

Análisis exploratorio de los datos

Aprendizaje de Máquina aplicado



Marco Teran
EAFIT

2025

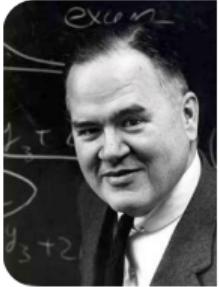
Contenido

- 1 Objetivos
- 2 Introducción a CRISP-DM
- 3 Principales desafíos del Machine Learning
- 4 Análisis Exploratorio de Datos (EDA)
- 5 Ingeniería de Características
- 6 Validación y Pipelines
 - Data leakage
 - Pipelines en *scikit-learn*
- 7 Bases de datos desbalanceadas
- 8 Proyecto integral: California Housing
- 9 Conclusiones y mejores prácticas

Objetivos

Objetivos de aprendizaje

- **Comprender CRISP–DM** como metodología integral e iterativa
- **Dominar el Análisis Exploratorio de Datos (EDA)**: estadísticas descriptivas, visualización y detección de patrones
- **Aplicar ingeniería de características** de forma sistemática y segura
- **Diagnosticar sobreajuste vs. subajuste** mediante análisis de sesgo-varianza
- **Validar modelos rigurosamente**: *holdout*, validación cruzada y detección de *data leakage*
- **Diseñar pipelines** reproducibles y preparados para producción



“Es mejor una respuesta aproximada a la pregunta correcta que una respuesta exacta a la pregunta equivocada.”

— John Tukey

Introducción a CRISP-DM

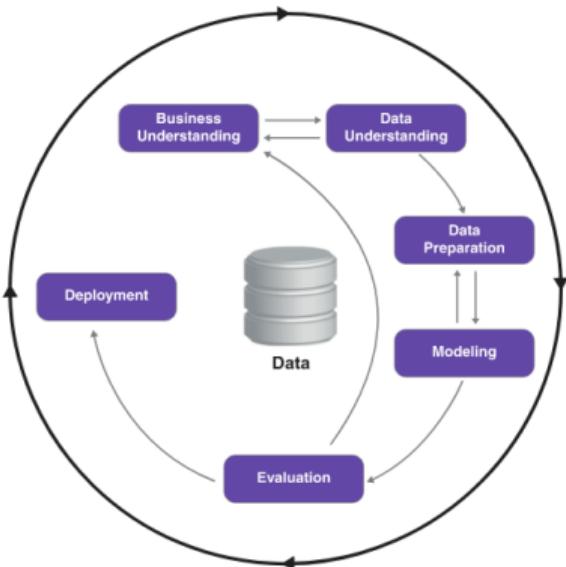
¿Por qué necesitamos una metodología?

- **Complejidad inherente:** Los proyectos de ML involucran múltiples disciplinas
- **Alto índice de fracaso:** 87% de proyectos de ML nunca llegan a producción
- **Naturaleza iterativa:** El descubrimiento en datos no es lineal
- **Comunicación:** Marco común entre científicos de datos y stakeholders

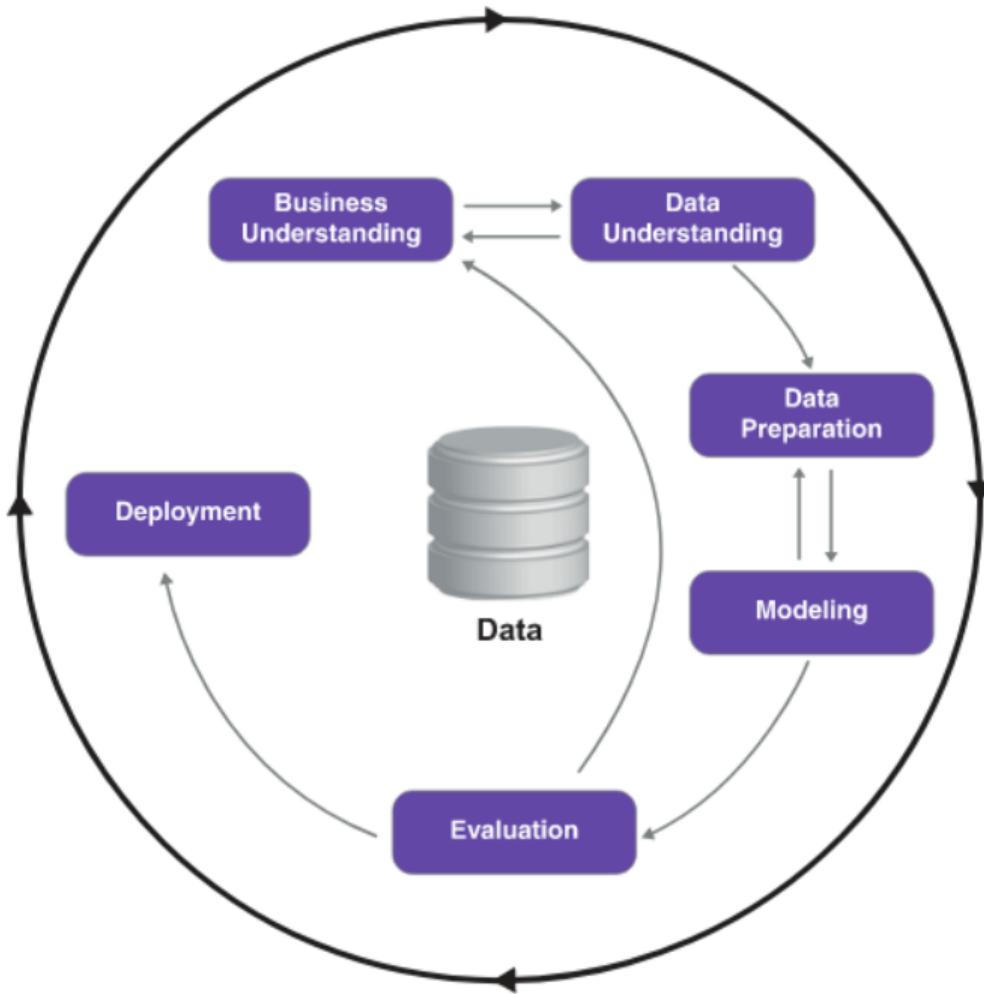
Sin metodología

Riesgo de resolver el problema equivocado, perfectamente

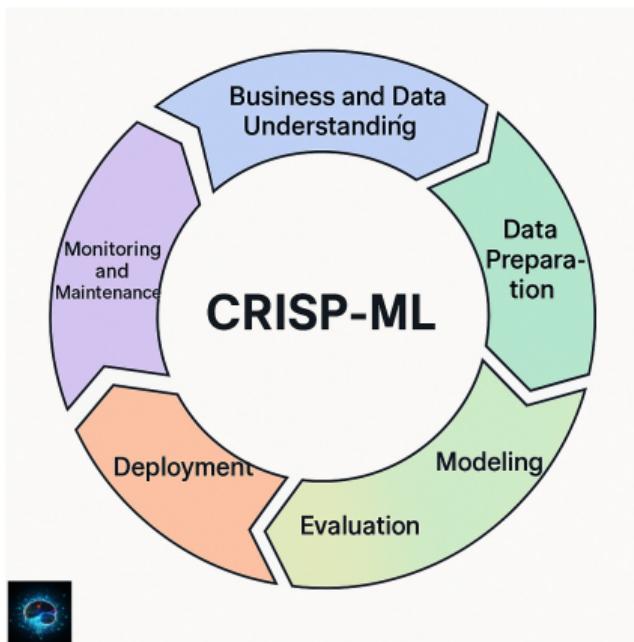
CRISP-DM



- **Cross Industry Standard Process for Data Mining:** metodología estándar creada en 1996–1999 para estructurar proyectos de **minería de datos**.
- Creada en 1996–1999 y sigue siendo la guía más usada en ciencia de datos.
- Desarrollado por un consorcio europeo liderado por Daimler-Benz, SPSS y NCR.
- **Objetivo principal:** proporcionar un marco claro y repetible para convertir datos en conocimiento aplicable al negocio.



Las 6 fases de CRISP-DM



- 1 Comprensión del Negocio (15-20 % del tiempo)**
 - Definir objetivos y KPIs medibles
 - Evaluar viabilidad y ROI esperado
- 2 Comprensión de los Datos (20-30 %)**
 - Recolección y auditoría de calidad
 - Exploración inicial y detección de problemas
- 3 Preparación de Datos (40-50 %)**
 - Limpieza, transformación e ingeniería de características
- 4 Modelado (10-20 %)**
 - Selección y entrenamiento de algoritmos
- 5 Evaluación (5-10 %)**
 - Validación contra objetivos de negocio
- 6 Despliegue (5-10 % + mantenimiento continuo)**

Caso de estudio: Sistema de detección de spam

Contexto del problema:

- 100 millones de emails/día
- 40% son spam
- Costo por falso positivo: \$50
- Costo por falso negativo: \$0.10

Decisión clave:

- Optimizar para alta precisión
- Tolerar algo de spam vs. perder emails legítimos



Métrica de negocio

$$\text{Minimizar costo total} = \$50 \times \text{FP} + \$0.10 \times \text{FN}$$



Fase 1: Comprensión del negocio - Profundización

Preguntas fundamentales:

- ¿Qué decisión específica mejorará el modelo?
- ¿Quién es el usuario final y cuál es su tolerancia al error?
- ¿Cuál es el proceso actual y sus limitaciones?
- ¿Qué constituye éxito? (métricas cuantificables)

Análisis de viabilidad:

- ¿Tenemos datos suficientes y de calidad?
- ¿El problema requiere ML o bastan reglas simples?
- ¿Existe capacidad técnica para mantener la solución?

Error común

Comenzar con la técnica (“Usemos deep learning”) en lugar del problema



Fase 2: Comprensión de datos - El arte del EDA

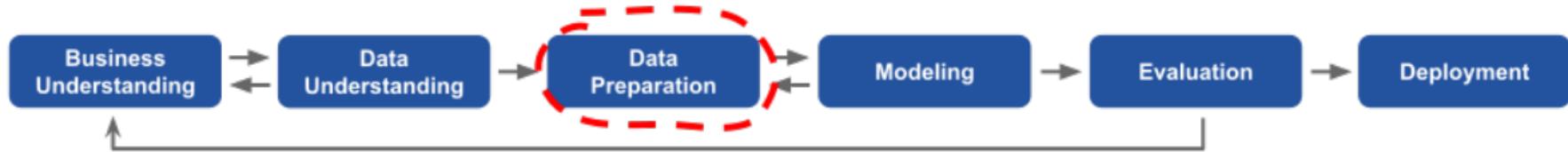
“Los datos crudos son como el petróleo crudo: valioso pero inutilizable sin refinamiento” – Michael Palmer

Análisis sistemático:

- **Volumen:** ¿Suficiente para aprendizaje estadístico?
- **Velocidad:** ¿Batch o streaming?
- **Variedad:** Estructurado, texto, imágenes
- **Veracidad:** Calidad, ruido, valores faltantes
- **Valor:** ¿Los datos contienen señal predictiva?

Principio de Tukey

Explorar sin hipótesis preconcebidas, dejar que los datos “hablen”



Fase 3: Preparación de datos - La fase crítica

¿Por qué consume 40-50 % del tiempo?

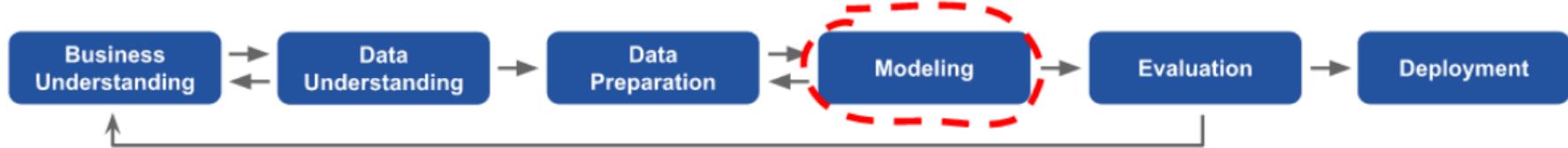
- Datos del mundo real son desordenados
- Cada decisión impacta el rendimiento final
- Requiere conocimiento del dominio + técnico

Tareas principales:

- 1 **Limpieza:** Outliers, duplicados, inconsistencias
- 2 **Integración:** Múltiples fuentes, formatos
- 3 **Transformación:** Normalización, encoding
- 4 **Reducción:** Selección de características
- 5 **Creación:** Ingeniería de nuevas variables

Regla de oro

Garbage In, Garbage Out (GIGO)



Fase 4: Modelado - Más allá de los algoritmos

No es solo elegir un algoritmo:

- Selección basada en naturaleza del problema
- *Trade-off* interpretabilidad vs. precisión
- Consideraciones computacionales
- Mantenibilidad en producción

Proceso iterativo:

- 1 Comenzar con baseline simple
- 2 Incrementar complejidad gradualmente
- 3 Validar cada mejora rigurosamente
- 4 Documentar decisiones y resultados

Trampa común

Enfocarse en accuracy ignorando interpretabilidad y costos computacionales



Fase 5: Evaluación - Validación holística

Más allá de las métricas técnicas:

- ¿Cumple los objetivos de negocio?
- ¿Es robusto ante cambios esperados?
- ¿Los stakeholders confían en él?
- ¿El costo-beneficio es positivo?

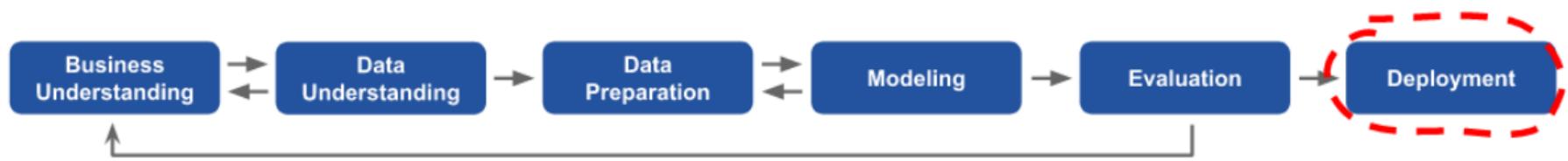
Evaluación multidimensional:

Técnica

- Precisión, recall, F1
- Validación cruzada
- Análisis de errores

Negocio

- ROI proyectado
- Tiempo de respuesta
- Escalabilidad



Fase 6: Despliegue - La última milla

Despliegue ≠ Fin del proyecto:

- Monitoreo continuo de performance
- Detección de *model drift*
- Actualización y reentrenamiento
- Gestión de versiones

Estrategias de despliegue:

- **Shadow mode:** Ejecutar en paralelo sin impacto
- **A/B testing:** Comparar con sistema actual
- **Gradual rollout:** Incrementar tráfico progresivamente
- **Blue-green:** Cambio instantáneo con rollback rápido

Fase 6: Despliegue - La última milla

Éxito en producción

Un modelo mediocre en producción vale más que uno perfecto en desarrollo

Principales desafíos del Machine Learning

Taxonomía de problemas en ML

Problemas de datos:

- Cantidad insuficiente
- Calidad pobre
- No representativos
- Valores faltantes
- Desbalance de clases

Problemas de modelo:

- Sobreajuste (*overfitting*)
- Subajuste (*underfitting*)
- Selección incorrecta
- Hiperparámetros subóptimos
- *Data leakage*

Principio fundamental

La calidad del modelo está limitada por la calidad de los datos

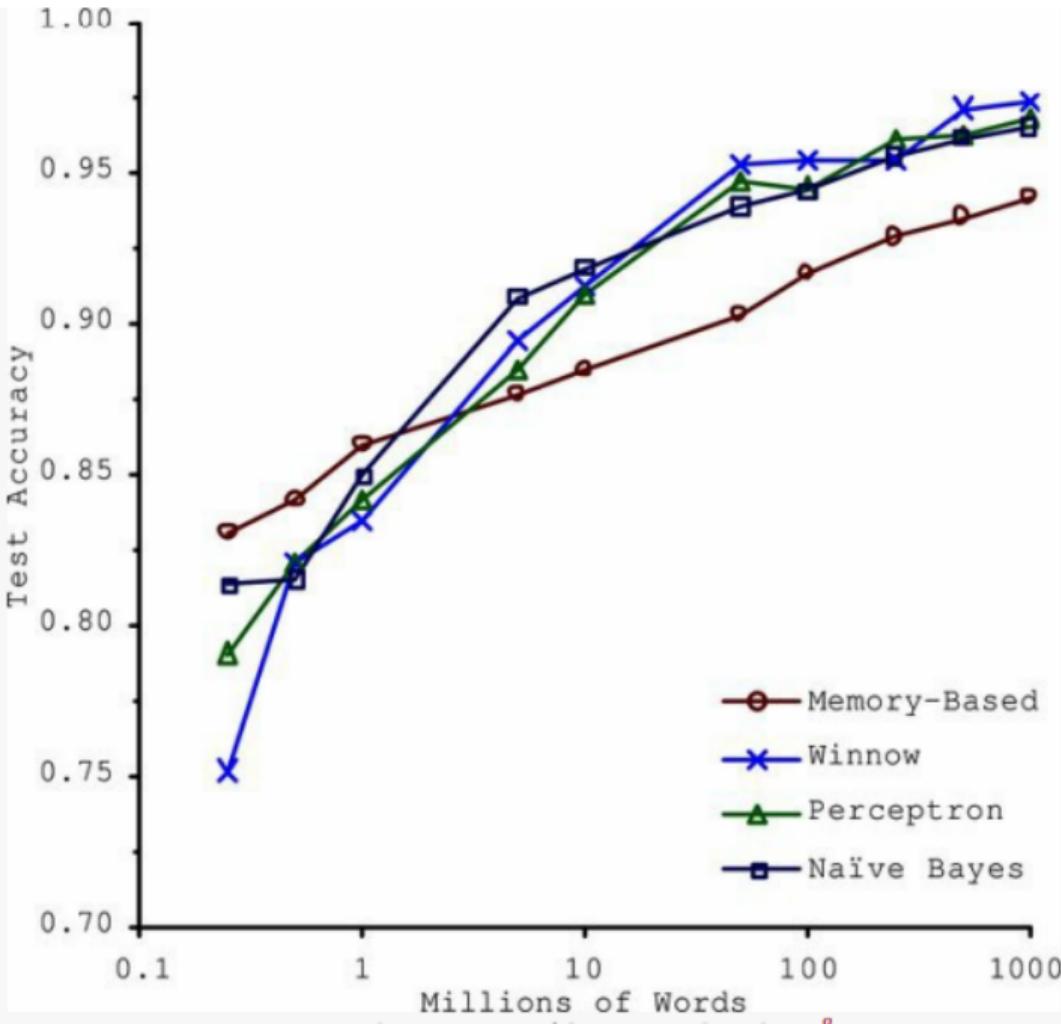
Cantidad insuficiente de datos

¿Cuántos datos necesitamos?

- Regla heurística: $10 \times$ más ejemplos que parámetros
- Para deep learning: Miles a millones según complejidad
- Depende de: ruido, dimensionalidad, complejidad del patrón

Estrategias cuando hay pocos datos:

- **Transfer learning:** Usar modelos pre-entrenados
- **Data augmentation:** Generar variaciones sintéticas
- **Regularización fuerte:** Limitar complejidad del modelo
- **Modelos simples:** Preferir interpretabilidad



Datos no representativos - Sesgo de muestreo

Tipos de sesgo:

- **Sesgo de selección:** Muestra no aleatoria
- **Sesgo de supervivencia:** Solo vemos casos exitosos
- **Sesgo temporal:** Datos antiguos para problema actual
- **Sesgo geográfico/demográfico:** Subrepresentación

Datos no representativos - Sesgo de muestreo

Ejemplo histórico:

- Encuesta presidencial 1936 (Literary Digest)
- 2.4 millones de respuestas
- Predijo victoria de Landon
- Roosevelt ganó por amplio margen
- Problema: Sesgo en método de muestreo (teléfonos/autos)

Consecuencia

Modelo con 99% accuracy en test puede fallar completamente en producción

Calidad de datos - Problemas comunes

Tipos de problemas:

- **Ruido:** Errores de medición
- **Outliers:** Valores extremos legítimos o errores
- **Inconsistencias:** Mismo concepto, diferentes formatos
- **Duplicados:** Registros repetidos
- **Valores faltantes:** Patrones MCAR, MAR, MNAR

Impacto:

- ↓ Precisión
- ↑ Varianza
- Sesgo sistemático
- Inestabilidad

Estrategia

Invertir tiempo en limpieza y validación de datos siempre paga dividendos

El trade-off sesgo-varianza

Descomposición del error:

$$\text{Error esperado} = \text{Sesgo}^2 + \text{Varianza} + \text{Ruido irreducible}$$

Alto sesgo (underfitting):

- Modelo demasiado simple
- No captura patrones
- Error alto en train y test

Alta varianza (overfitting):

- Modelo demasiado complejo
- Memoriza ruido
- Error bajo en train, alto en test

Objetivo

Encontrar el punto óptimo que minimice el error total

Características irrelevantes: formulación y efecto

■ **Características (features)**: atributos medibles que describen cada instancia.

Sea el conjunto de datos con n muestras y m características (features).

Definimos:

$$X \in \mathbb{R}^{n \times m} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix}, \quad y \in \mathbb{R}^n = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \beta \in \mathbb{R}^m = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{bmatrix}.$$

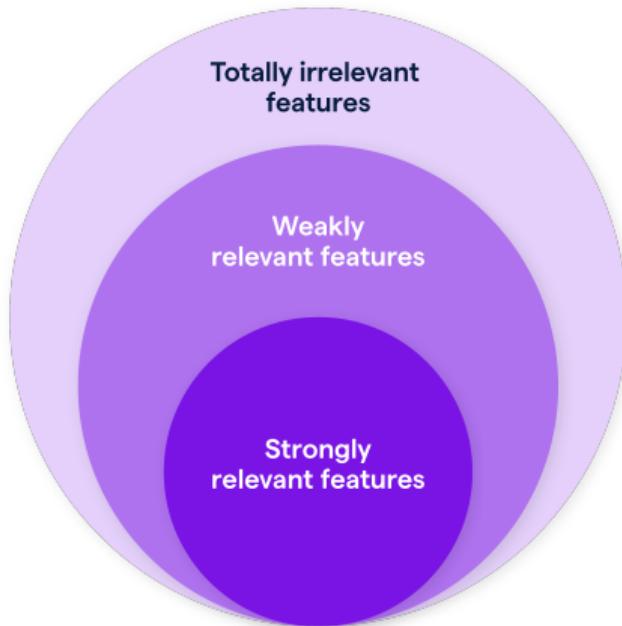
El modelo lineal clásico queda:

$$y = X\beta + \varepsilon, \quad \hat{y} = X\hat{\beta}, \quad \hat{\beta}_{\text{OLS}} = (X^\top X)^{-1} X^\top y.$$

Si particionamos las columnas de X en relevantes e irrelevantres,

$$X = [X_{\text{rel}} \ X_{\text{irr}}], \quad X^\top X = \begin{bmatrix} X_{\text{rel}}^\top X_{\text{rel}} & X_{\text{rel}}^\top X_{\text{irr}} \\ X_{\text{irr}}^\top X_{\text{rel}} & X_{\text{irr}}^\top X_{\text{irr}} \end{bmatrix},$$

Características irrelevantes

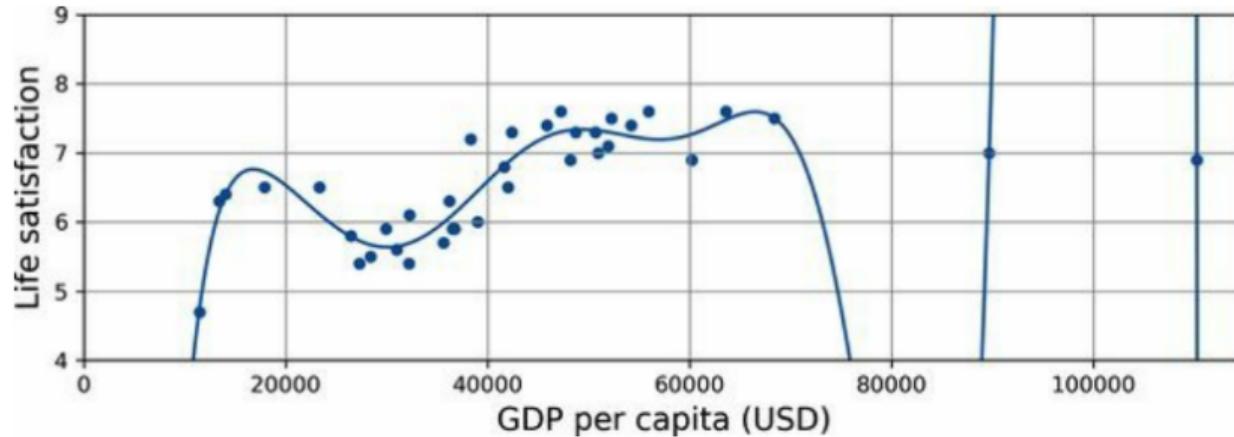


- El éxito depende de buenas representaciones.
- **Ingeniería:** selección, extracción y reducción.
- Cuidado con alta cardinalidad y multicolinealidad.

Sobreajuste (overfitting) y regularización

- Ajuste perfecto al ruido no generaliza.
- Modelos complejos requieren control de capacidad.

$$\underbrace{\mathbb{E}(y - \hat{f})^2}_{\text{riesgo}} = \sigma^2 + \text{bias}^2 + \text{varianza}$$



Teorema clave

Descomposición sesgo-varianza: guía el *trade-off* óptimo.

Cómo reducir el sobreajuste

- Simplificar modelo o usar regularización.
- Conseguir más y mejores datos.
- Reducir ruido y estabilizar etiquetas.

Regla robusta

Valide cada decisión de limpieza con CV estratificada.

Técnicas de regularización

¿Qué es regularización?

- Introduce penalizaciones en la función de costo para reducir complejidad y prevenir sobreajuste.
- Trade-off: simplicidad vs. ajuste a datos

$$\min_w \|y - Xw\|^2 + \lambda \Omega(w)$$

Tipos principales:

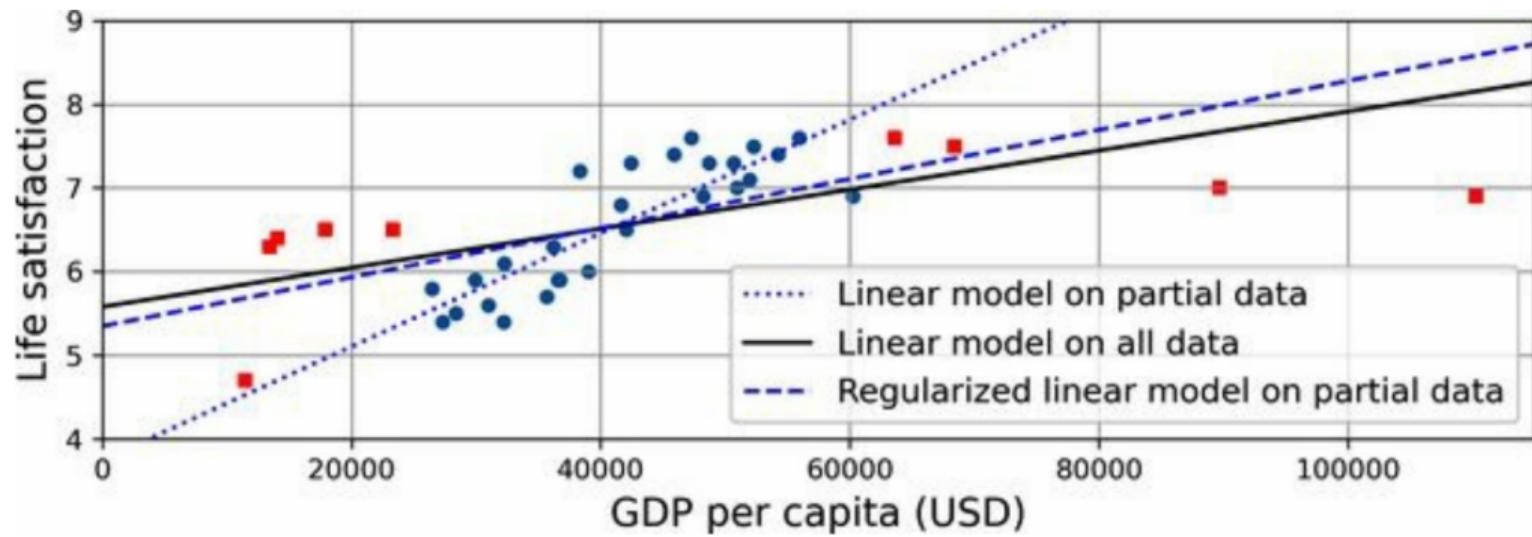
- **L1 (Lasso):** $\Omega(w) = \sum |w_i| \rightarrow$ fuerza a muchos $w_i = 0 \Rightarrow$ selección automática de características.
- **L2 (Ridge):** $\Omega(w) = \sum w_i^2 \rightarrow$ reduce magnitud de los coeficientes \Rightarrow estabiliza el modelo ante multicolinealidad.
- **Elastic Net:** mezcla L1+L2, útil con muchas variables correlacionadas.

Técnicas de regularización

Tipos principales:

- **Dropout**: Desactivar neuronas aleatoriamente
- **Early stopping**: Detener antes de sobreajustar
- **Data augmentation**: Más datos sintéticos

Técnicas de regularización



Principio

“Simpler is better” - Navaja de Occam en ML

Resumen: Desafíos y soluciones

Desafío	Solución principal
Pocos datos	Transfer learning, regularización
Datos no representativos	Muestreo estratificado, más diversidad
Baja calidad	Limpieza sistemática, validación
Características irrelevantes	Selección de características, PCA
Overfitting	Regularización, más datos, CV
Underfitting	Más características, modelo complejo
Data leakage	Pipelines correctos, validación temporal

Recordar

No hay solución universal - cada problema requiere diagnóstico específico

Análisis Exploratorio de Datos (EDA)

¿Qué es EDA? Filosofía y práctica

Definición: *Exploratory Data Analysis*

- Metodología para investigar datos y descubrir su estructura.
- Sin hipótesis preconcebidas
- Combinación de estadísticas descriptivas y visualización

Objetivos:

- Descubrir patrones: comprender estructura, calidad y señales útiles.
- Detectar anomalías
- Formular hipótesis
- Guiar decisiones de rasgos y validar supuestos.
- **Más un arte que una ciencia.**

¿Qué es EDA? Filosofía y práctica

John Tukey (1977):

"EDA es una actitud, una flexibilidad, y una confianza en el display gráfico"

Principio fundamental

Explorar primero, modelar después

Pipeline sistemático de EDA

1 Estructura de datos

- Dimensiones, tipos, memoria
- Generales: `df.head()`, `df.info()`, `df.shape`

2 Estadísticas univariadas numéricas (`df.describe()`)

- Centralidad: media, mediana, moda
- Dispersion: varianza, IQR, rango, MAD.
- Forma: asimetría, curtosis

3 Estadísticas univariadas categóricas: moda, frecuencias, % únicos.

4 Análisis y auditoría de valores faltantes

- Patrones: MCAR, MAR, MNAR
- Estrategias de imputación

5 Detección de outliers sistemáticos

- Métodos: IQR, Z-score, Isolation Forest

6 Análisis bivariado/multivariado

- Dependencias: correlación, Spearman e información mutua.

7 Estabilidad temporal y *drift* de covariables.

8 Balance de clases y separación de etiquetas.

Estadísticas robustas vs. clásicas

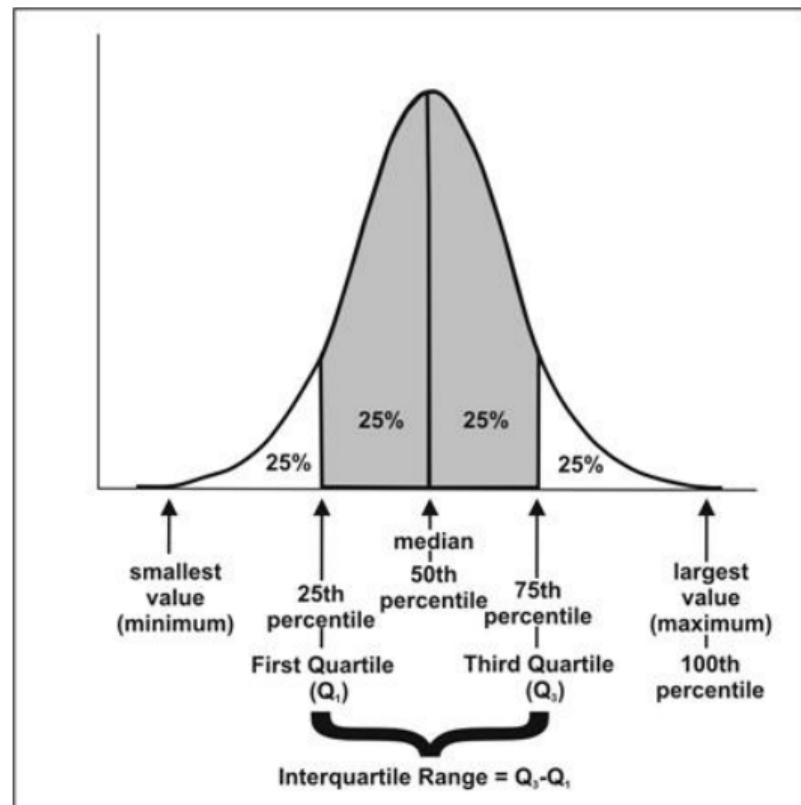
Estadísticas clásicas:

- Media: $\bar{x} = \frac{1}{n} \sum x_i$
- Varianza: $s^2 = \frac{1}{n-1} \sum (x_i - \bar{x})^2$

- Sensibles a outliers
- Asumen normalidad

Estadísticas robustas:

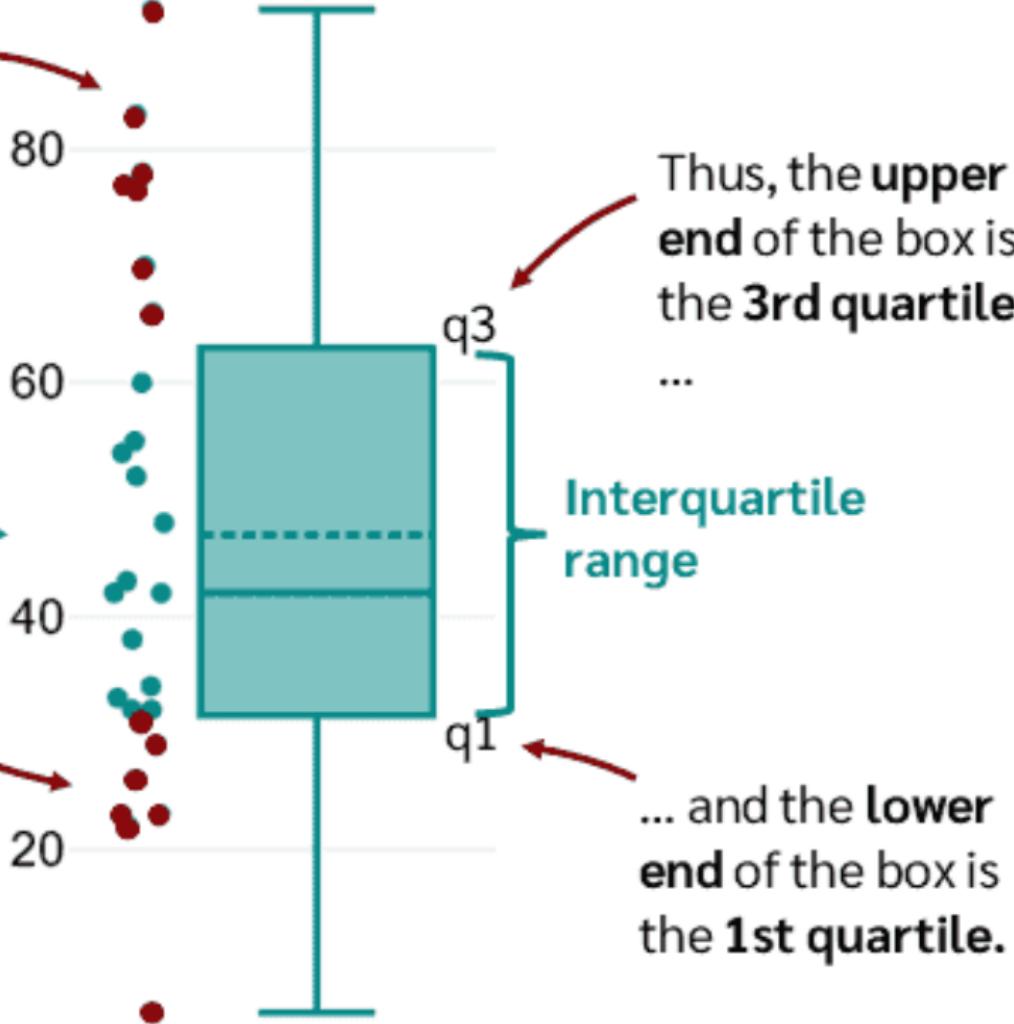
- Mediana: percentil 50
- IQR: $Q_3 - Q_1$
- MAD: mediana($|x_i - \text{mediana}|$)
- Resistentes a outliers



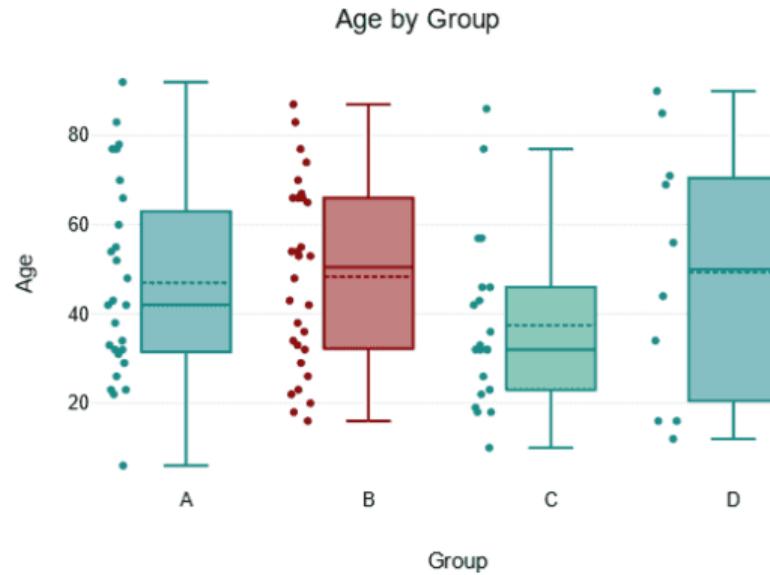
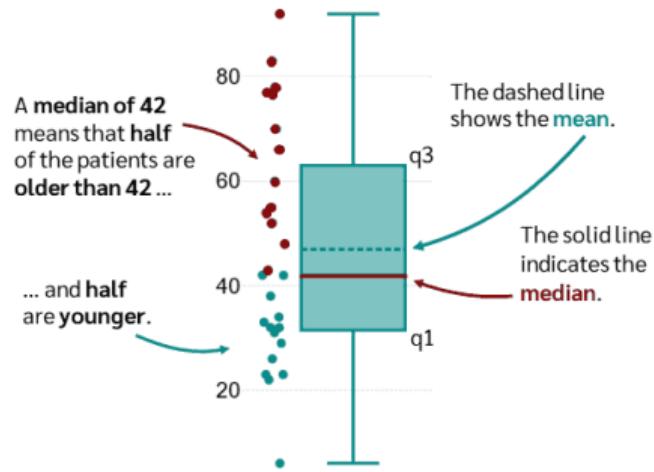
25% of the data
are above q3,

in the box we
find 50% of our
data

and 25% of
the data are
below q1.



Estadísticas robustas vs. clásicas

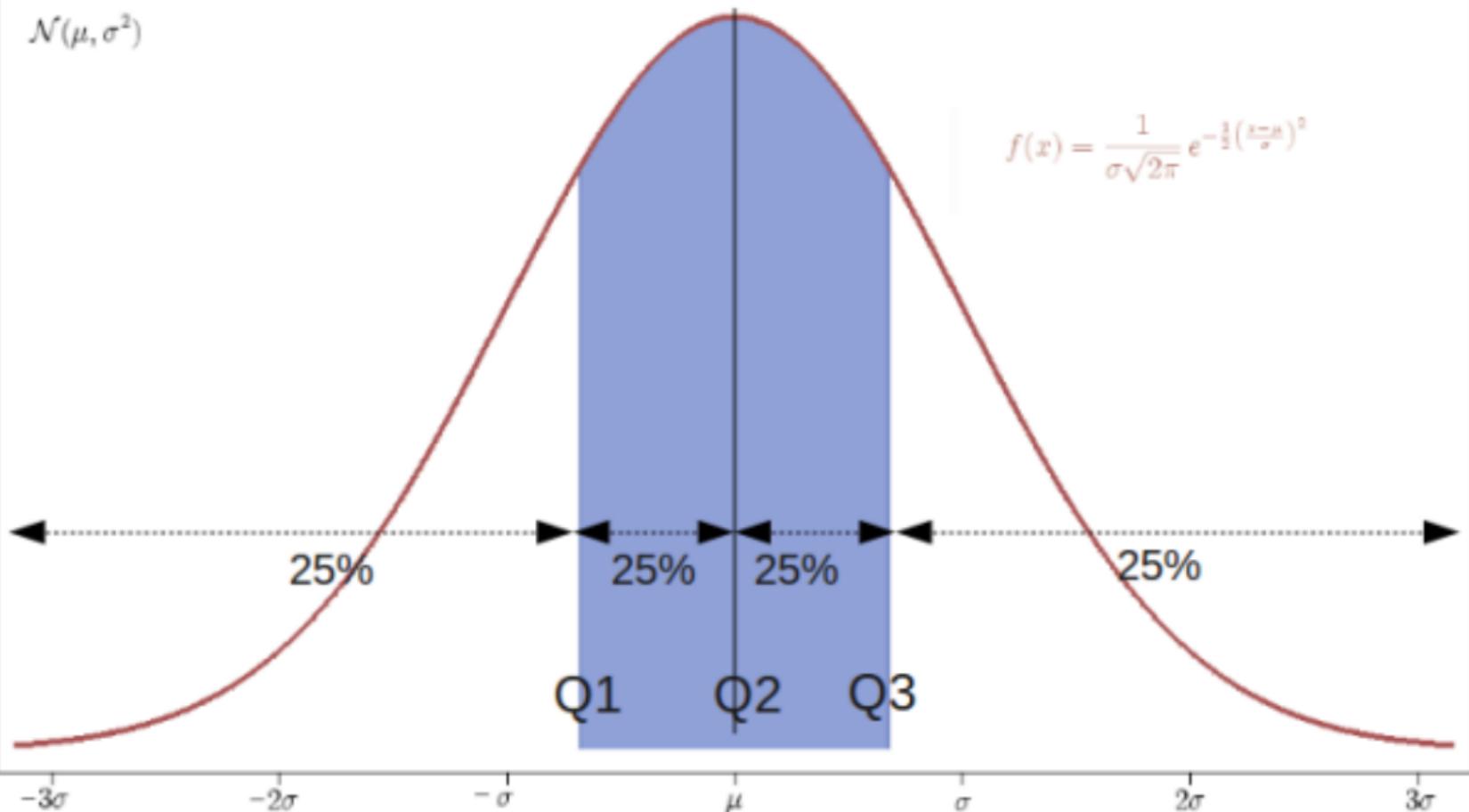


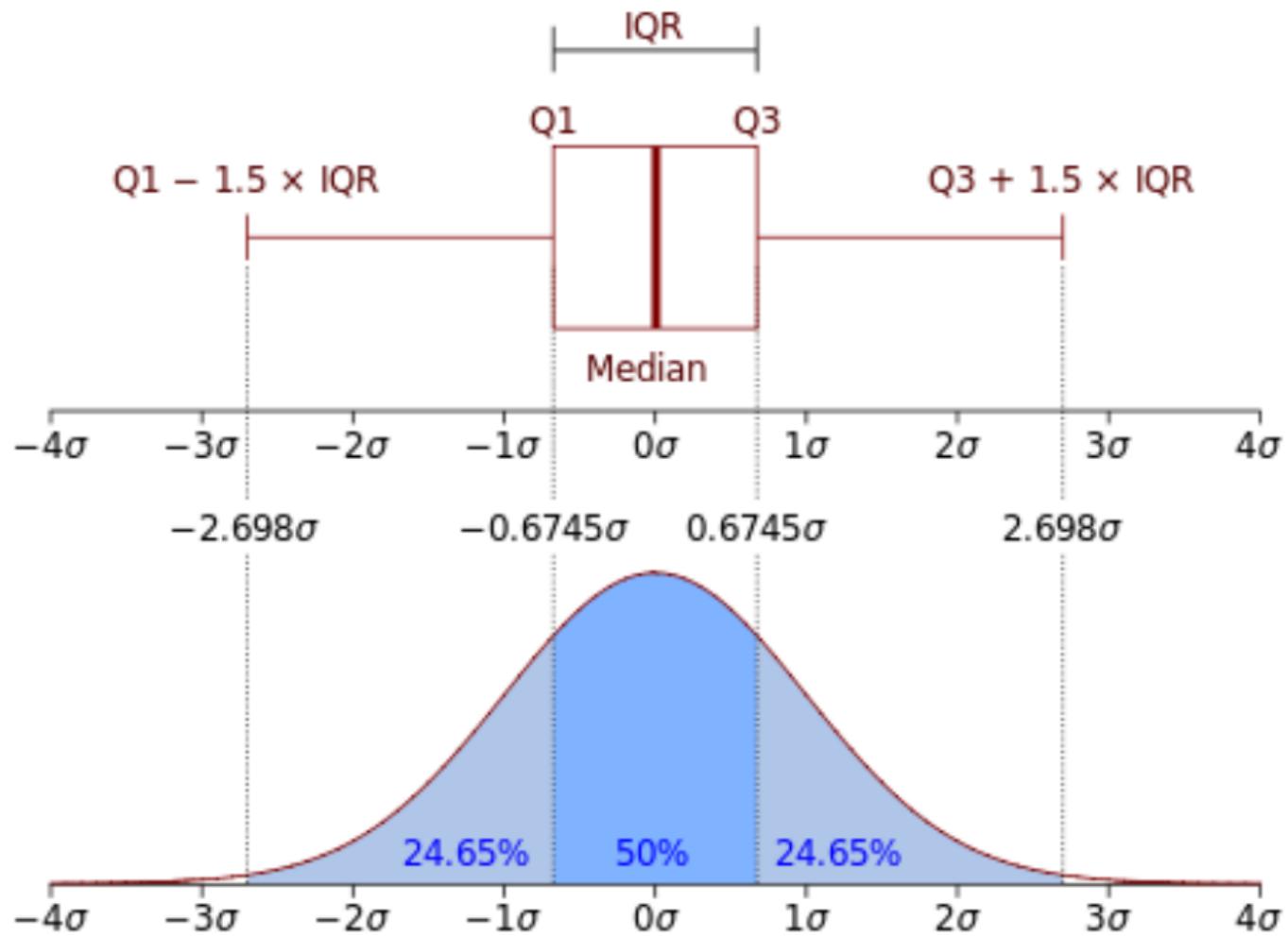
Punto de ruptura (*breakdown point*):

- Media: $\frac{1}{n} \rightarrow 0$ cuando $n \rightarrow \infty$
- Mediana: 0.5 (máximo posible)

$\mathcal{N}(\mu, \sigma^2)$

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$





Localización y dispersión robustas

- Media y varianza: sensibles a outliers.
- Mediana e IQR/MAD: opciones robustas por defecto.

$$IQR = Q_3 - Q_1, \quad \text{Outlier : } x \notin [Q_1 - 1.5 IQR, Q_3 + 1.5 IQR]$$

Regla práctica

Reporte siempre media \pm DE y mediana/IQR.

Estadística de la variable objetivo

- `df[target].value_counts()` para distribución de clases.
- `np.bincount(y)` alternativa para y numérico.
- Verificar desbalance y rareza de clases.

Código: numéricas y categóricas

```
# Característica numérica
import matplotlib.pyplot as plt
df[num_feature].plot.hist(bins=7)
plt.show()

# Característica categórica
import matplotlib.pyplot as plt

df[cat_feature].value_counts().plot.bar()
plt.show()
```

Análisis de distribuciones

Herramientas visuales:

- **Histograma**: distribución de frecuencias.
- **Boxplot**: resumen de 5 números + outliers.
- **Q–Q plot**: comparación con distribución teórica.
- **Violin plot**: combina boxplot + densidad.

Tests de normalidad:

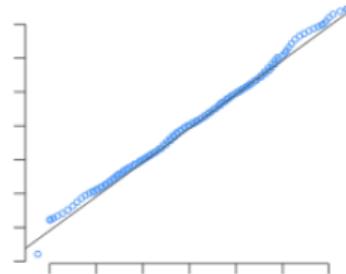
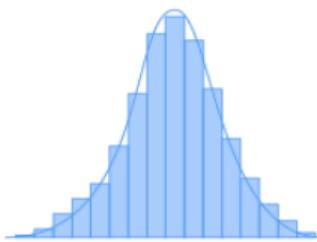
- **Shapiro–Wilk** ($n < 5000$): muy sensible, detecta desviaciones leves de normalidad.
- **Kolmogorov–Smirnov**: compara la distribución empírica con la teórica de normal.
- **Anderson–Darling**: variante más estricta, da más peso a las colas de

```
from scipy import stats

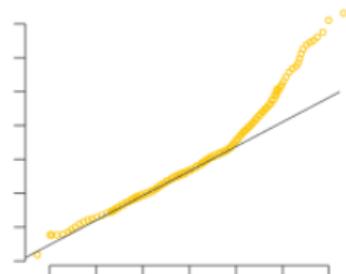
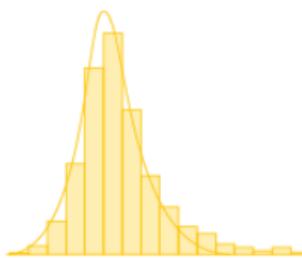
# Test de normalidad (Shapiro-Wilk)
w_stat, p = stats.shapiro(data)

# Test basado en skew y kurtosis
# (D'Agostino-Pearson)
k2, p2 = stats.normaltest(data)
```

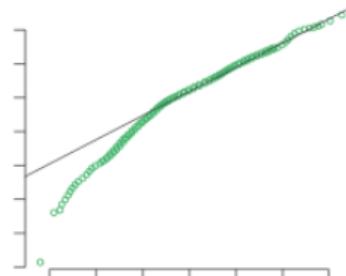
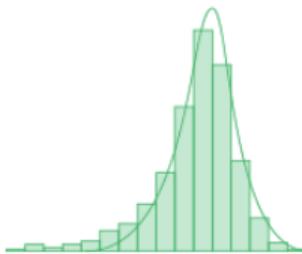
Normally distributed
data

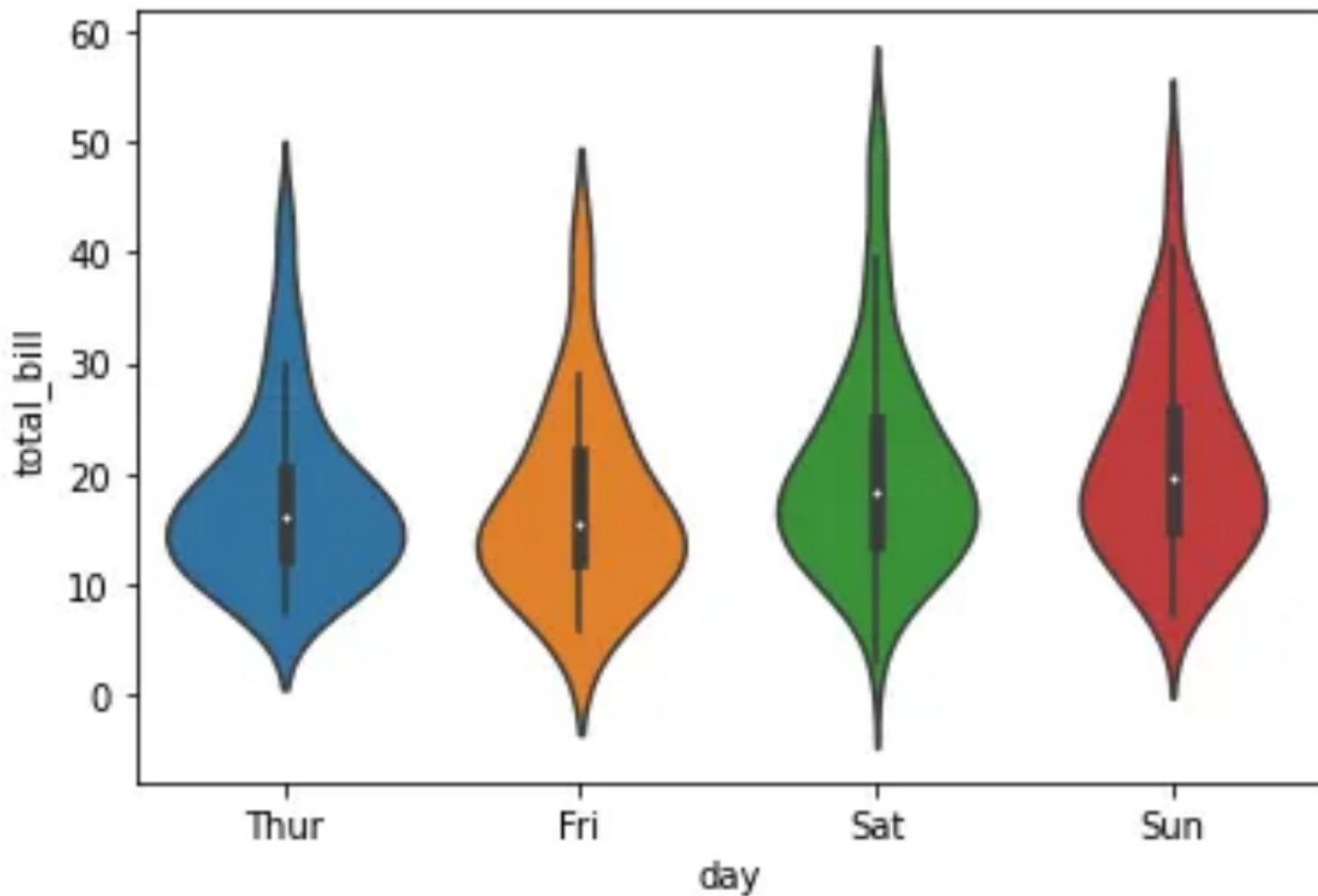


Right-skewed
data



Left-skewed
data





Forma y dependencia

- **Asimetría y curtosis:** forma de la distribución.
- **Pearson/Spearman:** lineal y monótona, respectivamente.
- **Información mutua:** dependencias no lineales.

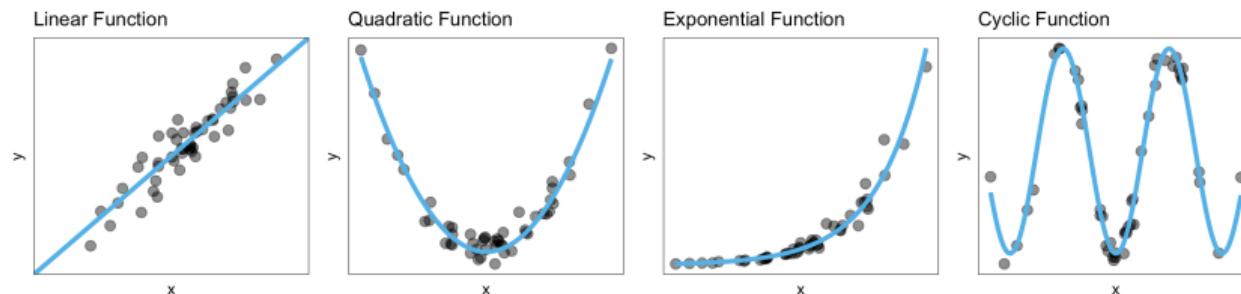
Forma Funcional de la Relación

Forma funcional

Ddescribe algebraicamente la relación entre dos atributos o variables.

Formas comunes:

- Lineal
- Cuadrática
- Exponencial
- Cíclica (ej. sinusoidal)



Nota: Los datos reales no siguen el patrón perfecto, se busca identificar la tendencia general.

Ejemplo: precio medio de vivienda vs. ingreso medio → tendencia lineal.

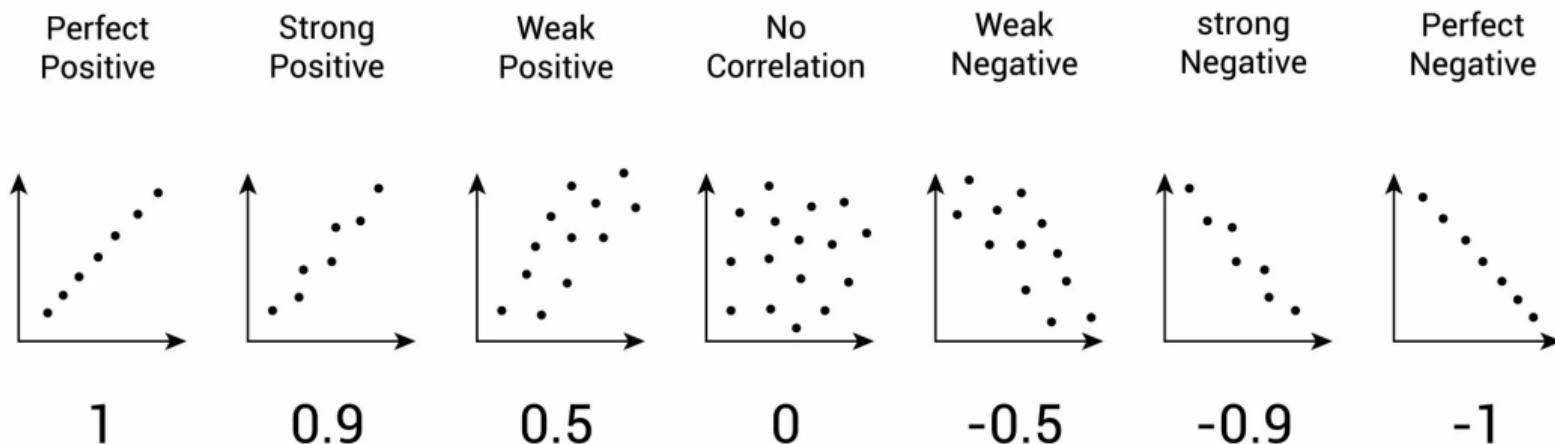
Correlación: Dirección y Tendencia

Dirección y Tendencia

Describe la pendiente general de los datos: positiva, negativa o nula.

Tipos:

- **Positiva:** $\uparrow x$ implica $\uparrow y$.
- **Negativa:** $\uparrow x$ implica $\downarrow y$.
- **Plana:** no existe relación.



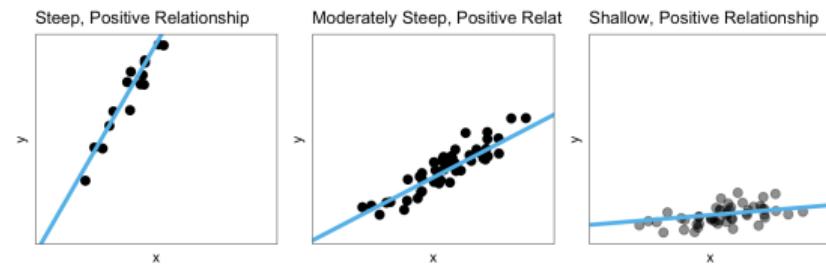
Magnitud de la Relación

Definición

La **magnitud** mide cuánto cambia la variable y ante un cambio en x .

Interpretación:

- Relación **empinada**: gran cambio en y .
- Relación **moderada**: cambio medio en y .
- Relación **suave**: cambio pequeño en y .



Ejemplo: tres relaciones lineales positivas con distinta magnitud (empinada, moderada, suave).

1

0.8

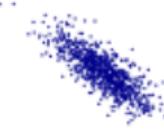
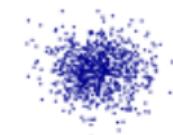
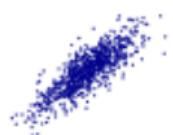
0.4

0

-0.4

-0.8

-1



1

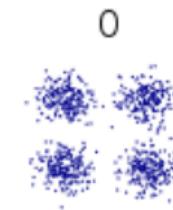
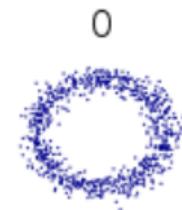
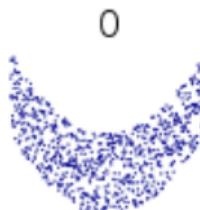
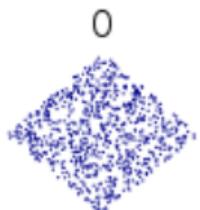
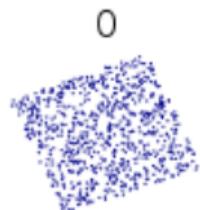
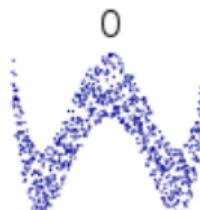
1

1

-1

-1

-1



Tipos de correlación

Pearson (lineal):

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

Spearman (monótona):

- Basada en rangos
- Robusta a outliers
- Detecta relaciones no lineales monótonas

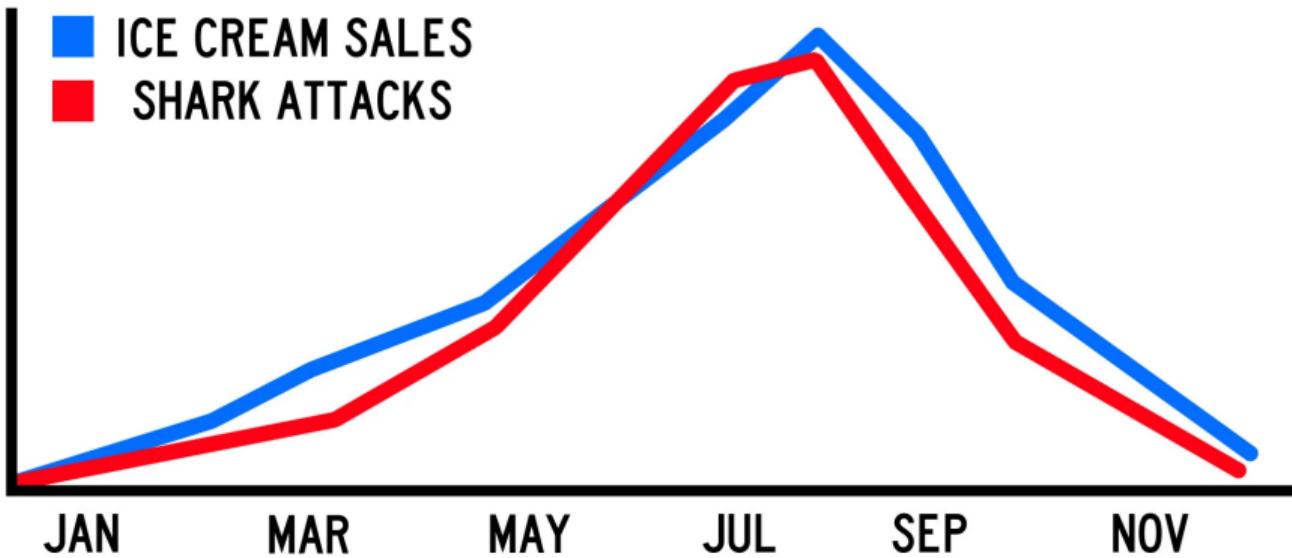
Información mutua (no lineal):

$$I(X;Y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

Importante

Correlación \neq Causalidad

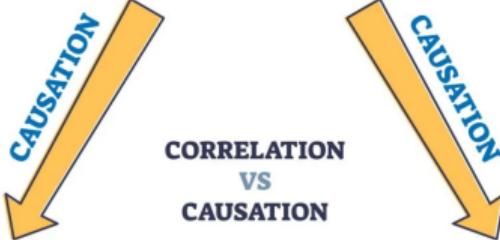
CORRELATION IS NOT CAUSATION!



Both ice cream sales and shark attacks increase when the weather is hot and sunny, but they are not caused by each other (they are caused by good weather, with lots of people at the beach, both eating ice cream and having a swim in the sea)



HOT WEATHER

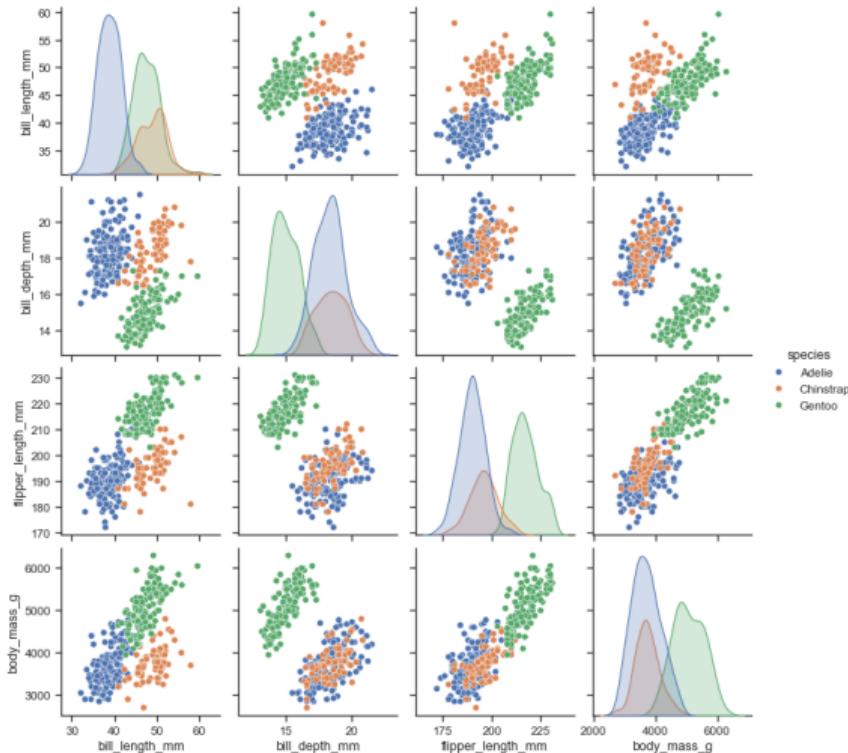


SUNBURN



ICE CREAM SALES

Scatterplot Matrix



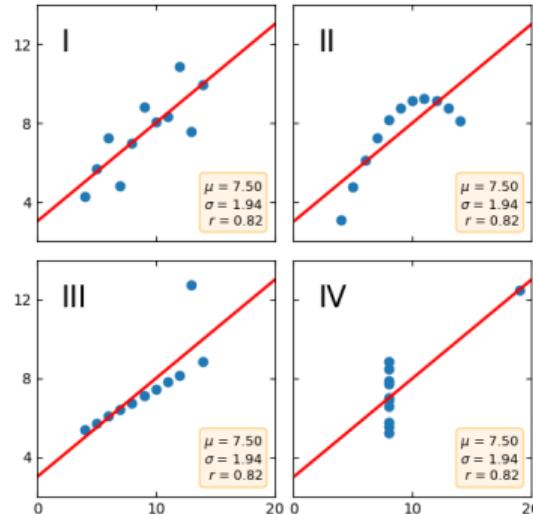
```
import seaborn as sns  
sns.set_theme(style="ticks")  
  
df = sns.load_dataset("penguins")  
sns.pairplot(df, hue="species")
```

Este gráfico fue generado con .

Dependencias y correlaciones

- Dispersión: `df.plot.scatter('f1','f2')`.
- Matriz de correlación: `df[cols].corr()`.
- Rango: $[-1, 1]$; 0 indica linealmente independiente.
- Multicolinealidad afecta regresiones (inversas de matrices).
- Árboles son menos sensibles a colinealidad.
- Correlación alta con el objetivo ayuda a lineales.

Visualización efectiva - Principios perceptuales



Canales preatentivos (procesamiento automático):

- Color → Categorías
- Tamaño → Magnitud
- Posición → Valores cuantitativos
- Forma → Tipos

Ley de Weber-Fechner:

- Percepción logarítmica de diferencias
- Justifica escalas log para rangos amplios

Visualización efectiva - Principios perceptuales

Principios de diseño:

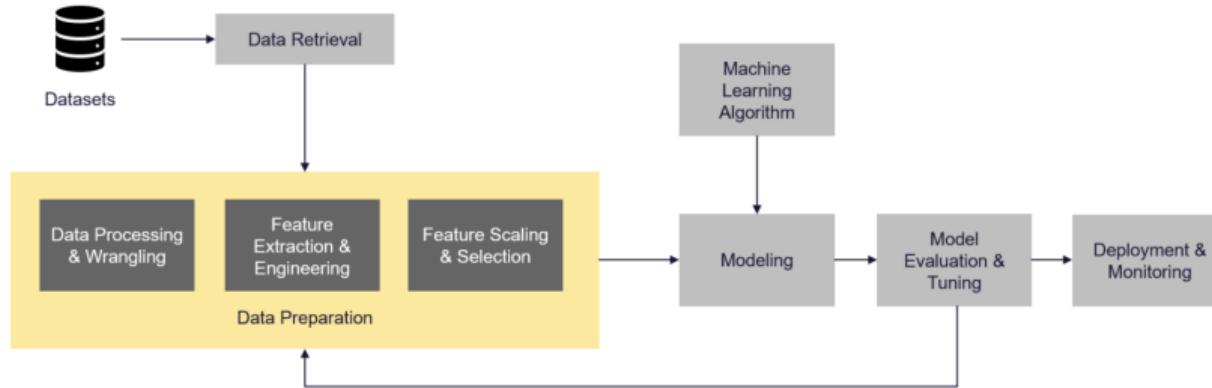
- Maximizar ratio datos/tinta (Tufte)
- Evitar *chartjunk*
- Mantener consistencia visual
- Usar color con propósito

Gramática de gráficos

Datos, estéticas, geometrías, facetas, estadísticas y temas.

Ingeniería de Características

¿Qué es la ingeniería de características?



"Feature engineering is the art of making data useful" – Domingos, 2012

Definición:

- Proceso de crear representaciones útiles (nuevas variables) de datos crudos.
- Traducir conocimiento del dominio a variables computables
- Maximizar señal predictiva
- **Intuición:** ¿Qué usaría un humano para predecir?

¿Qué es la ingeniería de características?

Impacto:

- Puede mejorar performance más que cambiar algoritmo
- Diferencia entre modelo mediocre y excepcional
- 80 % del éxito en competencias de Kaggle

Ecuación fundamental

Mejor data \times Algoritmo simple > Data pobre \times Algoritmo complejo

- Selección: filtrar irrelevantes y redundantes.
- Construcción: productos, polinomios, logaritmos, kernels.
- Codificación: representar categóricas numéricamente.

Tipos de transformaciones de características

1 Creación:

- Ratios: ingresos/gastos
- Interacciones: edad × educación
- Agregaciones: promedio_últimos_3_meses

2 Transformación:

- Log para distribuciones sesgadas
- Box-Cox para normalización
- Binning para discretización

3 Codificación:

- One-hot para nominales
- Ordinal para categorías ordenadas
- Target encoding (con regularización)

4 Selección:

- **Filtros:** correlación, información mutua
- **Wrapper:** RFE, forward/backward
- **Embedded:** LASSO, Random Forest

Manejo de valores faltantes

Tipos de faltantes (Rubin, 1976):

- **MCAR** (Missing Completely At Random): Aleatorio puro
 - **MAR** (Missing At Random): Depende de variables observadas
 - **MNAR** (Missing Not At Random): Depende del valor faltante

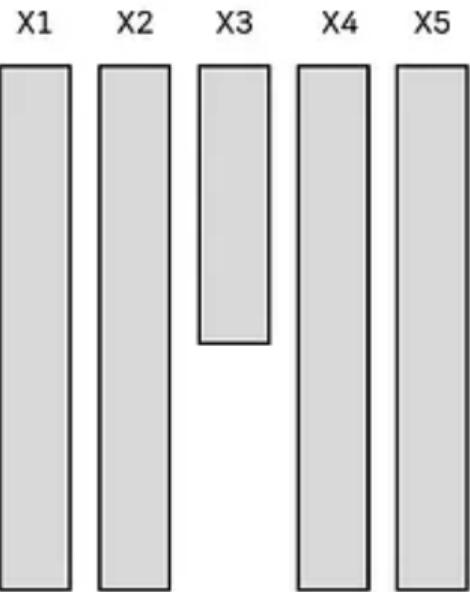
MCAR

Observed		Unobserved
Light Purple	Light Purple	Light Purple
Dark Grey	Teal	Teal
Teal	White	Teal
Teal	Teal	Teal
Teal	Teal	Teal
Dark Grey	White	Teal
Teal	White	Teal
Teal	Teal	Teal
Dark Grey	Teal	Teal

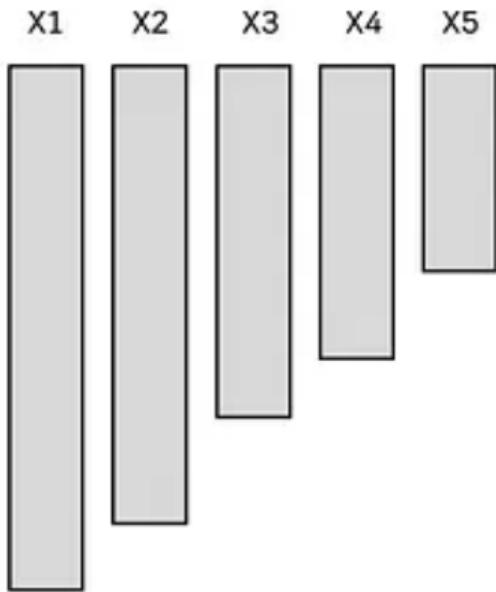
MAR

MNAR

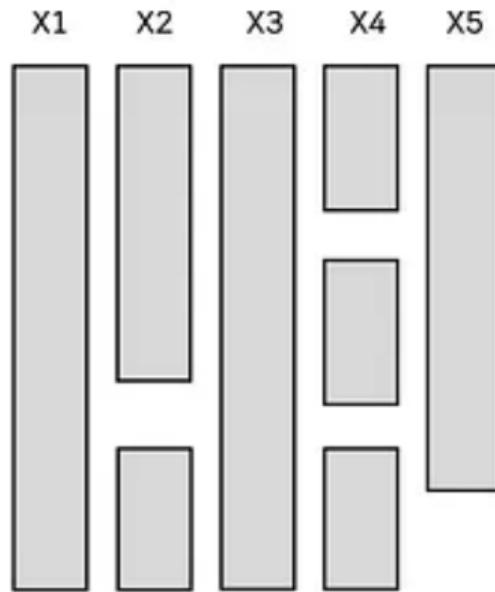
The figure consists of two vertical columns of colored bars. The left column, labeled "Observed", contains 10 bars arranged in a 2x5 grid. The right column, labeled "Unobserved", contains 10 bars arranged in a single vertical column. The colors of the bars vary across the columns, representing different categories or states.



a. Univariate Pattern

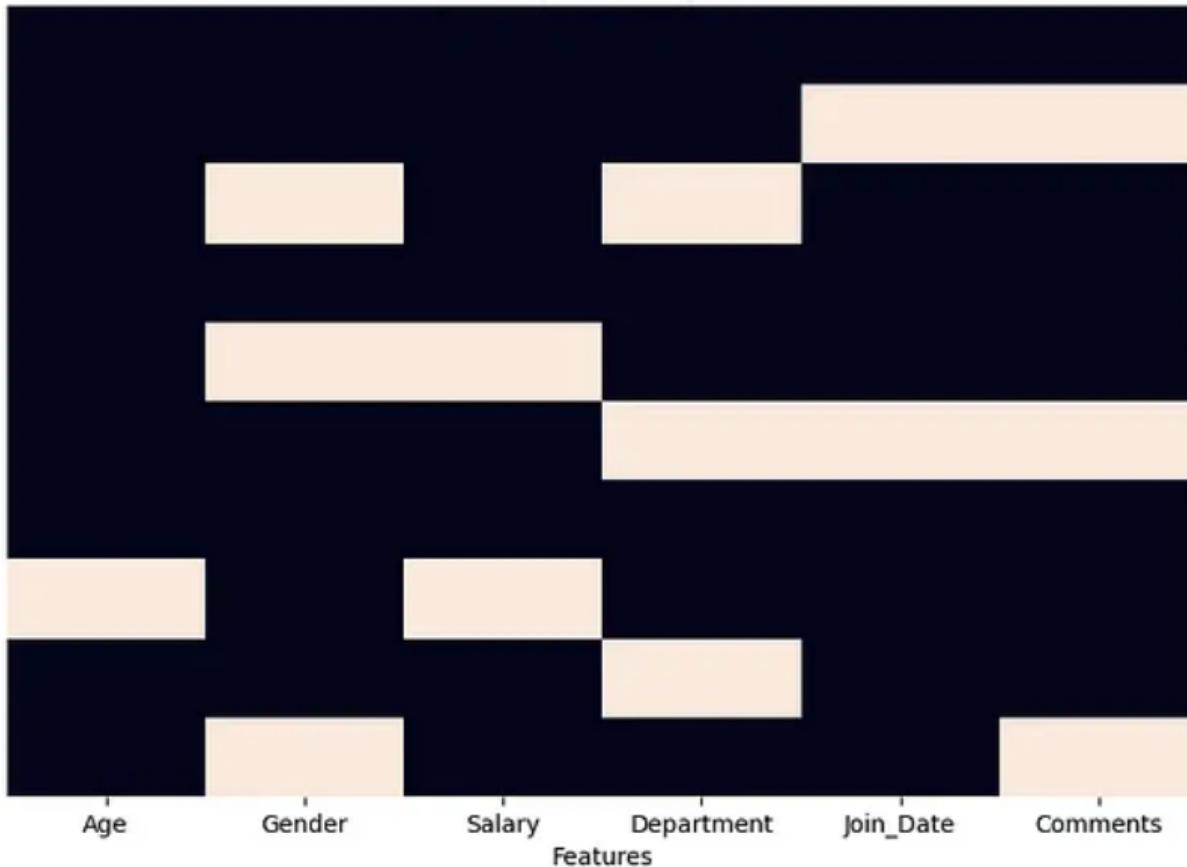


b. Monotone Pattern

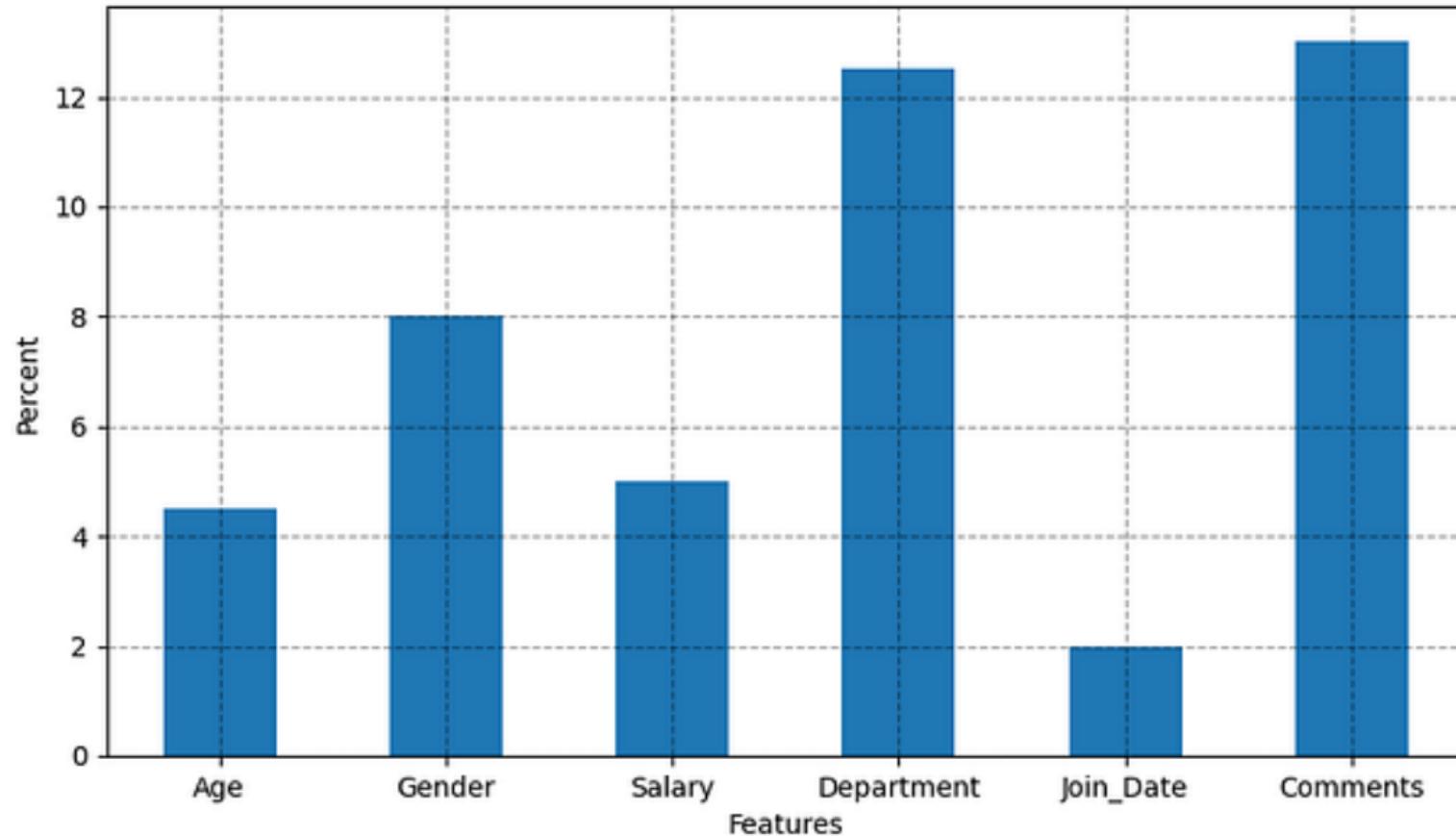


c. General Pattern

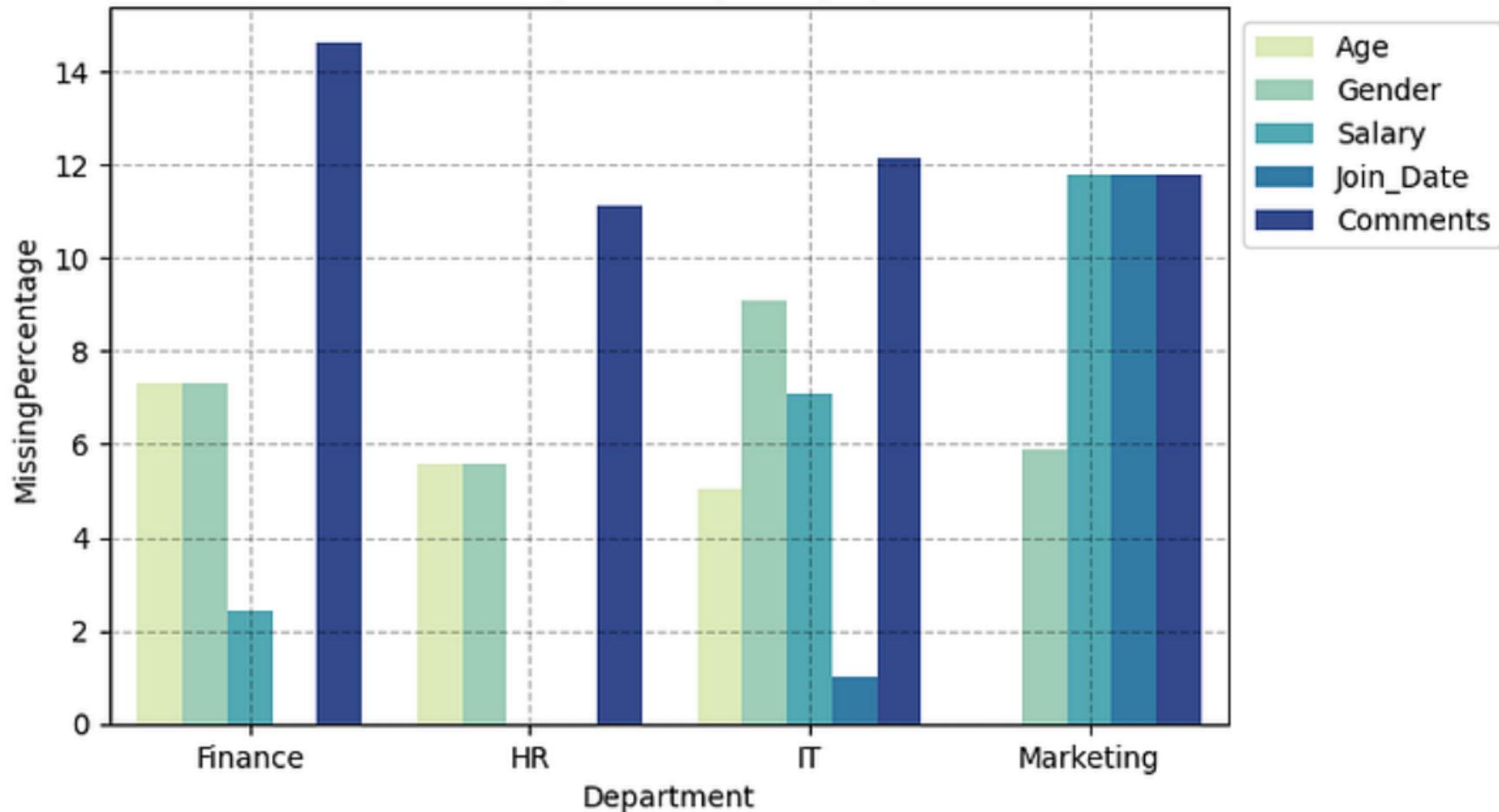
Heatmap of Missing Data



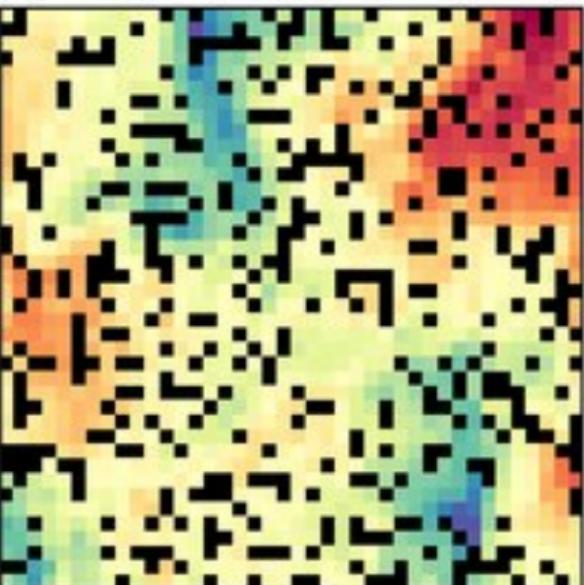
Bar plot of Missing Data



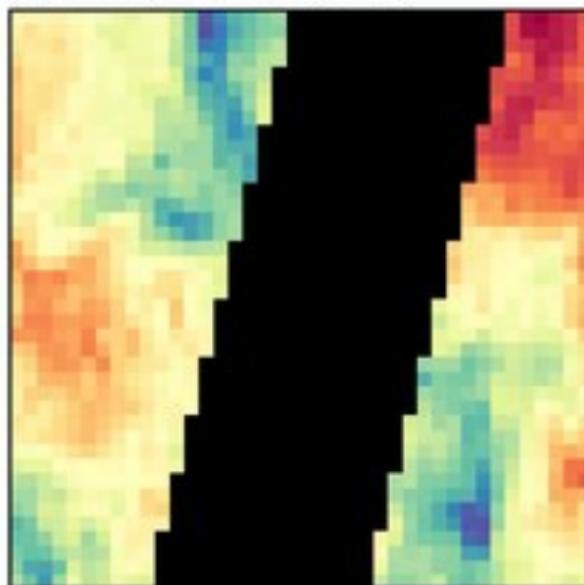
Missing Values by Category



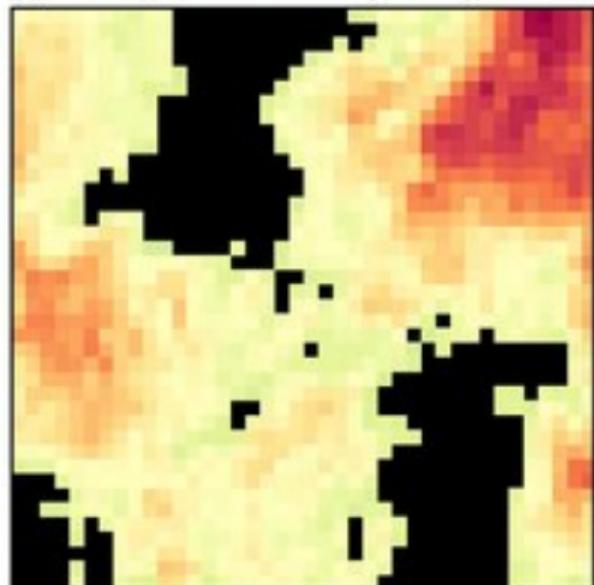
(a) Missing completely at random (MCAR)



(b) Missing at random (MAR)



(c) Missing not at random (MNAR)



Manejo de valores faltantes

Estrategias:

- 1 **Eliminación:** Descartar filas/columnas (cuidado con pérdida de señal). Si MCAR y $< 5\%$
- 2 **Imputación simple:** Media, mediana, moda; categóricas: moda.
- 3 **Imputación avanzada:** imputación con ML (p.ej., Datawig, KNN, MICE, missForest)
- 4 **Indicador:** Variable binaria “era_faltante”
- 5 **Placeholder:** valor constante reservado.
- 6 Código: `df['col'].fillna(df['col'].mean()),
df['col'].fillna(df['col'].mode())`.

Cuidado

La estrategia depende del mecanismo de faltantes

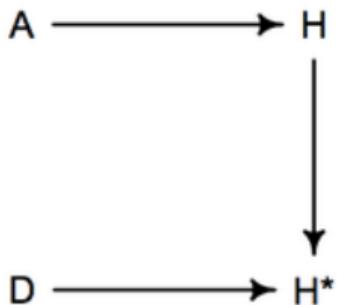
H: Homework

H*: Homework with missing values

A: Attribute of student

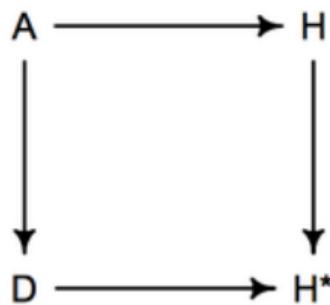
D: Dog (missingness mechanism)

DOG EATS
ANY
HOMEWORK



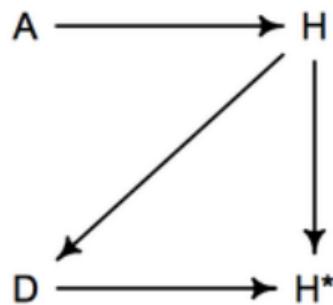
MISSING COMPLETELY
AT RANDOM

DOG EATS
STUDENTS'
HOMEWORK



MISSING
AT RANDOM

DOG EATS
BAD
HOMEWORK



MISSING NOT
AT RANDOM

SimpleImputer en sklearn

- `SimpleImputer(missing_values=np.nan, strategy='mean', fill_value=None)`.
- Numéricas: `strategy='mean'` o `'median'`.
- Numéricas/Categóricas: `'most_frequent'` o `'constant'`.
- Métodos: `.fit()` aprende; `.transform()` aplica; `.fit_transform()`.

Escalamiento y normalización

¿Por qué escalar?

- Algoritmos basados en distancia (KNN, SVM) y redes dependen de escala.
- Convergencia más rápida en gradient descent
- Evitar dominancia numérica
- Siempre *fit* en *train*; *transform* en *dev/test*.

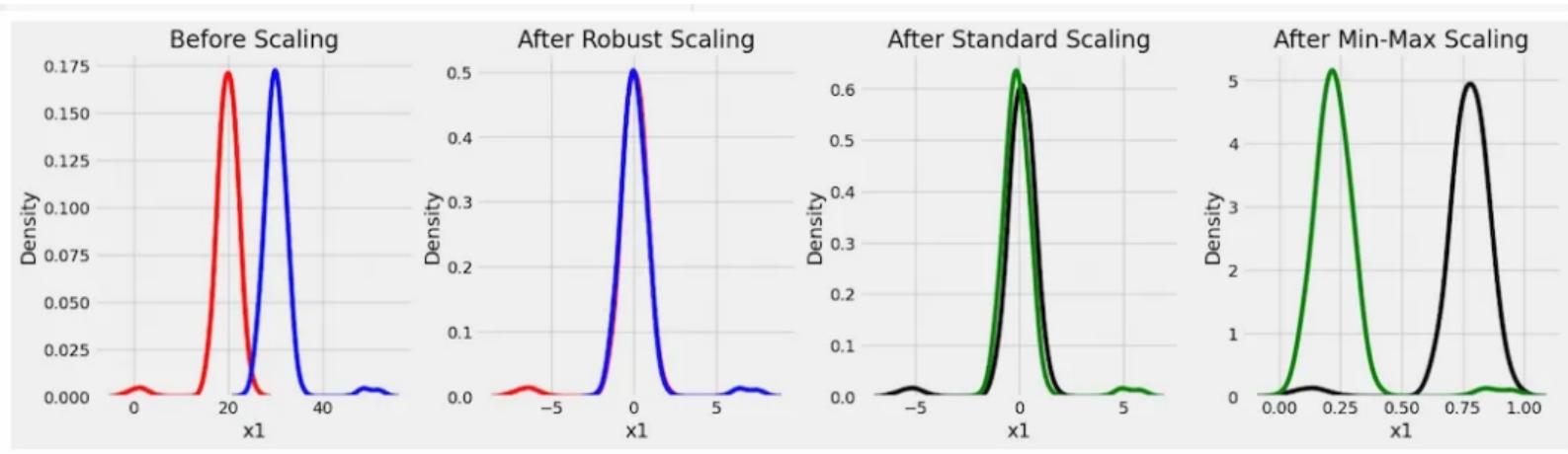
Escalamiento y normalización

Métodos principales:

Método	Fórmula	Uso
StandardScaler	$z = \frac{x - \mu}{\sigma}$	Normal, media 0 y DE 1 por columna, sin outliers
MinMaxScaler	$x' = \frac{x - \min}{\max - \min}$	Rango acotado, mapea a [0,1] por columna
RobustScaler	$x' = \frac{x - Q_2}{Q_3 - Q_1}$	Con outliers
Normalizer	$x' = \frac{x}{\ x\ _p}$	Vectores unitarios

- Standard vs. MinMax vs. Robust.
- Box–Cox y log para positividad y colas.
- Quantile para normalización marginal.

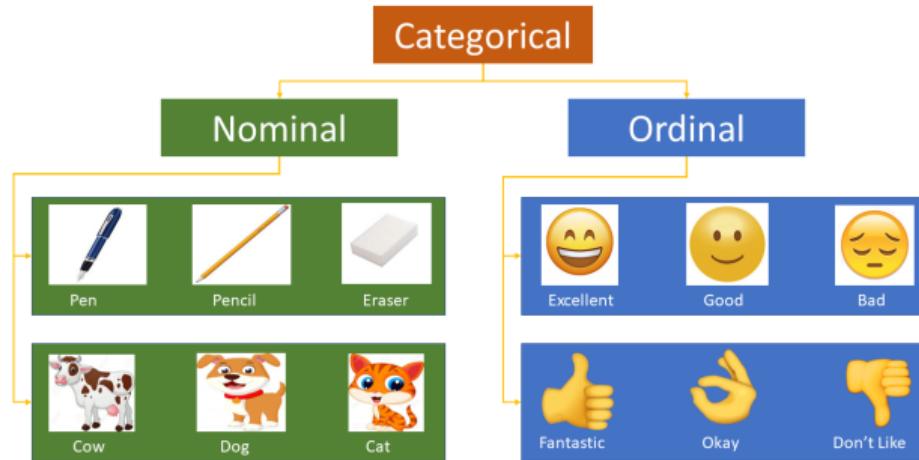
Codificación (encoding)



Regla práctica

Fit en train, transform en test (evitar leakage)

Codificación (encoding)



- **Ordinal**: categorías con orden (ej.: talla S<M<L).
- **Nominal**: sin orden (ej.: color {green, red, blue}).
- Cuidado: enteros ordinales pueden inducir órdenes ficticios.
- LabelEncoder: codifica una sola columna o el objetivo.
- OrdinalEncoder: codifica múltiples columnas categóricas.

Codificación de variables categóricas

One-Hot Encoding: ortogonalidad y simplicidad.

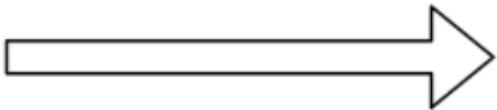
- Para nominales
- Evita orden artificial de enteros ordinales
- Crea k columnas binarias: expande en variables binarias por categoría
- Problema: alta dimensionalidad
- Pandas: `get_dummies`
- Para una sola variable: `LabelBinarizer`

Ejemplo:

Color	R	G	B
Rojo	1	0	0
Verde	0	1	0
Azul	0	0	1

Color
Red
Green
Blue

One-hot
encoding



d1	d2	d3
1	0	0
0	1	0
0	0	1

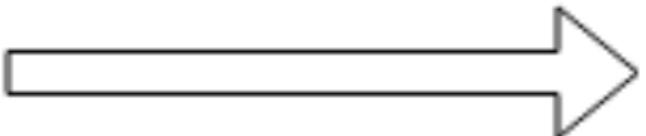
Color

Red

Green

Blue

Dummy
encoding



d1

d2

1

0

0

1

0

0

Codificación de variables categóricas

Target Encoding:

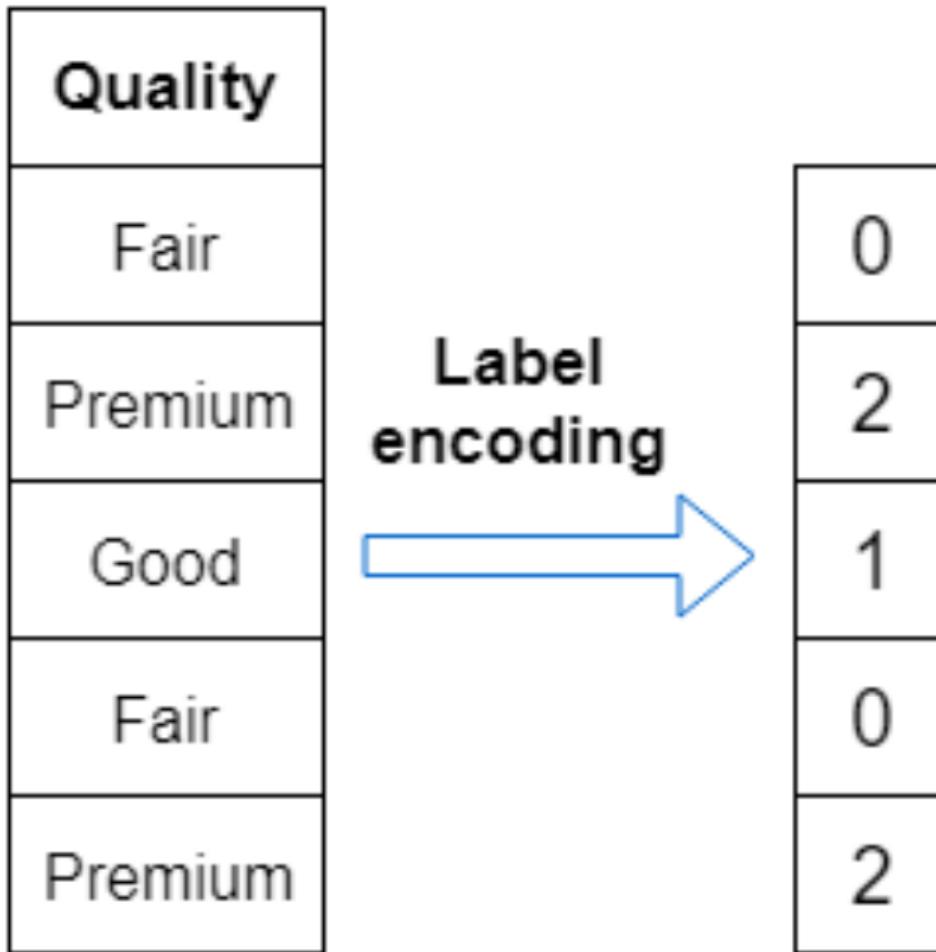
- Reemplaza con media del target
- Requiere regularización
- Riesgo de overfitting
- Con *shrinkage* para alta cardinalidad

Fórmula con smoothing:

$$\hat{y}_c = \frac{n_c \bar{y}_c + \alpha \bar{y}}{n_c + \alpha}$$

Data leakage

Target encoding debe hacerse dentro de CV folds



Original Data

Team	Points
A	25
A	12
B	15
B	14
B	19
B	23
C	25
C	29

Label Encoded Data



Team	Points
0	25
0	12
1	15
1	14
1	19
1	23
2	25
2	29

Código: LabelEncoder

```
from sklearn.preprocessing import LabelEncoder

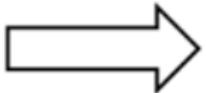
le = LabelEncoder()
y_enc = le.fit_transform(df["classlabel"])

# Inversa:
y_inv = le.inverse_transform(y_enc)

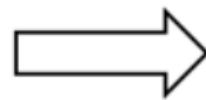
# Mapeo etiqueta->id:
classes = {lab:i for i, lab in enumerate(le.classes_)}
```

Count Encoding

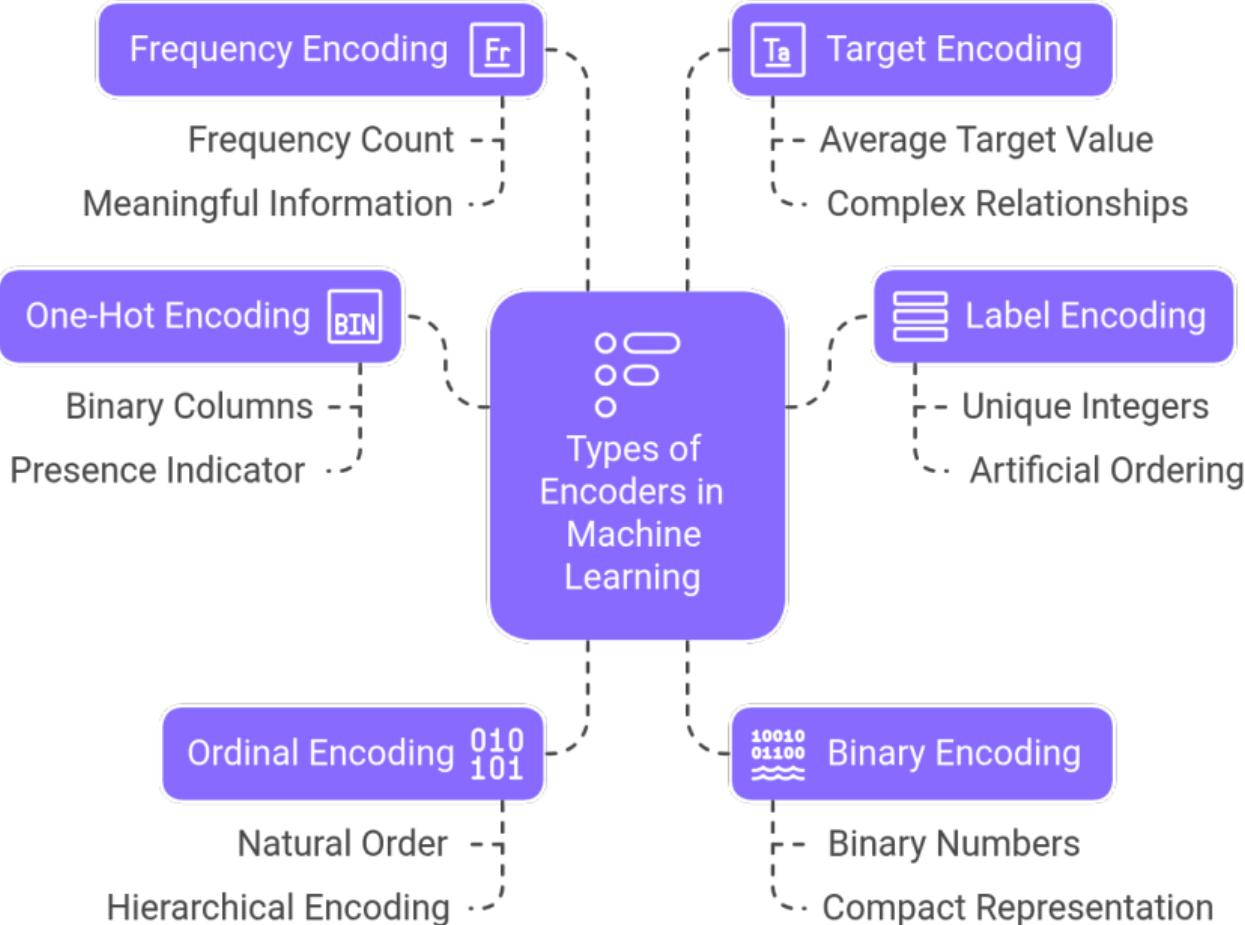
Feature
Apple
Banana
Apple
Banana
Banana



Feature	Count
Apple	2
Banana	3



Feature	Encoded
Apple	2
Banana	3
Apple	2
Banana	3
Banana	3



Código: OrdinalEncoder

```
from sklearn.preprocessing import OrdinalEncoder

oe = OrdinalEncoder(
    categories=[[ "blue", "green", "red"], [ "XS", "S", "M", "L", "XL"]],  
    handle_unknown="use_encoded_value", unknown_value=-1  

)
X_ord = oe.fit_transform(df[["color", "size"]])
```

Demasiadas categorías

- Jerarquías: región → estado → ciudad.
- Elegir nivel apropiado de agregación.
- Agrupar en *bins* (ej.: grupos de edades).

Creación de características - Ejemplos prácticos

Dominio temporal:

- Día de la semana, mes, trimestre
- Es fin de semana, es feriado
- Días desde último evento
- Características cíclicas: $\sin(2\pi t/T)$, $\cos(2\pi t/T)$

Dominio texto:

- Longitud, número de palabras
- TF-IDF, n-gramas
- Embeddings (Word2Vec, BERT)

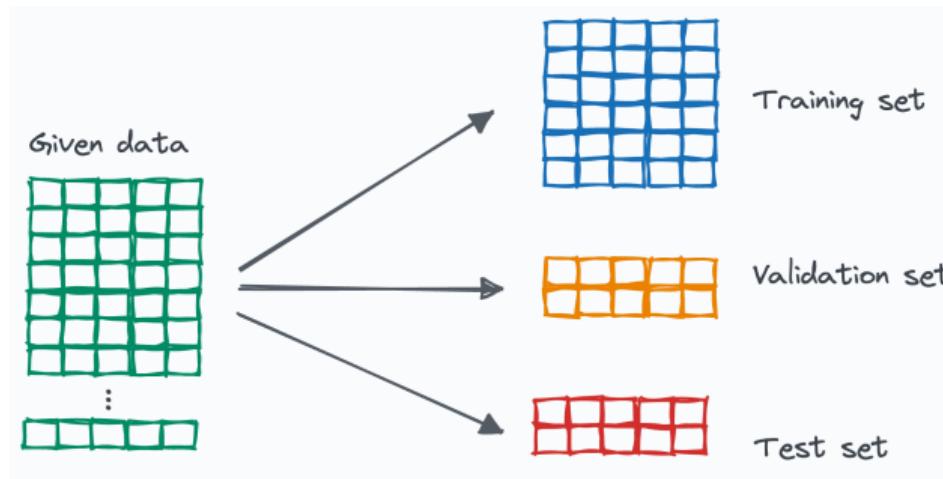
Dominio numérico:

- Polinomios: x^2 , x^3 , \sqrt{x}
- Interacciones: $x_1 \times x_2$
- Ratios: $\frac{x_1}{x_2}$
- Estadísticas de ventana: rolling mean/std
- Fourier para estacionalidad y periodicidad.
- Autoencoders para representación compacta.

Validación y Pipelines

Principios de validación

- Generalización se estima con datos no vistos.
- **División de datos:** Separar entrenamiento, validación y prueba.
 - Tamaño del *test* depende de la data.
- Evitar adaptar decisiones al conjunto de prueba.



División de datos - Estrategias

División básica:

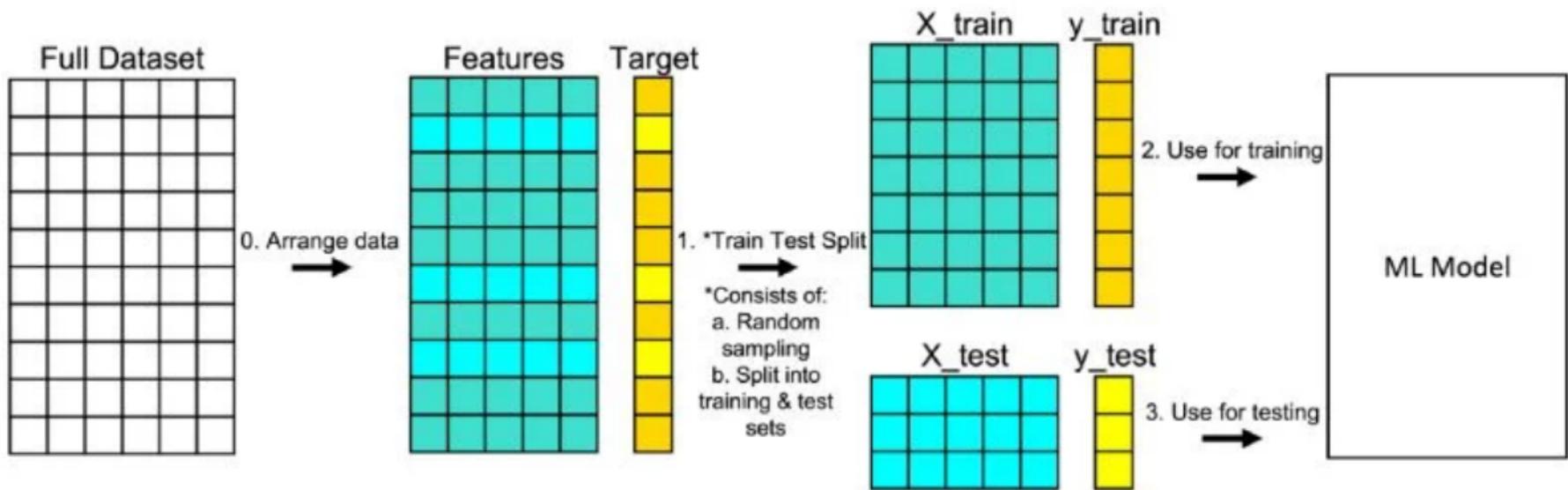
- Train (60-70%): Entrenar modelo
- Validation (15-20%): Ajustar hiperparámetros
- Test (15-20%): Evaluación final

Consideraciones especiales:

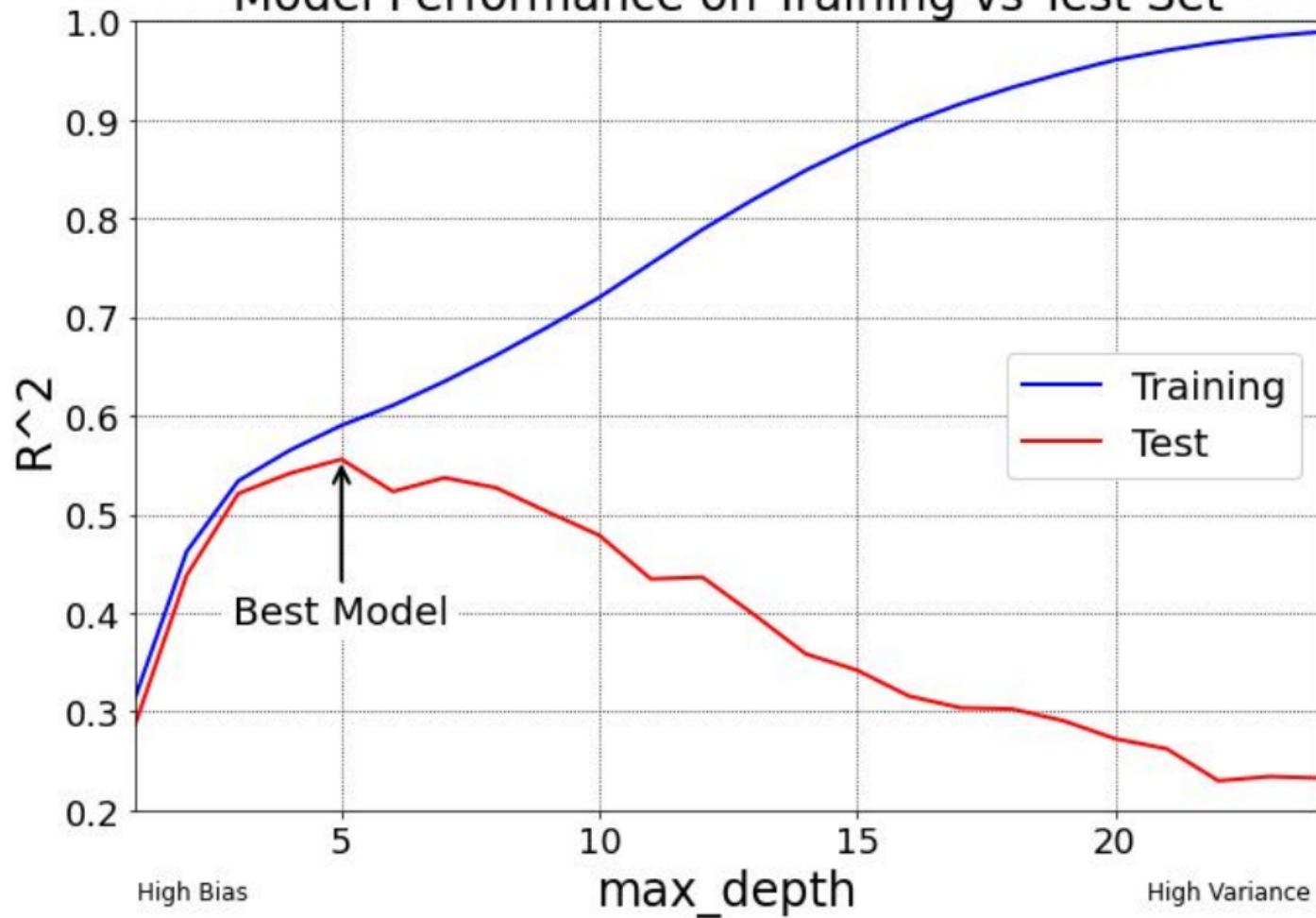
- **Temporal:** Respetar orden cronológico
- **Estratificada:** Mantener proporciones
- **Por grupos:** GroupKFold para datos dependientes
- **Espacial:** Evitar autocorrelación geográfica
- Validación y prueba deben emular producción.

Regla de oro

Test set es sagrado - usar solo una vez al final



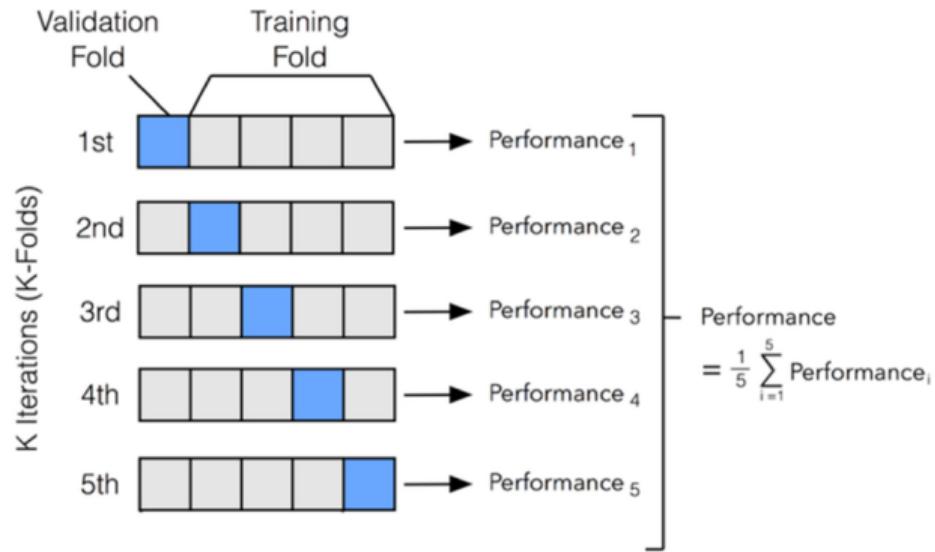
Model Performance on Training vs Test Set



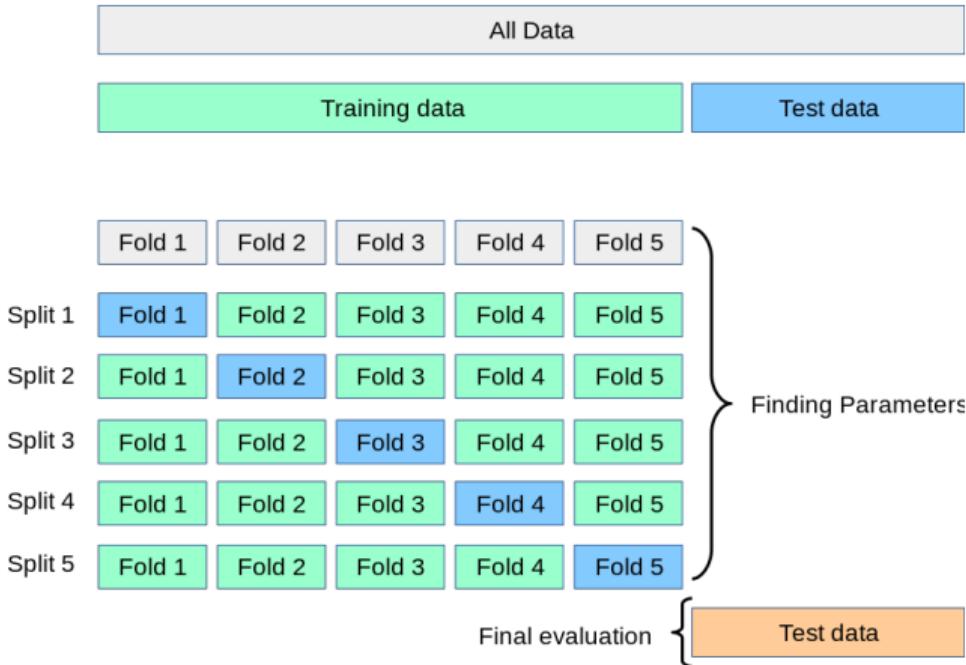
Validación cruzada - Tipos y usos

K-Fold estándar:

- Divide en K partes iguales
- Entrena K modelos
- Promedia resultados



Validación cruzada - Tipos y usos

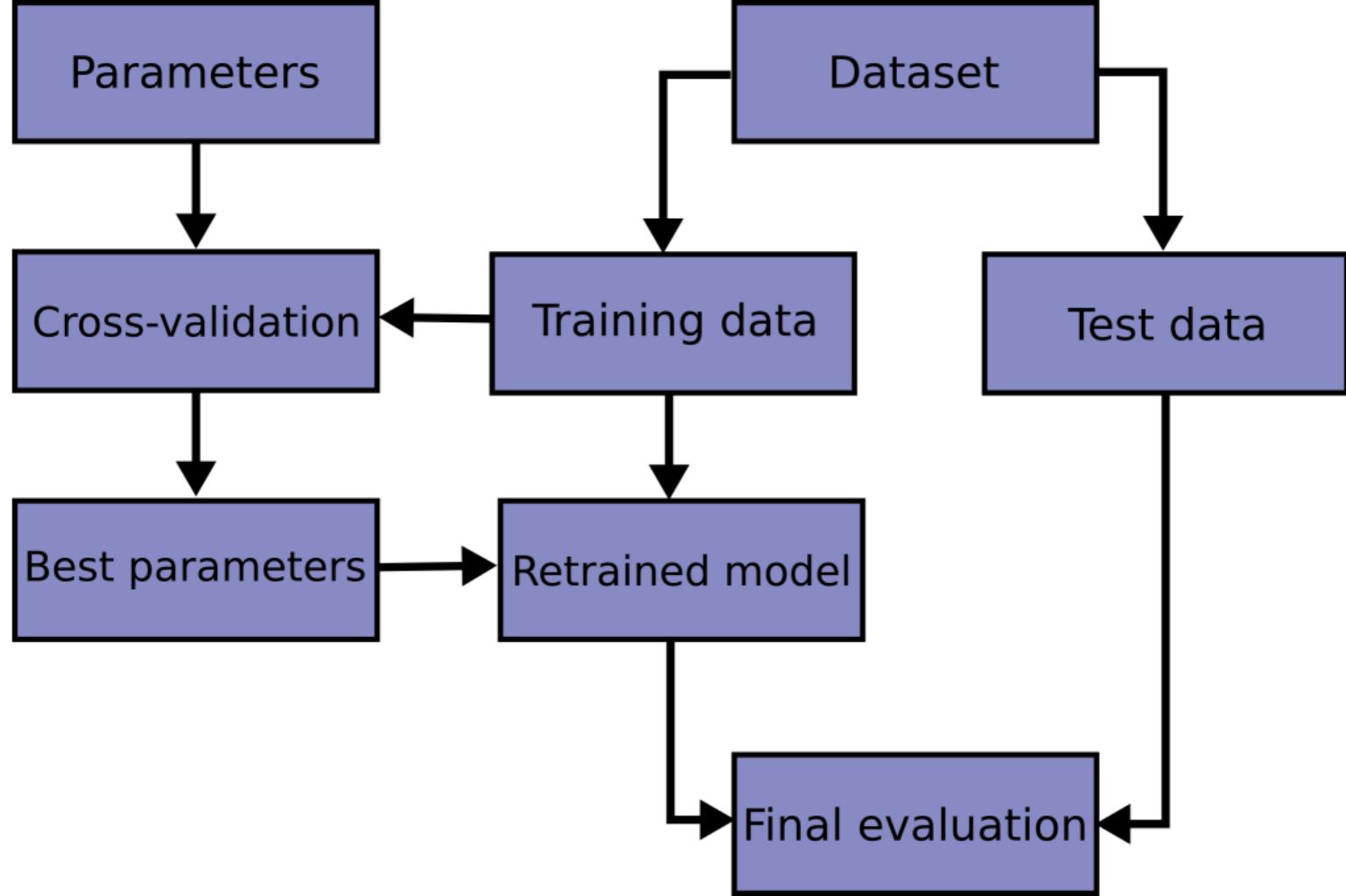


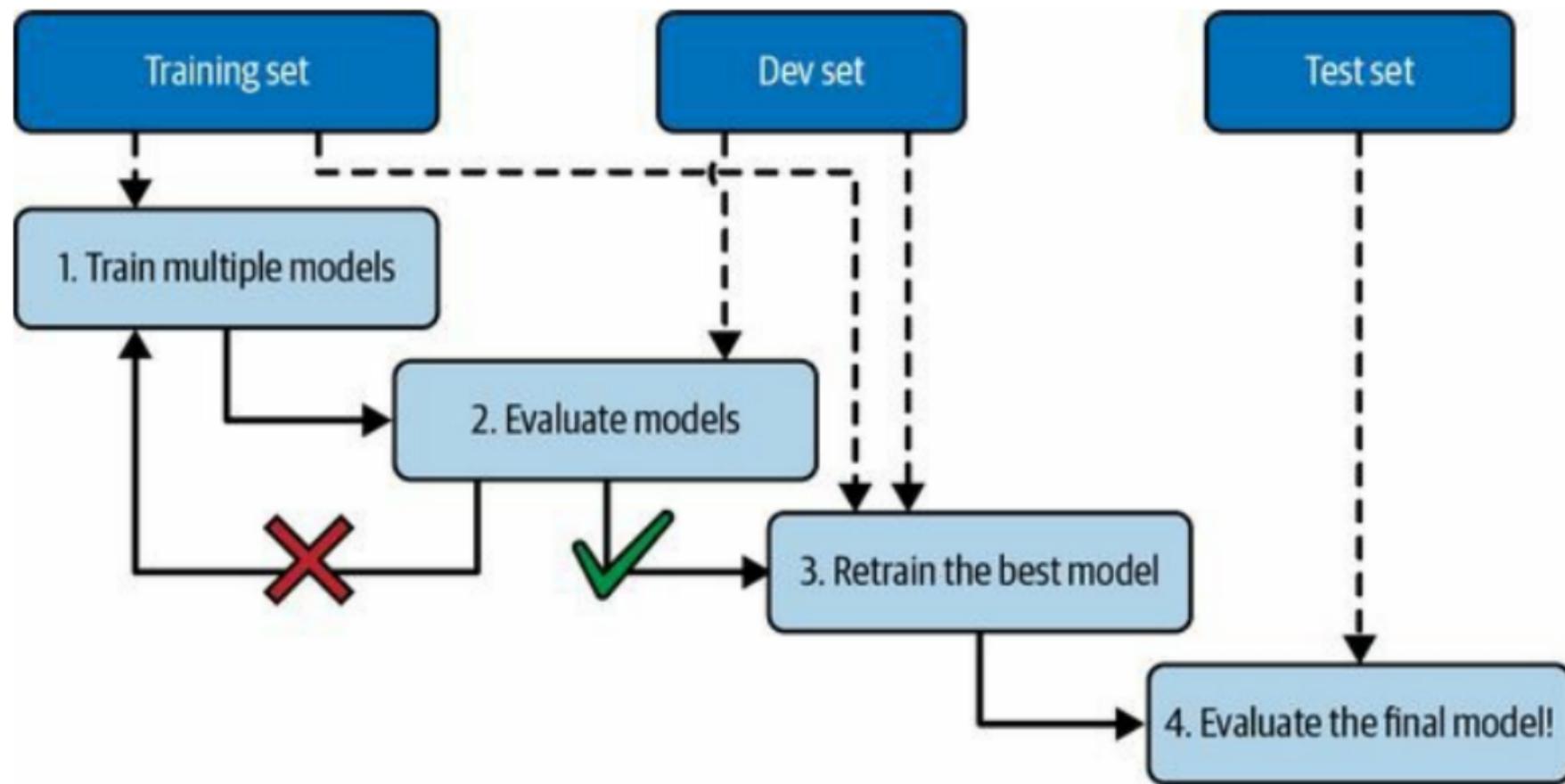
Ventajas:

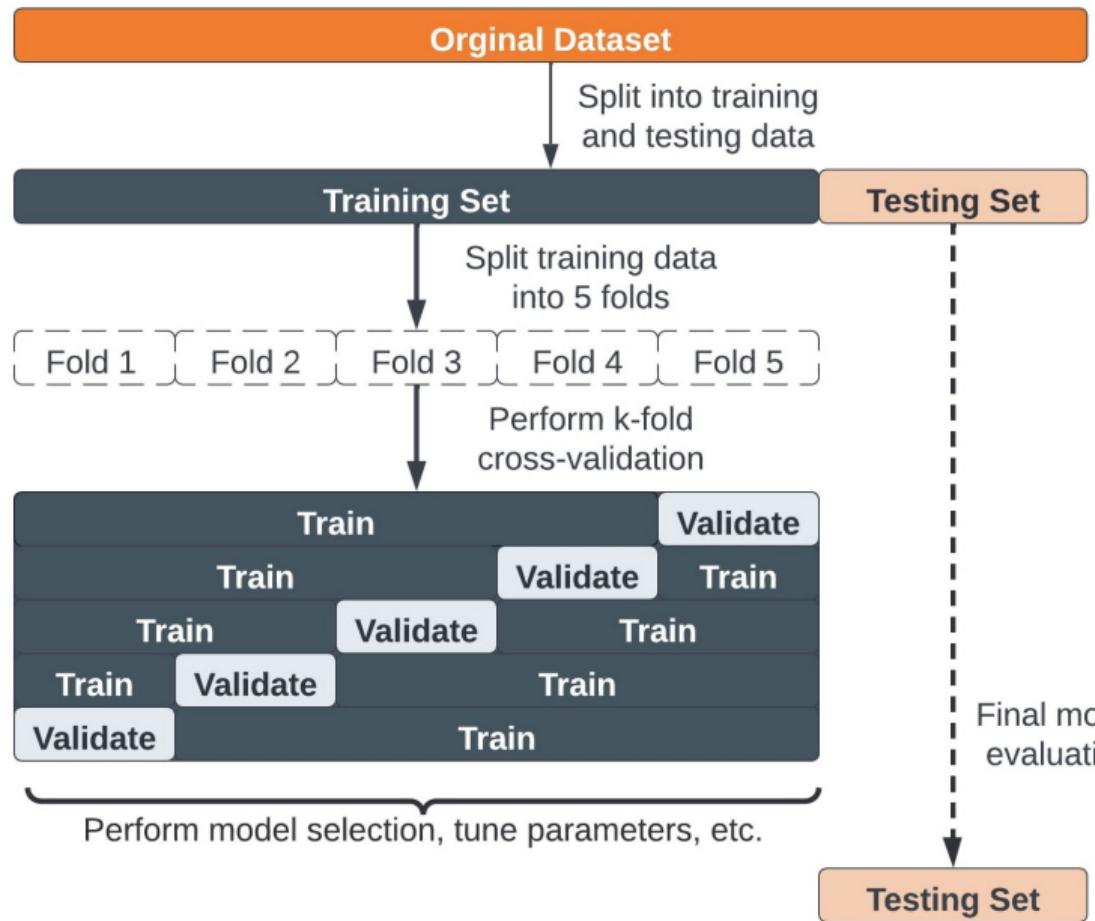
- Estimación robusta
- Usa todos los datos
- Reduce varianza de estimación.
- Mejora estabilidad con costo extra.

Desventajas:

- K veces más lento
- Complejidad adicional



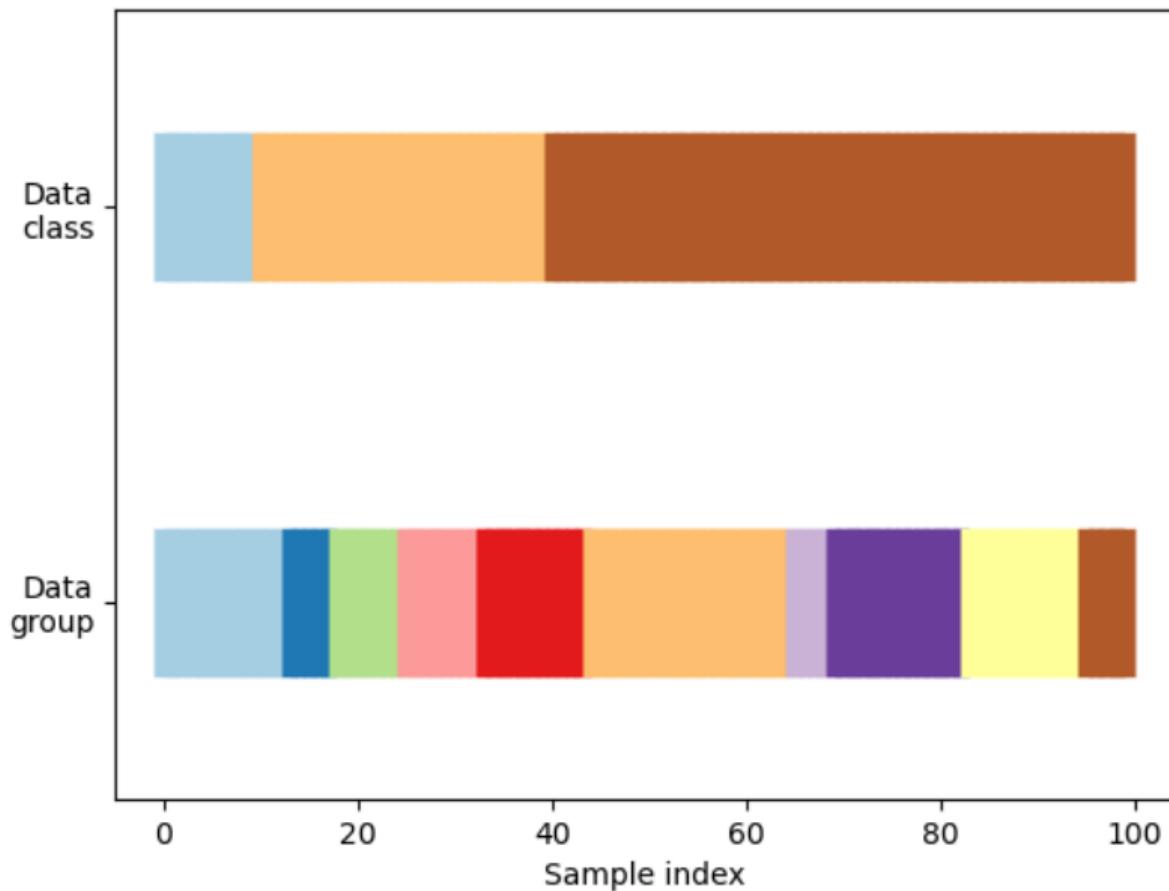




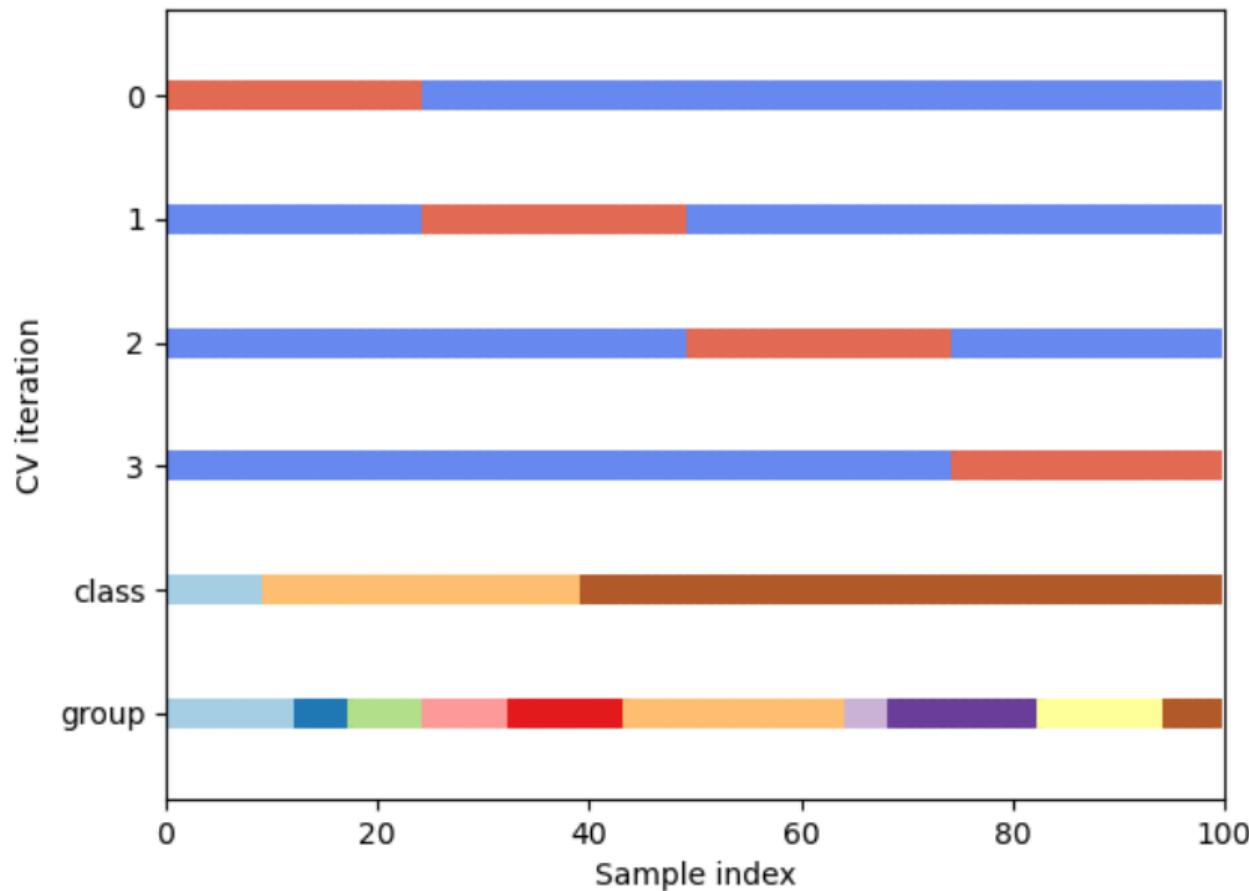
Validación cruzada - Tipos y usos

Variantes especializadas:

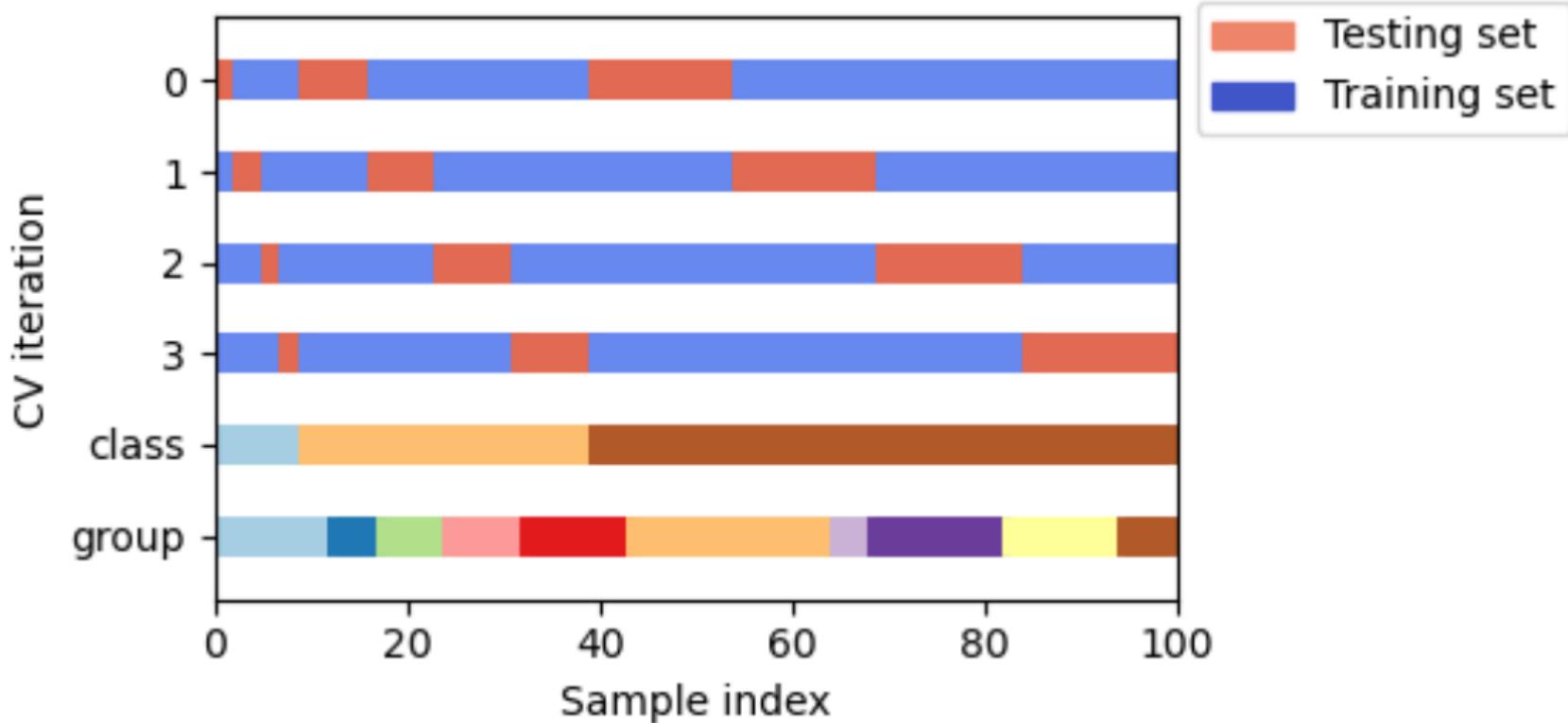
- **StratifiedKFold**: Para clasificación
- **TimeSeriesSplit**: Datos temporales
- **GroupKFold**: Datos agrupados
- **LeaveOneOut**: N folds (costoso)



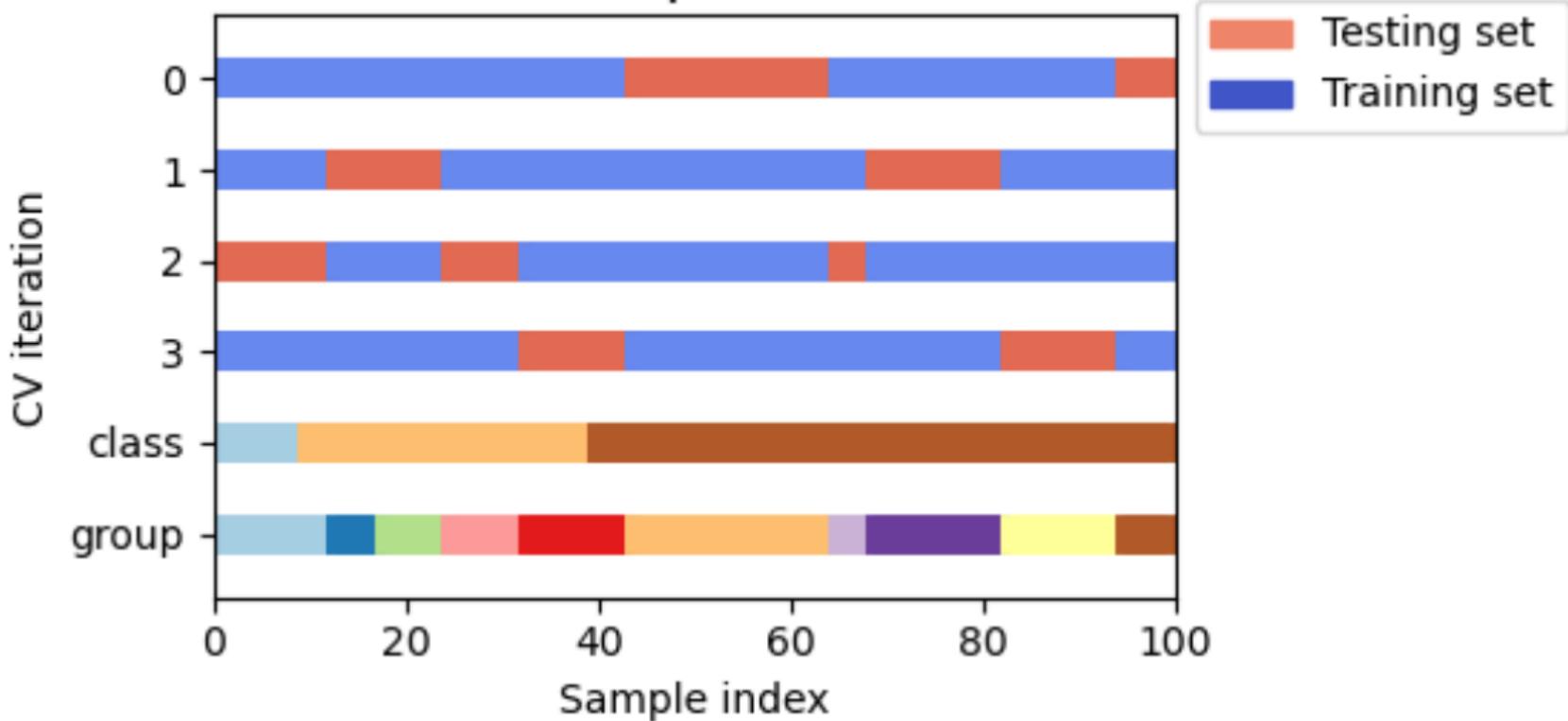
KFold



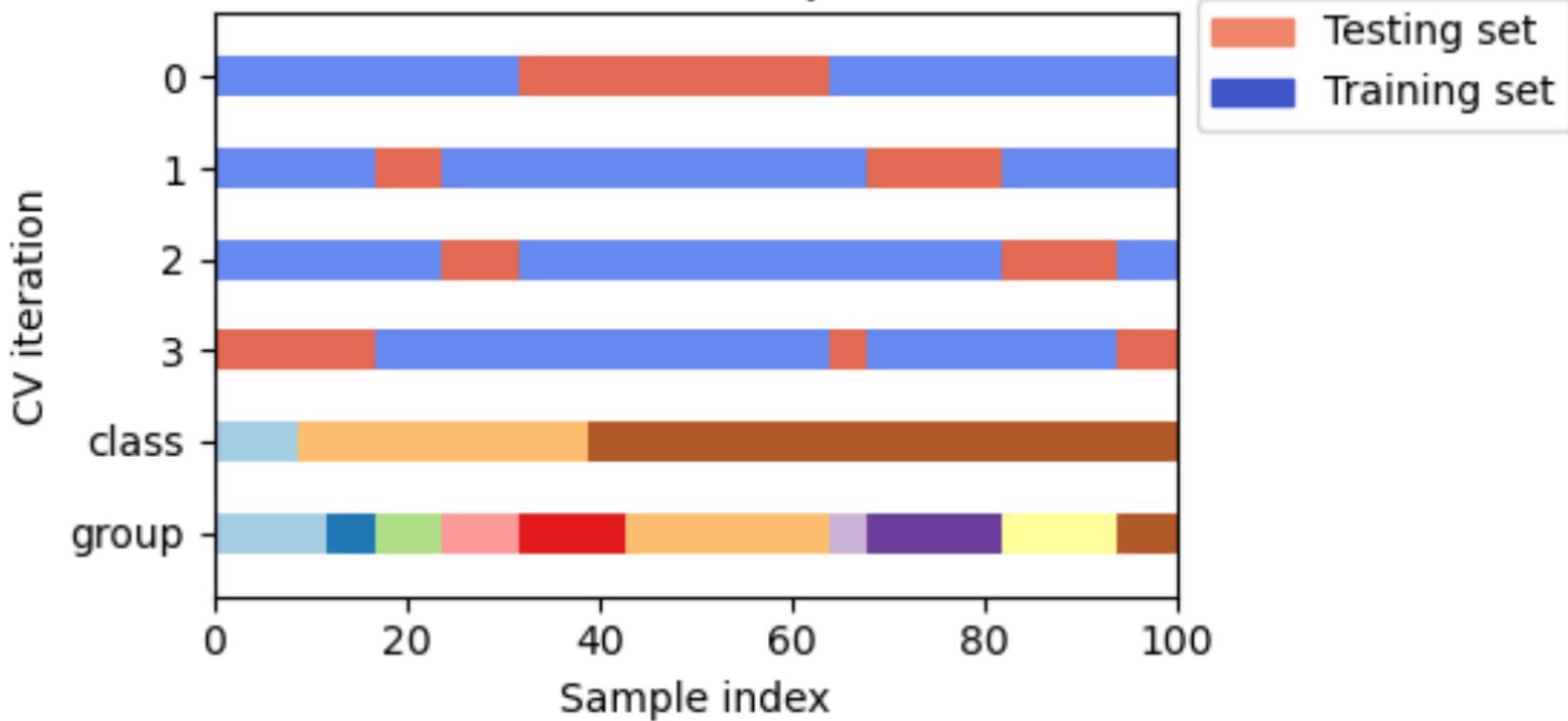
StratifiedKFold



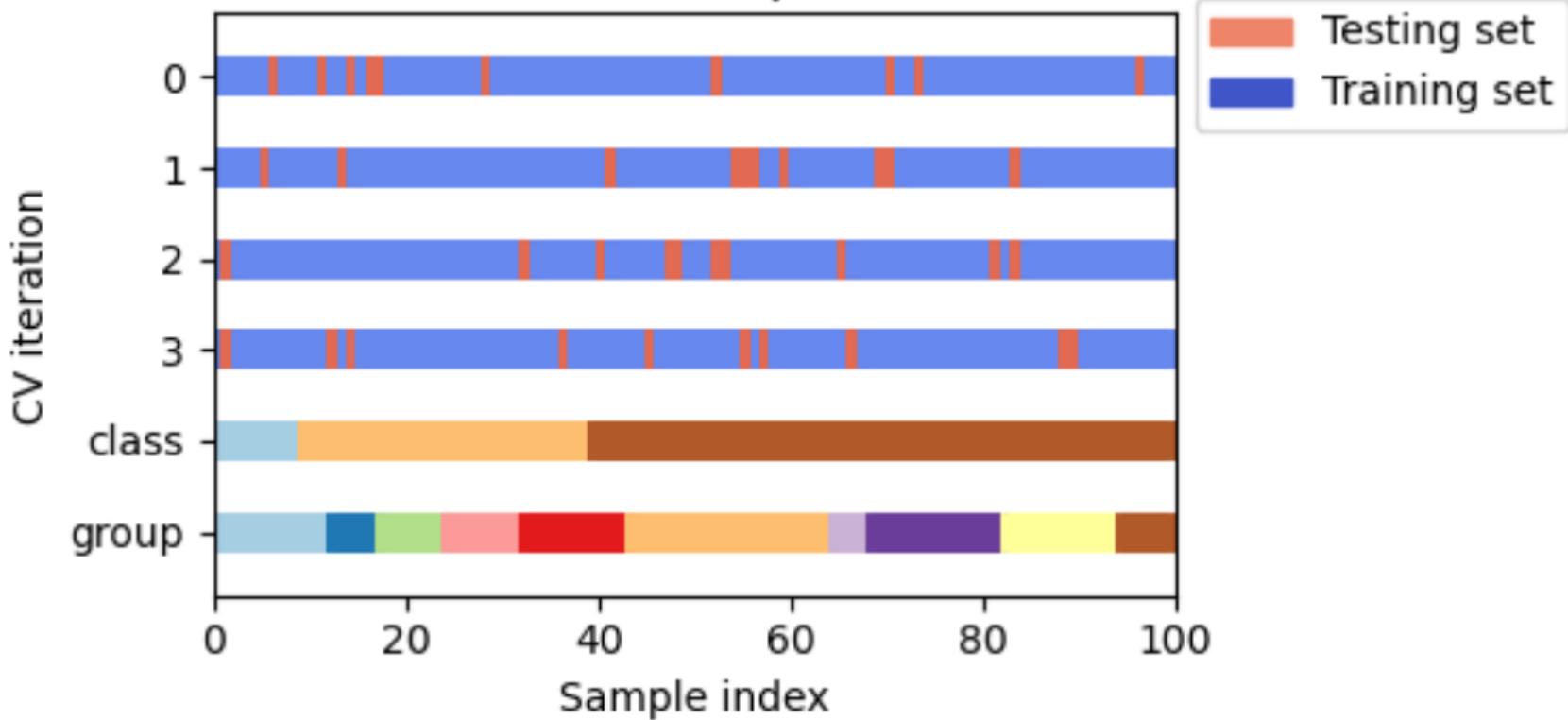
GroupKFold



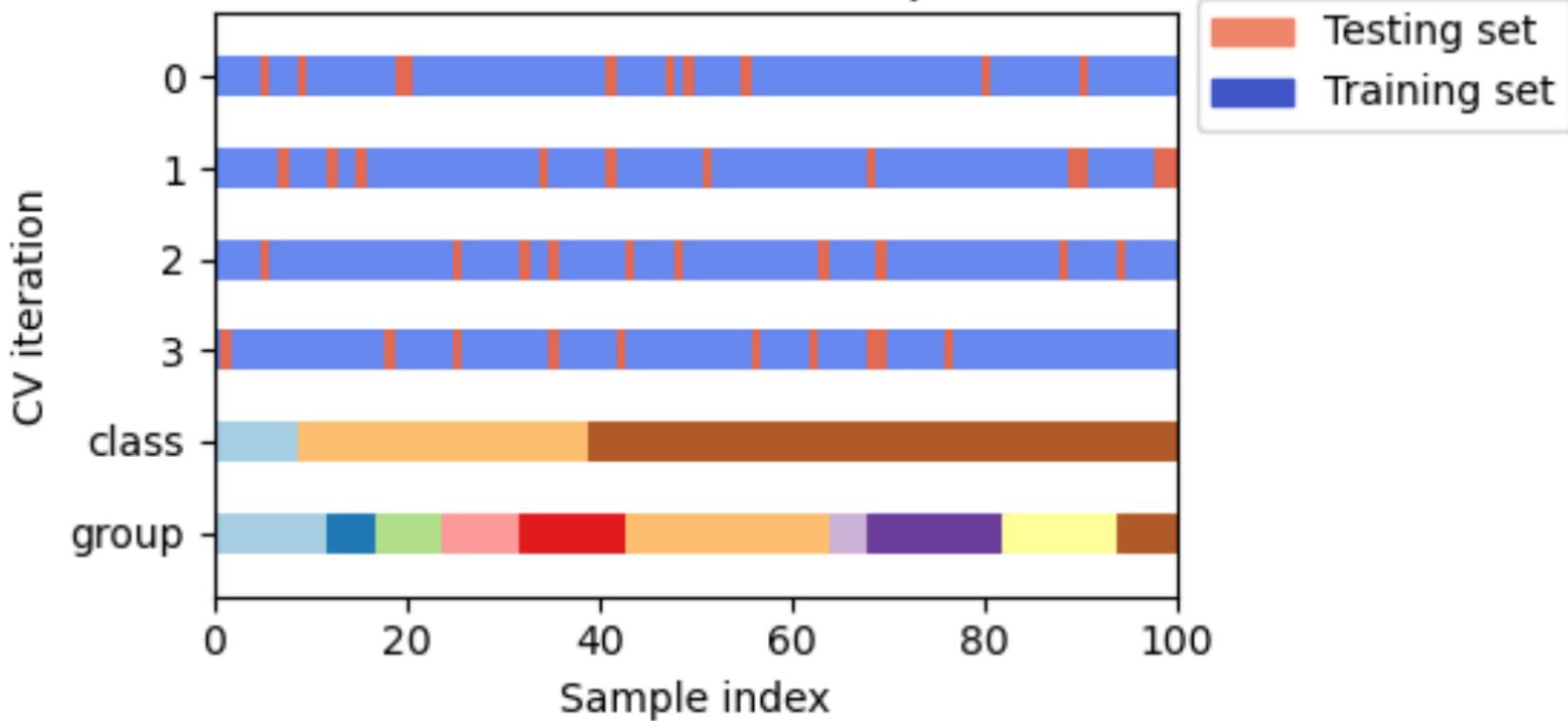
StratifiedGroupKFold



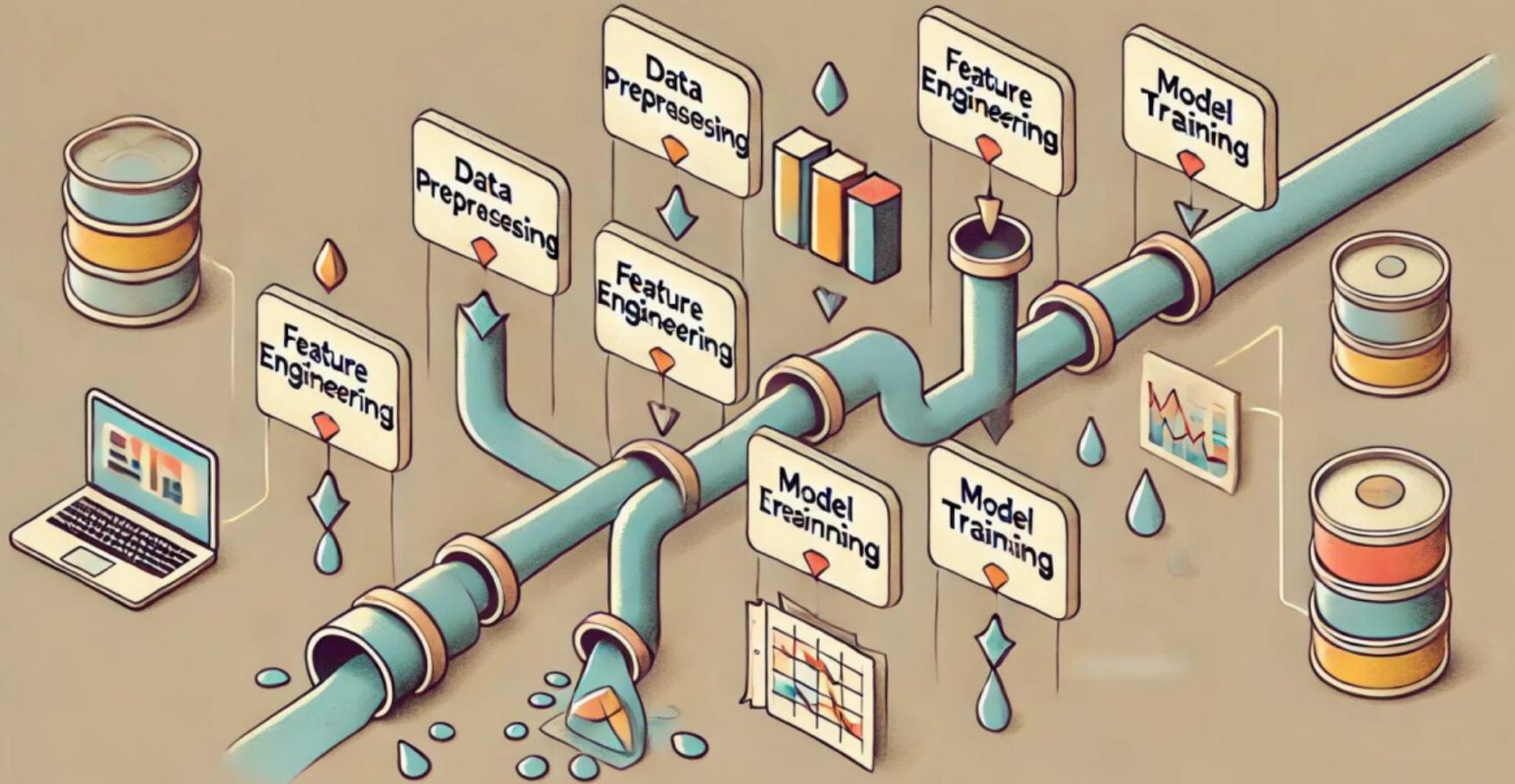
ShuffleSplit



StratifiedShuffleSplit



Data leakage



Data leakage - El enemigo silencioso

¿Qué es data leakage?

- Información del futuro o del test en entrenamiento
- Causa modelos con performance irreal
- Falla catastrófica en producción

Data leakage - El enemigo silencioso

Tipos comunes:

- 1 **Target leakage:** Variable derivada del target
- 2 **Train-test contamination:** Preprocesar antes de dividir
- 3 **Temporal:** Usar futuro para predecir pasado
- 4 **Duplicados:** Mismo ejemplo en train y test

Prevención:

- Pipelines correctos
- Validación temporal
- Auditoría de características
- Feature importance analysis
- Calcular estadísticas *dentro* de cada partición de CV.

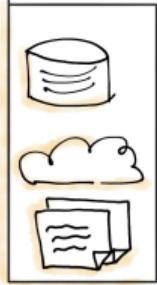


Diagnóstico de *mismatch*

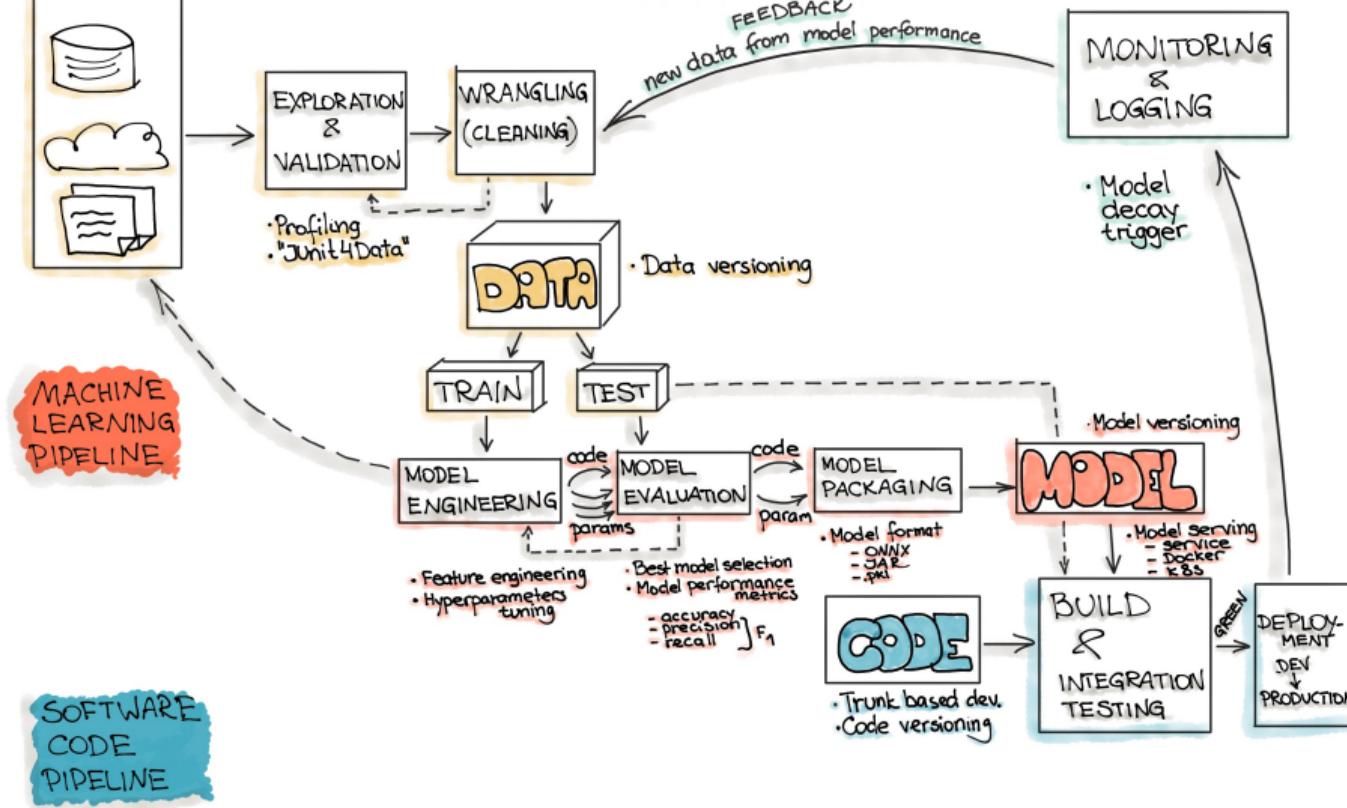
- Mal en train y dev: simplifique, limpie, más datos.
- Bien en train y dev, mal en test: preprocesar como producción.
- Sólo entonces, estimar con prueba final una vez.

Pipelines en scikit-learn

DATA PIPELINE



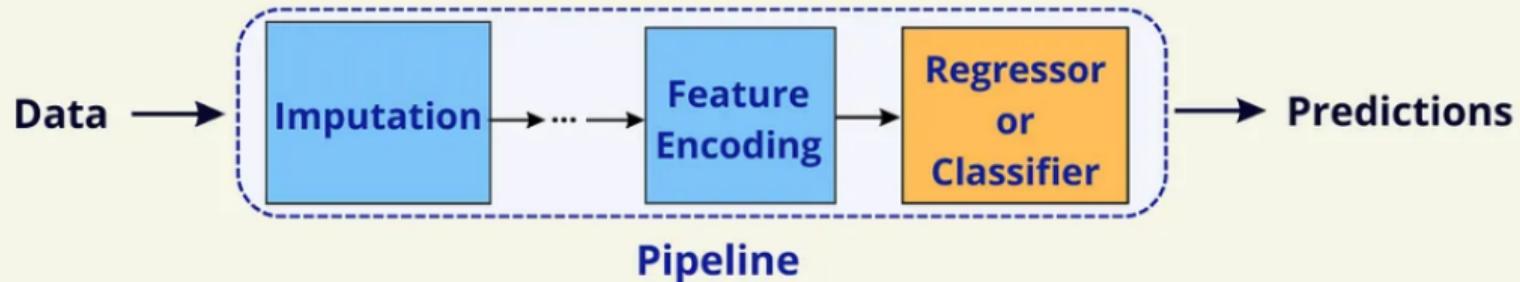
MACHINE LEARNING ENGINEERING.



Pipelines en *scikit-learn*

Pipeline: Secuencia de transformaciones que termina en estimador. Las pipelines son muy comunes en los sistemas de Machine learning, puesto que hay muchos datos que manipular y muchas transformaciones de datos que aplicar.

Simplify Machine Learning Workflow With Scikit-Learn Pipelines



Pipelines en *scikit-learn*

¿Por qué usar pipelines?

- Implementa `.fit()` y `.predict()`.
- Previene data leakage automáticamente: las estadísticas se aprenden sólo en *train*.
- Código más limpio y mantenible
- Facilita deployment
- Reproducibilidad garantizada
- Buenas prácticas: Encapsular, versionar, ser deterministas y trazables.

Ejemplo de Pipeline en Python

```
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import KNeighborsClassifier

knn_pipe = Pipeline([
    ("imputer", SimpleImputer(strategy="median")),
    ("scaler", MinMaxScaler()),
    ("clf", KNeighborsClassifier(n_neighbors=5)),
])
knn_pipe.fit(X_train, y_train)
y_pred = knn_pipe.predict(X_test)
```

Ejemplo de Pipeline en Python

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer

pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler()),
    ('model', RandomForestRegressor())
])

pipeline.fit(X_train, y_train)
predictions = pipeline.predict(X_test)
```

Atajo con make_pipeline

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

pipe = make_pipeline(StandardScaler(),
                     LogisticRegression(max_iter=1000))
pipe.fit(X_train, y_train)
y_proba = pipe.predict_proba(X_new)
```

Transformers comunes

- SimpleImputer, StandardScaler, MinMaxScaler.
- LabelEncoder, OrdinalEncoder, OneHotEncoder.
- CountVectorizer para texto.
- Todos: .fit(), .transform(), .fit_transform().

ColumnTransformer en sklearn

- Aplica transformaciones por subconjunto de columnas.
- Concatena salidas en una sola matriz de rasgos.
- Métodos: `.fit()` y `.transform()`.

ColumnTransformer - Diferentes transformaciones por tipo

```
from sklearn.compose import ColumnTransformer

numeric_features = ['age', 'income']
categorical_features = ['city', 'gender']

preprocessor = ColumnTransformer([
    ('num', numeric_pipeline, numeric_features),
    ('cat', categorical_pipeline, categorical_features)
])

full_pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier())
])
```

Ventaja clave

Cada tipo de dato recibe el tratamiento apropiado

ColumnTransformer + Pipeline (ejemplo)

```
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler, OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.neighbors import KNeighborsClassifier

num_cols, cat_cols = ["feature1", "feature3"], ["feature0", "feature2"]

num_proc = Pipeline([
    ("imp", SimpleImputer(strategy="median")),
    ("scaler", MinMaxScaler())
])
cat_proc = Pipeline([
    ("imp", SimpleImputer(strategy="constant",
                           fill_value="missing")),
    ("ohe", OneHotEncoder(handle_unknown="ignore",
                           sparse_output=False))
])

pre = ColumnTransformer([
    ("num", num_proc, num_cols),
    ("cat", cat_proc, cat_cols)
])

model = Pipeline([
    ("prep", pre),
    ("clf", KNeighborsClassifier(n_neighbors=5))
])

model.fit(X_train, y_train)
y_hat = model.predict(X_test)
```

Bases de datos desbalanceadas

El problema del desbalance de clases

¿Cuándo ocurre?

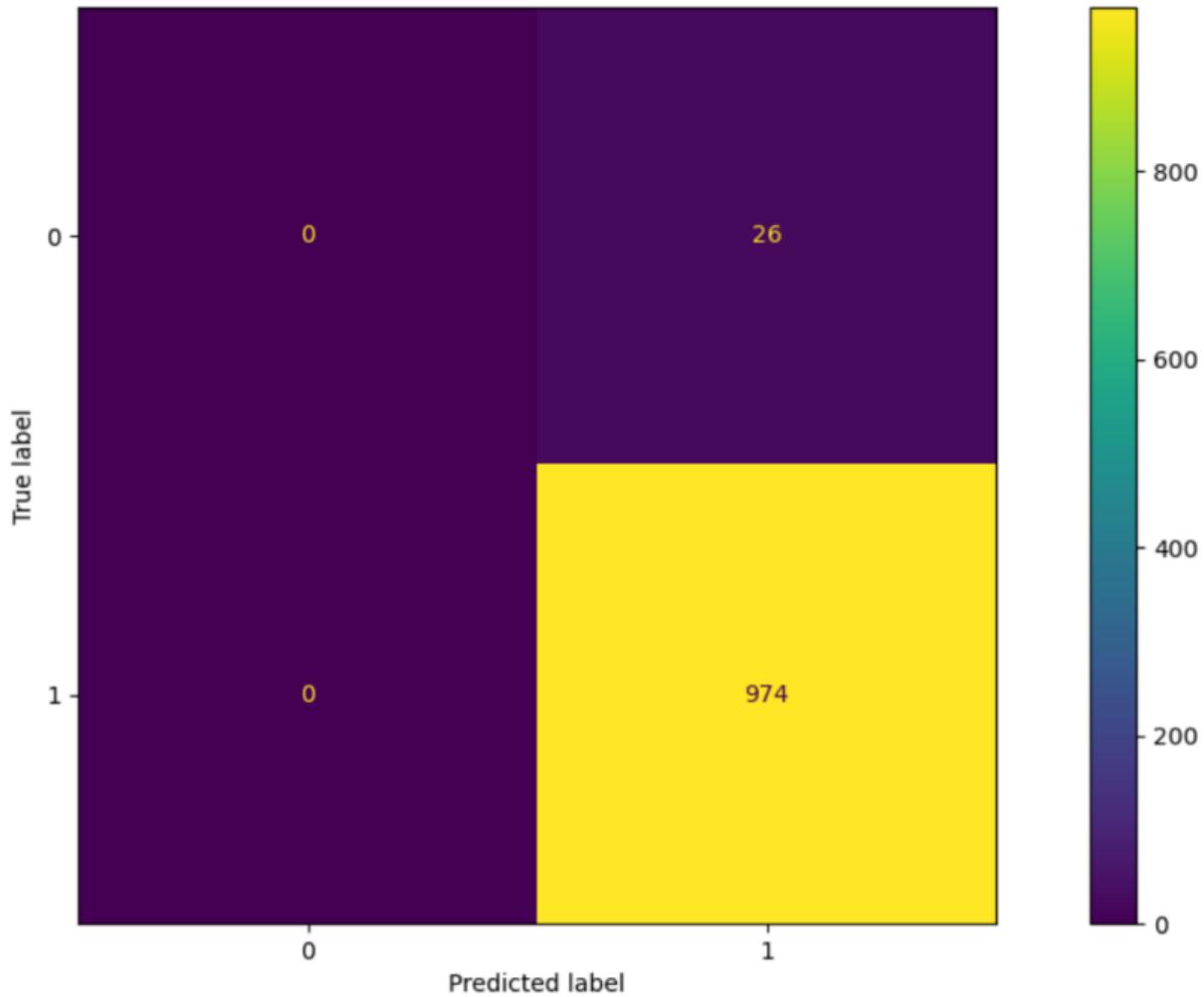
- Detección de fraude: 0.1% positivos
- Diagnóstico médico: enfermedades raras
- Predicción de *churn*: 5% abandonan
- Detección de anomalías

¿Por qué es problemático?

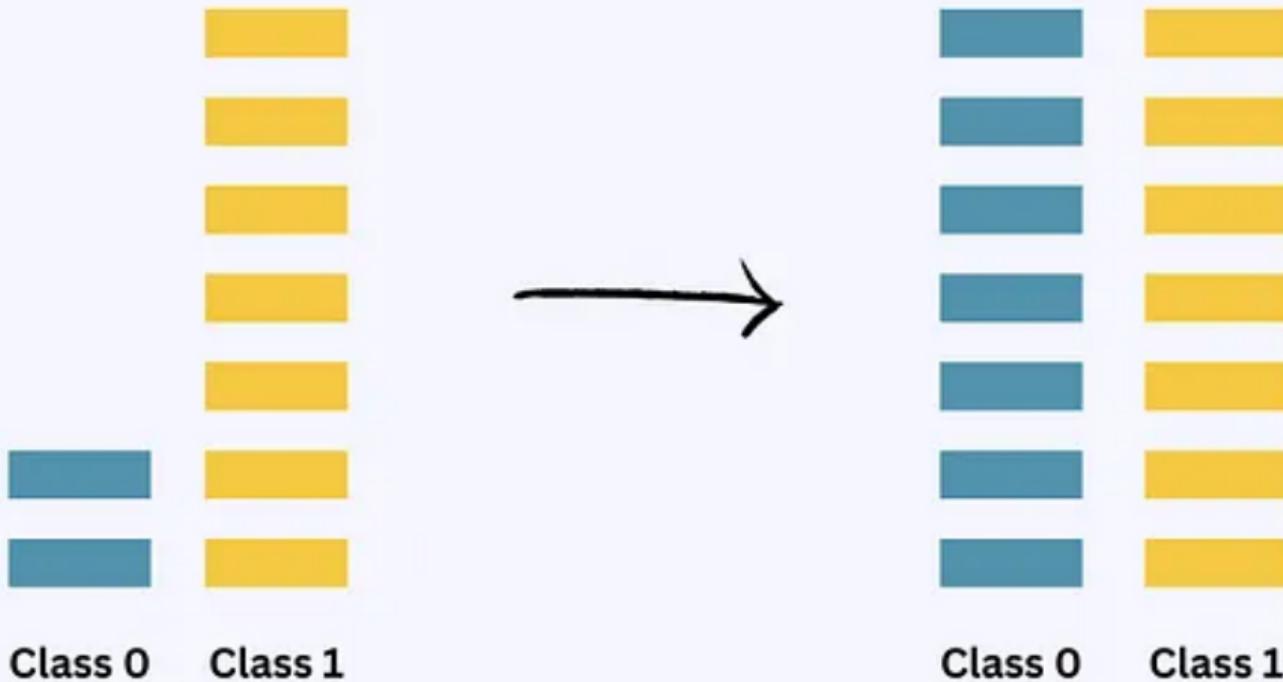
- Modelo trivial (predecir siempre mayoría) obtiene alta accuracy
- Riesgo: ignorar la clase rara.
- Algoritmos sesgados hacia clase mayoritaria
- Métricas engañosas

Ejemplo:

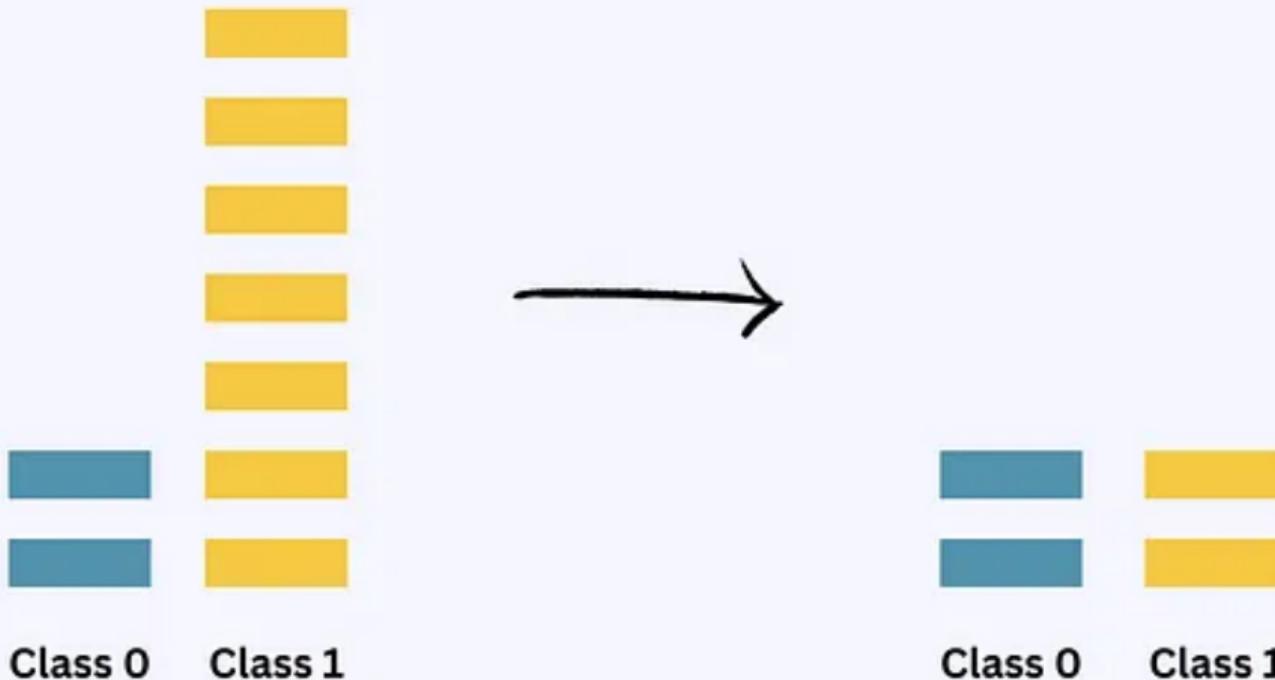
- 99% clase 0, 1% clase 1
- Modelo que siempre predice 0: 99% accuracy
- Pero 0% recall para clase minoritaria
- Ej.: *Amazon reviews*: 5 estrellas domina.



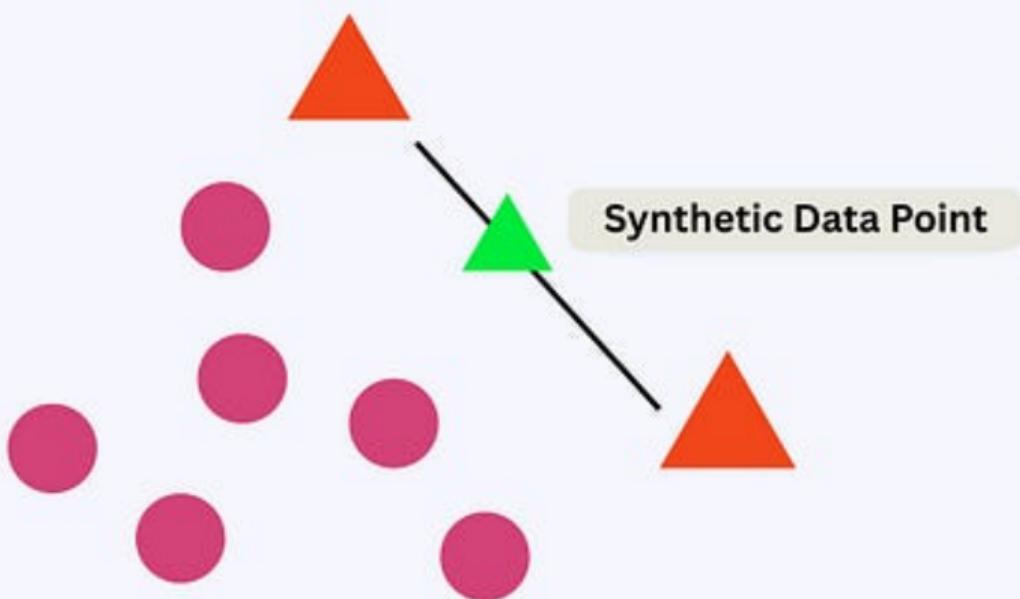
OVERSAMPLING



UNDERSAMPLING



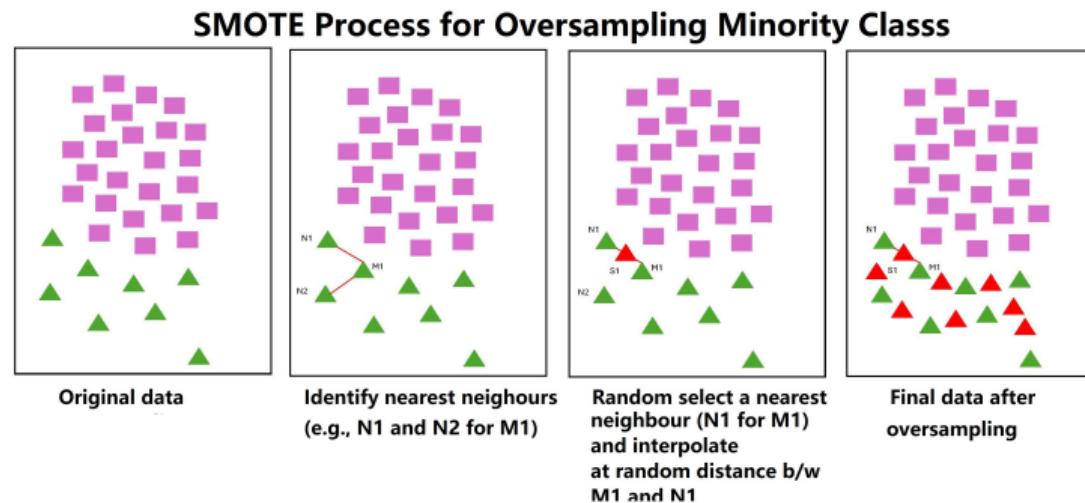
SMOTE



Estrategias para datos desbalanceados

Nivel de datos:

- Ajustes sólo en *train*; dev/test conservan distribución.
- **Undersampling**: Submuestreo (reducción) de la clase mayoritaria.
- **Oversampling**: Remuestreo (aumento) de clases minoritarias.
- Generación sintética
 - **SMOTE**: Synthetic Minority Oversampling
 - **ADASYN**: Adaptive Synthetic Sampling



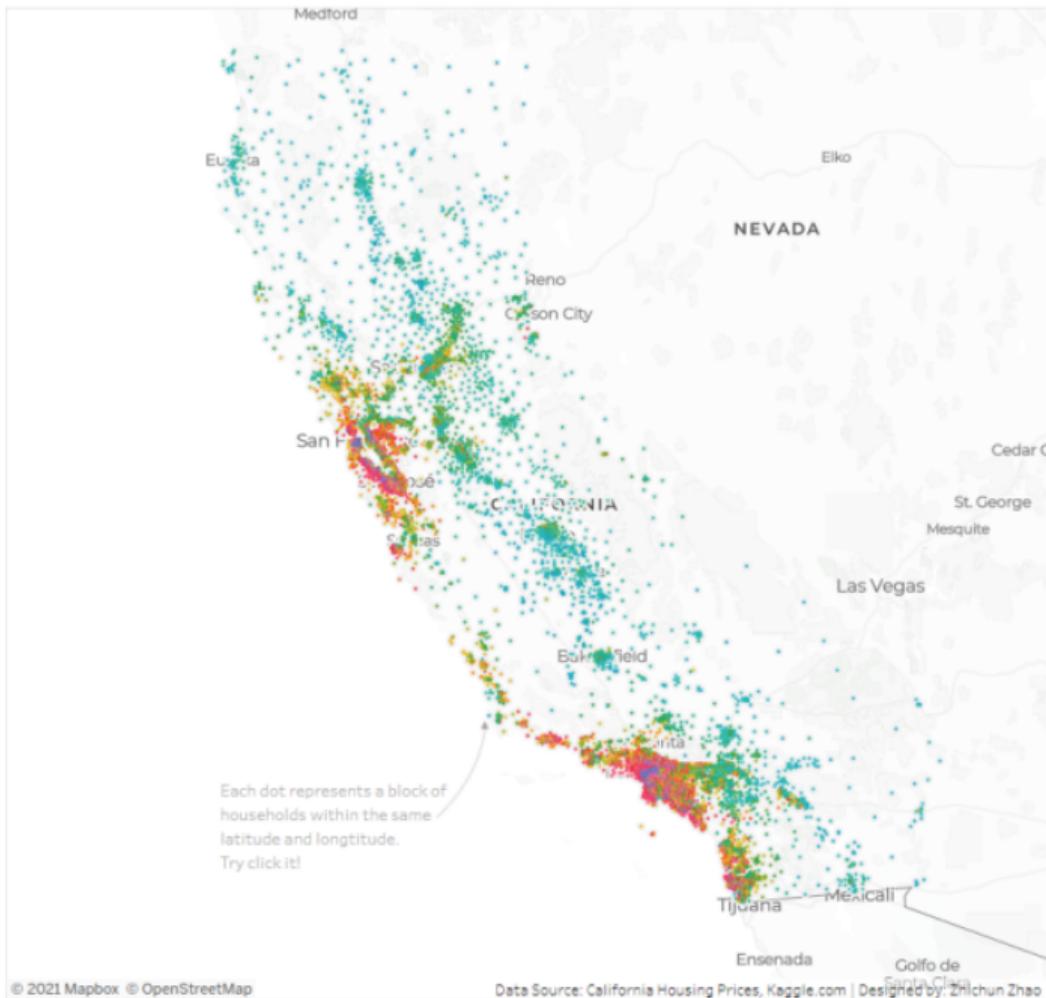
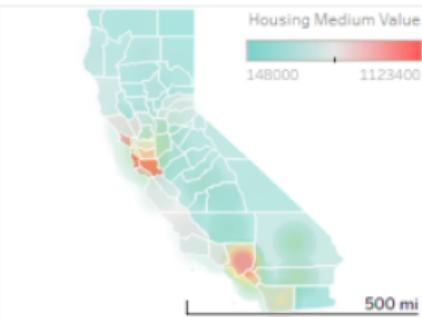
Proyecto integral: California Housing

1990 CALIFORNIA HOUSING PRICE

California, ranked as the first in GDP among all states in the U.S. meanwhile with the most population, has one of the most expensive housing prices. California is attracting not only the us citizens, but also people over the world to reside. As the IT industry keeps blooming in the region, with its appealing weather, the demand for real estate industry is further rising.

1,425 206.86K

Average Population Median House Value



INLAND

California inland has the lowest median house price overall. 53% of the residents live in houses valued from 50k to 100k.



<1H OCEAN

Most people reside in houses less than 1 hour to the ocean. 39% of the residents live in houses valued from 150k to 200k.



NEAR BAY

Least people choose to live near bay. 28% of the residents lives in houses valued from 150k to 200k.



NEAR OCEAN

26% of the near ocean residents dwells in houses valued from 150k to 200k.



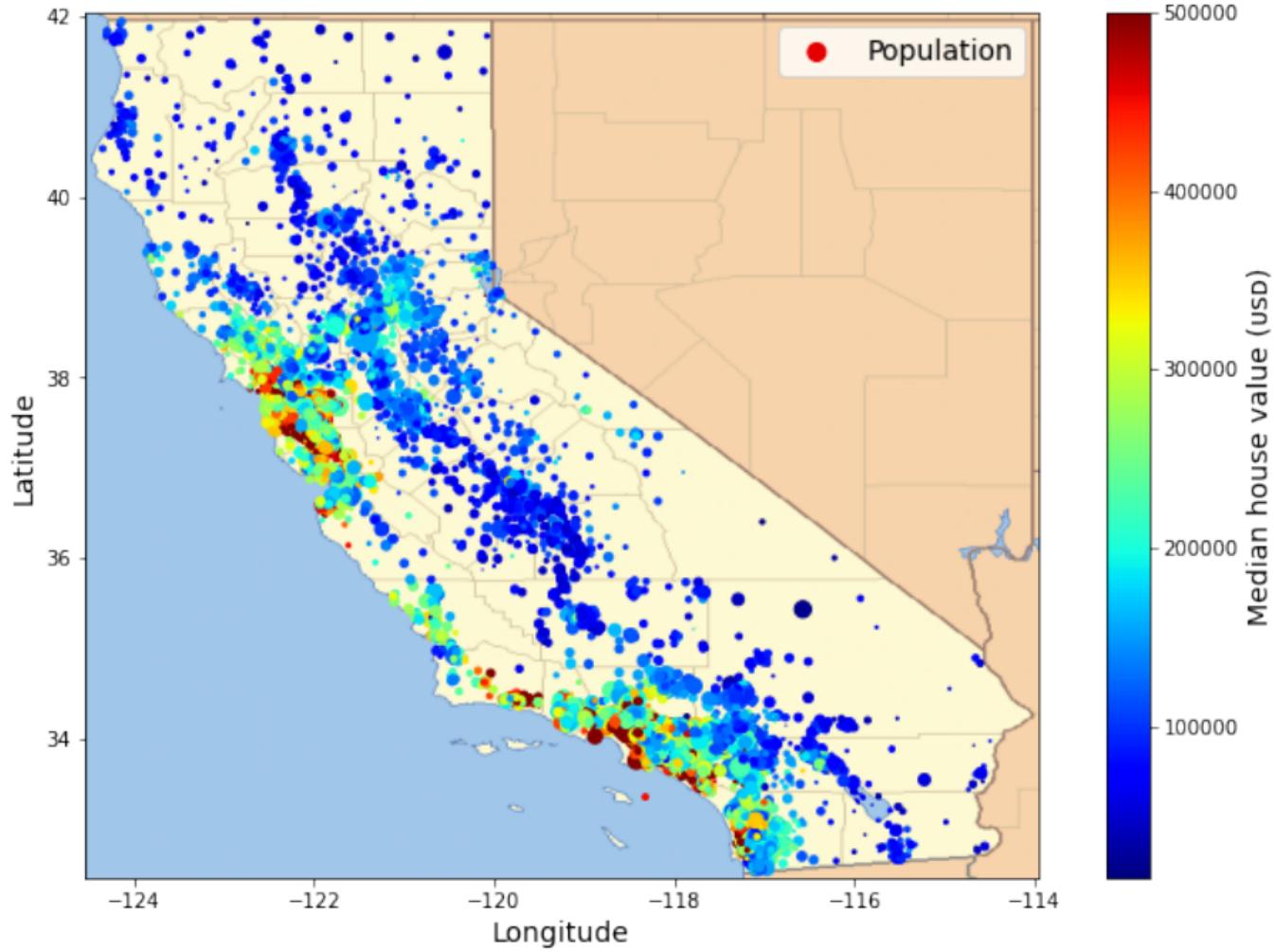
Descripción del proyecto

Objetivo:

- Predecir precio medio de viviendas en California y analizar patrones.
- Dataset del censo de 1990
- 20,640 distritos censales (instancias), 8 atributos demográficos y geográficos.
- Rangos heterogéneos: considerar escalado o transformaciones.

Variables disponibles:

- longitude, latitude: Ubicación
- housing_median_age: Edad media
- total_rooms, total_bedrooms: Tamaño
- population, households: Demografía
- median_income: Ingreso medio
- ocean_proximity: Cercanía al océano
- median_house_value: **TARGET**



Pipeline completo del proyecto

1 Comprensión del negocio:

- Objetivo: Sistema automático de tasación
- Métrica: RMSE < \$50,000

2 EDA inicial:

- Distribuciones, correlaciones
- 207 valores faltantes en total_bedrooms
- Precio censurado en \$500,001

3 Ingeniería de características:

- rooms_per_household
- bedrooms_per_room
- population_per_household

4 Modelado:

- Baseline → Linear → Decision Tree → Random Forest
- Cross-validation para selección

Conclusiones y mejores prácticas

Lecciones clave del capítulo

- 1 CRISP-DM proporciona **estructura**, pero es flexible e iterativo
- 2 EDA es **fundamental**: “Los datos bien explorados están medio modelados”. EDA riguroso antes de cualquier modelado.
- 3 La **calidad de datos determina el techo** de performance del modelo
- 4 Feature engineering es **donde está el valor**: Conocimiento del dominio × creatividad. Rasgos bien diseñados superan modelos exóticos.
- 5 Validación rigurosa es **no negociable**: Sin ella, el fracaso está garantizado. Validación honesta gana a métricas infladas.
- 6 Pipelines previenen errores costosos y facilitan producción. *Pipelines* versionados y monitoreados.
- 7 El contexto de negocio importa más que la métrica técnica
- 8 Iterar con propósito: negocio primero, siempre.

Checklist para proyectos de ML

Antes de empezar:

- ¿Está definido el problema de negocio?
- ¿Hay datos suficientes y de calidad?
- ¿ML es la solución apropiada?

Durante el desarrollo:

- ¿Se realizó EDA exhaustivo?
- ¿Se validó contra data leakage?
- ¿Se usó validación apropiada?
- ¿Se documentaron decisiones?

Antes de producción:

- ¿El modelo cumple objetivos de negocio?
- ¿Hay plan de monitoreo?
- ¿Existe estrategia de actualización?

Errores comunes a evitar

- 1 Empezar por el modelo**, no por el problema
- 2 Ignorar la exploración de datos** y saltar a modelado
- 3 No validar correctamente** (usar test set múltiples veces)
- 4 Data leakage sutil** (preprocesar antes de dividir)
- 5 Optimizar la métrica equivocada** (accuracy en datos desbalanceados)
- 6 Ignorar el costo computacional** en producción
- 7 No documentar** decisiones y supuestos
- 8 Subestimar mantenimiento** post-deployment

Recuerda

Un modelo en producción vale más que diez en desarrollo

Preguntas para reflexión

- 1 ¿Por qué CRISP-DM enfatiza la naturaleza iterativa del proceso?
- 2 ¿Cuándo preferirías estadísticas robustas sobre clásicas?
- 3 ¿Cómo detectarías data leakage en un sistema en producción?
- 4 ¿Qué trade-offs existen entre interpretabilidad y precisión?
- 5 ¿Cómo manejarías un dataset con 50 % de valores faltantes?
- 6 ¿Cuándo NO usarías machine learning para resolver un problema?

Para profundizar

Implementa un proyecto completo siguiendo CRISP-DM con un dataset de tu elección

Referencias y lecturas recomendadas I

Libros fundamentales:

- Tukey, J.W. (1977). *Exploratory Data Analysis*. Addison-Wesley.
- Cleveland, W.S. (1993). *Visualizing Data*. Hobart Press.
- Kuhn, M. & Johnson, K. (2019). *Feature Engineering and Selection*. Chapman & Hall.
- Cover & Thomas, *Elements of Information Theory* (2006).
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. Springer.

Papers clave:

- Chapman et al. (2000). CRISP-DM 1.0: Step-by-step data mining guide.
- Wilkinson, L. (2005). *The Grammar of Graphics*. Springer.
- Rubin, D.B. (1976). Inference and missing data. *Biometrika*, 63(3).

Referencias y lecturas recomendadas II

Recursos online:

- Documentación de scikit-learn sobre preprocesamiento
- Kaggle Learn - Cursos prácticos
- imbalanced-learn - Manejo de desbalance
- Datawig (imputación con NN).

Herramientas recomendadas:

- pandas-profiling: EDA automático
- missingno: Visualización de valores faltantes
- yellowbrick: Visualizaciones para ML
- SHAP: Interpretabilidad de modelos

¡Muchas gracias por su atención!

¿Preguntas?



Contacto: Marco Teran
webpage: marcoteran.github.io/
e-mail: mtteranl@eafit.edu.co