



DEPARTAMENTO
DE COMPUTACION

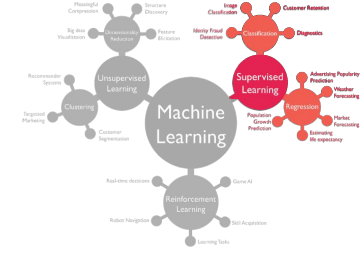
Facultad de Ciencias Exactas y Naturales - UBA

Aprendizaje Automático

Clase 2:

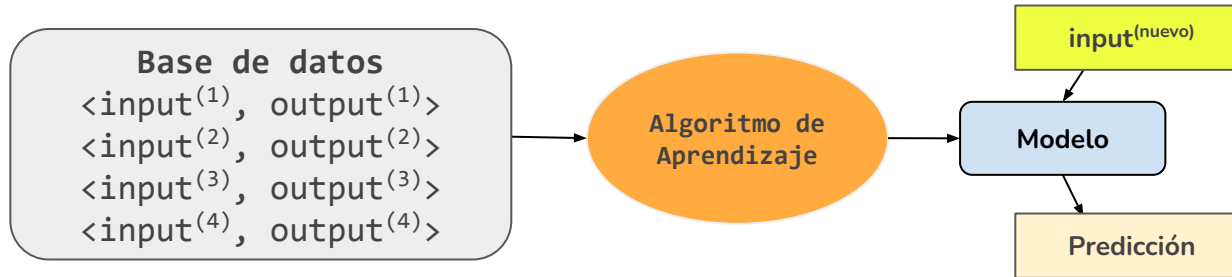
Árboles de decisión

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

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

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

¿De dónde sale y?

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

¿Y qué es $error(\mathbf{A}, \mathbf{B})$?

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

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

Sesgo Inductivo

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 utiliza 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

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

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}

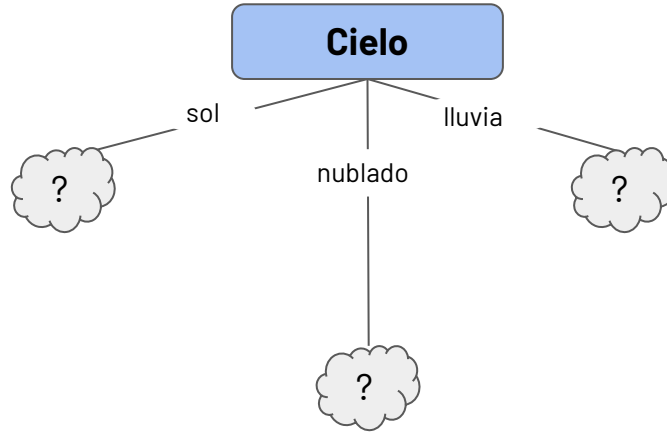
Por lo tanto, crearemos la función **Camina?**, que quedaría:

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.

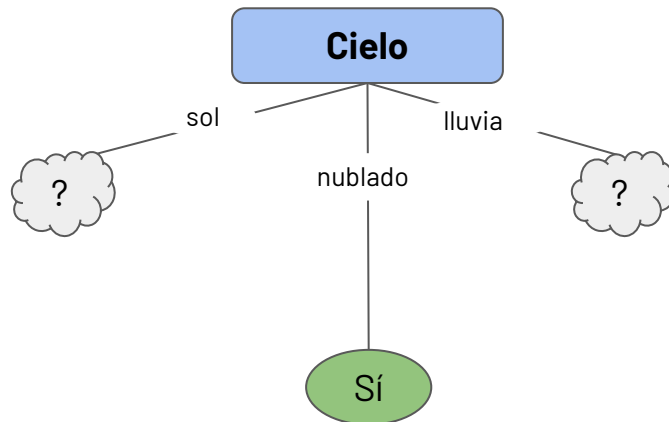
atributos				clase
Cielo	Temperatura	Humedad	Viento	¿Camina?
Sol	Calor	Alta	Débil	No
Sol	Calor	Alta	Fuerte	No
Nublado	Calor	Alta	Débil	Sí
Lluvia	Templado	Alta	Débil	Sí
Lluvia	Frío	Normal	Débil	Sí
Lluvia	Frío	Normal	Fuerte	No
Nublado	Frío	Normal	Fuerte	Sí
Sol	Templado	Alta	Débil	No
Sol	Frío	Normal	Débil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

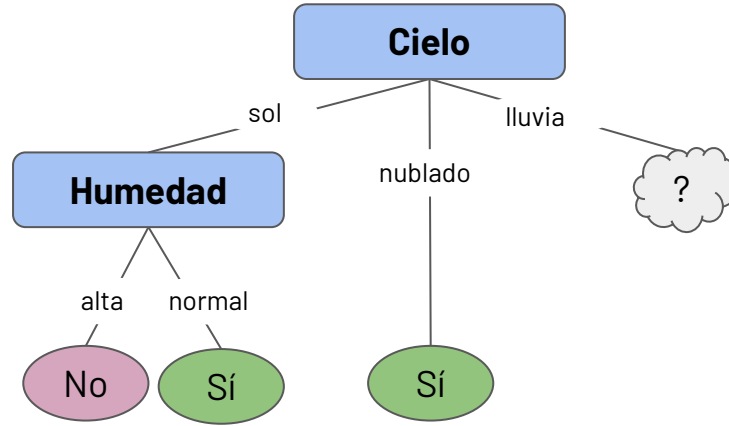
Construyendo un árbol de decisión



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

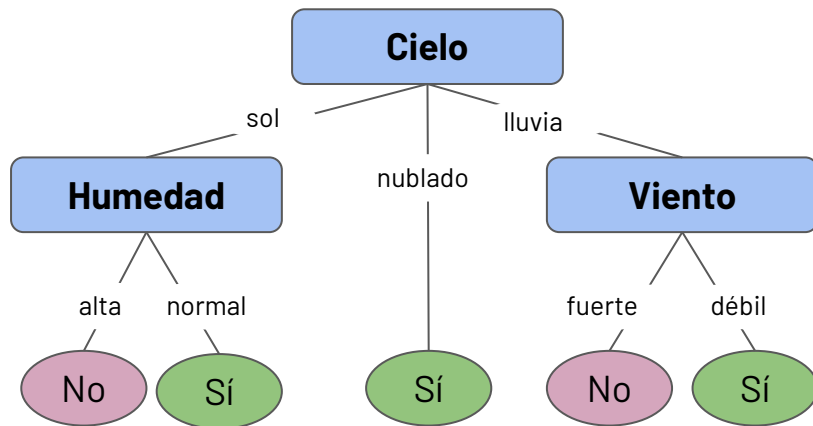
Las instancias con **Cielo == Nublado** son todas positivas.

Construyendo un árbol de decisión



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

Construyendo un árbol de decisión



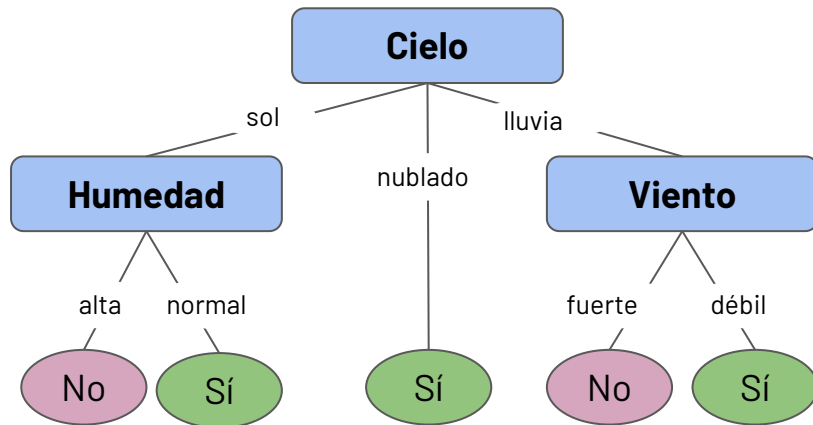
Este árbol representa la función **Camina?** :: Cielo x Temperatura x Humedad x Viento \rightarrow {Sí, No}

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**?

¿Hay sesgo inductivo?

Aproximación de Funciones: Ejemplo 2

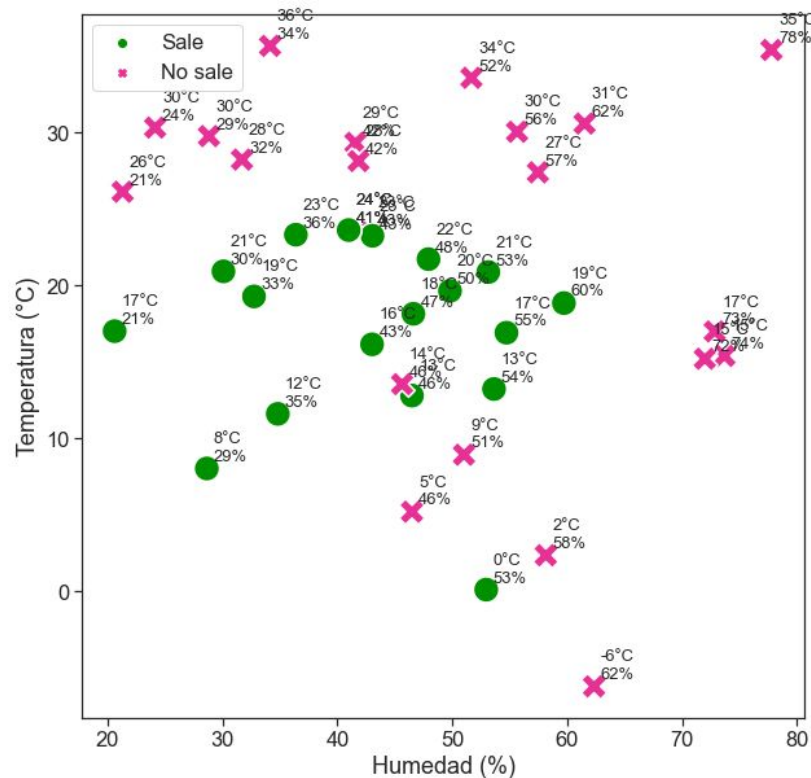
Imaginemos ahora que sólo tenemos acceso a las siguientes características del día:

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

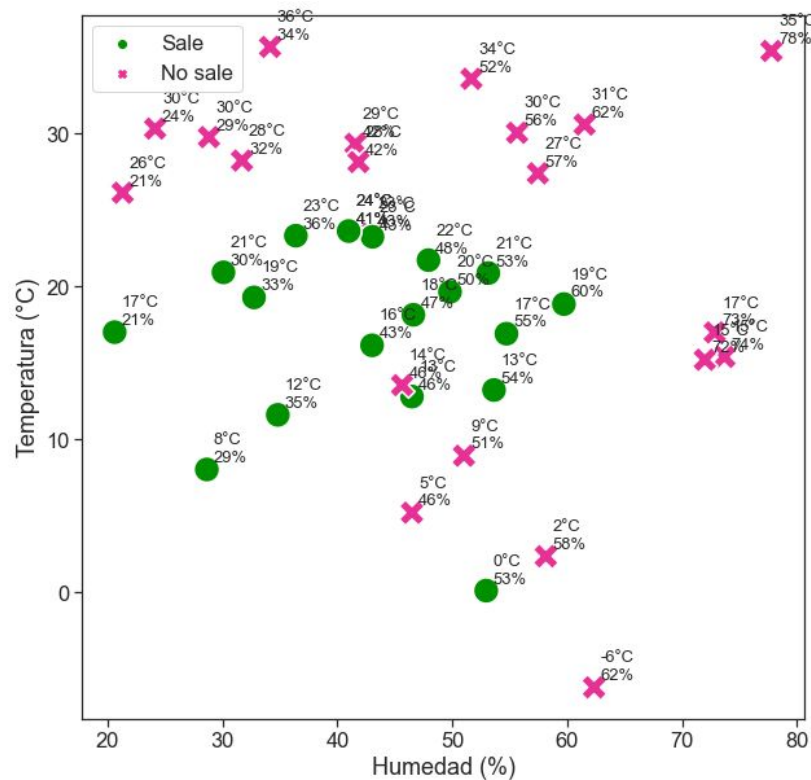
Por lo tanto, crearemos la función **Caminar** quedaría:

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

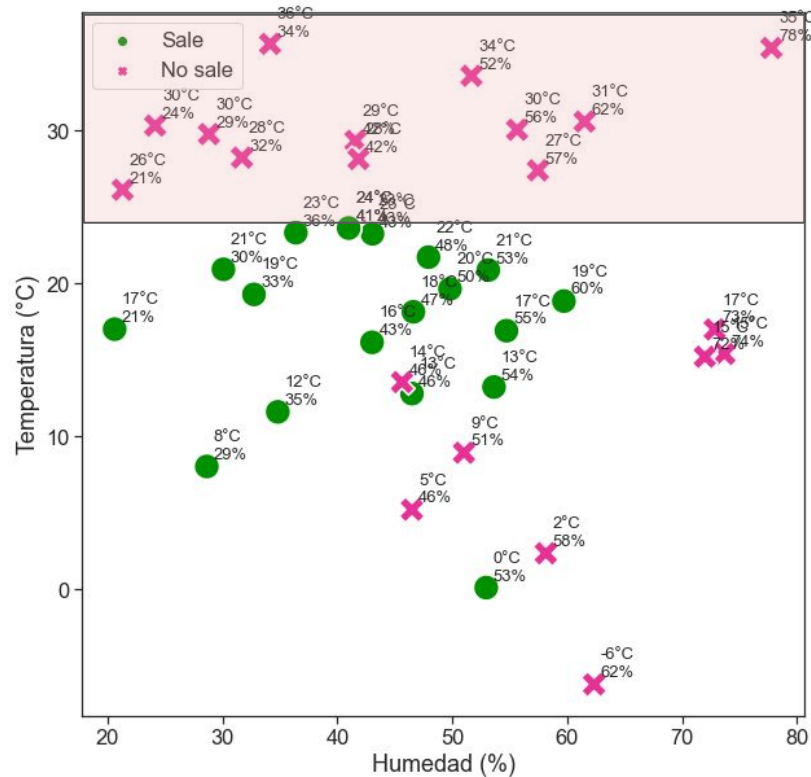
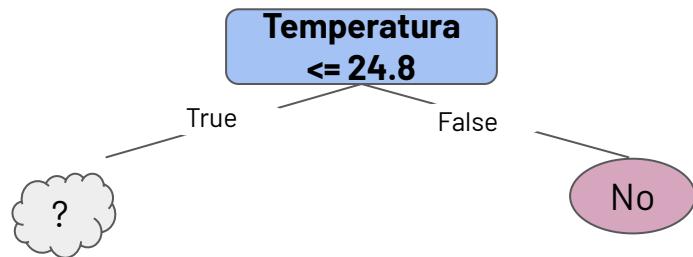
Por lo que empezamos por juntar datos, registramos el comportamiento del vecino durante unas semanas y armamos el siguiente gráfico.



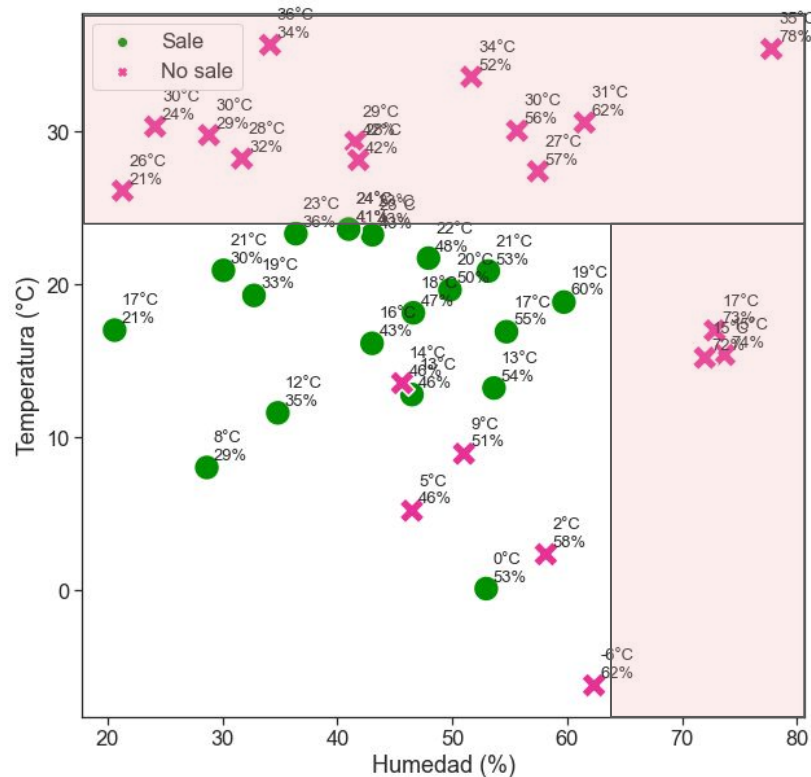
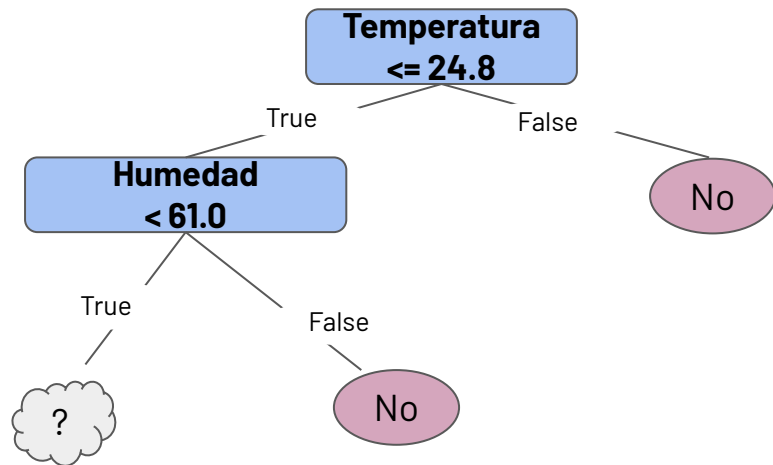
Construyendo un árbol de decisión



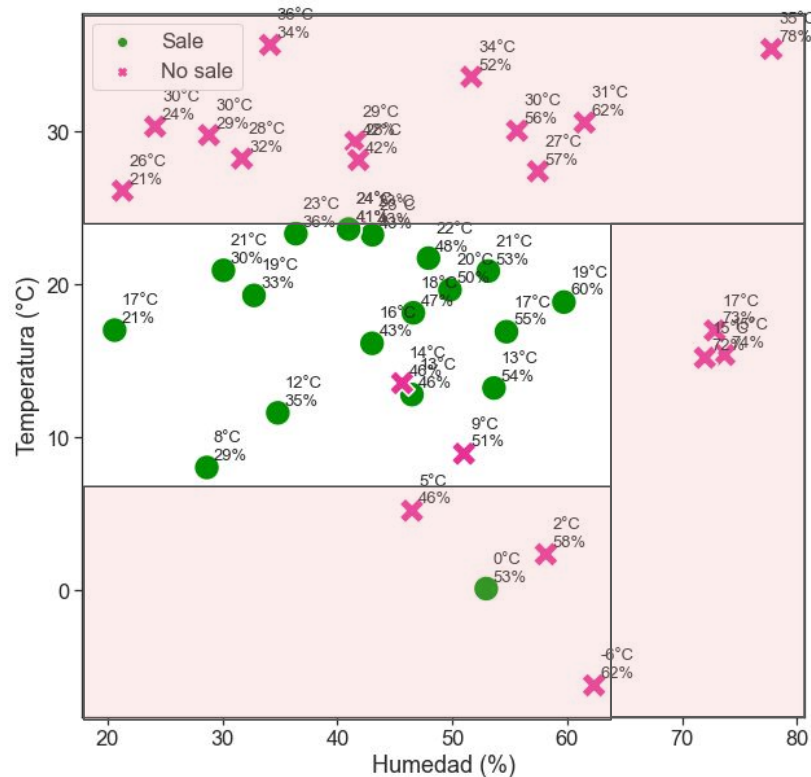
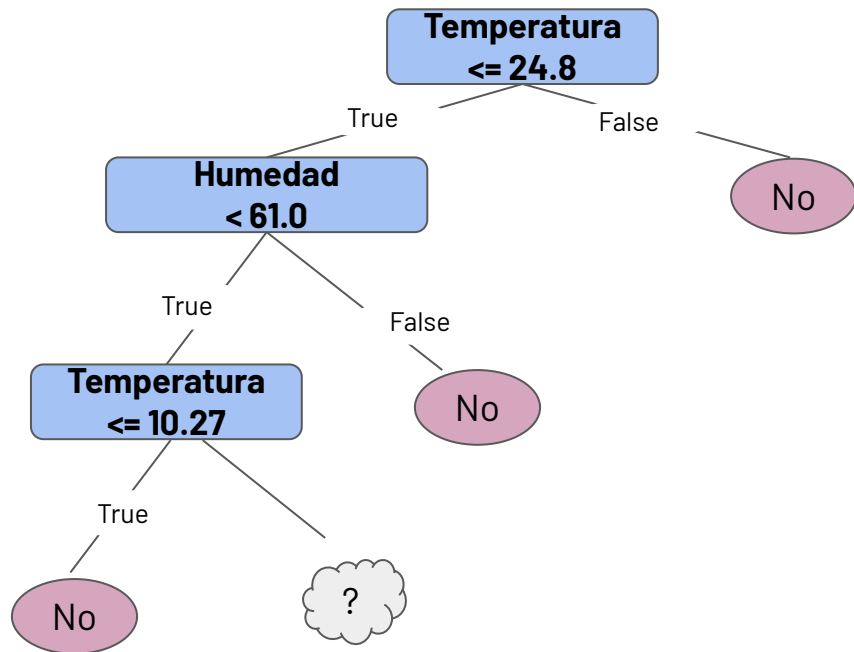
Construyendo un árbol de decisión



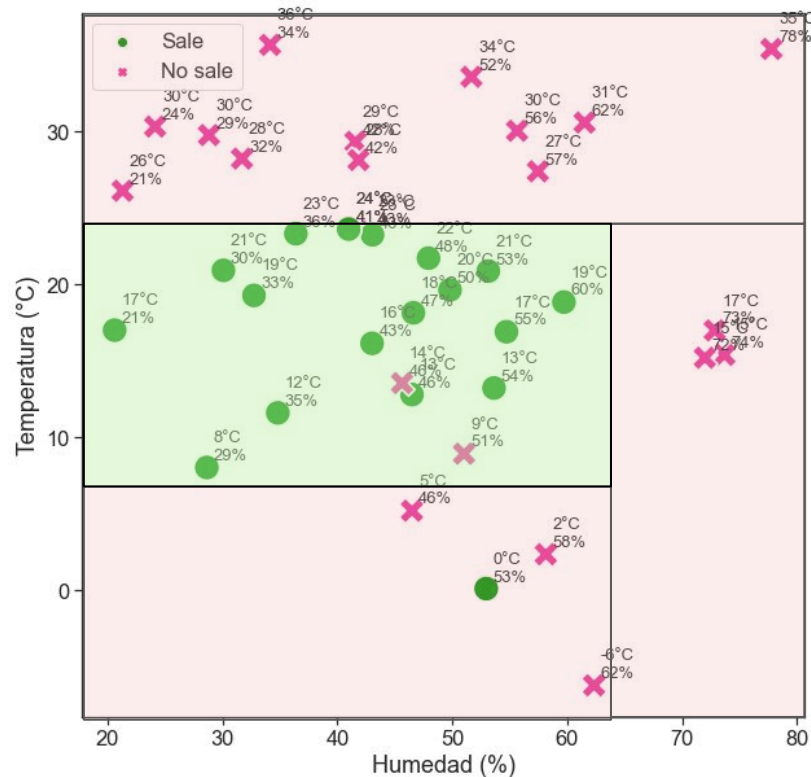
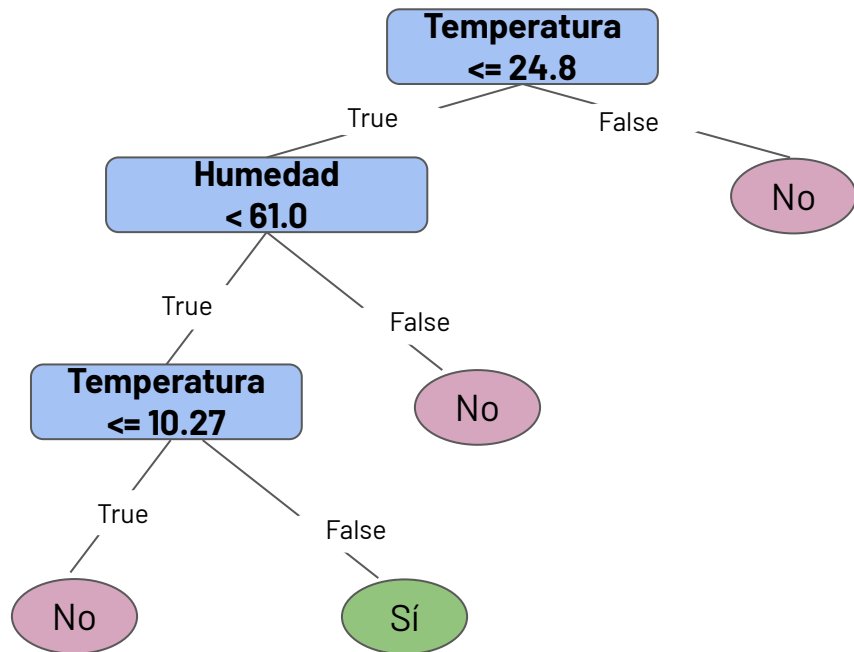
Construyendo un árbol de decisión



Construyendo un árbol de decisión



Construyendo un árbol de decisión

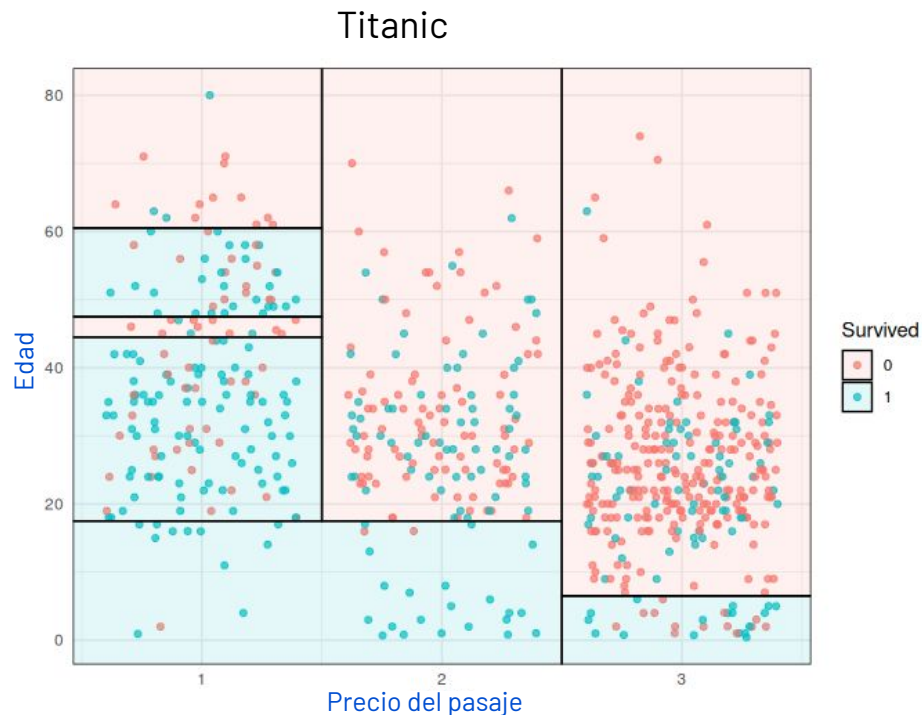


Visualización de fronteras de decisión ("Decision Boundaries")

Fronteras de decisión: Regiones del espacio de características 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 características donde se asigna una etiqueta de clase.



Encontrar la mejor partición

Idea de algoritmo

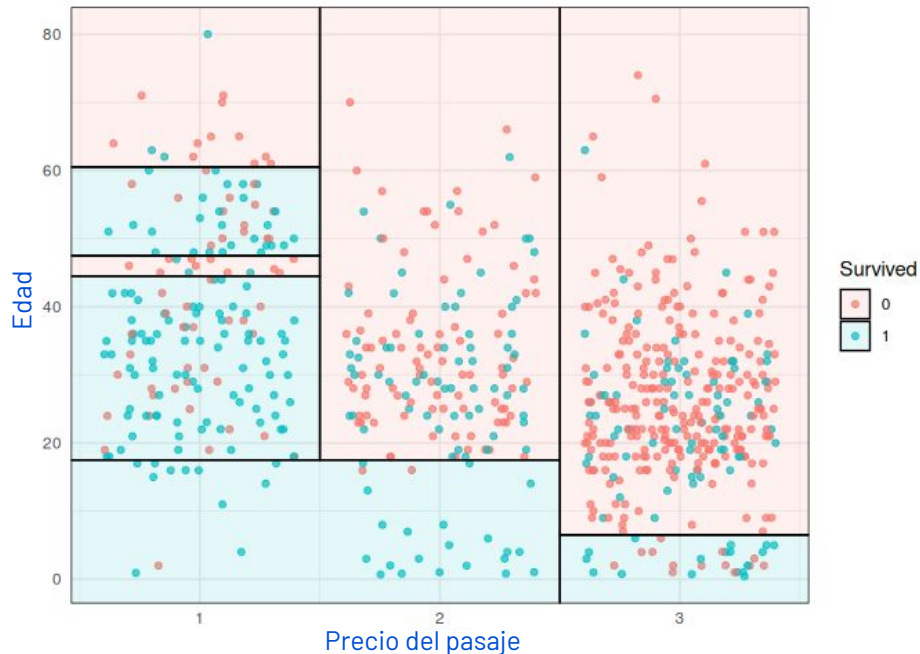
Buscar **la mejor partición** en regiones (hiper-)rectangulares / jerárquicas, en el espacio de atributos.

¿Qué complejidad computacional tendría?

Buscar todas las particiones en (hiper-)rectángulos posibles y testear cada solución es **NP-Hard** (tan difícil como el halting problem) \Rightarrow Impracticable.

Esto se parece mucho a lo que vimos la clase pasada (el problema de recorrer espacios tan grandes de hipótesis).

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



Inducción Top-Down de Árboles de Decisión

Algoritmos ID3^(*) y C4.5^(**), Quinlan (para atributos discretos)^(***)

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*⁽¹⁾ divide a S como nodo de decisión para *nodo actual*.
2. Para cada valor *posible*⁽²⁾ 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 para cada hijo en el que haya instancias de más de una clase definiendo $A_i \leftarrow A - \{a\}$.

⁽¹⁾ ¿Mejor?

⁽²⁾ ¿Conocemos todos los posibles valores que puede tomar cada atributo?

^(*) 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]

Inducción Top-Down de Árboles de Decisión

Algoritmo CART^(*) (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⁽¹⁾ pares que mejor⁽²⁾ 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 para cada hijo en el que haya instancias de más de una clase⁽³⁾.

⁽¹⁾ Son infinitos.. ¿Cuántos y cuáles evalúo?

⁽²⁾ Seguimos con el problema de "mejor".

⁽³⁾ ¿Por qué no quitamos a para los nodos hijos de *nodo_actual*?

^(*) Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). Cart. *Classification and regression trees*.

Ejercicio: Costo computacional y sesgo inductivo

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

- Construcción: ??
- Consulta de nuevas instancias (inferencia): ??

¿Cuál es el sesgo inductivo hasta ahora?

Costo computacional y Sesgo Inductivo

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

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

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

¿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?)

^(a) P. E. Utgoff. "Incremental induction of decision trees". Machine Learning, 4(2):161-186, **1989**

^(b) J. W. Shavlik, R. J. Mooney, and G. Towell. "Symbolic and neural learning algorithm: An experimental comparison". Machine Learning, 6:111-143, **1991**.

¿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”?

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 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)

Recordar

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

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

- $\text{Prop} = \#(S_x) / \#(S)$ la proporción de instancias en la nueva región.
- $M(S)$ una **medida** que nos indique la “homogeneidad” de la región. Ya veremos algunas opciones.

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

Métricas de homogeneidad de una región

Tasa de error de clasificación

En regresión, veremos que MSE (mean square error) de la región es una buena métrica a optimizar (a minimizar).

Primera idea entonces: **error de clasificación**.

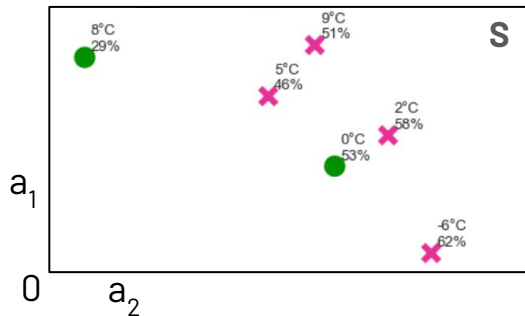
Tasa de error de clasificación (Classification Error Rate)

Podemos definir nuestra métrica de “la calidad de una región” como:

CER(S) = 1 - max_{k ∈ 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.

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



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

Métricas de homogeneidad de una región

Tasa de error de clasificación

En regresión, veremos que MSE (mean square error) de la región es una buena métrica a optimizar (a minimizar).

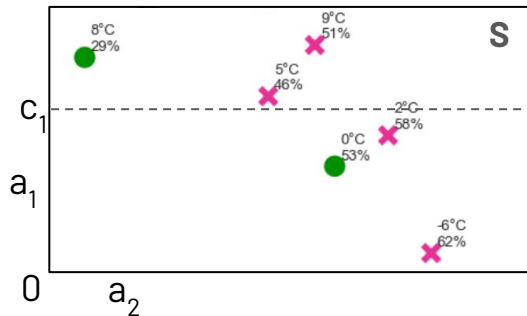
Primera idea entonces: **error de clasificación**.

Tasa de error de clasificación (Classification Error Rate)

Podemos definir nuestra métrica de “la calidad de una región” como:

CER(S) = 1 - max_{k∈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.



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

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

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

Métricas de homogeneidad de una región

Tasa de error de clasificación

En regresión, veremos que MSE (mean square error) de la región es una buena métrica a optimizar (a minimizar).

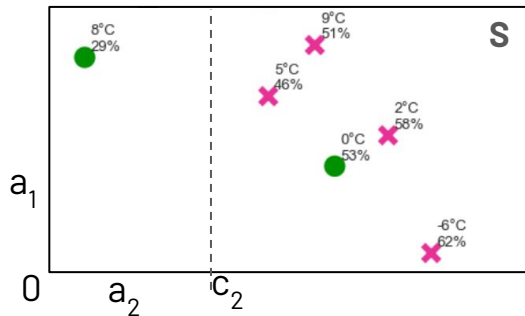
Primera idea entonces: **error de clasificación**.

Tasa de error de clasificación (Classification Error Rate)

Podemos definir nuestra métrica de “la calidad de una región” como:

CER(S) = 1 - max_{k∈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.



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

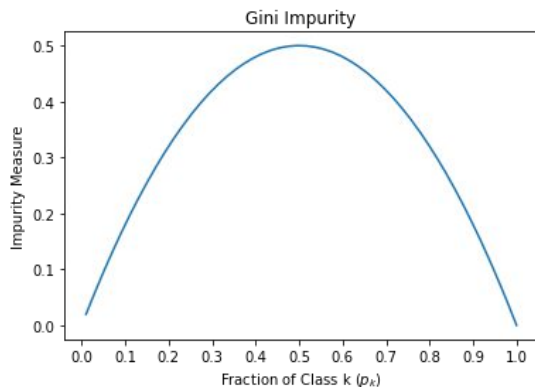
$$\begin{aligned} \Delta CER(S, < a_2, c_2 >) &= \frac{2}{6} - (Prop_{\leq} * CER(S_{\leq}) + Prop_{>} * CER(S_{>})) \\ &= \frac{2}{6} - (\frac{1}{6} * CER(S_{\leq}) + \frac{5}{6} * CER(S_{>})) \\ &= \frac{2}{6} - (\frac{1}{6} * 0 + \frac{5}{6} * \frac{1}{5}) \\ &= \frac{1}{6} \quad (\text{un mejor corte})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

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

Métricas de homogeneidad de una región

Ganancia Gini (reducción de Impureza)

$$Gini(S) = 1 - \sum_{k \in clases(K)} (p_s(k))^2$$



```
def gini(S):
    label_counts = count_labels(S)
    # Devuelve un dict de label → cuenta. Ej {"circulo" : 10, "cruz": 20}
    n = len(S)

    res = 1
    for label, count in label_counts.items():
        label_prob = count / n
        res -= label_prob ** 2

    return res
```

La Impureza de Gini mide la **probabilidad de que una instancia particular sea clasificada erróneamente** si esta fuese etiquetada aleatoriamente siguiendo la distribución de clases dentro de la región

$$Gini_Gain(S, < a, c >) = Gini(S) - (Prop_{\leq} \cdot Gini(S_{\leq}) + Prop_{>} \cdot Gini(S_{>}))$$

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

Métricas de homogeneidad de una región

Ganancia de información (reducción de entropía)

Definamos ahora **Ganancia de información**: **Cuánta entropía removemos** al hacer un corte.

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

Entropía de una región S:

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

Abrimos 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 (*).

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

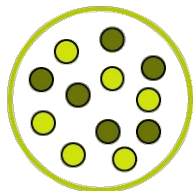
```
# Ejemplo (python)
from math import log2

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')

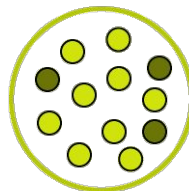
prop_clase_0 = 10/100
prop_clase_1 = 90/100
entropia(p0=prop_clase_0, p1=prop_clase_1)

>> entropía: 0.47 bits
```

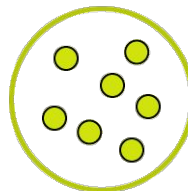
H=1



H=0.81



H=0



Entropía

Sorpresa esperada

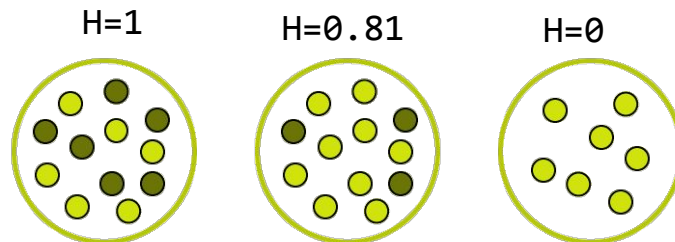
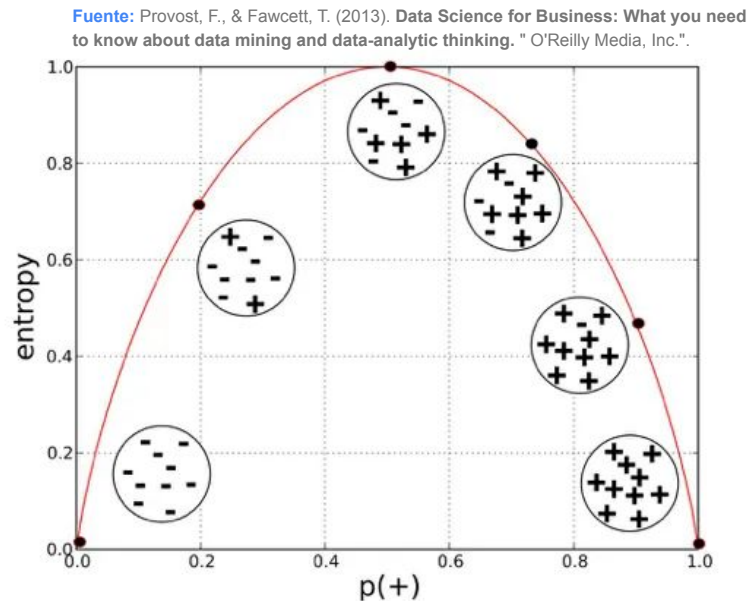
Entropía en diversas tareas:

- 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

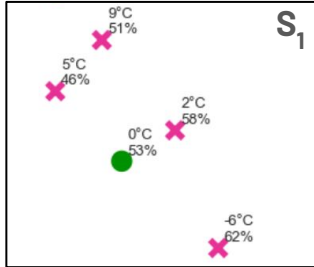
Mide el **nivel promedio de "información", "sorpresa" o "incertidumbre"** inherente a los posibles resultados de una variable aleatoria (*).

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



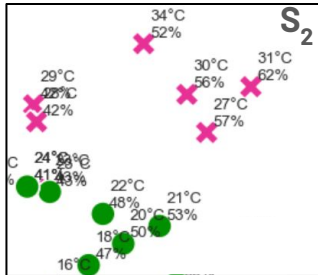
Entropía

Sorpresa esperada

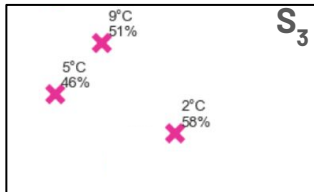


Cuál sería la sorpresa de, eligiendo al azar, encontrar:

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



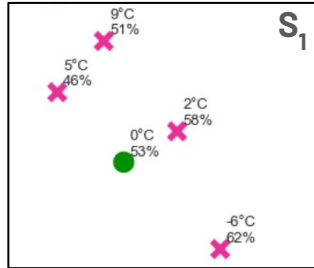
una cruz en $S_2 \rightarrow$ Mediana
un círculo en $S_2 \rightarrow$ Mediana



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

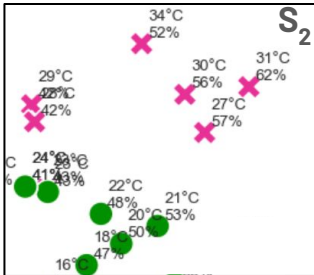
Entropía

Sorpresa esperada

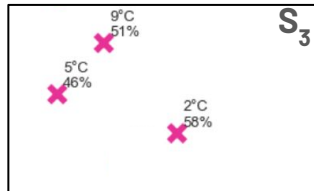


Cuál sería la sorpresa de, eligiendo al azar, encontrar:

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

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

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

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

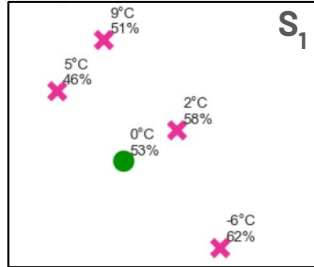
$$\text{sorpresa}(S_2, \text{circ}) = 1 / (6/12) = 2$$

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

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

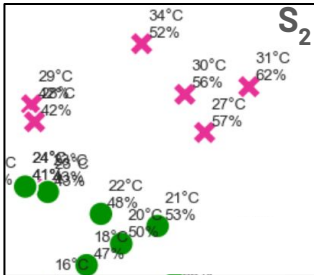
Entropía

Sorpresa esperada

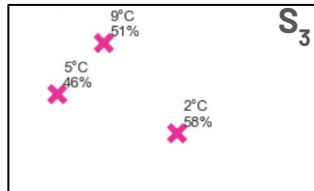


Cuál sería la sorpresa de, eligiendo al azar, encontrar:

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.

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

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

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

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

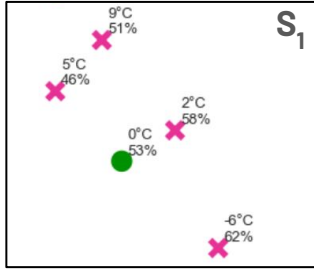
$$\text{sorpresa}(S_2, \text{circ}) = \log_2(1 / (6/12)) = 1$$

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

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

Entropía

Sorpresa esperada






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

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

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

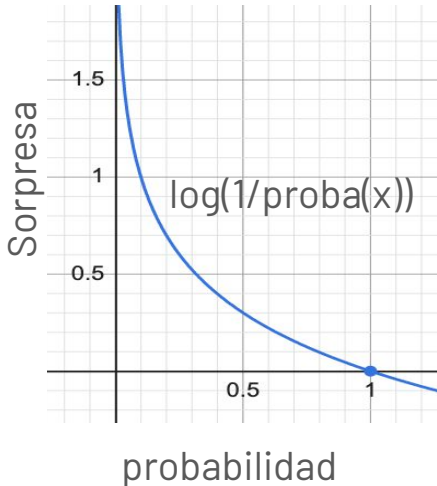
$$\text{sorpresa}(S1, \text{circ}) = \log_2(1 / (1/5)) = 2.32$$

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

$$\text{proba}(\text{pink cross} \text{ pink cross} \text{ green circle}) = (4/5) * (4/5) * (1/5)$$

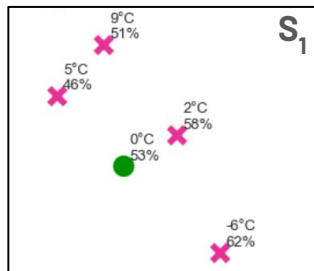
$$\begin{aligned} \text{sorpresa}(\text{pink cross} \text{ pink cross} \text{ green circle}) &= \log_2(1 / \text{proba}(\text{pink cross} \text{ pink cross} \text{ green circle})) \\ &= \log_2(1) - (\log_2(4/5) + \log_2(4/5) + \log_2(1/5)) \\ &= -(\log_2(4/5) + \log_2(4/5) + \log_2(1/5)) \\ &= 0.89 \end{aligned}$$

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?

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

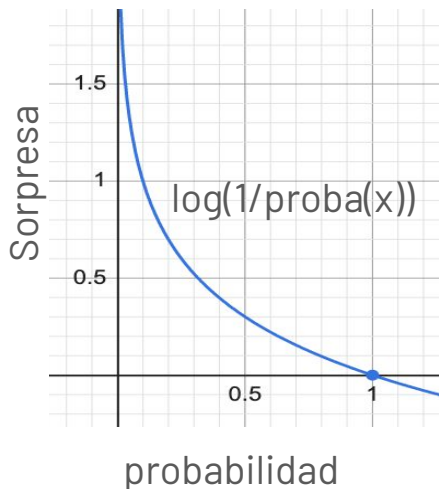
¿Y la sorpresa esperada promedio para 100 muestras?

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

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

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

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



Cerramos paréntesis

Métricas de homogeneidad de una región

Ganancia de información (reducción de entropía)

Volviendo. Definamos ahora **Ganancia de información**: **Cuánta entropía removemos** al hacer un corte.

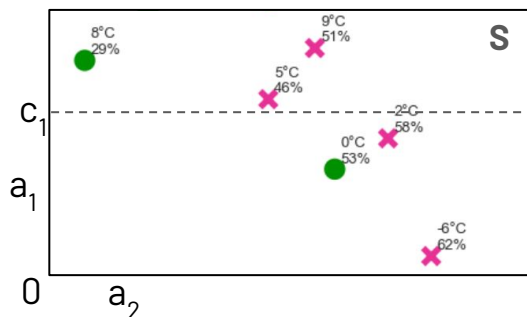
$$InfoGain(S, \langle a, c \rangle) = H(S) - (Prop_{\leq} \cdot H(S_{\leq}) + Prop_{>} \cdot H(S_{>}))$$

Y recordando que medimos la ganancia en un corte según

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

Definimos $\Delta M(S, \langle a, c \rangle) = InfoGain(S, \langle a, c \rangle)$

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



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

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

Métricas de homogeneidad de una región

Ganancia de información (reducción de entropía)

Volviendo. Definamos ahora **Ganancia de información**: **Cuánta entropía removemos** al hacer un corte.

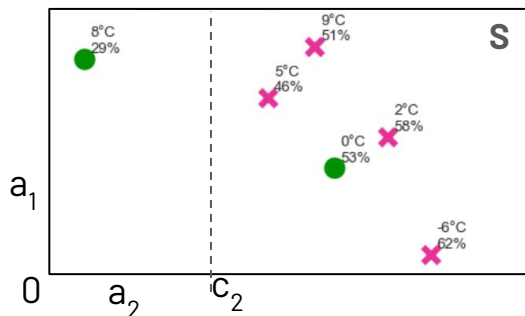
$$InfoGain(S, < a, c >) = H(S) - (Prop_{\leq} \cdot H(S_{\leq}) + Prop_{>} \cdot H(S_{>}))$$

Y recordando que medimos la ganancia en un corte según

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

Definimos $\Delta M(S, < a, c >) = InfoGain(S, < a, c >)$

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



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

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

Resumen hasta el momento

Necesitamos evaluar la calidad de un corte.

Para ello definimos la ganancia según distintas métricas:

- Ganancia en tasa de clasificación
- Ganancia de información (reducción de entropía de Shanon).
- Ganancia Gini (reducción de Impureza Gini)

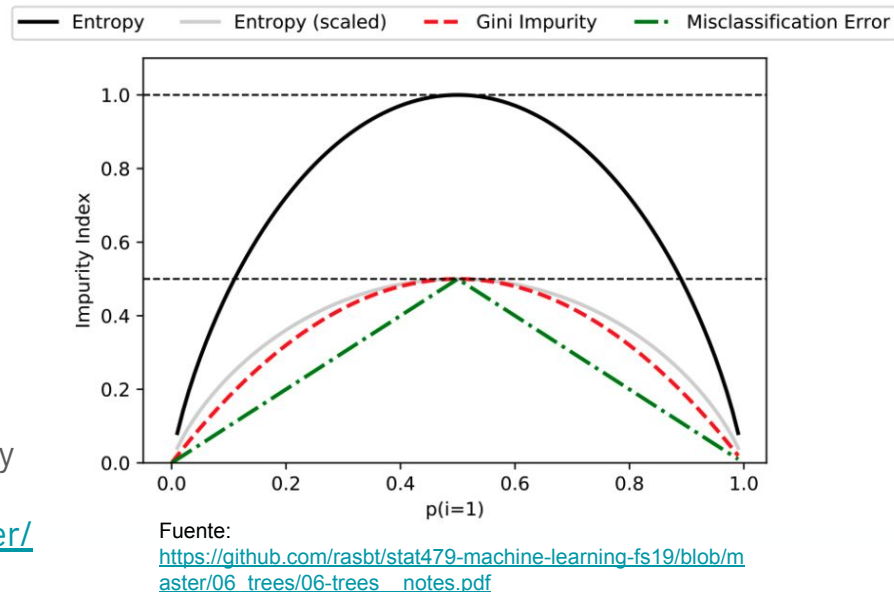
La **ganancia de información** está definida en función de la **entropía**.

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

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



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

Causas: (discutir)

¿Cómo evitarlo?

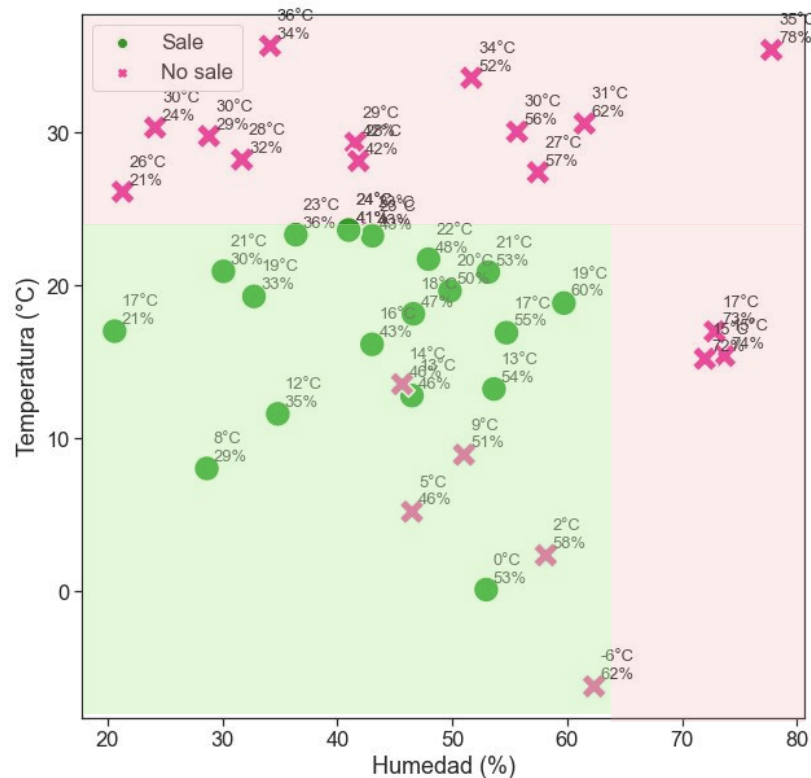
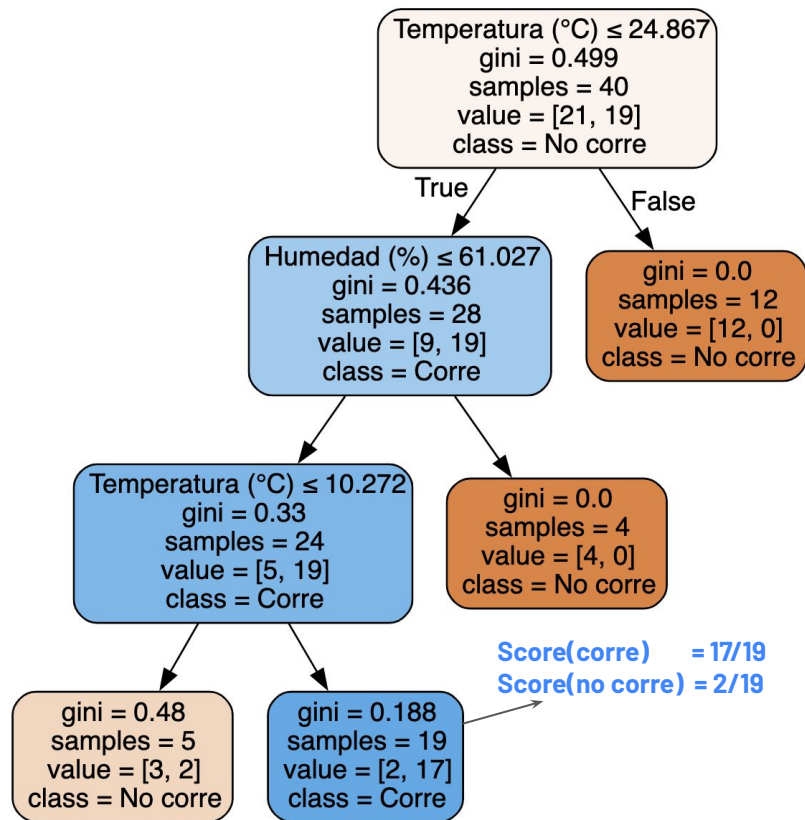
- **Criterio de parada:** No construir más allá de cierta profundidad.
- **Pruning (poda):** Construir el árbol entero; podar las ramas cuando ello mejore la performance sobre datos separados. Funciona mejor en la práctica.
- **Rule post-pruning:** Construir el árbol entero; convertir árbol a reglas; 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.

Asignando scores (pseudo-probabilidades)

El score asignado a una predicción está dado por la fracción de instancias de la clase mayoritaria en cada hoja.



Medidas de Importancia de atributos

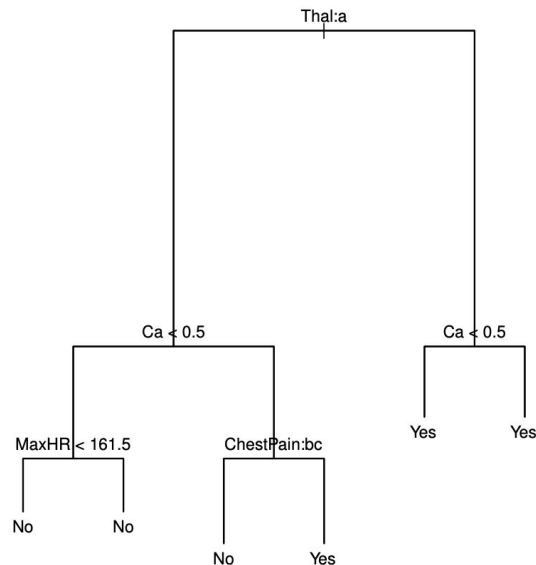
Medidas de Importancia de atributos

Importancia basada en impurezas (método para árboles)

Una manera directa es analizar la estructura del árbol resultante:

1. Construir el árbol.
2. Para cada nodo, analizar la **ganancia** que se obtuvo al dividir el nodo según el atributo seleccionado.
3. **Combinar** (sumar) los valores de ganancia de información (o gini) para cada atributo para obtener una medida general.
4. Escalar a $[0,1]$ (dividir por la suma total)

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



Medidas de Importancia de atributos

Importancia de Permutación (método agnóstico al modelo)

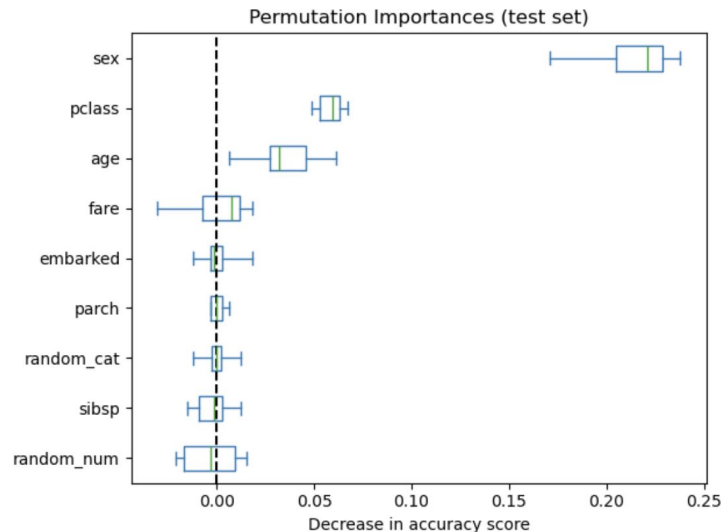
Una manera (**agnóstica al algoritmo**) de medir la importancia de atributos es evaluar el efecto de quitar un atributo al momento de predecir y ver cuánto empeora. Pero.. ¿quitar cómo?

Algorithm: Importancia de Permutación

Require: Modelo entrenado M , conjunto de datos D .

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



¿Cuál utilizo?

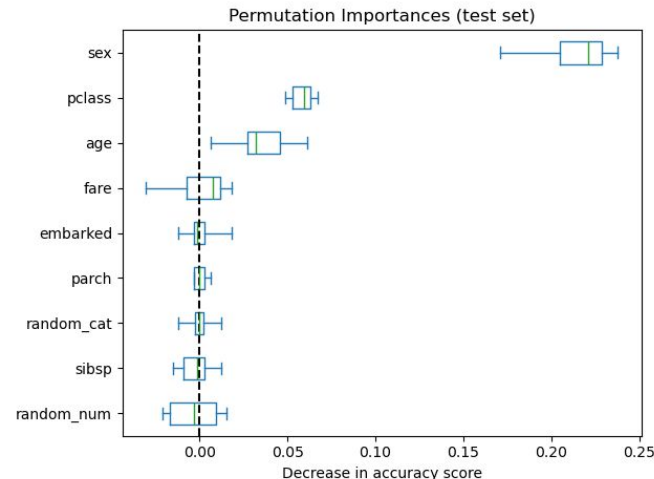
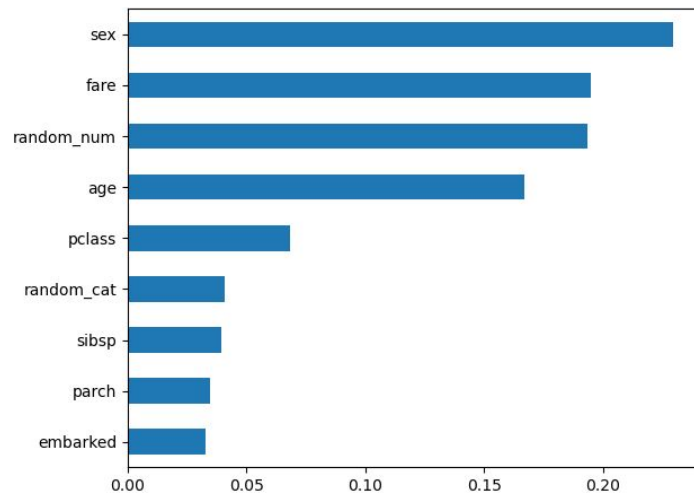
▲ La importancia de permutación se pueden calcular en el **conjunto de entrenamiento o en un conjunto separado** ⇒ qué atributos contribuyen más al poder de **generalización** del modelo inspeccionado.

⚠ **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](#)



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

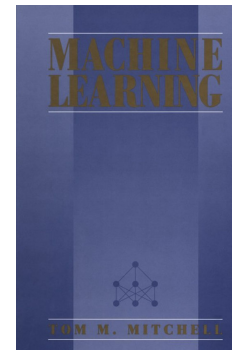
- Árboles de decisión: construcción y consulta.
- Métricas para evaluar atributos: Error de clasificación, Impureza Gini y Ganancia de la Información.
- Espacio de hipótesis. Sesgo inductivo. Complejidad temporal. Atributos discretos y numéricos.
- Visualización del espacio de atributos y fronteras de decisión.
- Sobreajuste en árboles: criterios de parada; poda.
- Obtener probabilidades en las predicciones.
- Importancia de atributos en árboles e importancia agnóstica al modelo.
- Ventajas y desventajas de estos modelos.

TAREA

- Empezar a resolver la guía de ejercicios de árboles
- Empezar a leer **Capítulo 3** del “Machine Learning” (Mitchell).

[Descargar](#)

(el cuestionario lo largamos el martes recién)



Tom Mitchell