



# INTRODUCCIÓN AL APRENDIZAJE AUTOMÁTICO

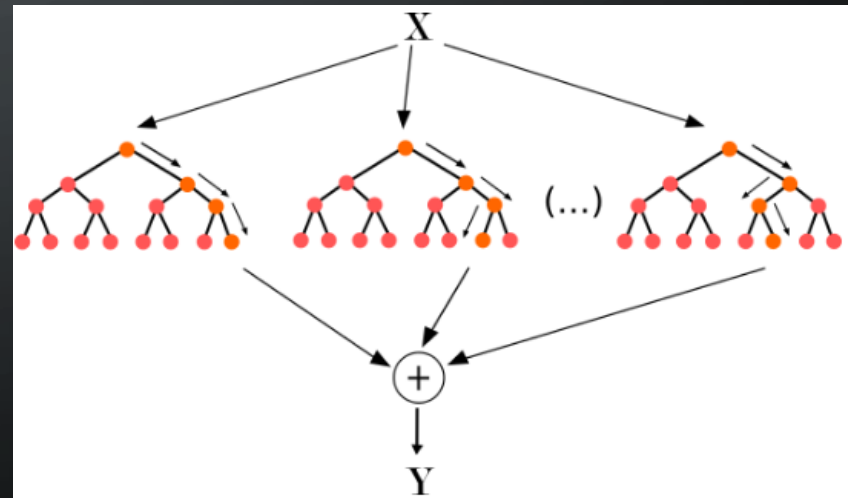
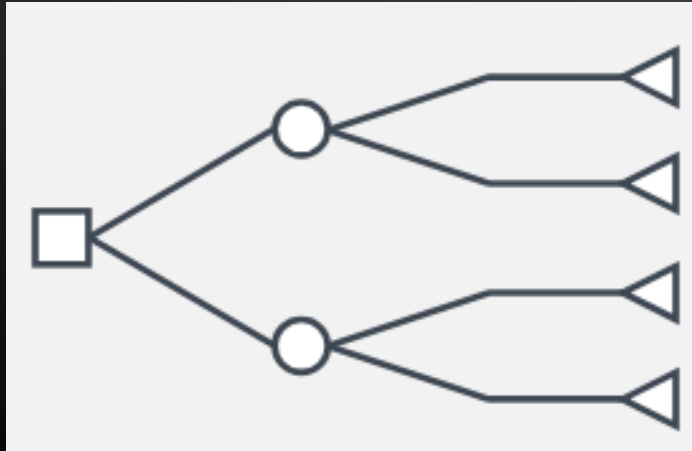
ROMÁN DE LAS HERAS

CIENTÍFICO DE DATOS

SAP / AGILE SOLUTIONS


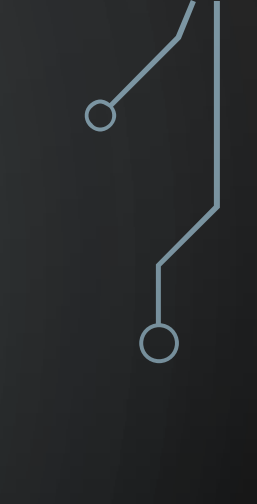
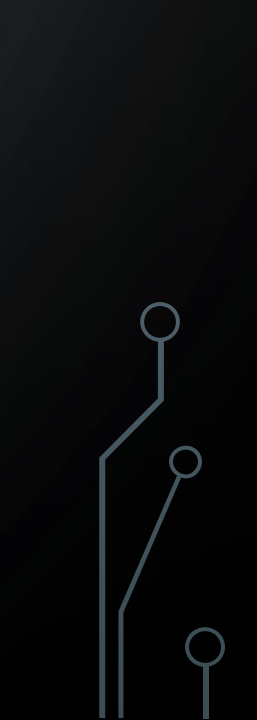
Input  $\longrightarrow$  Output

## DESDE REGLAS SIMPLES HASTA MODELOS BASADOS EN ÁRBOLES DE DECISIÓN





# CONTENIDO

1. Conceptos Básicos
  2. Reglas Simples
  3. Árboles de Decisión
  4. Bosques Aleatorios
- 
- 
- 

# 1. CONCEPTOS BÁSICOS

# INTELIGENCIA ARTIFICIAL

- John McCarthy (1956): la ciencia e ingeniería de hacer máquinas inteligentes, en especial programas de cómputo inteligentes.

# INTELIGENCIA ARTIFICIAL



# INTELIGENCIA ARTIFICIAL

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
from sklearn.ensemble import GradientBoostingRegressor

def main():
    print("READING TRAINING SET...")
    ds = pd.read_csv('train3.csv')

    print("SPLITTING INTO TRAIN & TEST SET...")
    x_train, x_test, y_train, y_test = train_test_split(ds.ix[:, 'cont1': 'cat116HR'], np.log(ds.loss), test_size = 0.4, random_state = 42)

    print("BUILDING MODEL...")
    rf = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, min_samples_split=6, min_samples_leaf=3, max_features=0.1, verbose=1)
    rf.fit(x_train, y_train)

    print("EVALUATING PERFORMANCE...")
    pred = np.exp(rf.predict(x_test))
    y_test = np.exp(y_test)
    mae = mean_absolute_error(y_test, pred)
    print("MAE: " + str(mae))

    print("READING AND PREDICTING TEST SET...")
    test = pd.read_csv('test3.csv')
    ids = test.id
    test = test.ix[:, 'cont1': 'cat116HR']
    pred = np.exp(rf.predict(test))

    print("WRITING SUBMISSION FILE...")
    pd.DataFrame({'id': ids, 'loss': pred}, columns = ['id', 'loss']).to_csv('predictions.csv', index = False)

if __name__ == "__main__":
    main()
```

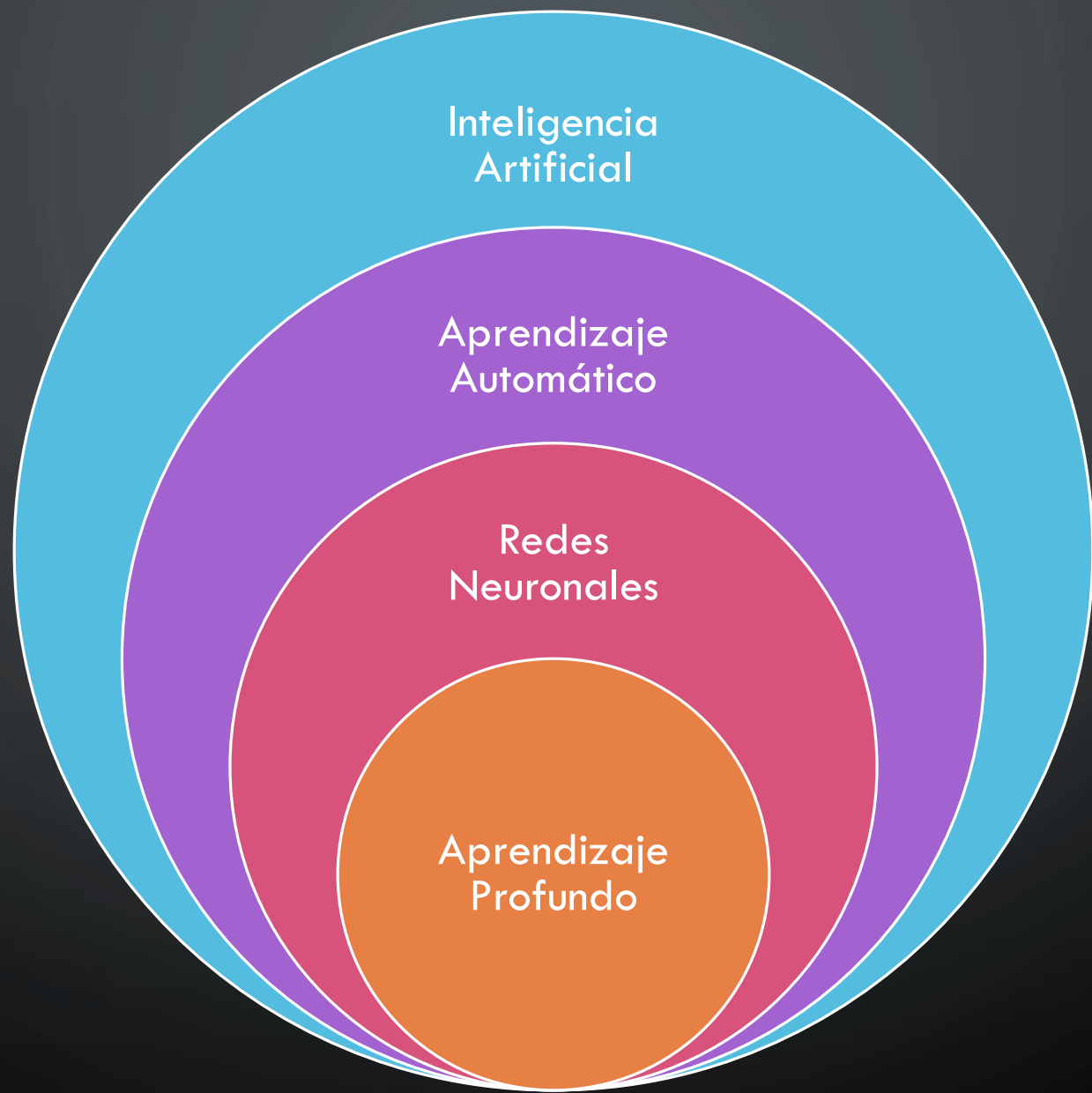


# INTELIGENCIA ARTIFICIAL

- Primera “Inteligencia Artificial”: Máquina Sumadora (Blaise Pascal – 1642)







# MACHINE LEARNING

Arthur Samuel (1959): es el campo de estudio que le da a las computadoras la habilidad de aprender sin ser programadas explícitamente.



# MACHINE LEARNING



# MACHINE LEARNING

- Tom Mitchell (1998): Un programa se dice que aprende de la experiencia E con respecto a una tarea T y alguna medida de rendimiento P, si su rendimiento en T, medido por P, mejora con la experiencia E.
  - Experiencia (E): conjunto de datos
  - Tarea (T): objetivo del algoritmo / función
  - Rendimiento (P): métrica de evaluación / éxito

# MACHINE LEARNING

EXPERIENCIA (DATOS)	TAREA (OBJETIVO)	RENDIMIENTO (MÉTRICA DE ÉXITO)
Fotografías de rostros, con sus caras marcadas (Facebook)	Reconocimiento facial	Porcentaje de caras reconocidas correctamente
Gustos de canciones (Spotify)	Recomendación de canciones	Porcentaje de canciones recomendadas que gustaron
Datos de conducción y acciones tomadas (Google)	Conducción autónoma de vehículos	Distancia recorrida sin cometer errores

# MACHINE LEARNING

**Objetivo:** Aproximar una función

$$f: X \rightarrow Y$$

- $X = \{x_1, x_2, \dots, x_n\}$  (entrada, propiedades, ...)
- $Y = \{y_1, y_2, \dots, y_m\}$  (salida, objetivo, ...)

# MACHINE LEARNING

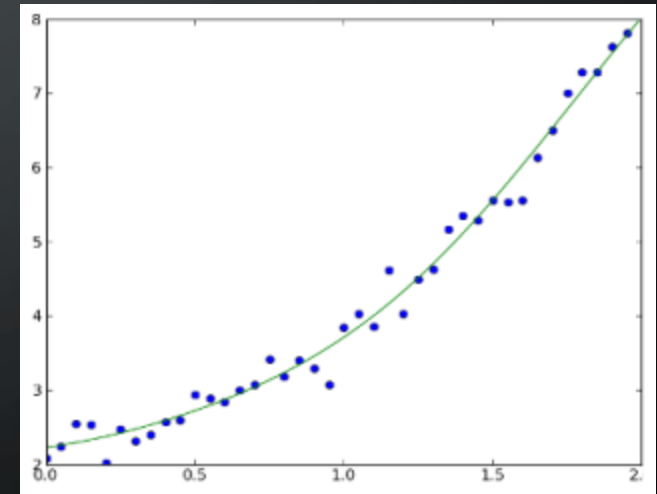
## Tipo de Variable:

- Categórica → Clasificación

Ej:  $y \in \{"spam", "no - spam"\}$

- Numérica → Regresión

Ej:  $y \in [0, 1]$





# MACHINE LEARNING

## Tipos de Aprendizaje Automático:

- **Supervisado:** Y es conocida (Regresión / Clasificación)
- **No-Supervisado:** Y es desconocida
- **Refuerzo:** Aprendizaje gradual basado en recompensas y penalizaciones por acciones sobre el ambiente

# MACHINE LEARNING

- **ENTRENAMIENTO**

para cada instancia  $(x_i, y_i)$  en el conjunto de datos:

`modelo.aprender(xi, yi)`

# modelo para calcular y a partir de x

# el modelo es “aprendido” por un algoritmo

- **PREDICCIÓN**

`y_pred = modelo.calcular(x_nueva)`

# MACHINE LEARNING

- **MÉTRICAS DE EVALUACIÓN:**

- Exactitud (Accuracy) → Clasificación:

$$exactitud = \frac{\text{No. de Aciertos}}{\text{Total de Casos}}$$

- Raíz del Error Cuadrático Medio (Root Mean Squared Error) → Regresión:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2}$$

# MACHINE LEARNING

## ENTRENAMIENTO VS VALIDACIÓN


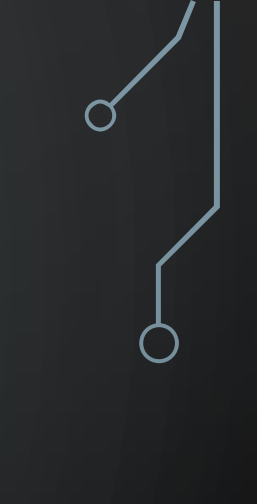
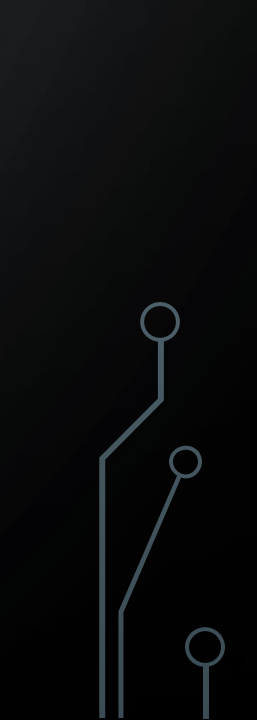
- Usualmente se separa el conjunto de datos en dos partes:
  - Entrenamiento: para construir (entrenar) el modelo.
  - Validación: para evaluar el rendimiento del modelo sobre datos nuevos.
- Deben ser muestras aleatorias.
- El conjunto de validación se reserva aparte.

ENTRENAMIENTO  
(100- $q$ )%

VALIDACIÓN  
 $q$ %



# MACHINE LEARNING

- Conductores Claves de su Progreso:
    - Cantidad de datos (Big Data)
    - Innovación en Algoritmos (Deep Learning)
    - Capacidad Computacional (GPU)
- 
- 
- 

## 2. REGLAS SIMPLES

# REGLAS SIMPLES

si (condicion1) entonces:

# código1

si (condicion2) entonces:

# código2

...

sino:

# códigoN

- Programación tradicional
- Código realizado explícitamente para resolver un problema.
- Las reglas simples son claramente definidas por humanos (“expertos”).
- Deterministas por naturaleza.



# REGLAS SIMPLES

- CLASIFICACIÓN:

SI (CIELO = Nublado Y TIEMPO = DIA) ENTONCES

TOMAR\_TREN = SI

SI (CIELO = Lluvia) ENTONCES

TOMAR\_TREN = SI

SI (CIELO = Sol Y TEMPERATURA < 20) ENTONCES

TOMAR\_TREN = SI

SINO

TOMAR\_TREN = NO

- REGRESIÓN:

SI (SALARIO <= 152000) ENTONCES

ISR = 0

SINO SI (SALARIO <= 232100) ENTONCES

ISR = SALARIO \* 0.15

SINO SI (SALARIO <= 539790) ENTONCES

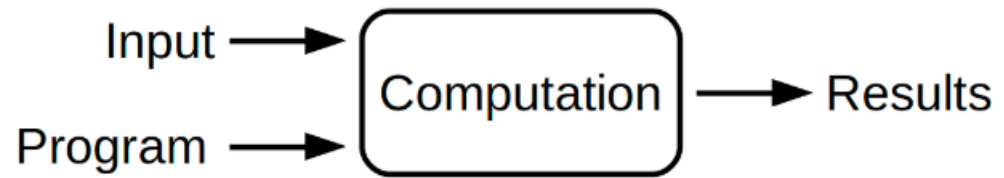
ISR = SALARIO \* 0.2

SINO

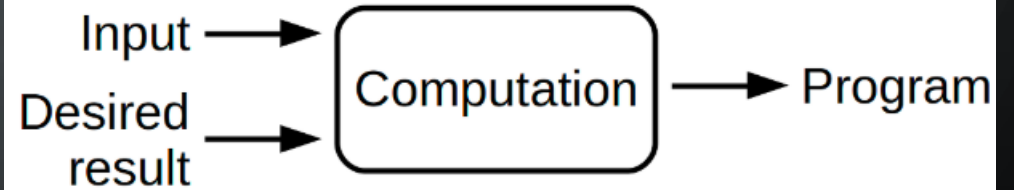
ISR = SALARIO \* 0.25

# REGLAS SIMPLES

## Traditional programming



## Machine learning

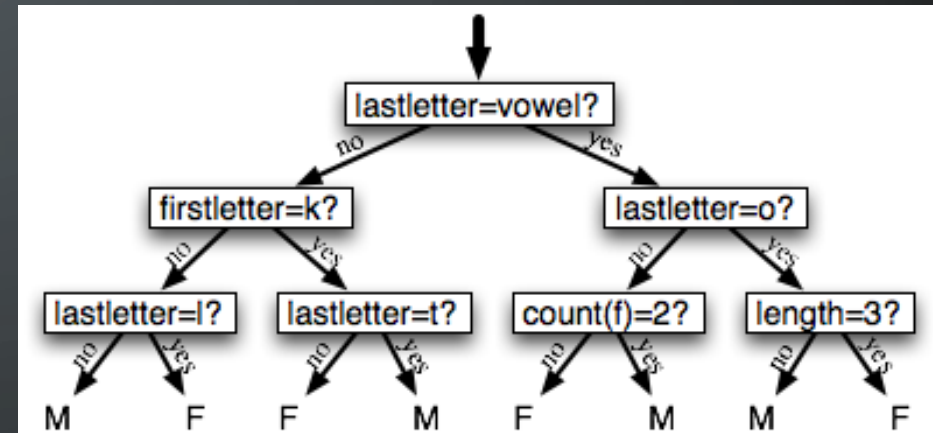


The background is a dark gray gradient. In the corners, there are decorative white line art elements that resemble circuit traces or stylized tree branches. These elements consist of thin lines connecting small circles, creating a network-like pattern. The top-left and bottom-left corners have more complex, branching structures, while the top-right and bottom-right corners have simpler, more linear structures.

# 3. ÁRBOLES DE DECISIÓN (DECISION TREE)

# ÁRBOLES DE DECISIÓN

- Un árbol de decisión consiste en un flujo de:
  - **Nodos de decisión**: revisan condiciones en los valores de entrada
  - **Nodo raíz**: primer nodo de decisión
  - **Nodos hoja**: asignan las etiquetas
- Sencillo de utilizar e interpretar.
- Complicado de crear...



Ejemplo: Árbol de decisión para determinar el género de un nombre

# ÁRBOLES DE DECISIÓN

- **ENTROPÍA:** medida de la incertidumbre en un conjunto de datos.

$$H(S) = - \sum_{i \in \text{etiquetas}} P(i) \cdot \log_2(P(i))$$

- **GANANCIA DE INFORMACIÓN:** mide que tan organizados se vuelven los datos al dividirlos usando un atributo.

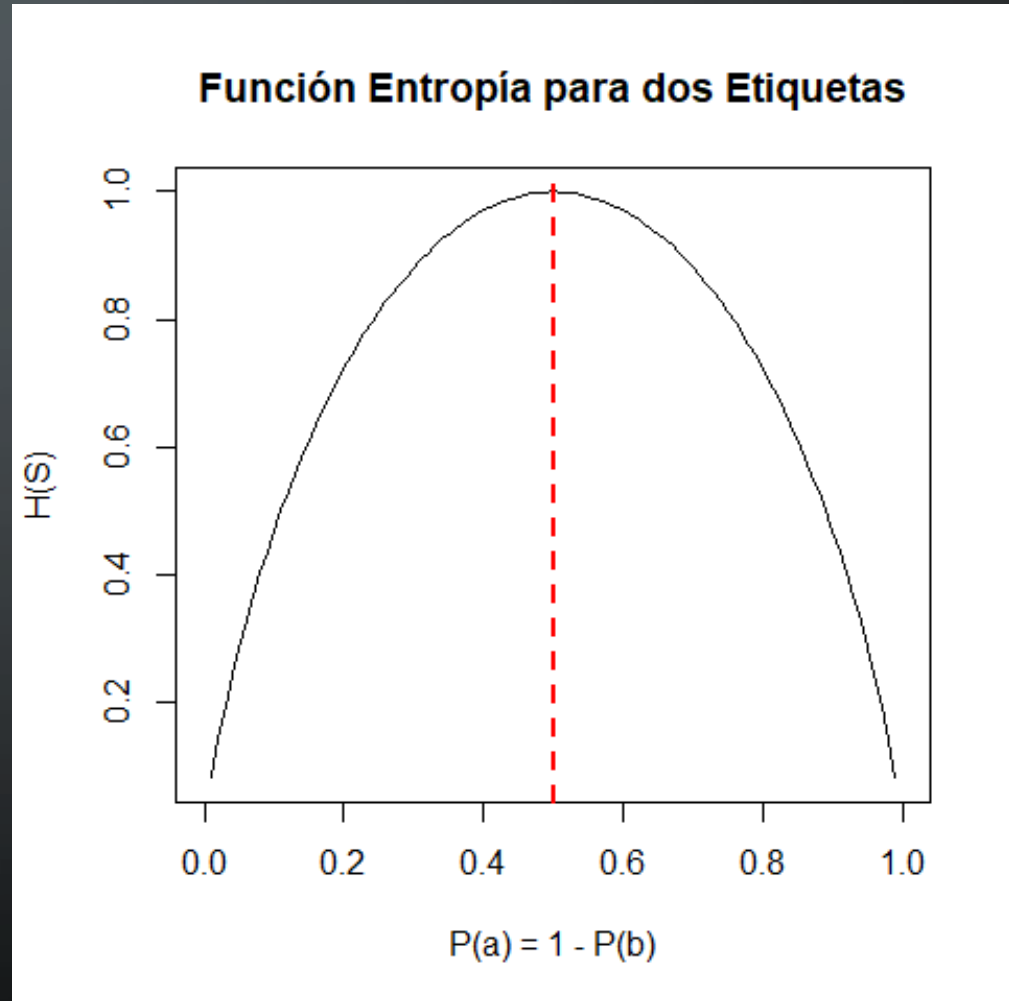
$$IG(S, a) = H(S) - H(S|a)$$

$$* IG(\text{Conjunto}, \text{atributo}) = \text{Entropia Padre} - \text{promedio}(\text{Entropia Hijos})$$

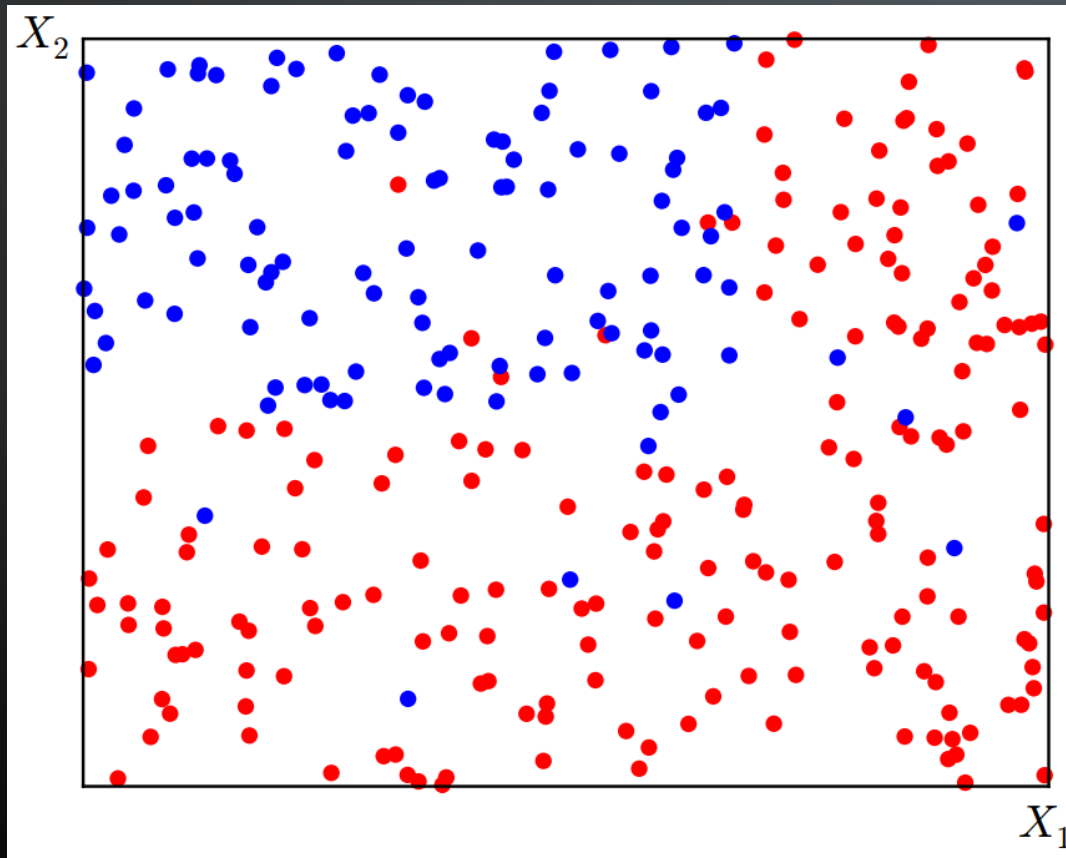
- \* **Objetivo:** Maximizar la Ganancia de Información

# ÁRBOLES DE DECISIÓN

- Si todos los valores son iguales, la entropía es 0:
  - $S = \{1, 1, 1, 1\}$
  - $S = \{0, 0, 0, 0\}$
- Si los valores están distribuidos uniformemente, la entropía es 1:
  - $S = \{1, 1, 0, 0\}$



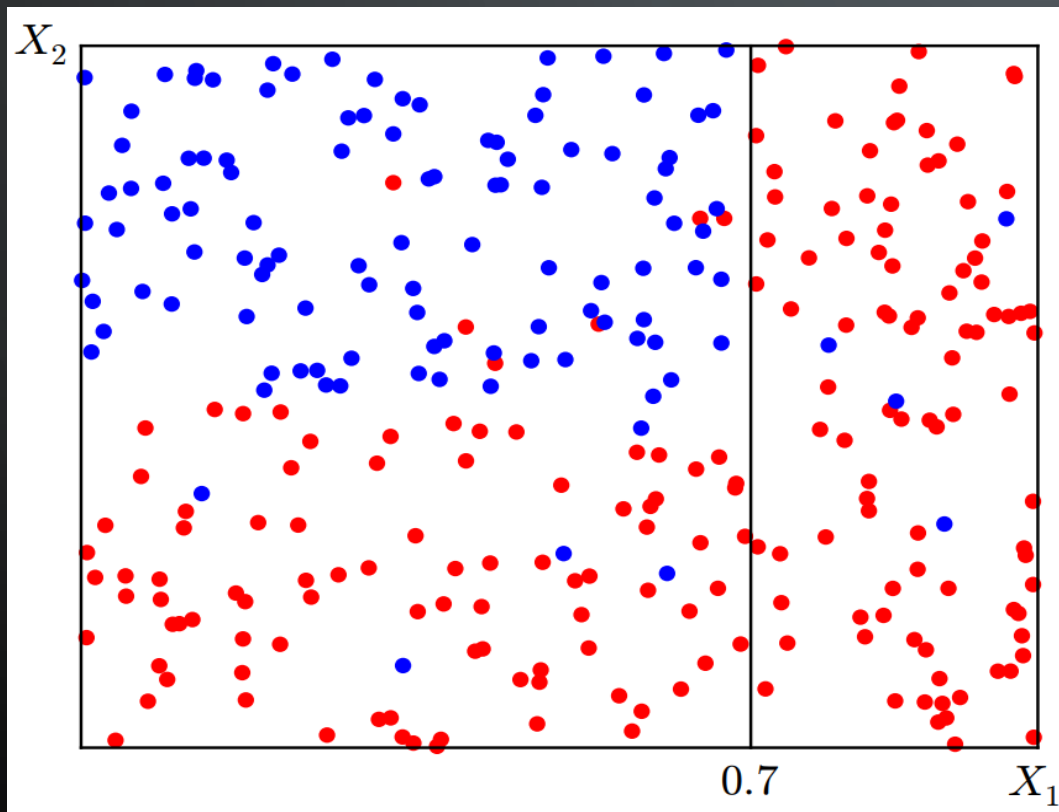
# ÁRBOLES DE DECISIÓN



$$H(S) = 0.92$$

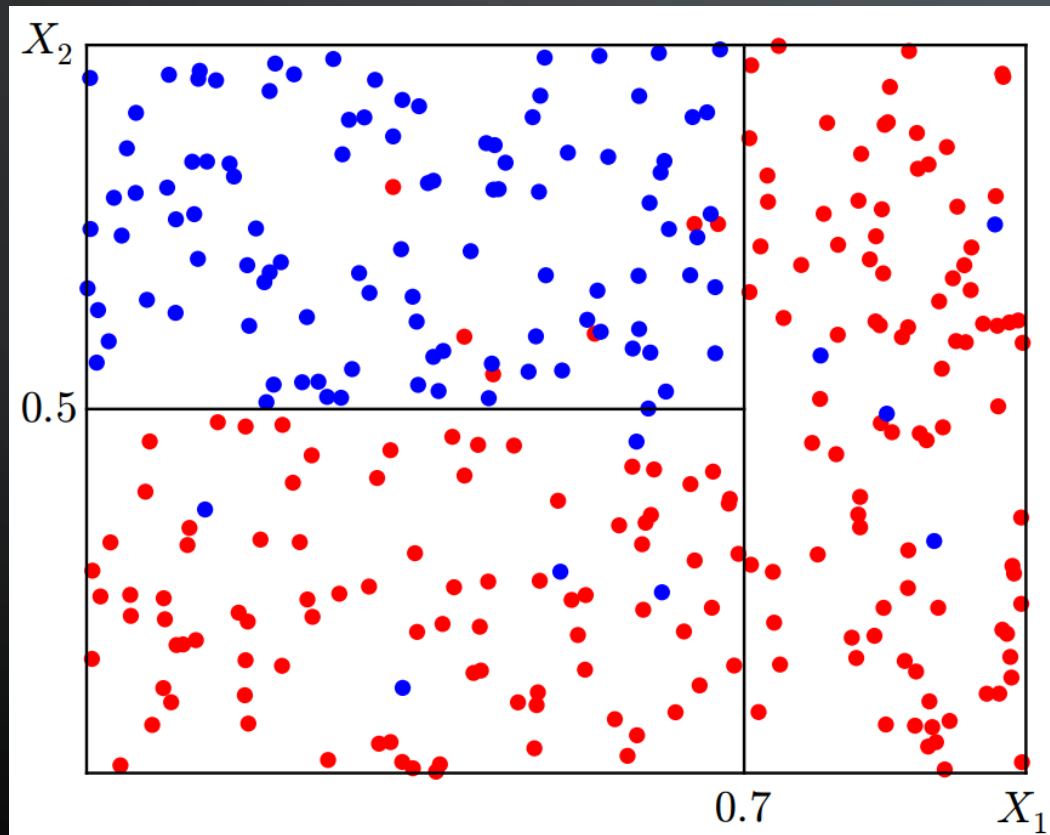


# ÁRBOLES DE DECISIÓN



- $IG(S, X_1) = H(S) - H(S|X_1)$
- $IG(S, X_1) = 0.92 - 0.62$
- $IG(S, X_1) = 0.3$

# ÁRBOLES DE DECISIÓN

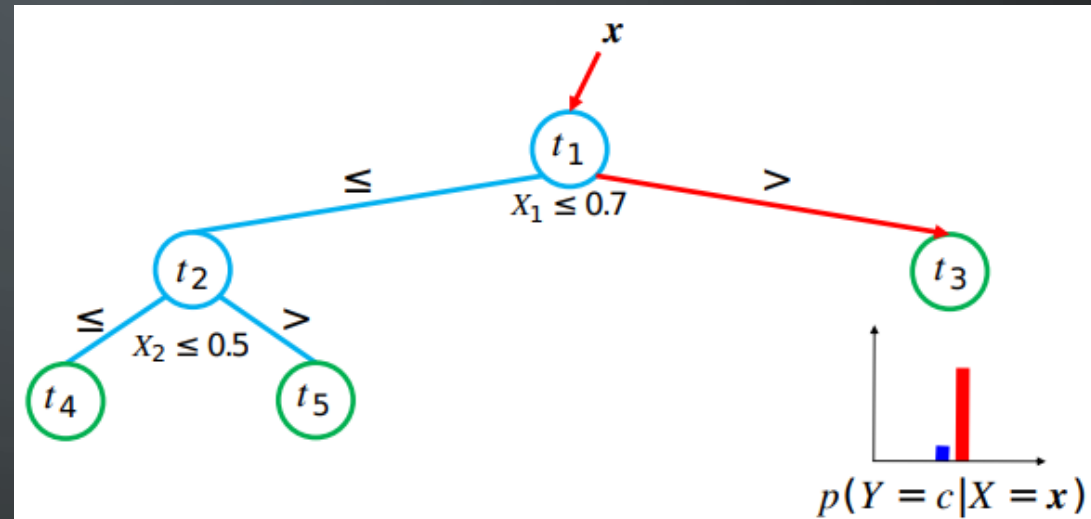
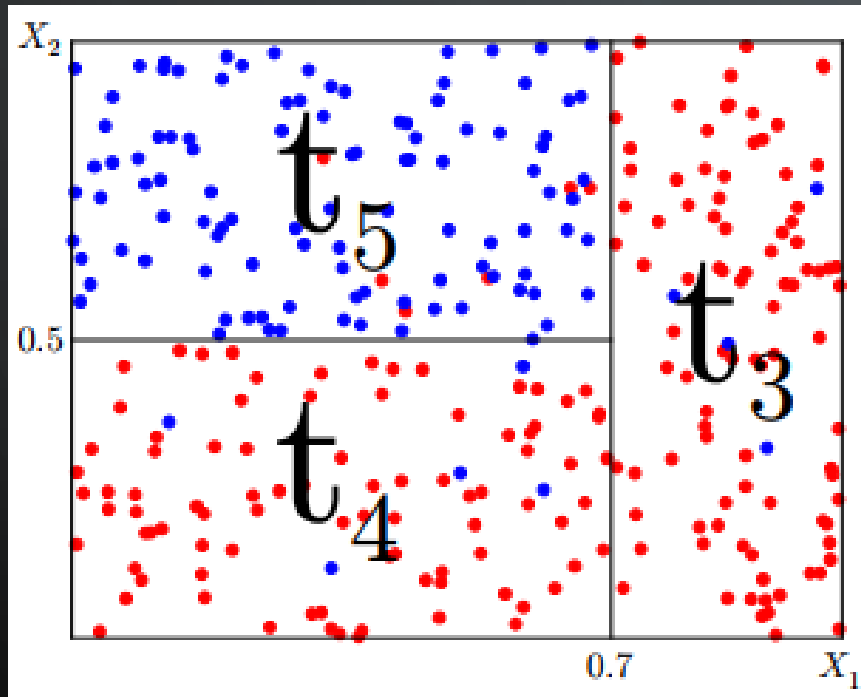


- $IG(S, X_2) = H(S) - H(S|X_2)$
- $IG(S, X_2) = 0.995 - 0.317$
- $IG(S, X_2) = 0.678$

TOTAL:

- $IG(S, X_1) + IG(S, X_2) = 0.978$

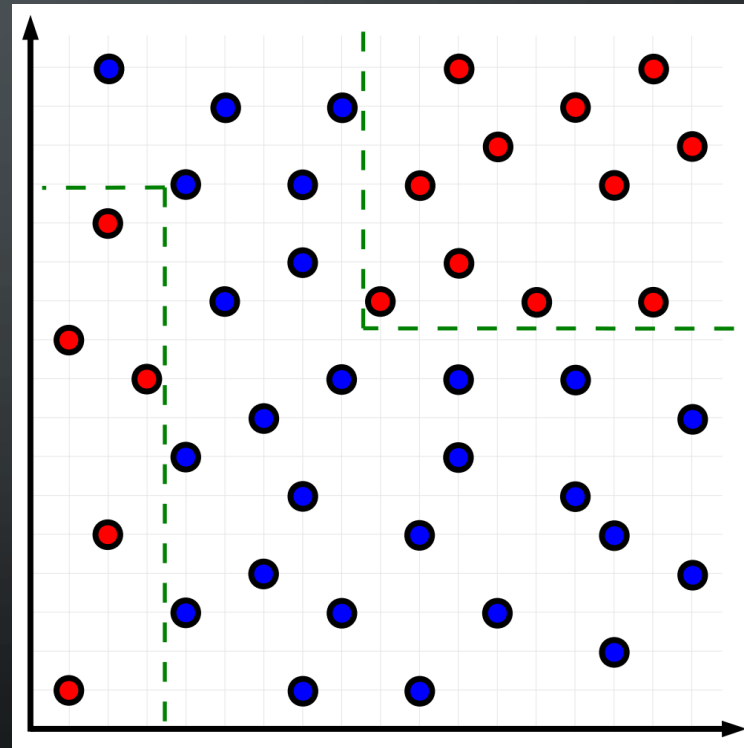
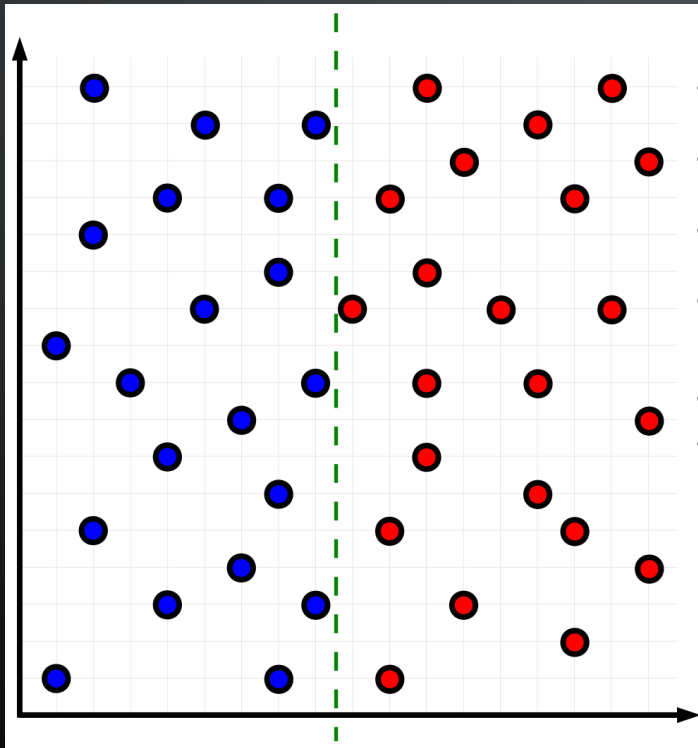
# ÁRBOLES DE DECISIÓN



Un árbol de decisión no solo predice la salida  $y$ , sino que también provee la probabilidad de dicha predicción.

# ÁRBOLES DE DECISIÓN

Cada nodo hoja del árbol representa una partición de la región.



# ÁRBOLES DE DECISIÓN

Parámetros del algoritmo:

- Máxima profundidad (Entero)
- Muestras mínimas para partición (Entero o Proporción)
- Muestras mínimas para hoja (Entero Impar o Proporción)
- Atributos máximos (Entero o Proporción)
- Máximos nodos hojas (Entero)

# ÁRBOLES DE DECISIÓN

## PROS

- Altamente interpretables
- Eficientes en espacio y procesamiento
- Usualmente tienen poco sesgo
- Pueden trabajar con datos sucios o incompletos

## CONTRAS

- Son propensos a sobre-ajustarse
- A menudo las predicciones tienen alta varianza
- Algoritmo Voraz (Puede no llegar al árbol óptimo)

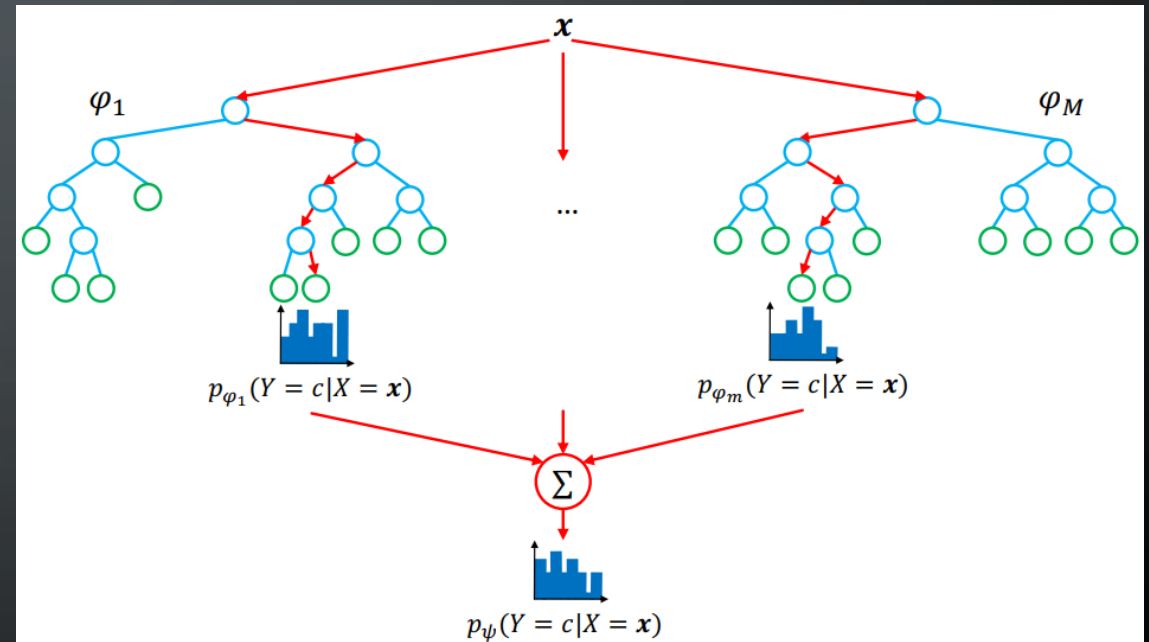
The background is a dark gray gradient. In the corners, there are decorative white line art elements resembling circuit boards or neural network connections. These elements consist of thin lines that branch out and terminate in small circles, creating a symmetrical, abstract pattern in each corner.

# 4. BOSQUES ALEATORIOS (RANDOM FOREST)



# BOSQUES ALEATORIOS

- Es una colección de árboles de decisión.
- La salida es la combinación de la votación individual de cada árbol:
  - Clasificación: Moda
  - Regresión: Media o Mediana
- En cada árbol se toma una muestra aleatoria de instancias y atributos.



# BOSQUES ALEATORIOS

- **Teorema de Juicio de Condorcet:** Considere un grupo de  $M$  votantes. Si cada votante tiene una probabilidad independiente  $p > \frac{1}{2}$  de votar por la decisión correcta, entonces agregar más votantes incrementa la probabilidad de que la decisión mayoritaria sea correcta.

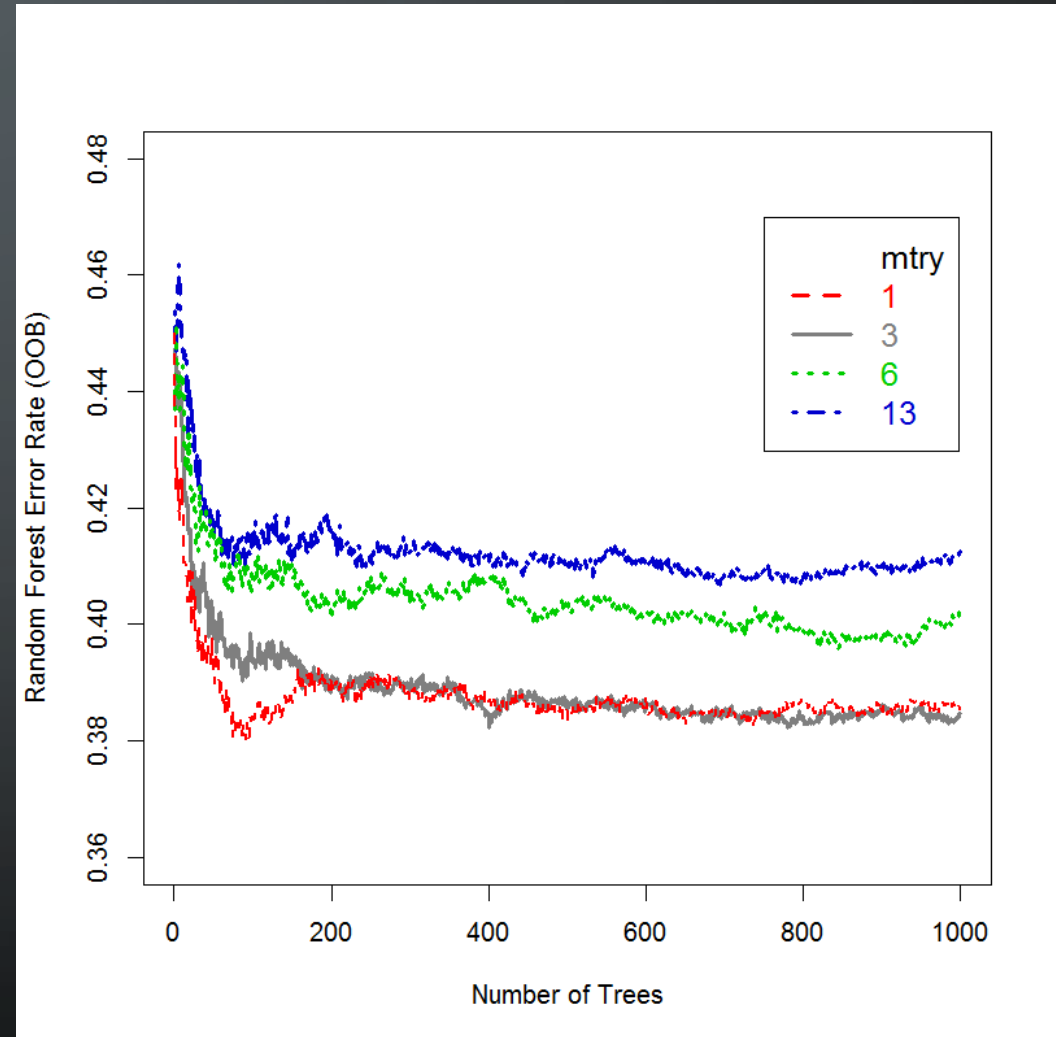
Conforme  $M \rightarrow \infty$ , la probabilidad que la decisión mayoritaria sea correcta converge a 1.



# BOSQUES ALEATORIOS

Parámetros del algoritmo:

- Los mismos que para árboles de decisión.
- **Número de árboles** ( $\sim 500+$ ).
- **Tamaño de la muestra** (Entero o Proporción)
- **Estado aleatorio**: valor semilla para RNG.



# BOSQUES ALEATORIOS

## PROS

- Robustos y efectivos
- Proveen la importancia de los atributos
- Eficientes en procesamiento (paralelo)
- Usualmente tienen poco sesgo y varianza, pero alta exactitud
- Pueden trabajar con datos sucios o incompletos

## CONTRAS

- Alto tiempo de entrenamiento
- Alto consumo de memoria (cientos o miles de árboles)

The background is a dark gray gradient. In the corners, there are white line-art illustrations of circuit boards or neural network connections. These lines are thin and connect to small white circles, resembling nodes or solder points. The patterns are symmetrical and extend from the edges towards the center.

# DEMO (PYTHON)



“

LA PARTE MÁS IMPORTANTE DEL APRENDIZAJE  
ES EN REALIDAD OLVIDAR.

”

– Naftali Tishby



¡MUCHAS GRACIAS POR SU ATENCIÓN!

