MÓDULO II

Aprendizaje Supervisado

Curso:

Machine Learning Supervisado

🚀 Machine Learning Supervisado

Sesión 01: Ingeniería de ML y Baselines Robustos

De "Correr Celdas" a Construir Software Predictivo

 BLOQUE 0

Conociéndonos

👤 Tu Profesor: Jordan King Rodriguez Mallqui

🎓 Formación Académica

- **Cand. PhD** en Ciencias e Ingeniería Estadística (UNI)
- Ms Data Science (UPC),
- Bch Ingeniería Estadística (UNI)

💼 Experiencia Profesional

- **Senior Manager** Pricing Analytics & Data en **BBVA**
- Lidera equipo de Data Scientists y Data Specialists
- Modelos creados en área comercial, riesgos, finanzas. Sector público y privado

📚 Docencia

- Profesor en **UNI** (Centro de Formación Continua)
- Cursos: Fundamentos de Ciencia de Datos, Power BI, Data Storytelling, Machine Learning

🔬 Intereses de Investigación

- Estimación de precios
- Productos financieros
- Datos georeferenciados

🔗 Conectemos

📧 Contacto

- **Email:** jrodriguezm216@gmail.com
- **Web:** jordandataexpert.com

🌐 Redes Profesionales

- **GitHub:** github.com/JordanKingPeru
- **LinkedIn:** linkedin.com/in/jordan-rodriguez-peru

💡 Filosofía del Curso

> *"No enseño a correr celdas, enseño a construir sistemas de ML que funcionan en producción."*

🎯 Mi Compromiso

- Casos reales de distintas industrias
- Código que puedes usar mañana en tu trabajo
- Feedback continuo y soporte

🙋 ¿Y ustedes?

Cuéntanos en el chat:

1. **Nombre** y empresa/universidad
2. **¿Qué experiencia tienes con ML?** (Ninguna / Básica / Intermedia)
3. **¿Qué problema quieres resolver?** (Clasificación de clientes, predicción de ventas, etc.)

📋 Agenda de Hoy

Bloque	Tema	Duración
1	🗺️ Mapa del Curso, Taxonomía e Intuición ML	25 min
2	💀 Anti-Patterns: La Galería de los Horrores	20 min
3	🏗️ Pipelines de Scikit-Learn	30 min
4A	📈 Regresión Lineal: El Modelo Más Simple	25 min
4B	📊 Regresión Logística: De Valores a Probabilidades	30 min
☕	Break	15 min
5	🕵️ Data Leakage: El Asesino Silencioso	15 min
6	💻 Hands-On: Construyendo el Pipeline	40 min

🗺️ BLOQUE 1

El Mapa del Curso

🎯 ¿Qué aprenderemos en este curso?

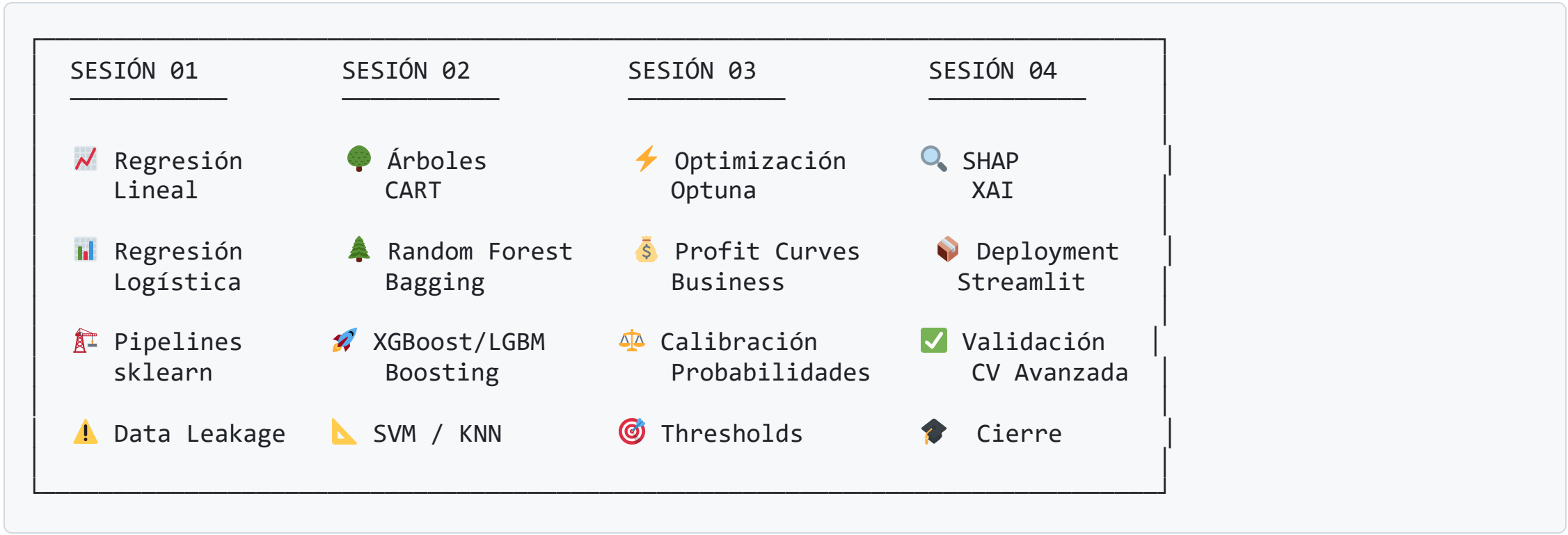
Algoritmos que dominaremos:

- ✅ Regresión Logística (Baseline)
- ✅ Árboles de Decisión (CART)
- ✅ Random Forest (Bagging)
- ✅ **XGBoost / LightGBM** (SOTA)
- ✅ SVM (Casos específicos)
- ✅ KNN (Imputación + Clasificación)
- ✅ Naive Bayes (Texto)
- ✅ Redes Neuronales (Intro)

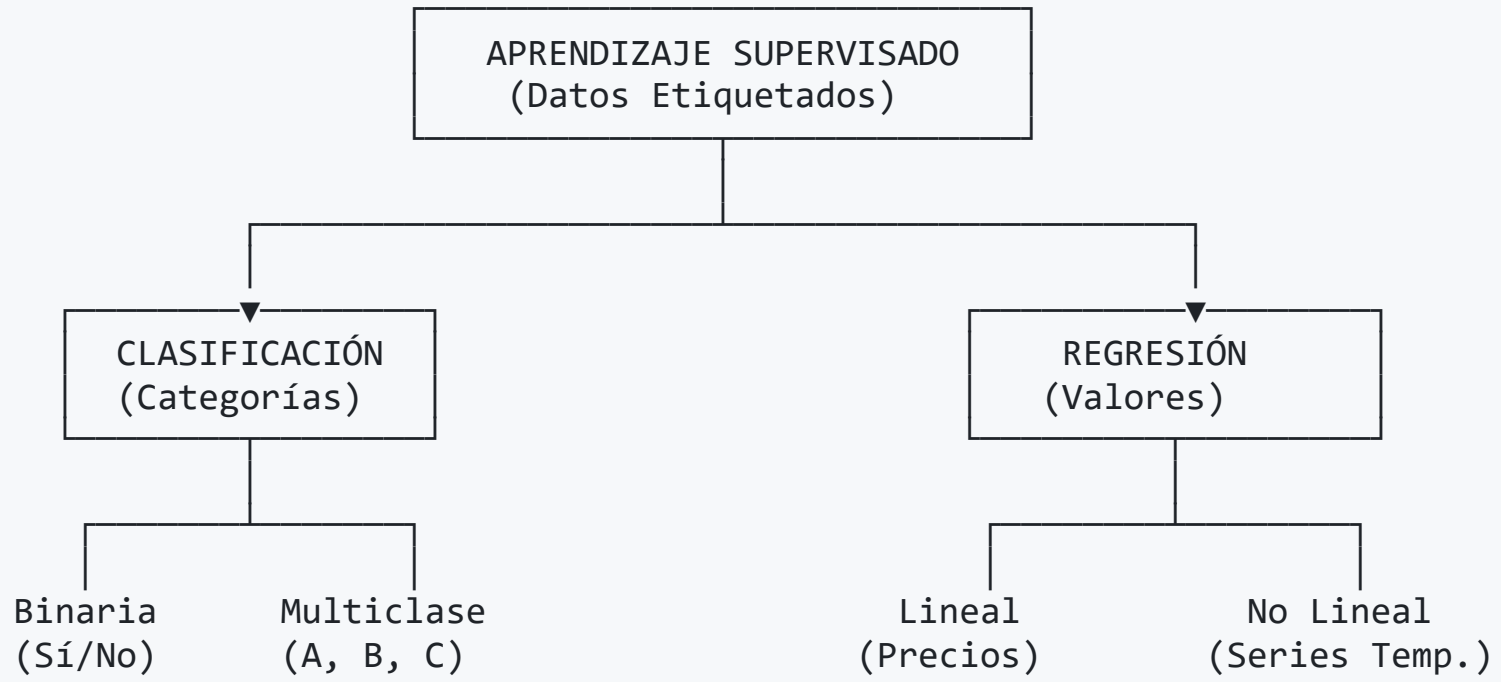
Habilidades de Ingeniero:

- 🔧 Pipelines reproducibles
- 📈 Optimización con Optuna
- 💰 Métricas de negocio (Profit Curves)
- 🔍 Interpretabilidad (SHAP)
- 🚀 Despliegue (Streamlit/FastAPI)

📅 Roadmap del Curso



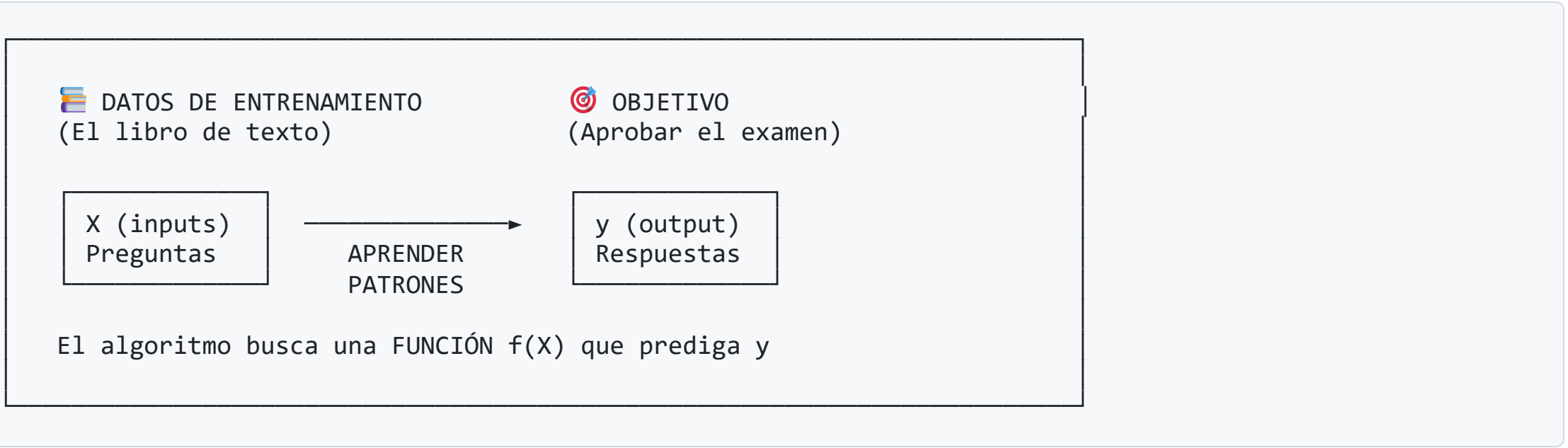
🧭 Taxonomía del Aprendizaje Supervisado



En este curso: Enfoque principal en **Clasificación Binaria** (Riesgo, Fraude, Churn).

🧠 ¿Cómo "Aprende" un Algoritmo de ML?

La Analogía del Estudiante:



> *"El modelo es un estudiante que busca patrones en los datos para responder preguntas nuevas."*

🎯 El Proceso de Aprendizaje: 3 Pasos

1️⃣ Hacer una Predicción

El modelo "adivina" una respuesta inicial.

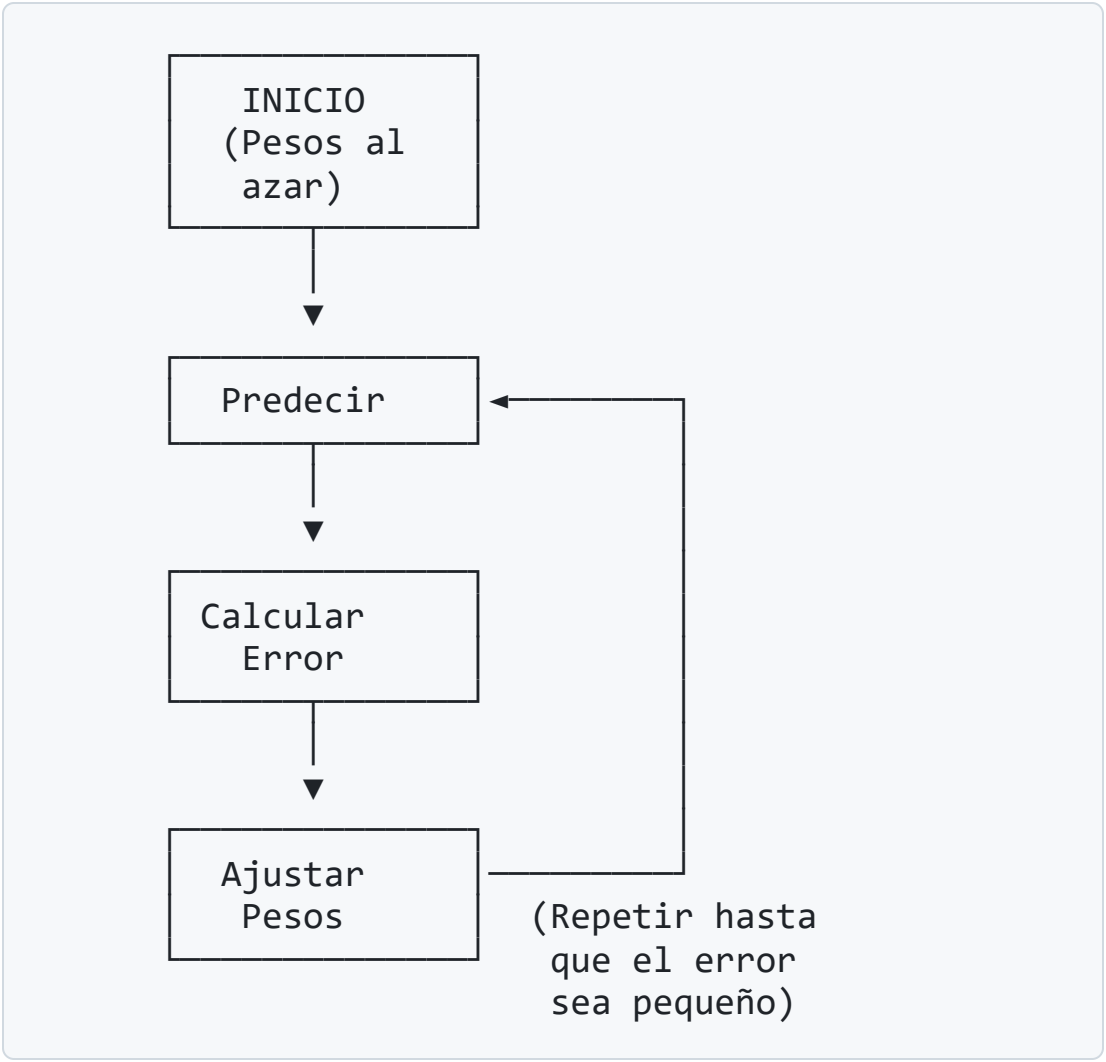
2️⃣ Medir el Error

Comparamos la predicción con la realidad.

$$\text{Error} = \text{Predicción} - \text{Realidad}$$

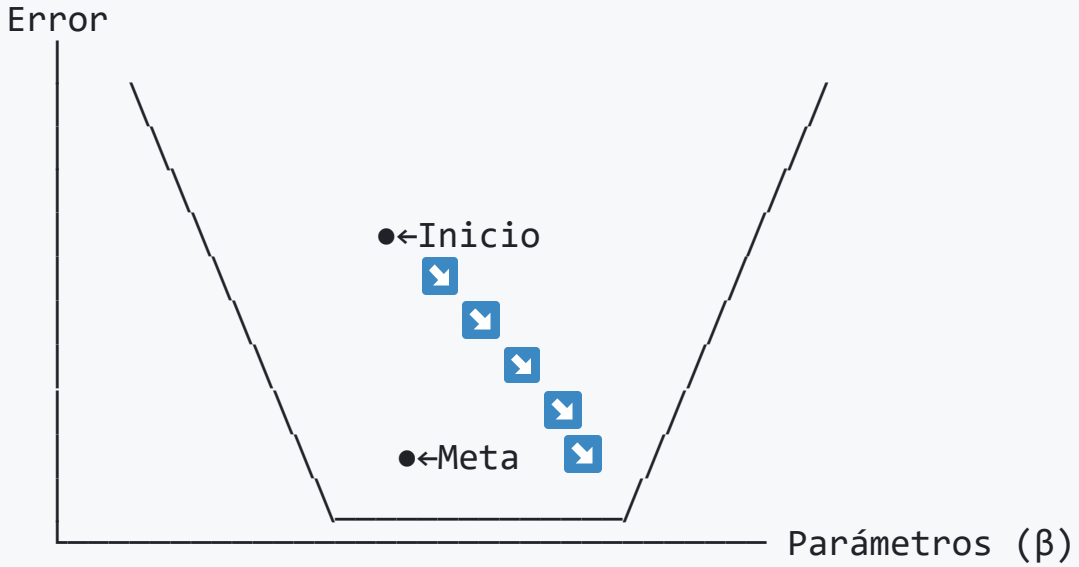
3️⃣ Ajustar los Parámetros

El modelo modifica sus "pesos" para reducir el error.



📉 Intuición: Minimizar el Error

Imagina que buscas el punto más bajo de un valle:



Gradiente Descendente: El algoritmo da "pasitos" hacia abajo hasta encontrar el mínimo.

> *"Cada iteración de entrenamiento es un paso cuesta abajo buscando el error mínimo."*

🔑 Los Parámetros: Lo que el Modelo Aprende

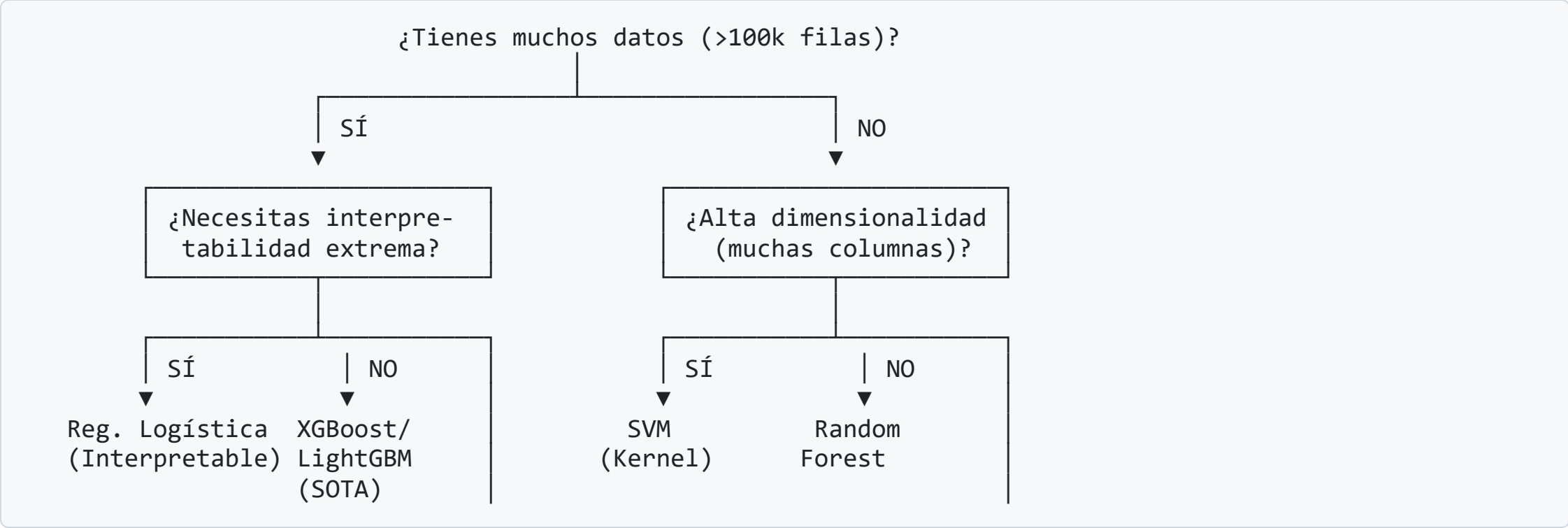
En Regresión Lineal/Logística, los parámetros son los coeficientes (β):

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

Símbolo	Nombre	Qué representa
β_0	**Intercepto**	Valor base cuando todas las X = 0
$\beta_1, \beta_2 \ldots$	**Pesos**	Cuánto influye cada variable
$x_1, x_2 \ldots$	**Features**	Los datos de entrada
$\hat{y}$	**Predicción**	Lo que el modelo predice

El modelo "aprende" los valores óptimos de β que minimizan el error.

🤔 ¿Cuándo usar cada algoritmo?



Regla de Oro: *Empieza siempre con Regresión Logística como Baselie.*

📊 Matriz de Cobertura de Algoritmos

Tema	Nivel de Profundidad	Sesión
Regresión Lineal/Logística	🟢🟢🟢 **Alto** (con Pipelines)	01
Árboles / Random Forest	🟢🟢🟢🟢 **Muy Alto**	02
XGBoost / LightGBM / CatBoost	🟢🟢🟢🟢 **Muy Alto**	02
SVM	🟡🟡 **Medio** (Concepto y uso básico)	02
KNN	🟡🟡 **Medio**	02
Naive Bayes	🟡 **Rápido** (Mención y ejemplo)	02
Redes Neuronales (MLP)	🟡🟡 **Medio**	04
Métricas / Validación	🟢🟢🟢🟢 **Muy Alto** (Business + Tech)	03-04

📚 Bibliografía del Curso (Referencias APA)

Libros Principales:

- Géron, A. (2022). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (3rd ed.). O'Reilly Media.
- Provost, F., & Fawcett, T. (2013). *Data Science for Business*. O'Reilly Media.
- Molnar, C. (2022). *Interpretable Machine Learning* (2nd ed.). Leanpub. https://christophm.github.io/interpretable-ml-book/

Recursos Adicionales:

- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning* (2nd ed.). Springer.
- Thakur, A. (2020). *Approaching (Almost) Any Machine Learning Problem*. Amazon.
- Zheng, A., & Casari, A. (2018). *Feature Engineering for Machine Learning*. O'Reilly Media.

💀 BLOQUE 2

Anti-Patterns: La Galería de los Horrores

❌ El Código "Spaghetti" que Todos Hemos Escrito

```
# 🚨 ANTI-PATTERN: El notebook caótico
df = pd.read_csv('data.csv')
df.fillna(0)  # Sin asignar, no hace nada
df = df.dropna()  # Perdemos datos sin criterio

# Encoding manual que explota en producción
df = pd.get_dummies(df)

# ¿Media de TODO el dataset? 👀 Data Leakage...
df['income'] = df['income'].fillna(df['income'].mean())

# Split al final, después de contaminar
X_train, X_test = train_test_split(df)
```

Pregunta: *¿Qué pasa si llega un cliente nuevo mañana con una categoría que no existía?*

🎯 Los 6 Pecados Capitales del ML

#	Anti-Pattern	Consecuencia
1	**Data Leakage**	Accuracy 99% → 60% en producción
2	**Código Irreproducible**	"En mi máquina funciona..."
3	**Encoding Frágil**	Crash cuando llega categoría nueva
4	**Olvidar Escalar**	Modelo no converge o tarda horas
5	**IDs como Features**	El modelo memoriza índices
6	**Evaluar en Train**	Overfitting disfrazado de éxito

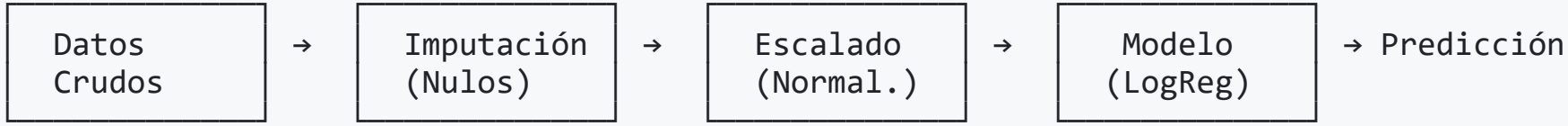
> 📖 **Lectura Recomendada:** Sculley, D. et al. (2015). *Hidden Technical Debt in Machine Learning Systems*. NeurIPS.

🏗 BLOQUE 3

Pipelines de Scikit-Learn

✅ La Solución: Pipelines

> *"Un tubo donde entran datos crudos y sale una predicción."*



Ventajas:

1. **Modular:** Cada paso es intercambiable.
2. **Seguro:** Evita Data Leakage (aprende solo del train).
3. **Serializable:** Se guarda en un solo archivo `.joblib`.
4. **Productivo:** `pipeline.predict(nuevo_cliente)` y listo.

🔧 Componentes del Pipeline

Imputación (Nulos)

```
# Básico (Estadística simple)
SimpleImputer(strategy='median')

# Pro (Vecinos cercanos)
KNNImputer(n_neighbors=5)

# Expert (Iterativo/MICE)
IterativeImputer()
```

Encoding (Categóricas)

```
# Básico (Explosión de columnas)
OneHotEncoder(handle_unknown='ignore')

# Pro (Usa el target)
TargetEncoder(min_samples_leaf=10)

# Expert (Estadístico)
WoEEncoder()  # Weight of Evidence
```

> ⚠️ **Warning:** `OneHotEncoder` con 500 categorías = 500 columnas nuevas.

🎛️ ColumnTransformer: El Router

```python
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

# Rutas diferentes para diferentes tipos de datos
preprocessor = ColumnTransformer([
    ('num', Pipeline([
        ('imputer', SimpleImputer(strategy='median')),
        ('scaler', StandardScaler())
    ]), numeric_features),

    ('cat', Pipeline([
        ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
        ('encoder', TargetEncoder())
    ]), categorical_features)
])

# Pipeline Final
model = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression())
])
```

📈 BLOQUE 4A

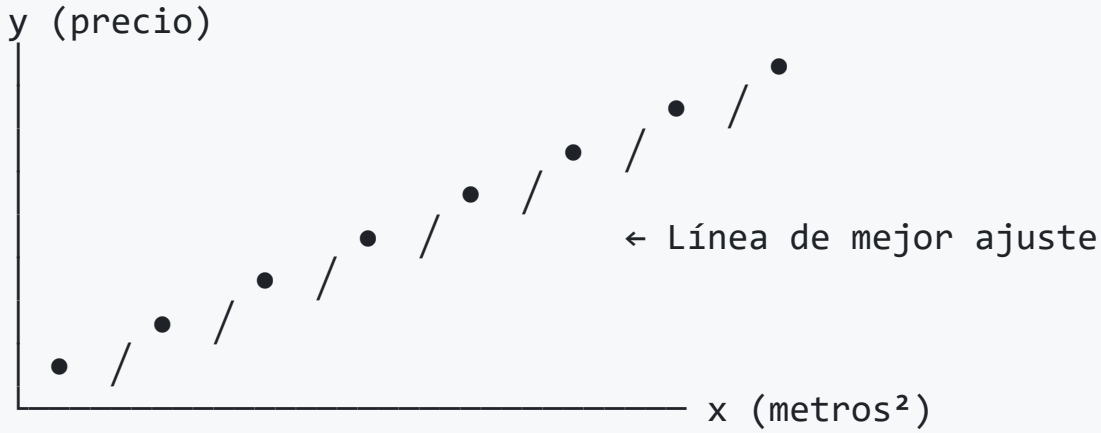
Regresión Lineal: El Modelo Más Simple

🎯 ¿Qué es la Regresión Lineal?

> *"Encontrar la mejor línea recta que pase cerca de todos los puntos."*

El modelo más simple y fundamental de ML supervisado:

$$\hat{y} = \beta_0 + \beta_1 x$$



Objetivo: Predecir un **valor numérico continuo** (precios, temperaturas, ventas).

📐 La Ecuación de la Recta

Ecuación Simple (1 variable):

$$\hat{y} = \beta_0 + \beta_1 x$$

Ecuación Múltiple (n variables):

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

Interpretación de los Parámetros:

Parámetro	Significado
β_0	**Intercepto** - Valor de y cuando x=0
β_1	**Pendiente** - Cuánto cambia y por cada unidad de x

Ejemplo:

$$\text{Precio} = 50{,}000 + 1{,}200 \times \text{metros}^2$$

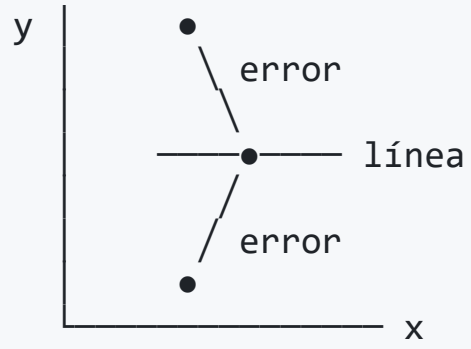
"Una casa de 0 m² cuesta $50,000 (base) y cada m² adicional suma $1,200."

¿Cómo Encuentra la Mejor Línea?

Método de Mínimos Cuadrados Ordinarios (OLS)

Objetivo: Minimizar la suma de los errores al cuadrado.

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



¿Por qué al cuadrado?

1. Elimina signos negativos
2. Penaliza más los errores grandes
3. Tiene solución matemática cerrada (única)

🔧 Regresión Lineal en Scikit-Learn

```python
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Datos
X = df[['metros_cuadrados', 'habitaciones', 'antiguedad']]
y = df['precio']

# Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Entrenar
modelo = LinearRegression()
modelo.fit(X_train, y_train)

# Evaluar
y_pred = modelo.predict(X_test)
print(f"R² Score: {r2_score(y_test, y_pred):.3f}")
print(f"RMSE: {mean_squared_error(y_test, y_pred, squared=False):.2f}")
```

📊 Interpretación de Coeficientes

```python
# Ver los coeficientes aprendidos
print(f"Intercepto (β₀): {modelo.intercept_:,.0f}")
for feature, coef in zip(X.columns, modelo.coef_):
    print(f"{feature}: {coef:,.2f}")
```

Output:

```
Intercepto (β₀): 45,000
metros_cuadrados: 1,250.00   → +$1,250 por cada m² adicional
habitaciones: 8,500.00       → +$8,500 por cada habitación extra
antiguedad: -1,200.00        → -$1,200 por cada año de antigüedad
```

Lectura para el negocio:

> *"Una casa gana $1,250 por cada metro cuadrado, pero pierde $1,200 de valor por cada año de antigüedad."*

📏 Métricas de Evaluación para Regresión

Métrica	Fórmula	Interpretación
MSE	$\frac{1}{n}\sum(y-\hat{y})^2$	Error cuadrático medio
RMSE	$\sqrt{MSE}$	Error en las mismas unidades que y
MAE	$\frac{1}{n}\sum\|y-\hat{y}\|$	
R²	$1-\frac{SS_{res}}{SS_{tot}}$	% de varianza explicada (0-1)

Guía Práctica:

- **R² = 0.85** → El modelo explica el 85% de la variabilidad
- **RMSE = $15,000** → En promedio, nos equivocamos por $15,000
- **R² < 0.5** → El modelo no es muy útil, buscar más features

⚠️ Supuestos de la Regresión Lineal

Para que el modelo sea válido, se asume:

Supuesto	Qué significa	Cómo verificar
Linealidad	Relación lineal entre X e y	Gráfico de dispersión
Homocedasticidad	Varianza del error constante	Residuos vs predicciones
Independencia	Errores no correlacionados	Durbin-Watson test
Normalidad	Errores siguen distribución normal	QQ-plot
No multicolinealidad	Features no muy correlacionadas	VIF < 5

> ⚠️ **En la práctica:** Muchos problemas reales violan estos supuestos. Por eso usamos modelos más robustos como árboles o regularización.

🔄 De Regresión Lineal a Logística

El Problema:

Regresión Lineal	Regresión Logística
Predice **valores continuos**	Predice **probabilidades**
Salida: $-\infty$ a $+\infty$	Salida: 0 a 1
Ejemplo: Precio de casa	Ejemplo: ¿Cliente pagará? (Sí/No)

```
Lineal:     y = β₀ + β₁x                    (puede dar -50 o 150)
                    ↓
Logística:  P(y=1) = σ(β₀ + β₁x)            (siempre entre 0 y 1)
```

La función σ (sigmoide) "aplasta" la salida al rango [0, 1].

📊 BLOQUE 4B

Regresión Logística: De Valores a Probabilidades

📜 Historia y Contexto

Orígenes:

- **1838:** Pierre-François Verhulst introduce la curva logística para modelar crecimiento poblacional.
- **1958:** David Cox formaliza la Regresión Logística para clasificación binaria.
- **Hoy:** Sigue siendo el **estándar de oro** en banca, salud y seguros por su interpretabilidad.

¿Por qué sigue vigente?

- ✅ **Regulaciones (Basilea III, FDA):** Exigen modelos explicables.
- ✅ **Baseline imbatible:** Si tu modelo complejo no gana al LogReg, algo está mal.
- ✅ **Probabilidades calibradas:** Directamente interpretables.

> 📖 Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied Logistic Regression* (3rd ed.). Wiley.

🧮 Fundamento Matemático

El Problema:

Queremos predecir la **probabilidad** de un evento binario (Sí/No).
Pero las probabilidades están entre 0 y 1, y la regresión lineal no respeta esos límites.

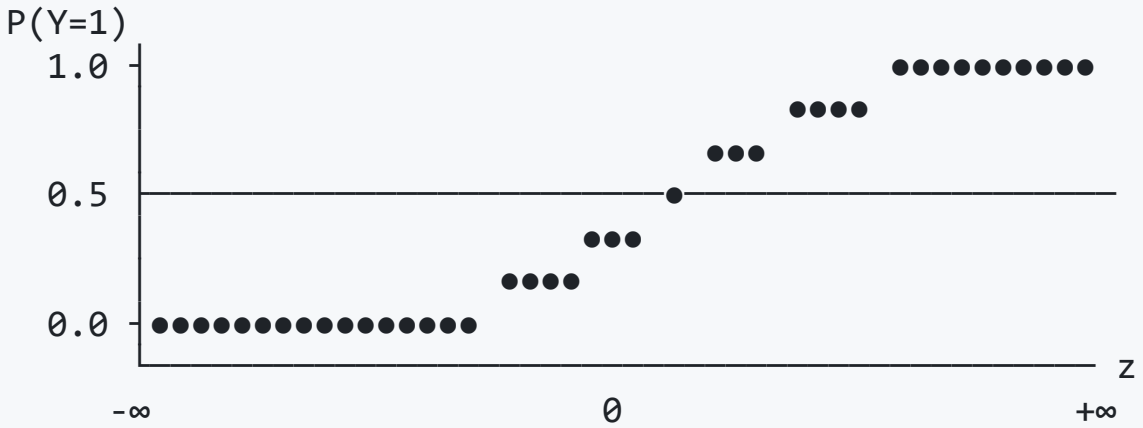
La Solución: La Función Sigmoide (Logística)

$$P(Y = 1|X) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

Donde z es la **combinación lineal** de las variables:

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n$$

📈 Visualización de la Sigmoide



Interpretación:

- Cuando $z \to -\infty$: $P \to 0$ (evento muy improbable)
- Cuando $z = 0$: $P = 0.5$ (punto de decisión por defecto)
- Cuando $z \to +\infty$: $P \to 1$ (evento casi seguro)

🎲 Odds y Log-Odds (Logit)

Odds (Momios):

$$\text{Odds} = \frac{P(Y=1)}{P(Y=0)} = \frac{P}{1-P}$$

Ejemplo: Si P=0.8, Odds = 0.8/0.2 = 4 → "4 a 1 a favor"

Log-Odds (Logit):

$$\text{logit}(P) = \ln\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n$$

¡Esta es la ecuación clave! El modelo aprende los β que mejor ajustan los datos.

🔢 ¿Por qué usamos Log-Odds?

Probabilidad (P)	Odds (P/(1-P))	Log-Odds (ln)
0.01	0.01	**-4.60**
0.10	0.11	**-2.20**
0.50	1.00	**0.00**
0.90	9.00	**+2.20**
0.99	99.0	**+4.60**

Ventaja: El Log-Odds va de $-\infty$ a $+\infty$, ¡igual que una recta de regresión lineal!

Esto permite modelar probabilidades con una ecuación lineal.

Interpretación de Coeficientes (β)

Odds Ratio (OR):

$$\text{OR} = e^{\beta}$$

Coeficiente (β)	Odds Ratio (e^{β})	Interpretación
$\beta = 0$	OR = 1.0	Variable **sin efecto**
$\beta = 0.5$	OR = 1.65	↑ **65%** en odds por unidad
$\beta = -0.3$	OR = 0.74	↓ **26%** en odds por unidad
$\beta = 1.0$	OR = 2.72	↑ **172%** en odds por unidad

Lectura: "*Por cada unidad de aumento en X, los odds del evento se multiplican por OR.*"

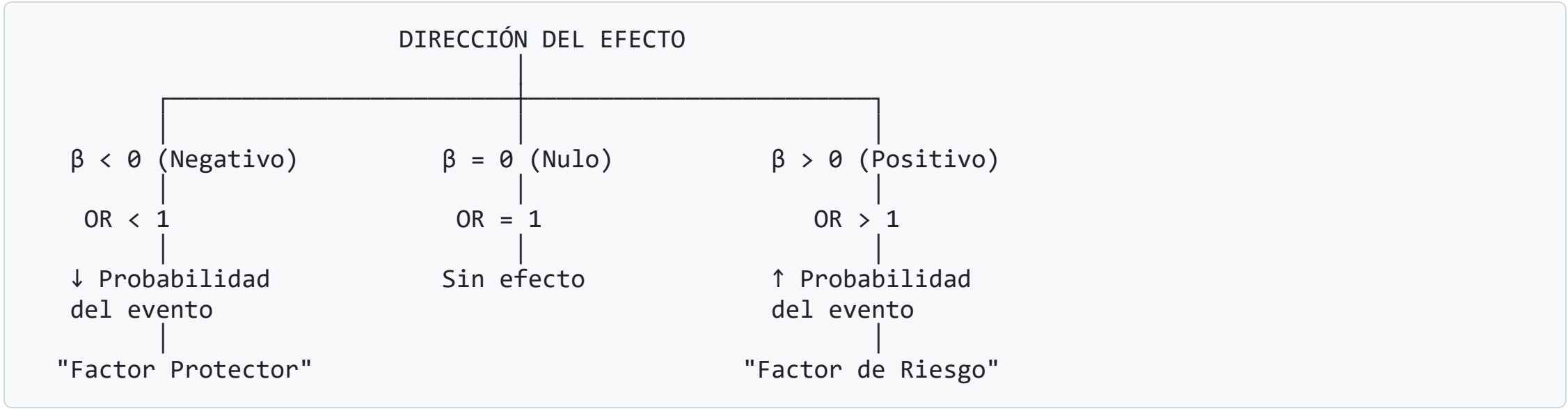
🎯 Ejemplo de Interpretación Real: Credit Scoring

Variable	β	OR (e^β)	Interpretación
Intercepto	-2.30	0.10	Odds base (muy baja prob.)
Edad (años)	-0.02	0.98	↓ 2% odds por año más
Deuda/Ingreso	1.50	4.48	↑ 348% por unidad ratio
Tiene_Hipoteca	-0.40	0.67	↓ 33% vs no tener hipoteca
Meses_Último_Atraso	0.08	1.08	↑ 8% por mes de atraso

Lectura para el negocio:

> *"Un cliente con ratio Deuda/Ingreso alto tiene **4.5 veces más probabilidad** de caer en default que uno con ratio bajo, manteniendo todo lo demás constante."*

🔍 Dirección e Impacto de los Coeficientes



Ejemplo: En credit scoring, **edad** suele tener $\beta < 0$ (protector), mientras que **número de atrasos** tiene $\beta > 0$ (riesgo).

⚖️ Regularización: Ridge, Lasso y ElasticNet

¿Por qué regularizar?

- **Multicolinealidad:** Variables correlacionadas inflan los coeficientes.
- **Overfitting:** Muchas variables → modelo memoriza ruido.

Tipos de Regularización:

Tipo	Fórmula	Efecto	Uso Típico
Ridge (L2)	$\lambda \sum \beta_j^2$	Encoge coeficientes	Multicolinealidad
Lasso (L1)	$\lambda \sum \lvert\beta_j\rvert$	Lleva β a **cero**	Selección automática
ElasticNet	L1 + L2	Combinación	Mejor de ambos mundos

```
LogisticRegression(penalty='l2', C=1.0)  # C = 1/λ (menor C = más regularización)
```

📉 Función de Costo: Log-Loss (Cross-Entropy)

El modelo encuentra los β que **minimizan** el error de predicción:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{p}_i) + (1-y_i)\log(1-\hat{p}_i)]$$

Interpretación intuitiva:

Predicción ($\hat{p}$)	Realidad (y)	Penalización
0.9	1	Baja ✅
0.9	0	**Alta** ❌
0.1	0	Baja ✅
0.1	1	**Alta** ❌

> El modelo aprende a dar alta probabilidad cuando $y = 1$ y baja cuando $y = 0$.

🔧 Hiperparámetros Clave en Scikit-Learn

```python
from sklearn.linear_model import LogisticRegression

model = LogisticRegression(
    penalty='l2',           # Regularización: 'l1', 'l2', 'elasticnet', 'none'
    C=1.0,                  # Inverso de λ (menor C = más regularización)
    class_weight='balanced', # Manejo de desbalance automático
    solver='liblinear',     # Optimizador: 'lbfgs', 'liblinear', 'saga'
    max_iter=1000,          # Iteraciones máximas para convergencia
    random_state=42         # Reproducibilidad
)
```

⚠️ Consejos Prácticos:

- **Desbalance de clases:** Usa `class_weight='balanced'` o SMOTE
- **Muchas variables:** Usa `penalty='l1'` para selección automática
- **No converge:** Aumenta `max_iter` o **escala los datos primero**

📊 Métricas de Evaluación

Matriz de Confusión:

	Predicción 0 (No)	Predicción 1 (Sí)	
Real: 0	TN	FP	← Especificidad = TN/(TN+FP)
Real: 1	FN	TP	← Sensibilidad = TP/(TP+FN)

Métricas Derivadas:

- **Accuracy:** $(TP + TN)/N$ → ⚠️ **Engañosa en desbalance**
- **Precision:** $TP/(TP + FP)$ → "De los positivos predichos, ¿cuántos acerté?"
- **Recall:** $TP/(TP + FN)$ → "De los positivos reales, ¿cuántos detecté?"
- **ROC-AUC:** Área bajo la curva ROC (independiente del umbral)

🤔 ¿Cuándo usar Regresión Logística?

✅ Casos Ideales:

- Necesitas **explicar** las decisiones (regulación, auditoría)
- Tienes **relaciones aproximadamente lineales** con el log-odds
- Quieres un **baseline sólido** antes de probar modelos complejos
- Datos con **pocas no-linealidades**

❌ Evitar cuando:

- Relaciones muy complejas/no-lineales
- Muchas interacciones entre variables
- Imágenes, texto, series temporales complejas
- Necesitas el máximo rendimiento predictivo (usa XGBoost)

📚 Para Profundizar: Lecturas Recomendadas

Teoría Matemática:

- Agresti, A. (2013). *Categorical Data Analysis* (3rd ed.). Wiley.
- McCullagh, P., & Nelder, J. A. (1989). *Generalized Linear Models* (2nd ed.). Chapman & Hall.

Aplicaciones en Riesgo Crediticio:

- Siddiqi, N. (2017). *Intelligent Credit Scoring* (2nd ed.). Wiley.
- Thomas, L. C. (2009). *Consumer Credit Models*. Oxford University Press.

Papers Fundacionales:

- Cox, D. R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society, Series B*, 20(2), 215–242.

BREAK

15 minutos

🕵️ BLOQUE 5

Data Leakage: El Asesino Silencioso

🚨 ¿Qué es Data Leakage?

> *"Cuando el modelo 'hace trampa' usando información que no estaría disponible en el momento de la predicción real."*

Tipos de Leakage:

Tipo	Ejemplo	Consecuencia
Target Leakage	Usar "Estado del Préstamo" para predecir default	El target está codificado en los features
Train-Test Contamination	Calcular la media con todo el dataset	El modelo "ve" el futuro
Temporal Leakage	Usar datos de Julio para predecir Junio	Información del futuro

🔍 Síntomas de Data Leakage

🚩 Red Flags que deberían preocuparte:

1. **Accuracy sospechosamente alto** (>95% en problemas reales)
2. **Una variable domina** (Feature Importance > 80%)
3. **El modelo es "perfecto" en test pero falla en producción**
4. **La variable parece "demasiado predictiva"**

Ejemplo Real en Credit Scoring:

```
Variable: "dias_desde_ultimo_pago"
Valor para clientes morosos: 90+
Valor para clientes buenos: 0-30

🤔 ¿Esta variable existía ANTES de que el cliente cayera en mora?
   ¡NO! El atraso ES la mora. Es Target Leakage.
```

🛡️ Cómo Prevenir el Leakage

Reglas de Oro:

1. Split PRIMERO, procesa DESPUÉS:

```
# ✅ CORRECTO
X_train, X_test = train_test_split(X, y)
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

2. Usa Pipelines (encapsulan el fit solo en train)

3. Pregúntate: "*¿Esta variable existiría en el momento de la decisión?*"

4. Validación Temporal para datos con fechas (TimeSeriesSplit)

💡 Checklist Anti-Leakage

- [ ] ¿Hice el split ANTES de cualquier preprocesamiento?
- [ ] ¿Cada variable existiría en el momento de predicción real?
- [ ] ¿Ninguna variable está derivada del target?
- [ ] ¿Mi accuracy es "demasiado bueno para ser verdad"?
- [ ] ¿Usé Pipeline de scikit-learn para encapsular el fit?
- [ ] ¿Validé con datos temporalmente posteriores (si aplica)?

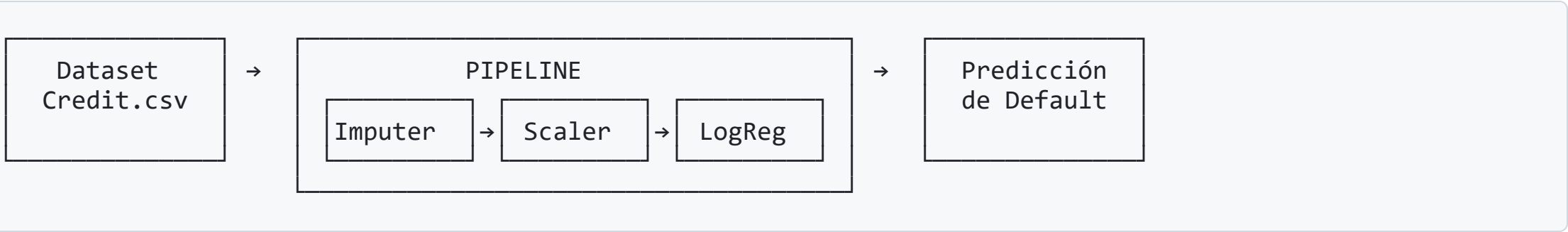
> 📖 Kaufman, S., Rosset, S., & Perlich, C. (2012). Leakage in data mining: Formulation, detection, and avoidance. *ACM TKDD*, 6(4), 1–21.

💻 BLOQUE 6

Hands-On: Construyendo el Pipeline

🎯 Objetivo del Taller

Construir un Pipeline completo para **Credit Scoring**:



Notebooks disponibles:

- 📘 `02_Pipelines_y_Baselines.ipynb` → Versión Profesor (Completa)
- 📕 `02_Pipelines_y_Baselines_Alumnos.ipynb` → Versión Alumnos (Ejercicios)

📝 Checklist del Pipeline Completo

- [ ] **1. Cargar datos** con `load_data()` centralizado
- [ ] **2. Definir constantes** (`TARGET_COL` , `COLS_TO_DROP` , `RANDOM_STATE`)
- [ ] **3. Split estratificado** ANTES de tocar los datos
- [ ] **4. Identificar tipos** (numéricas vs categóricas automáticamente)
- [ ] **5. Pipeline numérico** (Imputer + Scaler)
- [ ] **6. Pipeline categórico** (Imputer + Encoder)
- [ ] **7. ColumnTransformer** (Unir ambos pipelines)
- [ ] **8. Pipeline final** (Preprocessor + Modelo)
- [ ] **9. Entrenar y evaluar** en datos de TEST
- [ ] **10. Serializar** con `joblib.dump()`

🏆 Resumen de la Sesión

Hoy aprendimos:

1. ✅ **El mapa completo** del curso y la taxonomía del ML Supervisado
2. ✅ **Los 6 Anti-Patterns** que destruyen modelos en producción
3. ✅ **Pipelines de Scikit-Learn** para código robusto y reproducible
4. ✅ **Regresión Logística en profundidad** (Odds Ratios, Interpretación, Regularización)
5. ✅ **Data Leakage** y cómo prevenirlo

Próxima sesión (Sesión 02):

🌳 **Árboles de Decisión, Random Forest y Gradient Boosting (XGBoost/LightGBM)**

📚 Referencias Completas

- Agresti, A. (2013). *Categorical Data Analysis* (3rd ed.). Wiley.
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
- Burkov, A. (2020). *Machine Learning Engineering*. True Positive Inc.
- Cox, D. R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society, Series B*, 20(2), 215–242.
- Géron, A. (2022). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (3rd ed.). O'Reilly Media.
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning* (2nd ed.). Springer.
- Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied Logistic Regression* (3rd ed.). Wiley.
- Kaufman, S., Rosset, S., & Perlich, C. (2012). Leakage in data mining. *ACM TKDD*, 6(4), 1–21.
- Molnar, C. (2022). *Interpretable Machine Learning* (2nd ed.). Leanpub.
- Provost, F., & Fawcett, T. (2013). *Data Science for Business*. O'Reilly Media.
- Sculley, D. et al. (2015). Hidden technical debt in ML systems. *NeurIPS*, 28.
- Siddiqi, N. (2017). *Intelligent Credit Scoring* (2nd ed.). Wiley.
- Thakur, A. (2020). *Approaching (Almost) Any Machine Learning Problem*. Amazon.
- Zheng, A., & Casari, A. (2018). *Feature Engineering for Machine Learning*. O'Reilly Media.

🙋 ¿Preguntas?

📧 jrodriguezm216@gmail.com

🌐 jordandataexpert.com

💻 github.com/JordanKingPeru

💻 ¡Vamos al Código!

Abrir: `notebooks/02_Pipelines_y_Baselines_Alumnos.ipynb`