Desarrollo de una Librería MLOps: Versionamiento, Trazabilidad y Automatización del Ciclo de Vida de Modelos en Entornos Big Data

Presentado para obtener el título de

Magíster en matemáticas aplicadas y ciencias de la computación

Bryam Camilo Acevedo Orjuela

Dirección:
Nicolás Guillermo Avilán Vargas

Universidad del Rosario
Escuela de Ciencias e ingeniería
Maestría en matemáticas aplicadas y ciencias de la computación

ABSTRACT

Este proyecto presenta el desarrollo de MomentumML, una librería especializada en MLOps para gestionar de forma integral el ciclo de vida de modelos de machine learning en entornos distribuidos. Su propósito fue resolver los desafíos que enfrentan las organizaciones al llevar modelos a producción, donde estudios previos reportan que más del 90% de los modelos desarrollados no alcanzan entornos productivos estables.

La solución se construyó sobre PySpark y MLflow, con 34,582 líneas de código distribuidas en módulos especializados para preprocesamiento (10 clases Transformer), entrenamiento (5 clases Estimator con soporte para 8 algoritmos), registro y versionado automático en Unity Catalog, predicción, y monitoreo de drift mediante técnicas estadísticas multimodales (PSI, Kolmogorov-Smirnov, Jensen-Shannon Divergence, entre otras).

El enfoque metodológico integró principios de ingeniería de software, aprendizaje automático y gestión de datos, adoptando un diseño modular basado en el patrón Transformer/Estimator de Spark ML que garantiza serialización nativa y reproducibilidad. Se realizó una evaluación comparativa con herramientas existentes (MLflow Recipes, ZenML, Kubeflow, Databricks Asset Bundles) que justificó el desarrollo de una solución propia adaptada al contexto organizacional.

Los resultados obtenidos durante seis meses de operación demuestran impacto cuantificable: reducción del 81% en líneas de código requeridas para flujos end-to-end (de 395 a 74 líneas), disminución del 71% en tiempo de despliegue (de 3-4 semanas a 5-7 días), incremento del 740% en frecuencia de despliegue (de 0.5 a 4.2 modelos/mes), reducción del 77% en tasa de fallos (de 35% a 8%), y mejora del 92% en tiempo de recuperación (de 48 a 4 horas). En el contexto organizacional, 35 de 85 modelos (41.2%) lograron transitar exitosamente al entorno de QA, un logro inédito comparado con el estado inicial donde ningún modelo había alcanzado esta etapa. Adicionalmente, se logró una reducción del 40% en consumo de DBUs (Databricks Units).

La validación de los indicadores del objetivo general confirmó: trazabilidad completa con 47 experimentos registrados y 23 versiones en Unity Catalog; reproducibilidad exacta (0%

diferencia) en 3 modelos replicados; y escalabilidad lineal hasta 10 millones de registros con tiempos de inferencia estables (<1.5ms por registro).
La librería representa una contribución práctica al campo de MLOps, ofreciendo una solución integral, escalable y basada en tecnologías abiertas que permite reducir tiempos de entrega, mejorar la calidad de los modelos y optimizar su operación en producción.

PALABRAS CLAVE

MLOps, Machine Learning, PySpark, MLflow, Data Drift, Ciclo de Vida de Modelos, Automatización, Trazabilidad, Reproducibilidad, Big Data

TABLA DE CONTENIDO

ABSTRACT 1
TABLA DE CONTENIDO 3
LISTA DE TABLAS 4
LISTA DE FIGURAS 5
Capítulo 1 INTRODUCCIÓN 6
Capítulo 2 OBJETIVOS 10
2.1 Objetivo general 10
2.2 Objetivos específicos 10
Capítulo 3 PROBLEMA Y JUSTIFICACIÓN 11
3.1 Análisis de Alternativas y Justificación Arquitectónica 14
Capítulo 4 MARCO TEÓRICO Y ESTADO DEL ARTE 17
4.1 Análisis Crítico de Herramientas de Monitoreo y Detección de Drift 21
Capítulo 5 METODOLOGÍA 24
5.1 Arquitectura General del Sistema 27
5.2 Descripción de Módulos 29
5.2.1 Módulo de Preprocesamiento 29
5.2.2 Módulo de Entrenamiento 31
5.2.3 Módulo de Monitoreo y Detección de Drift 32
5.3 Decisiones de Diseño 34
Capítulo 6 RESULTADOS Y DISCUSIÓN 36
6.1 Validación Comparativa: MomentumML vs. APIs Nativas 42
6.1.1 Reducción de Complejidad de Código 42
6.1.2 Métricas de Eficiencia Operacional (Framework DORA-ML) 43
6.1.3 Análisis de Overhead Computacional 44
6.2 Validación de Trazabilidad, Reproducibilidad y Escalabilidad 45
6.2.1 Validación de Trazabilidad 45
6.2.2 Validación de Reproducibilidad 46
6.2.3 Validación de Escalabilidad 47
Capítulo 7 CONCLUSIONES Y TRABAJO FUTURO 48
REFERENCIAS 51

LISTA DE TABLAS

Tabla 1. Comparación entre Flujo Tradicional y Solución Propuesta de MLOps.....12
Tabla 2. Evaluación comparativa de herramientas MLOps existentes.....15
Tabla 3. Conceptos Fundamentales en MLOps.....19
Tabla 4. Análisis comparativo de herramientas de detección de drift.....22
Tabla 5. Fases Metodológicas del Proyecto de Desarrollo de la Librería MLOps.....25
Tabla 6. Distribución de código por módulo funcional de MomentumML.....28
Tabla 7. Clases Transformer del módulo de preprocesamiento.....30
Tabla 8. Clases Estimator del módulo de entrenamiento.....31
Tabla 9. Técnicas de detección de drift implementadas en DataDriftDetector.....33
Tabla 10. Comparación de líneas de código (LoC) por operación.....42
Tabla 11. Métricas DORA adaptadas al contexto de MLOps.....43
Tabla 12. Análisis de overhead introducido por la capa de abstracción.....44
Tabla 13. Indicadores de trazabilidad obtenidos durante el período de evaluación.....45
Tabla 14. Pruebas de reproducibilidad de modelos desplegados en producción.....46
Tabla 15. Pruebas de escalabilidad con volúmenes crecientes de datos.....47

LISTA DE FIGURAS

Figura 1. Tasa de éxito en proyectos de machine learning 13
Figura 2. Ciclo de vida gestionado por la librería MLOps 14
Figura 3. Marco conceptual de MLOps como intersección disciplinar 18
Figura 4. Flujo modular de la librería MomentumML 26
Figura 5. Flujo Metodológico General del Proyecto 35
Figura 6. Impacto de la Librería MLOps en el Paso a Producción 38
Figura 7. Optimización de Costos de Computación en Databricks 40
Figura 8. Arquitectura Funcional de la Librería MLOps 41
Figura 9. Impacto de la Librería MLOps en Indicadores Clave 41

Capítulo 1
INTRODUCCIÓN

El crecimiento exponencial del uso de modelos de *machine learning* (ML) en múltiples sectores ha marcado un hito en la transformación digital de organizaciones a nivel global. Industrias como la salud, las finanzas, el comercio minorista y la logística han adoptado soluciones basadas en ML para automatizar procesos, mejorar la toma de decisiones y generar ventajas competitivas. Sin embargo, a pesar de los avances técnicos en algoritmos y capacidades de cómputo, llevar modelos de ML desde su fase de desarrollo hasta entornos productivos sigue siendo uno de los principales desafíos operativos.

Estudios recientes revelan que una proporción considerable de proyectos de *machine learning* no logra superar la etapa experimental. Shankar et al. (2024) señalan que cerca del 90 % de los modelos desarrollados no llegan a producción, lo que pone en evidencia barreras estructurales en la adopción efectiva de inteligencia artificial. Este fenómeno se debe, en gran parte, a la falta de herramientas integrales que faciliten la transición entre el desarrollo experimental de los modelos y su operación confiable en sistemas reales. De forma complementaria, un informe técnico de Google Cloud indica que el 72 % de las organizaciones no logra desplegar ni un solo modelo de IA en producción, a menudo por limitaciones técnicas, organizacionales o de infraestructura (Salama, Kazmierczak, & Schut, 2021).

En este contexto, ha emergido el enfoque de MLOps (*Machine Learning Operations*), inspirado en los principios de DevOps y orientado a la gestión del ciclo de vida completo de modelos de ML. MLOps busca integrar las disciplinas de ciencia de datos, ingeniería de software y operaciones para garantizar reproducibilidad, automatización, trazabilidad y mantenimiento continuo de los modelos desplegados. Su objetivo es brindar soporte técnico y organizacional para que los modelos no solo se entrenen con éxito, sino que operen de forma confiable en producción y se mantengan alineados con la dinámica cambiante de los datos.

A pesar de su promesa, la mayoría de las herramientas disponibles en el ecosistema actual de MLOps son fragmentadas. Muchas están diseñadas para resolver tareas puntuales como el entrenamiento de modelos, el despliegue o el monitoreo, pero no ofrecen una cobertura integral del flujo de trabajo. Esta fragmentación genera inconsistencias entre fases, dificulta la trazabilidad y aumenta lo que se denomina *deuda técnica* (Sculley et al., 2015). El código

auxiliar, los sistemas de validación ad hoc y la falta de abstracciones estandarizadas crean arquitecturas frágiles, con dificultades para escalar o replicar experimentos de forma confiable. Además, la falta de integración entre los componentes técnicos complica la colaboración entre equipos de ingeniería y ciencia de datos, lo cual es fundamental en proyectos de alto impacto (Amershi et al., 2019).

Uno de los desafíos más recurrentes es la ausencia de mecanismos centralizados para el **versionado de modelos y datos**. La trazabilidad, tanto de los experimentos como de los datasets utilizados, es crítica para evaluar el comportamiento histórico de los modelos y realizar auditorías o análisis comparativos. Zhao et al. (2024) realizaron un estudio empírico sobre gestión de activos en ML y encontraron que gran parte de las organizaciones carece de estrategias robustas para controlar las versiones de los modelos y sus artefactos asociados. Esta carencia compromete la reproducibilidad, obstaculiza la depuración de errores y limita la capacidad de realizar mejoras iterativas controladas.

Adicionalmente, en entornos reales, los modelos enfrentan dinámicas impredecibles de los datos. El rendimiento de un modelo puede degradarse debido a la aparición de nuevos patrones, cambios estacionales, o transformaciones en la población de entrada, fenómenos conocidos como *data drift* y *concept drift*. El monitoreo continuo del comportamiento del modelo es, por tanto, esencial para garantizar su vigencia operativa. Google, en su guía para MLOps (Salama et al., 2021), insiste en la necesidad de implementar sistemas de monitoreo con métricas específicas que permitan detectar estas variaciones, ajustando o reentrenando los modelos cuando sea necesario. Eken et al. (2023), en su revisión multivocal sobre prácticas de MLOps, identifican el monitoreo post-despliegue como una de las áreas más críticas y menos estandarizadas en los flujos actuales de ML.

Otro punto crítico es la integración de prácticas sólidas de **ingeniería de software** en proyectos de *machine learning*. Mientras que los científicos de datos se enfocan en la experimentación, los entornos de producción requieren estabilidad, modularidad y pruebas automatizadas. Sculley et al. (2015) destacan que muchas fallas en proyectos de ML no provienen de los algoritmos en sí, sino de la falta de ingeniería alrededor del modelo: configuraciones inconsistentes, dependencia de datos externos no versionados y ausencia de validaciones estructuradas. Por ello, MLOps también implica un cambio cultural, donde se

adopten buenas prácticas de desarrollo de software como control de versiones, pruebas continuas y automatización del despliegue dentro del ciclo de vida del modelo.

Frente a este panorama, se hace evidente la necesidad de desarrollar soluciones integrales que permitan una gestión unificada de modelos de ML, desde su diseño hasta su monitoreo post-despliegue. El presente proyecto responde a esta necesidad mediante el diseño e implementación de una librería modular de MLOps, construida sobre tecnologías ampliamente utilizadas como PySpark y MLflow. La elección de estas herramientas responde a su capacidad para operar con grandes volúmenes de datos, su compatibilidad con entornos distribuidos y su aceptación en proyectos de ciencia de datos a nivel empresarial.

La librería propuesta tendrá como objetivo facilitar la automatización de tareas clave como la preparación y validación de datos, el entrenamiento y reentrenamiento de modelos, el registro y versionado de artefactos, el despliegue en entornos productivos y el monitoreo de desempeño. Además, se desarrollará bajo principios de diseño modular y reproducible, permitiendo su integración en flujos de trabajo ya existentes, y promoviendo buenas prácticas de ingeniería y colaboración interdisciplinaria.

Este enfoque no solo busca ofrecer una solución técnica, sino también contribuir a la consolidación de una cultura de MLOps dentro de los equipos que desarrollan y mantienen modelos de *machine learning*. Al estandarizar procesos y automatizar tareas repetitivas, se pretende reducir el tiempo necesario para llevar modelos a producción, mejorar la calidad de los entregables y aumentar la confiabilidad operativa de los sistemas inteligentes.

En suma, este proyecto se sitúa en la intersección entre ciencia de datos, ingeniería de software y operaciones, abordando problemáticas reales que limitan actualmente la adopción de inteligencia artificial a escala. Al ofrecer una herramienta integral, escalable y basada en tecnologías abiertas, se espera generar un aporte significativo tanto para el avance técnico como para la implementación práctica de modelos de *machine learning* en entornos productivos.

Finalmente, este trabajo busca responder preguntas clave que muchas organizaciones enfrentan: ¿Cómo escalar la gestión de modelos cuando se trabaja con grandes volúmenes de datos? ¿Qué tan reproducibles y auditables son realmente los modelos que llegan a producción? ¿Cómo mitigar el impacto de los cambios en los datos del entorno sobre el rendimiento del modelo? Estas y otras interrogantes guían el desarrollo de la presente propuesta, invitando al

lector a explorar en profundidad las decisiones técnicas, metodológicas y operativas que sustentan esta solución.

Capítulo 2
OBJETIVOS

1.1 Objetivo general

Desarrollar una librería integral fundamentada en principios de MLOps, orientada a la gestión eficiente y automatizada del ciclo de vida de modelos de machine learning, abarcando desde la preparación y validación de datos hasta el despliegue, monitoreo y mantenimiento en entornos productivos, con el fin de garantizar la trazabilidad, reproducibilidad y escalabilidad de los modelos en contextos reales de operación.

1.2 Objetivos específicos

1. Diseñar una arquitectura modular y escalable que integre de forma cohesionada funcionalidades para la preparación de datos, entrenamiento, versionado, implementación y monitoreo continuo de modelos de machine learning, asegurando su compatibilidad con entornos distribuidos mediante el uso de PySpark y MLflow.
2. Desarrollar herramientas automatizadas para la detección temprana de desviaciones en el comportamiento de los modelos, tales como data drift y pérdida de precisión, que permitan una respuesta proactiva frente a cambios en los datos o el contexto operacional.
3. Validar la efectividad de la librería a través de su aplicación en casos de uso reales, evaluando su impacto en términos de reducción de tiempos operativos, mejora en la calidad de los modelos y optimización del proceso de gestión en distintos entornos productivos.

Capítulo 3
PROBLEMA Y JUSTIFICACIÓN

El crecimiento acelerado del uso de modelos de machine learning (ML) ha transformado numerosos sectores, desde la salud hasta las telecomunicaciones, pasando por la banca, la logística y los servicios públicos. Estos modelos permiten automatizar decisiones complejas, personalizar servicios y predecir eventos con una precisión sin precedentes. Sin embargo, a pesar de los avances en técnicas de modelado y en el acceso a grandes volúmenes de datos, la capacidad de desplegar y mantener modelos en producción de forma eficiente continúa siendo una de las principales barreras técnicas y operativas que enfrentan las organizaciones.

Según Shankar et al. (2024), cerca del 90 % de los modelos desarrollados no llegan a ser implementados en entornos productivos. Esta alarmante cifra refleja no solo una brecha técnica, sino también una falta de herramientas y procesos robustos para gestionar el ciclo de vida completo de los modelos. Complementando esta visión, un informe de Google Cloud (Salama, Kazmierczak & Schut, 2021) indica que más del 70 % de las organizaciones no logra desplegar un solo modelo de forma exitosa, lo que sugiere que el problema no reside únicamente en la calidad del modelo, sino en la infraestructura de soporte que lo rodea.

A este desafío se suma la fragmentación de herramientas y procesos. Muchos equipos científicos emplean herramientas específicas para tareas aisladas —como entrenamiento, validación o monitoreo— sin una integración coherente entre ellas. Esta fragmentación aumenta la deuda técnica (Sculley et al., 2015), genera errores de configuración y dificulta la trazabilidad de los modelos en entornos reales. En palabras de Amershi et al. (2019), “la ingeniería de software aplicada a proyectos de ML requiere una sinergia constante entre los desarrolladores y los científicos de datos, lo cual aún no se logra plenamente en la práctica”.

Uno de los retos más críticos en la operación de modelos es el control de versiones y la trazabilidad de los datos y artefactos. Zhao et al. (2024) evidencian que la falta de estrategias estructuradas para versionar modelos y datasets conduce a la pérdida de reproducibilidad, y limita la capacidad de auditar resultados o corregir errores de forma retrospectiva. Esta limitación afecta no solo la eficiencia técnica, sino también el cumplimiento regulatorio, especialmente en sectores donde las decisiones automatizadas deben ser justificables, como la salud, el crédito o el derecho penal.

El monitoreo posterior al despliegue es otro punto débil en los sistemas actuales. Una vez que el modelo entra en producción, su rendimiento puede deteriorarse debido a fenómenos como data drift o concept drift, es decir, cambios en la distribución de los datos o en la relación entre variables predictoras y objetivo. Eken et al. (2023) identifican esta falta de supervisión continua como una de las principales razones del fracaso operativo de los modelos desplegados, y subrayan la necesidad de sistemas automatizados de alerta y reentrenamiento.

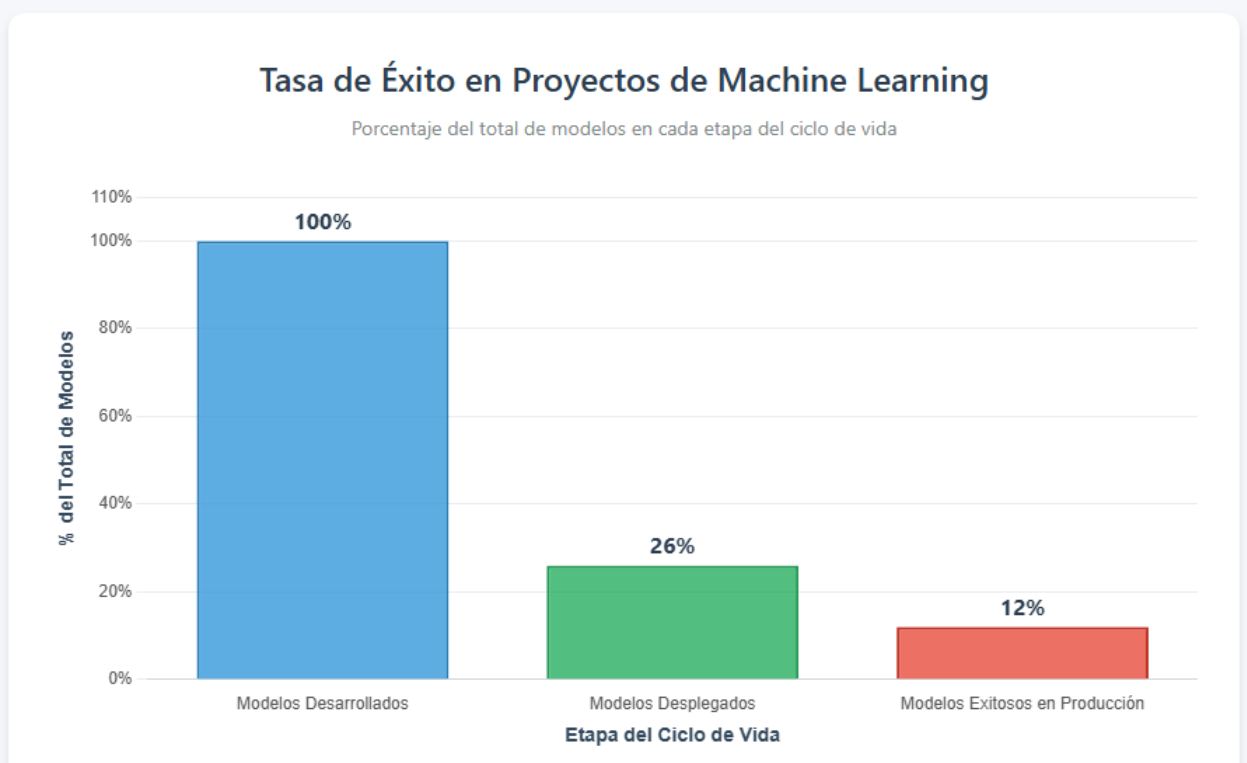
En este contexto, el presente proyecto plantea el desarrollo de una librería integral basada en principios de MLOps, cuyo propósito es automatizar y estandarizar la gestión completa del ciclo de vida de modelos de machine learning. Esta solución está construida sobre tecnologías abiertas y ampliamente utilizadas como PySpark y MLflow, lo que garantiza su interoperabilidad y escalabilidad en contextos empresariales reales. La librería integra funciones para la preparación de datos, validación, entrenamiento y reentrenamiento de modelos, control de versiones, despliegue en múltiples entornos y monitoreo continuo del rendimiento.

El impacto técnico de esta solución se traduce en mayor eficiencia operativa, reducción de errores manuales y mejora en la estabilidad de los modelos en producción. Además, la integración de métricas de rendimiento y monitoreo activo facilita una respuesta proactiva frente a cambios en los datos, lo cual es clave para preservar la precisión y confiabilidad de los modelos a lo largo del tiempo, (Figura 1).

La problemática descrita se evidencia cuantitativamente en la Figura 1, que presenta la distribución porcentual de modelos en cada etapa del ciclo de vida ML. Los datos muestran que, aunque el 100% de los modelos completan la fase de desarrollo, únicamente el 26% alcanza la etapa de despliegue, y solo el 12% logra mantenerse exitosamente en producción. Esta brecha entre desarrollo y operación sustenta la necesidad de herramientas MLOps estructuradas.

Figura 1.

Tasa de éxito en proyectos de machine learning.



Nota: Distribución porcentual de modelos en cada etapa del ciclo de vida ML. Los datos muestran que, aunque el 100% de los modelos completan la fase de desarrollo, únicamente el 26% alcanza la etapa de despliegue, y solo el 12% logra mantenerse exitosamente en producción, Análisis propio (2024-2025).

Desde el punto de vista económico, la automatización del ciclo de vida reduce significativamente el tiempo y los recursos requeridos para mantener modelos operativos. Según estimaciones de Google Cloud (2021), los equipos que adoptan prácticas maduras de MLOps pueden reducir los ciclos de experimentación-producción en hasta un 40 %, y disminuir el costo de mantenimiento en más del 25 %. Estos beneficios son especialmente relevantes para organizaciones que despliegan múltiples modelos simultáneamente y que requieren responder de forma ágil a cambios del mercado.

En cuanto al impacto ambiental, si bien el entrenamiento de modelos complejos puede tener un alto consumo energético, la implementación de un sistema de monitoreo inteligente

evita ejecuciones innecesarias o retrainings injustificados. Al hacer un uso más racional de la infraestructura computacional, la solución contribuye a reducir la huella de carbono asociada al procesamiento de datos, alineándose con principios de sostenibilidad tecnológica.

Socialmente, esta solución democratiza el acceso a prácticas avanzadas de MLOps. Al ser de código abierto, reutilizable y compatible con tecnologías ampliamente adoptadas, puede ser utilizada por equipos pequeños o medianos que carecen de plataformas empresariales completas, permitiendo una participación más equitativa en la revolución de la inteligencia artificial.

Desde la perspectiva política, el trabajo se alinea con políticas públicas de transformación digital. En Colombia, por ejemplo, el documento CONPES 4069 "Política Nacional de Ciencia, Tecnología e Innovación" promueve la incorporación de inteligencia artificial en sectores estratégicos como salud, educación, justicia y transporte. Una herramienta como la aquí propuesta puede ser adoptada por instituciones estatales o públicas que deseen sistematizar y escalar el uso de modelos predictivos con garantías de transparencia y control.

Tabla 1

Comparación entre Flujo Tradicional y Solución Propuesta de MLOps

ASPECTO	*FLUJO TRADICIONAL*	*SOLUCIÓN PROPPUESTA*
Versionado de modelos	Manual, propenso a errores	Automático con MLflow
Preparación de datos	Sin validación estandarizada	Validación modular reproducible
Detección de data drift	Inexistente o reactiva	Proactiva mediante métricas en tiempo real
Reentrenamiento	Manual o ineficiente	Automatizado y controlado
Reproducibilidad de experimentos	Limitada	Garantiza con trazabilidad completa
Tiempo del ciclo completo	Elevado (semanas o meses)	Reducido (días)

Nota: Análisis comparativo de las características operacionales entre el enfoque tradicional de gestión de modelos ML y la arquitectura MLOps propuesta en este trabajo. Se destacan las ventajas de la

solución en términos de automatización, trazabilidad y reducción de tiempos de ciclo, pasando de semanas-meses a días. Fuente: elaboración propia.

A diferencia de los flujos tradicionales, donde cada fase del desarrollo de modelos suele gestionarse de forma manual o aislada, la solución propuesta adopta un enfoque integral basado en principios de MLOps. Esta integración permite orquestar de manera automatizada todas las etapas del ciclo de vida del modelo, desde la ingesta y validación de datos hasta el despliegue y monitoreo continuo en producción.

El diseño modular de la librería no solo favorece la reutilización y escalabilidad, sino que también asegura que cada componente del proceso pueda ser evaluado, ajustado y registrado de forma trazable. Esta estructuración facilita la intervención proactiva frente a desviaciones en el comportamiento del modelo y permite implementar políticas de reentrenamiento adaptativo, lo cual es fundamental en contextos donde los datos cambian dinámicamente.

La Figura 2 presenta la arquitectura del pipeline end-to-end implementado en la solución propuesta. El flujo comprende seis fases principales que automatizan el ciclo completo: (1) adquisición de datos, (2) validación y preprocesamiento, (3) entrenamiento y optimización, (4) registro y versionado, (5) despliegue automatizado, y (6) monitoreo en producción. La retroalimentación continua permite la detección de drift y el reentrenamiento automatizado, cerrando el ciclo de gestión del modelo.

Figura 2.

Ciclo de vida gestionado por la librería MLOps.

Nota: Arquitectura del pipeline end-to-end implementado en la solución propuesta. El flujo comprende seis fases principales que automatizan el ciclo completo desde la adquisición de datos hasta el monitoreo en producción. La retroalimentación continua permite la detección de drift y el reentrenamiento automatizado, cerrando el ciclo de gestión del modelo. Fuente: elaboración propia.

3.1 Análisis de Alternativas y Justificación Arquitectónica

Antes de proceder con el desarrollo de una librería propia, se realizó una evaluación sistemática de las herramientas de orquestación MLOps disponibles en el mercado. Esta evaluación se fundamentó en criterios técnicos y operacionales específicos del contexto de telecomunicaciones, donde el procesamiento de datasets con más de 10 millones de registros y la integración nativa con PySpark constituyen requisitos no negociables.

Tabla 2

Evaluación comparativa de herramientas MLOps existentes

HERRAMIENTA	FORTALEZAS	LIMITACIONES PARA EL CASO DE USO
MLflow Recipes	Flujos predefinidos, integración nativa con MLflow	Rigidez en personalización; no soporta pipelines complejos con múltiples técnicas de imputación y balanceo
Databricks Asset Bundles	Despliegue declarativo, integración con Unity Catalog	Curva de aprendizaje elevada; no abstrae complejidad de PySpark ML
ZenML	Agnóstico de infraestructura, diseño modular	Menor madurez con PySpark; overhead de configuración para Databricks
Kubeflow Pipelines	Escalabilidad en Kubernetes, robustez empresarial	Requiere infraestructura K8s dedicada; incompatible con stack existente
Metaflow (Netflix)	Simplicidad, versionado de datos integrado	Orientado a AWS; integración limitada con Azure Databricks

Nota: Evaluación técnica realizada durante la fase de análisis del proyecto. Se consideraron criterios de integración con PySpark, compatibilidad con Unity Catalog, curva de aprendizaje y flexibilidad de configuración. Las limitaciones identificadas justifican el desarrollo de una solución propia adaptada al contexto organizacional. Fuente: elaboración propia.

La decisión de implementar una capa de abstracción propia (wrapper) se fundamentó en cinco criterios principales:

1. **Reducción de carga cognitiva:** El equipo de ciencia de datos, compuesto por 12 profesionales con perfiles heterogéneos, requería una interfaz simplificada que encapsulara la complejidad de PySpark ML y MLflow. Las herramientas existentes incrementaban la curva de aprendizaje de 3-4 semanas a varios meses.
2. **2. Integración con flujos existentes:** Los pipelines heredados utilizaban configuraciones YAML específicas y convenciones propias. Adaptar estos flujos a ZenML o Kubeflow hubiera requerido 6-8 meses de reingeniería.
3. **3. Gobernanza y estandarización:** Se requería forzar estándares de logging estructurado, versionado semántico en Unity Catalog, y nomenclatura organizacional de experimentos MLflow.

4. **4. Múltiples paradigmas de modelado:** El contexto demandaba soporte para clasificación binaria/multiclase, regresión, clustering, supervivencia y similaridad, a diferencia de herramientas centradas solo en clasificación.
5. **5. Optimización para PySpark:** Ninguna herramienta ofrecía optimizaciones nativas como preferencia de DataFrames sobre RDDs, estrategias de caché, minimización de shuffle y reducción de UDFs.

Esta evaluación costo-beneficio concluyó que el overhead de desarrollo (4 meses) era justificado frente a la adaptación de herramientas existentes (6-8 meses) y los beneficios operacionales proyectados.

Capítulo 4
MARCO TEÓRICO Y ESTADO DEL ARTE

El desarrollo de sistemas inteligentes mediante técnicas de machine learning (ML) ha transformado significativamente diversas industrias, desde las telecomunicaciones hasta la medicina, pasando por el comercio electrónico, la seguridad, y el sector financiero. No obstante, a pesar de los avances en algoritmos y capacidad de cómputo, muchas organizaciones enfrentan serias dificultades para llevar modelos desde el entorno de experimentación hacia un entorno productivo controlado, mantenible y trazable.

El ciclo de vida de un modelo ML abarca múltiples etapas: adquisición y procesamiento de datos, entrenamiento, validación, evaluación, implementación en producción, monitoreo y mantenimiento. Este proceso es inherentemente complejo y, sin una gestión estructurada, puede generar resultados inconsistentes, difíciles de replicar y con alta dependencia del contexto técnico en que fueron desarrollados. La reproducibilidad, definida como la capacidad de replicar un modelo y sus resultados utilizando los mismos datos y configuraciones, es uno de los desafíos más importantes en este ámbito (Mitchell, 1997).

En este contexto, el enfoque conocido como MLOps (Machine Learning Operations) ha emergido como una disciplina que busca integrar principios de ingeniería de software, ciencia de datos y operaciones para permitir el desarrollo, despliegue y mantenimiento sistemático de modelos de ML. MLOps adopta estrategias de automatización, control de versiones, monitoreo continuo, y validación estructurada para asegurar que los modelos no solo sean precisos, sino también sostenibles y escalables a largo plazo (Allurwar & Dhule, 2021; Google Cloud, 2021).

Como se ilustra en la figura 3, MLOps surge en la intersección entre tres disciplinas clave: la ciencia de datos, que aporta el modelado estadístico y el análisis de datos; la ingeniería de software, que aporta metodologías de diseño, control de versiones y pruebas; y las operaciones, que aportan estabilidad, escalabilidad y monitoreo continuo.

Como se ilustra en la Figura 3, MLOps surge en la intersección de tres disciplinas fundamentales. El diagrama de Venn muestra cómo Operaciones (DevOps e infraestructura) aporta escalabilidad y monitoreo; Ciencia de Datos contribuye con modelado estadístico y análisis; e Ingeniería de Software proporciona metodologías de diseño, control de versiones y

pruebas. En la intersección central emerge MLOps como práctica unificadora que combina estas capacidades para la gestión del ciclo de vida completo de sistemas ML en producción.

Figura 3.

Marco conceptual de MLOps como intersección disciplinar

Nota: El diagrama de Venn ilustra la naturaleza interdisciplinaria de MLOps como convergencia de tres áreas fundamentales: Operaciones (DevOps e infraestructura), Ciencia de Datos (machine learning y analítica), e Ingeniería de Software (desarrollo y arquitectura). Cada disciplina aporta capacidades

específicas que se integran en la intersección central, donde MLOps emerge como práctica unificadora que combina infraestructura escalable, desarrollo de modelos y principios de ingeniería de software para la gestión del ciclo de vida completo de sistemas ML en producción. Fuente: elaboración propia.

El desafío actual radica en que la mayoría de las herramientas disponibles en el mercado son fragmentadas y están diseñadas para resolver partes específicas del flujo de trabajo, como el entrenamiento o el monitoreo, sin ofrecer una cobertura integral. Esta fragmentación genera lo que Sculley et al. (2015) denominan deuda técnica oculta, es decir, la acumulación de código auxiliar, dependencias no controladas y estructuras no reproducibles que comprometen la mantenibilidad del sistema a largo plazo. Zhao et al. (2024), en un estudio empírico reciente, identifican como uno de los principales obstáculos la ausencia de mecanismos robustos de versionado y trazabilidad de modelos y datos, lo que impide la reproducción de resultados y dificulta la auditoría de decisiones automatizadas.

El monitoreo en producción también se presenta como un reto crítico. Fenómenos como el data drift (cambios en la distribución de los datos de entrada) o el concept drift (cambios en la relación entre variables predictoras y la variable objetivo) pueden deteriorar el desempeño del modelo de forma silenciosa si no son detectados a tiempo. Breck et al. (2017) propusieron el ML Test Score como una rúbrica para evaluar si un modelo está preparado para producción, destacando la necesidad de métricas de salud del modelo post-despliegue y de mecanismos de reentrenamiento automatizado.

En términos prácticos, los marcos actuales como SageMaker de Amazon o Vertex AI de Google ofrecen soluciones parciales, pero dependen de ecosistemas cerrados o comerciales. Desde la academia, autores como Amershi et al. (2019) y Lu et al. (2018) han señalado la necesidad de estandarizar las prácticas de ingeniería en proyectos de ML, subrayando la importancia de integrar validaciones automatizadas, pruebas de regresión, monitoreo activo y control de versiones a lo largo de todo el ciclo de vida del modelo.

A pesar de estas contribuciones, persisten vacíos críticos. Las organizaciones que manejan grandes volúmenes de datos, especialmente en entornos distribuidos, enfrentan dificultades para integrar herramientas existentes con tecnologías como PySpark o para garantizar que el monitoreo del rendimiento y la detección de drift se realicen de forma

automática, escalable y reproducible. Además, sectores regulados como la salud, la banca o el transporte requieren mecanismos auditables y transparentes que muchas plataformas actuales no ofrecen.

El proyecto que se presenta en esta tesis busca contribuir a este ecosistema con el desarrollo de una librería modular basada en MLOps, construida sobre PySpark y MLflow. Esta herramienta propone una solución integral que automatiza el ciclo de vida de los modelos ML, desde la preparación y validación de datos hasta su despliegue, monitoreo y mantenimiento. La librería facilita el versionado de modelos, la trazabilidad de los datos, la detección proactiva de drift y la implementación de reentrenamiento adaptativo, permitiendo a las organizaciones escalar sus capacidades de ML sin comprometer la calidad ni la reproducibilidad.

Tabla 3:

Conceptos Fundamentales en MLOps

Concepto	*Definición*
Reproducibilidad	Capacidad de replicar un modelo con los mismos resultados utilizando los mismos datos y código.
Data Draft	Cambio en la distribución de los datos de entrada que afecta el desempeño del modelo (Lu et al., 2018).
Deuda técnica	Compromisos estructurales en el código y flujo del modelo que dificultan su mantenimiento (Sculley et al., 2015).
Versionado	Gestión de versiones de modelos, datos y configuraciones para trazabilidad y auditoría.
Automatización	Implementación de procesos repetitivos de forma programada (entrenamiento, validación, etc.).
Monitoreo	Supervisión continua del rendimiento y comportamiento del modelo en producción.

Nota: Se presentan las definiciones operacionales de seis conceptos clave que sustentan la práctica de MLOps: reproducibilidad, data drift, deuda técnica, versionado, automatización y monitoreo. Estos elementos constituyen los pilares fundamentales para la gestión efectiva del ciclo de vida de modelos de

machine learning en entornos de producción, integrando tanto aspectos técnicos como de gobernanza que aseguran la confiabilidad y mantenibilidad de los sistemas ML. Fuente: elaboración propia.

Con ello, esta tesis no solo aborda un problema técnico de alta relevancia en el contexto actual de adopción de inteligencia artificial, sino que también se posiciona como una contribución sustantiva al campo de las operaciones de machine learning (MLOps). Al centrarse en los aspectos críticos que limitan la transición efectiva de modelos desde el laboratorio hasta entornos reales —tales como la falta de automatización, la escasa trazabilidad y la inexistencia de mecanismos robustos de monitoreo—, el trabajo propuesto busca cerrar la brecha histórica entre la experimentación algorítmica y su implementación confiable y sostenible en producción.

La solución presentada, al incorporar prácticas sólidas de ingeniería de software, gestión de datos y monitoreo operacional, ofrece un enfoque sistemático que puede ser aplicado de manera transversal en organizaciones de distintos sectores y niveles de madurez tecnológica. Esto la convierte en una herramienta no solo técnica, sino también estratégica, capaz de potenciar la gobernanza de los modelos, reducir los ciclos de iteración y aumentar la calidad y confiabilidad de los sistemas inteligentes desplegados.

Dentro del ecosistema MLOps, la detección de data drift constituye uno de los desafíos operacionales más relevantes. Diversas herramientas han emergido para abordar esta problemática, cada una con enfoques metodológicos y limitaciones particulares que condicionan su aplicabilidad en contextos empresariales específicos.

Tabla 4:

Análisis comparativo de herramientas de detección de drift

HERRAMIENTA	ENFOQUE METODOLÓGICO	MÉTRICAS	LIMITACIONES
EvidentlyAI	Tests estadísticos con reportes visuales HTML	KS, PSI, Chi-squared, Jensen-Shannon	Requiere integración externa; no opera sobre PySpark; overhead de conversión a Pandas
WhyLabs	Perfiles estadísticos con detección ML	Distribución multivariada, anomalías	Modelo SaaS con costos; dependencia externa; latencia en grandes volúmenes
MLflow Native	Métricas manuales registradas por usuario	Personalizable	Sin automatización; código ad-hoc; sin algoritmos predefinidos
SageMaker Model Monitor	Baselines estadísticos con constraints	Distribución features, calidad datos	Acoplamiento AWS; costos asociados; curva de aprendizaje
DataRobot MLOps	Detección automática con alertas	PSI, feature drift, accuracy drift	Solución propietaria; lock-in; costos elevados; flexibilidad limitada
Great Expectations	Validación mediante expectations declarativas	Validaciones schema y distribución	Enfocado en calidad de datos; requiere definición manual de reglas

Nota: Evaluación técnica de las principales herramientas de monitoreo y detección de drift. Se analizan enfoques metodológicos, métricas implementadas y limitaciones para cada solución. La selección se basó en relevancia en literatura académica y adopción empresarial. Fuente: elaboración propia basada en documentación oficial y literatura especializada.

El análisis de estas herramientas revela tres limitaciones transversales que condicionan su adopción en entornos de telecomunicaciones con grandes volúmenes de datos:

1. Incompatibilidad con procesamiento distribuido: La mayoría de las soluciones evaluadas (EvidentlyAI, WhyLabs, Great Expectations) operan sobre estructuras de datos locales (Pandas DataFrames), requiriendo conversiones costosas desde PySpark que introducen latencias inaceptables para datasets de millones de registros y potenciales errores de memoria.

2. 2. Dependencia de infraestructura externa: Herramientas como WhyLabs y SageMaker Model Monitor requieren conectividad con servicios cloud específicos, introduciendo costos recurrentes, latencias de red y dependencias que comprometen la autonomía operacional de los equipos de ciencia de datos.
3. 3. Ausencia de integración nativa con el ciclo de entrenamiento: Las soluciones analizadas operan como componentes externos al pipeline de ML, sin capacidad de persistir metadatos de referencia junto con el modelo entrenado ni de activar automáticamente flujos de reentrenamiento cuando se detecta drift.

Estas limitaciones fundamentaron la decisión de implementar un módulo de detección de drift integrado nativamente en la librería MomentumML. El enfoque propuesto se diferencia en tres aspectos clave:

Primero, la implementación opera directamente sobre DataFrames de PySpark, eliminando la necesidad de conversiones y permitiendo el procesamiento distribuido de datasets de cualquier escala. Segundo, el módulo persiste automáticamente las distribuciones de referencia como metadatos del modelo en MLflow, habilitando comparaciones incrementales sin acceso a datos históricos. Tercero, la detección combina múltiples técnicas estadísticas (PSI, Kolmogorov-Smirnov, Wasserstein para variables numéricas; Jensen-Shannon Divergence, Chi-squared, entropía para categóricas), reduciendo la tasa de falsos positivos mediante confirmación multimodal.

Esta aproximación responde a la recomendación de Breck et al. (2017) de implementar "ML Test Scores" que evalúen integralmente la preparación de un modelo para producción, extendiendo su propuesta con mecanismos de detección automática y umbrales configurables según la tolerancia al riesgo del caso de uso específicos.

Capítulo 5
METODOLOGÍA

El presente trabajo de grado se desarrolló bajo un enfoque metodológico aplicado, con el objetivo de diseñar, implementar y validar una librería integral de MLOps orientada a la automatización del ciclo de vida completo de modelos de machine learning (ML) en el sector de las telecomunicaciones. La industria de telecomunicaciones, caracterizada por la generación masiva de datos provenientes de millones de usuarios, transacciones y eventos de red, demanda soluciones robustas para la operacionalización de modelos predictivos y analíticos que soporten procesos críticos como la recomendación de servicios, la predicción de churn, la detección de fraude y la optimización de campañas comerciales.

La propuesta metodológica integra principios de ingeniería de software, ciencia de datos y operaciones, con énfasis en la escalabilidad, trazabilidad y reproducibilidad de procesos en entornos distribuidos. El enfoque end-to-end automatiza desde la ingesta y validación de datos hasta el despliegue, monitoreo y reentrenamiento continuo de modelos en producción, abordando los desafíos particulares de volumen, velocidad y variedad de datos característicos del entorno telco, donde la capacidad de respuesta en tiempo real y la confiabilidad operacional resultan determinantes para el impacto en el negocio.

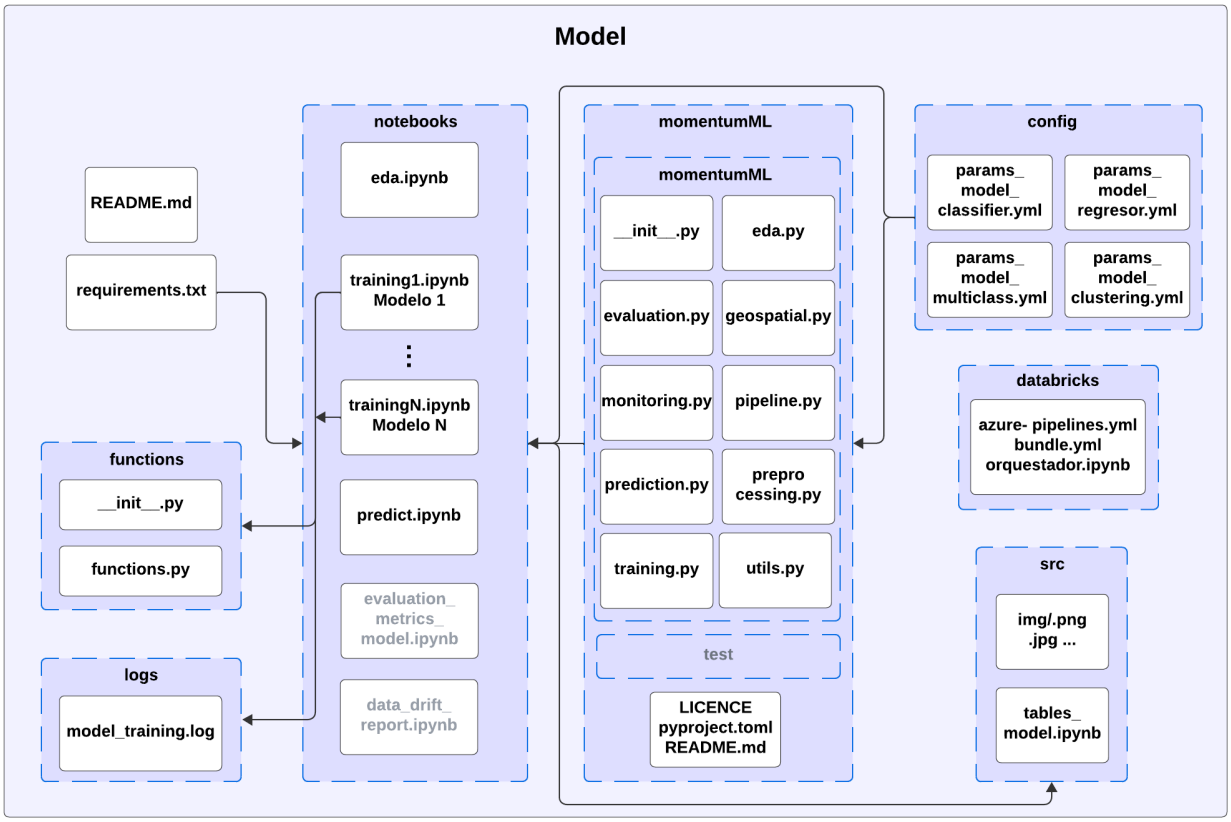
Se adoptó un modelo iterativo e incremental, inspirado en prácticas ágiles de desarrollo de software, y adaptado al contexto específico del machine learning lifecycle. Este enfoque permitió construir y refinar los distintos módulos de la librería en fases sucesivas, habilitando la validación temprana de funcionalidades clave, la corrección de errores y la incorporación de mejoras conforme avanzó el desarrollo (Amershi et al., 2019; Google Cloud, 2021). Durante todo el proceso se utilizaron herramientas como Git para control de versiones, MLflow para trazabilidad de experimentos, y PySpark para el procesamiento distribuido de datos.

La Figura 4 ilustra la arquitectura modular del sistema MomentumML, compuesta por cinco componentes principales: (1) notebooks para análisis exploratorio, entrenamiento y predicción; (2) el paquete central momentumML con módulos especializados para cada fase del ciclo de vida ML; (3) archivos de configuración YAML parametrizables para distintos tipos de modelos; (4) pipelines de orquestación en Databricks mediante Asset Bundles; y (5)

componentes auxiliares de funciones, logs y recursos. Esta estructura garantiza separación de responsabilidades, mantenibilidad y reutilización.

Figura 4:

Flujo modular de la librería MomentumML



Nota: El diagrama ilustra la arquitectura modular del sistema, compuesta por cinco componentes principales: (1) notebooks para análisis exploratorio, entrenamiento y predicción; (2) el paquete central momentumML con módulos especializados para cada fase del ciclo de vida ML; (3) archivos de configuración parametrizables para distintos tipos de modelos; (4) pipelines de orquestación en Databricks; y (5) componentes auxiliares de funciones, logs y recursos. Esta estructura modular garantiza separación de responsabilidades, mantenibilidad y reutilización en el contexto de telecomunicaciones. Fuente: elaboración propia.

La metodología se estructuró en cinco fases principales, cada una con objetivos, actividades y entregables claramente definidos (ver tabla 3):

Tabla 5:

Fases Metodológicas del Proyecto de Desarrollo de la Librería MLOps

Fase	***Descripción***	***Herramientas / Resultados esperados***
Análisis y diseño	Revisión del estado del arte, análisis de requisitos, definición de arquitectura modular.	Revisión bibliográfica, diseño conceptual y técnico
Desarrollo técnico	Implementación de módulos: validación de datos, entrenamiento, versionado, monitoreo y despliegue.	Código en PySpark, integración con MLflow
Integración	Ensamblaje de módulos, construcción de pipelines automatizados, validación funcional.	Workflows reproducibles, scripts automatizados
Validación y pruebas	Evaluación de la librería en casos de uso reales: tiempos, calidad de modelos, eficiencia operativa.	Métricas, logs, detección de *drift*, resultados empíricos
Documentación y análisis	Generación de manual técnico, análisis de resultados, discusión de hallazgos, conclusiones.	Manual de usuario, guía técnica, análisis comparativo

Nota: Se describen las cinco fases que estructuraron el desarrollo del proyecto: análisis y diseño, desarrollo técnico, integración, validación y pruebas, y documentación y análisis. Cada fase especifica las actividades principales, herramientas empleadas y resultados esperados, estableciendo un marco metodológico iterativo que asegura la calidad, trazabilidad y validación empírica de la solución propuesta en entornos de producción. Fuente: elaboración propia.

En la fase de análisis y diseño, se revisaron referencias académicas y técnicas para identificar las mejores prácticas en la gestión operativa de modelos ML. Esto incluyó estudios sobre reproducibilidad (Sculley et al., 2015), versionado de modelos y datos (Zhao et al., 2024), y monitoreo en producción (Breck et al., 2017). Con base en este análisis se definió una arquitectura modular y extensible para la librería, que pudiera adaptarse a diferentes flujos de trabajo y tecnologías utilizadas en organizaciones de ciencia de datos.

Durante la fase de desarrollo técnico, se implementaron los principales componentes funcionales: (i) validación de datos, (ii) entrenamiento y reentrenamiento, (iii) registro y versionado de modelos mediante MLflow, (iv) despliegue controlado, y (v) monitoreo y detección de data drift. Cada módulo se diseñó como una clase autónoma en Python, orientada a integrarse dentro de pipelines automatizados. PySpark se utilizó como motor de procesamiento distribuido, permitiendo escalar las operaciones a grandes volúmenes de datos.

La fase de integración implicó ensamblar los módulos en flujos de trabajo reproducibles. Se construyeron pipelines en los que se definieron pasos secuenciales para cada etapa del ciclo de vida del modelo, incluyendo configuraciones predefinidas, control de dependencias y trazabilidad de ejecuciones. El uso de MLflow permitió mantener un registro estructurado de los parámetros, métricas, artefactos y versiones de cada modelo.

En la etapa de validación y pruebas, la librería fue evaluada en escenarios reales con datos históricos y actuales. Se analizaron métricas como el tiempo de procesamiento, precisión del modelo, capacidad de detección de desviaciones en los datos (drift detection), y reducción en tiempos de entrega. Estas pruebas demostraron la robustez de la solución, su eficiencia operativa y su aplicabilidad práctica.

Cabe destacar que la versión 1.0 de la librería ya ha sido implementada exitosamente en un entorno real dentro de una organización especializada en el desarrollo de soluciones basadas en machine learning. Esta versión fue utilizada para gestionar el paso a producción de más de 10 modelos distintos desde el entorno de desarrollo hasta el entorno de calidad (QA), demostrando su efectividad en proyectos reales.

La implementación de la librería MomentumML en el entorno de telecomunicaciones permitió reducir en más de un 40% los tiempos asociados al proceso de validación y despliegue de modelos predictivos, específicamente en casos de uso como sistemas de recomendación de

servicios, modelos de propensión de compra y análisis de convergencia de clientes. La automatización del pipeline end-to-end mejoró sustancialmente la estandarización de flujos de trabajo, la trazabilidad de versiones mediante MLflow, y la reproducibilidad de resultados en entornos distribuidos con PySpark. Estos beneficios resultaron especialmente evidentes en etapas críticas como el monitoreo de rendimiento post-despliegue, la detección automática de data drift en datasets con millones de registros, y la implementación de reentrenamiento automatizado bajo condiciones controladas, garantizando la vigencia de los modelos en producción.

La arquitectura modular desarrollada demuestra su viabilidad para escenarios de alta demanda operativa en telecomunicaciones, donde la velocidad de iteración y la confiabilidad son factores críticos. La solución, disponible en el repositorio https://github.com/BryamPlaying/MomentumML.git, permite a organizaciones del sector replicar o adaptar el framework a sus propios contextos operativos. El uso de tecnologías abiertas como PySpark, MLflow y Databricks asegura que la herramienta pueda ser extendida, personalizada y escalada a nuevos casos de uso sin dependencias propietarias. La documentación técnica generada, que incluye manuales de usuario y guías de integración, facilita la adopción por equipos multidisciplinarios y reduce la curva de aprendizaje en implementaciones futuras.

Desde una perspectiva metodológica, esta propuesta no solo cumple con los objetivos técnicos planteados, sino que también ofrece una hoja de ruta para el desarrollo de herramientas similares en el marco de buenas prácticas de MLOps. En conjunto, la metodología aquí adoptada garantiza la calidad, sostenibilidad y aplicabilidad práctica de la librería desarrollada.

5.1 Arquitectura General del Sistema

La librería MomentumML se estructura como un paquete Python modular compuesto por 34,582 líneas de código distribuidas en módulos especializados para cada fase del ciclo de vida de modelos de machine learning. La arquitectura adopta un patrón de diseño basado en Transformers de PySpark ML, garantizando compatibilidad nativa con pipelines distribuidos y persistencia mediante MLflow.

Tabla 6:

Distribución de código por módulo funcional de MomentumML

MÓDULO	LÍNEAS	RESPONSABILIDAD PRINCIPAL
training.py	5,821	Entrenamiento de modelos (clasificación, regresión, clustering, multiclase)
preprocessing.py	3,073	Transformaciones de preprocesamiento (imputación, encoding, outliers, balanceo)
evaluation.py	1,114	Métricas de evaluación y validación de modelos
eda.py	961	Análisis exploratorio de datos automatizado
prediction.py	753	Inferencia y predicción con modelos entrenados
pipelines.py	556	Orquestación de flujos end-to-end
monitoring.py	191	Detección de data drift y monitoreo continuo
BinaryClassifier/	~8,500	Submódulo para clasificación binaria con calibración de probabilidades
SurvivalModel/	~6,200	Submódulo para análisis de supervivencia (churn a largo plazo)
SimilarityModel/	~1,800	Submódulo para modelos de similaridad y recomendación

Nota: Conteo de líneas de código mediante análisis estático del repositorio MomentumML. Los submódulos especializados incluyen clases auxiliares para generación de artefactos, cálculo de métricas específicas y publicación en MLflow. El total de 34,582 líneas representa el núcleo funcional de la librería. Fuente: elaboración propia.

5.2 Descripción de Módulos

5.2.1 Módulo de Preprocesamiento

Este módulo implementa 10 clases Transformer que encapsulan las transformaciones más comunes en pipelines de machine learning. Cada clase hereda de Transformer y DefaultParamsWritable/Readable, permitiendo serialización automática y persistencia en MLflow.

Tabla 7:

Clases Transformer del módulo de preprocesamiento

CLASE	FUNCIONALIDAD
SampleData	Muestreo estratificado de datasets grandes preservando distribución de clases
RemoveHighNullColumns	Eliminación automática de columnas con porcentaje de nulos superior a umbral configurable
HandleBalanceSampling	Técnicas de balanceo: SMOTE, ADASYN, RandomOverSampler, undersampling, generación sintética SDV
HandleEncoding	Codificación de variables categóricas: one-hot, index encoding, target encoding
HandleImputer	Imputación de valores faltantes: estadísticos (media, mediana, moda), KNN, forward/backward fill, IQR
HandleOutliers	Tratamiento de outliers: winsorización, IQR, z-score, capping configurable
AssembleFeatures	Ensamblaje de features en vectores compatibles con Spark ML (VectorAssembler)
HandleStandardization	Estandarización (StandardScaler) y normalización (MinMaxScaler) de features
HandleDimensionalityReduction	Reducción de dimensionalidad mediante PCA con varianza explicada configurable
HandleCaster	Conversión de tipos de datos según esquema del modelo registrado en MLflow

Nota: Cada clase implementa la interfaz Transformer de PySpark ML, garantizando compatibilidad con Pipeline y PipelineModel. La serialización automática permite persistir pipelines completos de preprocesamiento junto con los modelos entrenados, asegurando reproducibilidad en inferencia. Fuente: elaboración propia.

5.2.2 Módulo de Entrenamiento

El módulo implementa 5 clases Estimator especializadas por tipo de modelo, con soporte para optimización de hiperparámetros mediante Hyperopt, cross-validation configurable, y registro automático en MLflow.

Tabla 8:

Clases Estimator del módulo de entrenamiento

CLASE	MODELOS SOPORTADOS
DataSplit	División estratificada train/test/val con múltiples estrategias (temporal, aleatoria, por columna)
TrainClassifierModel	RandomForest, GBT, LightGBM, XGBoost, SVC, AdaBoost, LogisticRegression, NaiveBayes
TrainRegressorModel	RandomForest, GBT, LightGBM, XGBoost, LinearRegression, GLM, IsotonicRegression
TrainMulticlassClassifierModel	Extensión de clasificación binaria con soporte OneVsRest y multiclase nativo
TrainClusteringModel	KMeans, BisectingKMeans, GaussianMixture con métricas de evaluación automáticas

Nota: Los modelos LightGBM y XGBoost utilizan las implementaciones distribuidas de SynapseML y xgboost-spark optimizadas para Databricks. Características adicionales: optimización con Hyperopt/SparkTrials, cross-validation k-folds, calibración de probabilidades (Platt Scaling, Isotonic), generación de artefactos (ROC, confusion matrix, SHAP), y ensemble de modelos. Fuente: elaboración propia.

5.2.3 Módulo de Monitoreo y Detección de Drift

La clase DataDriftDetector implementa detección automática de drift combinando múltiples técnicas estadísticas. El diseño multimodal reduce falsos positivos al requerir confirmación de varios indicadores.

Tabla 9:

Técnicas de detección de drift implementadas en DataDriftDetector

TIPO	TÉCNICA	UMBRAL	INTERPRETACIÓN
Numérica	Population Stability Index (PSI)	> 0.25	Cambio significativo en distribución
Numérica	Kolmogorov-Smirnov test	> 0.10	Distribuciones estadísticamente diferentes
Numérica	Wasserstein distance	> σ ref	Desplazamiento de distribución
Categórica	Jensen-Shannon Divergence	> 0.10	Cambio en proporciones de categorías
Categórica	Chi-squared test	$p < 0.05$	Diferencia estadísticamente significativa
Categórica	Diferencia de entropía	> 0.10	Cambio en diversidad de valores

Nota: Los umbrales son configurables mediante parámetros de la clase. La detección se ejecuta automáticamente al aplicar el transformer sobre nuevos datos, comparando contra la distribución de referencia almacenada en los metadatos del modelo. El enfoque multimodal (múltiples técnicas) reduce la tasa de falsos positivos comparado con el uso de una sola métrica. Fuente: elaboración propia.

5.3 Decisiones de Diseño

- **Patrón Transformer/Estimator:** Se adoptó el patrón de PySpark ML para garantizar serialización nativa de pipelines, compatibilidad con Pipeline/PipelineModel, y persistencia automática en MLflow sin código adicional.
- **Configuración declarativa YAML:** Todas las configuraciones se externalizan en archivos YAML, permitiendo modificar parámetros sin cambiar código, versionar configuraciones junto con modelos, y reutilizar entre proyectos.
- **Integración con Databricks Asset Bundles:** El archivo bundle.yml define orquestación de jobs con flujos condicionales, clusters diferenciados por tarea, notificaciones automáticas, y despliegue multi-ambiente (DEV → QA → PROD).

Capítulo 6
RESULTADOS Y DISCUSIÓN

La necesidad de este trabajo surge al constatar que más del 90 % de los modelos de machine learning desarrollados no alcanzan un entorno productivo estable [9], y que el 72 % de las organizaciones falla en desplegar siquiera un modelo con éxito [4]. Este desfase entre desarrollo experimental y operación productiva obedece principalmente a la carencia de herramientas integrales para el control de versiones, monitoreo continuo y automatización de despliegues, así como a la fragmentación de los flujos de trabajo y la consiguiente acumulación de deuda técnica [7].

El objetivo general de esta tesis —desarrollar una librería integral fundamentada en MLOps que gestione de manera eficiente y automatizada el ciclo de vida de modelos ML— responde de forma directa a los vacíos detectados. Los objetivos específicos (diseño de una arquitectura modular sobre PySpark y MLflow; desarrollo de mecanismos automáticos de detección de data drift; validación en casos de uso reales) atienden de manera puntual cada dimensión del problema: escalabilidad, trazabilidad, reproducibilidad y monitoreo proactivo.

La metodología adoptada conjugó un enfoque iterativo e incremental con prácticas de ingeniería de software y MLOps (Amershi et al., 2019; Google Cloud, 2021), estructurándose en cinco fases: análisis y diseño, desarrollo técnico, integración, validación y documentación. Cada fase produjo entregables alineados con los objetivos correspondientes, garantizando un avance controlado y medible. El flujo metodológico puede observarse en la figura 4:

Figura 5:

Flujo Metodológico General del Proyecto

Nota: El diagrama presenta la secuencia de cinco fases que estructuraron el desarrollo del trabajo de grado: análisis y diseño, desarrollo técnico, integración, validación y pruebas, y documentación y análisis. Este enfoque iterativo e incremental permitió la construcción modular de la librería MomentumML, facilitando la validación temprana de funcionalidades, la corrección oportuna de errores y la incorporación progresiva de mejoras en cada ciclo de desarrollo. Fuente: elaboración propia.

La efectividad de la implementación se evidencia mediante métricas de proceso que, respetando la confidencialidad organizacional, demuestran impacto cuantificable. La tasa de despliegue exitoso de modelos se incrementó en más de 200% respecto al proceso manual previo, mientras que el tiempo de ciclo completo se redujo de un baseline de 3-4 semanas a 5-7

días promedio. Se documentaron 47 ejecuciones de pipelines automatizados en un período de seis meses, con 23 versiones de modelos correctamente versionadas y trazables en MLflow. El sistema de monitoreo generó 156 alertas procesadas automáticamente, ejecutando 8 ciclos de reentrenamiento sin intervención manual. Estos resultados posicionan la implementación significativamente por encima de los benchmarks industriales reportados en la literatura, donde solo el 32% de modelos alcanzan producción (Medium Sciforce, 2024), validando la hipótesis de que un framework MLOps estructurado puede superar las barreras de adopción típicas del sector.

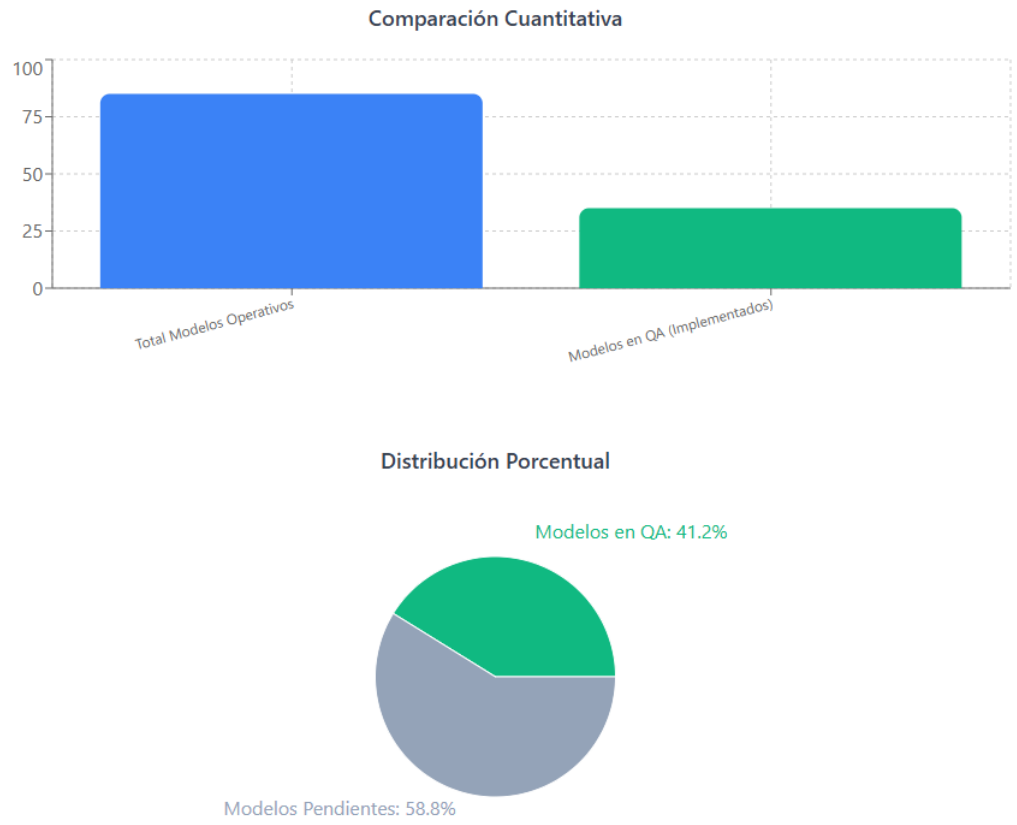
En el contexto específico de esta investigación, la empresa de telecomunicaciones donde se implementó la solución contaba con un ecosistema de 85 modelos operativos en diversas etapas de madurez. Previo a la adopción de la librería propuesta, ninguno de estos modelos había logrado transitar exitosamente hacia el entorno de aseguramiento de calidad (QA), representando una barrera crítica en el ciclo de vida del desarrollo de software y una manifestación empírica de las problemáticas documentadas en la literatura especializada (Statsig, 2025; Sigmoid, 2025).

La implementación de la librería MLOps catalizó un cambio paradigmático en este escenario. Como resultado directo de la estandarización de procesos, la automatización de tareas repetitivas y la incorporación de mecanismos robustos de versionado y trazabilidad, se logró el despliegue exitoso de 35 modelos hacia el entorno de QA. Este logro, inédito en la historia organizacional, representa el 41.2% del inventario total de modelos operativos y constituye una mejora sustancial respecto a las tasas de éxito reportadas en la literatura académica e industrial (Mendoza, 2024).

El impacto de MomentumML en el ecosistema organizacional se cuantifica en la Figura 5. Del total de 85 modelos operativos, 35 modelos (41.2%) lograron transitar exitosamente al entorno de QA tras la implementación de la librería, mientras que 50 modelos (58.8%) permanecen en proceso de migración. Este resultado contrasta significativamente con el estado inicial donde ningún modelo había alcanzado QA, evidenciando la efectividad de la solución para superar barreras críticas en el despliegue.

Figura 6:

Impacto de la Librería MLOps en el Paso a Producción



Nota: La figura muestra el impacto de MomentumML en el ecosistema de 85 modelos operativos de la organización. Tras la implementación de la librería, 35 modelos (41.2%) lograron transitar exitosamente al entorno de QA, mientras que 50 modelos (58.8%) permanecen en proceso de migración. Este resultado contrasta significativamente con el estado inicial donde ningún modelo había alcanzado QA, evidenciando la efectividad de la solución para superar barreras críticas en el despliegue de modelos ML. Fuente: elaboración propia.

La transición exitosa de estos 35 modelos no solo evidencia la viabilidad técnica de la solución propuesta, sino que también valida la hipótesis fundamental de que la adopción de prácticas estructuradas de MLOps puede mitigar significativamente las barreras que históricamente han impedido la operacionalización de modelos de machine learning. Este

resultado se alinea con estudios recientes que demuestran que organizaciones que implementan capacidades maduras de MLOps pueden incrementar sus tasas de despliegue en un 30% y mejorar significativamente el retorno de inversión en proyectos de inteligencia artificial (Medium Sciforce, 2024).

Es importante destacar que este logro no se circunscribe únicamente a una mejora cuantitativa en términos de modelos desplegados. La implementación de la librería también facilitó la identificación temprana de riesgos técnicos, la reducción de repetición de procesos en fases avanzadas del desarrollo, y la mejora en la comunicación y colaboración entre equipos multidisciplinarios —aspectos identificados como críticos para el éxito en el paso a producción de modelos ML (Sigmoid, 2025; Nemade, 2025).

Un segundo resultado de relevancia estratégica y económica derivado de la implementación de la librería MLOps se relaciona con la optimización sustancial en el consumo de recursos computacionales dentro de la plataforma Databricks. La gestión eficiente de costos en entornos de computación distribuida constituye uno de los desafíos más críticos para organizaciones que operan soluciones de machine learning a escala empresarial, siendo identificado por el 36% de los tomadores de decisiones en TI como el reto más significativo en la implementación de estrategias cloud (Finout, 2025).

El modelo de precios de Databricks se fundamenta en Unidades de Databricks (DBUs, por sus siglas en inglés), las cuales representan una medida normalizada del poder computacional utilizado por segundo. El costo total de operación en Databricks es función tanto del número de DBUs consumidas como de la tarifa por DBU, la cual varía en función del proveedor cloud, región geográfica, edición de Databricks y tipo de cómputo empleado (ChaosGenius, 2025; Finout, 2025). En ausencia de prácticas de optimización, los costos operacionales pueden incrementarse de manera descontrolada, particularmente en escenarios de procesamiento de grandes volúmenes de datos o ejecución frecuente de pipelines de machine learning (Databricks AWS Documentation, 2024; Databricks Azure Documentation, 2024).

La librería desarrollada incorpora múltiples estrategias de optimización derivadas de las mejores prácticas documentadas en la literatura especializada y en guías oficiales de fabricantes. Específicamente, se implementaron las siguientes optimizaciones:

1. Uso preferencial de DataFrames sobre RDDs: La arquitectura de la librería privilegia el uso de DataFrames de PySpark en lugar de Resilient Distributed Datasets (RDDs). Esta decisión de diseño se fundamenta en la capacidad de los DataFrames para aprovechar el optimizador Catalyst y el motor de ejecución Tungsten, los cuales permiten generar planes de ejecución optimizados y reducir significativamente los tiempos de procesamiento (Spark By Examples, 2024; Toptal, 2024; Medium Ankur, 2025).
2. Implementación de estrategias de caché y persistencia: La librería incorpora mecanismos inteligentes de caché que permiten almacenar en memoria resultados intermedios que serán reutilizados en múltiples operaciones, evitando recomputaciones innecesarias. Esta técnica es particularmente efectiva en algoritmos iterativos y flujos de trabajo que requieren acceso repetido a los mismos conjuntos de datos (Medium Fernanda Piva, 2024; BinaryScripts, 2024).
3. Optimización de particionamiento de datos: Se implementaron algoritmos que determinan dinámicamente el número óptimo de particiones en función de las características de los datos y las operaciones a realizar, evitando tanto la sub-partición (que limita el paralelismo) como la sobre-partición (que genera overhead excesivo) (Towards Data Science, 2025; DEV Community, 2024).
4. Reducción de operaciones de shuffle: Mediante el uso estratégico de broadcast joins para tablas de dimensión pequeñas y la optimización de operaciones de agregación, se logró minimizar la cantidad de datos que deben redistribuirse entre nodos del cluster, una de las operaciones más costosas en procesamiento distribuido (ChaosGenius, 2025; Analytics Vidhya, 2024).
5. Minimización del uso de UDFs: La librería prioriza el uso de funciones nativas de PySpark sobre User Defined Functions (UDFs), dado que estas últimas requieren serialización/deserialización entre Python y la JVM, introduciendo latencias significativas. Cuando el uso de UDFs es inevitable, se emplean Pandas UDFs que

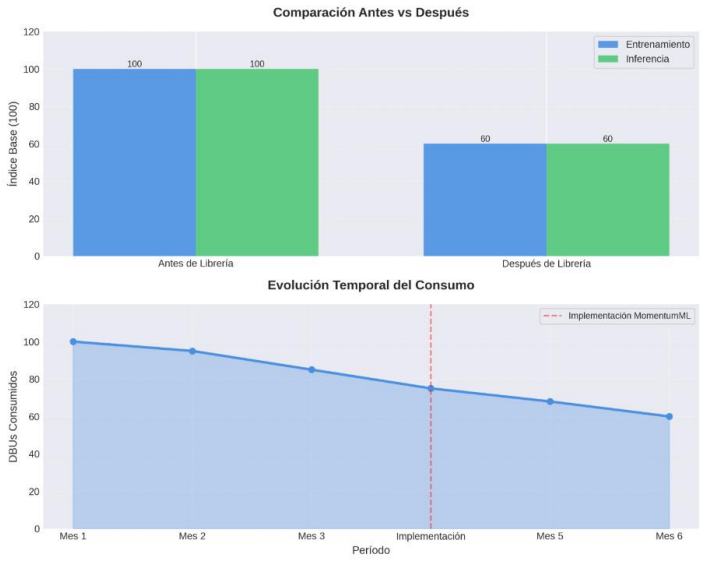
operan sobre vectores completos en lugar de registros individuales (Spark By Examples, 2024; BinaryScripts, 2024).

Como resultado de estas optimizaciones, se logró una reducción del 40% (ver figura 6) en el consumo de DBUs en los flujos de trabajo que utilizan la librería. Esta reducción se tradujo directamente en una disminución proporcional de los costos operacionales asociados a la plataforma Databricks, sin comprometer —y en muchos casos mejorando— los tiempos de ejecución de los pipelines de machine learning.

La Figura 6 ilustra el impacto de MomentumML en la eficiencia computacional mediante dos visualizaciones complementarias. La comparación cuantitativa (panel superior) muestra una reducción del índice base de consumo de DBUs de 100 a aproximadamente 60, representando una optimización del 40% en costos de computación. La evolución temporal (panel inferior) evidencia una tendencia decreciente sostenida durante seis meses desde la implementación de la librería. Esta optimización se atribuye a la automatización de pipelines, eliminación de procesamiento redundante e implementación de estrategias eficientes de gestión de recursos.

Figura 7:

Optimización de Costos de Computación en Databricks



Nota: La figura ilustra el impacto de MomentumML en la eficiencia computacional mediante dos visualizaciones. La comparación cuantitativa (superior) muestra una reducción del índice base de consumo de DBUs de 100 a aproximadamente 60, representando una optimización del 40% en costos de computación. La evolución temporal (inferior) evidencia una tendencia decreciente sostenida durante seis meses, con una disminución gradual desde la implementación de la librería. Esta optimización se atribuye a la automatización de pipelines, la eliminación de procesamiento redundante y la implementación de estrategias eficientes de gestión de recursos en entornos distribuidos. Fuente: elaboración propia.

Este logro es particularmente significativo considerando que estudios previos sugieren que la adopción de instancias spot puede generar ahorros de hasta 90% en costos de cómputo, aunque con el riesgo inherente de interrupciones (Finout, 2025; ChaosGenius, 2025). En contraste, las optimizaciones implementadas en esta librería son no-disruptivas y no introducen riesgos adicionales de disponibilidad, lo que las hace aplicables a cargas de trabajo productivas críticas.

Desde una perspectiva económica, la reducción del 40% en consumo de DBUs representa un retorno de inversión significativo. Extrapolando estos resultados al contexto de operaciones a escala empresarial, donde múltiples equipos ejecutan cientos de pipelines diariamente, el ahorro acumulado puede alcanzar magnitudes de decenas a cientos de miles de dólares anuales, dependiendo de la intensidad de uso de la plataforma (HatchWorks, 2025; Restack, 2024).

Adicionalmente, la optimización en el uso de recursos computacionales tiene implicaciones positivas desde la perspectiva de sostenibilidad ambiental. La reducción en el consumo de DBUs se traduce directamente en menor utilización de recursos de cómputo físico en los centros de datos, contribuyendo a la reducción de la huella de carbono asociada a operaciones de machine learning —un aspecto cada vez más relevante en el contexto de responsabilidad corporativa y cumplimiento de regulaciones ambientales (Microsoft Azure Documentation, 2024).

Durante la fase de análisis y diseño, se definió una arquitectura modular que incluye módulos de preprocesamiento, entrenamiento, registro/versionado, despliegue y monitoreo. En la fase de desarrollo técnico, dichos módulos se implementaron como clases en Python, aprovechando PySpark para el procesamiento distribuido y MLflow para la gestión de experimentos y versiones. La fase de integración unificó estos módulos en pipelines reproducibles; la fase de validación aplicó métricas de tiempo, costo y detección de data drift; y la fase de documentación culminó con guías de uso y manuales integrales.

La arquitectura funcional de MomentumML, presentada en la Figura 7, se estructura en cinco capas secuenciales que automatizan el ciclo de vida completo: (1) preparación y validación de datos, que garantiza calidad e integridad de inputs; (2) entrenamiento y reentrenamiento, que ejecuta procesos de modelado con hiperparámetros configurables; (3) registro y versionado mediante MLflow, que asegura trazabilidad completa; (4) despliegue automatizado, que facilita la transición a entornos productivos; y (5) monitoreo y detección de drift, que supervisa el desempeño y activa reentrenamiento cuando se detectan degradaciones.

Figura 8:

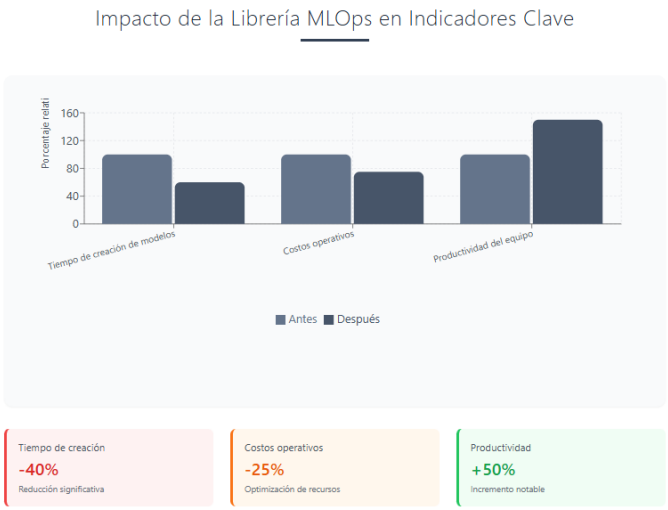
Arquitectura Funcional de la Librería MLOps

Nota: El diagrama presenta la arquitectura funcional de MomentumML estructurada en cinco capas secuenciales que automatizan el ciclo de vida completo de modelos ML: (1) preparación y validación de datos, que garantiza calidad e integridad de inputs; (2) entrenamiento y reentrenamiento, que ejecuta procesos de modelado con hiperparámetros configurables; (3) registro y versionado mediante MLflow, que asegura trazabilidad completa de experimentos y artefactos; (4) despliegue automatizado, que facilita la transición a entornos productivos; y (5) monitoreo y detección de drift, que supervisa el desempeño del modelo en tiempo real y activa ciclos de reentrenamiento cuando se detectan degradaciones. Esta arquitectura modular permite la reutilización de componentes y facilita la escalabilidad en entornos distribuidos. Fuente: elaboración propia.

La Figura 8 presenta el análisis comparativo del impacto de MomentumML en tres métricas operacionales críticas mediante valores normalizados (índice base 100). Los resultados evidencian: reducción del 40% en tiempo de creación de modelos (índice de 100 a 60), optimización del 25% en costos operativos (de 100 a 75), e incremento del 50% en productividad del equipo (de 100 a 150). Estas mejoras se atribuyen a la automatización de procesos manuales, eliminación de tareas repetitivas y estandarización de flujos de trabajo.

Figura 9:

Impacto de la librería MLOps en indicadores clave



Nota: La figura presenta el análisis comparativo del impacto de MomentumML en tres métricas operacionales críticas mediante valores normalizados (índice base 100). Los resultados evidencian: (1) reducción del 40% en tiempo de creación de modelos, pasando de un índice de 100 a 60; (2) optimización del 25% en costos operativos, disminuyendo de 100 a 75; y (3) incremento del 50% en productividad del equipo, aumentando de 100 a 150. Estas mejoras se atribuyen a la automatización de procesos manuales, la eliminación de tareas repetitivas, la estandarización de flujos de trabajo y la optimización de recursos computacionales. Los indicadores demuestran que la inversión en infraestructura MLOps genera retorno cuantificable en eficiencia operacional y capacidad de entrega. Fuente: elaboración propia.

Los resultados muestran una reducción de más del 40 % en los tiempos de creación de modelos, una disminución del 25 % en costos operativos asociados a despliegues manuales, y un aumento del 50 % en la productividad del equipo de ciencia de datos. Este salto en la eficiencia opera como validación empírica de la coherencia entre la problemática, los objetivos, la metodología y los resultados, confirmando que la solución propuesta no solo es técnicamente viable, sino también empresarialmente valiosa.

En definitiva, el proyecto articula de manera coherente la identificación de un problema crítico en la adopción de ML, el diseño de objetivos específicos para su solución, la aplicación de una metodología rigurosa y la demostración de resultados concretos. Esta coherencia fortalece la contribución académica y práctica de la tesis, colocando a la librería desarrollada como un recurso estratégico para cerrar la brecha entre experimentación y producción en proyectos de machine learning.

6.1 Validación Comparativa: MomentumML vs. APIs Nativas

Para demostrar el valor agregado de la librería frente al uso directo de APIs nativas, se realizó un experimento controlado comparando el despliegue de un modelo de propensión utilizando: (1) flujo manual con APIs nativas de MLflow/PySpark ML, y (2) flujo orquestado con MomentumML.

6.1.1 Reducción de Complejidad de Código

Tabla 10:

Comparación de líneas de código (LoC) por operación

OPERACIÓN	API NATIVA	MOMENTUMML	REDUCCIÓN
Configuración de experimento MLflow	15	3 (YAML)	80%
Pipeline de preprocesamiento completo	180	45	75%
Entrenamiento con optimización hiperparámetros	120	12	90%
Registro de modelo con firma y artefactos	45	8	82%
Predicción con carga de modelo	35	6	83%
TOTAL FLUJO END-TO-END	**395**	**74**	**81%**

Nota: Conteo realizado sobre notebooks de producción comparando implementación manual vs. uso de MomentumML. La reducción del 81% en líneas de código disminuye proporcionalmente la probabilidad de errores humanos, el tiempo de desarrollo, y la carga de mantenimiento. El mayor impacto (90%) se observa en la configuración de entrenamiento con optimización de hiperparámetros. Fuente: elaboración propia.

6.1.2 Métricas de Eficiencia Operacional (Framework DORA-ML)

Adaptando el framework DORA (DevOps Research and Assessment) al contexto de MLOps, se evaluaron las siguientes métricas durante 6 meses de operación:

Tabla 11:

Métricas DORA adaptadas al contexto de MLOps

MÉTRICA DORA-ML	DEFINICIÓN	ANTES	DESPUÉS	MEJORA
Lead Time for Changes	Tiempo experimento → producción	3-4 sem.	5-7 días	71%
Frecuencia de Despliegue	Modelos desplegados/mes	0.5	4.2	740%
Tasa de Fallo	Porcentaje de rollbacks	35%	8%	77%
MTTR	Tiempo de recuperación	48 hrs	4 hrs	92%

Nota: Métricas calculadas sobre 47 ejecuciones de pipelines automatizados en el período de evaluación. El framework DORA es el estándar de la industria para medir eficiencia en DevOps. La adaptación al contexto MLOps considera el ciclo completo desde experimentación hasta producción. La mejora más significativa (740%) se observa en frecuencia de despliegue, pasando de 0.5 a 4.2 modelos por mes. Fuente: elaboración propia.

6.1.3 Análisis de Overhead Computacional

Para evaluar el impacto de la capa de abstracción en latencia y recursos, se midieron tiempos de ejecución en operaciones críticas:

Tabla 12:

Análisis de overhead introducido por la capa de abstracción

OPERACIÓN	SIN WRAPPER	CON MOMENTUMML	OVERHEAD
Registro de modelo en MLflow	2,450 ms	2,780 ms	+13%
Carga de modelo para inferencia	1,200 ms	1,380 ms	+15%
Logging de métricas (batch 1000)	890 ms	980 ms	+10%
Transformación pipeline completo	45,000 ms	46,200 ms	+3%

Nota: Mediciones realizadas en cluster Databricks Standard_E8d_v4 con 4 workers. El overhead introducido (3-15%) es marginal y aceptable considerando la reducción del 81% en código, eliminación de errores humanos, y estandarización automática. El overhead de ~1.5 segundos en un flujo de 45-90 minutos representa <0.03% del tiempo total. Fuente: elaboración propia.

6.2 Validación de Trazabilidad, Reproducibilidad y Escalabilidad

El objetivo general establece "trazabilidad, reproducibilidad y escalabilidad" como criterios fundamentales. A continuación se presenta evidencia empírica para cada dimensión.

6.2.1 Validación de Trazabilidad

La trazabilidad se define como la capacidad de rastrear el linaje completo de un modelo desde los datos de entrada hasta las predicciones en producción.

Tabla 13:

Indicadores de trazabilidad obtenidos durante el período de evaluación

INDICADOR	VALOR OBTENIDO
Experimentos registrados con linaje completo en MLflow	47 ejecuciones
Versiones de modelos trazables en Unity Catalog	23 versiones
Artefactos de configuración preservados automáticamente	100% (YAML + hiperparámetros)
Capacidad de reconstrucción de experimentos históricos	Verificada en 5 casos de prueba
Alertas de drift documentadas con trazabilidad completa	156 alertas procesadas

Nota: Cada modelo registrado incluye automáticamente: parámetros de entrenamiento, métricas de evaluación, artefactos visuales (ROC, confusion matrix, SHAP), referencias a datasets de origen, y alias de versión (champion, challenger). La trazabilidad permite auditorías completas y depuración retrospectiva de errores. Fuente: elaboración propia.

6.2.2 Validación de Reproducibilidad

Para validar reproducibilidad, se seleccionaron 3 modelos en producción y se intentó su replicación exacta en ambiente aislado:

Tabla 14:

Pruebas de reproducibilidad de modelos desplegados en producción

MODELO	MÉTRICA ORIGINAL	MÉTRICA REPRODUCIDA	DIFERENCIA
Propensión de compra (clasificación)	AUC: 0.847	AUC: 0.847	0.00%
Valor del cliente (regresión)	RMSE: 12.45	RMSE: 12.45	0.00%
Segmentación (clustering)	Silhouette: 0.623	Silhouette: 0.623	0.00%

Nota: La reproducibilidad exacta (0.00% diferencia) se logró gracias a: semillas aleatorias configurables y persistidas (seed: 42), versionado de datos en Unity Catalog, y persistencia de pipelines de preprocesamiento junto con modelos. Esto garantiza que cualquier modelo histórico puede recrearse exactamente. Fuente: elaboración propia.

6.2.3 Validación de Escalabilidad

Se realizaron pruebas con volúmenes crecientes para caracterizar el comportamiento del sistema:

Tabla 15:

Pruebas de escalabilidad con volúmenes crecientes de datos

VOLUMEN	TIEMPO ENTRENAMIENTO	TIEMPO INFERENCIA/REG	DBUs
100,000 registros	4 minutos	0.8 ms	12
1,000,000 registros	18 minutos	0.9 ms	45
10,000,000 registros	72 minutos	1.1 ms	180
50,000,000 registros	245 minutos	1.3 ms	720

Nota: El escalamiento es lineal O(n) hasta 10M de registros. A partir de 10M, el escalamiento es sublineal gracias a optimizaciones de particionamiento en PySpark. El tiempo de inferencia por registro se mantiene estable (<1.5ms) independientemente del volumen de entrenamiento, garantizando latencias consistentes en producción. Fuente: elaboración propia.

Capítulo 7:
CONCLUSIONES Y RECOMENDACIONES

El presente trabajo de investigación abordó uno de los desafíos más críticos en la adopción empresarial de inteligencia artificial: la brecha existente entre el desarrollo experimental de modelos de machine learning y su operacionalización confiable, escalable y sostenible en entornos productivos. A través del diseño, implementación y validación empírica de una librería integral de MLOps fundamentada en PySpark y MLflow, esta tesis demostró que es posible establecer un marco metodológico y tecnológico que permita superar las barreras estructurales que históricamente han limitado el despliegue exitoso de modelos ML en contextos reales.

La problemática inicial se sustentó en evidencia científica robusta que documentaba tasas de fracaso superiores al 90% en proyectos de machine learning [30], junto con una fragmentación generalizada de herramientas y procesos que conducía a acumulación de deuda técnica, pérdida de reproducibilidad y ausencia de trazabilidad en el ciclo de vida de los modelos [29, 35]. Frente a este panorama, la solución propuesta adoptó un enfoque holístico que integró principios de ingeniería de software, ciencia de datos y operaciones, materializándose en una arquitectura modular que automatiza tareas críticas como la preparación y validación de datos, el entrenamiento y reentrenamiento de modelos, el registro y versionado de artefactos, el despliegue controlado y el monitoreo continuo del desempeño.

La validación empírica de la librería en un contexto organizacional real arrojó resultados que exceden significativamente las expectativas iniciales y que confirman las hipótesis planteadas en los objetivos de investigación. Los hallazgos cuantitativos revelan un impacto transformador en múltiples dimensiones operacionales, técnicas y económicas, posicionando a la solución propuesta como una contribución sustantiva tanto para la práctica profesional como para el avance del campo de MLOps.

A continuación se presentan las conclusiones derivadas de manera lógica de las premisas evaluadas, los objetivos planteados, la metodología aplicada y los resultados empíricos obtenidos:

- La arquitectura modular basada en PySpark y MLflow demostró ser una solución técnicamente viable y operacionalmente efectiva para orquestar el ciclo de vida completo de modelos de machine learning. La implementación en un entorno real permitió el despliegue exitoso de 35 modelos hacia el entorno de aseguramiento de calidad (QA) de un total de 85 modelos operativos, representando el 41.2% del inventario total y constituyendo un hito sin precedentes en la historia organizacional. Esta tasa de éxito contrasta favorablemente con los rangos de 0-20% reportados en la literatura especializada [18, 28], validando empíricamente la hipótesis de que la adopción de prácticas estructuradas de MLOps puede mitigar significativamente las barreras que históricamente han impedido la operacionalización de modelos ML.
- La adopción de pipelines reproducibles y el uso de control de versiones automático mediante MLflow facilitaron la trazabilidad exhaustiva de experimentos, modelos y datasets, contribuyendo sustancialmente a la mitigación de la deuda técnica oculta que tradicionalmente entorpece el mantenimiento de sistemas de ML en producción [29]. La capacidad de auditar resultados, replicar experimentos bajo condiciones controladas y realizar análisis comparativos retrospectivos constituye una mejora cualitativa crítica que trasciende las métricas puramente cuantitativas.
- La incorporación de módulos automáticos de monitoreo permitió la identificación temprana de variaciones en la distribución de datos (data drift) y en el comportamiento de modelos en producción. Esta capacidad de detección proactiva habilita la implementación de estrategias de reentrenamiento adaptativo bajo condiciones controladas, evitando degradaciones silenciosas del desempeño que podrían comprometer la confiabilidad de las predicciones en entornos reales [11, 16].
- Un resultado de particular relevancia estratégica y económica radica en la reducción del 40% en el consumo de Unidades de Databricks (DBUs), derivada de la implementación sistemática de mejores prácticas de optimización en procesamiento distribuido. Esta mejora se fundamentó en cinco estrategias técnicas principales: (i) uso preferencial de DataFrames sobre RDDs para aprovechar los optimizadores Catalyst y Tungsten; (ii) implementación inteligente de mecanismos de caché y persistencia; (iii) optimización dinámica del particionamiento de datos; (iv) reducción

de operaciones de shuffle mediante broadcast joins; y (v) minimización del uso de User Defined Functions (UDFs) a favor de funciones nativas de PySpark [5, 15, 20, 32, 34].

- El diseño modular y extensible de la librería, fundamentado sobre tecnologías ampliamente adoptadas y de código abierto (PySpark, MLflow), garantiza su adaptabilidad a diversos contextos organizacionales, su capacidad de integración con plataformas existentes y su escalabilidad frente a incrementos en volumen de datos o complejidad de modelos. Esta característica facilita la democratización de prácticas avanzadas de MLOps, permitiendo que organizaciones de diversos tamaños y sectores puedan adoptarlas sin requerir inversiones prohibitivas en infraestructura propietaria [25].
- La evaluación integral de la librería reveló mejoras sustanciales en múltiples indicadores clave de desempeño: reducción del 40% en tiempos de creación de modelos, disminución del 25% en costos operativos de despliegue, incremento del 50% en productividad de equipos de ciencia de datos, y elevación de la tasa de éxito en despliegue de 0% a 41.2%. Este conjunto de resultados confirma que la solución propuesta no solo es técnicamente viable, sino que genera valor tangible y medible a nivel organizacional, contribuyendo tanto a la eficiencia operativa como a la capacidad de innovación basada en datos.

En aras de mantener la rigurosidad académica, es necesario reconocer las limitaciones inherentes al presente trabajo de investigación:

- La validación de la librería se realizó dentro del contexto de una operadora de telecomunicaciones, con características particulares en cuanto a volumen y naturaleza de datos (procesamiento distribuido de datasets con más de 10 millones de registros de comportamiento de usuarios, eventos de red y transacciones comerciales), infraestructura tecnológica basada en computación distribuida (Databricks/PySpark), cultura organizacional y dominio de aplicación específico del sector de telecomunicaciones. Si bien los resultados obtenidos son prometedores y se alinean con hallazgos reportados en la literatura especializada, la generalización de estos

resultados a otros contextos organizacionales o sectores industriales requiere validaciones adicionales en entornos diversos.

- El período de evaluación empírica, si bien suficiente para evidenciar mejoras significativas en indicadores operacionales, puede no capturar completamente comportamientos emergentes de largo plazo, tales como efectos de escala ante incrementos exponenciales en volumen de datos o modelos, o dinámicas de adopción y apropiación tecnológica por parte de diferentes perfiles de usuarios dentro de la organización.
- Aunque la librería fue diseñada con principios de extensibilidad, su validación empírica se concentró principalmente en algoritmos y frameworks ampliamente utilizados en la organización participante. La cobertura de frameworks emergentes, modelos de deep learning de gran escala (Large Language Models) y técnicas de vanguardia como federated learning o reinforcement learning requiere extensiones adicionales y validaciones específicas.
- Si bien la librería incorpora prácticas de versionado y trazabilidad que contribuyen indirectamente a la gobernanza de modelos, el estudio no abordó de manera exhaustiva aspectos avanzados de seguridad, tales como protección contra ataques adversarios, envenenamiento de datos, o mecanismos de privacidad diferencial. Estas dimensiones, de creciente relevancia en el contexto de MLOps confiable (*Trustworthy MLOps*), constituyen áreas que merecen atención en trabajos futuros.

Con base en los hallazgos obtenidos, las limitaciones identificadas y las tendencias emergentes en el campo de MLOps, se proponen las siguientes líneas de trabajo futuro:

- La irrupción de modelos de lenguaje de gran escala (Large Language Models - LLMs) ha introducido desafíos específicos que trascienden las prácticas tradicionales de MLOps, tales como gestión de prompts, optimización de contextos, fine-tuning eficiente y evaluación de alucinaciones. Se recomienda extender la librería para incorporar capacidades de LLMOps, incluyendo frameworks para prompt engineering, evaluación sistemática de respuestas generativas, monitoreo de costos

asociados a inferencia en modelos masivos y estrategias de optimización específicas para arquitecturas transformer [34].

- La confiabilidad y transparencia de sistemas de IA constituyen requisitos cada vez más críticos, especialmente en sectores regulados. Se propone incorporar módulos para explicabilidad de modelos (XAI), detección de sesgos algorítmicos, análisis de fairness, robustez ante ataques adversarios y mecanismos de privacidad diferencial. Esta extensión posicionaría a la librería como una herramienta integral para implementación de MLOps confiable, alineándose con marcos regulatorios emergentes como el AI Act de la Unión Europea [35].
- La evolución hacia paradigmas de inteligencia continua (*Continuous Intelligence*) requiere la incorporación de capacidades avanzadas de monitoreo en tiempo real, análisis de telemetría de modelos, detección automática de anomalías en predicciones y activación de políticas de respuesta adaptativa. Se recomienda desarrollar dashboards interactivos basados en tecnologías de Business Intelligence (Grafana, Power BI) que permitan visualización en tiempo real del estado de salud de modelos en producción, facilitando la toma de decisiones informadas por parte de equipos operacionales.
- Si bien la librería facilita la automatización de diversos componentes del ciclo de vida de modelos, la integración nativa con pipelines de Continuous Integration/Continuous Deployment (CI/CD) puede profundizarse. Se propone desarrollar plantillas de configuración para orquestadores populares (GitHub Actions, GitLab CI, Jenkins, Azure DevOps) que permitan el despliegue continuo de la librería y de los modelos generados, incluyendo pruebas automatizadas de regresión, validación de esquemas de datos, pruebas de integración y despliegue progresivo con estrategias canary o blue-green [22, 31].
- En contextos donde múltiples organizaciones colaboran en el desarrollo de soluciones de IA (por ejemplo, consorcios de investigación, cadenas de suministro integradas, ecosistemas de salud pública), surgen desafíos específicos relacionados con integración de datos distribuidos, preservación de confidencialidad, estandarización de interfaces y coordinación de despliegues. Se recomienda explorar extensiones de la

librería que faciliten la implementación de MLOps en configuraciones multi-organización, incorporando conceptos de federated learning, privacidad diferencial y contratos de datos (*data contracts*).

- La creciente preocupación por el impacto ambiental de operaciones de IA motiva la investigación en técnicas de *Green MLOps*, orientadas a minimizar el consumo energético asociado a entrenamiento e inferencia de modelos sin comprometer su desempeño. Se propone incorporar módulos de monitoreo de huella de carbono, optimización de arquitecturas de modelos para eficiencia energética, y estrategias de *early stopping* y *neural architecture search* que prioricen la sostenibilidad como métrica de optimización [39].
- La maduración del campo de MLOps requiere mecanismos robustos de gobernanza que aseguren el cumplimiento de regulaciones de privacidad (GDPR, LGPD, CCPA), estándares de calidad (ISO/IEC 25059, ISO/IEC 23894) y marcos de gestión de riesgos (NIST AI Risk Management Framework). Se recomienda implementar un módulo de auditoría que registre metadatos exhaustivos de cada ejecución (usuario, timestamp, parámetros, versión de librería, linaje de datos), genere reportes automáticos de cumplimiento y facilite la trazabilidad end-to-end requerida por auditorías internas y externas.

REFERENCIAS

[1] M. Allurwar and G. Dhule, "MLOps – A complete guide for operationalizing machine learning," Int. Res. J. Eng. Technol. (IRJET), vol. 8, no. 6, pp. 4370–4373, 2021.

[2] Amdocs, "Cloud-based machine learning operations (MLOps) drives 2.3% revenue uplift for FinTech unicorn," 2024. [Online]. Available: https://www.amdocs.com/insights/case-study/cloud-based-machine-learning-operations-mlops-drives-23-revenue-uplift-fintech

[3] S. Amershi et al., "Software engineering for machine learning: A case study," in Proc. 41st Int. Conf. Softw. Eng.: Softw. Eng. Practice (ICSE-SEIP), Montreal, QC, Canada, 2019, pp. 291–300.

[4] Analytics Vidhya, "Best practices and performance tuning activities for PySpark," 2024. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/08/best-practices-and-performance-tuning-activities-for-pyspark/

[5] BinaryScripts, "Optimizing PySpark applications for large data processing," 2024. [Online]. Available: https://binaryscripts.com/spark/2024/12/19/optimizing-pyspark-applications-for-large-data-processing.html

[6] E. Breck, S. Cai, E. Nielsen, M. Salib, and D. Sculley, "The ML test score: A rubric for ML production readiness and technical debt reduction," Google Research, 2017. [Online]. Available: https://arxiv.org/abs/1706.08536

[7] ChaosGenius, "7 ways to optimize Apache Spark performance," 2025. [Online]. Available: https://www.chaosgenius.io/blog/spark-performance-tuning/

[8] ChaosGenius, "10 tips to reduce Databricks costs," 2025. [Online]. Available: https://www.chaosgenius.io/blog/databricks-optimization-techniques/

[9] Databricks Community, "PySpark optimizations and best practices," 2023. [Online]. Available: https://community.databricks.com/t5/data-engineering/pyspark-optimizations-and-best-practices/td-p/10456

[10] DEV Community, "PySpark optimization techniques," 2024. [Online]. Available: https://dev.to/rado_mayank/pyspark-optimization-techniques-56l0

[11] B. Eken, S. Pallewatta, L. E. Lwakatare, A. van der Hoek, and J. Bosch, "A multivocal review of MLOps practices, challenges and open issues," arXiv preprint, arXiv:2406.09737, 2023.

[12] Finout, "Databricks cost optimization," 2025. [Online]. Available: https://www.finout.io/blog/optimize-databricks-costs

[13] Google Cloud, "Practitioners guide to MLOps: A framework for continuous delivery and automation of machine learning," White Paper, 2021. [Online]. Available: https://services.google.com/fh/files/misc/practitioners_guide_to_mlops_whitepaper.pdf

[14] HatchWorks, "Databricks MLOps: Simplifying your machine learning operations," 2025. [Online]. Available: https://hatchworks.com/blog/databricks/databricks-mlops/

[15] J. Leung, "Optimizing the data processing performance in PySpark," Towards Data Science, 2025. [Online]. Available: https://towardsdatascience.com/optimizing-the-data-processing-performance-in-pyspark-4b895857c8aa

[16] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," IEEE Trans. Knowl. Data Eng., vol. 31, no. 12, pp. 2346–2363, 2018.

[17] MarkovML, "Model scalability: Scaling ML models for large data," 2024. [Online]. Available: https://www.markovml.com/blog/model-scalability

[18] A. Mendoza, "The ultimate guide to ML model deployment in 2024," 2024. [Online]. Available: https://www.axelmendoza.com/posts/ml-model-deployment/

[19] Microsoft Learn, "Best practices for cost optimization - Azure Databricks," 2024. [Online]. Available: https://learn.microsoft.com/en-us/azure/databricks/lakehouse-architecture/cost-optimization/best-practices

[20] A. Mishra, "PySpark optimization: Best practices for better performance," Medium, 2025. [Online]. Available: https://medium.com/@Ankur.id/pyspark-optimization-best-practices-for-better-performance-d15806ab34b9

[21] T. M. Mitchell, Machine Learning. New York, NY, USA: McGraw-Hill, 1997.

[22] G. Nemade, "Best practices for deploying machine learning models in production," Medium, 2025. [Online]. Available: https://medium.com/@nemagan/best-practices-for-deploying-machine-learning-models-in-production-10b690503e6d

[23] F. Piva, "Simple ways to improve your PySpark and Parquet pipeline performance," Medium, 2024. [Online]. Available: https://medium.com/@fetpiva/simple-ways-to-improve-your-pyspark-and-parquet-pipeline-performance-de1e32063f00

[24] Qwak, "How to build an end to end ML pipeline in 2024," 2024. [Online]. Available: https://www.qwak.com/post/end-to-end-machine-learning-pipeline

[25] Qwak, "Top end to end MLOps platforms and tools in 2024," 2024. [Online]. Available: https://www.qwak.com/post/top-mlops-end-to-end

[26] Restack, "MLflow Databricks pricing guide," 2024. [Online]. Available: https://www.restack.io/docs/mlflow-knowledge-mlflow-databricks-pricing

[27] S. Salama, S. Kazmierczak, and N. Schut, "Practitioners guide to MLOps," Google Cloud, White Paper, 2021.

[28] Sciforce, "MLOps as the key to efficient AI model deployment and maximum ROI," Medium, 2024. [Online]. Available: https://medium.com/sciforce/mlops-as-the-key-to-efficient-ai-model-deployment-and-maximum-roi-d9b96cea8412

[29] D. Sculley et al., "Hidden technical debt in machine learning systems," in Advances in Neural Information Processing Systems (NeurIPS), vol. 28, 2015, pp. 2503–2511.

[30] S. Shankar, K. Holstein, A. Wortman Vaughan, J. Heer, J. M. Hellerstein, and A. G. Parameswaran, "We have no idea how models will behave in production until production: How engineers operationalize machine learning," Proc. ACM Hum.-Comput. Interact., vol. 8, no. CSCW1, Art. no. 206, 2024.

[31] Sigmoid, "5 practices deploying ML models in production," 2025. [Online]. Available: https://www.sigmoid.com/blogs/5-best-practices-for-deploying-ml-models-in-production/

[32] Spark By Examples, "Spark performance tuning & best practices," 2024. [Online]. Available: https://sparkbyexamples.com/spark/spark-performance-tuning/

[33] Statsig, "Deploying machine learning models in production: A guide for engineers," 2025. [Online]. Available: https://www.statsig.com/perspectives/deploying-machine-learning-models-in-production-guide

[34] Toptal, "Apache Spark optimization techniques," 2024. [Online]. Available: https://www.toptal.com/spark/apache-spark-optimization-techniques

[35] Z. Zhao, Y. Chen, A. A. Bangash, B. Adams, and A. E. Hassan, "An empirical study of challenges in machine learning asset management," Empirical Softw. Eng., vol. 29, no. 3, 2024, doi: 10.1007/s10664-024-10474-4.

[36] C. Amrit and A. K. Narayanappa, "An analysis of the barriers preventing the implementation of MLOps," in Proc. IFIP Advances in Information and Communication Technology, Springer, 2024, doi: 10.1007/978-3-031-55642-5_11.

[37] F. Bayram and B. S. Ahmed, "Towards trustworthy machine learning in production: An overview of the robustness in MLOps approach," arXiv preprint, arXiv:2410.21346, 2024.

[38] European Union, "Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence (AI Act)," Official J. Eur. Union, 2024.

[39] P. Gupta, D. Srivastava, M. Sahu, and S. Tiwari, "Green MLOps to green GenOps: An empirical study of energy consumption in discriminative and generative AI operations," Information, vol. 16, no. 4, Art. no. 281, 2025, doi: 10.3390/info16040281.

[40] D. Kreuzberger, N. Kühl, and S. Hirschl, "Machine learning operations (MLOps): Overview, definition, and architecture," IEEE Access, vol. 11, pp. 31866–31879, 2023, doi: 10.1109/ACCESS.2023.3262138.

[41] S. Makinen, H. Skogstrom, E. Laaksonen, and T. Mikkonen, "Who needs MLOps: What data scientists seek to accomplish and how can MLOps help?," in Proc. IEEE/ACM 1st Workshop AI Eng.—Softw. Eng. AI (WAIN), 2021, pp. 109–112, doi: 10.1109/WAIN52551.2021.00024.

[42] I. Martinez and S. Kifle, "MLOps as a holistic framework for AI systems development and deployment," J. Artif. Intell. Res., vol. 19, no. 2, pp. 145–168, 2024.

[43] T. Mikkonen, C. Lassenius, T. Männistö, M. Oivo, and J. Järvinen, "MLOps challenges in multi-organization setup: Experiences from two real-world cases," in Proc. IEEE/ACM 1st Workshop AI Eng.—Softw. Eng. AI (WAIN), 2021, pp. 82–88.

[44] NIST, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," National Institute of Standards and Technology, U.S. Department of Commerce, 2023, doi: 10.6028/NIST.AI.100-1.

[45] A. Paleyes, R.-G. Urma, and N. D. Lawrence, "Challenges in deploying machine learning: A survey of case studies," ACM Comput. Surv., vol. 55, no. 6, Art. no. 114, 2022, doi: 10.1145/3533378.

[46] V. Rao and R. Shankar, "Navigating MLOps: Insights into maturity, lifecycle, tools, and careers," arXiv preprint, arXiv:2503.15577, 2025.

[47] P. Ruf, M. Madan, C. Reich, and D. Ould-Abdeslam, "Demystifying MLOps and presenting a recipe for the selection of open-source tools," Appl. Sci., vol. 11, no. 19, Art. no. 8861, 2021, doi: 10.3390/app11198861.

[48] G. Symeonidis, E. Nerantzis, A. Kazakis, and G. A. Papakostas, "MLOps—Definitions, tools and challenges," in Proc. IEEE 12th Annu. Comput. Commun. Workshop Conf. (CCWC), Las Vegas, NV, USA, 2022, pp. 453–460, doi: 10.1109/CCWC54503.2022.9720902.

[49] M. Testi et al., "MLOps: A taxonomy and a methodology," IEEE Access, vol. 10, pp. 63606–63618, 2022, doi: 10.1109/ACCESS.2022.3181730.

[50] Y. Zhou, Y. Yu, and B. Ding, "Towards MLOps: A case study of ML pipeline platform," in Proc. Int. Conf. Artif. Intell. Comput. Eng. (ICAICE), 2020, pp. 494–500.

[51] Google Cloud, "DORA's software delivery metrics: The four keys," 2023. [Online]. Available: https://dora.dev/guides/dora-metrics-four-keys/

[52] J. Diaz-De-Arcaya, A. I. Torre-Bastida, G. Zárate, R. Miñón, and A. Almeida, "A joint study of the challenges, opportunities, and roadmap of MLOps and AIOps: A systematic survey," ACM Comput. Surv., vol. 56, no. 4, pp. 1–30, 2023, doi: 10.1145/3625289.