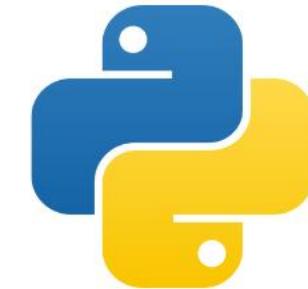




DE EXCEL A PYTHON: ANALISIS DE DATOS INTELIGENTE CON MACHINE LEARNING



www.claumiseimbett.com



Sesión 1. Fundamentos del Análisis Inteligente de Datos (4 h)





Contenido

Sesión 1. Fundamentos del Análisis Inteligente de Datos (4 h)

- Introducción al Machine Learning: tipos de modelos y aplicaciones.
- Conceptos clave: datasets, variables, features, target, entrenamiento y validación.
- Métricas de evaluación: precisión, R^2 , matriz de confusión, hiperparámetros
- Detección y limpieza de outliers (método IQR y Tukey).



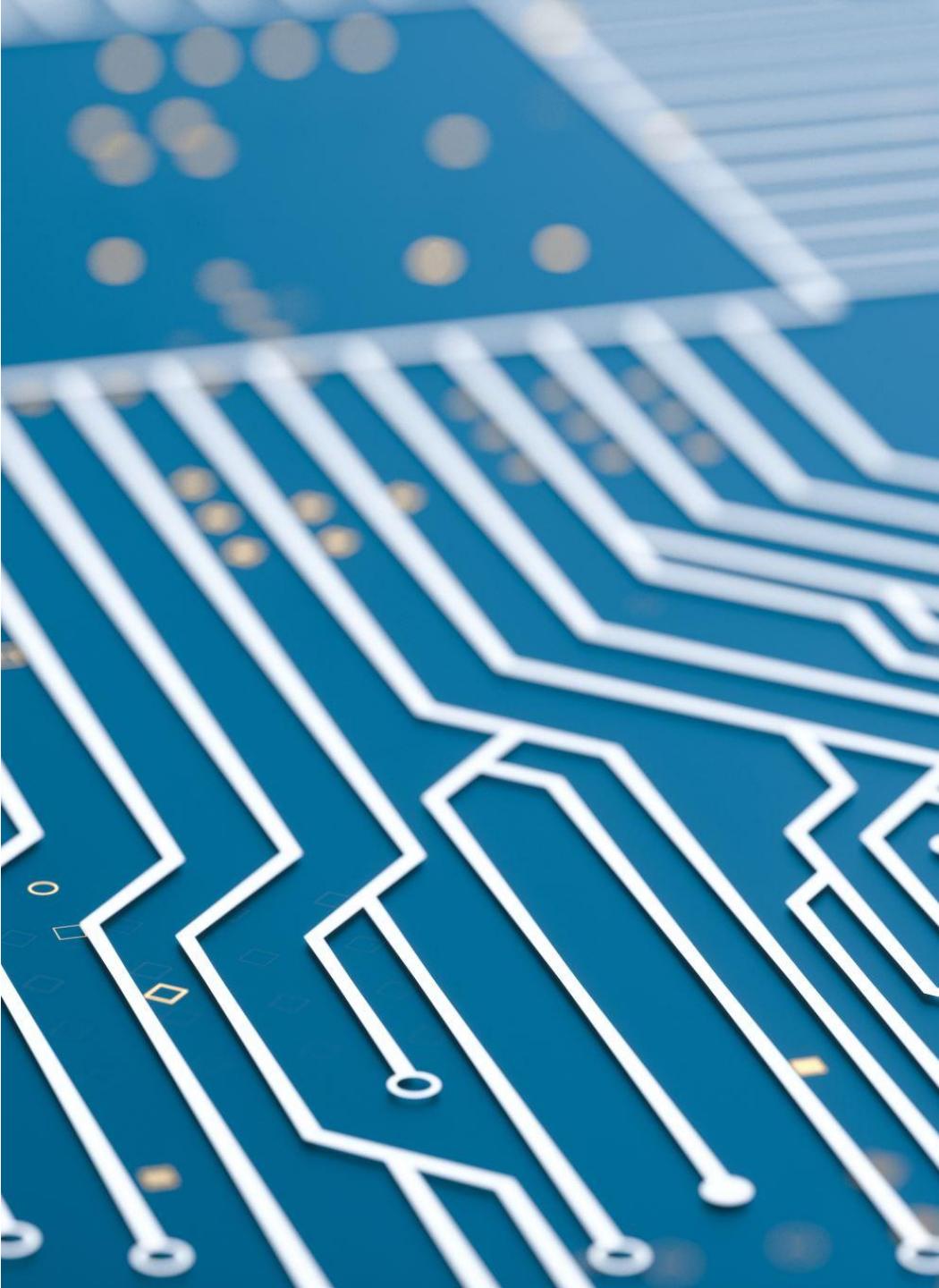
Sesión 1. Fundamentos del Análisis Inteligente de Datos (4 h)

Introducción al Machine Learning: tipos de modelos y aplicaciones.

Machine Learning

Machine learning (aprendizaje automático) es una rama de la **Inteligencia Artificial** que se centra en el desarrollo de algoritmos y modelos capaces de aprender patrones a partir de datos y mejorar su desempeño en una tarea específica sin ser programados de forma explícita para cada caso.

En lugar de seguir instrucciones fijas, un sistema de **machine learning** utiliza, datos históricos para entrenar un modelo, el cual luego puede realizar predicciones, clasificaciones, o detección de patrones.



Machine Learning



Datos: ejemplos o experiencias de los que el sistema aprende.



Modelo: representación matemática/estadística que captura patrones de los datos.



Entrenamiento: proceso de ajustar los parámetros del modelo usando datos.



Evaluación: medición del desempeño del modelo en datos no vistos (*nuevos datos o datos para test*).

Machine learning

¿Qué técnicas?



Supervisado → aprende con ejemplos correctos.



No supervisado → descubre patrones ocultos.



Refuerzo → aprende de recompensas.



Semi-supervisado → mezcla etiquetados y no etiquetados.



Ensembles → combina modelos para ser más robusto.

Técnicas de aprendizaje en Machine Learning

1. Aprendizaje Supervisado

El modelo aprende con **ejemplos etiquetados** (datos de entrada + la respuesta correcta).

- **Ejemplos:**
 - Predecir **biomasa o rendimiento de un cultivo** a partir de imágenes y datos de campo.
 - Predecir el consumo eléctrico dependiendo del estrato y ciudad.



Técnicas de aprendizaje en Machine Learning

2. Aprendizaje No Supervisado

El modelo busca **patrones ocultos sin etiquetas**.

- **Ejemplos:**
 - Agrupar potreros en **zonas homogéneas de vigor vegetal** para fertilización diferenciada.
 - Segmentar **clientes** por patrones de consumo.



Técnicas de aprendizaje en Machine Learning

3. Aprendizaje Semi-Supervisado

Se combinan **pocos datos etiquetados** con **muchos sin etiquetar**.

- **Ejemplos:**

- Identificar **clientes ideales para oferta solar** según su consumo, sector, ciudad, etc; sin tener todos los datos de consumo.
- Mapear **malezas en pasturas**, usando pocas muestras tomadas en terreno y grandes áreas sin etiqueta.



Técnicas de aprendizaje en Machine Learning

4. Aprendizaje por Refuerzo

El modelo aprende por **prueba y error**, recibiendo **recompensas o penalizaciones**.

- **Ejemplos:**

- Optimizar **sistemas de riego inteligentes**, ajustando el agua según humedad satelital + pronóstico climático.
- Un sistema que aprenda a **dar consejos diarios/semanales al cliente para reducir su factura**



Técnicas de aprendizaje en Machine Learning

5. Aprendizaje por Ensamblaje (Ensemble Learning)

Se combinan varios modelos para obtener mayor precisión.

- **Ejemplos :**
 - Usar **Random Forest + XGBoost** para predecir **rendimiento de maíz o pasturas** con datos de satélite + clima + suelo.
 - **Predecir si un cliente puede pagar solar**, o clasificar consumo en *alto / medio / bajo*.



1. Aprendizaje Supervisado

Algoritmo	Descripción
Regresión Lineal (Simple, Multiple y regularizada como: ElasticNet, Ridge y LASSO)	Relación entre variables continuas.
Regresión Logística	Clasifica en dos categorías.
Árboles de decisión	Clasificación mediante reglas jerárquicas.
Random Forest	Conjunto de árboles de decisión para mayor precisión.
SVM (Máquinas de Vectores de Soporte)	Encuentra fronteras óptimas para separar clases.
Redes Neuronales Artificiales	Modelos inspirados en el cerebro, flexibles.

2. Aprendizaje no supervisado

Algoritmo	Descripción
K-Means (Clustering)	Agrupa datos en clusters según similitud.
DBSCAN	Agrupamiento por densidad, detecta anomalías.
PCA (Análisis de Componentes Principales)	Reduce dimensiones conservando información.

3. Aprendizaje semisupervisado

Algoritmo	Descripción
Label Propagation	Propaga etiquetas de pocos datos hacia no etiquetados.
Autoencoders	Extraen características y reducen ruido.

4. Aprendizaje por refuerzo

Algoritmo

Q-Learning

Deep Q-Networks
(DQN)

Descripción

Aprende mediante recompensas acumuladas.

Combina redes neuronales + refuerzo.

5. Aprendizaje por ensamble

Algoritmo

Bagging (Random Forest)

Boosting (**XGBoost**, LightGBM)

Descripción

Combina modelos para reducir varianza.

Combina modelos secuenciales para mejorar precisión.

¿Para que se usan las técnicas de Aprendizaje?

Clasificación y predicción

Tipos de Modelos

Regresión

- Es una técnica que busca **predecir un valor numérico continuo** a partir de uno o varios atributos (variables independientes).
Meta: Encontrar una función que relacione las variables de entrada con una salida numérica.

Clasificación

- Es una técnica que busca **asignar una categoría o clase** a cada observación, en función de sus características.
- **Meta:** Encontrar una frontera de decisión que separe los datos en clases.

Fundamentos matemáticos y aplicaciones

[1] G. James, D. Witten, T. Hastie, and R. Tibshirani, **An Introduction to Statistical Learning with Applications in Python**. New York, NY, USA: Springer, 2023.

[2] A. C. Müller and S. Guido, **Introduction to Machine Learning with Python: A Guide for Data Scientists**. Sebastopol, CA, USA: O'Reilly Media, 2017.

[3] Sebastian Raschka y Vahid Mirjalli. Python Machine Learning. Aprendizaje automático y aprendizaje profundo con Python, scikit-learn y TensorFlow: Segunda Edicion, Marcombo, 2019



Material recomendado para el aprendizaje de la matemática de machine learning

https://youtu.be/ETWrit947nE?si=GnNdyNR_arybO7_8

Algunas técnicas de aprendizaje supervisado, supervisado y ensamble

1. Aprendizaje supervisado

Regresión Lineal (Simple, Multiple y regularizada como: ElasticNet, Ridge y LASSO)

Random Forest

SVM (Máquinas de Vectores de Soporte)

2. Aprendizaje no supervisado

K-Means (Clustering)

3. Aprendizaje por Ensamble

XGBoost

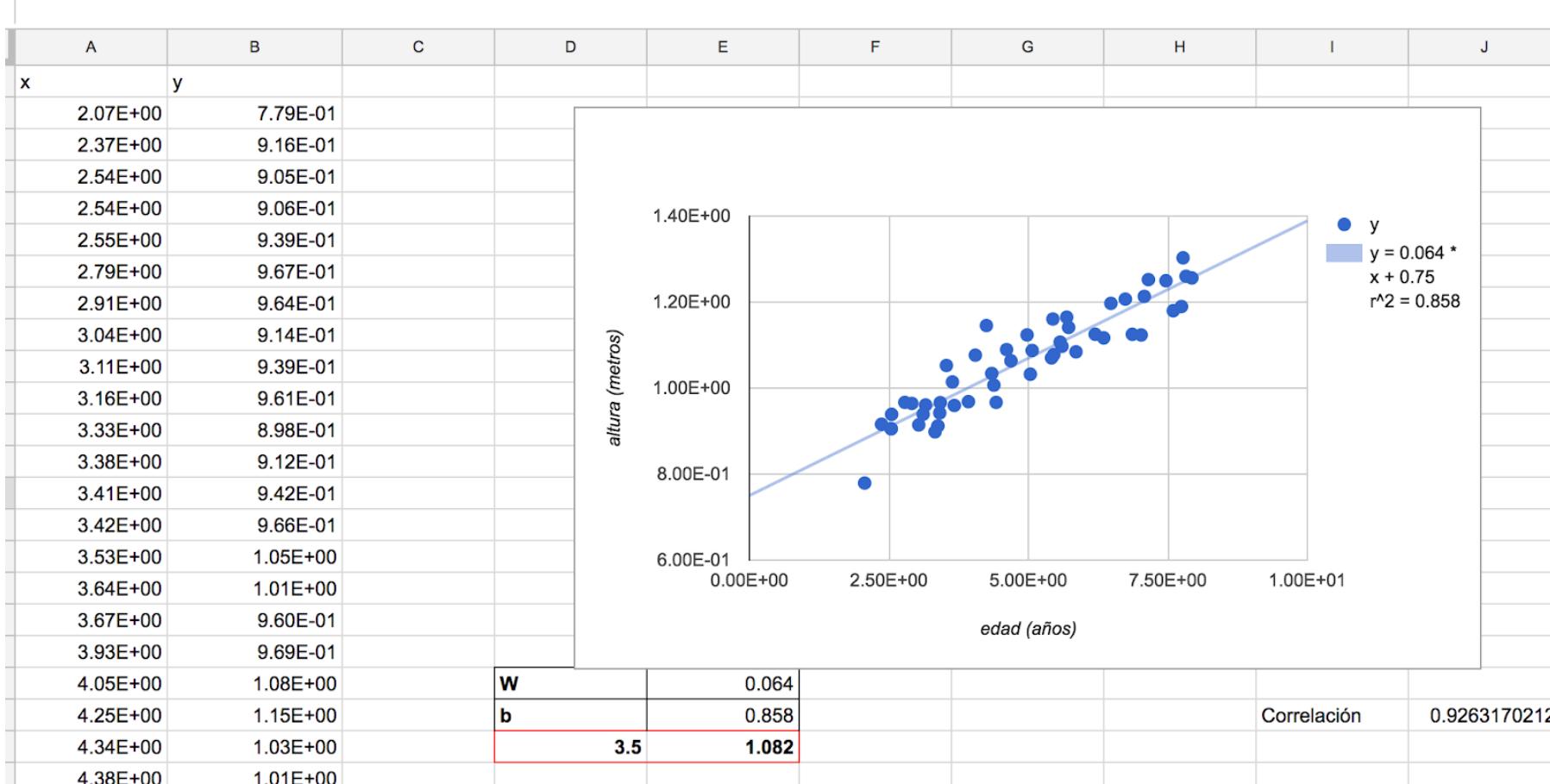
Técnicas de Aprendizaje

1. Aprendizaje supervisado

- Regresión Lineal
 - a. Simple
 - b. Múltiple
 - c. Regularizada como: ElasticNet, Ridge y LASSO)

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$
$$= \lim_{h \rightarrow 0} \frac{(x+h)^2 - x^2}{h}$$
$$= \lim_{h \rightarrow 0} \frac{x^2 + 2xh + h^2 - x^2}{h}$$
$$= \lim_{h \rightarrow 0} \frac{2xh + h^2}{h}$$
$$= 2x$$
$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$
$$f(0) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

Regresión Lineal Simple



Esta foto de Autor desconocido está bajo licencia CC BY-SA

Regresión lineal clásica (Estadística Vs. ML)

Regresión lineal clásica (estadística):

- Busca **explicar** la relación entre una variable dependiente Y y una o varias independientes X .
- El interés suele estar en los **coeficientes** (β s): cuánto cambia Y si X cambia también.
- Ejemplo: "Por cada hora adicional de estudio, la nota aumenta 0.5 puntos".



Regresión lineal clásica (Estadística Vs. ML)

Regresión lineal en machine learning (ML):

- Se usa más para **predecir** que para explicar.
- El foco no es tanto el valor de los coeficientes, sino que también se llegue a un modelo **generalizable** para **nuevos datos**.
- Ejemplo: predecir el consumo eléctrico dependiendo de la ciudad, estrato y clima.



Regresión lineal clásica (Estadística Vs. ML)

Clásica:

- Normalmente usa **Mínimos Cuadrados Ordinarios (OLS)**.
- Busca minimizar el **error cuadrático promedio**

Regresión lineal clásica (Estadística Vs. ML)

Machine Learning:

- Puede usar OLS para disminuir el error de ajuste con los datos de entrenamiento, pero también métodos iterativos (ej. Gradient Descent) que buscan minimizar el error y son ideales para manejar grandes volúmenes de datos.
- A menudo incluye **regularización** (Ridge, LASSO, Elastic Net) para evitar sobreajuste.
- El interés es optimizar la **función de pérdida + término de regularización**.



Objetivo con Machine Learning

Más allá de analizar las relaciones entre variables, el *machine learning* se centra en entrenar modelos con capacidad de **generalización**, es decir, que mantengan un buen desempeño al predecir sobre datos no utilizados durante el entrenamiento.

Revisión de la matemática de la Regresión Lineal

¿Qué hallamos en la regresión lineal?

Regresión Lineal simple

- La regresión lineal simple modela la relación entre una variable dependiente y una independiente a través de una recta:

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$$

donde:

y_i : variable dependiente (eje vertical - y)

x_i : variable independiente (eje horizontal - x)

β_0 : intercepto (eje vertical -y)

β_1 : pendiente (inclinación del a recta)

ε_i : término de error

Regresión lineal simple

- Los coeficientes β_0 y β_1 se estiman con:

$$\beta_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

El objetivo del modelo es **estimar los parámetros β_0 y β_1** a partir de los datos, usualmente usando el **método de mínimos cuadrados**:

Regresión lineal simple

$$\widehat{\beta}_0 = \bar{y} - \widehat{\beta}_1 \bar{x} \quad \widehat{\beta}_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2},$$

$\widehat{\beta}_1$ es la pendiente estimada

Las variables con “barras” son estimadas.

Regresión lineal múltiple

- La regresión lineal múltiple extiende la regresión lineal simple a varios predictores:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \varepsilon_i$$

donde:

- y_i : variable dependiente
- x_{ip} : variables independientes
- β_0 : intercepto
- β_p : coeficientes de regresión
- ε_i : término de error

Regresión lineal múltiple

El modelo puede escribirse en notación matricial como:

$$Y = X\beta + \varepsilon$$

- donde:
- Y es el vector de respuestas o salida ($n \times 1$)
- X es la matriz (arreglo de filas y columnas) de predictores ($n \times p$)
- β es el vector de coeficientes ($p \times 1$)
- ε es el vector de errores ($n \times 1$)

Los parámetros se estiman mediante el método de mínimos cuadrados (las variables con barra son las que se deben estimar):

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

Regresión Lineal Ridge, LASSO, ElasticNet

1. Regresión Ridge (o Tikhonov)

- Minimiza la suma de errores cuadrados más una penalización **L2**:

$$\widehat{\beta^{ridge}} = \arg \min_{\beta} \left\{ \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2 \right\}$$

Regresión Lineal Ridge, LASSO, ElasticNet

2. Regresión LASSO (Least Absolute Shrinkage and Selection Operator)

- Minimiza la suma de errores cuadrados más una penalización L1:

$$\widehat{\beta^{lasso}} = \arg \min_{\beta} \left\{ \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p |\beta_j| \right\}$$

Regresión Lineal Ridge, LASSO, ElasticNet

3. Regresión Elastic Net

- Combina L1 (LASSO) y L2 (Ridge) con un parámetro de mezcla α :

$$\widehat{\beta^{EN}} = \arg \min_{\beta} \left\{ \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \left[\alpha \sum_{j=1}^p |\beta_j| + (1 - \alpha) \sum_{j=1}^p \beta_j^2 \right] \right\}$$

A woman with a robotic head and a background of circles.

¿Cómo lo hago?

Calma! 😊 Estos modelos se implementan en un lenguaje de programación como Python.

¿Qué es penalización y regularización?

Una **penalización** es un término adicional (para **regularización**) que se agrega a la **función de costo** de un modelo con el objetivo de **evitar el sobreajuste (overfitting)**.

Esta penalización **castiga los coeficientes muy grandes sesgados a una variable** —que representan un exceso de peso en el modelo— o la presencia de **demasiadas variables altamente correlacionadas**, promoviendo así modelos más simples y generalizables.

Diferencia entre Ridge y LASSO



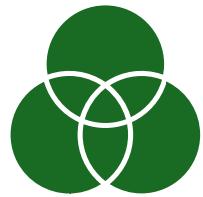
Ridge : reduce los coeficientes, y estos quedan “pequeños”, reduciendo el sesgo por alguna variable.



LASSO : puede eliminar variables completas, dejando algunos coeficientes exactamente en cero, promoviendo la reducción de variables altamente correlacionadas.

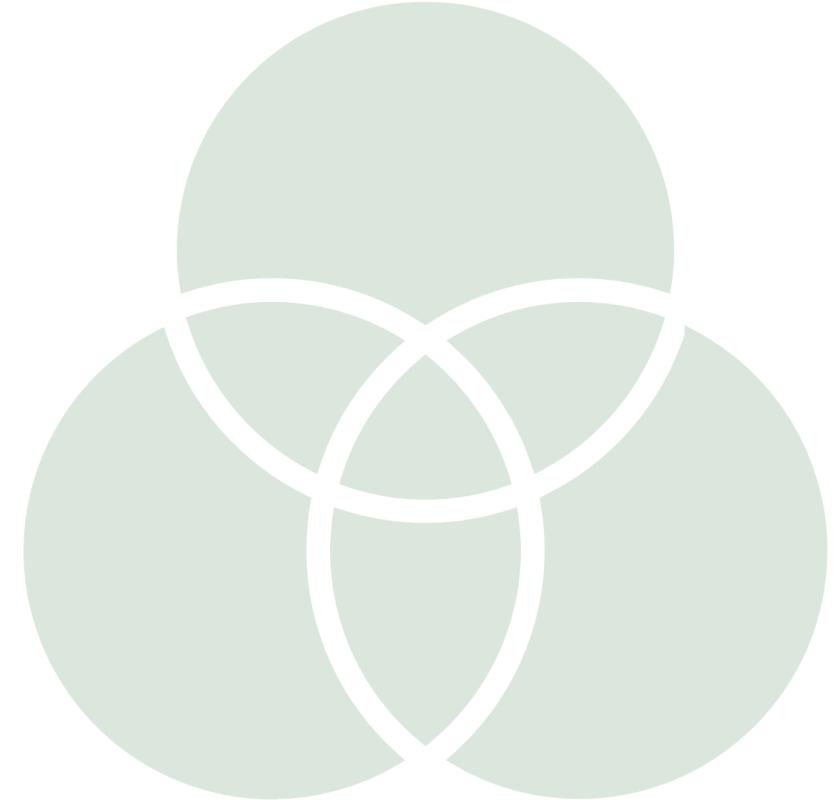


Estas penalizaciones están relacionadas con los *hiperparámetros* del modelo



Conclusión

Contamos con la regresión lineal simple y la regresión lineal múltiple como técnicas de **aprendizaje supervisado**. En esta última, la incorporación de términos de penalización permite “regular” el modelo, dando lugar a variantes de regresión lineal como **Lasso**, **Ridge** y **Elastic Net**, las cuales se presentan como excelentes opciones para trabajar con datos que incluyen múltiples variables. **Aplicación en Machine Learning: Regresión.**



Random Forest

Imaginemos algo más intuitivo: en lugar de confiar en un solo modelo, “sembramos” muchos árboles y les pedimos a todos su opinión. Cada árbol aprende de una parte distinta de los datos y genera su propia predicción. Luego, un **votante inteligente combina sus respuestas**: si es clasificación, elige la clase más votada; si es regresión, calcula el promedio.

Así funciona **Random Forest**: un equipo de árboles que, trabajando juntos, produce decisiones más estables y confiables que un único modelo aislado. **Aplicación en Machine Learning: Regresión y Clasificación**

Random Forest

Random Forest es un método de **aprendizaje supervisado** y conjunto (*ensemble learning*) que combina múltiples árboles de decisión para crear un modelo más robusto y preciso.

Random Forest se basa en el principio de **bagging**, que reduce la **varianza** del modelo mediante el promedio de múltiples estimadores.



¿Qué es el bagging (Bootstrap Aggregating)?

Es una estrategia que consiste en entrenar **varios modelos del mismo tipo** (en este caso, árboles de decisión) usando **diferentes subconjuntos aleatorios de los datos**, obtenidos mediante muestreo con reemplazo (*bootstrap*). Luego, sus predicciones se **combinan** para obtener una respuesta final más robusta.

¿Qué es el Bootstrap?

- **Bootstrap** es una técnica estadística que consiste en crear múltiples muestras a partir del conjunto de datos original mediante muestreo con reemplazo.
- Si el dataset tiene 100 registros, el bootstrap genera una muestra de tamaño 100 también, pero seleccionada aleatoriamente con reemplazo.

Resultado típico: algunos registros se repiten, otros quedan por fuera.

Ejemplo de la bolsa de caramelos para comprender el bootstrap

Imagina una bolsa con **5 caramelos** numerados del 1 al 5.

Si hacemos una muestra de 5 caramelos **con reemplazo**:

1. Sacas un caramelo → digamos el #2
2. **Lo devuelves a la bolsa**
3. Vuelves a sacar otro
4. Repites hasta tener 5 extracciones
5. Como cada caramelo vuelve a la bolsa después de ser tomado, **puede salir repetido**.

Una muestra posible podría ser:

[2, 4, 2, 1, 5]

El caramelo #2 apareció dos veces y el #3 no apareció.



¿Qué es la varianza?

La varianza es el **promedio de las diferencias al cuadrado entre cada dato y la media**.

- Para una muestra:

$$s^2 = \frac{1}{n - 1} \sum_{i=1}^n (x_i - \bar{x})^2$$

Random Forest reduce la **varianza** del modelo mediante el promedio de múltiples estimadores (árboles decisión).

Random Forest

Conjunto de datos

$$D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

Donde:

$x_i \in \mathbb{R}_P$ (p características)

$y_i \in \mathbb{R}$ (regresión) o $y_k \in 1, 2, \dots, K$ (clasificación)

n = número de muestras



Random Forest

Primer paso: Hacer Bootstrap Aggregating (Bagging)

Crear B muestras Bootstrap D_b mediante muestreo con reemplazo de los datos originales de cada árbol (unos se repiten y otros quedan fuera (son los datos out-of bag), pero son útiles para validación interna), buscando reducir la varianza en cada resultado con la muestra Bootstrap para mejorar la precisión del modelo.

Random Forest

Segundo paso: el entreno ☺

- Entrenar (Trainning) un árbol estimador T_b (árbol de decisión) para cada muestra “Bootstrap”
- Combinar predicciones (Aggregating)

Regresión (promedio), donde \hat{f} es lo predicho:

$$\hat{f}_{RF}(x) = \frac{1}{B} \sum_{b=1}^B T_b(x)$$

Clasificación (voto mayoritario), donde \hat{f} es lo también lo predicho:

$$\hat{f}_{RF}(x) = \text{mode}\{T_1(x), T_2(x), \dots, T_b(x)\}$$

Random Forest

Tercer paso: Feature Bagging

Segunda Fuente de Aleatoriedad es el Feature Bagging

En cada nodo del árbol, en lugar de evaluar todas las variables, solo se consideran m características aleatorias de las p (*predictoras*) totales

Se decorrelacionan los árboles (mejor split del conjunto para crecer el árbol), buscando que los árboles sean diferentes entre sí.

Valores típicos para hacer el split:

Clasificación: $m = \sqrt{p}$

Regresión: $m = \frac{p}{3}$

Random Forest

Crecimiento de árboles

- Cada árbol se expande hasta una cierta profundidad o hasta que mejore el ajuste.
- Normalmente no se podan (a diferencia de los árboles clásicos), porque la diversidad mejora el resultado.

Combinación de resultados

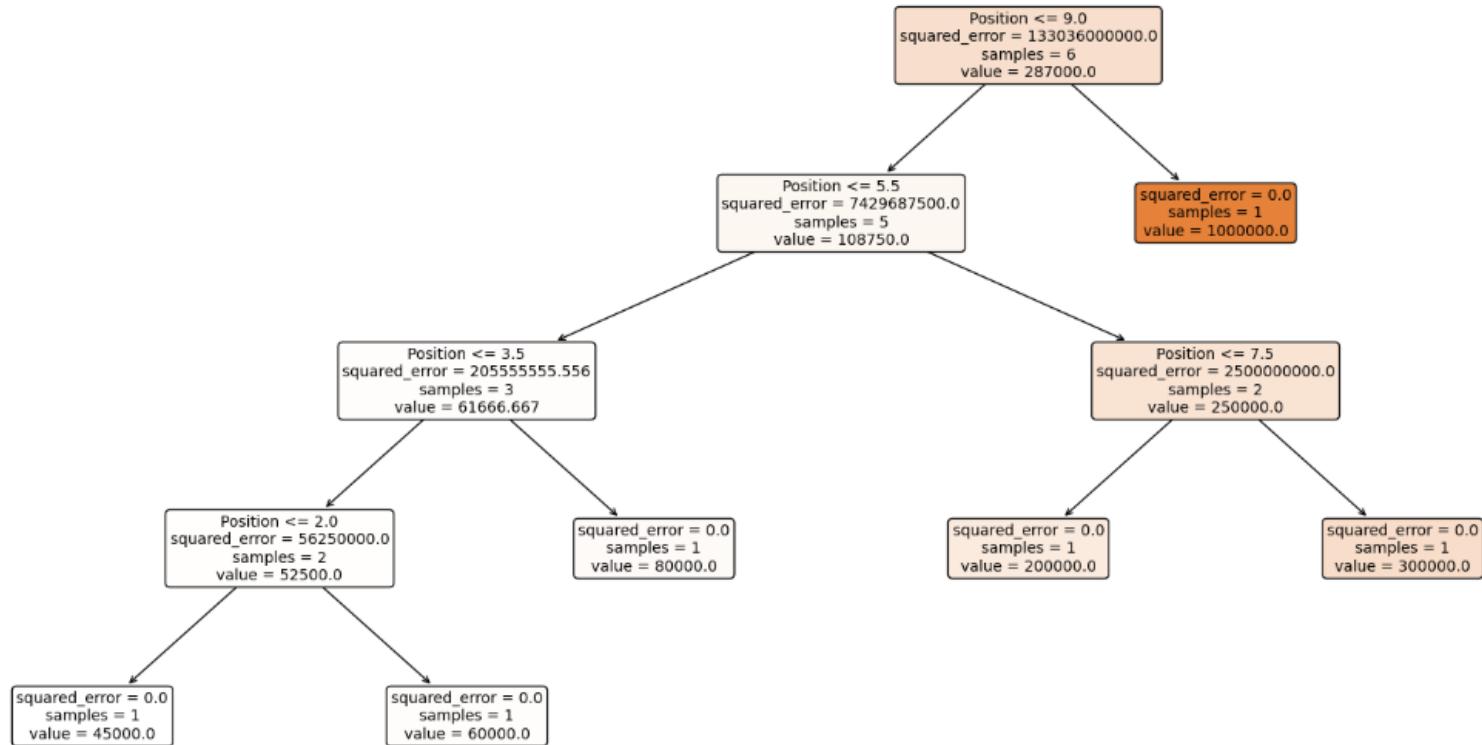
- Una vez construidos todos los árboles (n árboles en el bosque):
 - **Clasificación:** cada árbol “vota” por una clase, y la clase con más votos es la predicción final (mayoría).
 - **Regresión:** se promedian las predicciones de todos los árboles.

Random Forest

Cuarto paso: evaluación del modelo

- Se pueden usar los datos “out-of-bag” (los que quedaron por fuera de cada bootstrap) como validación interna.

Decision Tree from Random Forest



Random Forest

Random Forest

Queremos **predecir el rendimiento de un cultivo (kg/ha)** a partir de variables obtenidas por sensores e imágenes satelitales:

- **NDVI** (índice de vegetación).
- **Precipitación acumulada (mm).**
- **Temperatura media (°C).**

Esto es un **problema de regresión**, ya que la variable de salida es numérica (rendimiento).



Random Forest

Ejemplo:

Entrenamiento: Se construye un bosque de 200 árboles de decisión.

- **Predictión:** El modelo estima el rendimiento de cada parcela.
- **Importancia de variables:** El gráfico muestra cuáles variables (estrato, tipo de cliente, zona, u otra) influyen más en la predicción.

Random Forest

Predecir la probabilidad de compra de paneles solares, es clave elegir variables que reflejen **necesidad, capacidad económica, comportamiento, contexto y motivación verde usando estas variables:**

- kWh_promedio_mensual
- valor_factura_promedio
- kW_pico
- tipo_cliente
- estrato/tarifa_actual
- radiacion_solar
- zona



Random Forest

Ejemplo:

Entrenamiento: Se construye un bosque de 200 árboles de decisión.

- Predicción:** El modelo estima la probabilidad de compra de paneles solares

- Importancia de variables:** El gráfico muestra cuáles variables (NDVI, precipitación, temperatura) influyen más en la predicción.

Máquinas de Soporte Vectorial



Trazando fronteras con precisión: Las Máquinas de Vectores de Soporte nos muestran que, aun entre miles de datos, solo unos pocos puntos verdaderamente relevantes pueden definir el límite entre el acierto y el error. El verdadero valor está en identificar lo esencial para trazar fronteras claras y precisas. **Aplicación en Machine Learning: Clasificación.**

SVM (Máquinas de Vectores de Soporte)

SVM (Support Vector Machine o Máquina de Vectores de Soporte) es un algoritmo de **aprendizaje supervisado** cuyo modelo matemático se basa en encontrar el **hiperplano óptimo** que separa las clases de datos con el margen máximo.



SVM (Máquinas de Vectores de Soporte)

El objetivo es encontrar un hiperplano definido por:

$$w \cdot x + b = 0$$

- w es el vector de pesos (normal al hiperplano)
- x es el vector de características
- b es el término de sesgo



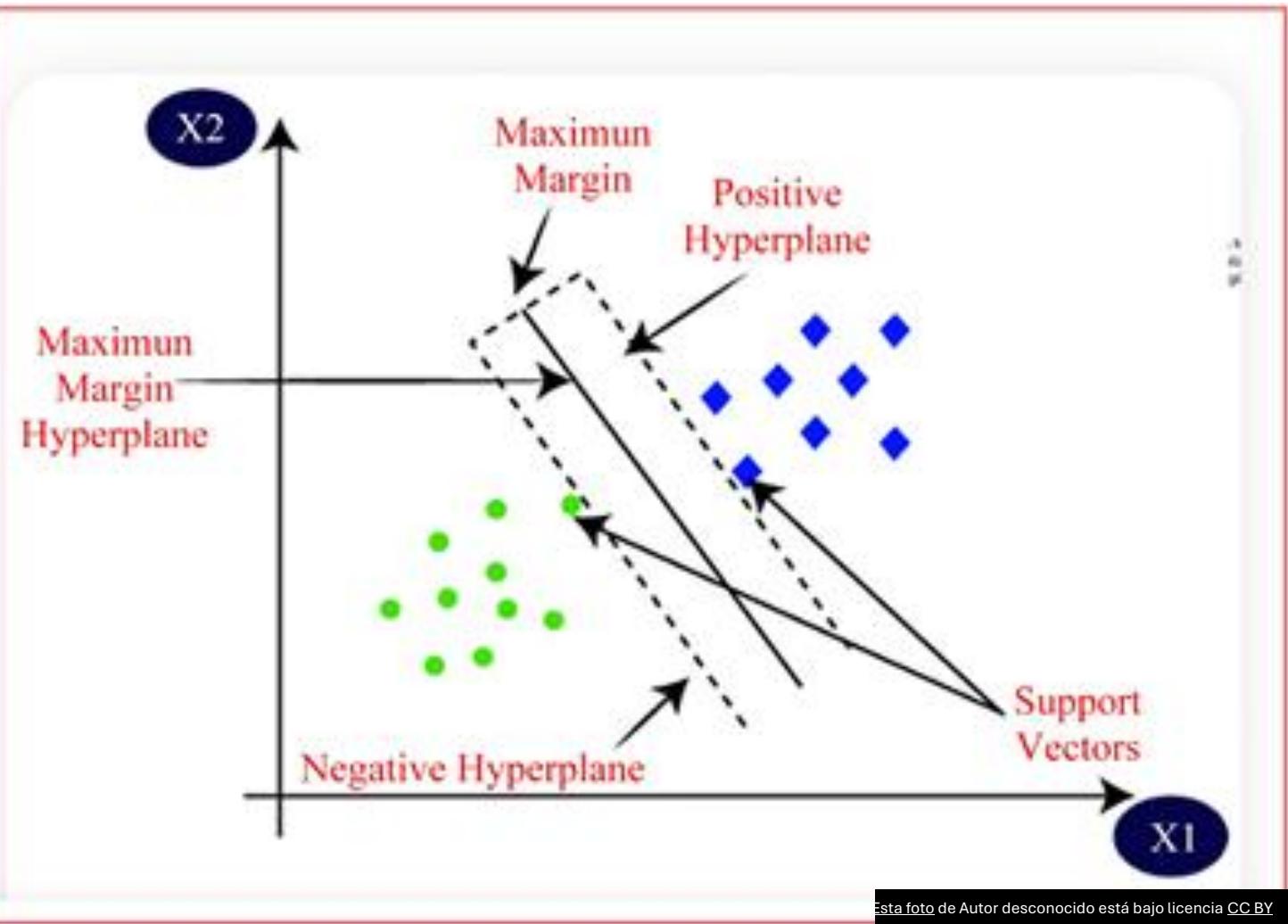
¿Qué es un hiperplano?

Un **hiperplano** es una superficie geométrica de dimensión $n - 1$ inmersa en un espacio de dimensión n .

En un espacio 2D es una línea recta



SVM gráfico



¿Qué son los vectores de soporte?

En el contexto de **Support Vector Machines (SVM)**, los **vectores de soporte** son los puntos de datos más relevantes para definir la frontera de decisión entre clases.

En una clasificación 2D:

- El **hiperplano** es la línea que divide las clases.
- Los **márgenes** son dos líneas paralelas a esa frontera.
- Los **vectores de soporte** son los puntos que tocan o quedan más cerca de esos márgenes.

Idea clave: Los vectores de soporte son los “*guardianes*” del límite de decisión; gracias a ellos, el modelo logra la máxima separación entre clases.



Ejemplo con SVM

Problema: Queremos clasificar zonas de un cultivo en dos clases usando imágenes satelitales:

- Clase 1: “estrés hídrico” 
- Clase 2: “vegetación sana” 

Variables (features) de entrada:

- NDVI (índice de vegetación).
- Humedad del suelo.
- Temperatura superficial.

Salida: Etiqueta binaria \rightarrow 0 = estrés, 1 = sano.



Ejemplo con SVM

Problema: Queremos clasificar los clientes en dos clases usando la información de estrato, ciudad y consumo energético

- Clase 1: “**compra panel**”
- Clase 2: “**no compra panel**”

Salida: Etiqueta binaria $\rightarrow 0 = \text{no compra}$, $1 = \text{compra}$.



K-means

K-Means nos recuerda que, aun en el aparente desorden, siempre es posible encontrar grupos con significado. Cuando aprendemos a ver patrones donde otros solo ven datos dispersos, transformamos información en conocimiento y decisiones con claridad. Aplicación en Machine Learning: Clasificación.

K-means (k-medias)

K-means es un **algoritmo de Machine Learning no supervisado** que se usa para **clustering** (agrupamiento). Su objetivo es **dividir un conjunto de datos en k grupos** (clusters), de tal forma que:

- Los puntos dentro de un mismo grupo sean **lo más parecidos posible**.
- Los puntos de diferentes grupos sean **lo más distintos posible**.

K-means (k-medias)

Minimizar la suma de distancias cuadradas intracluster:

$$J = \sum_{i=1}^k \sum_{x \in c_i} \|x - \mu_i\|^2$$

K es el numero de cluster

c_i es el conjunto de puntos en el cluster i

μ_i es el centroide del cluster i

$\|x - \mu_i\|^2$ distancia euclíadiana al cuadrado.



¿Qué es un cluster?

- Un cluster es un conjunto de datos que se agrupan porque presentan similitudes, formando patrones o categorías naturales sin necesidad de etiquetas previas

K-means (k-medias)

Pasos del algoritmo:

1. Inicialización: seleccionar k centroides $\mu_1, \mu_2, \mu_3, \dots, \mu_k$
2. Asignación: Cada punto x_j se asigna al clúster más cercano:

$$c_i = \left\{ x_j : \|x_j - \mu_i\|^2 \leq \|x_j - \mu_l\|^2 : \forall l \in \{1, \dots, k\} \right\}$$

3. Actualización : Recalcular centroides como medidas de puntos asignados:

$$\mu_i = \frac{1}{C_i} \sum_{x \in C_i} x$$

4. Convergencia: Repetir 2-3 hasta que centroides no cambien o $\Delta J < \epsilon$

Ejemplo k-means - Agricultura



Supongamos que tengo imágenes satelitales de un cultivo y quiero **segmentar áreas según vigor** (NDVI):



Pongo $k=3$ → tres grupos:

Cluster 1: áreas de bajo vigor (estrés hídrico).

Cluster 2: vigor medio.

Cluster 3: alto vigor (pastura sana).



El algoritmo encuentra los centroides y agrupa los píxeles.



Resultado: un **mapa temático** con tres zonas de manejo.



Ventajas: No se requiere entrenamiento.

Ejemplo k-means - Energía

Segmentación de clientes para paneles solares con K-Means (Resumen)

Variables recomendadas para el clustering:

- Consumo promedio mensual (kWh)
- Valor de la factura de energía
- Nivel socioeconómico / capacidad de pago
- Tipo de vivienda (casa, apartamento, finca)
- Interés previo en energía solar
- Uso de electrodomésticos eficientes (comportamiento eco)
- Ubicación geográfica (potencial solar)



Ejemplo k-means

Ejemplo de clusters resultantes:

- **Cluster 1:** Alto consumo + alta capacidad → **Cliente ideal para sistema solar completo**
- **Cluster 2:** Consumo medio + interés alto + baja capacidad → **Cliente potencial con financiación**
- **Cluster 3:** Bajo consumo + bajo interés → **Baja probabilidad de compra**

Beneficios de segmentar con K-Means:

- Priorizar clientes con mayor probabilidad de compra
- Personalizar ofertas según el perfil del cliente
- Reducir costos comerciales y de marketing
- Definir estrategias de venta y financiación más efectivas



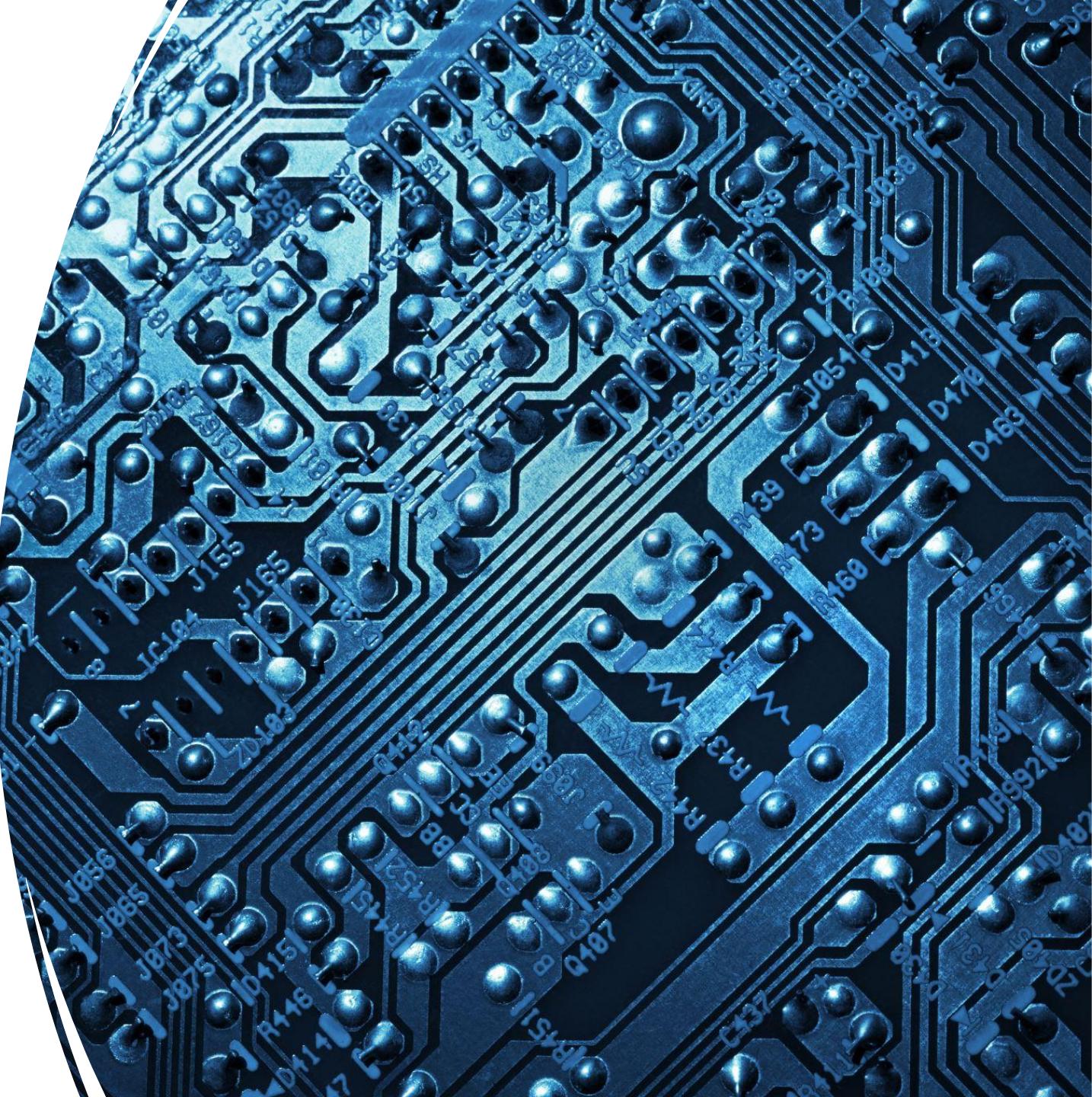
Datos procesados con machine learning

los datos dejan de ser simples cifras para convertirse en **información significativa y estratégica**: permiten construir modelos de predicción, clasificar comportamientos, segmentar poblaciones o clientes, y respaldar **decisiones inteligentes basadas en evidencia**, generando a la vez conocimiento y ventaja competitiva. Machine Learning es un pilar de la Inteligencia Artificial, siendo que el aprendizaje es un primer paso para sistemas que emulen la inteligencia humana.

Inteligencia Artificial (IA)

La Inteligencia Artificial (IA) busca crear sistemas capaces de realizar tareas que normalmente requieren inteligencia humana, como el razonamiento, el aprendizaje, la percepción, la comprensión del lenguaje y la toma de decisiones.

En otras palabras, la IA trata de dotar a las máquinas de la capacidad de simular la inteligencia humana en distintos niveles, desde acciones simples (reconocer una voz o clasificar una imagen) hasta complejas (conducir un vehículo autónomo o diagnosticar una enfermedad).



Diferencias IA y ML

Aspecto	Inteligencia Artificial	Machine Learning
Alcance	Campo general	Subcampo de la IA
Método	Usa reglas, lógica, conocimiento experto y ML	Usa algoritmos de aprendizaje basados en datos
Objetivo	Simular inteligencia humana	Aprender de datos para predecir o clasificar
Ejemplo	Un robot que planifica rutas	Un modelo que predice la ruta más rápida con datos históricos

Diferencias entre IA y ML

IA ES EL
UNIVERSO
COMPLETO.



ML ES UN
PLANETA
DENTRO DE ESE
UNIVERSO.



Sesión 1. **Fundamentos del Análisis Inteligente de Datos (4 h)**

Conceptos clave: datasets, variables, features, target, entrenamiento y validación.

Dataset



Un **dataset** es un conjunto organizado de datos utilizado para analizarlos o entrenar modelos de Machine Learning. Incluye **filas** y **columnas**



Filas → cada una representa un caso, instancia u observación



Columnas → representan variables o características del estudio



Ejemplo sencillo:
Un dataset de clientes para paneles solares podría incluir: edad, consumo eléctrico, ciudad, ingresos y si compró o no el sistema.

Diferencia entre datos y dataset

- **Datos:** Son elementos individuales de información, como números, textos, mediciones o registros aislados. Ejemplo: “350 kWh”, “Estrato 4”, “Sí compró”.
- **Dataset:** Es un conjunto estructurado de datos, organizado generalmente en forma de tabla con filas (observaciones) y columnas (variables). Es decir, es una colección de datos lista para analizar o usar en Machine Learning. El dataset puede contener datos atípicos (outliers) y requerir de limpieza.

Tipos de conjuntos de datos (dataset)

Conjunto de datos (dataset) de entrenamiento

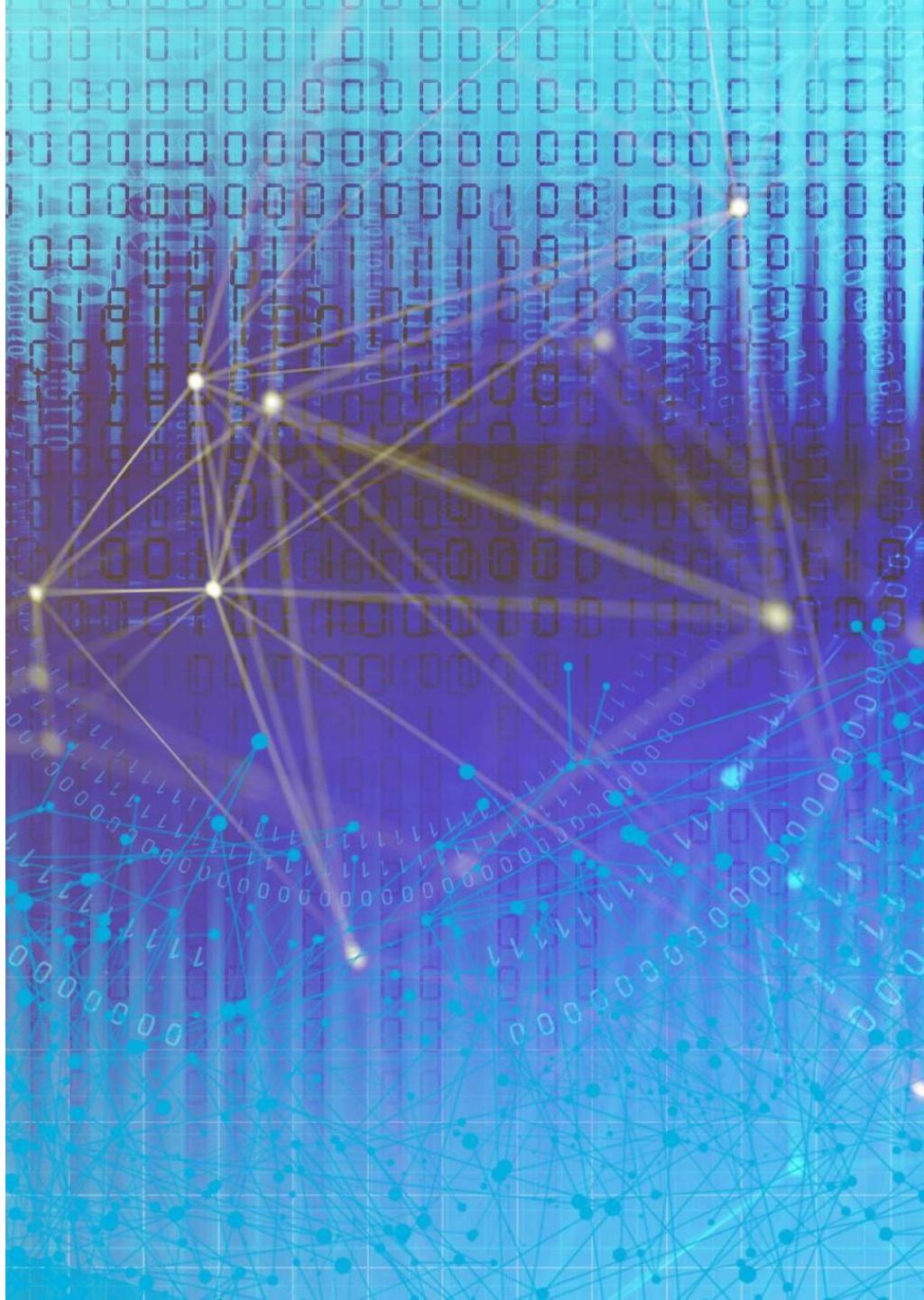
El conjunto de entrenamiento se utiliza para enseñar al modelo a identificar patrones y aprender de los datos.

Conjunto de datos (dataset) de validación

El conjunto de validación ajusta los parámetros del modelo y ayuda a prevenir el sobreajuste a los datos de entrenamiento.

Conjunto de datos (dataset) de prueba

El conjunto de prueba evalúa el rendimiento final del modelo y verifica su capacidad de generalización con datos no vistos.



Tipos de conjuntos de datos (dataset)

- Entrenamiento (training set)
- Validación (validation set)
- Prueba (test set)

Generalmente se divide el dataset así:

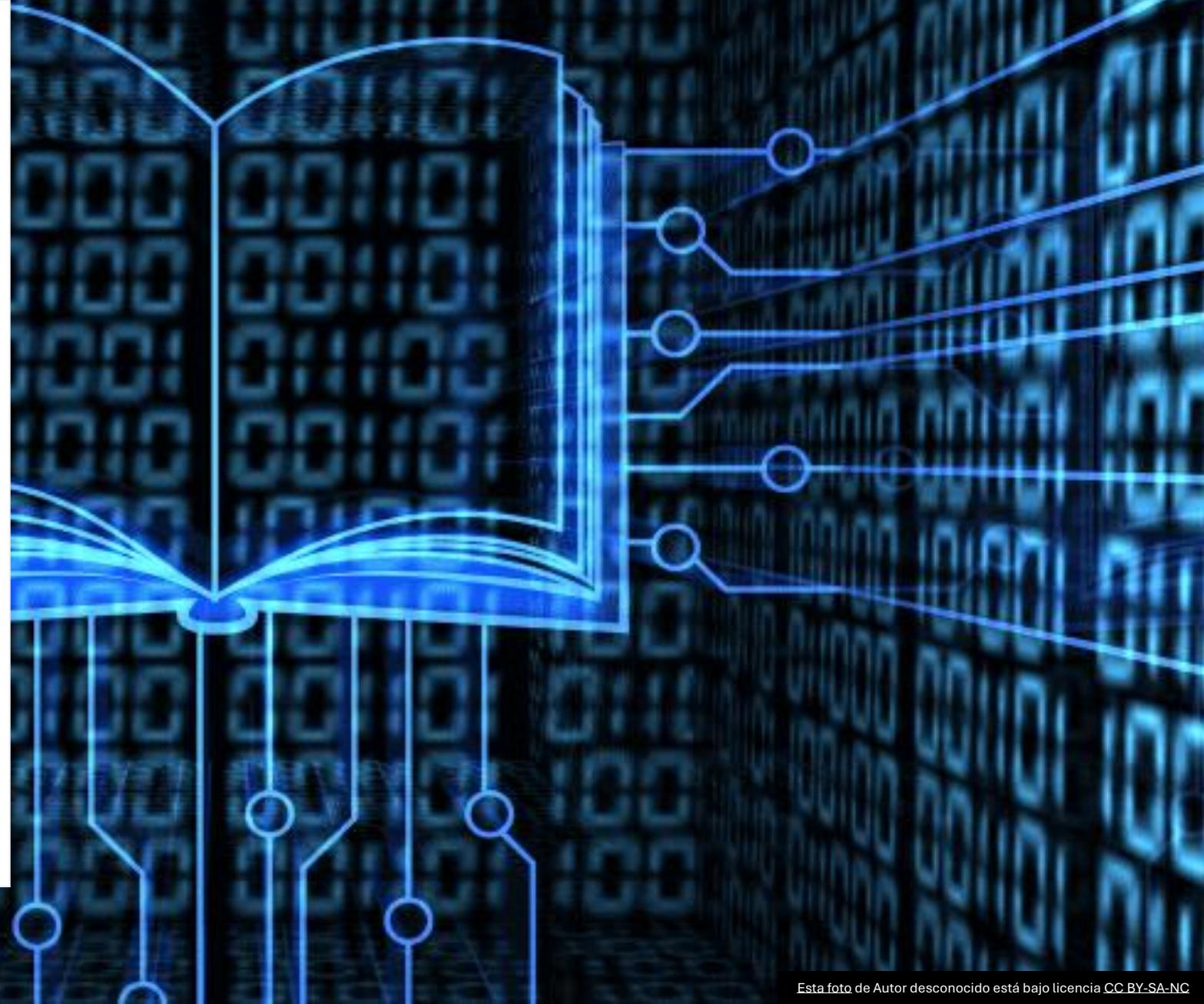
70% para entrenamiento y 30 % validación, o 80% para entrenamiento y 20 % validación de un dataset inicial. Y un dataset de prueba para probar la generalización del modelo.

Repositorios de datasets

<https://zenodo.org/>

<https://huggingface.co/> (machine learning)

<https://data.mendeley.com/>



Tipos de variables: categóricas y no categóricas



Variables categóricas

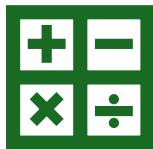
- Son variables que representan **categorías o grupos**. No expresan cantidad numérica sino *tipos* o *clases*.

Ejemplos:

- Género: *Masculino / Femenino*
- Tipo de vivienda: *Casa / Apartamento / Finca*
- Ciudad: *Bogotá / Medellín / Cali*
- ¿Compró paneles solares?: *Sí / No*

Identifican categorías, no cantidades.

Tipos de variables: categóricas y no categóricas



Variables no categóricas (numéricas)

Son variables que representan **valores numéricos medibles** y permiten operaciones matemáticas como sumar, restar o promediar.

Ejemplos:

- Consumo eléctrico: 350 kWh/mes
- Edad: 42 años
- Ingresos mensuales: $\$3.500.000$
- Temperatura: 28°C

Miden cantidades y permiten análisis matemático.



Features

Los features son las **columnas** del dataset que contienen la información usada para predecir una variable objetivo (*target*) tanto para regresión como para predicción.

Ejemplo de variables feature



CONSUMO
ELÉCTRICO
MENSUAL (KWH)



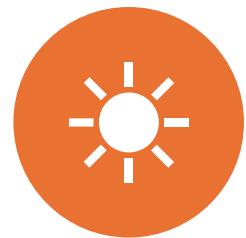
INGRESOS DEL
HOGAR



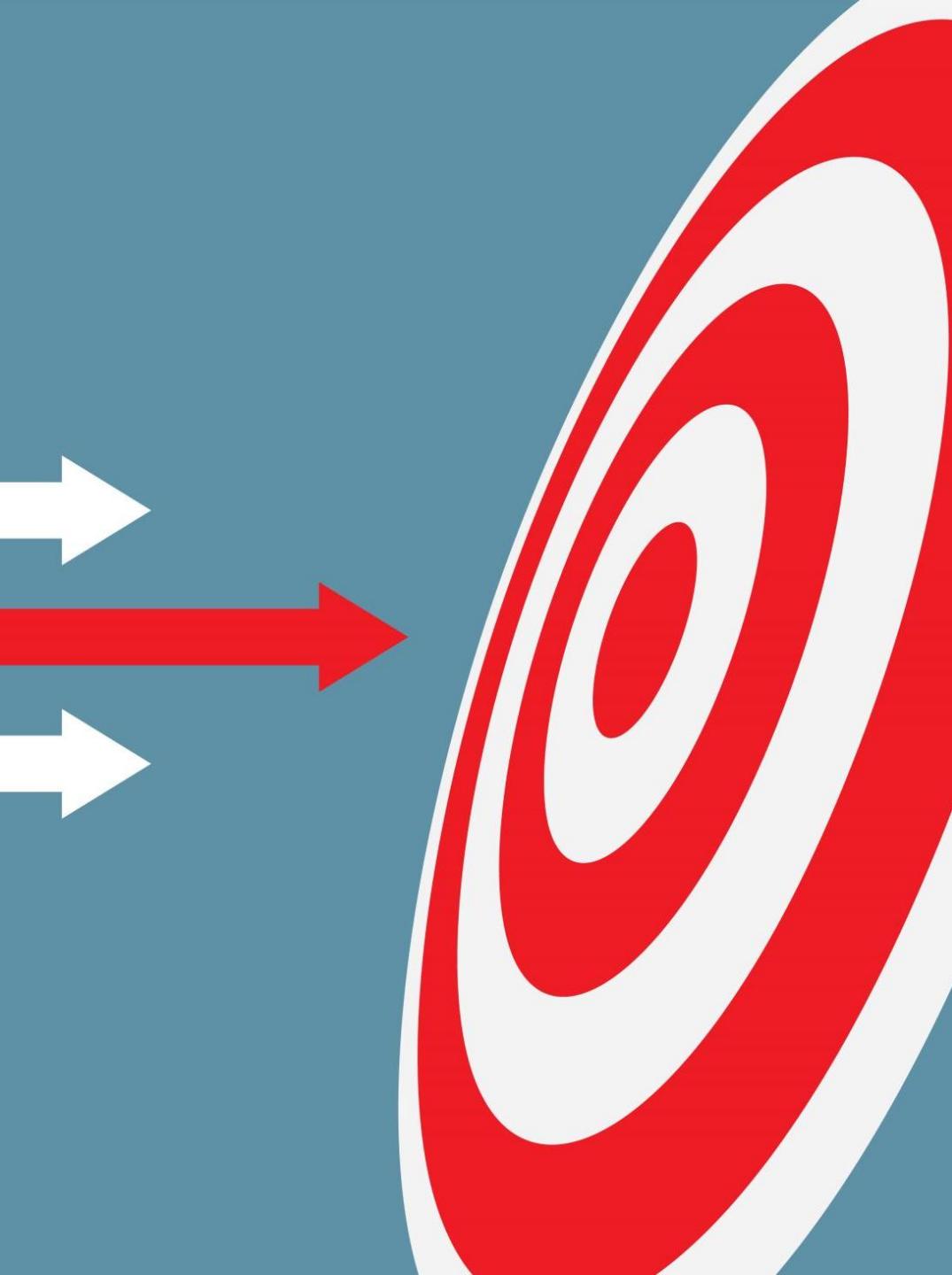
TIPO DE VIVIENDA



CIUDAD



INTERÉS PREVIO
EN ENERGÍA
SOLAR



Target

es la **variable objetivo** que se desea predecir o estimar a partir de los datos. El *target* es el resultado o la respuesta que el modelo intenta aprender.

Ejemplo de targets

Contexto	Target
Energía solar	¿Comprará paneles? (Sí/No)
Predicción de consumo	Consumo eléctrico futuro (kWh)
Marketing	Probabilidad de conversión (%)
Salud	Diagnóstico: sano/enfermo



Sesión 1. Fundamentos del Análisis Inteligente de Datos (4 h)

Métricas de evaluación: precisión, R^2 , matriz de confusión, hiperparámetros

¿Cómo evalúo mi modelo de machine learning?

Cuantificar los errores en Machine Learning es esencial: nos permite evaluar el desempeño del modelo, comparar enfoques y ajustar sus parámetros para reducir la diferencia entre las predicciones y la realidad. Medir el error es el primer paso hacia un modelo más preciso y robusto

Métricas de evaluación de modelos ML (regresión)

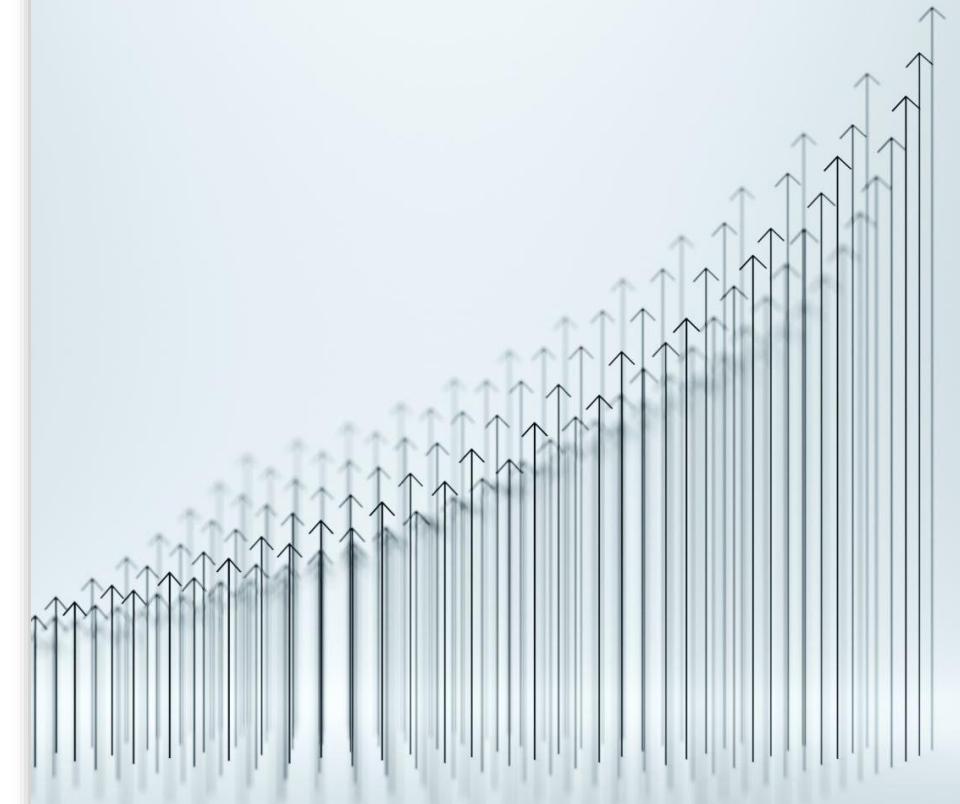
1. MAE – Mean Absolute Error (Error Absoluto Medio)

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

Calcula el **promedio de los errores absolutos**.

Es fácil de interpretar: indica, en promedio, cuánto se equivoca el modelo en las predicciones.

Ejemplo: si MAE = 2, significa que en promedio me equivoco **2 unidades** (kg, cm, etc.).



Métricas de evaluación de modelos ML (regresión)

2. RMSE – Root Mean Squared Error (Raíz del Error Cuadrático Medio)

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

- Penaliza más los **errores grandes**, porque los eleva al cuadrado. Errores grandes tienen más peso. Más sensible a outliers (datos atípicos)
- Es útil cuando nos importa mucho que no haya errores grandes.
- También, al igual que el anterior, está en las **mismas unidades de la variable** que estamos prediciendo.

Ejemplo: en agricultura, si predigo biomasa y su RMSE = 50 kg/ha, significa que en promedio el error varía alrededor de **50 kg/ha**.



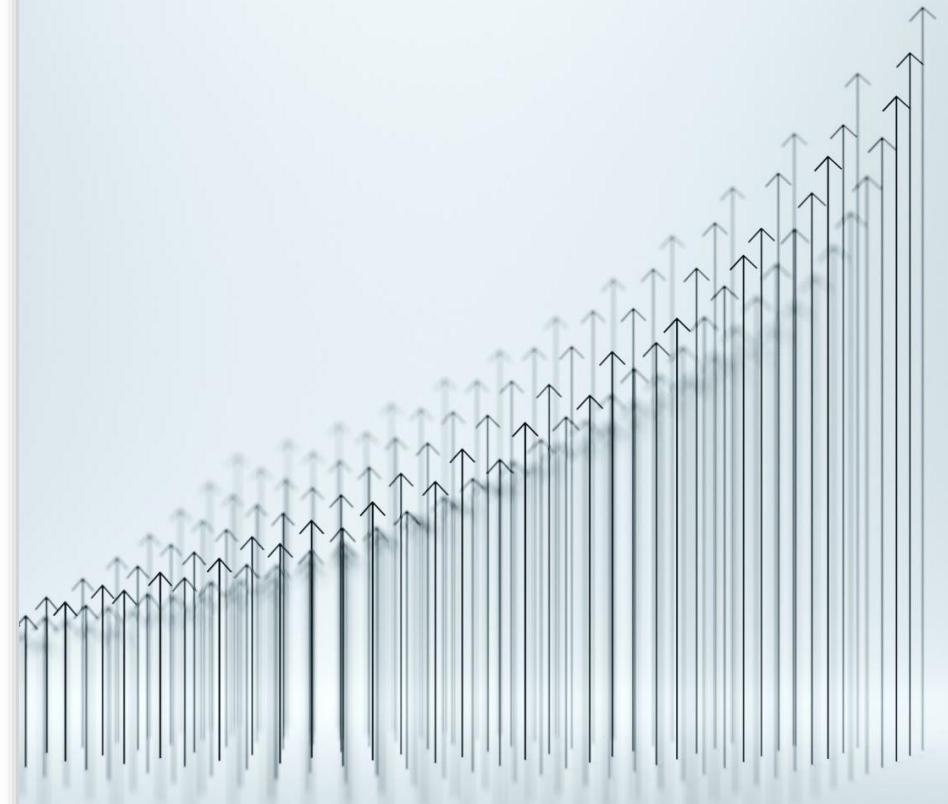
Métricas de evaluación de modelos ML (regresión)

Coeficiente de Determinación - R^2

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

Mide qué proporción de la variabilidad de los datos es explicada por el modelo.

- Varía entre **0 y 1** (aunque puede ser negativa si el modelo es muy malo).
- $R^2 = 1$ el modelo explica toda la variabilidad.
- $R^2 = 0$ el modelo no explica nada (igual que usar el promedio).
- Ejemplo: $R^2 = 0,85 \rightarrow$ el modelo explica el **85% de la variación de los datos.**



Métricas de evaluación de modelos ML (regresión)

- **MAE** → promedio del error (robusto, fácil de interpretar).
- **RMSE** → como el MAE pero indica en máximo cuánto me equivoco, y castiga más los errores grandes.
- **R²** → qué tan bien se ajusta el modelo a los datos.



Precisión

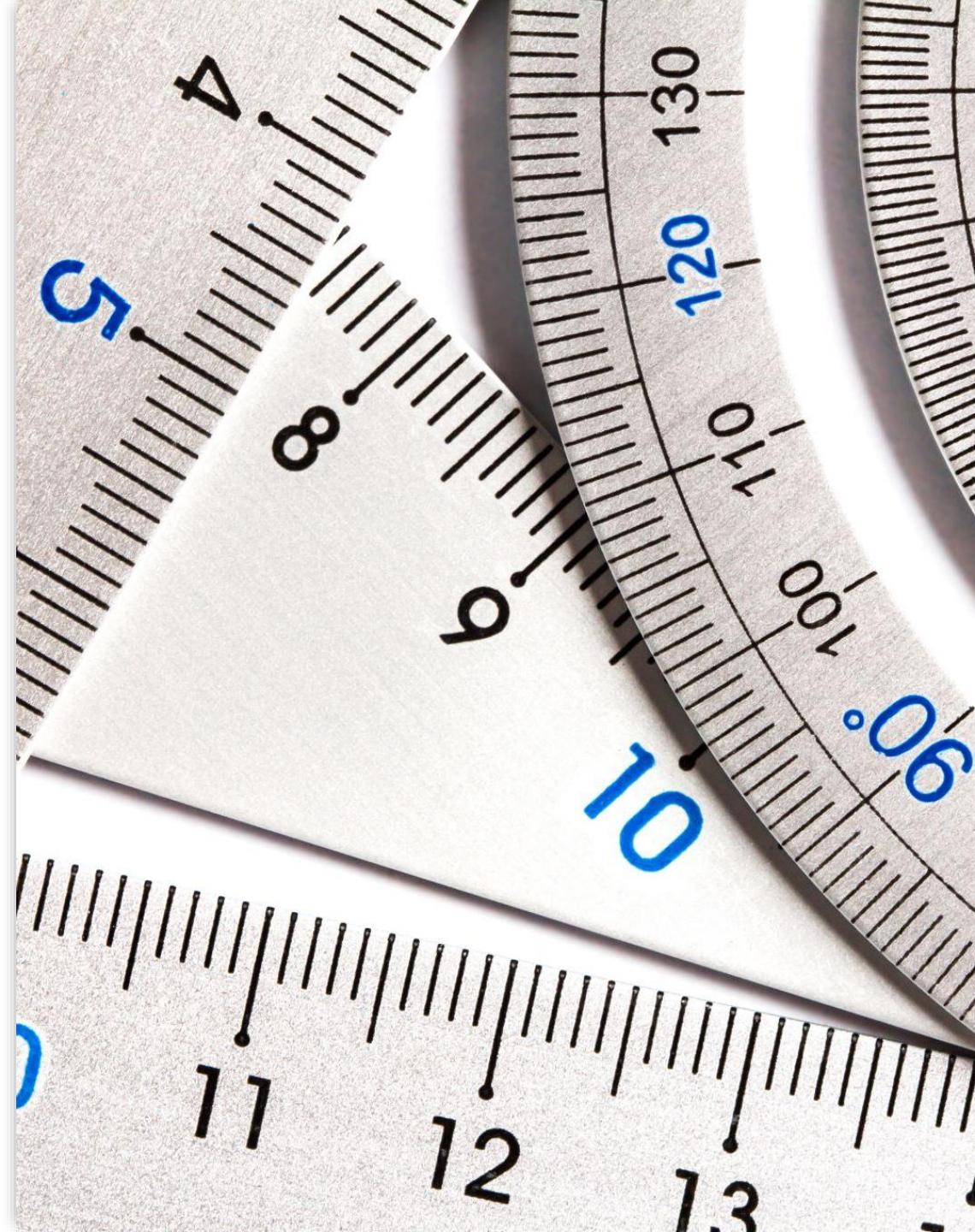
la **precisión (Precision)** es una métrica utilizada principalmente en **clasiﬁcación**, especialmente cuando queremos saber qué tan correctas son las predicciones positivas del modelo.

- Mide la **proporción de predicciones positivas que realmente son correctas**.
- **Precision = De todos los casos que el modelo predijo como “positivos”, ¿cuántos eran realmente positivos?**

Fórmula:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **TP (True Positives):** positivos predichos correctamente
- **FP (False Positives):** positivos predichos incorrectamente



Ejemplo de Precisión

Si el modelo predice que **20 clientes** comprarán paneles solares, pero **solo 15 realmente compran**:

$$\text{Precision} = \frac{15}{20} = 0.75 = 75\%$$

Interpretación:

El 75% de las veces que el modelo dijo “sí compra”, acertó.



Diferencia entre Precision y Accuracy

Métrica	¿Qué mide?	Pregunta que responde	Fórmula	Uso recomendado
Accuracy (Exactitud)	Qué porcentaje total de predicciones fueron correctas	<i>¿Qué tan bien acierta el modelo en general?</i>	$TP + TN / (TP + TN + FP + FN)$	Cuando las clases están balanceadas
Precision (Precisión)	De los que el modelo predijo como positivos, cuántos realmente lo eran	<i>Cuando el modelo dice “sí”, ¿qué tan confiable es?</i>	$TP / (TP + FP)$	Cuando los falsos positivos son costosos

Precisión y Accuracy

**Accuracy =
eficacia global
del modelo**

**Precision =
confianza en los
positivos
predichos**

Sensibilidad (Recall)

La **sensibilidad** (también llamada **Recall** o **TPR – True Positive Rate**) es una métrica de clasificación que mide la capacidad del modelo para **detectar los casos positivos reales**.

De todos los casos que realmente eran positivos, ¿cuántos logró identificar correctamente el modelo?

Sensibilidad (Recall)

Recall=TP/FN+ TP

- **TP (True Positives):** positivos correctamente identificados
- **FN (False Negatives):** positivos que el modelo no detectó (los pasó por alto)

Métricas de evaluación de modelos ML (clasificación)

F1-Score

$$F1 = 2 \frac{\text{Precisión} * \text{Recall}}{\text{Precisión} + \text{Recall}}$$

- Es la **media armónica** entre precisión y recall (**sensibilidad**).
- Útil cuando hay **clases desbalanceadas** (ej: 90% sanos, 10% enfermos).

Interpretación:

- **Precisión** = de los que predije como positivos, ¿cuántos realmente lo son?
- **Recall (sensibilidad)** = de los positivos reales, ¿cuántos detecté?
- El **F1-score** equilibra ambas cosas.
- F1 = 1 es el valor ideal y se da cuando la precisión y recall es 1.



Métricas de evaluación de modelos ML (claseficación)

Ejemplo:

- En un modelo que predice si una planta tiene una enfermedad:
- Alta precisión, pero bajo recall → casi no doy falsos positivos, pero me pierdo muchos enfermos.
- Alta recall, pero baja precisión → detecto casi todos los enfermos, pero doy muchas falsas alarmas.
- F1 mide el **balance** entre ambos.



AUC

- El AUC mide el **área bajo la curva ROC** y representa la **probabilidad de que el modelo clasifique correctamente un caso positivo frente a uno negativo**.
- El **AUC** es una métrica utilizada en problemas de **clasificación**, especialmente binaria, para evaluar qué tan bien un modelo puede **distinguir entre clases** (por ejemplo, “compra” vs “no compra”).
- Se usa junto con la curva **ROC (Receiver Operating Characteristic)**.
- **AUC indica qué tan separables son las clases para el modelo.**



AUC – Ejemplo Energía

Si un modelo para predecir compra de paneles solares tiene un **AUC = 0.85**:

- El 85% de las veces el modelo puede diferenciar correctamente a un cliente comprador de uno no comprador.

ROC

La **ROC (Receiver Operating Characteristic)** es una **curva de evaluación para modelos de clasificación**, especialmente binaria, que muestra el desempeño del modelo comparando su capacidad para detectar positivos frente a los errores que comete al hacerlo.

ROC

Forma de la curva ROC

Muy cerca del vértice superior izquierdo

Línea diagonal (45°)

Debajo de la diagonal

Conclusión

Excelente modelo

Modelo no mejor que el azar

Modelo peor que el azar (clasifica al revés)

ROC

La curva ROC grafica dos métricas a diferentes umbrales de clasificación:

- **Eje Y:** Tasa de Verdaderos Positivos (**TPR o Recall**)

$$TPR = \frac{TP}{TP + FN}$$

- **Eje X:** Tasa de Falsos Positivos (**FPR**)

$$FPR = \frac{FP}{TP + TN}$$

Cada punto en la curva representa un **umbral distinto** usado para decidir si una predicción es positiva o negativa (por ejemplo, 0.2, 0.5, 0.8, etc.).

Métricas de evaluación de modelos ML (clasificación)



AUC – Area Under the Curve (Área bajo la curva ROC)



Se calcula a partir de la **curva ROC** (Receiver Operating Characteristic).



La curva ROC grafica:

Eje Y: Tasa de verdaderos positivos (Recall o Sensibilidad).

Eje X: Tasa de falsos positivos (1 - Especificidad).



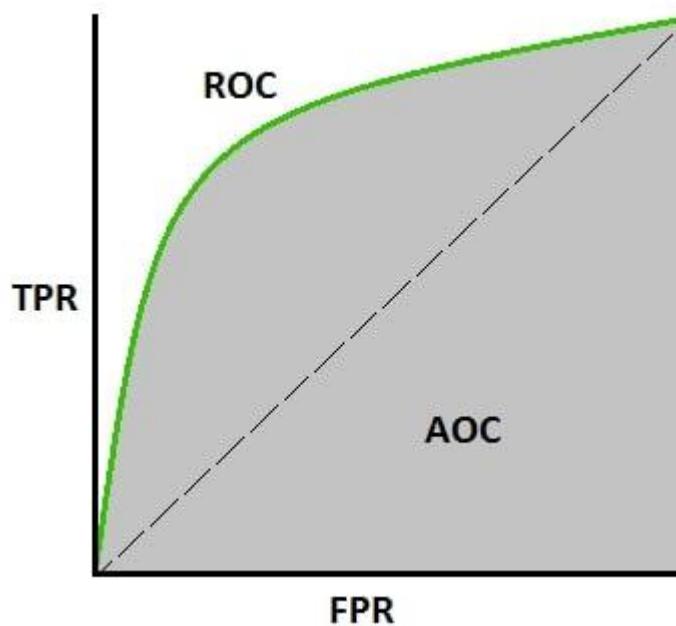
Interpretación:

AUC = 1 → el modelo separa perfectamente las clases.

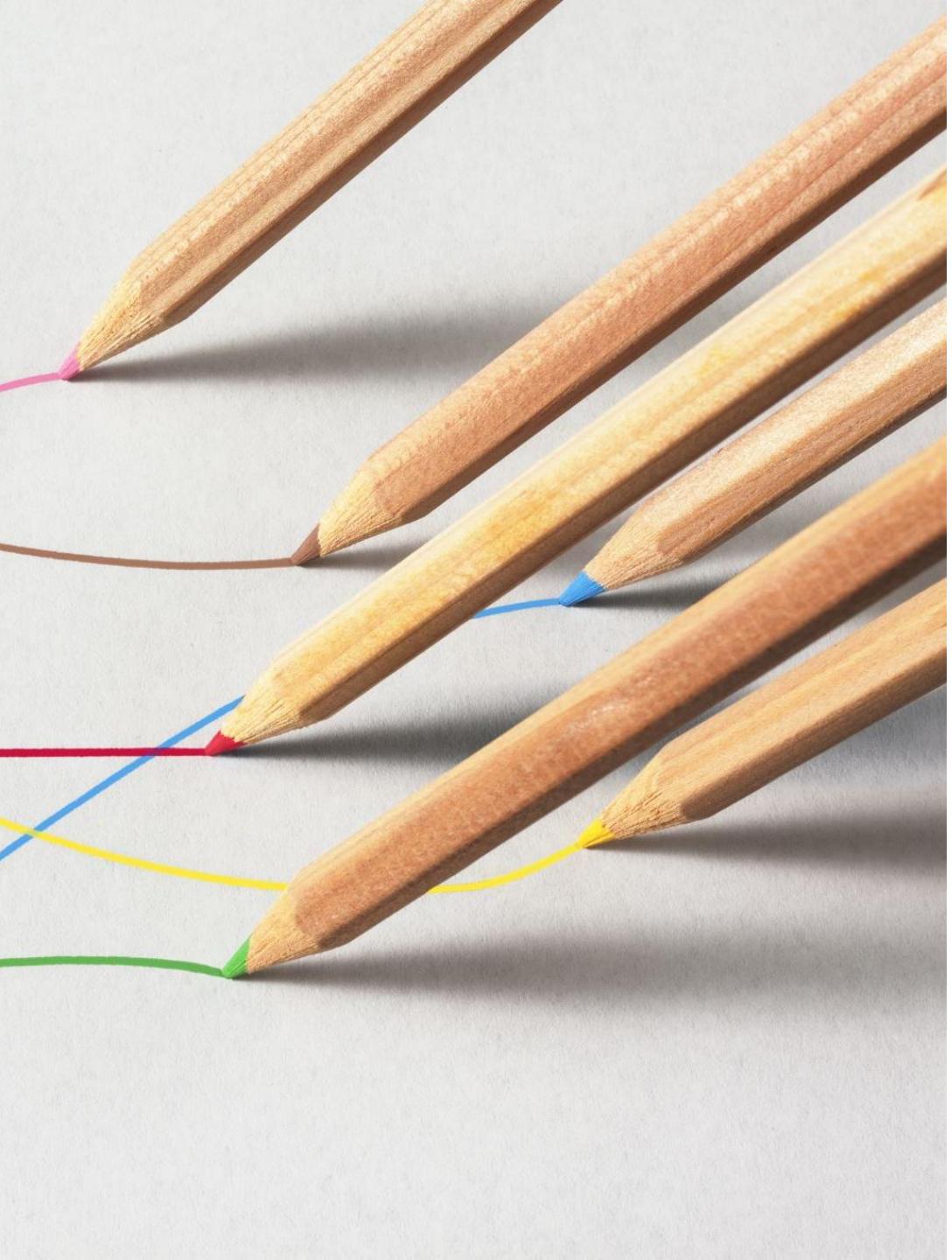
AUC = 0.5 → el modelo no es mejor que adivinar al azar.

AUC < 0.5 → peor que azar (clasifica al revés).

ROC



[Esta foto](#) de Autor desconocido está bajo licencia [CC BY-SA](#)



ROC y AUC

- La curva ROC muestra la capacidad del modelo para separar correctamente las clases en distintos umbrales, y el AUC es el área bajo esa curva.

Métricas de evaluación de modelos ML (claseficación)

Ejemplo en agricultura digital:

- Si quiero clasificar imágenes satelitales en “**pasto sano**” vs “**pasto estresado**”,
 - **AUC = 0.90** significa que hay un 90% de probabilidad de que el modelo dé una puntuación mayor a un píxel realmente “**sano**” que a uno realmente “**estresado**”.



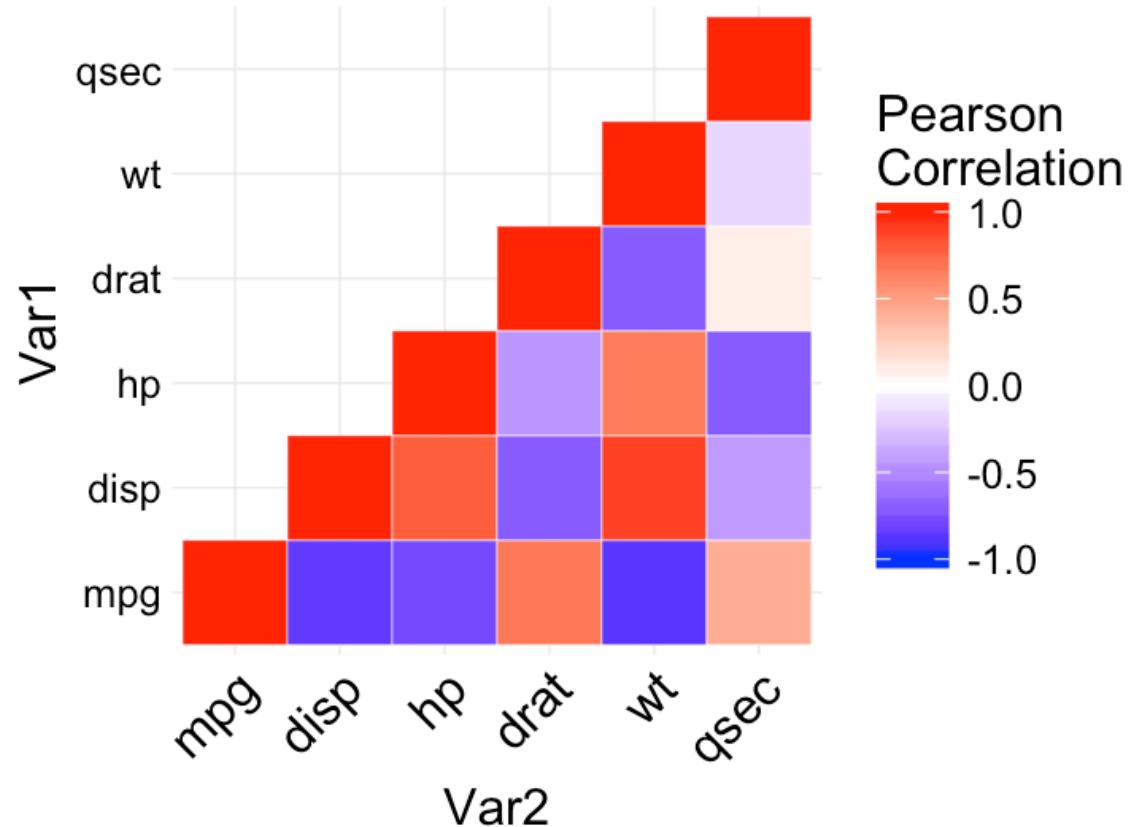
Matriz de Coeficientes de correlación de Pearson

- El **coeficiente de correlación de Pearson** es una medida estadística que cuantifica la **fuerza y dirección de la relación lineal entre dos variables cuantitativas**.

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

- r = coeficiente de correlación de Pearson
- x_i, y_i = valores de cada variable
- \bar{x}, \bar{y} = medias de las variables
- n = número de observaciones

Ejemplo de una matriz de correlación de Pearson



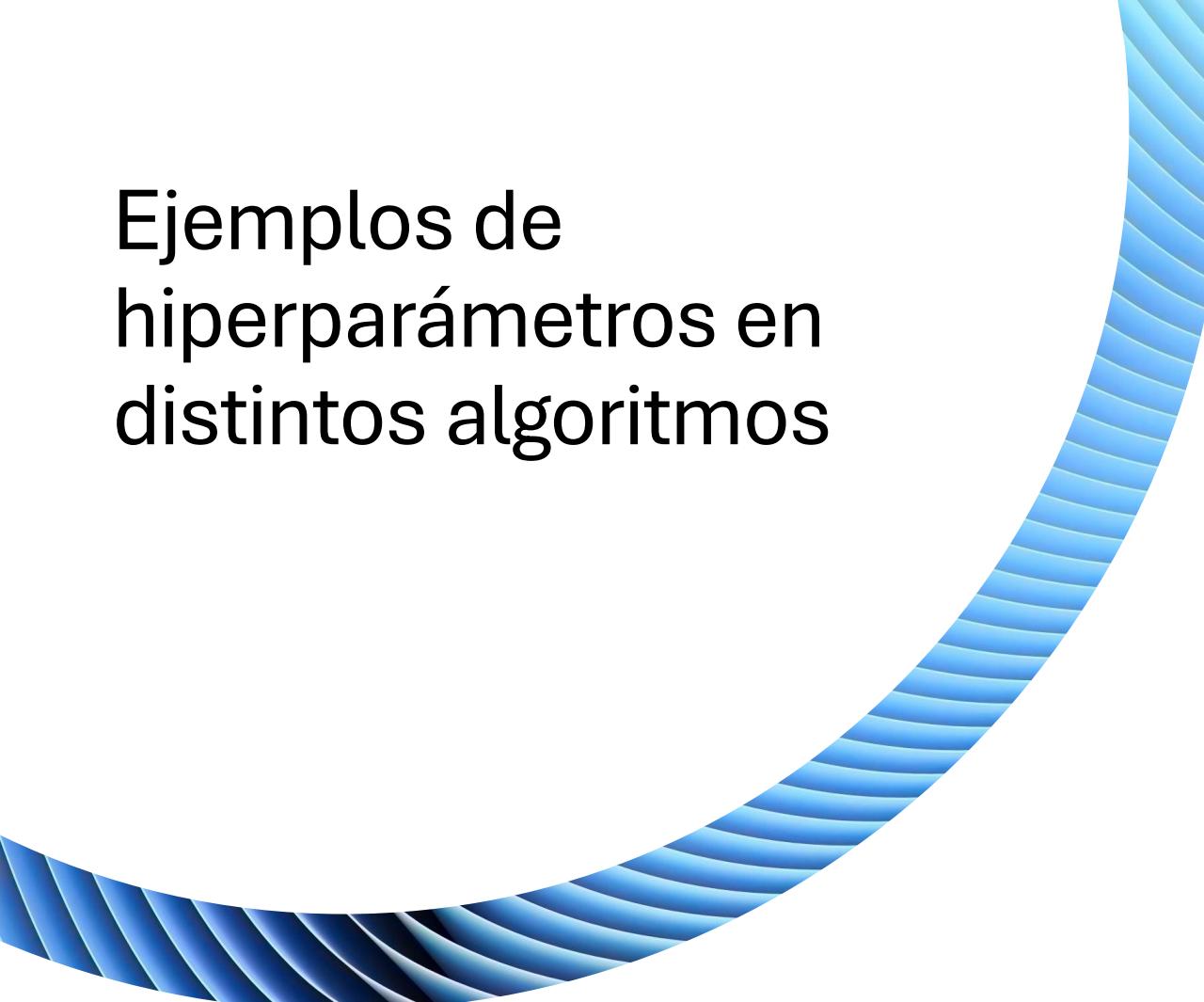
Qué son los hiperparámetros en ML

En **machine learning**, los **hiperparámetros** son parámetros que **no se aprenden automáticamente del entrenamiento**, sino que **tú los defines antes de entrenar el modelo** y controlan el comportamiento del algoritmo de aprendizaje.

Parámetros: valores aprendidos por el modelo durante el entrenamiento (por ejemplo, los coeficientes β en una regresión lineal, o los pesos en una red neuronal).

Hiperparámetros: valores definidos por el usuario para guiar cómo se entrena los parámetros (ejemplo: número de árboles en un Random Forest, tasa de aprendizaje en XGBoost, número de clusters en K-means).

Ejemplos de hiperparámetros en distintos algoritmos



Regresión lineal Elastic Net/ LASSO / Ridge:

Coeficiente de regularización (α o λ) o L1 o L2.

Árboles de decisión / Random Forest:

Profundidad máxima del árbol (max_depth).

Número de árboles (n_estimators).

Número mínimo de muestras por hoja (min_samples_leaf).

XGBoost / Gradient Boosting:

Tasa de aprendizaje (learning_rate).

Número de iteraciones (n_estimators).

Profundidad máxima (max_depth).

Ejemplos de hiperparámetros en distintos algoritmos

K-means:

Número de clusters (k).

Número de iteraciones máximas.

SVM (Support Vector Machine):

Parámetro de penalización (C).

Tipo de kernel (linear, rbf, poly).

Parámetro gamma en kernels no lineales.

Ajuste de hiperparámetros



Para encontrar los valores óptimos se usan técnicas como:



Grid Search: probar combinaciones de hiperparámetros en una cuadrícula.



Random Search: probar combinaciones aleatorias.



Bayesian Optimization / AutoML: búsqueda más inteligente y automática.

Underfitting y overfitting

Underfitting (subajuste)

Ocurre cuando el modelo **es demasiado simple** para capturar la complejidad de los datos.

- El modelo **no aprende lo suficiente** y tiene un **bajo rendimiento tanto en entrenamiento como en validación/prueba**.
- Ejemplo: usar una **regresión lineal simple** para predecir el crecimiento de un cultivo que depende de múltiples variables (clima, suelo, riego, fertilización, etc.). El modelo no logra adaptarse.

Síntomas:

- Error alto en entrenamiento y validación.

Underfitting y overfitting

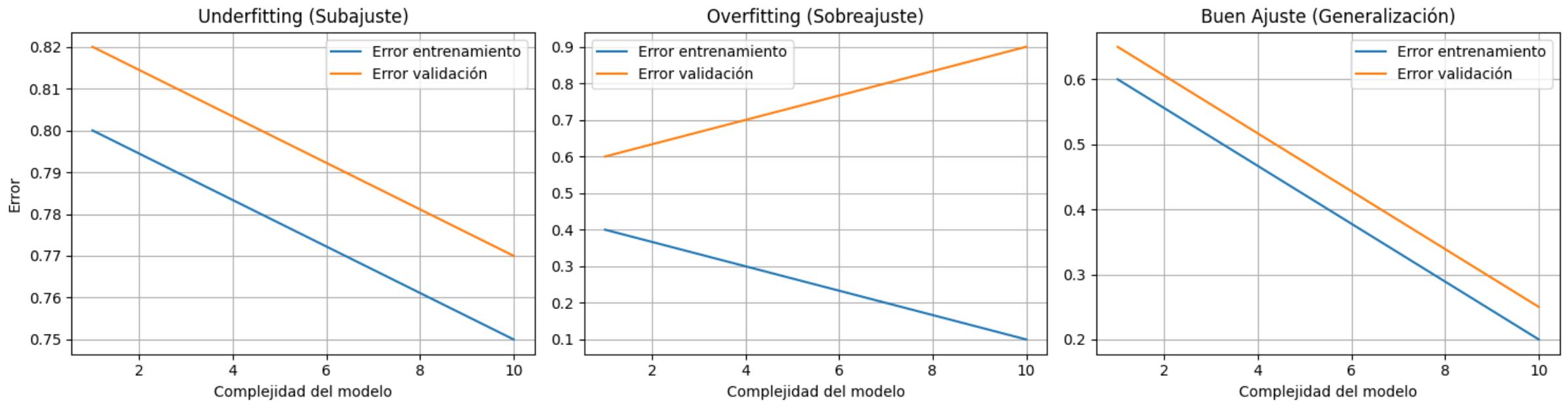
Overfitting (sobreajuste)

- Ocurre cuando el modelo **es demasiado complejo** y se adapta **excesivamente a los datos de entrenamiento**, incluyendo ruido y patrones irrelevantes.
- El modelo **memoriza** en vez de generalizar, por lo que funciona muy bien en entrenamiento, pero **falla con datos nuevos**.
- Ejemplo: un **Random Forest con demasiados árboles profundos** que predice perfectamente el rendimiento de pasto en el conjunto de entrenamiento, pero da valores erráticos con nuevas parcelas.

Síntomas:

- Error muy bajo en entrenamiento, pero alto en validación/prueba.
- Gran diferencia entre desempeño en train y test.

Underfitting y overfitting





Sesión 1. *Fundamentos del Análisis Inteligente de Datos* (4 h)

Detección y limpieza de outliers (método IQR y Tukey).

Preprocesamiento de datos

1. Revisión inicial de los datos

Explorar el dataset: mirar filas, columnas, tipos de variables (numéricas, categóricas, fechas, texto).

Buscar valores nulos o faltantes (NaN).

Detectar valores atípicos (outliers) con boxplots o reglas estadísticas.

Revisar la distribución de cada variable (histogramas, densidad).

Preprocesamiento de datos

2. Limpieza de datos

Manejo de valores faltantes:

Imputación con la media, mediana o moda.

Imputación con modelos (ejemplo: KNNImputer).

Eliminar registros/columnas si hay demasiados nulos.

Tratamiento de outliers:

Recortar valores extremos.

Aplicar transformaciones (log, sqrt).

Usar métodos robustos a outliers (valores atípicos) (ejemplo: RMSE).

Eliminar duplicados de registros.

Preprocesamiento de datos

3. Transformación de variables

Normalización o estandarización

Cuando los algoritmos son sensibles a escalas: K-means, PCA, SVM, redes neuronales

Normalización (0–1): **MinMaxScaler**.

Estandarización (media 0, varianza 1):
StandardScaler.

Codificación de variables categóricas:

One-Hot Encoding (para variables sin orden).

Label Encoding (para variables ordinales).

Ingeniería de características:

Crear nuevas variables derivadas (ejemplo: de una fecha → día, mes, año, estación).

Reducir dimensionalidad con **PCA** o selección de características.

Preprocesamiento de datos

5. Balanceo de clases (solo clasificación)

Si el dataset está desbalanceado (ejemplo: 95% clase A, 5% clase B):

Oversampling: SMOTE (sintético).

Undersampling: reducir la clase mayoritaria.

Class Weights: darle más peso a las clases minoritarias.

Inicio del procesamien to con Machine Learning

4. División del dataset

Separar en **train/test** (70-30, 80-20, etc.).

Usar **validación cruzada (Cross-Validation)** para evaluar de forma robusta qué tanto predecirá el modelo nuevos datos.

Al aplicar machine learning

-
6.
Preparación
final **Guardar los datos transformados**
con joblib o pickle para
reproducibilidad (librería sk-learn
de Python)
-

Documentar los pasos aplicados
(pipeline).

Usar pipeline de sklearn para
automatizar transformaciones y
entrenamiento.

Validación Cruzada

Validación cruzada (Cross-Validation, CV) es una técnica de evaluación de modelos de *Machine Learning* que permite medir su capacidad de generalización, evitando el sobreajuste (*overfitting*). Consiste en dividir los datos en varios subconjuntos para entrenar y validar el modelo varias veces con distintas particiones.

Metodo de validación cruzada más común

El método más común es **k-fold**, que sigue estos pasos:

- Se divide el conjunto de datos en **k** partes (**folds**) de igual tamaño.
- Se realizan **k experimentos**:
 - En cada experimento, se usa **1 fold** como conjunto de prueba (**test**) y los **k-folds restantes** para entrenamiento.
 - Se calculan las métricas en cada iteración.
 - Se promedian los resultados para obtener un rendimiento más estable y confiable.

Otras variantes para hacer validación cruzada

Tipo	Uso principal
k-fold estándar	Datos generales
Stratified k-fold	Clasificación con clases desbalanceadas (mantiene proporciones de clases)
Leave-One-Out (LOOCV)	Conjuntos de datos muy pequeños
Time Series CV (expanding window)	Datos temporales (respeta la secuencia en el tiempo)

Método IQR para eliminar outliers o datos atípicos

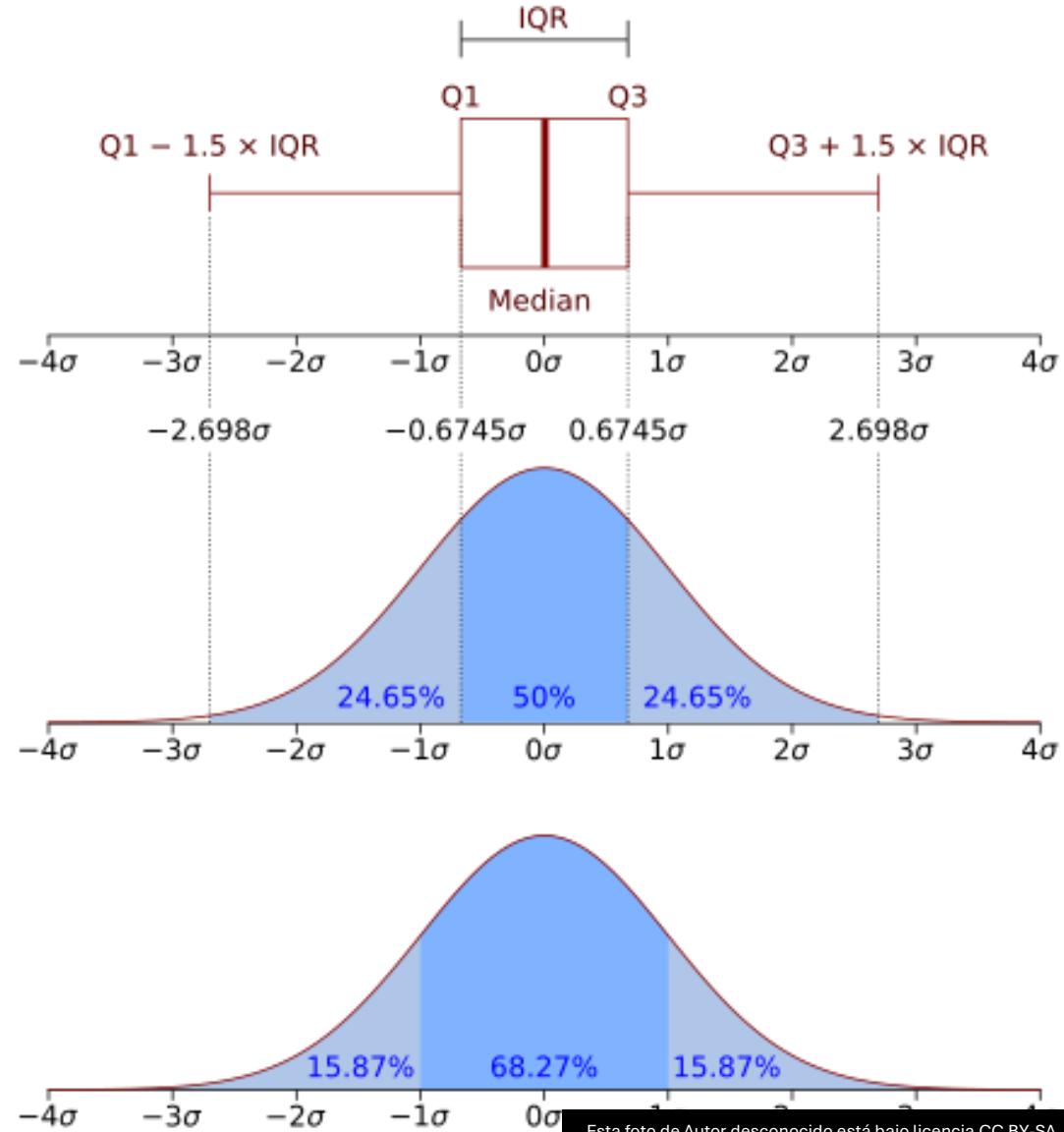
El **método IQR (Interquartile Range)** es una técnica estadística para detectar *outliers* (valores atípicos) basada en la dispersión de los datos entre los cuartiles.

El **IQR** mide el rango intercuartílico, es decir, la diferencia entre el **3er cuartil (Q3)** y el **1er cuartil (Q1)**:

$$\text{IQR} = Q3 - Q1$$

- **Q1 (25%)**: el valor bajo el cual está el 25% de los datos.
- **Q3 (75%)**: el valor bajo el cual está el 75% de los datos.

Método IQR para eliminar outliers o datos atípicos



Método IQR para eliminar outliers o datos atípicos

Regla para detectar outliers

Un dato se considera *outlier* si está **muy por debajo de Q1 o muy por encima de Q3**.

El criterio clásico es:

$$\text{Límite Inferior} = Q1 - 1.5 \cdot IQR$$

$$\text{Límite Superior} = Q3 + 1.5 \cdot IQR$$

Un valor x es outlier si:

$$x < Q1 - 1.5 \cdot IQR \text{ o } x > Q3 + 1.5 \cdot IQR$$

1.5 × IQR → identifica *outliers moderados*

3 × IQR → identifica *outliers extremos*

GRACIAS

