

Machine Learning Operations (MLOps)

Data Science (CC3084) - 2026

Machine Learning Operations (MLOps)

Semestre 02, 2026

De Dónde Venimos

Las lecciones anteriores construyeron modelos de series de tiempo, redes neuronales, clasificadores de texto y análisis de redes. En todos los casos el trabajo terminó en un notebook con una métrica de evaluación.

- El modelo se entrenó una sola vez, con un conjunto de datos fijo.
- La evaluación se hizo sobre un conjunto de prueba que no cambia.
- Nadie más depende de las predicciones.

Una organización no obtiene valor de un notebook. El valor aparece cuando el modelo recibe datos reales, responde a otros sistemas y se mantiene correcto mientras el mundo cambia. Esta lección trata de todo lo que hace falta entre el notebook y ese punto.

El Problema de Producción

El modelo es una fracción pequeña del sistema

Sculley et al. (NeurIPS 2015) observaron en los sistemas de Google que el código de aprendizaje automático ocupa una porción mínima de un sistema real. El resto es infraestructura.

- Configuración
- Recolección de datos
- Verificación de datos
- Extracción de características
- Administración de recursos de cómputo
- Análisis del modelo
- Administración de procesos y metadatos
- Infraestructura de serving
- Monitoreo

El mismo artículo estima que un sistema maduro puede llegar a ser como máximo 5 % código de aprendizaje automático y al menos 95 % código de integración.

Deuda técnica en sistemas de ML

La deuda técnica es el costo futuro de mantenimiento que se acumula por decisiones rápidas tomadas hoy. En ML aparece en formas que el software tradicional no conoce.

- CACE, Changing Anything Changes Everything: cambiar una característica, un hiperparámetro o el muestreo altera el comportamiento de todo el modelo, porque las entradas no están aisladas entre sí.

- Ciclos de retroalimentación ocultos: las predicciones del modelo influyen en los datos con los que se reentrenará, directamente o a través de otro sistema.
- Consumidores no declarados: otros sistemas leen las salidas del modelo sin que el equipo lo sepa, de modo que cualquier mejora puede romperlos.
- Código pegamento: grandes cantidades de código para meter y sacar datos de bibliotecas genéricas, que congelan el sistema alrededor de una herramienta.
- Pipeline jungles: cadenas de extracciones, uniones y archivos intermedios que crecen sin diseño hasta que nadie puede probarlas ni repararlas.
- Rutas experimentales muertas: ramas condicionales de experimentos viejos que quedan dentro del código de producción.

Los autores atribuyen buena parte de esta deuda a la separación entre los roles de investigación y de ingeniería.

La brecha entre desarrollo y operación

Kreuzberger, Kühn y Hirschl (IEEE Access, 2023) señalan que muchos proyectos industriales de ML nunca llegan a producción, y que la causa principal no es la calidad del modelo sino la dificultad de automatizarlo y operarlo.

- El científico de datos entrega un archivo con el modelo entrenado y el equipo de ingeniería lo reimplementa para producción.
- La transformación de datos que se programó en el notebook se vuelve a escribir en otro lenguaje para el servicio.
- Nadie registra con qué datos, qué código y qué hiperparámetros se produjo el modelo que está en producción.
- El modelo se deja de revisar una vez desplegado.

El resultado es un modelo que se actualiza pocas veces al año, que se comporta distinto en producción que en el notebook y que se degrada sin que nadie lo detecte.

MLOps

Definición

MLOps (Machine Learning Operations) es un paradigma que reúne buenas prácticas, conceptos y una cultura de desarrollo para la conceptualización, implementación, monitoreo, despliegue y escalabilidad de productos de aprendizaje automático de extremo a extremo. Es ante todo una práctica de ingeniería que combina tres disciplinas, aprendizaje automático, ingeniería de software (en particular DevOps) e ingeniería de datos. Su objetivo es llevar sistemas de ML a producción cerrando la brecha entre desarrollo y operación.

— Kreuzberger, Kühl y Hirschl, 2023

Las tres disciplinas

- Aprendizaje automático: selección de algoritmos, entrenamiento y evaluación de modelos.
- Ingeniería de software y DevOps: control de versiones, pruebas automáticas, integración y entrega continua.
- Ingeniería de datos: ingesta, transformación, validación y almacenamiento de los datos que alimentan al modelo.

DevOps y MLOps

MLOps hereda las prácticas de DevOps, pero un sistema de ML tiene un tercer insumo que el software tradicional no tiene, los datos.

Aspecto	DevOps	MLOps
Se versiona	Código	Código, datos y modelo
CI prueba	Código y componentes	Además, esquemas de datos, calidad de datos y modelo
CD entrega	Un paquete o servicio	Un pipeline de entrenamiento que a su vez despliega el modelo
Falla	De forma visible (error, caída)	En silencio (predicciones peores sin error)
Exclusivo	—	Entrenamiento continuo (CT)

Un programa que funcionaba ayer funciona hoy mientras no cambie el código. Un modelo que funcionaba ayer puede fallar hoy sin que cambie una sola línea, porque cambiaron los datos que recibe.

El Ciclo de Vida del ML

Las ocho etapas

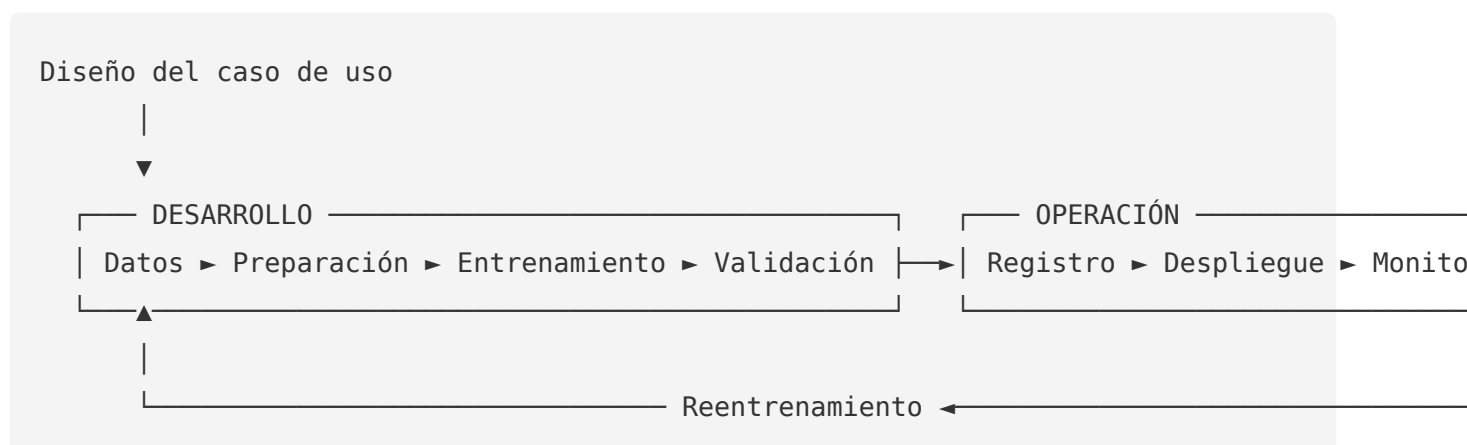
Google Cloud describe el proceso de ciencia de datos de un sistema en producción en ocho etapas. Cada una puede ejecutarse a mano o de forma automática, y el grado de automatización define la madurez del proceso.

1. Extracción de datos: seleccionar e integrar los datos relevantes desde sus fuentes.

2. **Análisis de datos:** análisis exploratorio para entender el esquema y las características de los datos.
3. **Preparación de datos:** limpieza, partición en entrenamiento, validación y prueba, y transformaciones.
4. **Entrenamiento:** implementación de algoritmos y búsqueda de hiperparámetros.
5. **Evaluación:** medición de la calidad sobre un conjunto de prueba reservado.
6. **Validación:** confirmar que el modelo supera una línea base y que es apto para desplegarse.
7. **Serving:** desplegar el modelo como servicio REST, como componente embebido o como sistema de predicción por lotes.
8. **Monitoreo:** medir el desempeño de las predicciones para disparar una nueva iteración.

Ciclo de vida de MLOps

El ciclo se representa como un ciclo infinito. La mitad de desarrollo produce el modelo y la mitad de operación lo mantiene en producción. El monitoreo detecta la deriva y dispara un reentrenamiento que regresa a los datos.



Un proceso iterativo

El ciclo no termina en el despliegue. El monitoreo de la etapa 8 alimenta de nuevo la etapa 1, y el proceso completo se organiza en tres fases.

- Diseño: entender el negocio y los datos, priorizar un caso de uso y definir requisitos, estrategia de serving y pruebas.
- Experimentación y desarrollo: prueba de concepto, iteración sobre datos y modelos hasta obtener un modelo estable.
- Operación: entrega del modelo a producción con pruebas, versionado, entrega continua y monitoreo.

Principios

Kreuzberger et al. identifican nueve principios a partir de una revisión de literatura, una revisión de herramientas y entrevistas con practicantes.

- P1, automatización CI/CD: compilación, pruebas, entrega y despliegue automáticos, con retroalimentación rápida a quien desarrolla.
- P2, orquestación de flujos: las tareas del pipeline se coordinan como un grafo dirigido acíclico que define el orden de ejecución y las dependencias.
- P3, reproducibilidad: un experimento puede repetirse y producir exactamente el mismo resultado.
- P4, versionado: datos, código y modelo tienen versión, lo que permite reproducir y también auditar.
- P5, colaboración: trabajo conjunto sobre datos, modelo y código, con una cultura que reduce los silos entre roles.
- P6, entrenamiento y evaluación continuos: reentrenamiento periódico con datos nuevos, siempre acompañado de una evaluación.

- P7, seguimiento de metadatos: cada ejecución registra fecha, duración, parámetros, métricas y linaje de datos y código.
- P8, monitoreo continuo: evaluación periódica de datos, modelo, código, infraestructura y desempeño de las predicciones.
- P9, ciclos de retroalimentación: lo aprendido en la evaluación y el monitoreo regresa al desarrollo y al reentrenamiento.

La lección de fuentes y preparación de datos planteó la reproducibilidad como una práctica de documentación. En MLOps la reproducibilidad se garantiza con infraestructura, versionando automáticamente cada insumo de cada ejecución.

Automatización

Niveles de madurez

Google Cloud clasifica la madurez de un equipo según cuánto del ciclo está automatizado.

Nivel 0, proceso manual

- Cada paso se ejecuta a mano, normalmente en notebooks.
- El científico de datos entrega un archivo del modelo y otro equipo lo despliega.
- Se publica una versión nueva del modelo un par de veces al año.
- No hay CI ni CD, y las pruebas ocurren dentro del propio notebook.
- Se despliega solo el servicio de predicción, no el sistema que lo produce.
- No hay monitoreo activo de las predicciones.

El riesgo de este nivel es el training-serving skew, la diferencia entre cómo se transforman los datos durante el entrenamiento y cómo se transforman al servir, porque son dos implementaciones distintas escritas por dos equipos distintos.

Nivel 1, automatización del pipeline

- Las etapas de preparación, entrenamiento, evaluación y validación forman un pipeline orquestado.
- El modelo se reentrena automáticamente en producción con datos nuevos.
- El mismo pipeline corre en desarrollo, preproducción y producción.
- Los componentes son modulares y corren en contenedores con su propio entorno.
- Se despliega el pipeline completo, y el pipeline despliega el modelo.

El límite de este nivel aparece cuando el pipeline mismo cambia con frecuencia. Probar y desplegar a mano cada cambio del pipeline vuelve a ser un cuello de botella.

Nivel 2, automatización CI/CD del pipeline

- Cada cambio al código del pipeline pasa por CI, que compila y ejecuta pruebas.
- CD despliega el pipeline nuevo al entorno correspondiente.
- El pipeline desplegado se ejecuta por calendario o por disparador y registra el modelo resultante.
- El modelo registrado se despliega como servicio y su desempeño se monitorea.

En este nivel el análisis de datos y el análisis del modelo siguen siendo manuales. Lo automatizado es todo lo que se repite.

La adopción es gradual. Una organización puede automatizar una etapa a la vez sin saltar de un nivel al siguiente.

Continuous X

- CI, integración continua: además de probar el código, valida esquemas de datos, calidad de datos y modelos.
- CD, entrega continua: entrega un pipeline de entrenamiento que a su vez despliega el servicio de predicción.
- CT, entrenamiento continuo: reentrena y vuelve a desplegar el modelo automáticamente. Es la práctica exclusiva de ML.
- CM, monitoreo continuo: mide datos y métricas del modelo en producción y las relaciona con métricas de negocio.

Disparadores de reentrenamiento

El entrenamiento continuo necesita una señal que decida cuándo correr el pipeline.

- A demanda: una persona lo ejecuta de forma manual.
- Por calendario: diario, semanal o mensual, según la velocidad con que cambian los datos.
- Por disponibilidad de datos: cuando llega un lote nuevo de datos etiquetados.
- Por degradación: cuando una métrica en producción cae por debajo de un umbral.
- Por cambio en la distribución: cuando el monitoreo detecta que los datos de entrada ya no se parecen a los de entrenamiento.

Reentrenar con mayor frecuencia no siempre es mejor. Cada ejecución consume cómputo, y reentrenar sobre datos defectuosos propaga el defecto a producción.

Validación de datos dentro del pipeline

Antes de entrenar, el pipeline compara los datos nuevos contra un esquema construido a partir de los datos de entrenamiento anteriores.

- Schema skew: aparecen columnas inesperadas, faltan columnas o un valor no corresponde a su tipo. El pipeline se detiene para que una persona investigue.
- Data values skew: los datos son válidos pero sus estadísticas cambiaron de forma significativa. El pipeline continúa y reentrena.

Validación del modelo

Un modelo recién entrenado no pasa directamente a producción.

- Validación offline: se evalúa contra el conjunto de prueba, se compara contra el modelo que está en producción y se revisa el desempeño por segmentos de datos.
- Validación de infraestructura: se confirma que el modelo cabe en la memoria y el hardware de destino y que la API responde con el formato esperado.
- Canary deployment: el modelo nuevo recibe una fracción pequeña del tráfico real mientras se vigilan sus errores.
- A/B testing: dos versiones reciben tráfico en paralelo y se comparan sobre una métrica de negocio.

Pruebas automáticas en CI

- Pruebas unitarias de la ingeniería de características, por ejemplo que la codificación one-hot produzca las columnas correctas.
- Pruebas de que el entrenamiento converge, es decir, que la pérdida baja en unas cuantas iteraciones.

- Pruebas de que el entrenamiento no produce valores NaN.
- Pruebas de que cada componente genera los artefactos esperados y de que los componentes se integran entre sí.
- Pruebas de carga del servicio de predicción, en consultas por segundo y latencia.
- Pruebas de que el entrenamiento y el serving producen la misma predicción para la misma entrada.

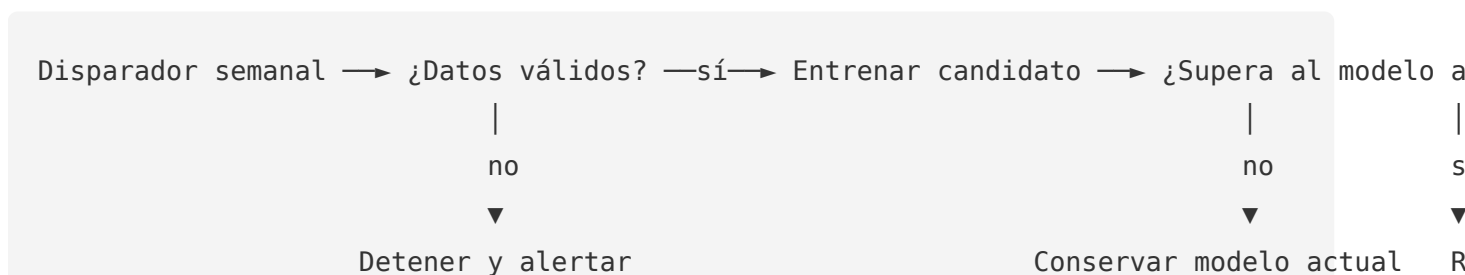
Promoción entre entornos

- Entorno de pruebas: despliegue automático en cada merge.
- Preproducción: despliegue semiautomático, después de la aprobación de un revisor.
- Producción: despliegue manual, después de varias ejecuciones exitosas en preproducción.

Si la versión nueva falla, el model registry conserva la versión anterior y el rollback consiste en volver a apuntar el servicio a ella.

Caso, reentrenamiento automático

Enunciado. Una cadena de supermercados usa un modelo de pronóstico de demanda para decidir cuánto inventario pedir a cada proveedor. El modelo debe reentrenarse cada semana con las ventas recientes, sin intervención humana, y nunca debe reemplazarse por una versión que pronostique peor que la actual.



Lo que gana el negocio con este flujo.

- El pronóstico incorpora cada semana los cambios en los hábitos de compra.
- Un lote de datos defectuoso detiene el proceso antes de llegar a los pedidos.
- Ninguna versión nueva entra a producción si pronostica peor que la actual.
- El equipo de datos dedica su tiempo a mejorar el modelo, no a reentrenarlo a mano.

Componentes Técnicos

Los nueve componentes

- CI/CD: compila, prueba, entrega y despliega. Ejemplos, GitHub Actions y Jenkins.
- Repositorio de código: almacena y versiona el código. Ejemplos, GitHub y GitLab.
- Orquestador de flujos: ejecuta el pipeline como grafo dirigido acíclico. Ejemplos, Apache Airflow, Kubeflow Pipelines y SageMaker Pipelines.
- Feature store: almacén central de características para entrenamiento y predicción. Ejemplos, Feast, Tecton y Hopsworks.
- Infraestructura de entrenamiento: CPU, memoria y GPU, distribuida o no. Ejemplo, Kubernetes.
- Model registry: almacena los modelos entrenados junto con sus metadatos. Ejemplos, MLflow y SageMaker Model Registry.

- Almacén de metadatos: registra parámetros, métricas y linaje de cada ejecución.
- Serving: expone el modelo para predicción en línea o por lotes. Ejemplos, KServe, TensorFlow Serving y Vertex AI.
- Monitoreo: vigila el desempeño del modelo y de la infraestructura. Ejemplos, Prometheus con Grafana y Evidently.

Feature store

Un feature store define cada característica una sola vez y la sirve desde dos bases de datos.

- Almacén offline: guarda el histórico de características con latencia normal, para entrenamiento y experimentación.
- Almacén online: guarda el valor más reciente con latencia baja, para predicciones en producción.

Como el entrenamiento y la predicción leen la misma definición, el training-serving skew desaparece por construcción. Además, una característica calculada por un equipo queda disponible para los demás. Uber reportó en 2017 cerca de diez mil características en el feature store de su plataforma Michelangelo.

Model registry y metadatos

Cada ejecución del pipeline registra lo necesario para reconstruir y auditar el modelo que produjo.

- Versión del código y de los datos de entrenamiento.
- Parámetros e hiperparámetros.
- Métricas de entrenamiento y de prueba.

- Referencias a los artefactos intermedios, lo que permite reanudar una ejecución desde el paso que falló.
- Referencia al modelo anterior, lo que permite hacer rollback.
- Estado del modelo dentro del ciclo, por ejemplo candidato, en preproducción, en producción o archivado.

Monitoreo y Deriva

La degradación silenciosa

Un servicio web que falla devuelve un error. Un modelo que falla sigue devolviendo predicciones con el formato correcto, solo que equivocadas. Ninguna alerta de infraestructura lo detecta.

Tipos de deriva

- Data drift: cambia la distribución de las variables de entrada. Por ejemplo, una aplicación de crédito empieza a recibir solicitantes más jóvenes después de una campaña en redes sociales.
- Concept drift: cambia la relación entre las entradas y la variable objetivo. Por ejemplo, el mismo patrón de compra que antes indicaba fraude deja de indicarlo cuando cambian los hábitos de pago.
- Prediction drift: cambia la distribución de las predicciones del modelo, aunque todavía no se conozcan las etiquetas reales.
- Training-serving skew: los datos en producción se transforman distinto que en el entrenamiento por un defecto de implementación, no por un cambio en el mundo.

El retraso de las etiquetas

Medir la exactitud en producción exige conocer la respuesta correcta, y esa respuesta suele llegar tarde. Un modelo de impago de préstamos conoce su error meses después de la predicción. Por eso el monitoreo combina dos tipos de señales.

- Señales inmediatas: distribución de las entradas, distribución de las predicciones, valores faltantes y latencia.
- Señales diferidas: exactitud, precisión, recall y F1 calculadas cuando llegan las etiquetas reales.

Sesgo de predicción

Sculley et al. proponen una verificación sencilla. En un sistema que funciona, la proporción de clases predichas debe coincidir con la proporción de clases observadas. Si el modelo predice fraude en 2 % de las transacciones y el fraude real es 0.5 %, algo cambió, aunque esta prueba no basta por sí sola, porque un modelo que predice siempre el promedio también la cumple.

Qué monitorear

- Cambios en las dependencias, como la versión de los datos, los sistemas fuente o las bibliotecas.
- Datos de entrada contra el esquema de entrenamiento.
- Coincidencia entre las características de entrenamiento y las de serving.
- Valores NaN o infinitos.
- Uso de recursos de cómputo, memoria, red y disco, para estimar costos.
- Edad del modelo, porque los modelos más viejos tienden a degradarse.
- Calidad predictiva sobre los datos servidos, a lo largo del tiempo.

Roles

MLOps es un proceso interdisciplinario. Kreuzberger et al. identifican siete roles.

- Stakeholder de negocio: define el objetivo de negocio y comunica el retorno de la inversión.
- Arquitecto de soluciones: diseña la arquitectura y elige las tecnologías.
- Científico de datos: traduce el problema de negocio en un problema de ML y desarrolla el modelo.
- Ingeniero de datos: construye los pipelines de datos y alimenta el feature store.
- Ingeniero de software: aplica patrones de diseño y buenas prácticas para convertir el modelo en producto.
- Ingeniero DevOps: administra CI/CD, orquestación, despliegue y monitoreo.
- Ingeniero de ML o ingeniero de MLOps: combina las habilidades de los roles anteriores, construye y opera la infraestructura de ML y los pipelines automatizados.

Beneficios

- Velocidad: el tiempo entre una idea y un modelo en producción se reduce, porque el camino a producción ya existe y no se reconstruye en cada proyecto.
- Confiabilidad: las pruebas automáticas, la validación y los despliegues graduales detectan defectos antes de que lleguen al usuario.
- Reproducibilidad: cualquier modelo en producción puede reconstruirse a partir de su versión de código, datos y parámetros.

- Trazabilidad: el linaje de cada modelo responde qué datos lo entrenaron, quién lo aprobó y cuándo se desplegó.
- Escalabilidad: un mismo equipo puede operar decenas o cientos de modelos, porque el trabajo repetido está automatizado.
- Vigencia del modelo: el monitoreo y el entrenamiento continuo mantienen el desempeño mientras los datos cambian.
- Colaboración: científicos de datos e ingenieros trabajan sobre el mismo pipeline en lugar de reimplementar el trabajo del otro.

Impacto en el Negocio

Del modelo al valor

Un modelo genera retorno únicamente mientras está en producción y sus predicciones son correctas. Toda la inversión previa en datos, personal y cómputo queda sin retorno si el modelo nunca se despliega o si se degrada sin que nadie lo note.

- Tiempo de llegada al mercado: cada semana que un modelo pasa esperando despliegue es una semana sin beneficio.
- Costo de operación: sin automatización, cada modelo nuevo agrega trabajo manual permanente, y el costo crece con el número de modelos.
- Costo del error: un modelo degradado toma decisiones equivocadas a escala, sobre miles de clientes al mismo tiempo.

Métricas técnicas y métricas de negocio

Una mejora en la métrica del modelo no garantiza una mejora en el negocio. MLOps exige verificar que la función de pérdida se correlacione con la métrica que la organización persigue.

Métrica técnica	Métrica de negocio asociada
Recall del detector de fraude	Monto de pérdidas evitadas
Precisión del modelo de abandono	Costo de las campañas de retención
Error del pronóstico de demanda	Inventario sobrante y ventas perdidas
Latencia del recomendador	Tasa de conversión de la página

El A/B testing es el mecanismo que conecta las dos columnas, porque mide el efecto del modelo sobre la métrica de negocio con tráfico real.

Riesgo, cumplimiento y auditoría

- Las regulaciones sobre inteligencia artificial, como el AI Act de la Unión Europea, exigen documentación técnica y registros de funcionamiento para los sistemas de alto riesgo.
- Un banco que rechaza un crédito con un modelo debe poder explicar qué versión del modelo tomó la decisión y con qué datos fue entrenada.
- El versionado y el seguimiento de metadatos producen esa evidencia como subproducto del pipeline, sin trabajo adicional.

Caso, Uber Michelangelo

Antes de Michelangelo, los equipos de Uber entrenaban modelos con herramientas distintas, limitados a lo que cabía en la computadora del científico de datos, y cada proyecto construía su propio contenedor de serving. No existía un camino establecido a producción.

Michelangelo estandarizó el flujo en seis pasos, administración de datos, entrenamiento, evaluación, despliegue, predicción y monitoreo. Según el reporte

de la empresa en 2017, la plataforma atendía decenas de equipos, con cerca de diez mil características compartidas en su feature store y modelos que servían más de 250 000 predicciones por segundo. El tiempo estimado de entrega de UberEATS es uno de los modelos que corrían sobre ella.

Caso, Zillow Offers

Zillow Offers compraba casas a un precio fijado por modelos de valuación y las revendía. En noviembre de 2021 la empresa cerró el programa. Su reporte anual de 2021 registra una reducción de inventario de 407.9 millones de dólares, y el cierre implicó recortar cerca de 25 % de su personal. El director ejecutivo explicó que la impredecibilidad del pronóstico de precios superó por mucho lo que la empresa había anticipado.

El caso muestra que el riesgo de un modelo en producción no es técnico sino financiero. Cuando las predicciones se convierten automáticamente en decisiones de compra, el tiempo que tarda una organización en detectar que el modelo ya no refleja el mercado se mide en dinero.

Adopción Gradual

ML Test Score

Breck et al. (Google, 2017) proponen una rúbrica para medir qué tan listo para producción está un sistema de ML. Se evalúan cuatro secciones, pruebas de datos, pruebas del modelo, pruebas de infraestructura y monitoreo.

- Cada prueba ejecutada a mano y documentada suma 0.5 puntos.
- Cada prueba ejecutada automáticamente de forma repetida suma 1 punto.
- El puntaje final es el mínimo de las cuatro secciones.

Puntaje	Interpretación
0	Más cercano a un proyecto de investigación
(0, 1]	Algunas pruebas, con huecos probables de confiabilidad
(1, 2]	Productización básica
(2, 3]	Razonablemente probado
(3, 5]	Pruebas y monitoreo automatizados sólidos

Como el puntaje es el mínimo, la mejora se concentra primero en la sección más débil.

Por dónde empezar

1. Versionar código, datos y modelos.
2. Registrar cada experimento con sus parámetros y métricas.
3. Convertir el notebook en un pipeline de pasos ejecutables.
4. Agregar validación de datos antes del entrenamiento.
5. Automatizar el despliegue y el monitoreo de las predicciones.
6. Agregar el entrenamiento continuo con disparadores.

Herramientas del Ecosistema

- Seguimiento y registro: MLflow, Weights & Biases.
- Versionado de datos: DVC, lakeFS.
- Orquestación: Apache Airflow, Kubeflow Pipelines, Prefect.

- Feature store: Feast, Tecton, Hopsworks.
- Serving: KServe, BentoML, TensorFlow Serving.
- Monitoreo: Evidently, Prometheus con Grafana.
- Plataformas integradas: Amazon SageMaker, Google Vertex AI, Azure Machine Learning, Databricks.

Referencias

- Sculley, D. et al. (2015). Hidden Technical Debt in Machine Learning Systems. NeurIPS.
- Breck, E. et al. (2017). The ML Test Score, A Rubric for ML Production Readiness and Technical Debt Reduction. IEEE Big Data.
- Kreuzberger, D., Kühl, N. y Hirschl, S. (2023). Machine Learning Operations (MLOps), Overview, Definition, and Architecture. IEEE Access.
- Google Cloud. MLOps, Continuous delivery and automation pipelines in machine learning.
- ml-ops.org. MLOps Principles.
- Hermann, J. y Del Balso, M. (2017). Meet Michelangelo, Uber's Machine Learning Platform. Uber Engineering.
- Zillow Group (2022). Form 10-K, fiscal year 2021.