{L,D}) > Err_{true}(h')$$

Análogamente,  $\mathbf{h}_{L,D}$  **sobreajusta** si:

$$(\exists h' : H) Err_{train}(h_{L,D}) \leq Err_{train}(h') \wedge Err_{true}(h_{L,D}) > Err_{true}(h')$$

En donde

$$Err_{train}(h) = \frac{1}{|\mathbf{D}|} \sum_{(x^{(i)}, y^{(i)}) \in \mathbf{D}} error(h(x^{(i)}), y^{(i)})$$

$$Err_{true}(h) = \mathbb{E}_x[error(h(x), y)] = \int error(h(x), y) P(x, y) dx$$

¿De dónde sale y?

Ya veremos más adelante que  $\mathbf{y}^{(i)} \approx \mathbf{f}^*(\mathbf{x}^{(i)})$

¿Y qué es error(A, B)?

Depende del problema y del tipo de problema (reg, clasif, etc)

# Sesgo Inductivo

Un **conjunto finito** de datos nunca alcanza para **inferir** un único modelo. Es por ello que tenemos que agregar supuestos.

## Definición: Sesgo Inductivo

Considere un algoritmo de aprendizaje  $L$  para el dominio  $X$ . Sean

$f^*$  una función arbitraria (generalmente desconocida) definida sobre  $X$ .

$D = \{(x^{(i)}, f^*(x^{(i)}))\}_{i=1}^N$  un conjunto arbitrario de ejemplos de entrenamiento (o dataset).

$f_{L,D}(x^{(i)})$  la clasificación asignada a la instancia  $x^{(i)}$  por  $f_{L,D}$  (una función que aproxima a  $f^*$ ) construida utilizando  $L$  sobre los datos  $D$ .

Dada cualquier instancia del dominio  $X$ , llamémosla  $x$ , el **sesgo inductivo** del modelo  $f_{L,D}$  es cualquier conjunto mínimo de afirmaciones  $B$  tal que para cualquier función objetivo  $f^*$  y los ejemplos de entrenamiento de  $D$ , se puede deducir el valor  $f_{L,D}(x)$ .

Mitchell lo sintetiza como:

$$(\forall x \in X)[(B \wedge D \wedge x) \vdash f_{L,D}(x)]$$

En otras palabras: **Conjunto de supuestos** que el algoritmo de aprendizaje integra al modelo para hacer predicciones sobre nuevos datos no vistos, basándose en los datos de entrenamiento que ha visto.

Estos supuestos pueden provenir de una variedad de fuentes, incluyendo:

- Estructura del modelo
- La función objetivo que el algoritmo esté optimizando
- Características de funcionamiento del algoritmo (cómo recorre el espacio de hipótesis hasta elegir un único modelo, la semilla utilizada, etc)
- etc

El sesgo inductivo determina los **tipos de funciones** que el algoritmo **puede aprender** y los **tipos de errores que se espera que cometa**.

Función objetivo  
 $f^* : X \rightarrow Y$   
(desconocida)

Ejemplos de entrenamiento  
 $D = \{ \langle x^{(i)}, f^*(x^{(i)}) \rangle \}$

Algoritmo de  
aprendizaje:  
 $L$

Hipótesis ó Modelo:  
 $f_{L,D} : X \rightarrow Y$

**Espacio de hipótesis**  
(muy grandes o infinito)

Espacio restringido por **Sesgo inductivo**  
del algoritmo  $L$

Espacio restringido  
por los **datos de**  
**entrenamiento  $D$**

$f_{L,D}$

☆  
 $f^*$

# Árboles de decisión

# Aproximación de Funciones

## Ejemplo 1

Los sábados a la mañana, un vecino a veces sale a caminar y a veces o no. Desconocemos su criterio para salir a caminar o no (función objetivo desconocida que llamaremos **Camina?** ), pero sospechamos que depende del estado del tiempo.

Queremos aprender una función **Camina?** que aproxime al criterio del vecino (es decir, que aproxime lo mejor posible a **Camina?**):

**Camina?** :: “EstadoDelDía” -> {Sí, No}

Para representar el estado del día, tenemos acceso a los siguientes atributos:

- **Cielo:** {Sol, Nublado, Lluvia}
- **Temperatura:** {Calor, Templado, Frío}
- **Humedad:** {Alta, Normal}
- **Viento:** {Fuerte, Débil}

Por lo tanto, crearemos la función **Camina?**, que quedaría:

**Camina?** :: Cielo x Temperatura x Humedad x Viento  $\rightarrow$  {Sí, No}

Por lo que empezamos por juntar datos, registramos el comportamiento del vecino durante unas semanas.



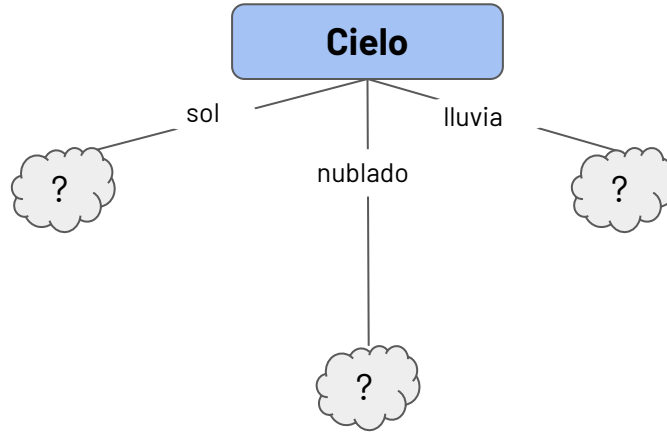
atributos

clase

instancias

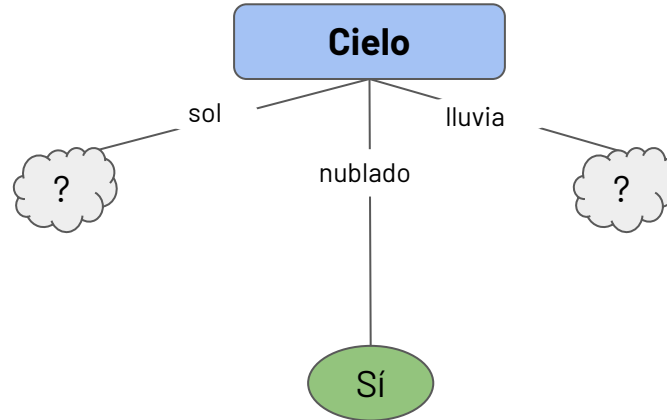
Cielo	Temperatura	Humedad	Viento	¿Camina?
Sol	Calor	Alta	Débil	No
Sol	Calor	Alta	Fuerte	No
Nublado	Calor	Alta	Débil	Sí
Lluvia	Templado	Alta	Débil	Sí
Lluvia	Frío	Normal	Débil	Sí
Lluvia	Frío	Normal	Fuerte	No
Nublado	Frío	Normal	Fuerte	Sí
Sol	Templado	Alta	Débil	No
Sol	Frío	Normal	Débil	Sí
Lluvia	Templado	Normal	Débil	Sí
Sol	Templado	Normal	Fuerte	Sí
Nublado	Templado	Alta	Fuerte	Sí
Nublado	Calor	Normal	Débil	Sí
Lluvia	Templado	Alta	Fuerte	No

# Construyendo un árbol de decisión



El atributo **Cielo** parece ser bueno para comenzar el árbol...

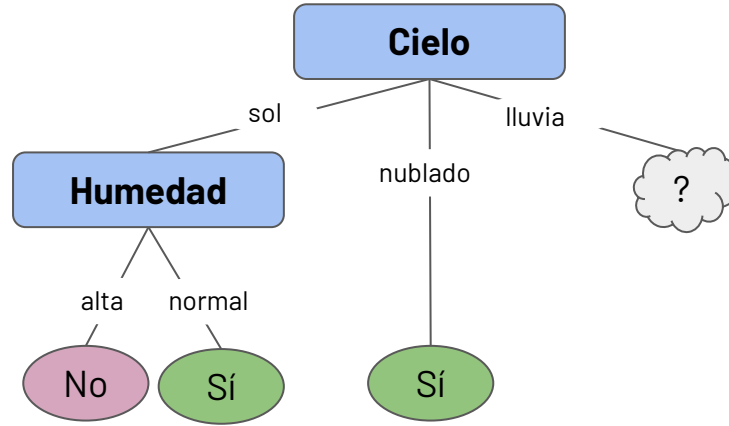
# Construyendo un árbol de decisión



El atributo **Cielo** parece ser bueno para comenzar el árbol...

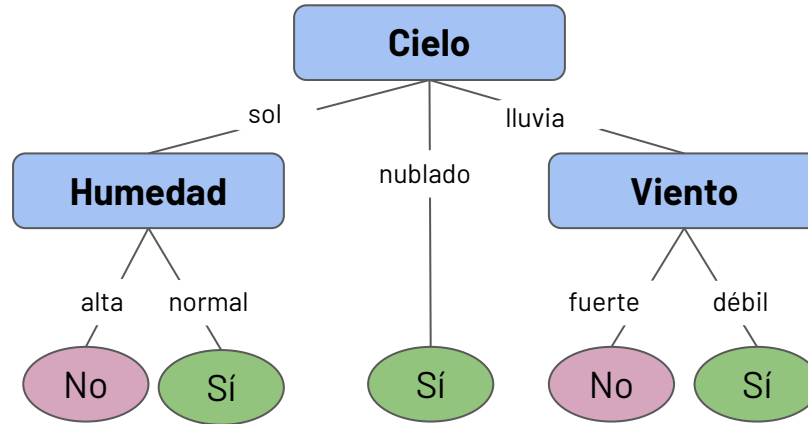
Las instancias con **Cielo == Nublado** son todas positivas.

# Construyendo un árbol de decisión



Para las instancias con **Cielo == Sol** continuamos con el atributo **Humedad**, que separa perfectamente.

# Construyendo un árbol de decisión



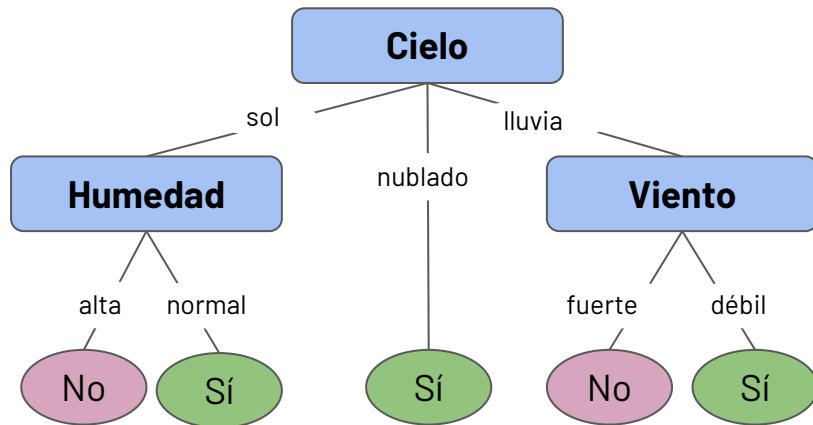
Este árbol representa la función **Camina?** :: Cielo x Temperatura x Humedad x Viento  $\rightarrow$  {Sí, No}

Cada nodo interno evalúa un atributo discreto **a**

Cada rama corresponde a un valor para **a**

Cada hoja predice un valor de **Y**

# Construyendo un árbol de decisión



Este árbol representa la función **Camina?** :: Cielo x Temperatura x Humedad x Viento → {Sí, No}

**Camina?** :: Cielo x Temperatura x Humedad x Viento

```
Camina?(c, t, h, v) ≡ if (c == Nublado) | ((c == Sol) & (h == Normal) | (c == Lluvia) & (v == Debil))  
  then Sí  
  else No
```

# Ejercicio

Dibujar el árbol correspondiente a las siguientes funciones

La convertimos en una función a Bool para simplificar: “Sí” = True, “No” = False.

**Camina?(c, t, h, v)  $\equiv$**

1.  $((c == \text{Sol}) \ \& \ (h == \text{Normal})) \mid ((c == \text{Nublado}) \ \& \ (v == \text{Fuerte}))$
2.  $((c == \text{Sol}) \ \& \ (h == \text{Normal}) \ \& \ (v == \text{Debil})) \mid ((h == \text{Alta}) \ \& \ (v == \text{Debil}))$
3.  $(c == \text{Sol}) \mid (\sim(c == \text{Sol}) \ \& \ \sim(v == \text{Debil}))$
4.  $\sim(c == \text{Sol}) \Rightarrow (v == \text{Debil})$

¿Puede un árbol representar **cualquier fórmula lógica**?

¿Hasta ahora cuál es el **sesgo inductivo**?

(hay restricciones que estamos imponiendo en el espacio de hipótesis?)

# Aproximación de Funciones

## Ejemplo 2

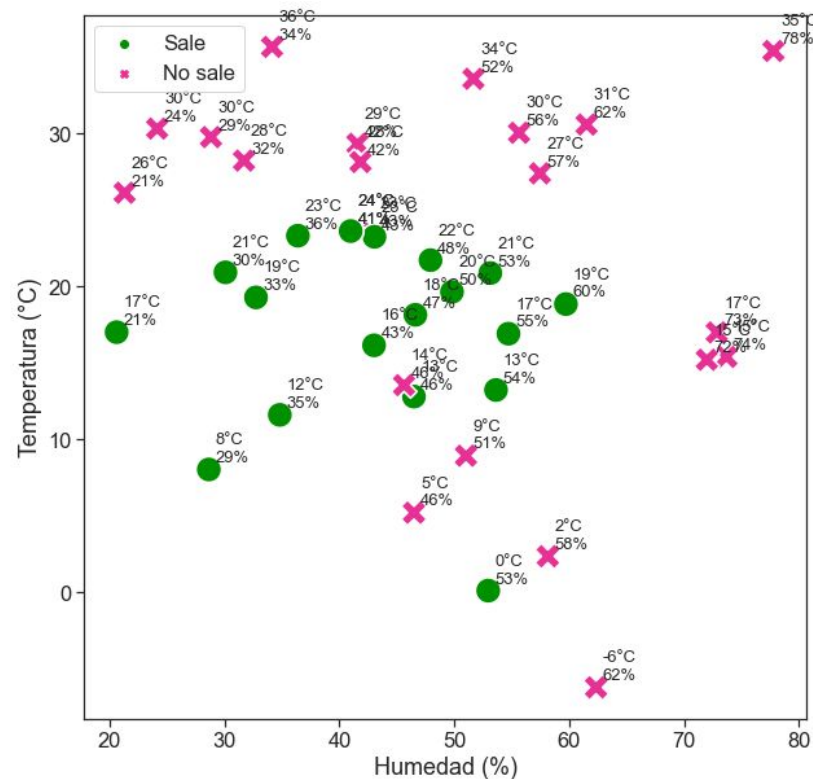
Imaginemos ahora que sólo tenemos acceso a los siguientes atributos del día:

- **Temperatura:**  $\mathbb{R} [-50, 80]$
- **Humedad:**  $\mathbb{R} [0, 100]$

Por lo tanto, crearemos la función **Caminar** quedaría:

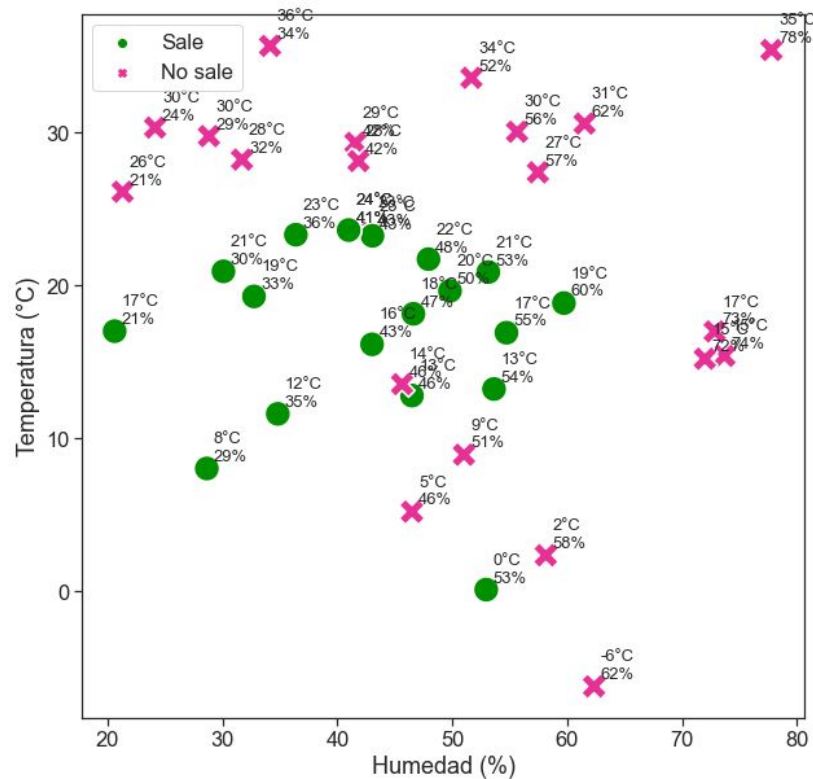
**Caminar** : Temperatura x Humedad  $\rightarrow \{\text{Sí}, \text{No}\}$

Por lo que empezamos por juntar datos, registramos el comportamiento del vecino durante unas semanas y armamos el siguiente gráfico.

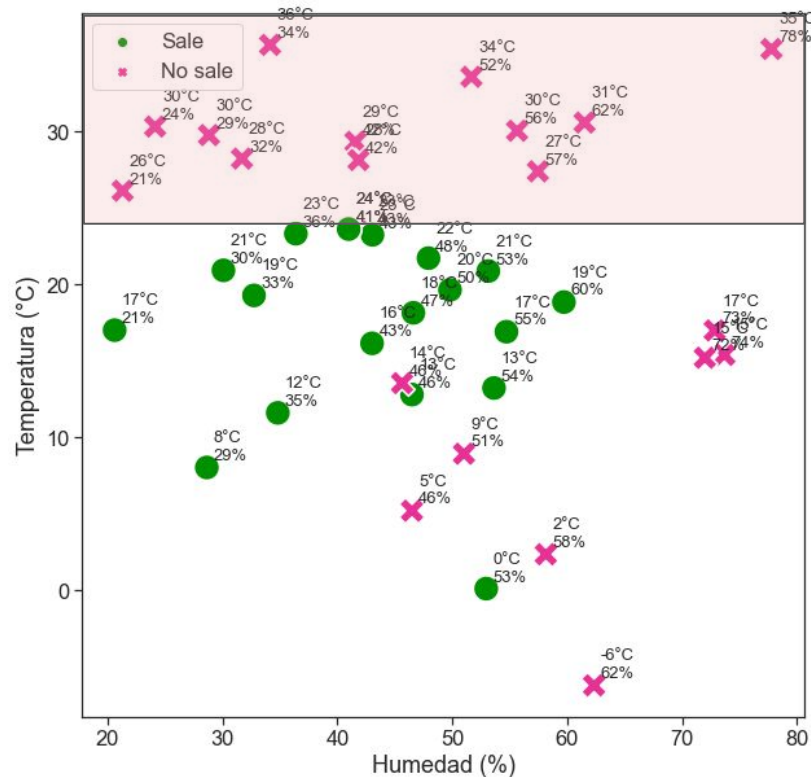
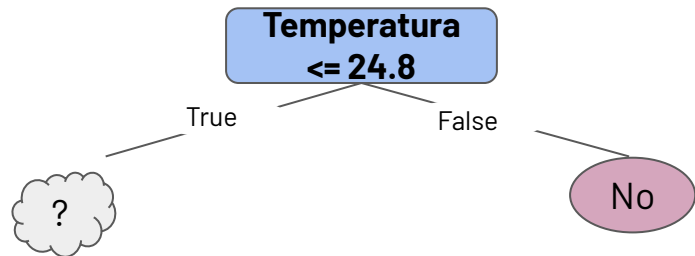




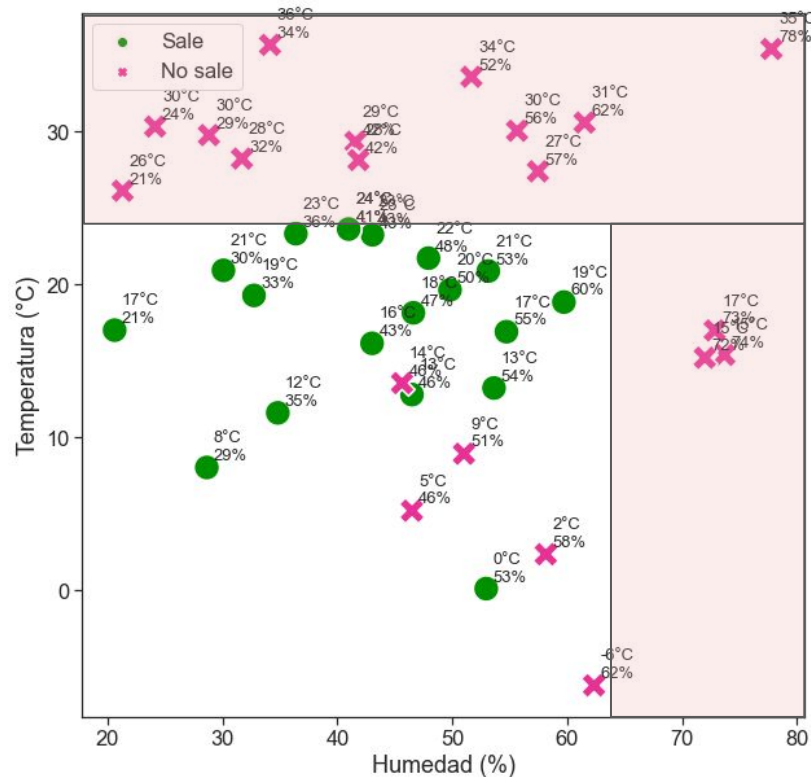
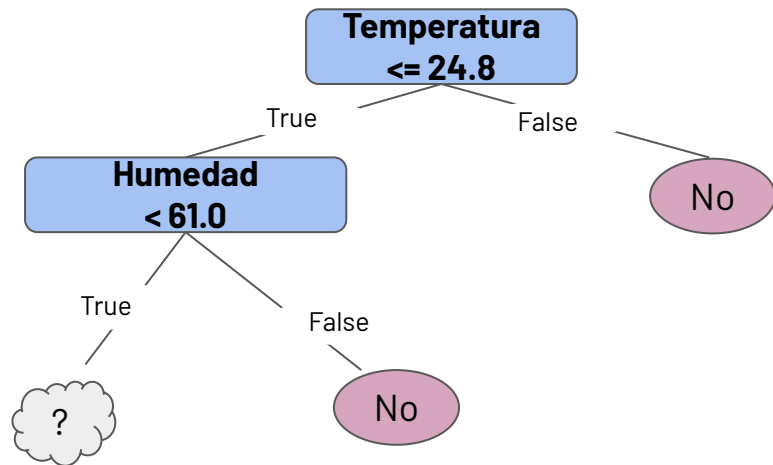
# Construyendo un árbol de decisión



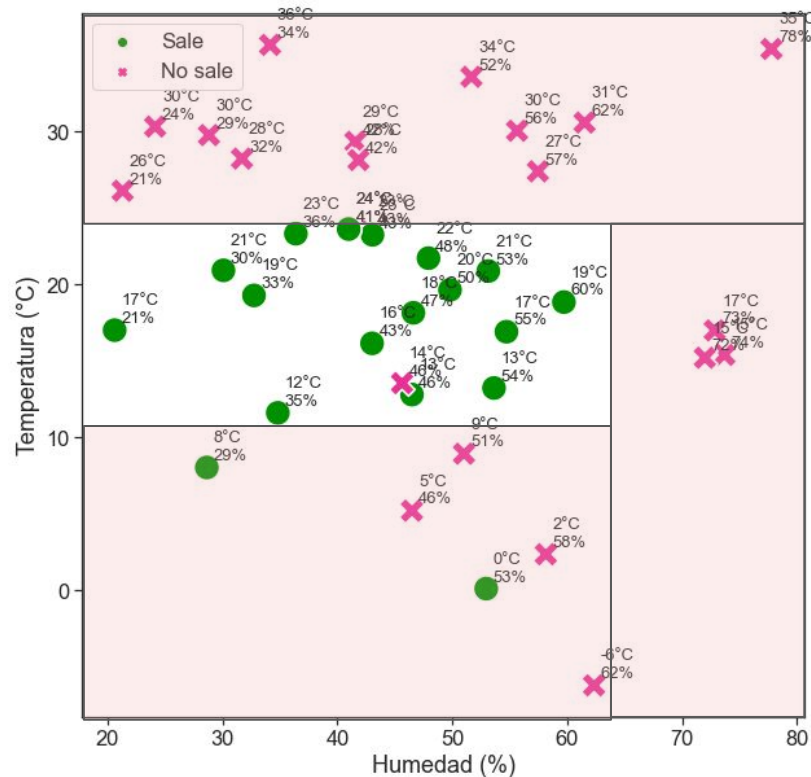
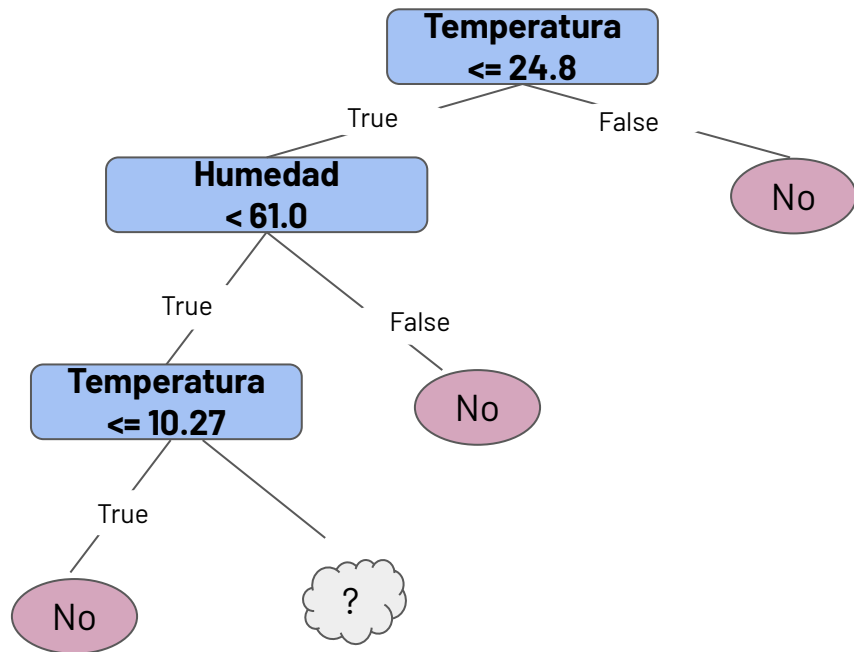
# Construyendo un árbol de decisión



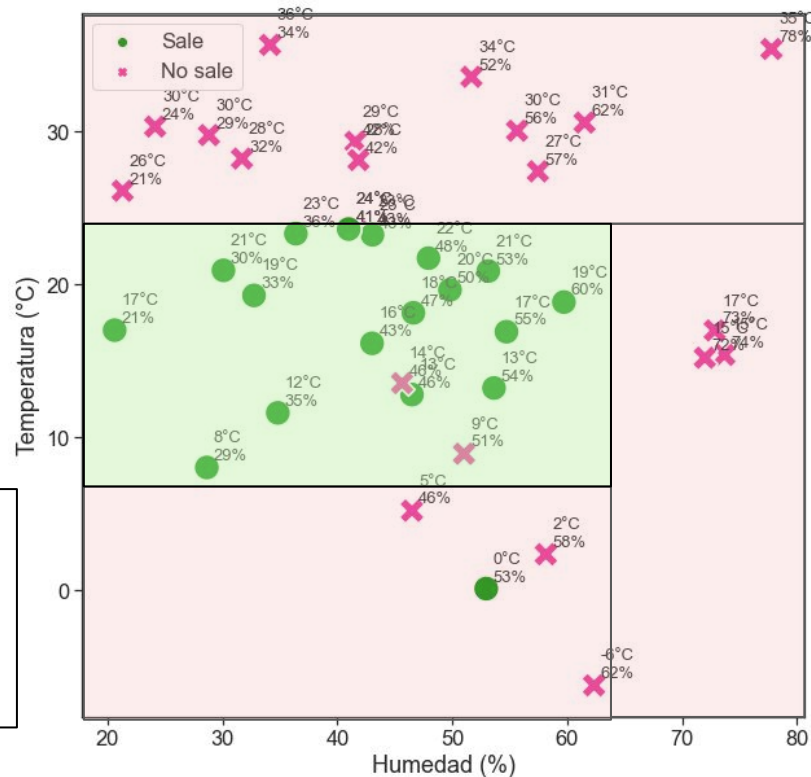
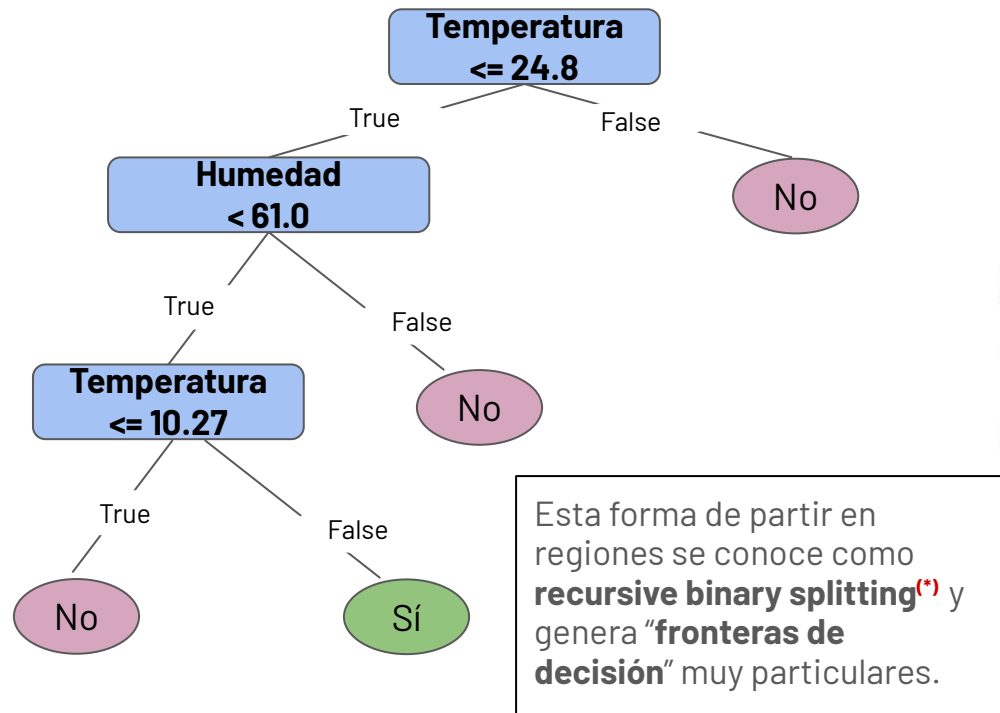
# Construyendo un árbol de decisión



# Construyendo un árbol de decisión



# Construyendo un árbol de decisión



<sup>(\*)</sup>James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). *An introduction to statistical learning: With applications in python*. Springer Nature.

# Fronteras de decisión

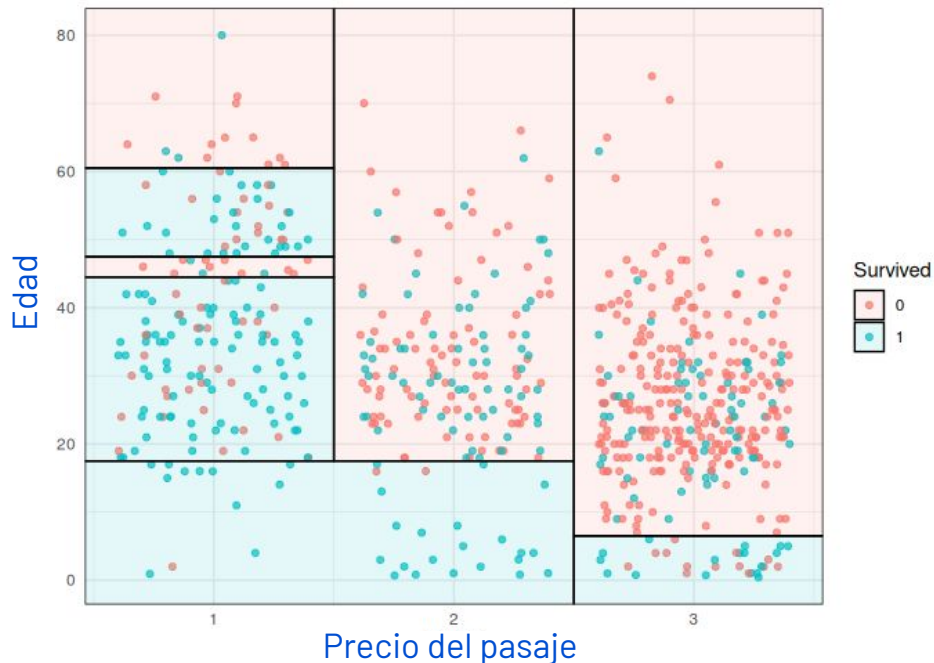
"Decision Boundaries"

**Fronteras de decisión:** Regiones del espacio de atributos en un problema de clasificación donde un modelo de aprendizaje automático toma decisiones de clasificación diferentes.

**En árboles** binarios para atributos continuos:

- Forma **jerárquica** (cada región se divide en exactamente 2 regiones o ninguna)
- **Rectangulares** (hiper-rectángulos si  $D > 2$ ).
- Cada **hoja** representa una región del espacio de atributos donde se asigna una etiqueta de clase.

Dataset Titanic



# Encontrar la “mejor” partición

## Idea de algoritmo

Buscar **la mejor partición** en regiones (hiper-)rectangulares / jerárquicas, en el espacio de atributos.

¿Qué es “**mejor**”? (ya veremos)

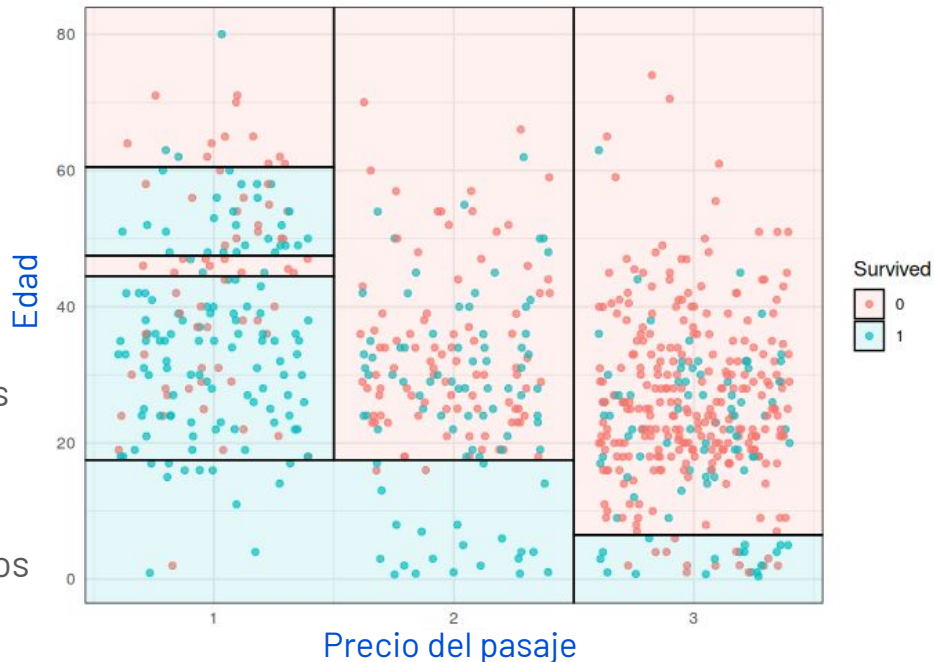
¿Qué **complejidad** computacional tendría?

Buscar las particiones en (hiper-)rectángulos posibles para los puntos de un dataset y testear cada solución es **NP-Hard** (tan difícil como el halting problem)

(problema mencionado la clase pasa, recorrer espacios grandes/infinitos de hipótesis).

¿Entonces?: **utilizaremos heurísticas**. En este caso, greedy para converger a un máximo local.

Dataset Titanic



# Inducción Top-Down de Árboles de Decisión

## Algoritmo CART<sup>(\*)</sup> (para atributos continuos)

Sea **S** una muestra de instancias con atributos **A**. Para construir un árbol de decisión ejecutamos:

1. Elegimos el par  $a \in A$ ,  $c \in \mathbb{R}$  entre los posibles<sup>(1)</sup> “cortes” que mejor<sup>(2)</sup> divida a **S** para **nodo actual**.
2. Crear dos hijos del **nodo actual**.
3. Dividir las instancias de **S** en los nuevos nodos, según  $\langle a, c \rangle$ :  
$$S_{\leq} \leftarrow \{x \mid x \in S \wedge x[a] \leq c\}$$
$$S_{>} \leftarrow \{x \mid x \in S \wedge x[a] > c\}$$
4. **Repetir por cada hijo** hasta cumplir algún criterio de parada (ej. Altura máxima)
5. El valor de una hoja será la etiqueta más frecuente en la región

<sup>(1)</sup> Son infinitos.. ¿Cuántos y cuáles evalúo?

<sup>(2)</sup> “mejor” ???.

<sup>(\*)</sup> Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Cart. Classification and regression trees*.



# Inducción Top-Down de Árboles de Decisión

Algoritmos ID3<sup>(\*)</sup> y C4.5<sup>(\*\*)</sup>, Quinlan (**para atributos discretos**)<sup>(\*\*\*)</sup>

Sea **S** una muestra de instancias con atributos **A**. Para construir un árbol de decisión ejecutamos:

1. Elegimos  $a \in A$ , el atributo que **mejor**<sup>(1)</sup> divide a S como nodo de decisión para **nodo actual**.
2. Para cada valor **posible**<sup>(2)</sup>  $v_i$  del atributo **a**, crear un nuevo hijo del **nodo actual**.
3. Repartir las instancias de **S** en los nuevos nodos, según su valor para el atributo **a**:  
$$S_i \leftarrow \{x \mid x \in S \wedge x[a] = v_i\}$$
4. **repetir** hasta cumplir algún criterio de parada (ej. Altura máxima) **definiendo**  $A_i \leftarrow A - \{a\}$ <sup>(3)</sup>.
5. El valor de una hoja será la etiqueta más frecuente en la región

<sup>(1)</sup> Seguimos con el problema de “mejor”.

<sup>(2)</sup> ¿Conocemos todos los posibles valores que puede tomar cada atributo?

<sup>(3)</sup> ¿Por qué en la versión continua no quitamos el atributo del conjunto?

**(\*)** Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1, 81-106.

**(\*\*)** Quinlan, J. R. (1987). Simplifying decision trees. *International journal of man-machine studies*, 27(3), 221-234.

**(\*\*\*)** Versión simplificada. [Mitchell 3 “Decision Tree Learning”. p56]

# Costo computacional

Complejidad temporal para caso discreto (**n**: #instancias, **p**: #atributos)

- Construcción:  **$O(n p^2)$**  peor caso <sup>(a)</sup>,  **$O(n p)$**  promedio <sup>(b)</sup>
- Consulta de nuevas instancias (inferencia):  **$O(p)$**

Como vimos, estos algoritmos greedy **no evalúan toda** combinación de regiones posibles.

<sup>(a)</sup> P. E. Utgoff. "Incremental induction of decision trees". Machine Learning, 4(2):161–186, 1989

<sup>(b)</sup> J. W. Shavlik, R. J. Mooney, and G. Towell. "Symbolic and neural learning algorithm: An experimental comparison". Machine Learning, 6:111–143, 1991.

# Sesgo inductivo

¿Cuál es el sesgo inductivo hasta ahora?

- 1) El tipo de regiones de decisión que puede generar tiene **forma de hiper-rectángulos**.
- 2) Las regiones que exploramos se determinan de manera **Greedy**. [Hill Climbing](#) sin backtracking (converge a un máximo local. Para pensar, ¿máximo de qué función?)

(ver [LIN, Jimmy, et al. Generalized and scalable optimal sparse decision trees. En International Conference on Machine Learning. PMLR, 2020. p. 6150-6160.] para soluciones modernas)

¿Cómo elegir el mejor corte?

# Criterios de selección del mejor atributo

¿Cómo comparamos posibles opciones de atributos / pares atributos-cortes para quedarse con el “mejor”?

Sea  $M(S)$  una **medida** que nos indique la “**homogeneidad**” de la región.

Decimos que el par  $\langle a_1, c_1 \rangle$  es mejor que el par  $\langle a_2, c_2 \rangle$  si sucede que:

$$\Delta M(S, \langle a_1, c_1 \rangle) > \Delta M(S, \langle a_2, c_2 \rangle)$$

*“la ganancia en homogeneidad del corte  $\langle a_1, c_1 \rangle$  es mayor que la de  $\langle a_2, c_2 \rangle$ ”.*

En donde:

- $\Delta M(S, \langle a, c \rangle) = M(S) - (\text{Prop}_{\leq} * M(S_{\leq}) + \text{Prop}_{>} * M(S_{>}))$

*“cuánto gano si divido a  $S$  en regiones  $S_{\leq}$  y  $S_{>}$  según la medida  $M$ .”*

(caso binario, si no, un término por cada  $S_i$  con su proporción)

- $\text{Prop} = \#(S_x) / \#(S)$  la proporción de instancias en la nueva región.

Recordar

$$S_{\leq} = \{x \mid x \in S \wedge x[a] \leq c\}$$

$$S_{>} = \{x \mid x \in S \wedge x[a] > c\}$$

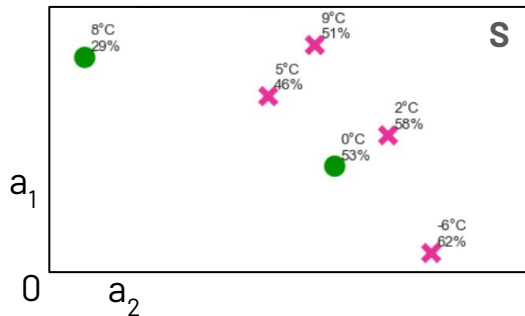
# Medidas de homogeneidad de una región

## Classification Error Rate

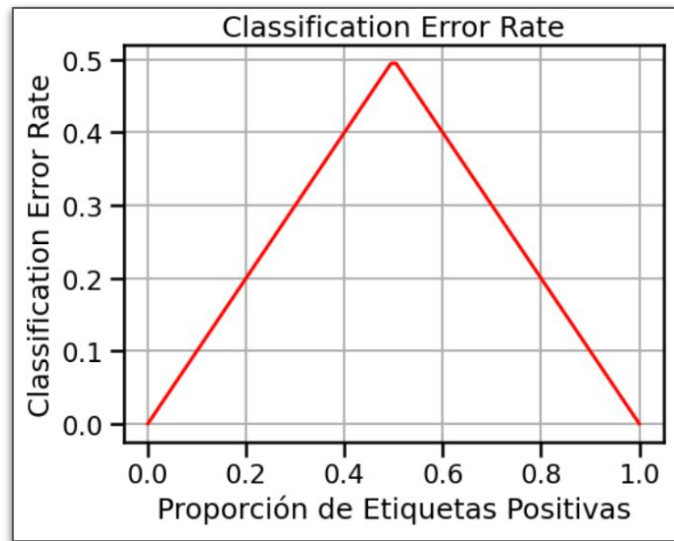
Podemos definir nuestra métrica de "la calidad de una región" como:

$CER(S) = 1 - \max_{k \in K} (p(k))$  en donde  $p(k)$  representa la proporción de la **clase**  $k$  en la región  $S$ .

Es decir, la proporción de instancias que **serían mal clasificadas** si consideramos a la **clase mayoritaria** como clase para la región. Ejemplo: Sea  $S$  la siguiente región.



$$CER(S) = 1 - (4/6) = 2/6$$



# Medidas de homogeneidad de una región

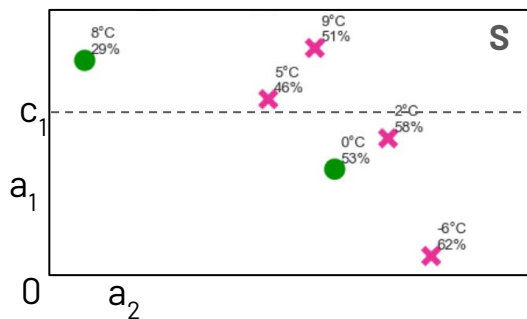
## Criterio 1: Tasa de error de clasificación (Classification Error Rate)

Podemos definir nuestra métrica de "la calidad de una región" como:

$CER(S) = 1 - \max_{k \in K} (p(k))$  en donde  $p(k)$  representa la proporción de la **clase**  $k$  en la región  $S$ .

Es decir, la proporción de instancias que **serían mal clasificadas** si consideramos a la **clase mayoritaria** como clase para la región. Ejemplo: Sea  $S$  la siguiente región.

La **Ganancia en Tasa de Error de clasificación** cuantifica cuánto mejora la CER luego de realizar un corte



$$CER(S) = 1 - (4/6) = 2/6$$

$$\begin{aligned}\Delta CER(S, < a_1, c_1 >) &= \frac{2}{6} - (Prop_{\leq} * CER(S_{\leq}) + Prop_{>} * CER(S_{>})) \\ &= \frac{2}{6} - \left( \frac{3}{6} * CER(S_{\leq}) + \frac{3}{6} * CER(S_{>}) \right) \\ &= \frac{2}{6} - \left( \frac{3}{6} * \frac{1}{3} + \frac{3}{6} * \frac{1}{3} \right) \\ &= 0 \quad (\text{un mal corte})\end{aligned}$$

# Medidas de homogeneidad de una región

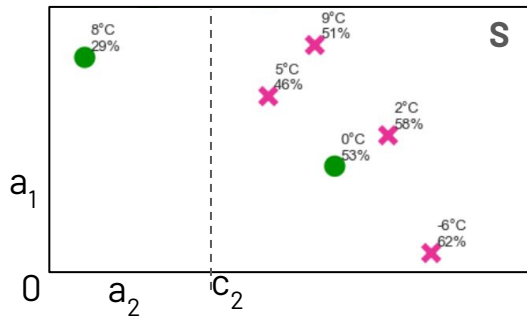
## Criterio 1: Tasa de error de clasificación (Classification Error Rate)

Podemos definir nuestra métrica de "la calidad de una región" como:

$CER(S) = 1 - \max_{k \in K} (p(k))$  en donde  $p(k)$  representa la proporción de la **clase**  $k$  en la región  $S$ .

Es decir, la proporción de instancias que **serían mal clasificadas** si consideramos a la **clase mayoritaria** como clase para la región. Ejemplo: Sea  $S$  la siguiente región.

La **Ganancia en Tasa de Error de clasificación** cuantifica cuánto mejora la CER luego de realizar un corte



$$CER(S) = 1 - (4/6) = 2/6$$

$$\begin{aligned} \Delta CER(S, < a_2, c_2 >) &= \frac{2}{6} - (Prop_{\leq} * CER(S_{\leq}) + Prop_{>} * CER(S_{>})) \\ &= \frac{2}{6} - (\frac{1}{6} * CER(S_{\leq}) + \frac{5}{6} * CER(S_{>})) \\ &= \frac{2}{6} - (\frac{1}{6} * 0 + \frac{5}{6} * \frac{1}{5}) \\ &= \frac{1}{6} \quad (\text{un mejor corte}) \end{aligned}$$



# Medidas de homogeneidad de una región

## Impureza Gini

**Impureza Gini** ("Gini Impurity"):

Supongamos que:

1. Elegimos aleatoriamente un elemento en nuestro conjunto de datos.
2. Lo clasificamos aleatoriamente según la distribución de clases en el conjunto.

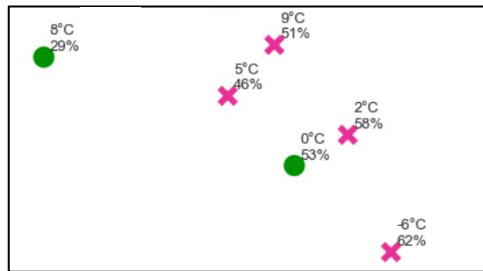
En el ejemplo, clasificaríamos el punto como círculo el **2/6** de las veces y como cruz el **4/6** del tiempo.

¿Cuál es la probabilidad de clasificar el elemento incorrectamente? La respuesta a esa pregunta es la impureza de Gini.

En otras palabras, intenta reflejar la **frecuencia** con la que un elemento elegido aleatoriamente de un conjunto estaría **etiquetado incorrectamente** si

*(en vez de asignar la clase mayoritaria como CER)*

**se etiquetara aleatoriamente** de acuerdo con la distribución de etiquetas en el conjunto



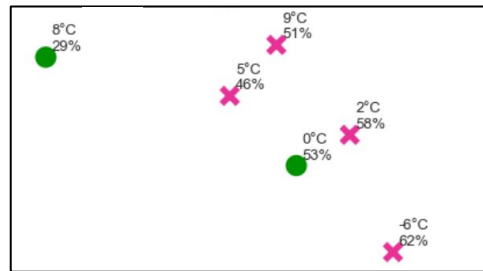
# Medidas de homogeneidad de una región

**Criterio 2:** Ganancia Gini (Reducción de la impureza)

$$\begin{aligned} Gini(S) &= \sum_{k \in \text{clases}(K)} p(k) (1 - p(k)) \\ &= \sum_{k \in \text{clases}(K)} (p(k) - p(k)^2) \\ &= \left( \sum_{k \in \text{clases}(K)} p(k) \right) - \left( \sum_{k \in \text{clases}(K)} p(k)^2 \right) \\ &= 1 - \sum_{k \in \text{clases}(K)} p(k)^2 \end{aligned}$$

**p(k):** Probabilidad de elegir un elemento de la clase k (la estimamos como es la proporción de elementos de la clase i en la región)

Para el ejemplo, **Gini(S)** =  $1 - [(2/6)^2 + (4/6)^2] = 0.44$



La **Ganancia de Gini** cuantifica cuánto mejora la impureza después de realizar una división en los datos.

$$Gini\_Gain(S, \langle a, c \rangle) = Gini(S) - (Prop_{\leq} \cdot Gini(S_{\leq}) + Prop_{>} \cdot Gini(S_{>}))$$

# Medidas de homogeneidad de una región

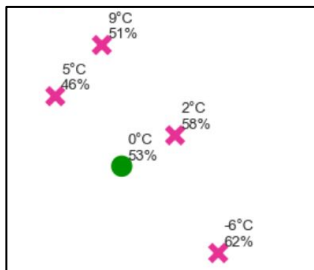
## Entropía de una región

### Entropía de una región S:

$$H(S) = - \sum_{k \in \text{clases}(S)} p(k) \log_2 p(k)$$

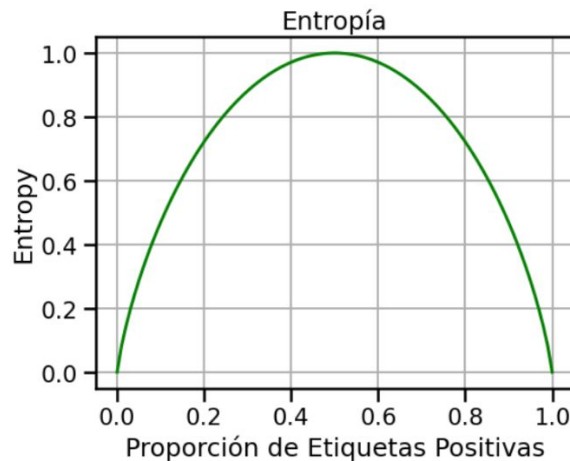
**p(k):** Probabilidad de elegir un elemento de la clase k.

Mide la **sorpresa que esperamos** obtener si sacáramos elementos al azar de la región conociendo las proporciones de las clases.



Si saco una cruz en  $S_1 \rightarrow$  sorpresa Baja (0.32)

Si saco un círculo en  $S_1 \rightarrow$  sorpresa Alta (2.32)



Ver paréntesis al final de la clase.

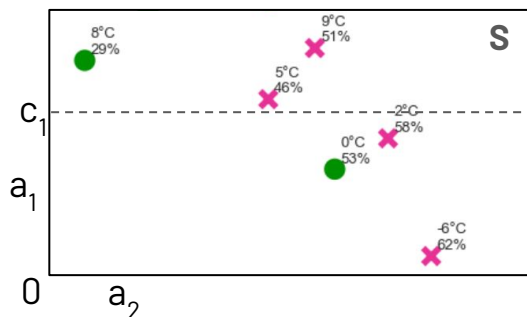
# Medidas de homogeneidad de una región

**Criterio 3:** Ganancia de Información (reducción de entropía)

$$H(S) = - \sum_{k \in \text{clases}(S)} p(k) \log_2 p(k)$$

**Ganancia de información: Cuánta entropía removemos (cuánta información ganamos) al hacer un corte.**

$$\text{InfoGain}(S, \langle a, c \rangle) = H(S) - (\text{Prop}_{\leq} \cdot H(S_{\leq}) + \text{Prop}_{>} \cdot H(S_{>}))$$



$$H(S) = - \left( \frac{2}{6} \cdot \log_2 \left( \frac{2}{6} \right) + \frac{4}{6} \cdot \log_2 \left( \frac{4}{6} \right) \right) = 0.92$$

$$\begin{aligned} \text{InfoGain}(S, \langle a_1, c_1 \rangle) &= 0.92 - (\text{Prop}_{\leq} \cdot H(S_{\leq}) + \text{Prop}_{>} \cdot H(S_{>})) \\ &= 0.92 - \left( \frac{3}{6} \cdot H(S_{\leq}) + \frac{3}{6} \cdot H(S_{>}) \right) \\ &= 0.92 - \left( \frac{3}{6} \cdot 0.92 + \frac{3}{6} \cdot 0.92 \right) \\ &= 0 \end{aligned}$$

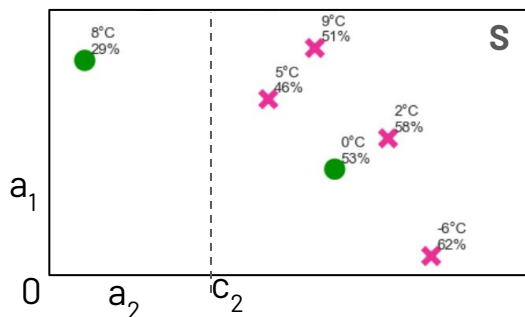
$$H(S) = - \sum_{k \in \text{clases}(S)} p(k) \log_2 p(k)$$

# Medidas de homogeneidad de una región

**Criterio 3:** Ganancia de Información (reducción de entropía)

**Ganancia de información: Cuánta entropía removemos (cuánta información ganamos) al hacer un corte.**

$$\text{InfoGain}(S, \langle a, c \rangle) = H(S) - (\text{Prop}_{\leq} \cdot H(S_{\leq}) + \text{Prop}_{>} \cdot H(S_{>}))$$

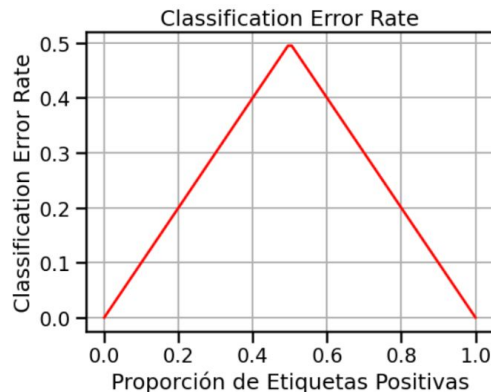
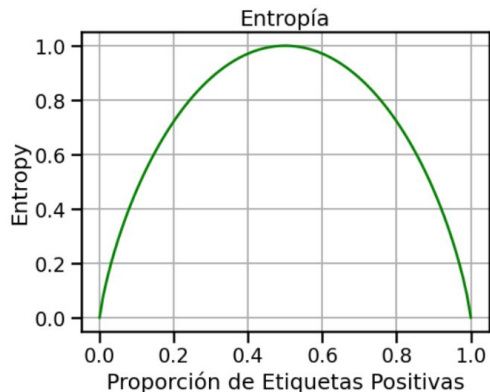


$$H(S) = - \left( \frac{2}{6} \cdot \log_2 \left( \frac{2}{6} \right) + \frac{4}{6} \cdot \log_2 \left( \frac{4}{6} \right) \right) = 0.92$$

$$\begin{aligned} \text{InfoGain}(S, \langle a_2, c_2 \rangle) &= 0.92 - (\text{Prop}_{\leq} \cdot H(S_{\leq}) + \text{Prop}_{>} \cdot H(S_{>})) \\ &= 0.92 - \left( \frac{1}{6} \cdot H(S_{\leq}) + \frac{5}{6} \cdot H(S_{>}) \right) \\ &= 0.92 - \left( \frac{1}{6} \cdot 0 + \frac{5}{6} \cdot 0.72 \right) \\ &= 0.32 \end{aligned}$$

# Resumen hasta el momento

Necesitamos evaluar la **calidad de un corte**. Para ello definimos la ganancia según distintas medidas:



Ganancia en tasa de clasificación **no se utiliza en general** (muy ruidoso y otros problemas). Leer sección 6.8 de [github.com/rasbt/stat479-machine-learning-fs19/blob/master/06\\_trees/06-trees\\_\\_notes.pdf](https://github.com/rasbt/stat479-machine-learning-fs19/blob/master/06_trees/06-trees__notes.pdf)

En todos los casos, calculamos la ganancia en realizar el corte  $\langle a, c \rangle$  como:

$$\Delta M(S, \langle a, c \rangle) = M(S) - (Prop_{\leq} \cdot M(S_{\leq}) + Prop_{>} \cdot M(S_{>}))$$

# Sobreajuste en árboles

# Sobreajuste en árboles



$$(\exists h' : H) Err_{train}(h_{L,D}) \leq Err_{train}(h') \wedge Err_{true}(h_{L,D}) > Err_{true}(h')$$

**Causas:** (discutir)

¿Cómo evitarlo?

- **Criterio de parada:** No construir más allá de cierta profundidad.
- **Pruning (poda):** Construir el árbol entero; podar las ramas cuando ello mejore la performance sobre datos separados. Funciona mejor en la práctica.
- **Rule post-pruning:** Construir el árbol entero; convertir árbol a reglas; sacar precondiciones de las reglas cuando ello mejore su performance sobre datos separados; reordenar las reglas según accuracy.
- **¡Veremos** (en la clase de ensambles) **que a veces no conviene evitarlo!**



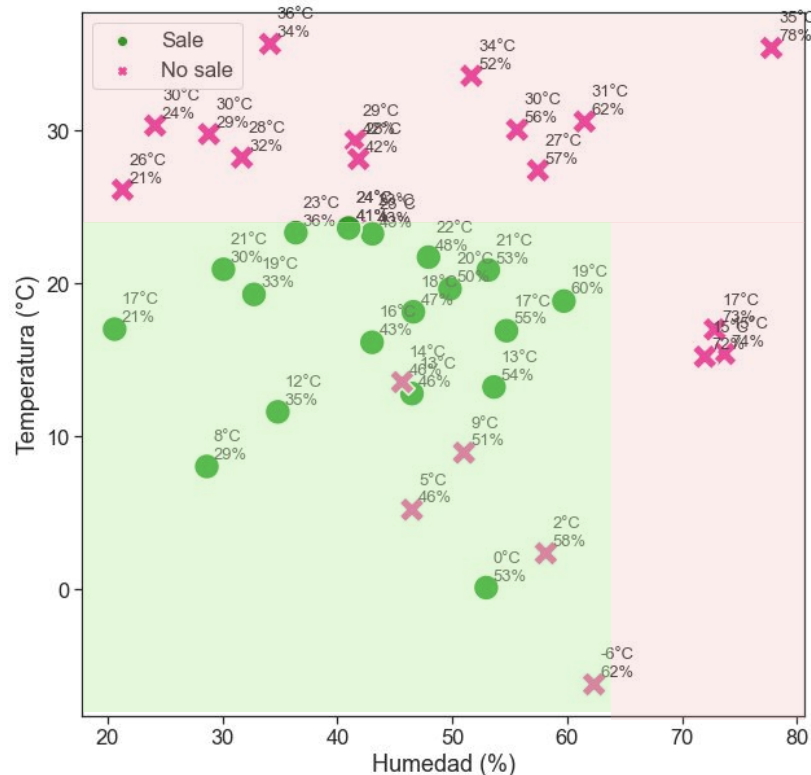
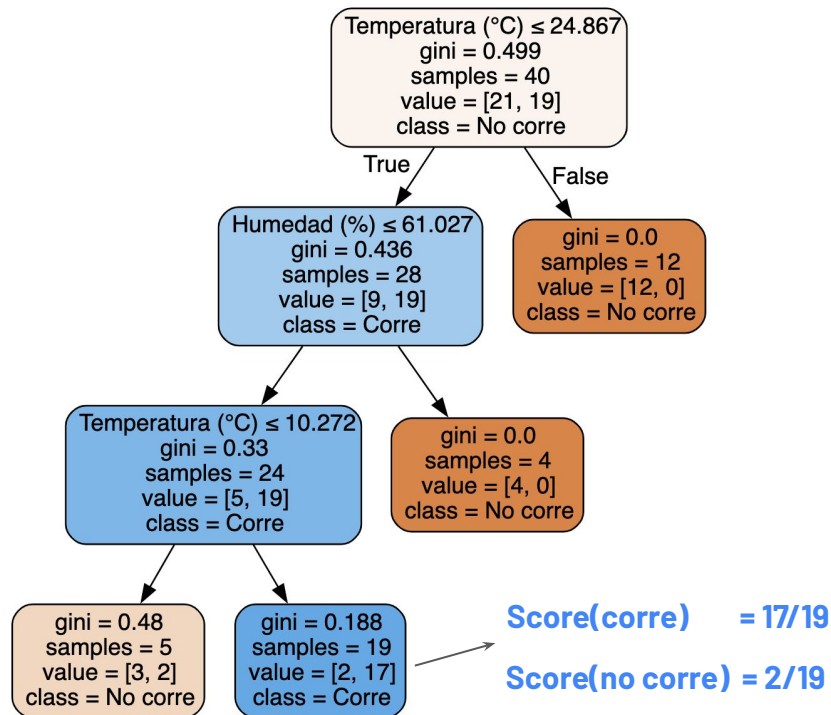
# Sesgo inductivo en árboles

1. El tipo de regiones de decisión que puede generar tiene forma de (hiper-)**rectángulos**.
2. Las regiones que exploramos se determinan de manera **Greedy**. Hill Climbing sin backtracking (converge a un máximo local)
3. Atributos **más discriminativos** → **cerca de la raíz**.
4. Árboles más pequeños y menos complejos en términos de su estructura de acuerdo al **criterio de parada** establecido.
5. De acuerdo a cómo se **exploran** los **distintos cortes** para un atributo dado, se pueden perder soluciones.

# Scores en árboles de decisión

# Asignando scores (pseudo-probabilidades)

El score asignado a una predicción podemos calcularlo como la fracción de instancias de la clase mayoritaria en cada hoja.



# Árboles de Decisión para Regresión

# Regresión

## Objetivo del aprendizaje supervisado (caso regresión)

Estimar la **función**  $f^*(\mathbf{X})$  que determina la relación  $X \rightarrow Y$ :

Es decir, estimar  $f^*$  tal que  $\mathbf{Y} \sim f^*(\mathbf{X})$  a través de un modelo

En donde  $\mathbf{Y} \in \mathbb{R}$ .

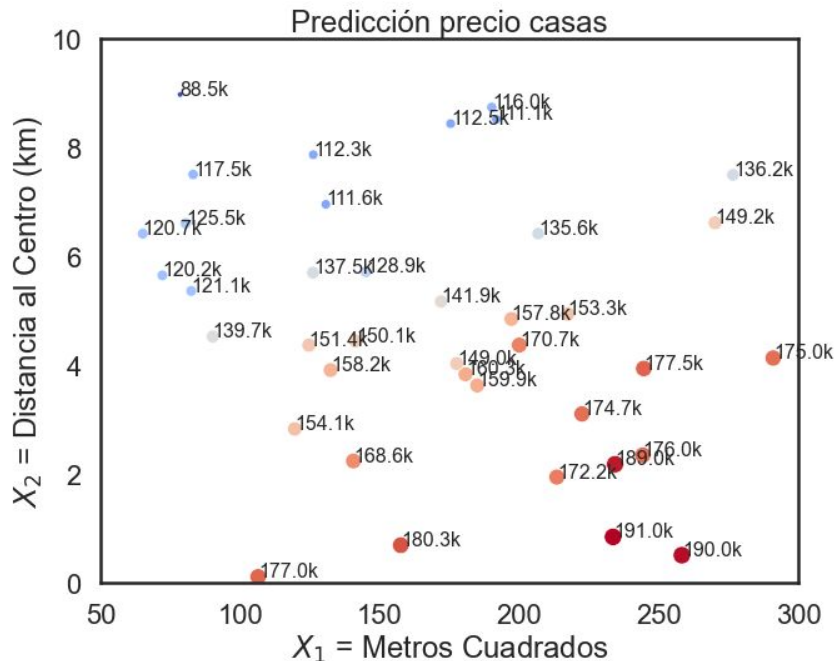
### Ejemplo

Estimar el  $Y =$  **precio de una casa** según

$X_1$  = los metros cuadrados cubiertos,

$X_2$  = su distancia al centro de la ciudad.

**Nota:** Otros atributos podrían haber sido la *cantidad de visitas al sitio de publicación* y el *color de la pintura*, el *nombre de la calle*, etc. Es decir, los atributos siguen teniendo la propiedad de ser atributos continuos, categóricos, nominales, etc, pero cambia lo que queremos predecir lo cual es determinado por **las etiquetas asociadas**.



# Ejercicio

Pensar el algoritmo para el caso de regresión.

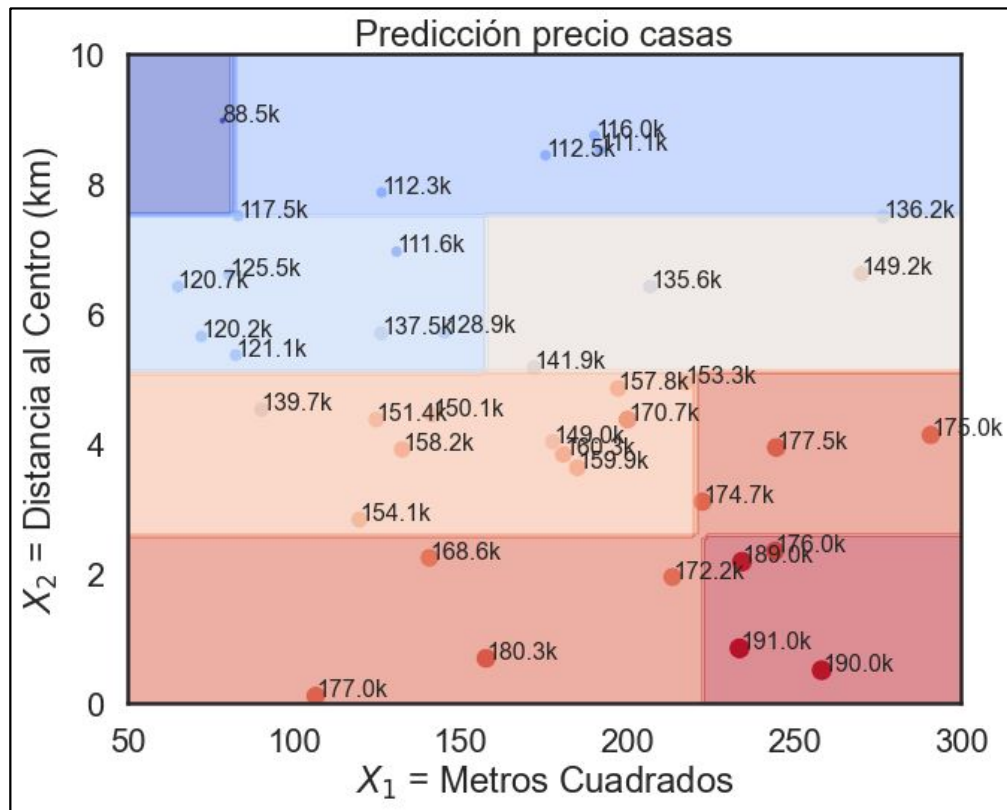
# Inducción Top-Down de Árboles de Decisión (caso regresión)

## Algoritmo CART<sup>(\*)</sup> (para atributos continuos)

Sea **S** una muestra de instancias con atributos **A**. Para construir un árbol de decisión ejecutamos:

1. Elegimos el par  $a \in A, c \in \mathbb{R}$  entre los posibles "cortes" que **mejor** divida a **S** para **nodo actual**.
2. Crear dos hijos del **nodo actual**.
3. Dividir las instancias de **S** en los nuevos nodos, según  $\langle a, c \rangle$ :  
$$S_{\leq} \leftarrow \{x \mid x \in S \wedge x[a] \leq c\}$$
$$S_{>} \leftarrow \{x \mid x \in S \wedge x[a] > c\}$$
4. **repetir** hasta cumplir algún criterio de parada (ej. Altura máxima)
5. El valor de una hoja será el promedio de las instancias de la región

¡No cambia el algoritmo! Diferencias: Medida para "homogeneidad" y valor de las hojas.  
Opción para la medida: Varianza de la región (equivale a MSE contra el promedio).



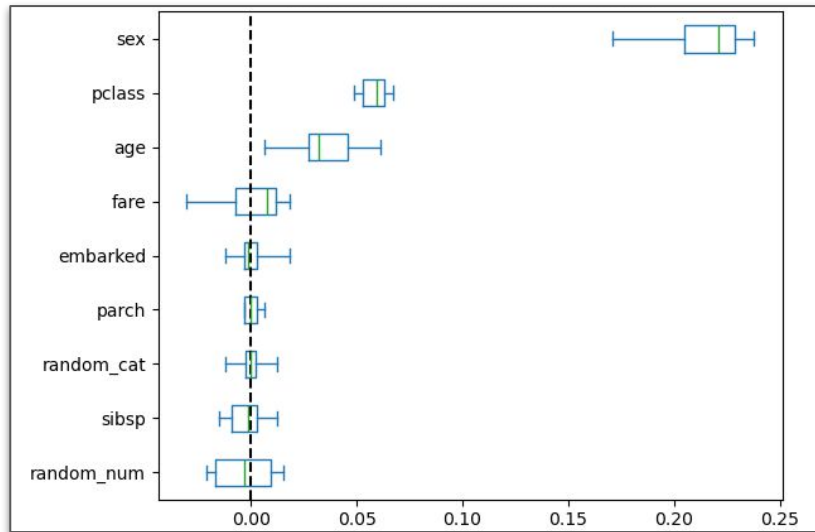


# Medidas de Importancia de atributos

# Medidas de Importancia de atributos

En los árboles de decisión, las medidas más populares para calcular la importancia de atributos incluyen:

- **Mean Decrease in Impurity** ("Gini Importance")
- **Mean Decrease in Accuracy** ("Permutation Importance")
- No veremos, pero exploren más adelante:
  - SHAP Values (<https://github.com/shap/shap>)
- etc.



Importancia de atributos  
(medidas en el test set)

# Medidas de Importancia de atributos

## Método 1: Mean Decrease in Impurity ("Gini Importance")

---

**Algorithm:** Mean Decrease in Impurity ("Gini Importance")

---

**Require:** Árbol de decisión entrenado con nodos  $N$  y atributos  $A$ .

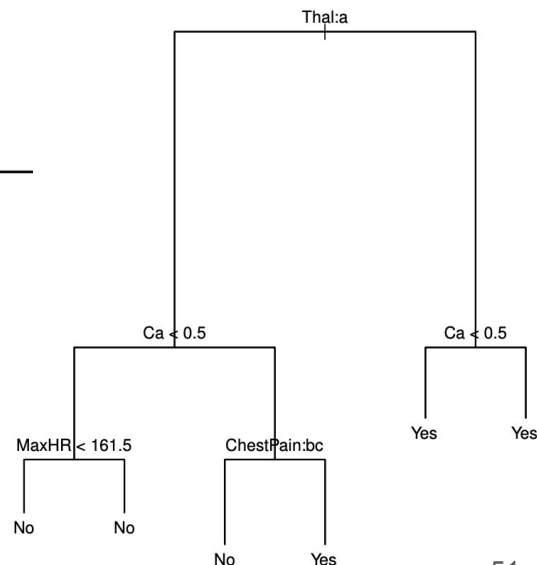
- 1:  $I_a = 0 \quad \forall a \in A$
  - 2: **for** cada  $n \in N$  **do**
  - 3:   Sea  $a$ , el atributo utilizado para decidir en el nodo  $n$
  - 4:   Sea  $w_n$  la proporción de instancias asignadas al nodo  $n$ .
  - 5:   Sumar la ganancia ponderada por la fracción de instancias en el nodo  $n$ :  
     $I_a = I_a + w_n * \Delta M(S, \langle a, c \rangle)$
  - 6: **end for**
- 

"Sumar la contribución pesada de cada atributo a lo largo de los cortes".

Si  $\Delta M(S, \langle a, c \rangle) = \text{GiniGain}(S, \langle a, c \rangle) \rightarrow$  "Gini Importance"

Si  $\Delta M(S, \langle a, c \rangle) = \text{InfoGain}(S, \langle a, c \rangle) \rightarrow$  simplemente "MDI"

En esta visualización, la **altura** de las aristas indica la información ganada en cada corte.



# Medidas de Importancia de atributos

## Método 2: Mean Decrease in Accuracy ("Permutation Importance")

**Algorithm:** Mean Decrease in Accuracy ("Permutation Importance")

**Require:** Modelo entrenado  $M$  (no necesariamente un árbol), conjunto de datos  $D$ .

- 1: Sea  $s$ : el rendimiento de  $M$  sobre  $D$ .
- 2: **for** cada atributo  $a$  **do**
- 3:     **for**  $k$  desde 1 a  $K$  **do**
- 4:         Crear  $\tilde{D}_{k,a}$ : una copia de  $D$  con columna  $a$  permutada.
- 5:         Calcular  $s_{k,a}$ : el rendimiento de  $M$  sobre  $D_{k,a}$ .
- 6:     **end for**
- 7:     La importancia de  $a$  estará dada por:  $I_a = s - \frac{1}{K} \sum_{i=1}^K s_{k,a}$ .
- 8: **end for**

Al romper la relación entre los atributos y el target, determinamos en qué medida el modelo depende de ese atributo en particular.

Una manera (**agnóstica al tipo de modelo**) de medir la importancia de atributos es evaluar el efecto de "quitar" un atributo al momento de predecir y ver cuánto empeora.

- "Quitar" (no podemos)
- **Muestrear** al azar sin reposición de la **distribución** del atributo (tampoco)
- **Permutar** la columna (aproximación a lo anterior). Sí podemos :)

Height at age 20 (cm)	Height at age 10 (cm)	...	Socks owned at age 10
182	155	...	20
175	147	...	10
...	...	...	...
156	142	...	8
153	130	...	24

# ¿Cuál utilizo?

▲ La importancia de permutación se pueden calcular en el **conjunto de entrenamiento o en un conjunto separado** ⇒ qué atributos contribuyen más al poder de **generalización** del modelo inspeccionado.

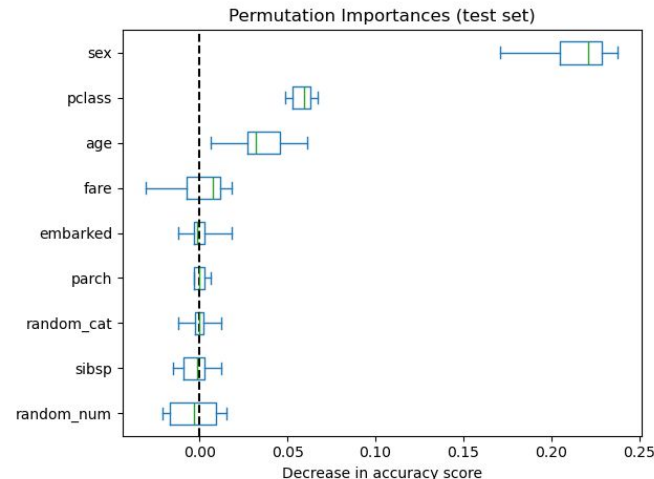
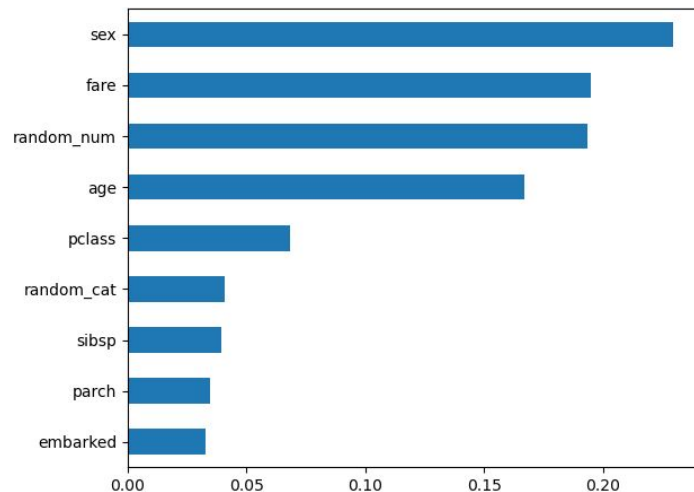
⚠ **Warning:** Los atributos que se consideran de baja importancia para un mal modelo podrían ser muy importantes para un buen modelo.

⚠ **Warning:** La importancia no refleja el valor predictivo intrínseco de un atributo en sí mismo, sino la importancia de este atributo para un modelo en particular.

Si les interesa el tema, pueden mirar

[Permutation Importance vs Random Forest Feature Importance \(MDI\) – scikit-learn](#)

No vimos SHAP, una de las técnicas más importantes de interpretabilidad. Escapa a los contenidos de la materia.



Cierre

# Ventajas y desventajas de los árboles

- ▲ Los árboles son muy **fáciles de explicar a la gente**. Son incluso más fáciles de explicar que la regresión lineal.
- ▲ Algunas personas creen que los árboles de decisión **reflejan** más fielmente la **toma de decisiones humana** que otros enfoques.
- ▲ Los árboles se pueden mostrar **gráficamente** y son fácilmente **interpretados** (si son pequeños) incluso por personas no expertas.
- ▲ Los árboles pueden manejar fácilmente **predictores cualitativos sin necesidad de crear variables ficticias**.
- ▼ Los árboles generalmente no tienen el mismo **nivel de poder predictivo** que algunos de los otros enfoques.
- ▼ Los árboles pueden ser muy **poco robustos**. Pequeños cambios en los datos pueden provocar un gran cambio en el árbol estimado final.

Sin embargo, mediante la agregación de muchos árboles de decisión, el uso de métodos como **ensambles (bagging, random forest, boosting, etc)**, el rendimiento predictivo de los árboles puede mejorarse sustancialmente. Introducimos estos conceptos en la clase 5.

# Resumen

- Árboles de decisión: construcción y consulta.
  - Para atributos discretos y numéricos.
  - Para clasificación y para regresión
- Métricas para evaluar atributos:
  - Error de clasificación (CLF)
  - Impureza Gini (CLF)
  - Ganancia de la Información (CLF)
  - Varianza (REG)
- Espacio de hipótesis. Sesgo inductivo. Complejidad temporal.
- Visualización del espacio de atributos y fronteras de decisión.
- Sobreajuste en árboles: criterios de parada; poda.
- Probabilidades en las predicciones.
- Importancia de atributos en árboles:
  - Sólo para árboles (Gini Importance)
  - Agnóstica al modelo (Permutation importance)



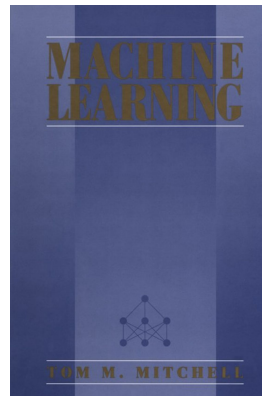
# TAREA

## Obligatorio:

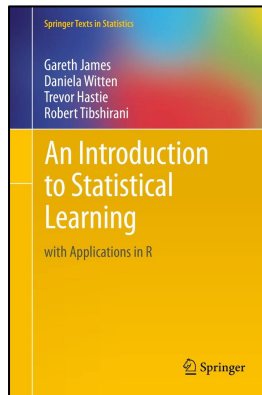
- Guía de ejercicios de árboles
- Notebooks [3 y 4](#).
- Leer [Capítulo 3](#) del "Machine Learning" (Mitchell). [Descargar](#)
- Cuestionario 2

## Opcional:

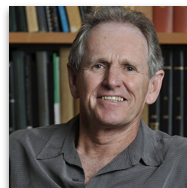
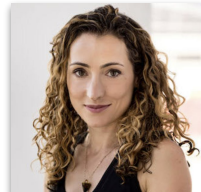
- [Sección 8.1 \(The Basics of Decision Trees\)](#) del an introduction to statistical learning: With applications in python. Springer Nature. <https://www.statlearning.com>



Tom Mitchell



Gareth James Daniela Witten



Trevor Hastie

Robert Tibshirani

# Paréntesis

## “Entropía (de Shannon)”

$$H(S) = - \sum_{k \in \text{clases}(S)} p(k) \log_2 p(k)$$

# Entropía

Sorpresa esperada

Entropía en diversas tareas:

- construcción de árboles,
- información mutua (feature selection),
- KL divergence (T-SNE, UMAP, etc),
- cross-entropy (reg logística, redes),
- etc

Generalmente usada para medir similitudes y diferencias

Mide el **nivel promedio de "información", "sorpresa" o "incertidumbre"** inherente a los posibles resultados de una variable aleatoria (\*).

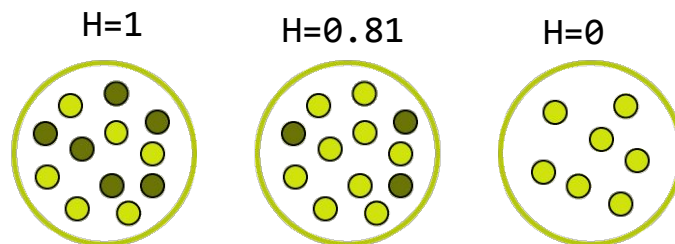
$$H(S) = - \sum_{k \in \text{clases}(S)} p(k) \log_2 p(k) = \mathbb{E}[-\log(p(k))]$$

```
# Ejemplo (python)
from math import log2

def entropia(p0, p1):
    # calcular entropía
    sorpresa_0 = p0 * log2(p0)
    sorpresa_1 = p1 * log2(p1)
    entropy = - (sorpresa_0 + sorpresa_1)
    # imprimir los resultados
    print(f'entropía: {entropy:0.2f} bits')

prop_clase_0 = 10/100
prop_clase_1 = 90/100
entropia(p0=prop_clase_0, p1=prop_clase_1)

>> entropía: 0.47 bits
```



# Entropía

Sorpresa esperada

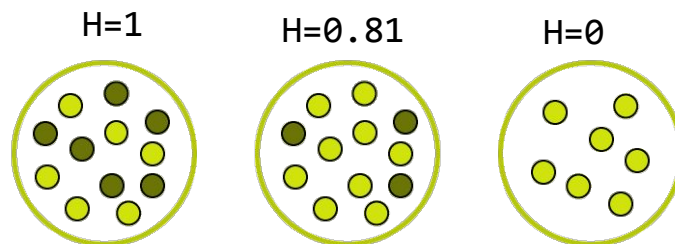
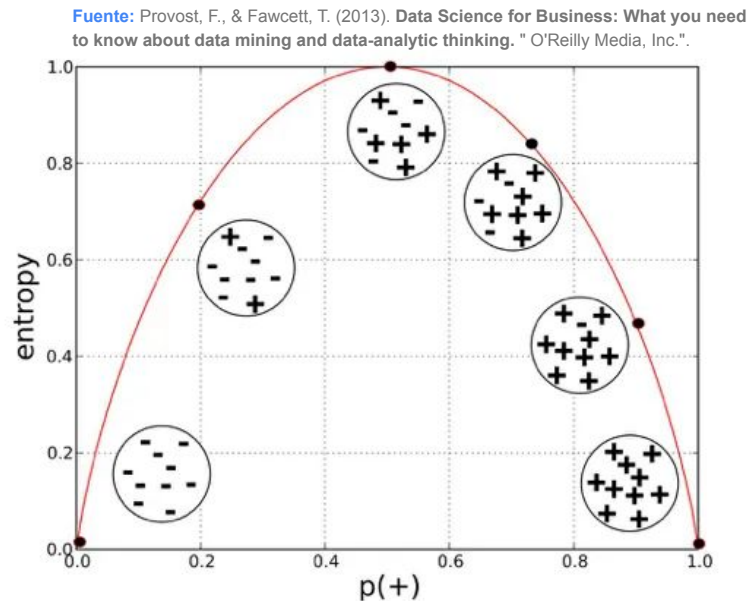
Entropía en diversas tareas:

- construcción de árboles,
- información mutua (feature selection),
- KL divergence (T-SNE, UMAP, etc),
- cross-entropy (reg logística, redes),
- etc

Generalmente usada para medir similitudes y diferencias

Mide el **nivel promedio de "información", "sorpresa" o "incertidumbre"** inherente a los posibles resultados de una variable aleatoria (\*).

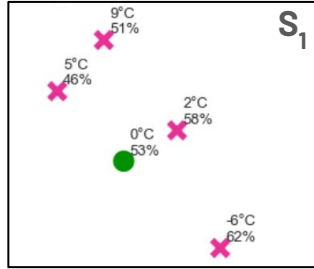
$$H(S) = - \sum_{k \in \text{clases}(S)} p(k) \log_2 p(k) = \mathbb{E}[-\log(p(k))]$$



(\*) Shannon, C. E. (1948). A mathematical theory of communication. The Bell system technical journal, 27(3), 379-423.

# Entropía

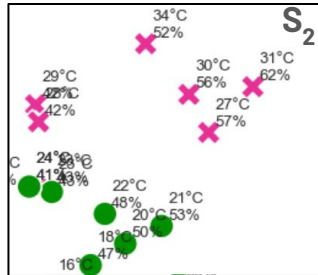
Sorpresa esperada



Cuál sería la sorpresa de, eligiendo al azar, encontrar:

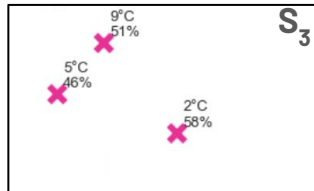
una cruz en  $S_1 \rightarrow ?$

un círculo en  $S_1 \rightarrow ?$



una cruz en  $S_2 \rightarrow ?$

un círculo en  $S_2 \rightarrow ?$

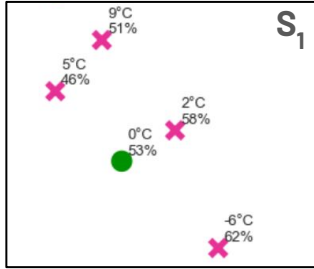


una cruz en  $S_3 \rightarrow ?$

un círculo en  $S_3 \rightarrow ?$

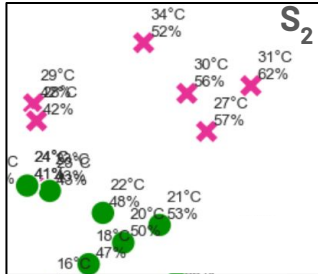
# Entropía

Sorpresa esperada

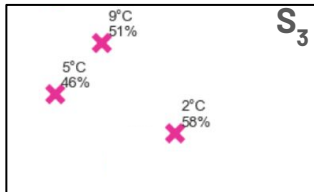


Cuál sería la sorpresa de, eligiendo al azar, encontrar:

una cruz en  $S_1 \rightarrow$  Baja  
un círculo en  $S_1 \rightarrow$  Alta



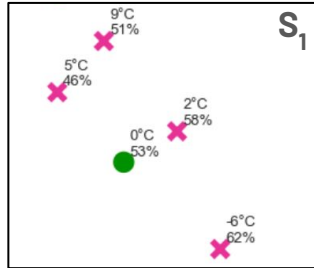
una cruz en  $S_2 \rightarrow$  Mediana  
un círculo en  $S_2 \rightarrow$  Mediana



una cruz en  $S_3 \rightarrow$  Cero  
un círculo en  $S_3 \rightarrow$  Indefinida

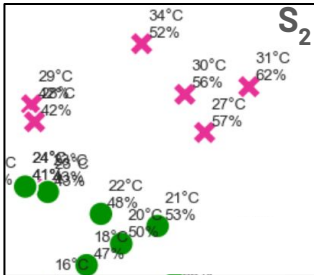
# Entropía

Sorpresa esperada

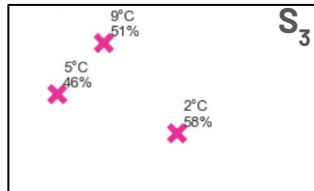


Cuál sería la sorpresa de, eligiendo al azar, encontrar:

una cruz en  $S_1 \rightarrow$  Baja  
un círculo en  $S_1 \rightarrow$  Alta



una cruz en  $S_2 \rightarrow$  Mediana  
un círculo en  $S_2 \rightarrow$  Mediana



una cruz en  $S_3 \rightarrow$  Cero  
un círculo en  $S_3 \rightarrow$  Indefinida

La sorpresa está inversamente relacionada con la probabilidad de que ocurra el evento.

¿Estará bien  $\text{sorpresa}(S, x) = 1 / \text{proba}(x)$ ?

$$\text{sorpresa}(S_1, \text{cruz}) = 1 / (4/5) = 1.25$$

$$\text{sorpresa}(S_1, \text{circ}) = 1 / (1/5) = 5$$

$$\text{sorpresa}(S_2, \text{cruz}) = 1 / (6/12) = 2$$

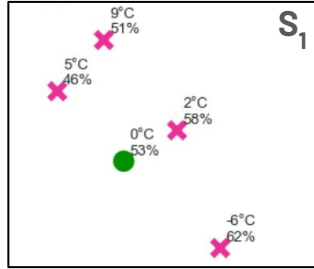
$$\text{sorpresa}(S_2, \text{circ}) = 1 / (6/12) = 2$$

$$\text{sorpresa}(S_3, \text{cruz}) = 1 / (3/3) = 1$$

$$\text{sorpresa}(S_3, \text{circ}) = 1 / (0/3) = \perp$$

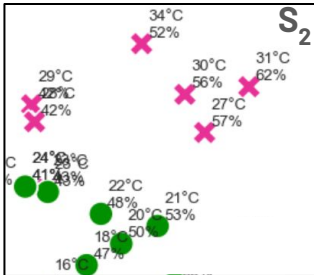
# Entropía

Sorpresa esperada

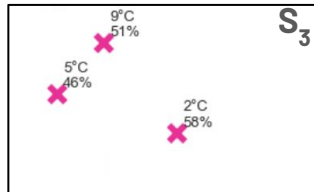


Cuál sería la sorpresa de, eligiendo al azar, encontrar:

una cruz en  $S_1 \rightarrow$  Baja  
un círculo en  $S_1 \rightarrow$  Alta



una cruz en  $S_2 \rightarrow$  Mediana  
un círculo en  $S_2 \rightarrow$  Mediana



una cruz en  $S_3 \rightarrow$  Cero  
un círculo en  $S_3 \rightarrow$  Indefinida

La sorpresa está inversamente relacionada con la probabilidad de que ocurra el evento.

$$\text{sorpresa}(S, x) = \log_2(1 / \text{proba}(x))$$

$$\text{sorpresa}(S_1, \text{cruz}) = \log_2(1 / (4/5)) = 0.32$$

$$\text{sorpresa}(S_1, \text{circ}) = \log_2(1 / (1/5)) = 2.32$$

$$\text{sorpresa}(S_2, \text{cruz}) = \log_2(1 / (6/12)) = 1$$

$$\text{sorpresa}(S_2, \text{circ}) = \log_2(1 / (6/12)) = 1$$

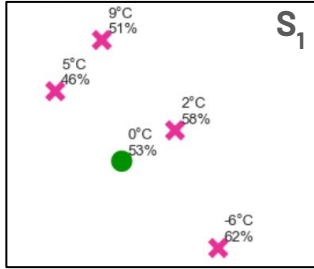
$$\text{sorpresa}(S_3, \text{cruz}) = \log_2(1 / (3/3)) = 0$$

$$\text{sorpresa}(S_3, \text{circ}) = \log_2(1 / (0/3)) = \perp$$



# Entropía

Sorpresa esperada




La sorpresa está inversamente relacionada con la probabilidad de que ocurra el evento.

$$\text{sorpresa}(S, x) = \log_2(1 / \text{proba}(x))$$

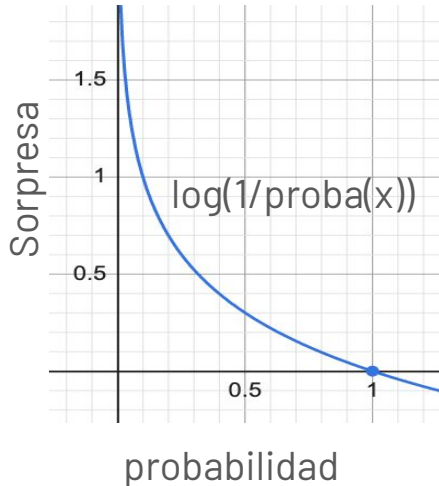
$$\text{sorpresa}(S1, \text{cruz}) = \log_2(1 / (4/5)) = 0.32$$

$$\text{sorpresa}(S1, \text{circ}) = \log_2(1 / (1/5)) = 2.32$$

¿Cuál será la sorpresa si tomamos 3 muestras y vemos:  ?

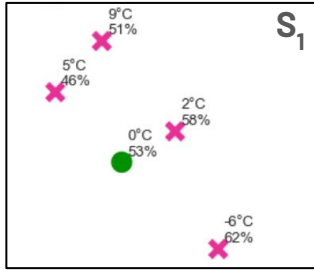
$$\begin{aligned} \text{proba}(\text{cruz} \text{ cruz} \text{ circ}) &= (4/5) * (4/5) * (1/5) \\ \text{sorpresa}(\text{cruz} \text{ cruz} \text{ circ}) &= \log_2(1 / \text{proba}(\text{cruz} \text{ cruz} \text{ circ})) \\ &= \log_2(1) - (\log_2(4/5) + \log_2(4/5) + \log_2(1/5)) \\ &= - (\log_2(4/5) + \log_2(4/5) + \log_2(1/5)) \\ &= 0.89 \end{aligned}$$

Convenientemente, la sorpresa de eventos independientes es la suma de las sorpresas de cada evento.



# Entropía

Sorpresa esperada



La sorpresa está inversamente relacionada con la probabilidad de que ocurra el evento.

$$\text{sorpresa}(S, x) = \log_2(1 / \text{proba}(x))$$

¿Cuál será la sorpresa **total** esperada si tomamos 100 muestras?

$$\text{sorpresa}_{-100}(S) = \text{proba}(\text{●}) * 100 * \text{sorpresa}(\text{●}) + \text{proba}(\text{✕}) * 100 * \text{sorpresa}(\text{✕})$$

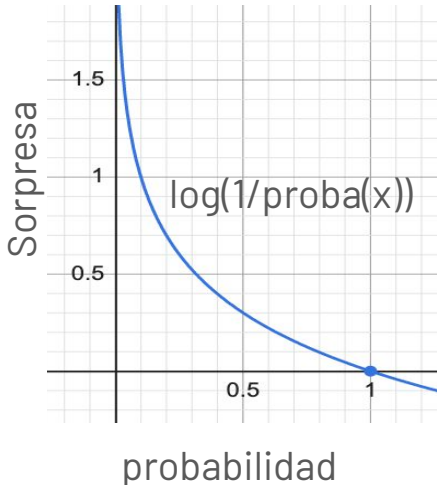
¿Y la sorpresa esperada **promedio** para 100 muestras?

$$\text{sorpresa}_{-100} / 100 = \text{proba}(\text{●}) * \text{sorpresa}(\text{●}) + \text{proba}(\text{✕}) * \text{sorpresa}(\text{✕})$$

$$\mathbb{E}[\text{sorpresa}(S)] = \sum_{k \in \text{clases}(S)} \text{sorpresa}(k) * p(k) \quad \text{Y esta es la } \mathbf{\text{entropía}} \text{ de la muestra } \mathbf{S}$$

$$= \sum_{k \in \text{clases}(S)} \log_2\left(\frac{1}{p(k)}\right) * p(k)$$

$$= - \sum_{k \in \text{clases}(S)} p * \log_2(p(k))$$



# Cerramos paréntesis

Recomendamos leer MacKay, D. J. (2003). Information theory, inference and learning algorithms, pág. 32