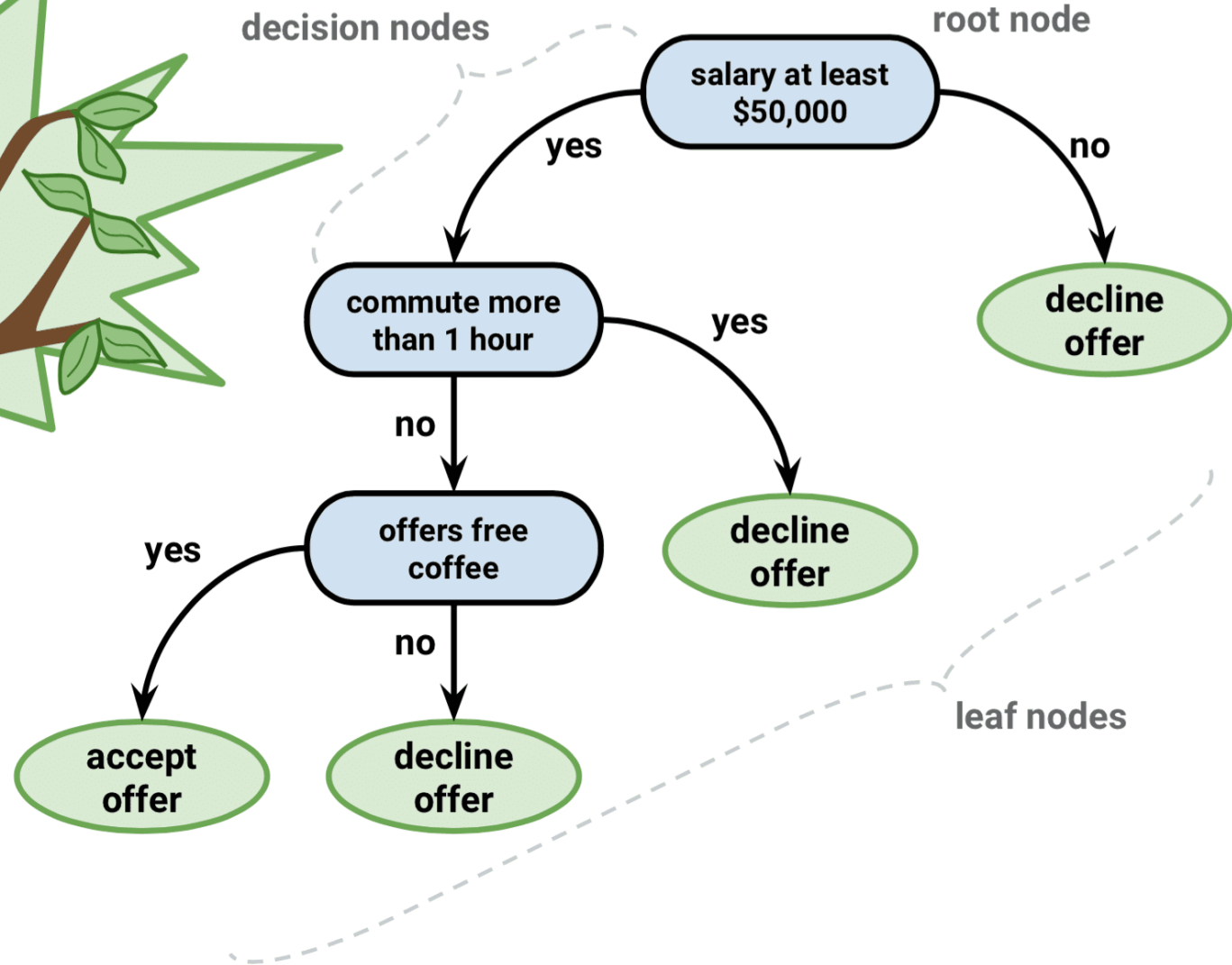


# Árboles de clasificación

CLAUDIA CHÁVEZ

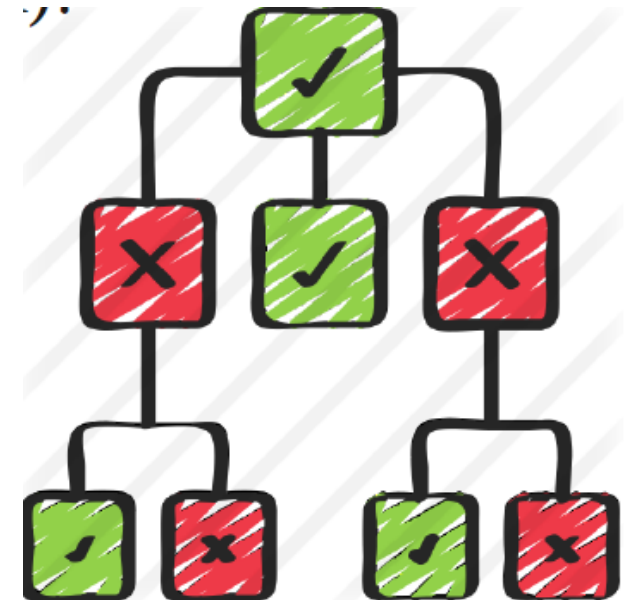
**Decision Tree:**  
**Should I accept a new  
job offer?**



# ¿Qué es un árbol de decisión?

- Método no paramétrico de clasificación (también hay de regresión).

El objetivo es crear un modelo que prediga el valor de una variable objetivo, en este caso una clase, mediante el aprendizaje de reglas de decisión simples inferidas a partir de las features de los datos.



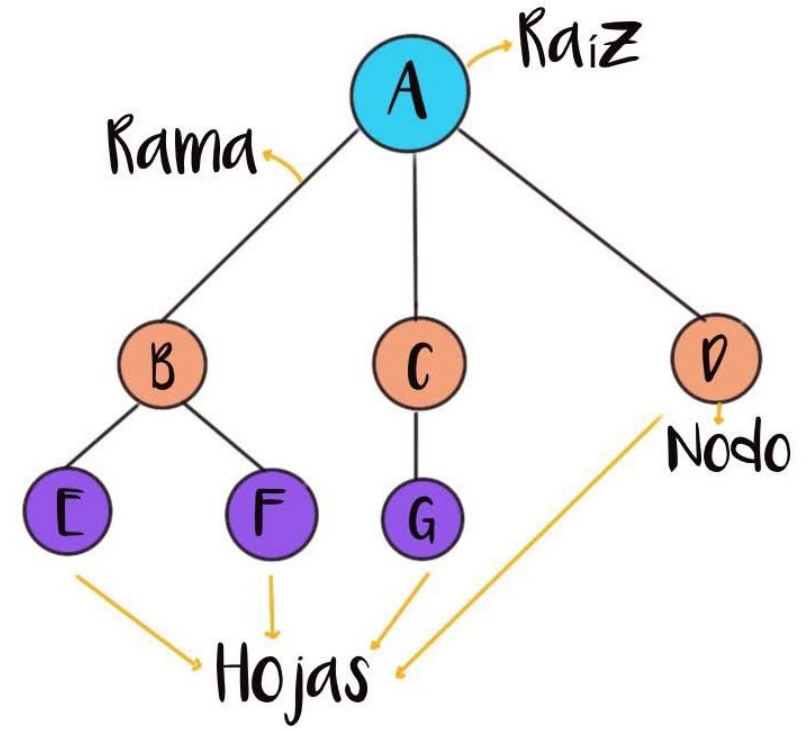
# ¿Dónde podría ser convenientes los árboles de decisión?

- ▶ Predicción de la deserción estudiantil
- ▶ Renovación de una suscripción o no
- ▶ Qué marca de auto comprará un cliente
- ▶ Qué falla es la más repetitiva en distintos tipos de motores



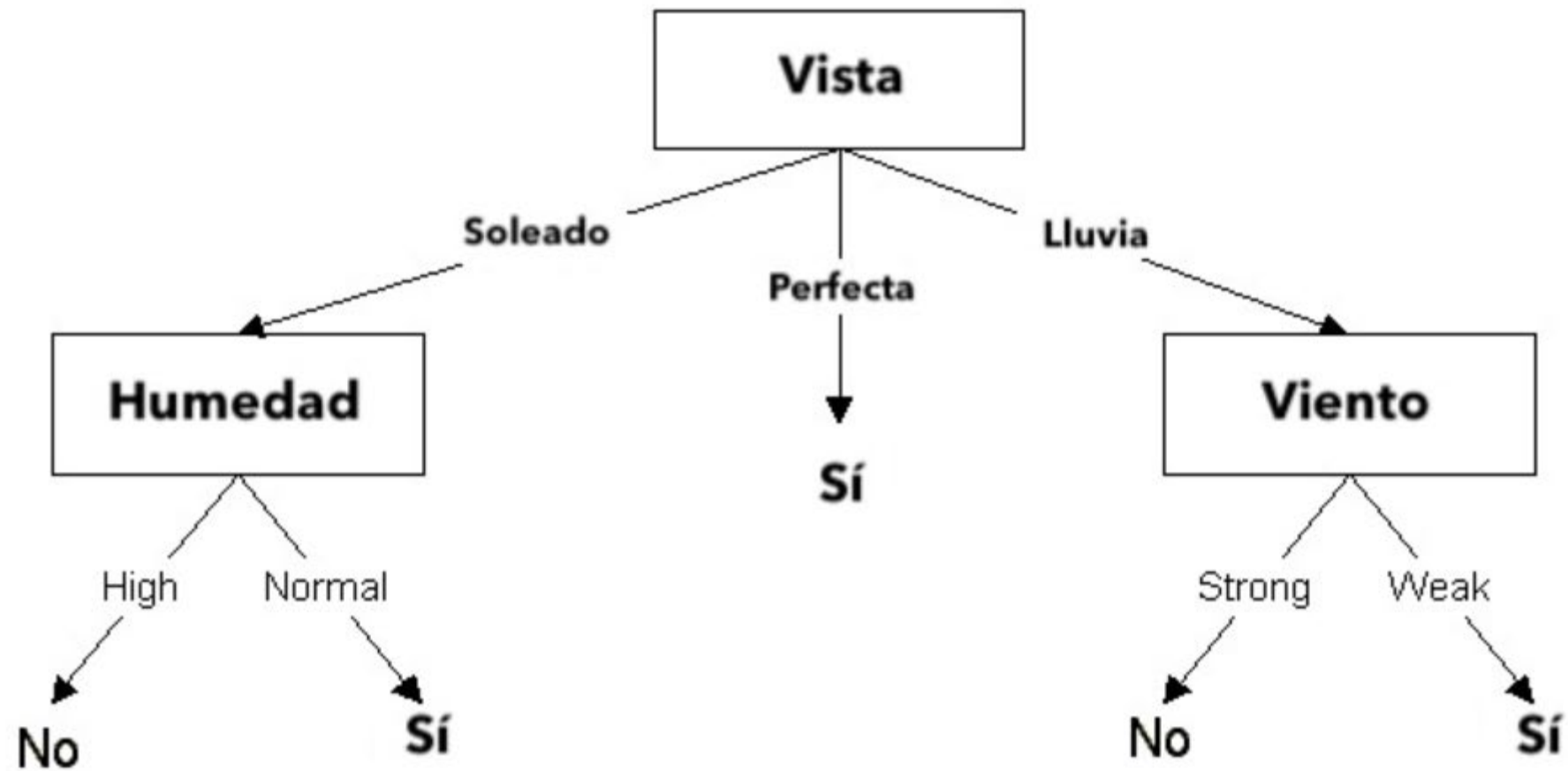
# Componentes de un árbol de decisión

- ▶ Tienen nodos, ramas y hojas
- ▶ Cada rama conecta un nodo, la profundidad del árbol se mide acorde al número de ramas que este tenga
- ▶ El nodo raíz genera la primera partición
- ▶ Los nodos intermedios también se denominan nodos hijos
- ▶ Las hojas corresponden a los nodos terminales del árbol, no tienen hijos



# Jugadores de tenis

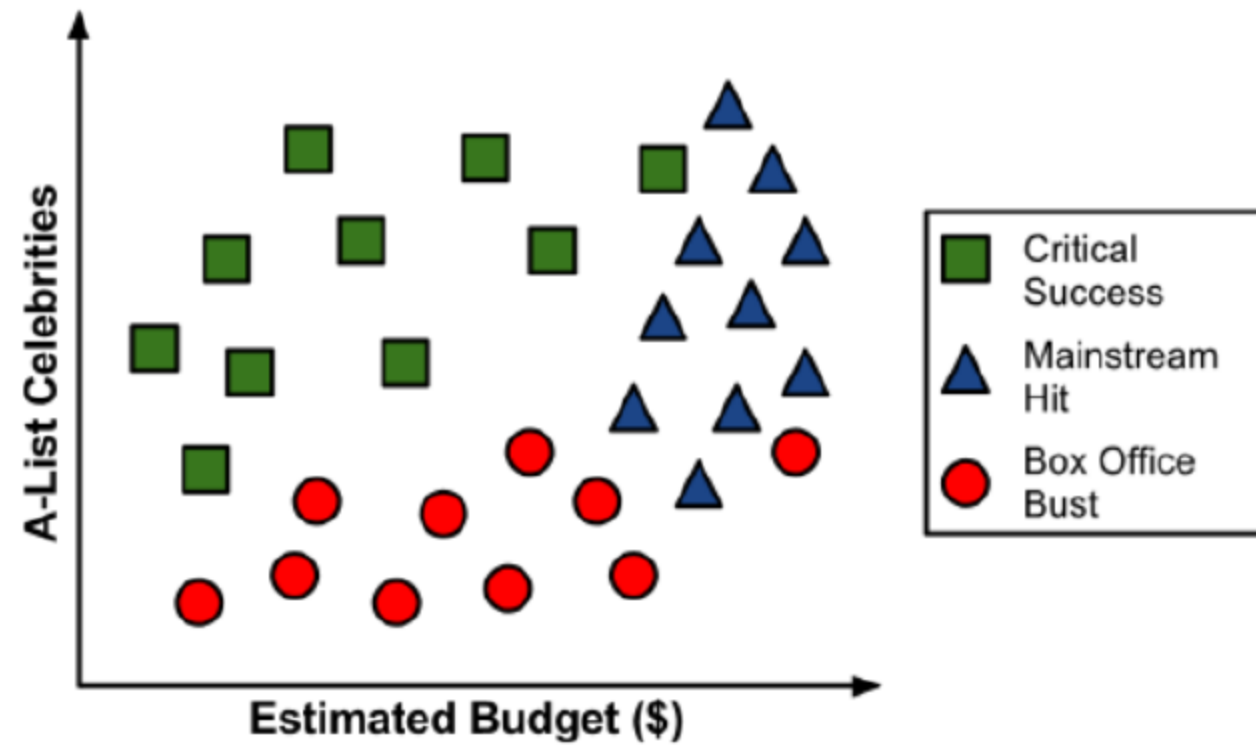
Temperature	Outlook	Humidity	Windy	Played?
Mild	Sunny	80	No	Yes
Hot	Sunny	75	Yes	<b>No</b>
Hot	Overcast	77	No	Yes
Cool	Rain	70	No	Yes
Cool	Overcast	72	Yes	Yes
Mild	Sunny	77	No	<b>No</b>

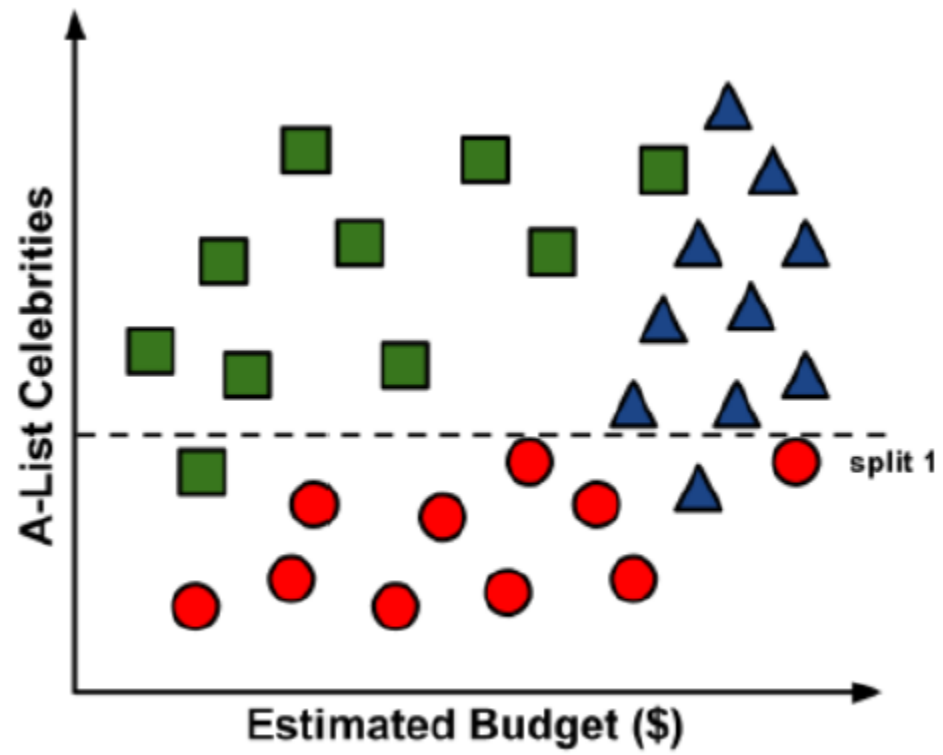


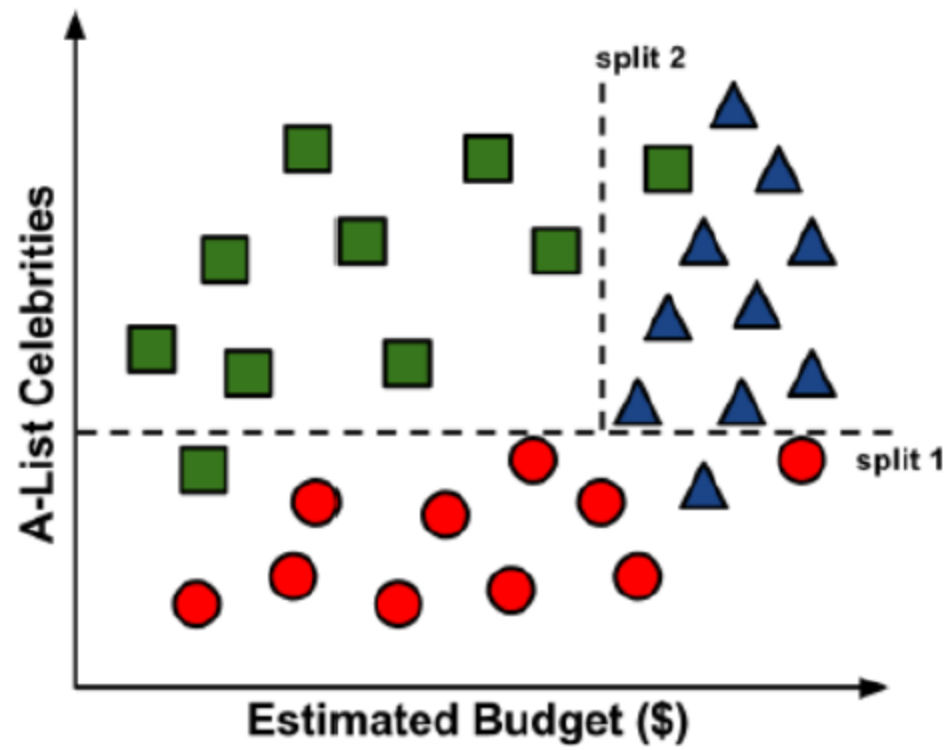
# Éxito de película ( caso 2X2)

- ▶ Features:
  - ▶ Presupuesto estimado
  - ▶ Cantidad de celebridades famosas
- ▶ Variable objetivo: Tipos de éxito de una película
  - ▶ Critical success (éxito de la crítica)
  - ▶ Mainstream hit (éxito de las masas)
  - ▶ Box office bust (fracaso)





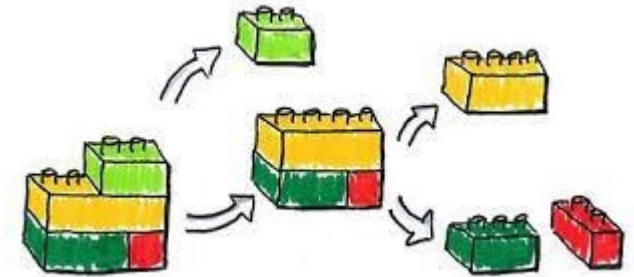




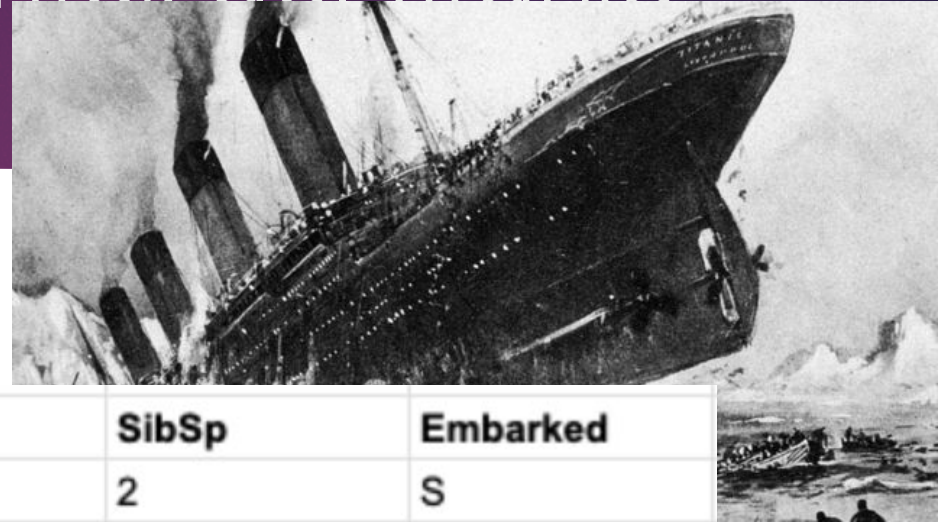
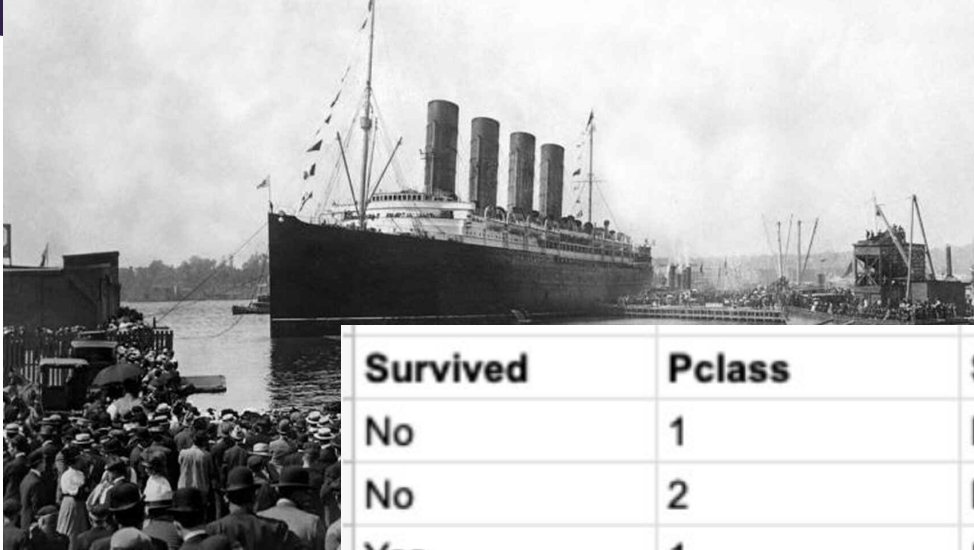
# Árbol de decisión

- ▶ Básicamente es un clasificador con estructura de árbol
- ▶ Se basan en heurística llamada particionamiento recursivo

- ▶ El algoritmo se detiene si:
  - ▶ Una de las particiones genera la mayoría de una clase
  - ▶ Cuando no hay más feature que los puedan seguir particionando
  - ▶ Cuando el árbol ha crecido a un límite predefinido



# Prediciendo sobrevivientes del Titanic



Survived	Pclass	Sex	SibSp	Embarked
No	1	M	2	S
No	2	M	1	S
Yes	1	F	1	S
Yes	3	F	0	C
Yes	1	F	0	Q
No	2	F	2	C
No	2	F	0	C
No	1	M	1	Q



# ¿Quién sobrevive y quién no?

- ▶ Pclass: Clase de boleto 1,2,3
- ▶ Sex: Sexo pasajero M,F
- ▶ SibSp: Acompañantes 0,1,2
- ▶ Embarked: puerto de embarque S,C,Q

Survived	Pclass	Sex	SibSp	Embarked
No	1	M	2	S
No	2	M	1	S
Yes	1	F	1	S
Yes	3	F	0	C
Yes	1	F	0	Q
No	2	F	2	C
No	2	F	0	C
No	1	M	1	Q

# Dividamos por sexo

Sexo

```
graph TD; A[Sexo] --> B[Femenino (+3, -2)]; A --> C[Masculino (-3)];
```

A decision tree diagram with a root node 'Sexo' in a pink box. It branches into two child nodes: 'Femenino (+3, -2)' and 'Masculino (-3)', both in orange boxes. The branches are represented by orange lines.

Femenino  
(+3, -2)

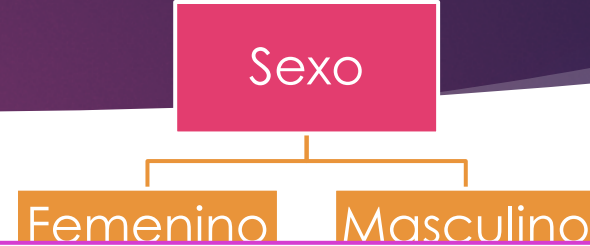
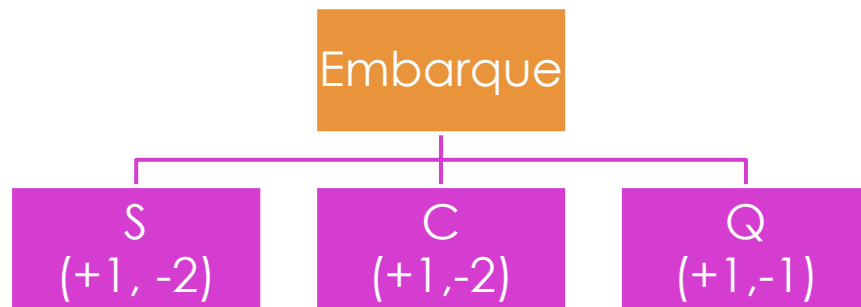
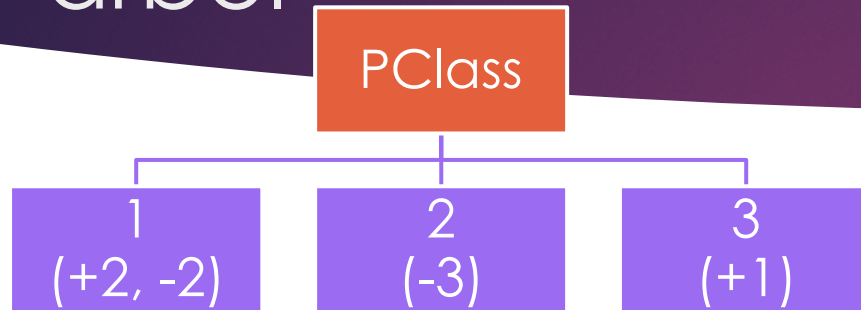
Masculino  
(-3)

¿Cómo crear un árbol  
con las otras  
variables?

¿Qué variables  
debemos elegir para  
predecir?

¿En qué orden deben  
estar?

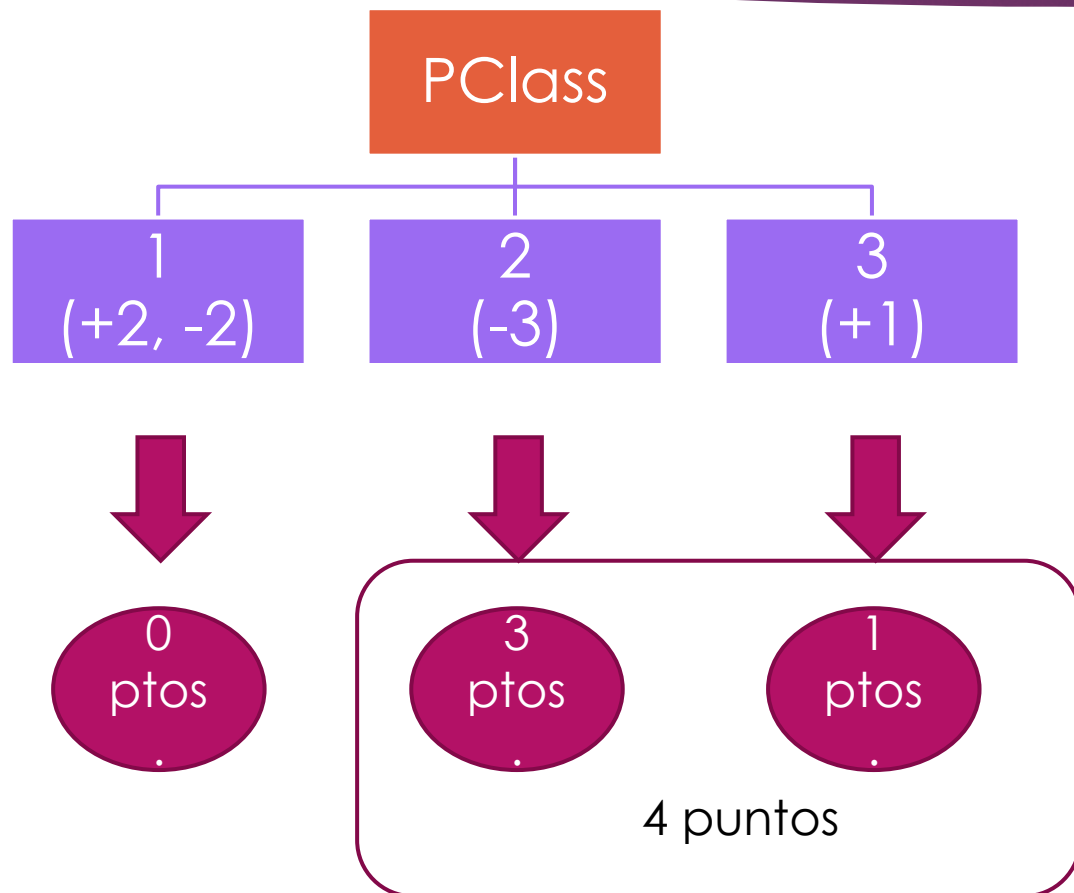
# Elegir mejor feature para dividir el árbol



¿Cuál es la mejor variable para dividir?  
Homogeneidad

# Asumamos que ...

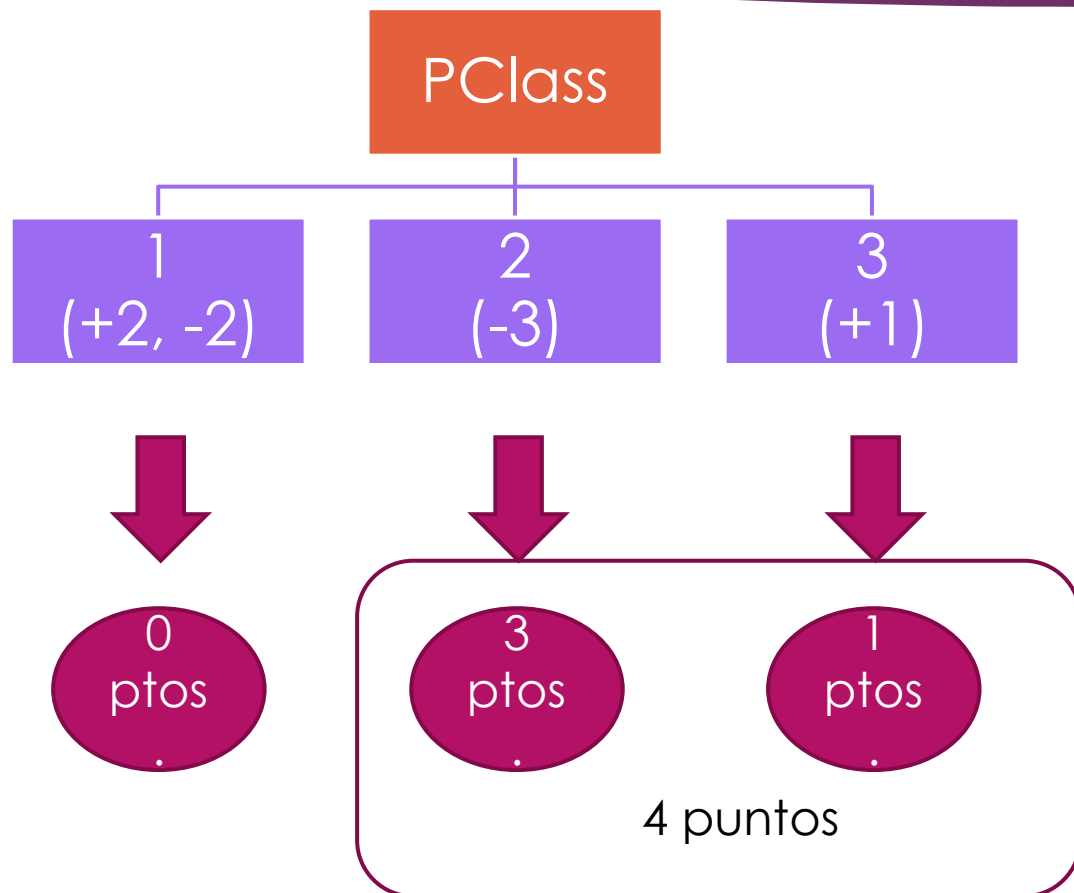
Asignamos puntajes acorde a que tan homogéneo es. Puntaje suma de clases iguales



Pclass: 4 puntos  
Sex: 3 puntos  
SibSp: 2 puntos  
Embarked: 0 puntos

# Asumamos que ...

Asignamos puntajes acorde a que tan homogéneo es. Puntaje suma de clases iguales



**Pclass: 4 puntos**

Sex: 3 puntos

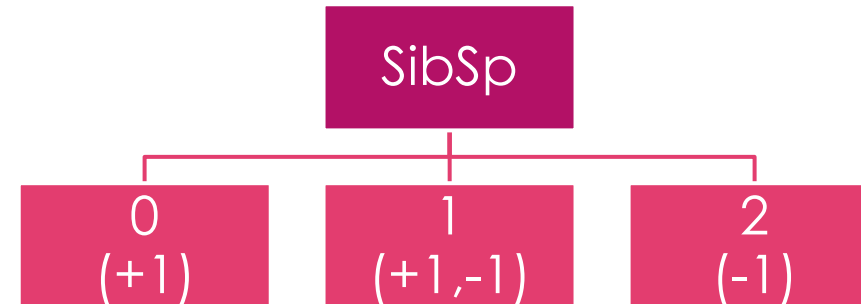
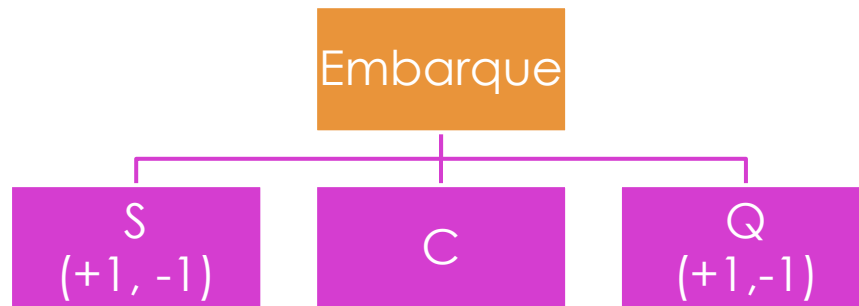
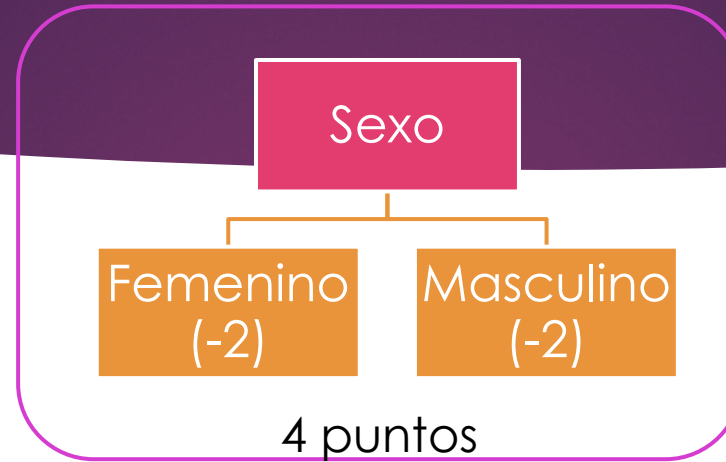
SibSp: 2 puntos

Embarked: 0 puntos

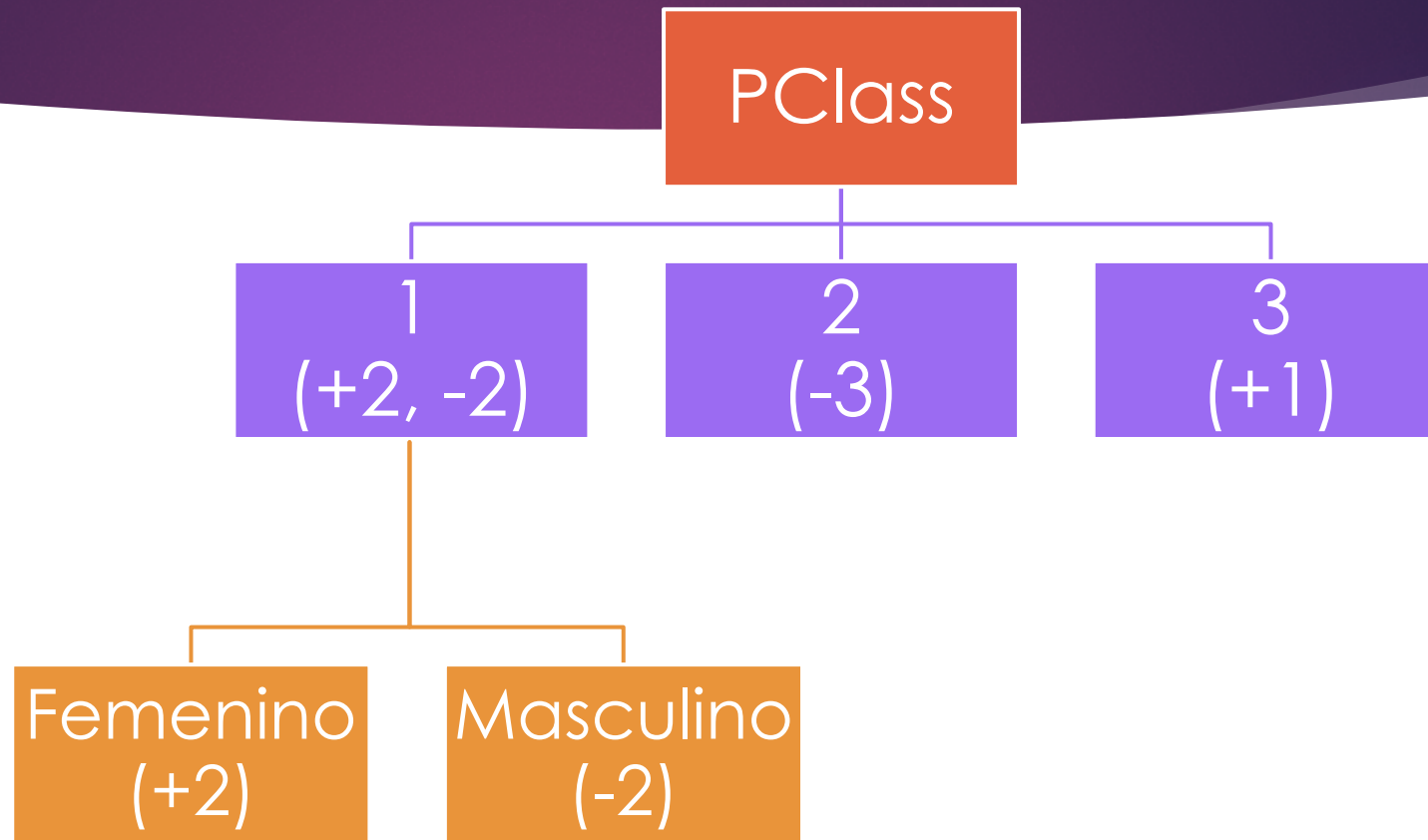
**Máxima  
Homogeneidad**



# Elegir mejor feature para dividir el árbol

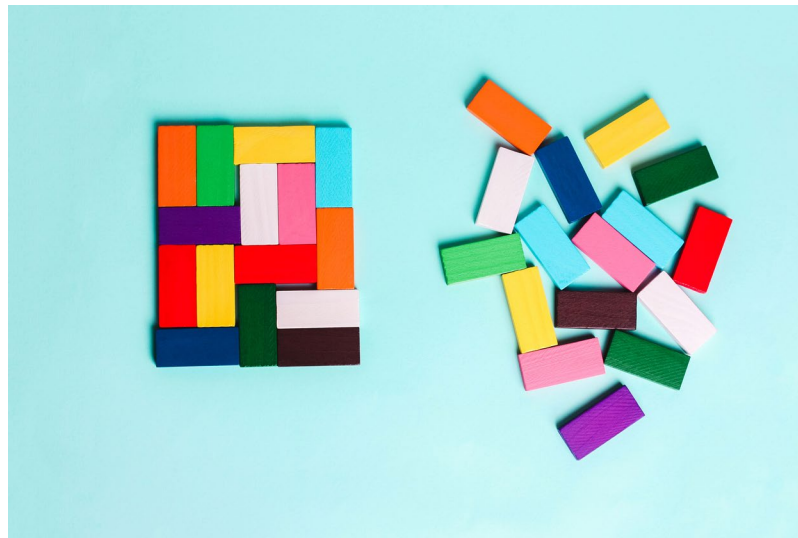


# Árbol final



# Problema

No siempre esto se puede hacer en grandes volúmenes de datos entonces necesitamos otra medida...



# Entropía

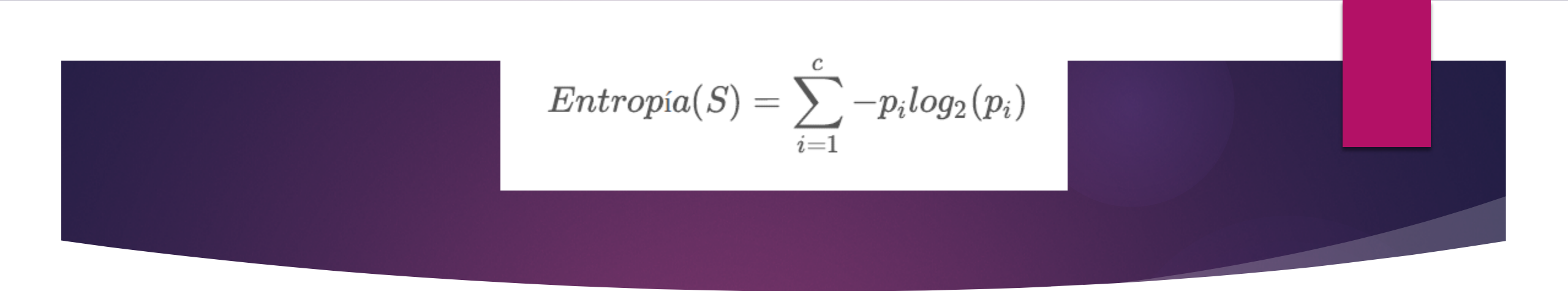
- ▶ Un árbol de decisión intenta encontrar un corte que reduzca la entropía, incrementando la homogeneidad dentro los subconjuntos.
- ▶ “Pureza” de la partición realizada por el feature.
- ▶ Se refiere al modo que tenemos para elegir aquella variable que haga que la partición siguiente sea lo más pura posible.

$$Entropía(S) = \sum_{i=1}^c -p_i \log_2(p_i)$$

# Entropía

- ▶ Evalúa el “desorden” de un conjunto de datos
- ▶ Conjuntos con alta entropía son diversos y proveen poca información, relación inversa
- ▶ La entropía también se mide en bits.
  - ▶ Para 2 clases sus valores van desde el 0 al 1.
  - ▶ Para  $n$  clases sus valores van desde 0 al  $\log_2(n)$
- ▶ El valor 0 indica que el subconjunto es completamente homogéneo y el valor 1 que es lo mas heterogéneo posible.




$$Entropía(S) = \sum_{i=1}^c -p_i \log_2(p_i)$$

- Sobrevive: 50%
- No sobrevive: 50%

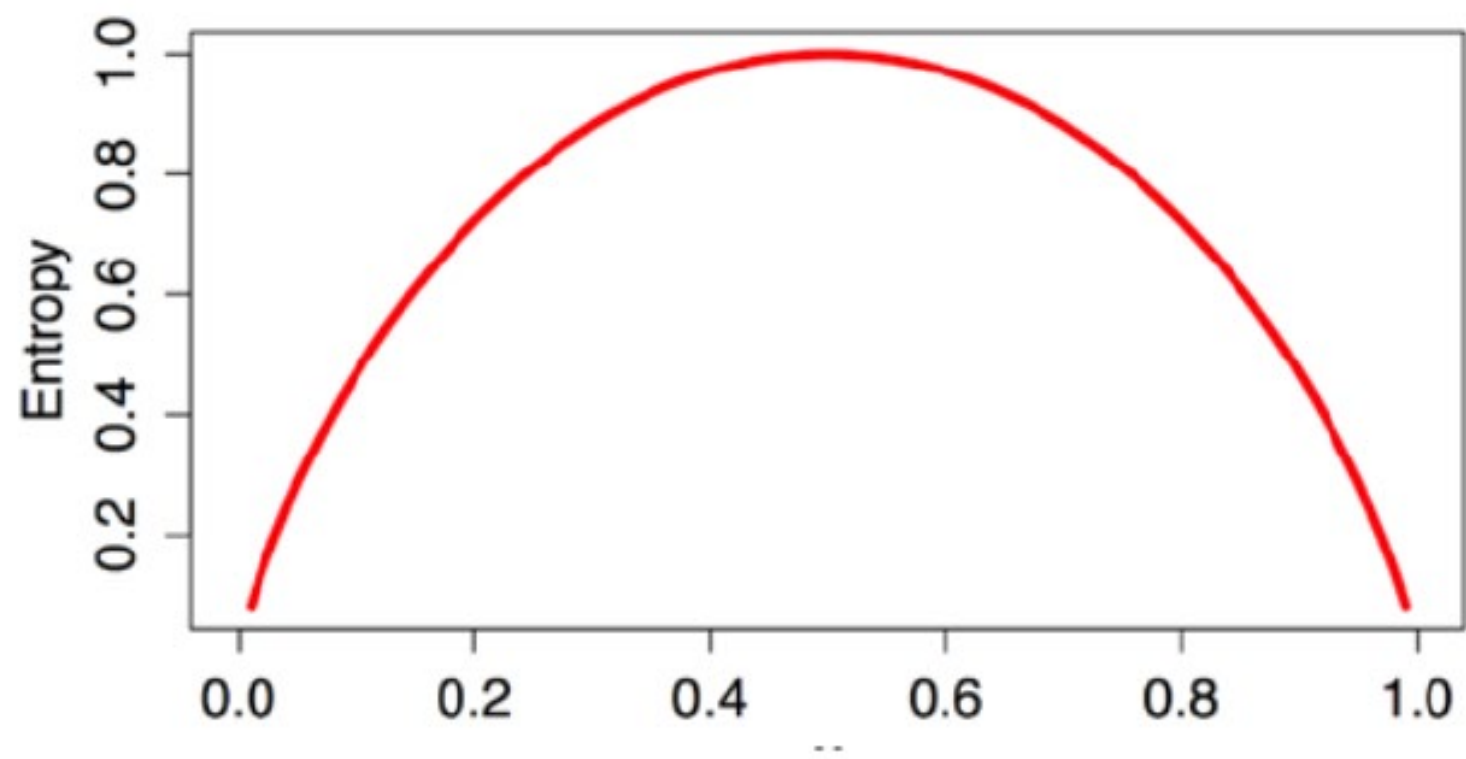
$$Entropía(S) = -0.5 * \log_2(0.5) - 0.5 * \log_2(0.5) = 1$$

- Sobrevive: 60%
- No sobrevive: 40%

$$Entropía(S) = -0.6 * \log_2(0.6) - 0.4 * \log_2(0.4) = 0.97$$

- Sobrevive: 99%
- No sobrevive: 1%

$$Entropía(S) = -0.99 * \log_2(0.99) - 0.01 * \log_2(0.01) = 0.08$$



# Otra medida de “pureza” Índice de Gini

- ▶ El índice de Gini es la función de coste más utilizada en los árboles de decisión.
- ▶ Este índice calcula la cantidad de probabilidad de que una característica específica se clasifique incorrectamente cuando se selecciona al azar
  - ▶ va de 0 (un corte puro) a 0,5 (corte completamente puro que divide los datos en partes iguales).

$$Gini = 1 - \sum_{i=1}^n (P_i)^2$$

# Ganancia de información

- ▶ Permite decidir que feature debe incluirse primero en el árbol que se está entrenando
- ▶ Es una medida de cuánto aporta agregar una feature en la predicción de las clases que deseamos obtener
- ▶ Una ganancia de información alta permite reducir la incertidumbre respecto a la clase que pertenece
- ▶ La ganancia de información debe tener un valor:
  - ▶ Máximo si el atributo es perfecto (discriminación perfecta)
  - ▶ Mínimo cuando el atributo es irrelevante

# Ganancia de información

- ▶ La ganancia de información por una feature se calcula como la diferencia de entropía entre un conjunto antes del corte ( $S_1$ ) y los subconjuntos después del corte ( $S_2$ ).

$$Ganancia(F) = Entropía(S_1) - Entropía(S_2)$$

- ▶ Cuando dividimos un conjunto, se generan mas de 1 subconjuntos, por lo que el cálculo de la entropía debe considerar varios subconjuntos. El cálculo se realiza como la suma ponderada de las entropías de los subconjuntos.

$$Entropía(S) = \sum_{i=1}^n w_i * Entropía(S_i)$$



# Ganancia de información

- ▶ Si la ganancia de información es alta, entonces el corte en esa feature es mejor creando grupos homogéneos.
- ▶ Si la ganancia de información es baja, entonces el corte en esa feature es peor creando grupos homogéneos.
- ▶ Los ejemplos anteriores asumen cortes en features categóricas, pero también es posible realizar cortes en features numéricas. Se debe escoger el corte numérico que entrega la mayor ganancia de información para esa feature.

# Pasos para construir el árbol de decisión

- ▶ Calcular la ganancia de información para todas las variables.
- ▶ Elegir como split aquel split que genere mayor *ganancia de información*.
- ▶ Repetir el proceso hasta que la ganancia de información no sea significativo o no supere un *threshold*.

# Como hacer el split

Para variables categóricas se debe calcular la ganancia de información para todas las combinaciones posibles de esa variable, excluyendo la opción que incluye a todas las opciones (ya que no estaría haciendo ningún split).

# Algoritmos

- ▶ ID3
- ▶ C4.5
- ▶ C5.0
- ▶ CART

# ID3

- ▶ ID3 (dicotomizador iterativo 3) fue desarrollado en 1986 por Ross Quinlan.
- ▶ El algoritmo crea un árbol de múltiples vías, encontrando para cada nodo (es decir, de manera codiciosa) el feature categórico que produce la mayor ganancia de información para la clasificación.
- ▶ Los árboles crecen hasta el máximo
  - ▶ Se aplica proceso de poda para mejorar la capacidad del árbol de generalizar a datos invisibles.
- ▶ Construye árbol de arriba abajo, de forma directa, sin hacer backtracking
- ▶ Usa entropía

# Algoritmo C4.5

- ▶ Sucesor de ID3
- ▶ Eliminó la restricción de que las características deben ser categóricas al definir dinámicamente un atributo discreto (numérico) que divide el valor del atributo continuo en un conjunto discreto de intervalos.
- ▶ C4.5 convierte los árboles entrenados (es decir, la salida del algoritmo ID3) en conjuntos de reglas si-entonces.
- ▶ Luego, se evalúa la precisión de cada regla para determinar el orden en el que deben aplicarse.
- ▶ La poda se realiza eliminando la condición previa de una regla si la precisión de la regla mejora sin ella.



# Algoritmo C5.0

- ▶ C5.0 es la última versión de Quinlan.
- ▶ Utiliza menos memoria y crea conjuntos de reglas más pequeños que C4.5 a la vez que es más preciso.
- ▶ Busco la feature con mayor ganancia de información.
  - ▶ Particiono por esa feature e itero hasta:
    - ▶ Todos o casi todos los casos de un nodo tienen una misma clase.
    - ▶ No quedan features para distinguir entre los casos.
    - ▶ Se ha llegado a un tamaño límite del árbol.
- ▶ Pruning (poda) del árbol: reducir el tamaño del un árbol.
  - ▶ pre-pruning: parar cuando se alcanza una cierta cantidad de nodos.
  - ▶ post-pruning: crecer un árbol intencionalmente y luego eliminar nodos hasta un nivel mas apropiado.

# Algoritmo CART

- ▶ Por sus siglas en Inglés *Classification and Regression Trees* o árboles de clasificación y regresión
  - ▶ Más usado actualmente
  - ▶ CART (árboles de clasificación y regresión) es muy similar a C4.5, pero se diferencia en que admite variables objetivo numéricas (regresión) y no calcula conjuntos de reglas. CART construye árboles binarios utilizando la feature y el umbral que producen la mayor ganancia de información en cada nodo.
- ▶ Implementado en Python en scikit-learn usa una versión optimizada del algoritmo CART

# Criterio de costo de complejidad - Cost complexity criterion

- ▶ Para encontrar el balance entre la profundidad y complejidad del árbol con respecto a la capacidad predictiva del modelo en datos de test
  - ▶ El árbol debe crecer hasta la mayor extensión para luego podarlo e identificar un árbol más pequeño óptimo

# Criterio de costo de complejidad

- ▶ Se usa el parámetro de costo de complejidad ( $\alpha$ ) que penaliza la función objetivo abajo para el número de nodos hoja en el árbol ( $T$ )

$$\textit{minimize} (SSE + \alpha|T|)$$

- ▶ Valores de  $\alpha$  se encuentra el árbol podado más pequeño (número de nodos hoja) que tiene el error más bajo de penalización.
- ▶ Se evalúan múltiples modelos a través de un espectro de  $\alpha$  identificando el óptimo (se puede usar CV).



¡Manos a la obra !