JIRRS
Journal International Review of Research Studies

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

WHY ML MODELS FAIL TO REACH PRODUCTION

Otávio Luís Pinheiro Oliveira
AlfaUnipac University Center, Brazil
Corresponding author: Otavioluispo@gmail.com

Abstract

Machine Learning (ML) models are developed at an unprecedented scale, yet only a fraction ever reach production — and among those that do, many fail within months. This article examines the organizational, technical, and cultural factors that prevent ML models from being successfully operationalized, and presents structured practices to address them. Through a narrative bibliographic review drawing on peer-reviewed literature and technical documentation from recognized industry organizations, the article maps seven primary barriers to ML deployment: misalignment between data science and engineering teams, data quality and governance failures, lack of reproducibility, absence of robust testing, inadequate infrastructure, neglected monitoring, and organizational resistance. As a response to these challenges, MLOps — the convergence of Machine Learning, DevOps, and Data Engineering — emerges as a systematic framework for treating models as products rather than experiments. The article describes the full ML model lifecycle in production, surveys the current tooling ecosystem, and outlines practical strategies for building an MLOps culture incrementally, regardless of organizational size. Real-world cases, including Spotify's large-scale ML platform and lessons from the regulated financial sector, illustrate how structured operationalization practices translate into measurable business outcomes. The findings suggest that the primary determinant of ML success in production is not algorithmic sophistication, but the organizational and engineering discipline surrounding model deployment, monitoring, and continuous improvement.

Keywords: machine learning operations; MLOps; model deployment; technical debt; data drift; reproducibility; ML lifecycle; experiment tracking; model monitoring; LLMOps.

1. Introduction

Machine Learning (ML) exists in a paradox: research output has never been greater, yet functional real-world deployments remain remarkably scarce. ML has evolved

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

from a purely academic discipline into a field capable of addressing concrete business problems — however, bringing models into production entails numerous challenges and difficulties (Paleyes et al., 2022).

The data are concerning, though they must be interpreted with methodological caution. Frequently cited figures suggest that only 10% to 32% of developed models ever reach production, and among those that do, many fail within months (Sculley et al., 2015). However, these estimates derive from surveys conducted predominantly in technology-adjacent organizations and large enterprises with established ML programs — populations that are not representative of the full spectrum of organizations attempting ML deployment. The lower bound of 10% originates from industry practitioner surveys with self-selected respondents, while the upper bound of 32% reflects estimates from consulting reports with limited methodological transparency. Neither figure has been validated through controlled longitudinal study. What the evidence does support consistently — across Paleyes et al. (2022), Recupito et al. (2025), and Woźniak et al. (2025) — is the qualitative pattern: a substantial proportion of ML projects fail to reach or sustain production, and the causes are predominantly organizational and operational rather than algorithmic. The precise magnitude of this gap is less important than its structural character, which is the focus of this article.

What distinguishes organizations that successfully deliver ML solutions from those that remain confined to experimental environments is not algorithmic quality. The primary obstacles are non-technical in nature: insufficient organizational support, lack of structured planning, and misalignment between the developed model and actual business objectives (Zi, 2024). Organizations that overcome these barriers tend to adopt structured practices that integrate model development, deployment, and continuous monitoring (Kreuzberger et al., 2023).

This article aims to identify these bottlenecks and present practical solutions to address them.

2. Methodological Clarification

This article adopts a narrative bibliographic review approach with scoping orientation, drawing on academic literature indexed in IEEE Xplore, ACM Digital Library, and ScienceDirect, supplemented by technical reference documentation from recognized industry organizations. The review protocol was structured following adapted

JIRRS
Journal International Review of Research Studies

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

elements of the PRISMA-ScR guidelines (Preferred Reporting Items for Systematic Reviews and Meta-Analyses extension for Scoping Reviews), adjusted to reflect the applied and multivocal nature of the MLOps literature.

2.1 Search Strategy

Searches were conducted between January and February 2025. The following query strings were applied across the three academic databases:

```
-"MLOps" AND ("model deployment" OR "production ML")

-"machine learning operations" AND ("pipeline" OR "lifecycle"
OR "monitoring")

-"ML deployment" AND ("failure" OR "barriers" OR "challenges")

-"technical debt" AND "machine learning"

-"LLMOps" OR "large language model operations"

-"model monitoring" AND ("data drift" OR "concept drift")

-"feature store" AND "machine learning"

-"reproducibility" AND "machine learning systems"
```

Searches were limited to publications from **2018 to 2025**, given that MLOps as a formalized discipline emerged after 2018 (Kreuzberger et al., 2023). Seminal works predating this window — specifically Sculley et al. (2015) on technical debt — were retained by exception due to their foundational status and citation density in the field.

2.2 Screening and Selection

The initial search returned approximately **410 documents** across the three databases. After removal of duplicates and application of title/abstract screening, **87 documents** advanced to full-text review. Of these, **34 peer-reviewed works** were retained for inclusion in the synthesis, based on direct relevance to the research question (barriers to ML deployment and MLOps practices as a structured response).

2.3 Inclusion and Exclusion Criteria

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

Included:

-Peer-reviewed journal articles, conference proceedings, and systematic/multivocal reviews addressing ML deployment, MLOps practices, or ML system operationalization

-Empirical case studies documenting real deployment challenges or MLOps adoption outcomes

-Technical documentation and grey literature produced by organizations with recognized standing in the field (Google, Carnegie Mellon University, AWS, IEEE), where such sources provided empirical evidence or architectural specifications not available in strictly academic literature

Excluded:

-Publications focused exclusively on model architecture or algorithmic performance without addressing deployment or operational concerns

-Opinion pieces, editorials, and blog posts without empirical grounding or institutional backing

-Works published prior to 2018, except foundational references retained by explicit exception

-Studies limited to a single narrow application domain without generalizable findings

2.4 Treatment of Grey Literature

The inclusion of grey literature — technical documentation, industry reports, and engineering blog posts from recognized organizations — follows the multivocal review methodology as described and justified by Recupito et al. (2025). This approach is explicitly appropriate for applied engineering fields where practitioner knowledge precedes formal academic codification. MLOps is a recognized instance of this pattern: as documented by Woźniak et al. (2025), a significant portion of operational best practices in the field originates from industry experience and has not yet been fully captured in peer-reviewed literature. Grey literature sources were evaluated against three criteria before inclusion: (1) institutional or organizational authorship with identifiable accountability, (2) presence of empirical grounding or reproducible technical specifications, and (3) relevance to the research question. Personal blog posts and vendor marketing materials without empirical content were excluded.

JIRRS Journal International Review of Research Studies

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

2.5 Scope and Limitations of the Method

This article does not aim to produce primary data, but rather to synthesize the current state of knowledge regarding the operationalization bottlenecks of ML models and their associated practical solutions, in a manner accessible to both practitioners and researchers in the field. As a narrative review with scoping orientation, it does not apply meta-analytic aggregation or formal inter-rater reliability scoring. The inherent limitations of this approach — including the potential for selection bias introduced by the predominance of industry sources, the absence of quantitative effect-size synthesis, and the sector-specific nature of several case studies — are discussed explicitly in Section 11.

3. Theoretical Framework and Epistemological Positioning

This work is epistemologically positioned within the pragmatist paradigm. Pragmatism is not grounded in a rigid ontological stance — neither purely objectivist nor relativist — but orients knowledge production toward the practical utility of results and the capacity to address concrete problems (Pretorius, 2024). From this perspective, the central question is not "what is true in the abstract," but rather "what works in practice" — which aligns directly with the focus of this article: identifying why ML models fail to reach production and what can be done to change that (Lim, 2023).

Within this epistemological orientation, the pragmatist framework carries three specific implications for how ML systems are evaluated and understood in this article. First, it redefines the criteria for model success: a model is not successful because it achieves high accuracy on a held-out test set, but because it delivers reliable, measurable utility within an operational environment over time. Accuracy, F1 score, and AUC-ROC are treated here as instrumental indicators — necessary but not sufficient conditions for success — rather than as ends in themselves. A model that achieves 97% accuracy in a notebook and fails silently in production within three months is, from a pragmatist standpoint, an unsuccessful system regardless of its benchmark performance (Sculley et al., 2015; Paleyes et al., 2022). Second, the pragmatist framework reframes algorithmic performance metrics as contextually bounded: a metric is only meaningful insofar as it reflects the conditions of actual deployment. This position motivates the emphasis throughout this article on production monitoring, data drift detection, and continuous retraining — practices that exist precisely because offline metrics do not generalize unconditionally to operational contexts (Hinder et al., 2024). Third, and most consequentially for the

JIRRS
Journal International Review of Research Studies
Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

organizational analysis developed in Sections 5 and 8, pragmatism prioritizes operational reliability over theoretical optimality. The most sophisticated model is not the most valuable one — the most valuable model is the one that is running, monitored, and continuously improved. This principle, which recurs as a practical conclusion in several sections of this article, is not merely a heuristic; it is a direct expression of the pragmatist epistemological commitment to knowledge evaluated by its consequences in practice (Lim, 2023; Pretorius, 2024).

The theoretical framework underpinning this analysis departs from the concept of technical debt in ML systems, introduced by Sculley et al. (2015), which demonstrated that model code represents only a fraction of the actual production system — the remainder comprising infrastructure, data pipelines, monitoring, and organizational processes. This finding is itself a pragmatist result: it emerged not from theoretical deduction but from the operational experience of engineering teams confronting the consequences of treating models as isolated artifacts rather than as components of sociotechnical systems. Building on this foundation, MLOps emerges as a systemic response: a set of practices integrating software engineering, DevOps, and data science to treat the model as a product rather than an experiment (Kreuzberger et al., 2023). A sociotechnical perspective also informs this work: the barriers to ML operationalization are not solely technical, but also cultural and organizational in nature — requiring an analysis that extends beyond tooling to encompass the processes, roles, and structures of organizations (Paleyes et al., 2022).

It is nonetheless important to engage critically with the foundational literature rather than treating it as uniformly authoritative. The technical debt framework of Sculley et al. (2015), while intellectually generative, was developed from the operational experience of Google-scale engineering organizations — contexts characterized by large dedicated infrastructure teams, significant tooling investment, and ML systems of a complexity and scale that most organizations will never encounter. Applying this framework unreflectively to smaller organizations or early-stage ML adoption risks pathologizing normal developmental states as debt, and may produce prescriptions — comprehensive monitoring infrastructure, automated retraining pipelines, full CI/CD for ML — that are technically correct but organizationally premature. Kreuzberger et al. (2023) partially address this limitation through their maturity model, which contextualizes practices by organizational level, but their framework retains an implicit teleology: Level 2 full automation is presented as the desirable end state, without sufficient interrogation of whether that end state is appropriate or achievable for the majority of organizations deploying ML. Paleyes et al. (2022), by contrast, adopt a more empirically grounded position — their findings emerge from documented failure cases rather than from prescriptive architecture, and their

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

conclusions are correspondingly more cautious about generalizing from large-scale deployments to the broader population of ML projects. The tension between these two approaches — prescriptive framework versus empirical case analysis — is productive rather than resolvable, and both perspectives inform the synthesis developed in this article.

4. What Is MLOps and Why Does It Exist

4.1 Definition of MLOps

MLOps is the convergence of Machine Learning, DevOps, and Data Engineering. Its purpose is to automate and standardize the entire model lifecycle — from data collection to production monitoring. While DevOps addresses software code, MLOps introduces ML-specific stages: data preparation, training, validation, deployment, and continuous retraining (Kreuzberger et al., 2023). In practice, this means treating ML models with the same rigor applied to any software system: version control, automated testing, continuous delivery, and monitoring (Google Cloud, 2024).

4.2 The Difference Between a Model That "Works" and a Model "in Production"

A model that performs well in a Jupyter Notebook is not necessarily a reliable system. The real challenge is not building the model — it is integrating it into a production environment and keeping it functional over time (Google Cloud, 2024). In production, data distributions shift, volumes scale, other systems depend on model predictions, and any failure has a direct business impact. While a prototype operates in a controlled environment, a production system must contend with unpredictable data, latency constraints, security requirements, and constant updates (DeCApria, 2024). This distinction is frequently underestimated by development teams, which accounts for a substantial share of deployment failures.

4.3 A Brief History

MLOps emerged from the practical experience of large organizations that attempted to scale ML and encountered significant obstacles along the way. The intellectual landmark of the field was a paper by Google engineers demonstrating how ML systems accumulate hidden maintenance costs far exceeding those expected of conventional software — a phenomenon that became known as "technical debt" in ML (Sculley et al., 2015). Shortly thereafter, companies such as Uber launched internal ML platforms to address these challenges at scale, making clear that ad hoc

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

solutions were insufficient (DeCApria, 2024). The term MLOps gained traction from 2018 onward and consolidated as a discipline once it became evident that operationalizing ML requires a systematic approach, not merely strong algorithms (Kreuzberger et al., 2023).

The historical narrative of MLOps, as it is typically told — including in the account above — deserves critical scrutiny. The canonical origin story centers on large technology companies: Google's technical debt paper, Uber's Michelangelo platform, Facebook's FBLearner Flow. This framing is accurate as a description of where MLOps tooling and vocabulary originated, but it carries an implicit bias: it presents the operational challenges of organizations deploying thousands of models as the defining problems of the field, and their solutions as the reference architecture for all practitioners. The result is a body of literature and tooling optimized for scale that smaller organizations are expected to adopt and simplify downward — a direction of travel that is operationally inefficient and sometimes counterproductive. Recupito et al. (2025) partially correct this bias through their multivocal review methodology, which incorporates practitioner experience from a broader range of organizational contexts. Their findings suggest that the most consequential MLOps challenges for the majority of organizations are not the scaling problems that motivated Michelangelo, but the foundational operationalization problems — reproducibility, monitoring, inter-team alignment — that precede scale entirely. This distinction has direct implications for how the recommendations in this article should be interpreted and prioritized.

5. The Seven Primary Reasons Why Models Fail to Reach Production

The identification of recurring barriers to ML deployment is not the product of a single study, but of a converging body of evidence accumulated across multiple independent research efforts. Paleyes et al. (2022), in a systematic survey of real deployment case studies, identified organizational misalignment, data quality failures, and inadequate infrastructure as the dominant causes of project abandonment. Recupito et al. (2025), in a multivocal review integrating both academic and practitioner literature, confirmed these patterns and extended them to include monitoring negligence and reproducibility failures as structurally recurrent across organizations of different sizes and sectors. Woźniak et al. (2025), in a systematic literature review of MLOps components and metrics, further corroborated that the barriers to operationalization cluster consistently around three structural dimensions: **data and environment**, **engineering and infrastructure**, and **organizational and cultural factors**.

JIRRS
Journal International Review of Research Studies
Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

This convergence across methodologically distinct studies — a systematic survey of case studies, a multivocal review, and a systematic literature review — provides the empirical grounding for the taxonomy adopted in this article. The seven barriers identified below are not an arbitrary enumeration; they represent the failure modes that appear with the highest frequency and cross-study consistency in the literature reviewed. Where studies diverge — for instance, in the relative weight assigned to cultural versus technical factors — those tensions are noted explicitly within the relevant subsections.

For analytical purposes, the seven barriers are organized into three conceptual clusters:

Cluster A — Data and Environment Failures: misalignment between training and production data environments (§5.2), and lack of reproducibility (§5.3)

Cluster B — Engineering and Infrastructure Failures: absence of robust testing and validation (§5.4), and inadequate or nonexistent infrastructure (§5.5)

Cluster C — Organizational and Process Failures: misalignment between data science and engineering teams (§5.1), neglected monitoring (§5.6), and organizational and cultural barriers (§5.7)

This clustering reflects the sociotechnical framing adopted throughout the article: deployment failures are rarely attributable to a single isolated cause, but to the interaction between technical deficiencies and the organizational structures within which they occur (Paleyes et al., 2022; Kreuzberger et al., 2023).

5.1 Misalignment Between Data Science and Engineering

(Cluster C — Organizational and Process Failures)

One of the most prevalent issues is the lack of communication between those who build the model and those who will operate it. Data teams and infrastructure teams work with different tools, processes, and priorities, generating conflict and rework (Zi, 2024). Models are developed without consideration of how they will be served, in what environment they will run, or what latency is acceptable. The result is that, at deployment time, the model must be rewritten — or is simply abandoned.

This barrier appears consistently across the literature, but with differing analytical emphasis. Paleyes et al. (2022) frame it primarily as a process failure — the absence of shared requirements between research and engineering phases — while Recupito

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

et al. (2025) locate it within a broader cultural pattern in which data science and software engineering communities operate with distinct professional norms and incentive structures. Woźniak et al. (2025) further note that this misalignment is particularly acute in organizations that adopted ML incrementally, where data science teams were established before MLOps infrastructure existed, creating structural silos that are difficult to dissolve retroactively. Multidisciplinary teams that align production requirements from the outset consistently achieve significantly better outcomes (Paleyes et al., 2022).

5.2 Data Quality and Governance

(Cluster A — Data and Environment Failures)

Training data that does not represent the real operational environment is a frequent cause of silent failure. When a model reaches production, the data it encounters differs from what it learned on, and performance degrades without any alert being triggered (Sahiner et al., 2023). This phenomenon is referred to as data drift — a shift in the distribution of input data — and concept drift — a change in the relationship between variables and the expected outcome (Hinder et al., 2024). Ignoring these risks from the outset of a project is a common mistake, as is the absence of reliable, traceable, and automated data pipelines (Recupito et al., 2025).

Paleyes et al. (2022) identify data issues as the single most frequently cited cause of deployment failure across their survey of case studies, appearing in a majority of the documented incidents. Recupito et al. (2025) corroborate this finding and add that data governance failures — including the absence of lineage tracking and unclear data ownership — compound the technical problem with an organizational one: even when drift is detected, the absence of accountable data owners delays corrective action. This convergence suggests that data quality is not merely a preprocessing concern, but a structural governance challenge that must be addressed at the organizational level, not only at the pipeline level.

5.3 Lack of Reproducibility

(Cluster A — Data and Environment Failures)

"It works on my machine" is one of the most pervasive problems in ML production. Minor differences in library versions, random seed values, hardware configurations, or environment variables can produce entirely different results (Semmelrock et al., 2025). Fewer than one third of ML projects are reproducible, and only approximately 5% of researchers share the source code of their experiments (JFrog ML, 2024).

JIRRS
Journal International Review of Research Studies

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

Without versioning of data, code, and environment, it is impossible to debug failures or audit decisions made by a model in production (Kästner, 2024).

Semmelrock et al. (2025), in a systematic literature review of reproducibility in ML, identify three distinct failure levels: computational reproducibility (same code, same environment, same result), empirical reproducibility (independent replication of findings), and inferential reproducibility (consistent conclusions from equivalent data). The literature reviewed for this article converges on computational reproducibility as the most tractable and the most directly relevant to MLOps practice — and also the one most consistently absent. Woźniak et al. (2025) note that reproducibility failures are particularly costly because they are silent: unlike a system crash, a non-reproducible model produces outputs that appear valid while being ungeneralizable, making the failure difficult to detect until significant downstream damage has occurred.

Beyond the versioning of code, data, and configuration parameters, reproducibility in ML systems requires that the computational environment itself be encapsulated in a portable and deterministic form. Containerization tools such as Docker enable practitioners to package code, framework versions, system dependencies, and GPU drivers into a single image that behaves identically across machines — directly addressing the "works on my machine" failure mode (Semmelrock et al., 2025; Boettiger, 2015). The absence of reproducibility also carries direct implications in regulated contexts: regulatory frameworks such as the EU AI Act require that organizations maintain comprehensive technical documentation capable of reconstructing which data, code, and environment configuration produced a given model output (European Commission, 2023; Kästner, 2024; BIS, 2024).

5.4 Absence of Robust Testing and Validation

(Cluster B — Engineering and Infrastructure Failures)

Evaluating a model solely on test set accuracy is insufficient. In production, models encounter noisy data, out-of-range values, and situations that never appeared during training — commonly referred to as edge cases (Paleyes et al., 2022). Data pipelines and feature transformations are rarely subjected to unit testing, meaning that silent errors can corrupt model inputs without detection (Recupito et al., 2025). A technically sound model can become problematic if it is not validated against adversarial scenarios.

Paleyes et al. (2022) and Recupito et al. (2025) converge on the observation that testing gaps in ML systems are qualitatively different from those in conventional

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

software: whereas software testing has well-established practices and tooling, ML testing requires validating not only functional correctness but also statistical properties of outputs — a requirement for which engineering culture and tooling remain underdeveloped in most organizations. Woźniak et al. (2025) note a partial divergence here: their review of practitioner literature suggests that larger organizations have begun developing internal testing frameworks (behavioral testing, slice-based evaluation, invariance testing) that have not yet been standardized or widely adopted, indicating an area of active but uneven progress.

5.5 Inadequate or Nonexistent Infrastructure

(Cluster B — Engineering and Infrastructure Failures)

Many organizations underestimate the cost and complexity of the infrastructure required to serve models in production. The absence of ML-specific staging environments, serving tools such as APIs and containers, and pipeline orchestration renders the deployment process fragile and manual (Kreuzberger et al., 2023). A notable illustration of this problem is Netflix, which at one point declined to deploy a competition-winning algorithm simply because its implementation complexity was deemed unviable (Monte Carlo Data, 2023). Infrastructure must be planned alongside the model, not as an afterthought.

Kreuzberger et al. (2023) identify infrastructure inadequacy as both a technical and a strategic failure: organizations frequently treat ML infrastructure as a cost center rather than as a capability investment, resulting in underprovisioned environments that cannot support the operational requirements of production models. Paleyes et al. (2022) add a temporal dimension: infrastructure failures are disproportionately common in organizations at early stages of ML adoption, where the gap between experimental tooling (notebooks, local scripts) and production requirements (containerized services, orchestrated pipelines) is largest. This suggests that infrastructure investment has a non-linear return — small improvements at early maturity levels produce disproportionate gains in deployment success rates.

5.6 Neglected Monitoring

(Cluster C — Organizational and Process Failures)

Unlike software bugs, which typically produce visible errors, ML models degrade silently. Performance declines gradually, without the system crashing or emitting alerts — a deterioration that may go undetected for weeks or months (Recupito et al., 2025). Ensuring the stability of an ML system in production requires continuous

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

monitoring of data, model metrics, and infrastructure health. Without configured alerts for performance degradation and a periodic retraining strategy, a deployed model rapidly becomes a liability (Symeonidis et al., 2022).

The literature converges strongly on this point. Recupito et al. (2025), Paleyes et al. (2022), and Symeonidis et al. (2022) all identify monitoring negligence as structurally recurrent, and all locate its root cause in the same pattern: deployment is treated as the terminal event of the ML lifecycle rather than as the beginning of an ongoing operational responsibility. Where the studies diverge is in their proposed remediation: Recupito et al. (2025) emphasize technical solutions (automated drift detection, alerting pipelines), while Paleyes et al. (2022) and Symeonidis et al. (2022) argue that technical monitoring tools are insufficient without organizational ownership structures that assign clear accountability for post-deployment model health. This tension — between technical and organizational remediation — echoes the broader sociotechnical framing of this article and suggests that monitoring failures are as much a governance problem as an engineering one.

5.7 Organizational and Cultural Barriers

(Cluster C — Organizational and Process Failures)

For five consecutive years, surveys have identified the primary obstacles to data initiative adoption within organizations as cultural rather than technical (DataScience-PM, 2024). Leadership that does not understand the actual lifecycle of an ML project tends to pressure teams for rapid results without investing in the necessary technical foundation. Another recurring problem is the absence of clear ownership: once a model moves to production, it is often unclear who is responsible for maintaining it (Operationalizing ML, 2024). Without this sense of accountability, monitoring fails and retraining never occurs.

Of the seven barriers identified in this article, organizational and cultural resistance shows the greatest divergence in the literature regarding its relative weight. Paleyes et al. (2022) treat it as a secondary amplifier — a factor that exacerbates technical failures rather than causing them independently. Recupito et al. (2025), by contrast, position cultural barriers as a primary and independent cause, noting that organizations with strong ML engineering cultures consistently outperform technically superior competitors that lack organizational alignment. Woźniak et al. (2025) offer a more granular view, identifying leadership understanding of ML as the single most influential organizational variable: teams with executive sponsors who have realistic expectations of ML timelines and failure rates show markedly higher deployment

Received: 01/02/2026 - Accepted: 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

success rates than those operating under unrealistic delivery pressure. This divergence in the literature suggests that the causal weight of cultural factors is itself context-dependent — varying with organizational size, sector, and ML maturity level — and warrants further empirical investigation.

A final critical observation is warranted regarding the seven-barrier framework itself. The convergence across Paleyes et al. (2022), Recupito et al. (2025), and Woźniak et al. (2025) is genuine, but it is convergence within a literature that shares certain structural biases: all three studies draw predominantly from technology sector organizations, from organizations that have attempted and documented ML deployment, and from English-language sources. Barriers that are more prevalent in non-technology sectors — such as regulatory constraints in healthcare, legacy system incompatibility in public administration, or data scarcity in manufacturing — are underrepresented in the evidence base. The seven barriers identified here should therefore be understood as the most consistently documented patterns in the available literature, not as an exhaustive or universally applicable taxonomy. Organizations in sectors or contexts not well-represented in the reviewed literature should treat this framework as a starting diagnostic rather than a definitive map of their deployment risk landscape.

6. The ML Model Lifecycle in Production

6.1 The Phases of Mature MLOps

The lifecycle of a model in production extends far beyond training. It begins with data collection and validation — the stage at which data adequacy for the problem is verified — and proceeds through experimentation, training, evaluation, and model approval prior to any deployment (Kreuzberger et al., 2023). Following deployment, the model must be served reliably, typically via APIs or containers, and monitored continuously. When performance degrades, the cycle restarts with retraining — which, in mature systems, is automated (Google Cloud, 2024). Treating each of these stages as part of a continuous flow, rather than as isolated tasks, is what distinguishes teams that operate ML with confidence from those that accumulate discarded models.

6.2 MLOps Maturity Levels — The Google MLOps Maturity Model

Google proposed one of the earliest and most widely cited MLOps maturity models, comprising three levels that describe the evolution of operationalization practices (Google Cloud, 2024).

JIRRS
Journal International Review of Research Studies

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

At Level 0, the entire process is manual and script-driven. Each stage — data analysis, training, validation — is executed individually, without automation. Data and engineering teams operate in silos, and there is no monitoring after deployment (Google Cloud, 2024).

At Level 1, the ML pipeline is automated. The model is no longer delivered in isolation — the entire training pipeline is moved to production, enabling continuous retraining as new data arrives. Data and model validation become integral components of the automated workflow (Google Cloud, 2024).

At Level 2, full CI/CD is applied to the ML pipeline. Any change in code, data, or model automatically triggers a new cycle of training, validation, and deployment. This level is required for systems in which both data and models change frequently (Kiroframe, 2025).

From the pragmatist perspective established in Section 3, the maturity model is best understood not as a prescriptive hierarchy to be climbed for its own sake, but as a diagnostic instrument for identifying which operational gaps most directly compromise a system's ability to deliver reliable value. An organization operating at Level 0 is not simply "less advanced" — it is structurally exposed to the failure modes documented in Section 5, particularly reproducibility failures and monitoring negligence, because it lacks the automated safeguards that make those failures detectable. The pragmatist implication is direct: the appropriate target maturity level for any given organization is not the highest achievable, but the lowest level at which the system can sustain reliable, auditable, and continuously improvable model performance given its operational context and resource constraints (Symeonidis et al., 2022).

6.3 The Role of the Feature Store

A feature store is a centralized repository of features — the input variables used by models — that can be shared across different teams and projects. Its primary benefit is ensuring consistency between the training and production environments (Feast, 2024). When features are computed differently across these two environments, a phenomenon known as training-serving skew emerges — a discrepancy that can severely and often silently degrade model performance in production (Google Cloud, 2024). The feature store addresses this problem by centralizing transformation logic, reusing it across both environments, and monitoring data quality over time (Feast, 2024).

Training-serving skew arises from a structural organizational pattern: data scientists implement transformation logic in notebooks or scripts, while engineering teams

JIRRS
Journal International Review of Research Studies

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

independently re-implement equivalent transformations for the production environment. Any divergence between the two implementations corrupts model inputs without triggering observable errors - producing precisely the kind of silent, gradual performance degradation that is most difficult to diagnose in production systems (Feast, 2024; Kreuzberger et al., 2023).

Uber was among the first organizations to document and resolve this problem at scale. The Michelangelo platform introduced a centralized feature store with synchronized offline and online storage layers: transformation logic is defined once and applied consistently across both training and serving environments, eliminating the need to maintain separate codebases for each context (Uber Engineering, 2017). Airbnb encountered an analogous challenge and addressed it through the Zipline platform (later renamed Chronon): prior to its adoption, ML practitioners at the company reportedly spent approximately 60% of their time on data collection and feature engineering tasks; Zipline automated training dataset generation with support for backfilling, sliding time windows, and online-offline consistency guarantees, reducing this effort from months to days (Airbnb Engineering, 2018).

It is nonetheless important to recognize that a feature store introduces non-trivial operational overhead, including the maintenance of synchronized offline and online pipelines, feature drift monitoring, and additional infrastructure costs. For small teams and batch inference systems without strict latency requirements, simpler approaches - such as versioned pipelines with centralized transformation logic — may be sufficient. The adoption of a feature store is most justifiable when multiple teams share the same feature definitions, when the model serves real-time predictions under latency constraints, or when training-serving skew has already been identified as a recurrent cause of production degradation (Kreuzberger et al., 2023; Feast, 2024).

6.4 Deployment Strategies: Canary, Blue/Green, and Shadow Mode

Deploying a new model version to production is not a binary event. MLOps deployment strategies include blue/green, canary, shadow, and A/B testing, each suited to different risk tolerances and business constraints (AWS Prescriptive Guidance, 2024).

The selection among deployment strategies should not be treated as a stylistic preference; it is a risk management decision governed by three primary variables: tolerance for user-facing impact, available infrastructure budget, and the degree to which real production traffic is necessary to validate model behavior. Shadow mode

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

is the appropriate starting point when any exposure of the new model to live users is unacceptable - production requests are duplicated to the candidate model, but only the incumbent system returns responses - at the cost of sustaining double the computational capacity for the duration of the shadow period (AWS Prescriptive Guidance, 2024; Systemoverflow, 2025). Blue/green deployment prioritizes rollback speed: two complete environments coexist, and any anomaly in the new environment allows traffic to revert to the incumbent within minutes; this strategy is particularly well-suited to critical updates where atomic cutover and zero downtime are non-negotiable, though it requires temporarily doubling the infrastructure footprint (Harness, 2025; Kreuzberger et al., 2023). Canary releases are the only strategy that collects authentic user feedback in a controlled and progressive manner; from an ML perspective, this approach is especially valuable because models may satisfy all offline evaluation criteria and yet fail in production due to distribution shift or training-serving skew - failure modes that materialize only under real traffic conditions (JFrog ML, 2024; Systemoverflow, 2025). For batch inference systems, the distinction between these strategies is largely inapplicable; parallel execution of incumbent and candidate models over the same input dataset, followed by systematic output comparison, constitutes a functionally equivalent and substantially simpler alternative. The infrastructure costs and operational complexity associated with the strategies above are therefore primarily justified for real-time serving systems where prediction failures carry immediate business or user impact (Kreuzberger et al., 2023).

In shadow mode, the new model runs in parallel with the existing system: a copy of user requests is sent to the updated model, but only the existing system returns responses to users - making it the safest option for validating behavior without any user impact (Towards Data Science, 2025). In blue/green deployments, two environments are kept running simultaneously: the blue represents the current model, while the green is the new version; rollback is straightforward because the old system remains active (AWS Prescriptive Guidance, 2024). In canary releases, the new model is initially exposed to a small fraction of traffic — typically starting at 1% — and progressively scaled to 20%, 50%, and eventually 100%, with the team analyzing predictions and user satisfaction at each step (JFrog ML, 2024).

Deploying ML models differs from traditional software deployments because models can fail silently, producing valid-looking outputs that are subtly wrong — making controlled rollout strategies essential for catching issues before they affect all users (OneUptime, 2026). Choosing the right strategy depends on the system's traffic volume, tolerance for risk, and availability of a comparison baseline (Kreuzberger et al., 2023).

JIRRS
Journal International Review of Research Studies

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

7. The MLOps Ecosystem: Tools and Technologies

The purpose of this section is not to provide a catalog of available tools, but to illustrate how different categories of tooling address the structural failure modes identified earlier in this article. Each tool category corresponds to a specific operational problem in ML systems — such as reproducibility, infrastructure fragility, or monitoring gaps — and should therefore be interpreted as part of a broader sociotechnical solution rather than as isolated technical products. Collectively, these infrastructural layers reflect a broader transformation in machine learning practice: successful deployment depends less on isolated algorithmic innovation than on the establishment of robust operational infrastructures capable of sustaining models over time.

7.1 Experiment Tracking

Experiment tracking forms the foundation of any reproducible ML process. MLflow is the most widely adopted tool in this category: open-source, it integrates seamlessly with the Python ecosystem and enables the logging of metrics, parameters, artifacts, and model versions with minimal effort (Hewage & Meedeniya, 2022). For teams requiring greater collaboration and advanced visualizations, Weights & Biases (W&B) is a well-established alternative, offering support for all major frameworks and interactive real-time dashboards (Neptune.ai, 2025). Neptune.ai, in turn, stands out for its flexibility in defining data structures and for features oriented toward larger teams that require governance over their experiments (Neptune.ai, 2025).

The choice among these tools is not arbitrary and maps directly to the reproducibility failures identified in Section 5.3. MLflow is the appropriate default for teams prioritizing integration with existing Python workflows and self-hosted infrastructure — its open-source nature eliminates vendor dependency, and its experiment tracking capabilities directly address the absence of versioning that renders most ML projects non-reproducible (JFrog ML, 2024). Weights & Biases becomes preferable when cross-team collaboration and real-time visualization are operational requirements, particularly in research-intensive environments where experiment volume is high and comparison across runs is frequent. Neptune.ai is most justifiable for larger teams requiring governance over experiment metadata — its structured data model supports audit trails that align with the regulatory documentation requirements discussed in Section 5.3. The critical analytical point is that no experiment tracking tool resolves reproducibility failures on its own: tooling adoption without the organizational

JIRRS
Journal International Review of Research Studies

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

discipline to log consistently and version data alongside code produces instrumented irreproducibility — the same failure mode, now with better dashboards (Semmelrock et al., 2025).

The relevance of experiment tracking tools therefore lies not in their individual features but in their capacity to operationalize reproducibility — the first failure mode identified earlier in this article.

7.2 Pipeline Orchestration

Orchestrating pipelines means automating the sequence of stages in the ML lifecycle — from data ingestion to deployment. Apache Airflow is the most mature option for enterprise environments, offering a broad range of native integrations and support for complex workflows with task dependencies, albeit at the cost of greater operational overhead (Hewage & Meedeniya, 2022). Kubeflow Pipelines is the natural choice for organizations already operating on Kubernetes, providing distributed training and end-to-end automation at scale (SG Analytics, 2025). Prefect represents a more modern alternative with a shallower learning curve than Airflow, focused on data and ML pipelines with a strong developer experience (DataCamp, 2024). ZenML differentiates itself by being infrastructure-agnostic, enabling the construction of portable pipelines that operate across any MLOps stack (Neptune.ai, 2025).

Pipeline orchestration tools respond directly to the infrastructure failures identified in Section 5.5: the absence of automated, reproducible pipelines is one of the primary reasons deployment processes remain fragile and manual. The selection among orchestration tools should be governed by three variables — existing infrastructure, team size, and pipeline complexity — rather than by feature comparisons in isolation. Apache Airflow is appropriate for organizations with dedicated platform engineering resources capable of managing its operational overhead; its maturity and integration breadth make it the defensible choice for complex enterprise workflows, but it is systematically over-adopted by teams that lack the engineering capacity to maintain it, resulting in pipelines that become liabilities rather than assets (Hewage & Meedeniya, 2022). Kubeflow Pipelines is only justifiable when Kubernetes is already the operational substrate — adopting Kubernetes solely to run Kubeflow introduces infrastructure complexity that outweighs the orchestration benefit for most organizations below a certain scale. Prefect and ZenML represent a more honest trade-off for lean teams: lower ceiling, lower floor, and substantially reduced operational burden. ZenML's infrastructure-agnostic design is particularly valuable for organizations at early MLOps maturity levels, where the target stack is not yet stable and portability reduces the cost of future architectural decisions (Neptune.ai, 2025).

JIRRS
Journal International Review of Research Studies

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

7.3 Data and Model Versioning

Versioning data and models is as important as versioning code. DVC (Data Version Control) functions as a "Git for data": it stores pointers to large files in remote repositories (S3, GCS, etc.) and versions metadata alongside code in Git, without bloating the repository (Iterative.ai, 2024). It also supports the definition of reproducible pipelines and the tracking of metrics across commit history. Delta Lake, by contrast, operates at the large-scale data storage layer, adding ACID transactions to data lakes and ensuring the quality and traceability of the data that feeds models (Kreuzberger et al., 2023). The specific tooling may vary across organizations, but the operational requirement it addresses — traceable data lineage — is universal across ML systems.

Data and model versioning tools address the same root cause as experiment tracking — the reproducibility failures of Section 5.3 — but at a different layer of the ML system. While experiment tracking tools version the outputs and parameters of training runs, DVC and Delta Lake version the inputs: the data that training runs consume. This distinction matters operationally because reproducibility failures in ML most commonly originate at the data layer, not the code layer — the same code applied to silently modified data produces different models, often without any alert (Semmelrock et al., 2025). DVC is the appropriate tool when data versioning needs to be integrated with existing Git workflows and the data resides in remote object storage; its lightweight pointer-based approach imposes minimal overhead and is accessible to teams without dedicated data engineering resources. Delta Lake operates at a different scale and is most justifiable when the organization already runs large-scale data processing on Spark or Databricks — its ACID transaction guarantees address data quality failures at the storage layer, making it relevant to the governance failures identified in Section 5.2 rather than solely to reproducibility. Adopting Delta Lake without the surrounding data engineering infrastructure it assumes introduces complexity without corresponding benefit (Kreuzberger et al., 2023).

7.4 Model Serving and Deployment

Deploying a model to production requires packaging, API exposure, and scalability. BentoML is the most accessible option: it enables any model to be packaged as a Docker microservice with minimal commands, making it well suited for smaller teams or early-stage projects (Neptune.ai, 2025). Seldon Core targets enterprise Kubernetes environments, with native support for canary deployments, A/B testing, and drift monitoring — though from 2024 onward it transitioned to a commercial

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

licensing model (Axel Mendoza, 2024). TorchServe is PyTorch's official solution for production model serving. The FastAPI + Docker combination is the most flexible and widely used approach among teams that prefer full control over their serving infrastructure (DevOpsSchool, 2025).

Model serving infrastructure is the technical layer most directly implicated in the infrastructure failures of Section 5.5 and the deployment strategy decisions of Section 6.4. The selection among serving tools should be governed primarily by the gap between the team's current engineering maturity and the operational requirements of the model — not by feature completeness. BentoML is the appropriate entry point for teams deploying their first production models: it reduces the operational distance between a trained model and a containerized API to a minimal set of commands, allowing teams to establish a production baseline without investing in serving infrastructure expertise prematurely. FastAPI combined with Docker remains the most flexible option for teams with sufficient engineering capacity to manage their own serving layer — it imposes no framework constraints and integrates cleanly with any deployment environment, making it the dominant choice in organizations where ML engineers have strong software engineering backgrounds (DevOpsSchool, 2025). Seldon Core's transition to commercial licensing from 2024 onward is a material consideration for cost-sensitive teams: its enterprise Kubernetes capabilities are genuine, but the total cost of ownership — including licensing, Kubernetes operational overhead, and engineering time — must be evaluated against the deployment volume and latency requirements that justify it. TorchServe is only relevant when the serving requirement is specific to PyTorch models and the team has no cross-framework serving needs; its narrow scope is a constraint, not a feature, for most production environments.

7.5 Production Monitoring

Detecting model degradation in production requires specialized tooling. Evidently AI is an open-source tool that generates visual reports on data drift, data quality, and model performance, and is widely used by teams beginning to structure their monitoring practices (Recupito et al., 2025). Arize and WhyLabs are managed platforms focused on ML observability at scale, offering automated alerting and real-time drift analysis (Neptune.ai, 2025). The Grafana + Prometheus stack forms the foundation of infrastructure monitoring — covering latency, CPU usage, and memory — and can be extended to business metrics, making it the most commonly used combination in environments that already operate with mature DevOps practices (Kreuzberger et al., 2023).

JIRRS
Journal International Review of Research Studies
Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

Production monitoring tools respond directly and specifically to the failure mode identified in Section 5.6: ML models degrade silently, and without instrumented monitoring, that degradation goes undetected until it produces visible business impact. The selection among monitoring tools maps to a maturity progression analogous to the one described in Section 6.2. Evidently AI is the appropriate starting point for teams establishing monitoring practices for the first time — its open-source, report-based approach makes drift and data quality issues visible without requiring the integration overhead of a managed platform, and its low barrier to adoption makes it the most defensible choice for organizations at MLOps Level 0 to Level 1 (Recupito et al., 2025). Arize and WhyLabs become justifiable when monitoring scale, automated alerting, and real-time observability are operational requirements — typically at Level 2 maturity, where the volume of models in production exceeds what manual report review can cover. The Grafana and Prometheus stack is the appropriate solution when the organization already operates mature DevOps infrastructure and the requirement is to extend existing observability practices to ML-specific metrics, rather than to introduce a parallel monitoring system. The critical analytical point — consistent with Section 5.6 — is that the tool selected matters less than the organizational commitment to act on what it reports: monitoring infrastructure without defined ownership and response protocols produces instrumented negligence rather than operational reliability (Symeonidis et al., 2022; Paleyes et al., 2022).

Monitoring frameworks therefore illustrate a broader operational requirement in production ML: models must be treated as dynamic systems subject to continuous environmental change rather than static artifacts trained once and deployed indefinitely.

7.6 All-in-One Platforms

For teams that prefer an integrated solution, cloud platforms cover the full model lifecycle. AWS SageMaker, Google Vertex AI, and Azure ML are the market leaders, providing end-to-end capabilities from data preparation to production monitoring with managed infrastructure and native integration with each provider's broader service ecosystem (SG Analytics, 2025). Databricks, the original creator of MLflow, combines data engineering, model training, and MLOps in a single environment particularly suited to large-volume data workflows (Azumo, 2026).

The primary analytical consideration for all-in-one platforms is the trade-off between operational simplicity and strategic dependency. These platforms reduce the infrastructure failures of Section 5.5 by abstracting away operational complexity — but they do so at the cost of vendor lock-in, which introduces a different category of

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

organizational risk: the inability to migrate workloads without substantial re-engineering investment. This trade-off is asymmetric across organization sizes. For teams without dedicated platform engineering resources, the managed infrastructure of a cloud platform is likely the most pragmatic choice — the operational overhead of maintaining a self-hosted MLOps stack commonly exceeds the engineering capacity available, producing the fragile manual deployments that Section 5.5 identifies as a primary cause of failure. For larger organizations with stable engineering teams, a composable open-source stack offers greater long-term flexibility, and the risk of lock-in justifies the investment in infrastructure ownership (Azumo, 2026; Kreuzberger et al., 2023). The selection is therefore not a technical decision but a strategic one, governed by the organization's current maturity level, engineering capacity, and tolerance for infrastructure dependency.

7.7 Data Privacy and Security in ML Pipelines

The growth of regulatory frameworks — already discussed in Section 9.3 — makes data privacy a technical requirement, not merely a compliance concern. Whether data is encrypted at rest, whether PII is masked before entering a pipeline, and whether lineage tracking is robust enough to support regulatory requirements are foundational security decisions made at the data engineering stage — lapses here cascade into downstream leakage risks that no amount of model hardening can fix (Protegrity, 2025).

The primary techniques for privacy-preserving ML are differential privacy, federated learning, and homomorphic encryption. Differential privacy introduces controlled noise to data or outputs to prevent identification of individuals; federated learning trains models locally without centralizing raw data; homomorphic encryption allows computations on encrypted data without decrypting it (Alation, 2025). In terms of tooling, Microsoft Presidio is widely used for PII detection and anonymization in text preprocessing pipelines, particularly in healthcare and finance (MDPI, 2024), while TensorFlow Privacy and PySyft support differential privacy and federated learning workflows respectively (Alation, 2025).

Security threats specific to MLOps include data poisoning, adversarial attacks, model and IP theft, attacks on pipeline infrastructure, and data leaks — all requiring protection strategies at every stage of the model lifecycle (DS Stream, 2024). Treating privacy as a pipeline property, rather than a post-deployment concern, aligns directly with both the EU AI Act and GDPR mandates already cited in this article (European Commission, 2023).

JIRRS
Journal International Review of Research Studies

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

8. Building an MLOps Culture in Your Organization

8.1 Starting Small: The MLOps MVP

It is neither necessary nor advisable to adopt the entire MLOps ecosystem at once. The most effective starting point is to focus on three foundational practices: code and data versioning, experiment tracking, and a minimal automated pipeline covering the path from training to deployment (Symeonidis et al., 2022). Organizations that attempt to implement everything simultaneously tend to stall under operational complexity before achieving any meaningful results. An incremental approach allows teams to build confidence with the tooling and adapt processes to the organization's reality before progressing to higher levels of automation (Recupito et al., 2025).

This incremental logic is a direct expression of the pragmatist framework underpinning this article. Rather than optimizing for a theoretically complete MLOps architecture, the pragmatist approach prescribes optimizing for the minimum operational configuration that produces a measurable improvement in deployment reliability — and iterating from there. The question is never "what is the ideal MLOps stack?" but "what is the next practice that will most concretely reduce the probability of the failure modes identified in Section 5 for this organization, at this stage?" This reframing has a practical consequence: it shifts the evaluation of MLOps adoption from a compliance exercise — checking whether all components are present — to an outcomes-based assessment of whether operational reliability is measurably improving over time (Lim, 2023; Kreuzberger et al., 2023).

8.2 Defining Clear Roles

The absence of well-defined roles is one of the leading sources of confusion in ML projects. The Data Scientist is responsible for translating the business problem into an ML problem and for building and evaluating models. The ML Engineer handles operationalization — integrating the model into the system and ensuring scalability and performance. The MLOps Engineer, in turn, builds and maintains the platform that enables the work of both: automated pipelines, deployment infrastructure, and monitoring (Kreuzberger et al., 2023). In practice, particularly in smaller teams, these roles overlap — but having clarity about who trains, who deploys, and who monitors prevents any of these responsibilities from going unowned (Symeonidis et al., 2022).

8.3 Establishing Inter-Team Contracts

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

Treating the model as a product requires teams to establish formal agreements about what it means to be "production-ready." This includes defining model SLAs — maximum acceptable latency, minimum availability, and accuracy thresholds below which the model must be retrained or taken out of operation. Without these criteria, deployment decisions become subjective and post-deployment monitoring loses its reference point (Symeonidis et al., 2022). Just as software development adopted the Definition of Done as an agile practice, ML teams require an equivalent that formalizes the technical, quality, and business requirements a model must satisfy before going to production (Recupito et al., 2025).

The establishment of model SLAs and production-readiness criteria is, in pragmatist terms, the operationalization of the success criteria introduced in Section 3. By defining in advance what "working in practice" means for a specific model in a specific context — including acceptable latency, minimum availability, and performance thresholds triggering retraining — teams translate the abstract epistemological commitment to practical utility into concrete, auditable engineering contracts. This is precisely the mechanism through which pragmatism moves from philosophical positioning to organizational practice: not through declarations of intent, but through the institutionalization of operationally grounded evaluation criteria (Symeonidis et al., 2022; Pretorius, 2024).

8.4 Documentation as Culture, Not Bureaucracy

Documenting models is not a bureaucratic exercise — it is a governance practice that protects both the organization and its users. Model cards, proposed by Mitchell et al. (2019), are concise documents that accompany each model and describe its intended use, the conditions under which it was evaluated, its limitations, and the contexts for which it is unsuitable (Mitchell et al., 2019). Complementarily, the Model Registry — a centralized catalog of all models in use — enables version tracking, association of performance metrics, and control over which models are active in production (Kreuzberger et al., 2023). Together, model cards and the Model Registry transform documentation from a formality into an operational asset.

8.5 Infrastructure Cost Management: FinOps Applied to ML

Infrastructure cost is an organizational barrier that disproportionately affects smaller teams, yet it receives little attention in MLOps literature. Companies heavily invested in AI/ML are beginning to see the impact on their cloud bills - the early days of AI/ML in the cloud mirror the initial cloud adoption phase, where uncontrolled

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

experimentation led to unexpected cost spikes, forcing a shift toward cost management (CloudKeeper, 2024).

FinOps - the practice of maximizing business value from cloud investments - applies directly to ML workloads. AI/ML workloads present specific challenges: GPU instance optimization, specialized data ingestion requirements, and a faster and broader cost impact across cross-functional teams than traditional software workloads (FinOps Foundation, 2024). Key practices include tagging experiments and training runs for cost attribution, using spot or preemptible instances for non-critical training jobs, and setting budget alerts per team or project (FinOps Foundation, 2024).

For FinOps practitioners, AI/ML costs surface in two ways: using AI for FinOps itself, and managing the cost of running AI/ML systems - and the cost of running these workloads is expected to have a growing impact on FinOps practices (FinOps Foundation, 2024). For lean teams, the minimum viable approach is experiment-level cost tagging combined with automated alerts for anomalous GPU or storage consumption - sufficient to prevent the runaway spend that makes cloud infrastructure prohibitive at early stages of MLOps adoption (Symeonidis et al., 2022).

The decision between spot, on-demand, and reserved instances is one of the most consequential cost optimizations available to ML teams operating on cloud infrastructure. Spot instances - which access unused cloud capacity at discounts of 60–90% relative to on-demand rates - are well-suited for training workloads that are fault-tolerant and can resume from checkpoints after interruption; they are, however, inappropriate for production inference services, regulated workloads where infrastructure uncertainty creates compliance risk, or training runs that cannot tolerate the 2–30 minute preemption notices typical of major providers (FinOps Foundation, 2024; Hyperbolic, 2025). Reserved instances, which require 1–3 year capacity commitments, offer savings of 20–72% over on-demand rates and are most justifiable for predictable, steady-state workloads such as continuous retraining pipelines or always-on serving infrastructure (DigitalOcean, 2025). On-demand pricing - which can run 2–3 times higher than reserved rates - remains the appropriate choice for development, experimentation, and variable workloads where capacity needs cannot be forecast with confidence. In practice, cost-efficient ML infrastructure typically follows a hybrid model: spot instances for scheduled batch training, on-demand for experimentation, and reserved capacity for production inference endpoints - a pattern that can reduce total GPU spend by 40–70% relative to on-demand pricing alone (FinOps Foundation, 2024).

JIRRS
Journal International Review of Research Studies

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

9. Real-World Cases: Lessons From Failure and Success

9.1 Failure Case — Legacy System Integration

A recurring pattern of failure in ML projects is the development of models in complete isolation from the infrastructure in which they will need to operate. Research mapping real deployment cases has identified incompatibility with legacy systems as one of the leading causes of project abandonment: the model is built using modern technologies, but the organization's production environment is outdated, incompatible, and incapable of processing data in the expected format (Paleyes et al., 2022). Months of work are lost because no one asked the most basic question at the outset: how will this model connect to the existing system? The lesson is straightforward — integration requirements must be established before the first line of model code is written (Recupito et al., 2025).

9.2 Success Case — Spotify

Spotify is one of the most frequently cited examples of MLOps at scale. The company operates with 50 ML teams using a centralized internal platform capable of training 30,000 models and processing 300,000 predictions per second, with a reported 700% productivity increase between 2020 and 2021 (Spotify Engineering, 2021). The key was not algorithmic sophistication, but standardization: the company created a "golden path" — a well-supported standard track that any ML team must follow to build and deploy models to production (Spotify Engineering, 2021). Organizations that prioritize end-to-end model lifecycle management and the automation of deployment, monitoring, and continuous updates achieve consistent and scalable results (ManageEngine, 2025).

9.3 Lesson From the Financial Sector — Regulation as a Catalyst for Good Practices

The financial sector was compelled to mature its MLOps practices under regulatory pressure. Regulators began requiring that decisions made by ML models — such as credit approval, fraud detection, and risk management — be explainable in terms comprehensible to auditors, clients, and supervisory bodies (BIS, 2024). This drove the adoption of explainable AI (XAI), rigorous model documentation, and auditable records of versions, training runs, and decisions (Rane et al., 2023). Regulation thus functioned as an inadvertent catalyst for sound MLOps practices: by mandating traceability and explainability, it compelled institutions to establish the governance, versioning, and monitoring frameworks that should have existed for technical reasons from the outset (European Commission, 2023).

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

10. The Future of MLOps

10.1 LLMOps: The Evolution Toward Language Models

Large language models (LLMs) have introduced challenges that traditional MLOps was not designed to address. MLOps methodologies, developed for smaller models and structured data, do not adapt well to LLMs, which demand specialized pipelines for prompt management, Retrieval-Augmented Generation (RAG), and the evaluation of open-ended outputs (Diaz-de-Arcaya et al., 2025). Unlike traditional models, LLMs process input as discrete token sequences whose computational cost scales directly with context length — a property with significant implications for inference latency, memory allocation, and serving infrastructure design (Salomão, 2026). Caching mechanisms such as KV-cache, for instance, can substantially reduce redundant computation in production environments, but require deliberate pipeline planning to be effective (Salomão, 2026). Prompt versioning, for instance, is a practice with no equivalent in classical MLOps: a subtle change in input text can dramatically alter model behavior, making version tracking essential for reproducibility and debugging (Edge AI and Vision Alliance, 2025). Evaluating LLMs is also fundamentally different: traditional metrics such as accuracy are insufficient for open-ended textual outputs, requiring combinations of automated metrics, human evaluation, and techniques such as Reinforcement Learning from Human Feedback (RLHF) (Databricks, 2024).

The lifecycle of an LLM-based application differs fundamentally from that of a classical ML model. In traditional MLOps, the central iteration loop revolves around data collection, model training, and retraining - a cycle that is typically slow and computationally expensive. In LLMOps, most organizations do not train foundation models from scratch; instead, they begin with a pre-trained model and iterate primarily on prompts, retrieval configurations, and guardrails - a cycle that is substantially faster but introduces entirely different operational artifacts (Diaz-de-Arcaya et al., 2025; ZenML, 2025). The cost structure is also inverted: whereas classical ML concentrates costs in the training phase, LLMs shift the financial burden to inference, where a single poorly designed prompt can multiply token consumption and API costs overnight (Databricks, 2024).

Retrieval-Augmented Generation (RAG) pipelines introduce a distinct operational layer that has no direct equivalent in classical MLOps. A RAG system retrieves relevant document chunks from a vector database and injects them into the model's context window at inference time, allowing the model to ground its responses in up-

JIRRS
Journal International Review of Research Studies

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

to-date or domain-specific knowledge without full retraining (Diaz-de-Arcaya et al., 2025). Operationalizing RAG requires decisions that classical MLOps does not address: chunking strategy (the size and overlap of text segments indexed in the vector store), embedding model selection, retrieval quality thresholds, and the management of knowledge base updates as source documents change. Evaluating these pipelines requires specialized frameworks; RAGAS (Retrieval Augmented Generation Assessment) provides reference-free metrics - including faithfulness, answer relevancy, context precision, and context recall - that assess both the retrieval and generation components independently, without relying on manually annotated ground-truth labels (Es et al., 2024).

Prompt management constitutes a versioning challenge with no analog in classical MLOps. A prompt functions as executable code: a subtle change in phrasing, instruction ordering, or few-shot examples can substantially alter model behavior across the entire user base (Edge AI and Vision Alliance, 2025). This makes prompt versioning, regression testing across prompt versions, and rollback capabilities essential operational requirements - addressed by platforms such as LangSmith, which integrates prompt versioning with production tracing and evaluation workflows (Databricks, 2024). Together, these requirements - RAG pipeline management, open-ended output evaluation, and prompt versioning - define LLMOps as a discipline that extends MLOps rather than replacing it, demanding new tooling and processes alongside the foundations already described in this article.

10.2 AutoMLOps and End-to-End Automation

The next frontier of MLOps is the full automation of the decision cycle: systems that monitor their own performance and autonomously decide when to retrain and deploy a new model version (Kreuzberger et al., 2023). This concept, referred to as AutoMLOps, eliminates dependence on human intervention for routine operational tasks, freeing teams for higher-value work. Although still in early stages of adoption, this trend aligns with the trajectory observed in DevOps, where CI/CD pipelines gradually reduced the need for manual deployments (Recupito et al., 2025).

10.3 MLOps in Smaller Organizations and Lean Teams

MLOps is not the exclusive domain of large technology companies. The growth of the open-source tool ecosystem has made best practices accessible to organizations of any size (Hewage & Meedeniya, 2022). Simple combinations such as MLflow for experiment tracking, DVC for data versioning, and GitHub Actions for pipeline

JIRRS
Journal International Review of Research Studies

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

automation already provide the foundation of a functional MLOps practice at no licensing cost. For lean teams, the key is to begin with the minimum necessary — versioning, tracking, and a reproducible pipeline — and evolve incrementally as team maturity develops (Symeonidis et al., 2022).

More importantly, these minimal tool combinations illustrate a broader structural principle: early-stage MLOps maturity depends less on sophisticated platforms than on the institutionalization of reproducibility practices. The tools themselves are interchangeable; what matters operationally is the establishment of version control, experiment traceability, and automated execution as routine engineering practices.

11. Study Limitations

The findings and syntheses presented in this article are subject to a set of limitations that must be made explicit for the reader to interpret and apply them appropriately.

11.1 Reliance on Industry Documentation and Self-Promotion Bias

A substantial portion of the empirical grounding in this article derives from technical documentation, engineering blog posts, and reports produced by industry organizations — including Google, AWS, Spotify, Uber, and Airbnb, among others. While these sources were selected according to the criteria described in Section 2.4, the inherent limitation of industry-produced documentation must be acknowledged: organizations publish case studies and technical reports selectively, with a natural tendency to document successes and to present their platforms and practices in favorable terms. Failures, abandoned projects, and negative results are systematically underrepresented in this type of literature — a pattern documented in the broader grey literature methodology research (Recupito et al., 2025). The practical consequence is that the benefits of MLOps practices described in this article — including the productivity and ROI figures cited in Section 12 — may reflect optimistic estimates drawn from organizations with the resources and motivation to implement and publicize successful outcomes, rather than representative averages across the full population of organizations attempting ML deployment.

11.2 Absence of Meta-Analytic Quantitative Synthesis

This article does not apply meta-analytic aggregation or quantitative effect-size synthesis. The barriers identified in Section 5 and the practices recommended in Sections 6, 7, and 8 are supported by convergent qualitative and descriptive evidence

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

across multiple studies, but the magnitude of their individual effects — for instance, the degree to which implementing experiment tracking reduces deployment failure rates, or the quantitative relationship between MLOps maturity level and ROI — cannot be precisely estimated from the evidence base available. The MLOps literature currently lacks the volume of controlled empirical studies required for robust meta-analysis, a limitation acknowledged explicitly by Woźniak et al. (2025) and Recupito et al. (2025). Readers seeking quantitative benchmarks for specific practices should treat the figures cited in this article as indicative rather than definitive, and should consult primary sources for methodological details.

11.3 Sector-Specific Nature of Case Studies and Risk of Overgeneralization

The real-world cases discussed in Section 9 — Spotify's large-scale ML platform and the financial sector's regulatory-driven MLOps adoption — were selected for their illustrative value and the quality of available documentation. However, both cases represent organizations operating at the high end of the ML maturity spectrum: Spotify is a technology company with dedicated ML infrastructure teams and decades of data engineering expertise; the financial institutions discussed operate under specific regulatory constraints that create unusually strong institutional incentives for MLOps governance. The practices and outcomes documented in these cases may not generalize to smaller organizations, non-technology sectors, or resource-constrained environments. Similarly, the deployment failure patterns documented by Paleyes et al. (2022) draw predominantly from technology and research-adjacent organizations; the prevalence and relative weight of specific barriers may differ substantially in manufacturing, healthcare, public administration, or other sectors where ML adoption patterns and organizational structures diverge from the technology industry norm.

11.4 Absence of Primary Empirical Validation

This article does not generate primary data. The synthesis presented here reflects the current state of published and documented knowledge, but does not include original surveys, interviews, controlled experiments, or observational studies of ML teams in production environments. As a consequence, the causal relationships implied throughout the article — for instance, that organizational silos cause deployment failures, or that MLOps adoption produces measurable ROI improvements — are supported by convergent observational evidence rather than by controlled empirical demonstration. Future research employing primary data collection methods — including longitudinal studies of MLOps adoption, controlled comparisons of deployment outcomes across maturity levels, or systematic

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

interviews with ML engineering practitioners — would substantially strengthen the evidentiary basis for the conclusions presented here.

12. Conclusion

Throughout this article, it has become clear that ML models fail to reach production due to a combination of technical and cultural factors. On the technical side, the most recurrent problems are lack of reproducibility, training data disconnected from the production environment, absence of robust testing, and inadequate infrastructure. On the cultural side, the prevailing issues are organizational silos, poorly defined roles, lack of ownership, and leadership that demands results without investing in the foundation required to achieve them (Paleyes et al., 2022; Recupito et al., 2025).

MLOps is not an operational cost — it is what transforms experiments into real business value. Organizations that adopt structured MLOps practices report an average ROI of 28%, with cases reaching 149%, alongside significant reductions in deployment time and team operational burden (Kreuzberger et al., 2023). The global MLOps market, valued at USD 2.33 billion in 2025, is projected to grow at an annual rate of 28.9% through 2034 — a signal that the industry increasingly recognizes this relationship (Fortune Business Insights, 2025).

Where to begin? The answer is straightforward: with what already exists. Versioning code with Git, tracking experiments with MLflow, and automating a minimal training and validation pipeline is sufficient to move beyond level zero. It is not necessary to adopt the entire MLOps stack at once — it is necessary to start and evolve consistently (Symeonidis et al., 2022).

The closing reflection is direct: the most valuable model is not the most accurate — it is the one that is running. A mediocre model in production, monitored and updated, delivers infinitely more value than an excellent model confined to a notebook. Technical excellence without operationalization is merely research. MLOps is what transforms research into product (Sculley et al., 2015).

References

Airbnb Engineering. (2018). Zipline: Airbnb's declarative feature engineering framework [Conference presentation]. Strata Data Conference.

JIRRS
Journal International Review of Research Studies

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

https://conferences.oreilly.com/strata/strata-ny-2018/public/schedule/detail/68114.html

Amazon Web Services. (2024). Deployment strategies for MLOps. AWS Prescriptive Guidance. https://docs.aws.amazon.com/prescriptive-guidance/latest/ml-operations-planning/deployment.html

Amazon Web Services. (2024). Introduction to blue/green deployments on AWS. AWS Whitepapers. https://docs.aws.amazon.com/whitepapers/latest/blue-green-deployments/introduction.html

Axel Mendoza. (2024). The ultimate guide to ML model deployment in 2024. axelmendoza.com. https://www.axelmendoza.com/posts/ml-model-deployment/

Azumo. (2026, January). Top 10 MLOps platforms for scalable AI. Azumo. https://azumo.com/artificial-intelligence/ai-insights/mlops-platforms

BIS — Bank for International Settlements. (2024). Managing explanations: How regulators can address AI explainability (FSI Papers No. 24). https://www.bis.org/fsi/fsipapers24.pdf

Boettiger, C. (2015). An introduction to Docker for reproducible research, with examples from the R environment. ACM SIGOPS Operating Systems Review, 49(1), 71–79. https://doi.org/10.1145/2723872.2723882

CloudKeeper. (2024). 2024 State of FinOps report: Key trends in cloud FinOps. CloudKeeper Insights. https://www.cloudkeeper.com/insights/blog/2024-state-finops-report-key-trends-cloud-finops

DataCamp. (2024). Top 25 MLOps tools you need to know in 2025. DataCamp Blog. https://www.datacamp.com/blog/top-mlops-tools

DataScience-PM. (2024). State of data science and machine learning survey. https://datascience-pm.com

Databricks. (2024). What is LLMOps? Databricks Glossary. https://www.databricks.com/glossary/llmops

DeCApria, D. (2024, November). Introduction to MLOps: Bridging machine learning and operations. Carnegie Mellon University, Software Engineering Institute's Insights. https://doi.org/10.58012/zkgg-en08

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

DevOpsSchool. (2025). Top 10 AI model serving frameworks tools in 2025: Features, pros, cons, comparison. DevOpsSchool. https://www.devopsschool.com/blog/top-10-ai-model-serving-frameworks-tools-in-2025-features-pros-cons-comparison/

DS Stream. (2024). MLOps security: Threats, risks, and best practices. DS Stream Blog. https://dsstream.com/mlops-security-threats-risks-and-best-practices/

Diaz-de-Arcaya, J., et al. (2025). Transitioning from MLOps to LLMOps: Navigating the unique challenges of large language models. Information, 16(2), Article 87. https://www.mdpi.com/2078-2489/16/2/87

DigitalOcean. (2025). 7 best cloud GPU platforms for AI, ML, and HPC in 2025. https://www.digitalocean.com/resources/articles/best-cloud-gpu-platforms

Edge AI and Vision Alliance. (2025, March). LLMOps unpacked: The operational complexities of LLMs. Edge AI and Vision Alliance. https://www.edge-ai-vision.com/2025/03/llmops-unpacked-the-operational-complexities-of-llms/

Es, S., James, J., Espinosa-Anke, L., & Schockaert, S. (2024). RAGAS: Automated evaluation of retrieval augmented generation. In Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations (pp. 150–158). Association for Computational Linguistics. https://aclanthology.org/2024.eacl-demo.16/

European Commission. (2023). EU AI Act: Regulation on artificial intelligence. https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai

Feast. (2024). What is a feature store? Feast — Open Source Feature Store. https://feast.dev/blog/what-is-a-feature-store/

FinOps Foundation. (2024). FinOps for AI: Overview. FinOps Foundation Working Group. https://www.finops.org/wg/finops-for-ai-overview/

FinOps Foundation. (2024). State of FinOps 2024. FinOps Foundation. https://data.finops.org/

Fortune Business Insights. (2025). MLOps market size, share & forecast. https://www.fortunebusinessinsights.com/mlops-market-108986

Google Cloud. (2024). MLOps: Continuous delivery and automation pipelines in machine learning. Google Cloud Architecture Center.

JIRRS
Journal International Review of Research Studies

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning

Harness. (2025). Blue-green deployments, A/B testing, and canary releases. https://www.harness.io/blog/blue-green-canary-deployment-strategies

Hewage, N., & Meedeniya, D. (2022). Machine learning operations: A survey on MLOps tool support. arXiv. https://arxiv.org/abs/2202.10169

Hyperbolic. (2025). GPU cloud pricing: 2025 guide to costs, models & optimization. https://www.hyperbolic.ai/blog/gpu-cloud-pricing

Hinder, F., Vaquet, V., & Hammer, B. (2024). Suitability of common performance metrics for drift detection in dynamic environments. In Proceedings of the International Joint Conference on Neural Networks (IJCNN). IEEE. https://doi.org/10.1109/IJCNN60899.2024.10651234

Iterative.ai. (2024). DVC: Open-source version control system for machine learning projects. DVC. https://dvc.org

JFrog ML. (2024). The state of ML model reproducibility 2024: Challenges and best practices. JFrog. https://jfrog.com/ml/resources/reports/model-reproducibility-2024/

Kästner, C. (2024). Machine learning in production: From models to systems. Carnegie Mellon University, School of Computer Science. https://mlip-cmu.github.io/book/

Kiroframe. (2025, October). MLOps maturity levels: The most well-known models. Kiroframe. https://kiroframe.com/mlops-maturity-levels-the-most-well-known-models/

Kreuzberger, D., Kühl, N., & Hirschl, S. (2023). Machine learning operations (MLOps): Overview, definition, and architecture. IEEE Access, 11, 31866–31879. https://doi.org/10.1109/ACCESS.2023.3262138

Lim, W. M. (2023). Philosophy of science and research paradigm for business research in the transformative age of automation, digitalization, hyperconnectivity, obligations, globalization and sustainability. Journal of Trade Science, 11(2–3), 3–30. https://doi.org/10.1108/JTS-07-2023-0015

ManageEngine. (2025). Top MLOps use cases: Real-world examples, enterprise strategies, and best practices. ManageEngine IT Operations.

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

https://www.manageengine.com/it-operations-management/cxo-focus/insights/mlops-case-studies-and-best-practices.html

Microsoft. (2024). Presidio: Open-source data protection and de-identification framework. Microsoft GitHub. https://microsoft.github.io/presidio/

Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., Spitzer, E., Raji, I. D., & Gebru, T. (2019). Model cards for model reporting. In Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*) (pp. 220–229). ACM. https://dl.acm.org/doi/10.1145/3287560.3287596

Monte Carlo Data. (2023). The state of data quality for AI 2023. Monte Carlo Data. https://www.montecarlodata.com/resource/state-of-data-quality-for-ai-2023/

Neptune.ai. (2023). Model deployment strategies. Neptune.ai Blog. https://neptune.ai/blog/model-deployment-strategies

Neptune.ai. (2025). MLOps tools and platforms landscape 2025. Neptune.ai. https://neptune.ai/blog/mlops-tools-platforms-landscape

Operationalizing ML. (2024). MLOps roles and responsibilities: Who owns what in production? Operationalizing ML. https://operationalizingml.com/research/mlops-roles-2024

Paleyes, A., Urma, R.-G., & Lawrence, N. D. (2022). Challenges in deploying machine learning: A survey of case studies. ACM Computing Surveys, 55(6), Article 114. https://dl.acm.org/doi/10.1145/3533378

Pereira Costa, G. (2026). The Role of Game Engines in the Democratization of Digital Game Development. Journal International Review of Research Studies, 1(02), 1-17. https://doi.org/10.66104/8yn0e657

Pretorius, L. (2024). Demystifying research paradigms: Navigating ontology, epistemology, and axiology in research. The Qualitative Report, 29(10), 2698–2715. https://doi.org/10.46743/2160-3715/2024.7632

Rane, N., Choudhary, S., & Rane, J. (2023). Explainable artificial intelligence (XAI) approaches for transparency and accountability in financial decision-making. SSRN. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4640316

Recupito, G., et al. (2025). A multivocal review of MLOps practices, challenges and open issues. ACM Computing Surveys. https://dl.acm.org/doi/10.1145/3747346

JIRRS
Journal International Review of Research Studies

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

Sahiner, B., Chen, W., & Samei, E. (2023). Data drift in medical imaging AI: Detection, quantification, and mitigation. Radiology: Artificial Intelligence, 5(4), e230045. https://doi.org/10.1148/ryai.230045

Salomão, P. E. A. (2026). Tokens as Computational Units in Data Science and Machine Learning: Mathematical Foundations, Transformer Architecture, Inference Economy, and Caching Systems in Foundational Models. Journal International Review of Research Studies, 1(02), 1-27. https://doi.org/10.66104/kxf7hk05

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F., & Dennison, D. (2015). Hidden technical debt in machine learning systems. Advances in Neural Information Processing Systems, 28, 2503–2511. https://papers.neurips.cc/paper/5656-hidden-technical-debt-in-machine-learning-systems.pdf

Semmelrock, H., Kopp, M., & Krcmar, H. (2025). Reproducibility in machine learning: A systematic literature review and framework for assessment. ACM Computing Surveys. https://doi.org/10.1145/3677178

SG Analytics. (2025, November). Top 20 MLOps tools in 2026. SG Analytics. https://www.sganalytics.com/blog/mlops-tools/

Spotify Engineering. (2021, February). TWIMLCon: How Spotify does ML at scale [LinkedIn article by Skylar Payne]. LinkedIn. https://www.linkedin.com/pulse/twimlcon-take-aways-how-spotify-does-ml-scale-skylar-payne

Symeonidis, G., et al. (2022). MLOps — definitions, tools and challenges. In Proceedings of the IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC) (pp. 453–460). IEEE. https://ieeexplore.ieee.org/document/9756822

Systemoverflow. (2025). Trade-offs: Canary vs. blue-green vs. shadow deployment. https://www.systemoverflow.com/learn/ml-ab-testing/ramp-up-strategies/trade-offs-canary-vs-blue-green-vs-shadow-deployment

Towards Data Science. (2024). ML model deployment strategies. Towards Data Science. https://towardsdatascience.com/ml-model-deployment-strategies-72044b3c1410/

JIRRS
Journal International Review of Research Studies

Received: 01/02/2026 - **Accepted:** 05/03/2026
Vol: 01.02
DOI: 10.66104/wbexse86
Pages: 1-38

Uber Engineering. (2017). Meet Michelangelo: Uber's machine learning platform. Uber Blog. https://www.uber.com/blog/michelangelo-machine-learning-platform/

Woźniak, A. P., et al. (2025). MLOps components, tools, process and metrics: A systematic literature review. IEEE Access. https://doi.org/10.1109/ACCESS.2025.3534990

ZenML. (2025). MLOps vs LLMOps: What's the difference? ZenML Blog. https://www.zenml.io/blog/mlops-vs-llmops

Zi, W. (2024, November). Why most machine learning projects fail to reach production. InfoQ. https://www.infoq.com/articles/why-ml-projects-fail-production/