Unidad 1: Fundamentos y componentes de sistemas de IA

Arquitectura de Sistemas de IA

E. Ulises Moya Sánchez
Licenciatura en IA y Ciencia de Datos

Departamento de Innovación Tecnológica

28 de enero de 2026

Contenido

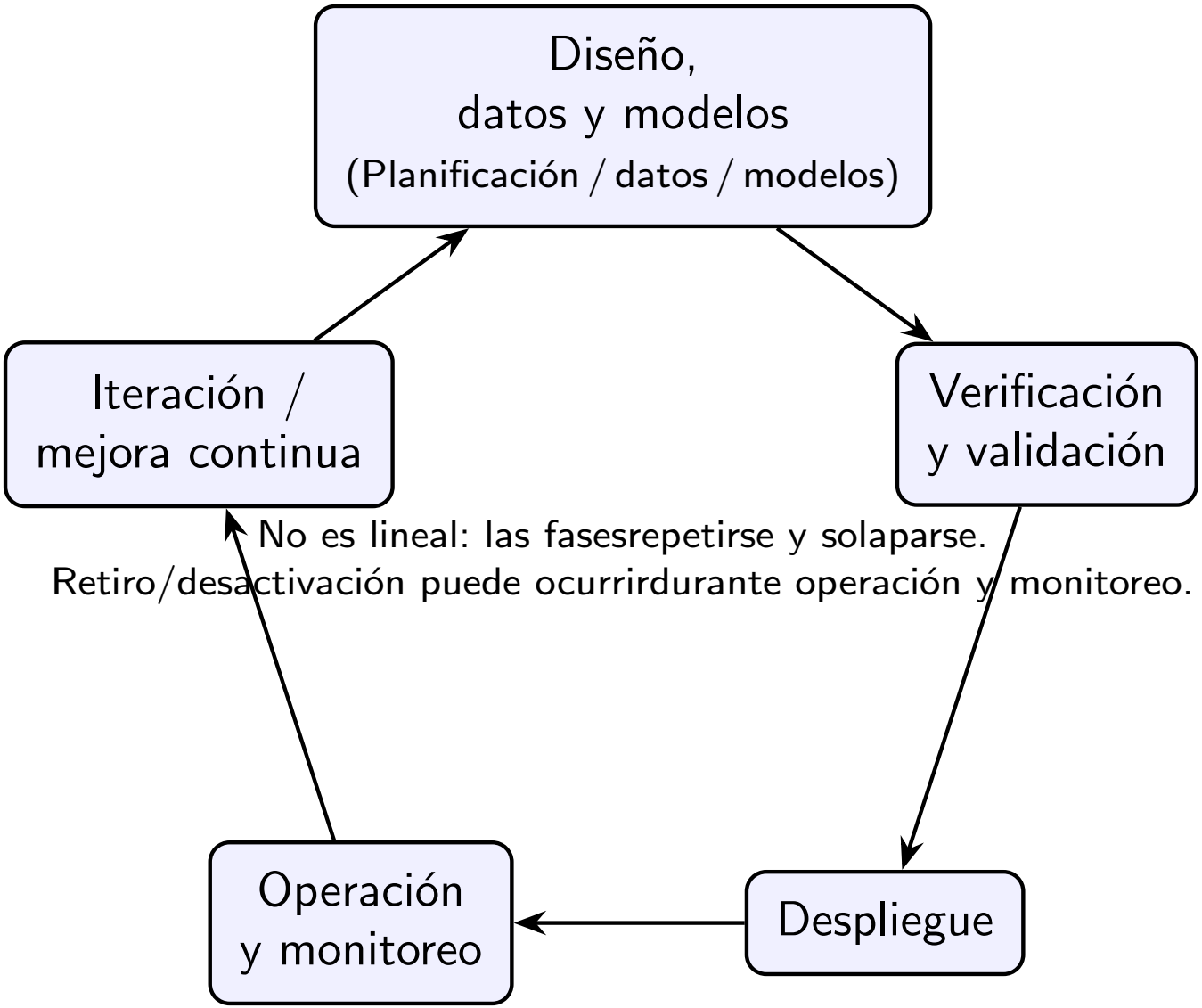
¿Qué es arquitectura de IA?

Idea central

Diseño de la estructura de un sistema que **crea valor con modelos** (ML/LLMs), considerando datos, entrenamiento, despliegue, monitoreo y gobierno.

- No es solo el modelo: incluye pipelines, servicios, almacenamiento, seguridad y observabilidad.
- La arquitectura debe soportar cambio: datos nuevos, drift, versiones de modelos.

Ciclo de vida de un sistema de IA (OECD)



Elementos principales del ciclo de vida (OECD)

OECD: *Advancing accountability in AI* (2448f04b-en)

Fases

1. **Diseño, datos y modelos** (secuencia dependiente del contexto):
   - **Planificación y diseño**: objetivo, alcance, requisitos, riesgos.
   - **Recolección y procesamiento de datos**: calidad, sesgos, trazabilidad.
   - **Construcción del modelo**: selección, entrenamiento e interpretación.
2. **Verificación y validación**: pruebas, robustez, seguridad, métricas y criterios de aceptación.
3. **Despliegue**: integración en sistemas/procesos, control de versiones, documentación.
4. **Operación y monitoreo**: rendimiento, drift, incidentes, actualizaciones y retiro cuando aplique.

1.1. Arquitectura general de sistemas de IA

Tres capas principales

1. **Capa de Datos**: Ingesta, almacenamiento y preparación de datos.
2. **Capa de Modelo**: Entrenamiento, evaluación y gestión de modelos.
3. **Capa de Consumo**: Despliegue, inferencia y monitoreo.

Tecnologías por capa

- **Datos**: Apache Kafka, Spark, DBT, Snowflake, BigQuery
- **Modelo**: TensorFlow, PyTorch, scikit-learn, MLflow, Weights & Biases
- **Consumo**: FastAPI, Docker, Kubernetes, Seldon, AWS SageMaker

Capa de datos: flujos y orquestación (comparativo)

Tecnología	Tipo	Cuándo usar	Notas
Apache Airflow	Orquestación batch (DAGs)	ETL/ELT programado, pipelines reproducibles	No es streaming; integra con muchos sistemas
Apache Kafka	Streaming / event log	Ingesta en tiempo real, desacoplar productores/consumidores	Alta escalabilidad; base para arquitecturas event-driven
Apache Spark (Structured Streaming)	Procesamiento batch + stream	Transformaciones pesadas, agregaciones, ETL a gran escala	Corre en clusters; buena integración con lake/lakehouse
Apache Flink	Streaming (estado)	Baja latencia, ventanas, exactamente-una-vez	Stream-first; pipelines complejos
AWS Kinesis	Streaming gestionado (AWS)	Alternativa a Kafka sin operar clúster	Integración con Lambda/S3/Glue
GCP Pub/Sub	Mensajería gestionada (GCP)	Eventos globales, fan-out, integración con Dataflow	Escalado automático; baja operación
Azure Event Hubs	Streaming gestionado (Azure)	Ingesta de eventos y telemetría en Azure	Integración con Stream Analytics

Componentes típicos en un sistema con IA

- Fuentes de datos (batch/stream)
- Ingesta y calidad de datos
- Feature store / embeddings store
- Entrenamiento y evaluación
- Registro de modelos (model registry)
- Servicio de inferencia (online/batch)
- Monitoreo (latencia, calidad, drift)

Preguntas de diseño

- ¿Tiempo real o batch?
- ¿Explicabilidad requerida?
- ¿Tolerancia a fallos?
- ¿Costo por predicción?

Requerimientos (no funcionales) comunes

- **Confiabilidad**: degradación controlada (fallback) si el modelo falla.
- **Observabilidad**: trazas, métricas, logs, y monitoreo de drift.
- **Seguridad y privacidad**: control de acceso, manejo de PII, data governance.
- **Reproducibilidad**: versionado de código, datos, features y modelos.
- **Cumplimiento**: auditoría, explicabilidad cuando aplique.

1.2. Componentes clave

Componentes

- Conjuntos de datos
- Características (features)
- Modelos
- Artefactos
- Registries (modelos, features)

Tecnologías asociadas

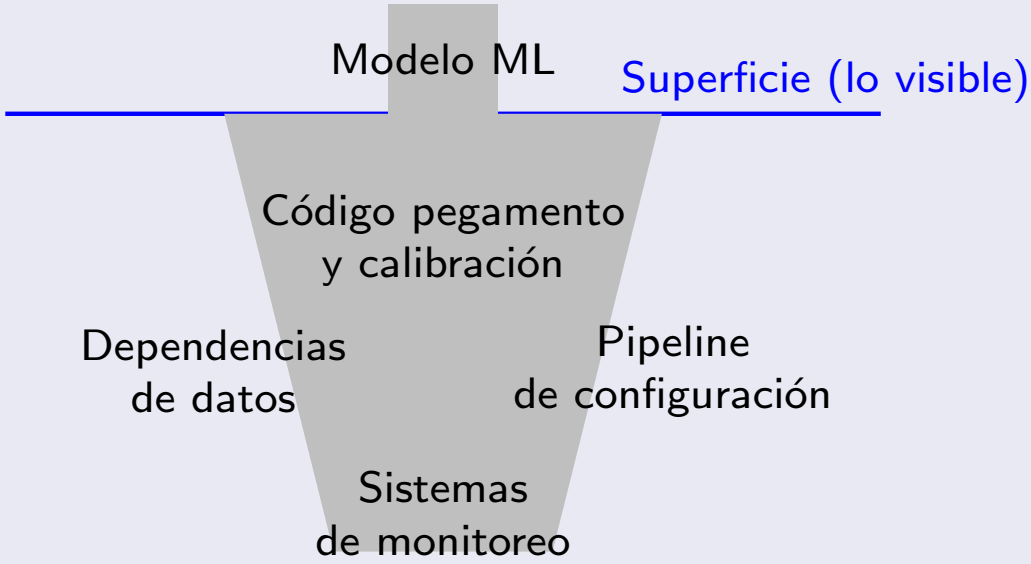
- **Datasets**: Hugging Face Datasets, TFDS, Pandas
- **Features**: Feast, Tecton, Hopsworks
- **Modelos**: ONNX, PMML, TensorFlow Serving
- **Artifacts**: MLflow, DVC, Neptune
- **Registries**: MLflow Registry, SageMaker Model Registry

El mito del modelo como componente central

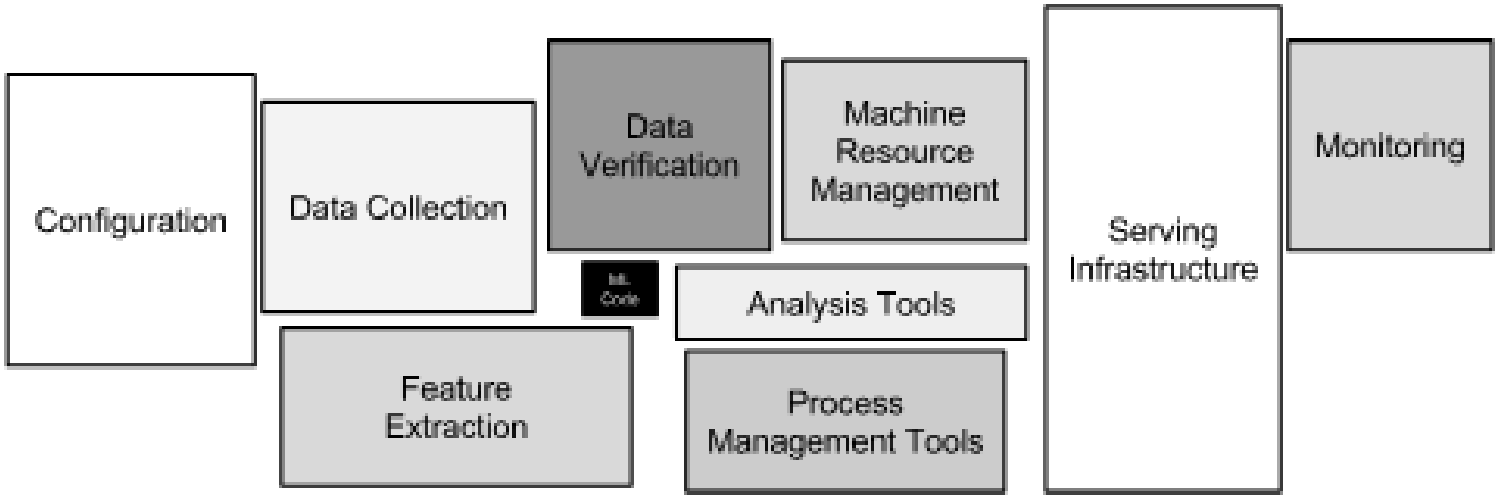
Paradoja del desarrollo ML

- **Desarrollo inicial**: Relativamente rápido y económico
- **Mantenimiento a largo plazo**: Extremadamente difícil y costoso
- **Analogía**: Como una "deuda técnica"que se acumula silenciosamente

El iceberg de los sistemas de ML



Technical Debt en sistemas de ML



Referencia: Sculley et al., *Hidden Technical Debt in Machine Learning Systems*, NeurIPS 2015.

Data-Centric AI (DCAI): idea clave

Enfoque

En lugar de asumir que el dataset es fijo, DCAI propone **mejorar sistemáticamente los datos** (calidad, cobertura y consistencia) para mejorar el desempeño del sistema.

- **Model-centric**: cambiar arquitectura/hiperparámetros para un dataset dado.
- **Data-centric**: iterar sobre el dataset (labels, sesgos, slices, distribución, duplicados, etc.).
- Mismo modelo + mejores datos suele dar mejoras reales y sostenibles.

Referencia: MIT CSAIL: Introduction to Data-Centric AI.

Técnicas DCAI: encontrar problemas en los datos

Diagnóstico

- **Análisis de errores**: revisar ejemplos mal predichos y patrones recurrentes.
- **Detección de errores de etiquetado** (label errors) y **duplicados**.
- **Desbalance de clases**, **outliers** y **distribution shift/drift**.
- **Evaluación por slices**: subpoblaciones donde el modelo falla (p. ej., por fuente, idioma, grupo, condiciones).

Referencia: MIT CSAIL: Introduction to Data-Centric AI.

Técnicas DCAI: mejorar el dataset

Intervenciones

- **Re-etiquetado dirigido**: corregir labels con mayor impacto (priorizar ejemplos influyentes).
- **Guías de etiquetado** + auditorías para consistencia entre anotadores.
- **Active learning**: muestrear ejemplos informativos para etiquetar.
- **Data augmentation** y **re-balanceo** (sobre/infra-muestreo) donde aplique.
- **Validaciones automáticas**: reglas de esquema, rangos, checks de calidad.
- **Versionado de datos** y trazabilidad (datasets, features, splits).

Referencia: MIT CSAIL: Introduction to Data-Centric AI.

1.3. Calidad de datos

Aspectos críticos

- **Sesgos de muestreo**: Representatividad insuficiente
- **Data drift**: Cambio en distribución de datos en producción
- **Validación**: Esquemas, rangos, integridad
- **Documentación**: Data cards, data statements

Tecnologías de calidad

- Great Expectations, TensorFlow Data Validation, Deequ (AWS)
- Evidently AI, Fiddler, WhyLabs (monitoreo)
- DataHub, Amundsen (catálogo y documentación)

1.4. Taxonomía de tareas y algoritmos

Supervisado

- Clasificación
- Regresión
- **Tecnologías**: scikit-learn, XGBoost, LightGBM

No supervisado

- Clustering
- Reducción dimensional
- **Tecnologías**: scikit-learn, HDBSCAN, UMAP

PLN / Visión

- PLN: NER, traducción, resumen
- Visión: detección, segmentación
- **Tecnologías**: Transformers (Hugging Face), OpenCV, Detectron2

1.5. Criterios de selección de algoritmos

Criterios por tarea

- **Clasificación**: Accuracy, precisión, recall, F1, AUC-ROC
- **Regresión**: MAE, MSE, RMSE, R^2
- **Clustering**: Silhouette score, Davies–Bouldin index
- **PLN/Visión**: BLEU, METEOR, mAP, IoU

Consideraciones técnicas

- Tamaño de datos
- Latencia requerida
- Coste computacional
- Interpretabilidad

1.6. Patrones de arquitectura

Batch vs Stream

- **Batch**: Procesamiento periódico
- **Stream**: Procesamiento en tiempo real
- **Tecnologías**: Apache Airflow (batch), Apache Flink/Kafka Streams (stream)

Offline vs Online

- **Offline**: Entrenamiento, evaluación
- **Online**: Inferencia, scoring
- **Tecnologías**: JupyterLab (offline), TensorFlow Serving/KFServing (online)

Feature Stores

- Almacén centralizado de características
- Reutilización y consistencia
- **Tecnologías**:
  - Feast (open source)
  - Tecton (SaaS)
  - SageMaker Feature Store (AWS)
  - Vertex AI Feature Store (GCP)

Resumen Unidad 1

Puntos clave

- Arquitectura de 3 capas: datos, modelo, consumo
- Componentes clave: datasets, features, modelos, artefactos
- Calidad de datos crítica para rendimiento y ética
- Selección de algoritmos basada en tarea y métricas
- Patrón adecuado según requisitos (batch/stream, offline/online)
- Feature stores facilitan reutilización y consistencia

Próximos pasos

- Unidad 2: Diseño de datos y entrenamiento reproducible
- Unidad 3: Integración, despliegue y escalabilidad
- Unidad 4: Ética, transparencia y tendencias

¡Gracias!

Preguntas y discusión