

# MÓDULO 18: GLOSARIO COMPLETO MLOps

## Diccionario Exhaustivo de A-Z con Explicaciones Profundas y Analogías

### Guía MLOps v5.0: Senior Edition | DuqueOM | Noviembre 2025

# MÓDULO 18: Glosario Completo MLOps

## Diccionario Exhaustivo con Explicaciones Profundas y Analogías

“Dominar el vocabulario técnico es el primer paso para comunicarte como Senior.”

Nivel	Duración
Referencia	Consulta continua

## Introducción

Este glosario define **todos** los términos técnicos utilizados en la Guía MLOps v5.0. Cada término incluye: - **Definición técnica** precisa - **Analogía** para facilitar comprensión - **Términos relacionados** para profundizar

## A

### Accuracy (Exactitud)

**Definición:** Métrica de clasificación:  $(TP + TN) / Total$ . Porcentaje de predicciones correctas.

**Analogía:** Arquero contando flechas en el blanco. 85 de 100 = 85% accuracy. Engañosa con clases desbalanceadas.

**Relacionados:** Precision, Recall, F1 Score

### ADR (Architecture Decision Record)

**Definición:** Documento que registra decisiones de arquitectura con contexto, opciones y consecuencias.

**Analogía:** Acta de reunión de arquitectos. Años después, cualquiera entiende POR QUÉ se decidió algo.

**Relacionados:** ML Canvas, C4 Model, Documentación

### API (Application Programming Interface)

**Definición:** Contrato que define cómo programas se comunican. En MLOps, APIs REST exponen modelos como servicios web.

**Analogía:** Mesero en restaurante. No vas a la cocina; el mesero lleva tu pedido y trae la respuesta.

```
@app.post("/predict")
def predict(data: CustomerData) -> PredictionResponse:
    return {"probability": model.predict_proba([data])[0, 1]}
```

**Relacionados:** REST, FastAPI, Endpoint, HTTP

## Artefacto (Artifact)

**Definición:** Archivo generado durante ML: modelos (.pkl), datasets, gráficos, reportes.

**Analogía:** Productos de fábrica. Planos → piezas → motor → auto. Cada artefacto debe ser rastreable.

**Relacionados:** MLflow, Model Registry, DVC

---

## ASGI (Asynchronous Server Gateway Interface)

**Definición:** Especificación para servidores web async en Python. Maneja múltiples requests concurrentemente.

**Analogía:** Mesero que anota pedido mesa 1, mientras espera va a mesa 2, etc. Maneja conversaciones “en paralelo”.

**Relacionados:** Uvicorn, FastAPI, Async/Await

---

## AUC-ROC

**Definición:** Área bajo curva ROC. Mide capacidad de distinguir clases. 1.0 = perfecto, 0.5 = aleatorio.

**Analogía:** Separando manzanas buenas de malas. AUC 0.9 = 90% de las veces asigna mayor score a la manzana buena.

**Relacionados:** ROC Curve, Precision, Recall, Threshold

---

## Auto-scaling

**Definición:** Sistema que aumenta/disminuye recursos automáticamente según demanda.

**Analogía:** Restaurante contratando meseros temporales cuando hay mucha gente.

**Relacionados:** HPA, Kubernetes, Load Balancer

---

## B

---

### Backpropagation

**Definición:** Algoritmo de entrenamiento de redes neuronales que propaga el error hacia atrás calculando gradientes.

**Analogía:** Equipo de relevos donde analizas hacia atrás quién contribuyó al fallo.

**Relacionados:** Gradient Descent, Learning Rate, Neural Network

---

### Baseline

**Definición:** Modelo simple como referencia. Si tu modelo complejo no lo supera, algo está mal.

**Analogía:** Antes de comprar auto deportivo, verifica que sea más rápido que tu bicicleta.

**Relacionados:** Benchmark, Model Evaluation

---

### BaseEstimator

**Definición:** Clase base sklearn con `get_params()` y `set_params()`. Todos los estimadores heredan de ella.

**Analogía:** Contrato estándar que todos los constructores deben seguir para que el sistema funcione.

**Relacionados:** TransformerMixin, Custom Transformer, Pipeline

---

### Batch Prediction

**Definición:** Procesar múltiples muestras a la vez, programadamente. Contrasta con online/real-time.

**Analogía:** Catering (cocinas todo de antemano) vs restaurante a la carta (cocinas cada plato al pedirlo).

**Relacionados:** Online Prediction, Latencia

---

## Black

**Definición:** Formateador Python opinado. Aplica estilo consistente automáticamente.

**Analogía:** Corrector que arregla gramática y estilo sin preguntarte.

```
black src/
```

**Relacionados:** Linting, Flake8, isort

---

## Branch (Rama)

**Definición:** Línea de desarrollo paralela en Git.

**Analogía:** Fotocopia del manuscrito para probar final alternativo sin afectar original.

```
git checkout -b feature/add-mlflow
```

**Relacionados:** Git, Merge, Pull Request

---

## C

---

### C4 Model

**Definición:** Visualización de arquitectura en 4 niveles: Context, Container, Component, Code.

**Analogía:** Google Maps con zoom. Mundo → País → Ciudad → Calle.

**Relacionados:** ADR, Arquitectura

---

### CI/CD

**Definición:** Continuous Integration (tests automáticos) + Continuous Deployment (deploy automático).

**Analogía:** Fábrica con control de calidad automatizado que envía autos aprobados al concesionario.

**Relacionados:** GitHub Actions, Pipeline, DevOps

---

### Classification

**Definición:** Problema ML supervisado para predecir categorías discretas.

**Analogía:** Doctor diagnosticando enfermedades (multiclase) o decidiendo operar/no operar (binaria).

**Relacionados:** Regression, Supervised Learning

---

### Class Imbalance

**Definición:** Una clase muy sobrerrepresentada (ej: 95% no-fraude, 5% fraude).

**Analogía:** Entrenar perro con 1000 piedras y 10 trufas. Aprende a decir “piedra” siempre.

**Soluciones:** `class_weight='balanced'`, SMOTE, métricas apropiadas

---

### Cold Start

**Definición:** Tiempo para que servicio esté listo tras iniciarse. Incluye cargar modelo en memoria.

**Analogía:** Encender auto en invierno. Debes esperar que el motor se caliente.

**Relacionados:** Serverless, Lambda, Latencia

---

### ColumnTransformer

**Definición:** Sklearn: aplica diferentes transformaciones a diferentes columnas.

**Analogía:** Lavandería con máquinas diferentes: color→encoder, blanca→scaler, delicados→passthrough.

```
preprocessor = ColumnTransformer([\n    ('num', StandardScaler(), num_cols),\n    ('cat', OneHotEncoder(), cat_cols),\n])
```

**Relacionados:** Pipeline, Transformer

---

## Commit

**Definición:** Snapshot de cambios en Git con hash único y mensaje.

**Analogía:** Foto de tu escritorio. Puedes volver a cualquier foto anterior.

```
git commit -m "feat: add probability calibration"
```

**Relacionados:** Git, Branch, Push

---

## Concept Drift

**Definición:** Cambio en relación features-target. Patrones aprendidos ya no son válidos.

**Analogía:** Modelo entrenado pre-pandemia predice gustos de películas post-pandemia incorrectamente.

**vs Data Drift:** Data Drift = cambia X. Concept Drift = cambia  $P(Y|X)$ .

---

## ConfigMap

**Definición:** Kubernetes: almacena configuración no sensible como pares clave-valor.

**Analogía:** Tablón de anuncios de oficina. Información pública que todos necesitan.

---

## Container (Contenedor)

**Definición:** Software empaquetado con código y dependencias. Ejecuta igual en cualquier ambiente.

**Analogía:** Contenedor de barco. Funciona igual en cualquier puerto.

**Relacionados:** Docker, Image, Kubernetes

---

## Coverage

**Definición:** Porcentaje de código ejecutado por tests. No garantiza corrección.

**Analogía:** Inspector que revisó 80% de habitaciones. No significa que encontró todos los problemas.

```
pytest --cov=src
```

**Target:** >80% para código crítico

---

## Cross-Validation

**Definición:** Evaluar modelo dividiendo datos en K folds. Entrena K veces con diferentes splits.

**Analogía:** 5 estudiantes, 5 rondas. En cada ronda, diferente estudiante es evaluado.

```
scores = cross_val_score(model, X, y, cv=5, scoring='roc_auc')
```

**Relacionados:** K-Fold, Overfitting

---

## Custom Transformer

**Definición:** Clase sklearn personalizada que hereda BaseEstimator + TransformerMixin.

**Analogía:** Pieza LEGO personalizada con conexiones estándar (fit/transform).

```
class RatioFeatures(BaseEstimator, TransformerMixin):
    def fit(self, X, y=None): return self
    def transform(self, X):
        X['ratio'] = X['Balance'] / (X['Products'] + 1)
        return X
```

---

## D

---

### DAG (Directed Acyclic Graph)

**Definición:** Grafo dirigido sin ciclos. Representa dependencias entre tareas.

**Analogía:** Instrucciones de receta. No puedes hornear antes de mezclar.

**Relacionados:** DVC, Pipeline, Airflow

---

### Data Drift

**Definición:** Cambio en distribución estadística de features entre entrenamiento y producción.

**Analogía:** Modelo de precios 2019 vs datos 2021 post-pandemia.

**Detección:** KS test, PSI, Evidently, NannyML

---

### Data Leakage

**Definición:** Información del futuro o test filtra al entrenamiento. Métricas infladas.

**Analogía:** Estudiar con las respuestas del mismo examen. 100% en práctica, 0% en real.

**Ejemplos:** `price_per_mile = price / miles`, normalizar antes de split.

---

### Dependency Injection

**Definición:** Dependencias se pasan desde afuera en lugar de crearse internamente.

**Analogía:** Cafetería recibe leche de proveedor en vez de tener vacas propias.

```
# Con DI: fácil de testear
class Predictor:
    def __init__(self, model: BaseEstimator):
        self.model = model # Inyectado
```

**Relacionados:** SOLID, Testing

---

### Deployment

**Definición:** Poner modelo/aplicación en ambiente donde usuarios reales lo usan.

**Analogía:** Abrir restaurante al público después de cocinar en casa y probar con amigos.

**Tipos:** Batch, REST API, Edge, Streaming

---

### Docker

**Definición:** Plataforma para aplicaciones en contenedores. Código + dependencias portables.

**Analogía:** Máquina del tiempo para código. Congelas ambiente exacto.

```
FROM python:3.11-slim
COPY . /app
RUN pip install -r requirements.txt
CMD ["uvicorn", "app:app"]
```

### Docstring

**Definición:** String de documentación al inicio de funciones/clases.

**Analogía:** Instrucciones en la caja de un producto.

```
def predict(data: pd.DataFrame) -> np.ndarray:
    """
    Genera predicciones de churn.

    Args:
        data: DataFrame con features.
    Returns:
        Array de probabilidades 0-1.
    """
```

### DVC (Data Version Control)

**Definición:** Versiona datasets y pipelines ML. Datos grandes en storage remoto, metadatos en Git.

**Analogía:** Git = álbum con miniaturas. DVC = almacén con fotos originales grandes.

```
dvc add data/dataset.csv
dvc push
git add data/dataset.csv.dvc
```

---

## E

---

### E2E Test

**Definición:** Test del sistema completo, desde entrada hasta salida final.

**Analogía:** Test drive de auto completo, no motor aislado.

---

### Early Stopping

**Definición:** Detiene entrenamiento cuando validación deja de mejorar. Evita overfitting.

**Analogía:** Sacar galletas del horno cuando están doradas, antes de que se quemen.

```
EarlyStopping(monitor='val_loss', patience=5)
```

---

### Embedding

**Definición:** Representación vectorial densa de datos de alta dimensionalidad.

**Analogía:** Mapear ciudades del mundo en papel 2D. Similares quedan cerca.

**Uso:** Word2Vec, Entity embeddings

---

### Endpoint

**Definición:** URL específica de API que realiza operación particular.

**Analogía:** Ventanillas de banco. Cada una hace algo diferente.

```
@app.get("/health")
@app.post("/predict")
```

---

### Ensemble

**Definición:** Combina múltiples modelos para mejores predicciones.

**Analogía:** 100 doctores opinando en vez de 1. Opinión agregada suele ser mejor.

**Tipos:** Bagging (Random Forest), Boosting (XGBoost), Stacking

---

### Environment

**Definición:** Conjunto aislado de dependencias donde ejecuta código.

**Analogía:** Diferentes cocinas para diferentes tipos de comida.

**Tipos:** Desarrollo, Staging, Producción

---

### Experiment Tracking

**Definición:** Registrar parámetros, métricas, artefactos de cada experimento ML.

**Analogía:** Cuaderno de laboratorio de científico.

**Herramientas:** MLflow, W&B, Neptune

---

---

## F

---

### F1 Score

**Definición:** Media armónica de Precision y Recall. Balance entre ambas.

**Fórmula:**  $F1 = 2 \times (Precision \times Recall) / (Precision + Recall)$

**Analogía:** Buscador de trufas. No sirve encontrar pocas muy precisamente ni todas con muchas falsas.

---

## FastAPI

**Definición:** Framework Python para APIs de alto rendimiento con validación automática.

**Analogía:** Mesero eficiente que valida pedidos, da menú descriptivo, atiende muchas mesas.

```
from fastapi import FastAPI
from pydantic import BaseModel

app = FastAPI()

@app.post("/predict")
async def predict(data: PredictionInput):
    return {"probability": model.predict_proba([data])[0, 1]}
```

**Relacionados:** Pydantic, Uvicorn, REST

---

## Feature

**Definición:** Variable de entrada para predicciones.

**Analogía:** Ingredientes de receta. Para predecir si pastel sale bien: harina, azúcar, temperatura.

**Tipos:** Numéricas, Categóricas, Binarias, Derivadas

---

## Feature Engineering

**Definición:** Crear/transformar/seleccionar features para mejorar modelo.

**Analogía:** Chef preparando ingredientes. Ingredientes crudos se transforman en algo digerible.

```
df['balance_per_product'] = df['Balance'] / (df['NumOfProducts'] + 1)
```

---

## Feature Store

**Definición:** Sistema centralizado para almacenar y servir features consistentemente.

**Analogía:** Almacén central de ingredientes preparados para cadena de restaurantes.

**Herramientas:** Feast, Tecton

---

## Fixture (pytest)

**Definición:** Función que provee datos/recursos reutilizables para tests.

**Analogía:** Setup de set de filmación antes de cada escena.

```
@pytest.fixture
def sample_customer():
    return {"Age": 35, "Balance": 50000}
```

---

## Flake8

**Definición:** Linting para Python: errores lógicos, estilo PEP8, complejidad.

**Analogía:** Corrector de estilo de periódico.

```
flake8 src/
```

---

# G

---

## Git

**Definición:** Control de versiones distribuido.

**Analogía:** “Deshacer” infinito. Volver a cualquier momento, ver qué cambió y por qué.

```
git add . && git commit -m "mensaje" && git push
```

---

## GitHub Actions

**Definición:** CI/CD integrado en GitHub.

**Analogía:** Mayordomo robot que ejecuta instrucciones automáticamente.

```
on: [push]
jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - run: pytest tests/
```

---

## Gitleaks

**Definición:** Detecta secrets accidentalmente commiteados.

**Analogía:** Detector de metales en aeropuerto para código.

---

## Gradient Descent

**Definición:** Algoritmo que encuentra parámetros que minimizan pérdida.

**Analogía:** En montaña con niebla, das pasos pequeños siempre cuesta abajo.

**Relacionados:** Learning Rate, Loss Function

---

## Grafana

**Definición:** Visualización y dashboards para métricas.

**Analogía:** Tablero de instrumentos de avión.

**Relacionados:** Prometheus, Observabilidad

---

# H

---

## Health Check

**Definición:** Endpoint que verifica si servicio funciona.

**Analogía:** Médico preguntando “¿cómo te sientes?”.

```
@app.get("/health")
def health():
    return {"status": "healthy"}
```

---

## HPA (Horizontal Pod Autoscaler)

**Definición:** Kubernetes: escala pods automáticamente según métricas.

**Analogía:** Gerente de restaurante que llama más meseros si hay muchas mesas ocupadas.

---

## Hyperparameter

**Definición:** Parámetro configurado ANTES del entrenamiento.

**Analogía:** Decisiones antes de hornear: temperatura, tiempo, tamaño de molde.

**Ejemplos:** n\_estimators, learning\_rate, max\_depth

---

## Hyperparameter Tuning

**Definición:** Encontrar combinación óptima de hiperparámetros.

**Analogía:** Afinar guitarra. Probar perillas hasta mejor sonido.

**Técnicas:** Grid Search, Random Search, Bayesian Optimization

---



## I

---

### Image (Docker)

**Definición:** Template inmutable para crear contenedores.

**Analogía:** Receta + ingredientes pre-empaquetados. Imagen es el kit, contenedor es el pastel horneado.

---

### Imputer

**Definición:** Rellena valores faltantes (NaN).

**Analogía:** Restaurador de pinturas rellenando huecos.

```
SimpleImputer(strategy='median')
```

---

### Inference

**Definición:** Usar modelo entrenado para predicciones sobre datos nuevos.

**Analogía:** Entrenamiento = estudiar, Inferencia = tomar el examen.

---

### Ingress

**Definición:** Kubernetes: gestiona acceso HTTP externo al cluster.

**Analogía:** Recepción de edificio que dirige tráfico.

---

### Integration Test

**Definición:** Verifica que múltiples componentes funcionan juntos.

**Analogía:** Probar que motor, transmisión y ruedas funcionan juntos.

---

### isort

**Definición:** Ordena imports de Python automáticamente.

**Analogía:** Organizador de armario que siempre pone ropa en mismo orden.

---

## J

---

### Job (GitHub Actions)

**Definición:** Conjunto de steps en mismo runner.

**Relacionados:** Workflow, Step, Runner

---

### Joblib

**Definición:** Serializa objetos Python, especialmente modelos sklearn.

```
joblib.dump(model, "model.pkl")
model = joblib.load("model.pkl")
```

---

## K

---

### Kubernetes (K8s)

**Definición:** Orquestador de contenedores para automatizar despliegue y escalado.

**Analogía:** Director de orquesta coordinando muchos músicos (contenedores).

**Recursos:** Pod, Deployment, Service, Ingress

---

**K-Fold**

**Definición:** Dividir datos en K partes para cross-validation.

**Relacionados:** Cross-Validation, Stratified

---

**L**

---

**Latency (Latencia)**

**Definición:** Tiempo de respuesta del sistema. En APIs ML: milisegundos.

**Analogía:** Tiempo entre pedir comida y que llegue.

**P95:** El 95% de requests responden en menos de X ms.

---

**Learning Rate**

**Definición:** Tamaño de paso en gradient descent.

**Analogía:** Paso grande = llegas rápido pero puedes pasar el mínimo. Paso pequeño = lento pero preciso.

---

**Linting**

**Definición:** Análisis estático para detectar errores y violaciones de estilo.

**Herramientas:** Flake8, pylint, mypy

---

**Load Balancer**

**Definición:** Distribuye tráfico entre múltiples servidores.

**Analogía:** Hostess de restaurante que asigna mesas equitativamente.

---

**Loss Function (Función de Pérdida)**

**Definición:** Mide qué tan mal son las predicciones. El entrenamiento la minimiza.

**Ejemplos:** MSE (regresión), Cross-Entropy (clasificación)

---

**M**

---

**Makefile**

**Definición:** Archivo con comandos abreviados para tareas comunes.

```
test:
  pytest tests/ -v
lint:
  black src/ && flake8 src/
```

---

**Matrix (GitHub Actions)**

**Definición:** Ejecutar job con múltiples combinaciones de parámetros.

```
strategy:
  matrix:
    python-version: [3.10, 3.11]
```

---

**Metric (Métrica)**

**Definición:** Valor numérico que mide rendimiento del modelo.

**Clasificación:** Accuracy, Precision, Recall, F1, AUC **Regresión:** MSE, RMSE, MAE, R²

---

**Middleware**

**Definición:** Código que intercepta requests/responses entre cliente y aplicación.

**Analogía:** Portero que revisa credenciales antes de dejarte pasar.

---

## MLflow

**Definición:** Plataforma open-source para gestionar ciclo de vida ML.

**Componentes:** Tracking, Projects, Models, Registry

```
with mlflow.start_run():
    mlflow.log_params(params)
    mlflow.log_metrics(metrics)
    mlflow.sklearn.log_model(model, "model")
```

---

## MLOps

**Definición:** Prácticas que combinan ML + DevOps + Data Engineering para sistemas ML en producción.

**Analogía:** DevOps fue para software. MLOps es para sistemas de ML.

---

## Model Card

**Definición:** Documento describiendo modelo: propósito, datos, métricas, limitaciones, ética.

**Analogía:** Prospecto de medicamento. Información completa sobre qué hace y sus efectos.

---

## Model Registry

**Definición:** Sistema para versionar y gestionar modelos ML.

**Estados:** Staging → Production → Archived

---

## mypy

**Definición:** Type checking estático para Python.

```
mypy src/
```

**Relacionados:** Type Hints, Pydantic

---

## N

---

### NaN (Not a Number)

**Definición:** Valor especial para datos faltantes o indefinidos.

```
import numpy as np
np.nan
```

---

## Namespace

**Definición:** Kubernetes: división lógica del cluster para aislamiento.

**Analogía:** Departamentos en una empresa. Cada uno tiene sus recursos.

---

## O

---

### Observability (Observabilidad)

**Definición:** Capacidad de entender estado interno de sistema desde outputs externos.

**3 Pilares:** Logs, Metrics, Traces

**Analogía:** Instrumentos de avión. Si no puedes ver, no puedes arreglar.

---

## One-Hot Encoding

**Definición:** Convierte variables categóricas en vectores binarios.

```
Country: [France, Spain, Germany]
France → [1, 0, 0]
Spain  → [0, 1, 0]
```

## Overfitting (Sobreajuste)

**Definición:** Modelo memoriza datos de entrenamiento, no generaliza.

**Analogía:** Estudiante que memoriza respuestas exactas pero no entiende conceptos.

**Señales:** Train accuracy muy alta, validation accuracy baja.

**Soluciones:** Más datos, regularización, early stopping, dropout

## P

### Pipeline (sklearn)

**Definición:** Secuencia de transformaciones y estimador final encadenados.

**Analogía:** Línea de ensamblaje. Cada estación hace una transformación.

```
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('model', RandomForestClassifier())
])
```

## Pod

**Definición:** Kubernetes: unidad de deployment más pequeña. Uno o más contenedores.

**Analogía:** Apartamento en edificio. Contenedores son habitaciones del apartamento.

## Precision (Precisión)

**Definición:** De predicciones positivas, ¿cuántas son correctas?  $TP / (TP + FP)$

**Analogía:** De las personas que detuviste como sospechosas, ¿cuántas eran realmente criminales?

## Pre-commit Hook

**Definición:** Script que se ejecuta automáticamente antes de cada commit.

**Analogía:** Control de calidad que revisa tu trabajo antes de entregarlo.

```
# .pre-commit-config.yaml
repos:
- repo: https://github.com/psf/black
  hooks:
  - id: black
```

## Prometheus

**Definición:** Sistema de monitoreo y alertas. Recolecta métricas de servicios.

**Relacionados:** Grafana, Metrics, Observabilidad

## Pull Request (PR)

**Definición:** Solicitud para integrar cambios con revisión de código.

**Analogía:** Propuesta formal que requiere aprobación antes de aceptarse.

## Pydantic

**Definición:** Validación de datos en Python usando type hints.

```
class Customer(BaseModel):
    age: int = Field(ge=18, le=100)
    name: str
```

**Relacionados:** Type Hints, FastAPI, Validation

---

## pytest

**Definición:** Framework de testing para Python.

```
def test_prediction():
    result = model.predict([[35, 50000]])
    assert result[0] in [0, 1]
```

## R

---

### Recall (Sensibilidad)

**Definición:** De todos los positivos reales, ¿cuántos detectamos?  $TP / (TP + FN)$

**Analogía:** De todos los criminales reales, ¿a cuántos capturaste?

---

### Regression (Regresión)

**Definición:** Problema ML para predecir valor numérico continuo.

**Ejemplos:** Precio de casa, temperatura, ventas

---

### Regularization (Regularización)

**Definición:** Técnicas para prevenir overfitting penalizando complejidad.

**Tipos:** L1 (Lasso), L2 (Ridge), Dropout, Early Stopping

---

## Replica

**Definición:** Copia de un pod/servicio para alta disponibilidad.

**Relacionados:** Deployment, ReplicaSet

---

### Reproducibility (Reproducibilidad)

**Definición:** Obtener mismos resultados con mismo código y datos.

**Clave:** Seeds, versionado de datos/código/ambiente

---

## REST API

**Definición:** Estilo arquitectónico con HTTP methods: GET, POST, PUT, DELETE.

**Relacionados:** API, HTTP, Endpoint

---

## S

---

### Scaling (Escalado de Features)

**Definición:** Normalizar features a rango similar.

**Técnicas:** StandardScaler (z-score), MinMaxScaler (0-1)

---

### Scikit-learn (sklearn)

**Definición:** Librería Python para ML clásico.

**Módulos:** preprocessing, model\_selection, ensemble, metrics

---

## Secret

**Definición:** Valor sensible (contraseña, API key) que no debe estar en código.

**Kubernetes:** Objeto Secret para almacenar datos sensibles encriptados.

---

## Seed (Random State)

**Definición:** Valor para inicializar generadores aleatorios. Garantiza reproducibilidad.

```
np.random.seed(42)
RandomForestClassifier(random_state=42)
```

---

## Service (Kubernetes)

**Definición:** Abstracción que expone pods como servicio de red.

**Tipos:** ClusterIP, NodePort, LoadBalancer

---

## SHAP

**Definición:** SHapley Additive exPlanations. Explica predicciones asignando importancia a features.

**Relacionados:** Interpretabilidad, Feature Importance

---

## SOLID

**Definición:** Principios de diseño orientado a objetos. - **S**: Single Responsibility - **O**: Open/Closed - **L**: Liskov Substitution - **I**: Interface Segregation - **D**: Dependence Inversion

---

## Staging

**Definición:** Ambiente que replica producción para testing final.

**Analogía:** Ensayo general antes del estreno.

---

## Stratified Split

**Definición:** División que mantiene proporción de clases en train y test.

```
train_test_split(X, y, stratify=y)
```

---

## T

---

### Target

**Definición:** Variable que queremos predecir. También llamada “label” o “y”.

---

### Terraform

**Definición:** Infrastructure as Code. Provisiona recursos en cloud con código.

```
resource "aws_instance" "ml_server" {
  instance_type = "t3.medium"
}
```

---

### Test Coverage

**Definición:** Porcentaje de código ejecutado durante tests.

**Relacionados:** Coverage, pytest

---

## Threshold (Umbral)

**Definición:** Punto de corte para convertir probabilidades en clases.

**Default:** 0.5, pero ajustable según necesidades de negocio.

## Throughput

**Definición:** Cantidad de predicciones/requests por unidad de tiempo.

**Analogía:** Cuántos platos puede servir el restaurante por hora.

## Traces

**Definición:** Seguimiento de requests a través de sistema distribuido.

**Herramientas:** Jaeger, OpenTelemetry

**Relacionados:** Observabilidad, Logs, Metrics

## TransformerMixin

**Definición:** Mixin sklearn que añade `fit_transform()` automáticamente.

**Relacionados:** BaseEstimator, Custom Transformer

## Trivy

**Definición:** Escáner de vulnerabilidades para contenedores.

```
trivy image my-app:latest
```

## Type Hints

**Definición:** Anotaciones en Python que indican tipos esperados.

```
def predict(data: pd.DataFrame) -> np.ndarray:
    pass
```

**Relacionados:** mypy, Pydantic

# U

## Underfitting (Subajuste)

**Definición:** Modelo demasiado simple. No captura patrones.

**Señales:** Train y validation accuracy bajas.

## Unit Test

**Definición:** Test de función/método individual en aislamiento.

```
def test_feature_ratio():
    result = compute_ratio(100, 2)
    assert result == 50
```

## Uvicorn

**Definición:** Servidor ASGI de alto rendimiento para FastAPI.

```
uvicorn app:app --host 0.0.0.0 --port 8000
```

# V

## Validation Set

**Definición:** Datos para ajustar hiperparámetros, separado de train y test.

**Split típico:** 60% train, 20% validation, 20% test

---

## Vendor Lock-in

**Definición:** Dependencia de proveedor específico que dificulta migración.

**Analogía:** Comprar auto donde repuestos solo existen en una tienda.

---

## Version Control

**Definición:** Sistema para rastrear cambios en archivos.

**Herramientas:** Git (código), DVC (datos), MLflow (modelos)

---

## Voting Classifier

**Definición:** Ensemble que combina predicciones por votación.

```
VotingClassifier([
    ('rf', RandomForestClassifier()),
    ('xgb', XGBClassifier())
], voting='soft')
```

---

# W

---

## Weights & Biases (W&B)

**Definición:** Plataforma SaaS para experiment tracking con visualizaciones avanzadas.

**Relacionados:** MLflow, Experiment Tracking

---

## Workflow (GitHub Actions)

**Definición:** Proceso automatizado definido en archivo YAML.

**Relacionados:** Job, Step, CI/CD

---

# X

---

## XGBoost

**Definición:** Implementación optimizada de gradient boosting. Muy popular en competencias.

```
from xgboost import XGBClassifier
model = XGBClassifier(n_estimators=100, learning_rate=0.1)
```

---

# Y

---

## YAML

**Definición:** Formato de serialización legible para configuración.

```
model:
  type: ensemble
  n_estimators: 100
```

---

# Z

---

## Zero-Downtime Deployment

**Definición:** Actualizar aplicación sin interrumpir servicio.



Técnicas: Rolling update, Blue-green deployment

## Símbolos y Abreviaciones

Símbolo	Significado
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
P95	Percentil 95
GHCR	GitHub Container Registry
IaC	Infrastructure as Code
DAG	Directed Acyclic Graph
OOM	Out of Memory
CRUD	Create, Read, Update, Delete
SLA	Service Level Agreement
SLO	Service Level Objective
TTL	Time To Live

## Tablas de Referencia Rápida

### Métricas de Clasificación

Métrica	Fórmula	Uso
Accuracy	$(TP+TN)/(Total)$	Balance general (clases balanceadas)
Precision	$TP/(TP+FP)$	Minimizar falsos positivos
Recall	$TP/(TP+FN)$	Minimizar falsos negativos
F1	$2 \times P \times R / (P+R)$	Balance P y R
AUC-ROC	Área bajo curva	Capacidad discriminatoria

### Tipos de Testing

Tipo	Alcance	Ejemplo
Unit	Función individual	<code>test_compute_ratio()</code>
Integration	Múltiples componentes	<code>test_pipeline_fit()</code>
E2E	Sistema completo	<code>test_api_predict_flow()</code>

### Ambientes

Ambiente	Propósito	Datos
Development	Desarrollo	Sintéticos/muestra
Staging	Testing final	Réplica producción
Production	Usuarios reales	Reales

Navegación

◀ Anterior	Índice	▶ Siguiente
17_PROYECTO_INTEGRADOR.md	Índice	19_DECISIONES_TECH.md

---

© 2025 DuqueOM - Guía MLOps v5.0: Senior Edition

Módulo 18 Completado