Vol. 8, No. 4, August 2026

ISI Journal of Information Systems and Informatics

Published By Asosiasi Doktor Sistem Informasi Indonesia

Machine Learning for Software Deployment in the Public Sector: A Systematic Review and Research Agenda for African Contexts

Johnson Nuviadenu[1], Themba Masombuka[2], Ernest Mnkandla[2], Malusi Sibiya[3]

[1,2,3,4]Centre for Augmented Intelligence and Data Science (CAIDS), Department of Computer Science, University of South Africa, Roodepoort, South Africa

Received:
February 19, 2026
Revised:
July 25, 2026
Accepted:
August 5, 2026
Published:
August 30, 2026

Corresponding Author:
Author Name*:
Johnson Nuviadenu
Email*:
51795868@mylife.unisa.ac.za

DOI:
10.63158/journalisi.v8i4.1839

Abstract. Failures in deploying digital public services can disrupt essential systems and affect millions of citizens. Machine learning (ML)-based software deployment decision support, including build risk prediction, release gating, autoscaling, rollback assistance, and post-deployment anomaly detection, offers opportunities for safer and more reliable releases. However, the extent of existing evidence in public-sector environments, particularly within African institutions, remains unclear. Following the PRISMA 2020 guidelines, this systematic literature review searched five databases using predefined inclusion criteria and a six-item quality assessment. A total of 33 peer-reviewed studies published between 2018 and 2025 were included, while studies focusing exclusively on MLOps were excluded. The findings reveal that none of the reviewed studies (0/33) explicitly evaluated ML deployment decision support in public-sector contexts or African institutions; existing evidence originates primarily from private-sector or unspecified environments. Research efforts are concentrated on autoscaling (12/33, 36%) and build prediction (9/33, 27%), with tree-based models being the dominant approach (16/33, 48%). Furthermore, only one study (3%) reported statistical significance testing or confidence intervals. This review identifies a research and evidence gap rather than confirming the absence of practical adoption. It proposes a staged research agenda toward explainable, lightweight, and context-aware ML deployment support for African public institutions.

Keywords: Machine Learning Deployment Support; Software Deployment Decision Support; Systematic Literature Review; Public Sector Digital Services; African Digital Transformation

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

1. INTRODUCTION

Software deployment remains one of the riskiest activities in software engineering. A single faulty release can interrupt digital services, corrupt data and expose security weaknesses [1], [2]. Release frequency keeps rising. As it does, every deployment decision carries greater consequence for service reliability, safety and public trust. Traditional deployment decision-making still depends heavily on manual judgement and rule-based checks [3]-[5]. These methods are slow. They are also error-prone and difficult to scale in continuous delivery environments, and this weakness has motivated the adoption of machine-learned assistance for build prediction, anomaly detection, autoscaling and rollback choices [6]-[12].

The setting in which such assistance operates matters greatly, and the public sector is not simply a slower version of the private sector. Public institutions procure software through rigid, multi-stage tendering cycles that can outlast a model's useful life; they run legacy systems that expose little of the rich telemetry which commercial studies take for granted; and they answer to statutory audit, data sovereignty and accountability obligations that restrict experimentation [13], [14]. Conversely, private technology firms treat rapid iteration as routine and absorb failure as a cost of learning. It follows that techniques validated in high-velocity commercial clouds cannot be assumed to transfer to a national identity registry or a health insurance portal without deliberate adaptation to procurement, governance and infrastructure realities.

In this review, ML for software deployment, or ML-for-deployment, refers to machine learning models that inform or automate decisions within the deployment phase of the software lifecycle: predicting build or release risk, gating progressive rollouts, autoscaling capacity, triggering rollback, and detecting post-deployment anomalies. The term is deliberately narrower than several neighbouring fields. It excludes MLOps, which concerns deploying and operating ML models themselves; general DevOps automation and release engineering tooling that involve no learned model; and software defect prediction aimed at code quality during development rather than at deployment decisions. Methodologically, this paper is a systematic literature review combined with an agenda-setting conceptual synthesis: Sections 2 and 3 report the review, Section 4 presents a synthesised case illustration, and Section 5 derives the agenda.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

The African public sector is the primary concern of this review for two reasons. First, digital government platforms are now central to service delivery across the continent: national identity registries, tax filing portals, health insurance systems and e-service platforms such as South African municipal e-government programmes, Kenya's eCitizen, Ghana.Gov and Rwanda's Irembo carry transactions on which citizens depend [15], [16]. Second, the deployment conditions surrounding these platforms differ materially from the settings in which most ML-for-deployment research is produced: procurement is slow and goods-oriented, legacy estates generate sparse telemetry, connectivity and cloud capacity are constrained, and specialist operations skills are scarce [15], [17], [18]. A deployment failure in such a platform is therefore both more likely to occur and harder to recover from, which makes evidence tailored to these conditions especially valuable.

The literature, however, remains fragmented, and evaluation practice is uneven, with the bulk of published research grounded in well-endowed private sector settings rather than government institutions. Related strands of work include software maintainability and defect prediction [19]-[22], CI/CD and deployment optimisation [23]-[25], the boundary with MLOps [26], broader syntheses of ML in software engineering [27]-[29], API deployment research [30] and reinforcement learning autoscaling [31]. Almost none of this body of work engages a public institution directly.

The study is guided by five research questions. RQ1: Which machine learning models are being used to improve software deployment, and at what stages of the pipeline? RQ2: How are these models designed, trained, and integrated into deployment workflows in practice? RQ3: What measurable gains do the models deliver compared to baselines, and how consistent are the gains across contexts? RQ4: To what extent have these approaches been applied in public sector deployments, and how do outcomes and constraints differ from private sector cases? RQ5: What gaps, risks, and open challenges limit the effectiveness or adoption of ML for deployment, and what research directions are most promising?

1.1. Software deployment pipeline

Software deployment is the process of moving code from development environments into production, where real users can access it [3]. Several stages are involved. The build stage compiles the source code, resolves dependencies and packages the resulting

binaries or containers, and a failure at any of these points can halt the release [3], [32]. Typical faults include missing dependencies, broken compilations and configuration errors. The test stage applies automated checks, from unit and integration tests to performance and security tests, to intercept defects before they reach production [33], [34]. The staging stage verifies behaviour in a production-like environment, while the canary or progressive rollout stage releases the new code to a small share of users before full deployment [3], [32]. Production release then makes the version generally available under continuous monitoring, with rollback to the previous stable version whenever serious anomalies appear. Each of these stages presents decisions that machine learning could inform: whether a build deserves priority, whether a test suite will expose the fault, whether a canary is trending badly, and when a rollback should fire.

Figure 1. Software deployment pipeline stages (adapted from [2], [5])

Figure 1 illustrates a modern deployment pipeline, beginning with compilation and artifact production, moving through automated inspection, verification in a production-like setting and staged user release, and ending with general availability. A de-escalation path returns the system to equilibrium when complications arise. Every phase is a control point for reliability, functionality and user safety in continuous delivery.

1.2. Machine learning task types

Many types of machine learning task are employed for software deployment [10], [11]. Classification forecasts build or deployment success and the associated level of risk [10]-[12], [35]. Regression and forecasting predict continuous values such as build duration, deployment time and resource requirements [11], [35], [36]. Anomaly detection identifies abnormal patterns in deployment data that frequently signal operational problems [11], [12], [35], [36], [37]. Reinforcement learning acquires autoscaling and deployment policies through interaction and feedback [31], [36], [38]-[40]. Clustering groups similar deployments or failures to surface common patterns [41], [42].

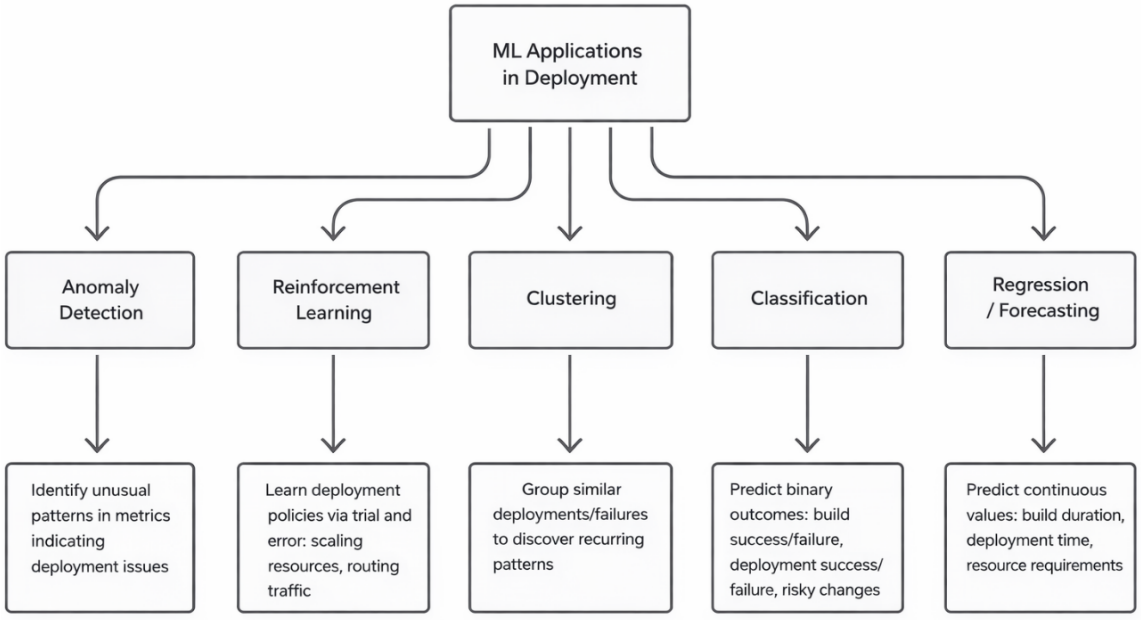

Figure 2. Machine learning applications in software deployment

Figure 2 divides the principal machine learning tasks in deployment pipelines into five groups: anomaly detection, reinforcement learning, clustering, classification, and regression or forecasting. Each serves a distinct purpose. Anomaly detection flags irregular deployment patterns, reinforcement learning improves decision-making policies,

clustering groups similar deployment events, classification predicts success or failure, and regression estimates continuous performance measures such as build time or resource consumption.

1.3. Challenges specific to deployment

Timing is critical, because predictions must arrive within the window available for the deployment decision [42], [43]. A ten-minute inference delay is of no use when the deployment window is five minutes. System behaviour also drifts, and a model may lose its effectiveness after a month of new data [43], [44]. Furthermore, failures are rare events, so training data contain many more successes than failures, and this imbalance can leave models poorly calibrated for the very cases that matter most [37]. Integrating ML into existing infrastructure, monitoring and change-management processes is a further burden that many teams cannot easily absorb [42]. Above all, high levels of trust are required before an organisation will permit a model to influence a release that could affect millions of users.

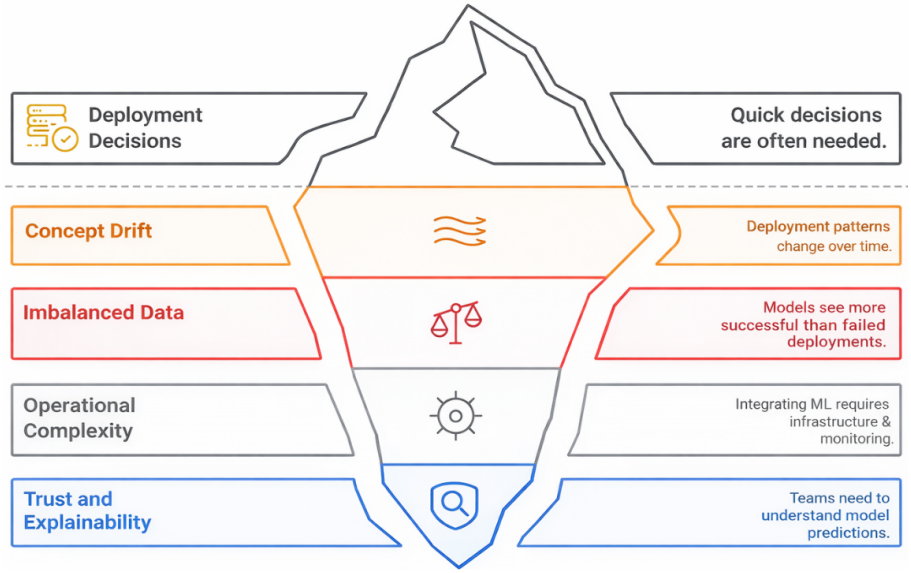

Figure 3. Key challenges in ML-based software deployment decisions (authors' own elaboration, adapted from [11], [34])

The iceberg diagram in Figure 3 summarises the less visible difficulties of machine-learning-controlled deployment, including concept drift, skewed data, operational complexity, and the trust and explainability concerns that undermine decision accuracy and long-term stability. The rest of this paper is organised as follows. Section 2 describes the systematic review methodology. Section 3 presents findings for each research

question. Section 4 presents a conceptual case study of ML-informed rollout and rollback decisions. Section 5 discusses implications and future directions, and Section 6 concludes.

2. METHODS

This study takes the form of a Systematic Literature Review (SLR) aimed at synthesising and critically analysing the use of machine learning in software deployment. The review protocol was defined in advance and follows the methodological guidance for SLRs in software engineering proposed by Kitchenham [45]. Reporting adheres to the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) 2020 statement [46] for transparency and reproducibility, and the completed checklist is provided as Supplementary File S2. A convergent mixed-methods design was chosen, combining quantitative summary statistics with qualitative thematic analysis. The aggregated evidence was additionally synthesised into a conceptual case (Section 4) to connect abstract findings to practice.

2.1. Search strategy

Retrieval bias was minimised through a comprehensive and replicable search strategy. Five large scholarly databases were searched: ACM Digital Library, IEEE Xplore, SpringerLink, Wiley Online Library and Elsevier ScienceDirect. A supplementary search in Google Scholar identified highly cited grey literature to reduce database-specific bias, although the core analysis used peer-reviewed sources only.

The search query was constructed on a Population (software deployment), Intervention (machine learning) and Context (pipelines) framework, with key terms identified through pilot searches and prior literature. The canonical Boolean query was: ("machine learning" OR "deep learning" OR "neural network" OR "classification" OR "regression" OR "reinforcement learning" OR "anomaly detection" OR "prediction") AND ("software deployment" OR "continuous delivery" OR "continuous deployment" OR "DevOps" OR "release engineering" OR "deployment pipeline" OR "canary deployment" OR "rollback").

To make the search reproducible for every information source, Table 1 records the string as executed in each database. The same two term blocks were applied throughout; only

the interface syntax and the field scope differ. No start-date restriction was imposed in any database, and no filters beyond document language were applied at the search stage.

Table 1. Search strings as executed in each information source

Source	Field scope	String as executed
IEEE Xplore	All metadata	("machine learning" OR "deep learning" OR "neural network" OR "classification" OR "regression" OR "reinforcement learning" OR "anomaly detection" OR "prediction") AND ("software deployment" OR "continuous delivery" OR "continuous deployment" OR "DevOps" OR "release engineering" OR "deployment pipeline" OR "canary deployment" OR "rollback")
ACM Digital Library	Title, abstract, keywords	("machine learning" OR "deep learning" OR "neural network" OR "classification" OR "regression" OR "reinforcement learning" OR "anomaly detection" OR "prediction") AND ("software deployment" OR "continuous delivery" OR "continuous deployment" OR "DevOps" OR "release engineering" OR "deployment pipeline" OR "canary deployment" OR "rollback")
SpringerLink	All fields	("machine learning" OR "deep learning" OR "neural network" OR "classification" OR "regression" OR "reinforcement learning" OR "anomaly detection" OR "prediction") AND ("software deployment" OR "continuous delivery" OR "continuous deployment" OR "DevOps" OR "release engineering" OR "deployment pipeline" OR "canary deployment" OR "rollback")
Wiley Online Library	All fields	("machine learning" OR "deep learning" OR "neural network" OR "classification" OR "regression" OR "reinforcement learning" OR "anomaly detection" OR "prediction") AND ("software deployment" OR "continuous delivery" OR "continuous deployment" OR "DevOps" OR

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

Source	Field scope	String as executed
		"release engineering" OR "deployment pipeline" OR "canary deployment" OR "rollback")
Elsevier ScienceDirect	Title, abstract, keywords	("machine learning" OR "deep learning" OR "neural network" OR "classification" OR "regression" OR "reinforcement learning" OR "anomaly detection" OR "prediction") AND ("software deployment" OR "continuous delivery" OR "continuous deployment" OR "DevOps" OR "release engineering" OR "deployment pipeline" OR "canary deployment" OR "rollback")
Google Scholar (supplementary)	All fields	("machine learning" OR "deep learning" OR "neural network" OR "classification" OR "regression" OR "reinforcement learning" OR "anomaly detection" OR "prediction") AND ("software deployment" OR "continuous delivery" OR "continuous deployment" OR "DevOps" OR "release engineering" OR "deployment pipeline" OR "canary deployment" OR "rollback")

Note: the identical term blocks were applied in every source; only interface syntax and field scope differ. No start-date restriction and no filters beyond document language were applied at the search stage.

The principal search was executed on 1 June 2025 and refreshed on 15 September 2025 to capture publications up to and including August 2025. With no start-date restriction, the searches returned an initial set of 869 records. Google Scholar functioned solely as a supplementary completeness check: its 1031 records were inspected for studies the databases might have missed, but none entered the 869-record pool, the deduplication step or the screening flow. Grey literature surfaced there, including preprints and theses, was excluded under EC1; the evidence base is therefore confined to peer-reviewed sources, and industry or government practice documented only outside peer review falls beyond the reach of this review. Duplicates were identified by exact DOI matching followed by normalised-title comparison during reference management. The duplicate count is high because the five databases overlap substantially in the venues they index,

so the same article was frequently retrieved from more than one source. Table 2 reports the number of records retrieved from each source before deduplication.

Table 2. Records retrieved per information source

Source	Records retrieved (before deduplication)
IEEE Xplore	131
ACM Digital Library	197
SpringerLink	114
Wiley Online Library	203
Elsevier ScienceDirect	224
Google Scholar (completeness check)	1031 – checked only, not counted.
Total	869

In response to peer review, a targeted supplementary search was designed to test directly whether African or public sector studies exist that the sector-neutral canonical query might have missed. The canonical query was combined with a third block of context terms: (Africa OR African OR "South Africa" OR "public sector" OR government OR "e-government" OR "digital government" OR "public administration" OR "civil service" OR "national digital service"), executed in the same five databases with the same field scopes as Table 1, on August 4, 2026. Table 3 reports the outcome per source: records retrieved, records remaining after deduplication, records screened at title and abstract, records excluded, full texts assessed, and studies meeting the inclusion criteria. In total, the targeted search retrieved 3,687 records across the five databases; after deduplication, 3,003 remained; 1,795 records were screened at title and abstract, 72 full texts were assessed, and no additional study met the inclusion criteria. The targeted search therefore identified no qualifying African or public sector deployment study, and the 0 of 33 findings reported in Section 3 reflect the combined main and targeted searches.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

Table 3. Targeted supplementary search: records and screening outcomes per source

Source	Retrieved	After dedup.	Screened (title /abstract)	Excluded	Full texts assessed	Included
IEEE Xplore	27	23	15	13	2	0
ACM Digital Library	6	4	4	3	1	0
SpringerLink	1,094	955	543	522	21	0
Wiley Online Library	1559	1217	722	697	25	0
Elsevier ScienceDirect	1,001	804	511	488	23	0
Total	**3,687**	**3,003**	**1,795**	**1,723**	**72**	**0**

Note: executed on August 4, 2026, reusing the Table 1 term blocks and field scopes with the Africa and public sector context block.

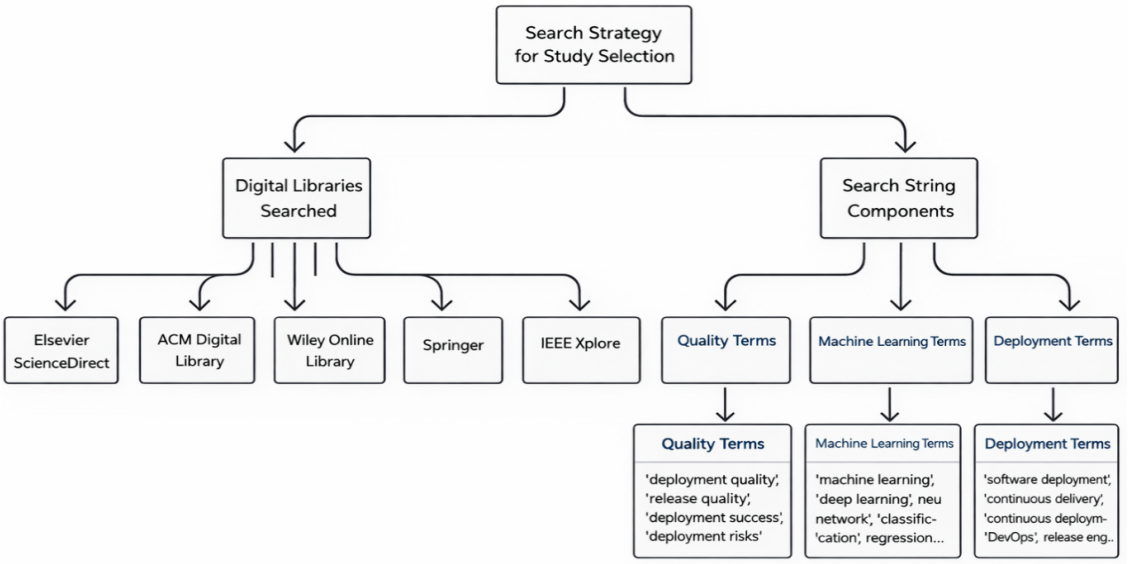

Figure 4. Search strategy for study selection

The diagram in Figure 4 illustrates the systematic search strategy. It covers the five digital libraries and defines the machine learning terms, deployment terms and quality terms that were combined with Boolean operators to locate relevant studies.

2.2. Study selection criteria

Selection was directed by a predetermined protocol that specified the following criteria, applied consistently to every record to reduce selection bias. The Inclusion criteria (IC) are as follows:

1) IC1 (publication type): full-length, peer-reviewed journal articles or conference proceedings papers.
2) IC2 (language): written in English.
3) IC3 (emphasis): ML applied as a principal mechanism in one or more parts of the software deployment pipeline (build, test, rollout, monitoring, rollback).
4) IC4 (evidence): provides substantive evidence on ML for deployment, whether a primary empirical assessment (experiment, case study or benchmark), a systematic secondary synthesis (review or survey), or a documented conceptual proposal; the study type of every included paper is recorded in Supplementary Table S1 (17 primary empirical studies, 11 secondary reviews or surveys, and 5 conceptual proposals). IC4 and EC4 operate together rather than in tension: EC4 removes position papers and visionary articles that offer neither implementation detail nor systematic synthesis, while IC4 retains conceptual proposals only where they document a concrete architecture, method or research design; the five retained conceptual papers all carry such documented technical substance.
5) Exclusion criteria (EC):
6) EC1 (publication venue): posters, theses, dissertations, short papers (fewer than 4 pages), workshop papers.
7) EC2 (topic): studies considering MLOps only, that is, the deployment and lifecycle management of ML models themselves.
8) EC3 (scope): papers concerned with software development, design or testing without a direct connection to deployment.
9) EC4 (content): position papers, visionary articles, or studies with no implementation or empirical data.
10) EC5 (duplicates): repeated publications of the same study.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

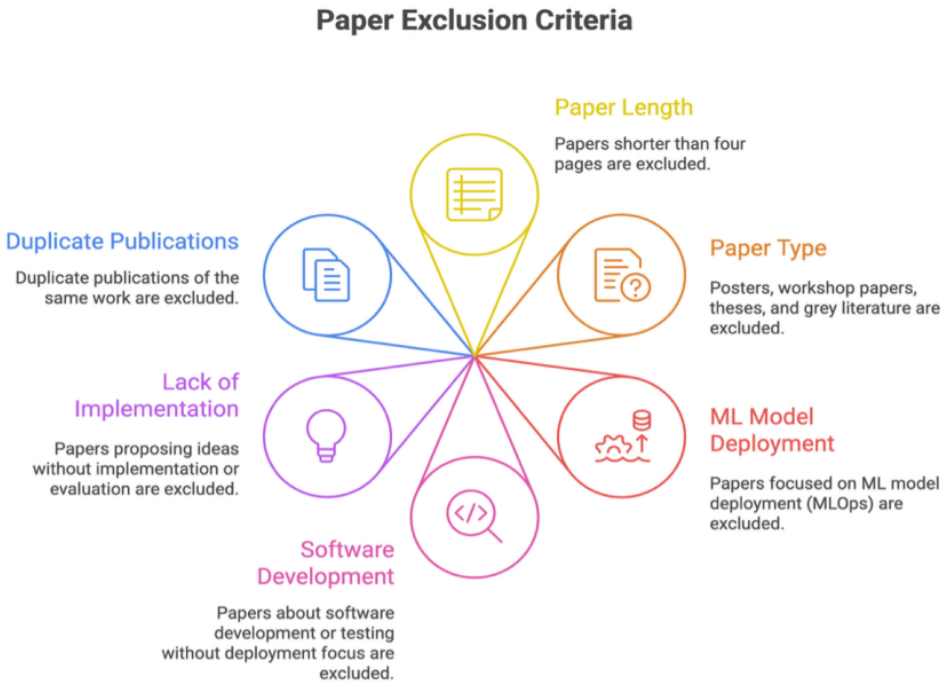

Figure 5. Paper exclusion criteria

Figure 5 summarizes the exclusion criteria applied during selection, covering manuscript length, document type, duplication, absence of implementation, a focus on development rather than deployment, and MLOps-only scope. This methodical filtering confined the corpus to relevant, empirically grounded studies.

2.3. Study selection process

Selection followed the PRISMA 2020 flow shown in Figure 6. After the initial search, 420 duplicate records were removed using automated tools (Zotero) with manual verification. A further 52 records were excluded for ineligible publication formats (book chapters, editorials) or missing metadata, leaving 397 records for title and abstract screening. Two reviewers screened these independently, and inter-rater reliability at this phase was strong (Cohen's kappa = 0.82). Disagreements were resolved through discussion. This step removed 324 records and advanced 73 articles to full-text review.

At the full-text stage, the two reviewers again assessed every article independently against the written criteria, and each exclusion required joint confirmation in a consensus meeting with the third reviewer; because every decision at this phase was consensus-validated in this way, a separate kappa statistic was not computed for it, whereas

agreement was quantified for the screening phase (kappa = 0.82). Of the 73 full texts, 40 papers were excluded: 11 for methodological weaknesses such as the absence of a comparative baseline, and 29 for being out of scope, for example MLOps-only or pure development studies. The final corpus comprised 33 included studies. Backward and forward citation searching of these studies yielded no additional qualifying papers, suggesting a saturated sample. The PRISMA flow diagram in Figure 6 shows the complete process of identification, screening, eligibility and inclusion, tracing how the dataset was refined from 869 initial records to 33 included studies; the per-source composition of those 869 records is given in Table 2.

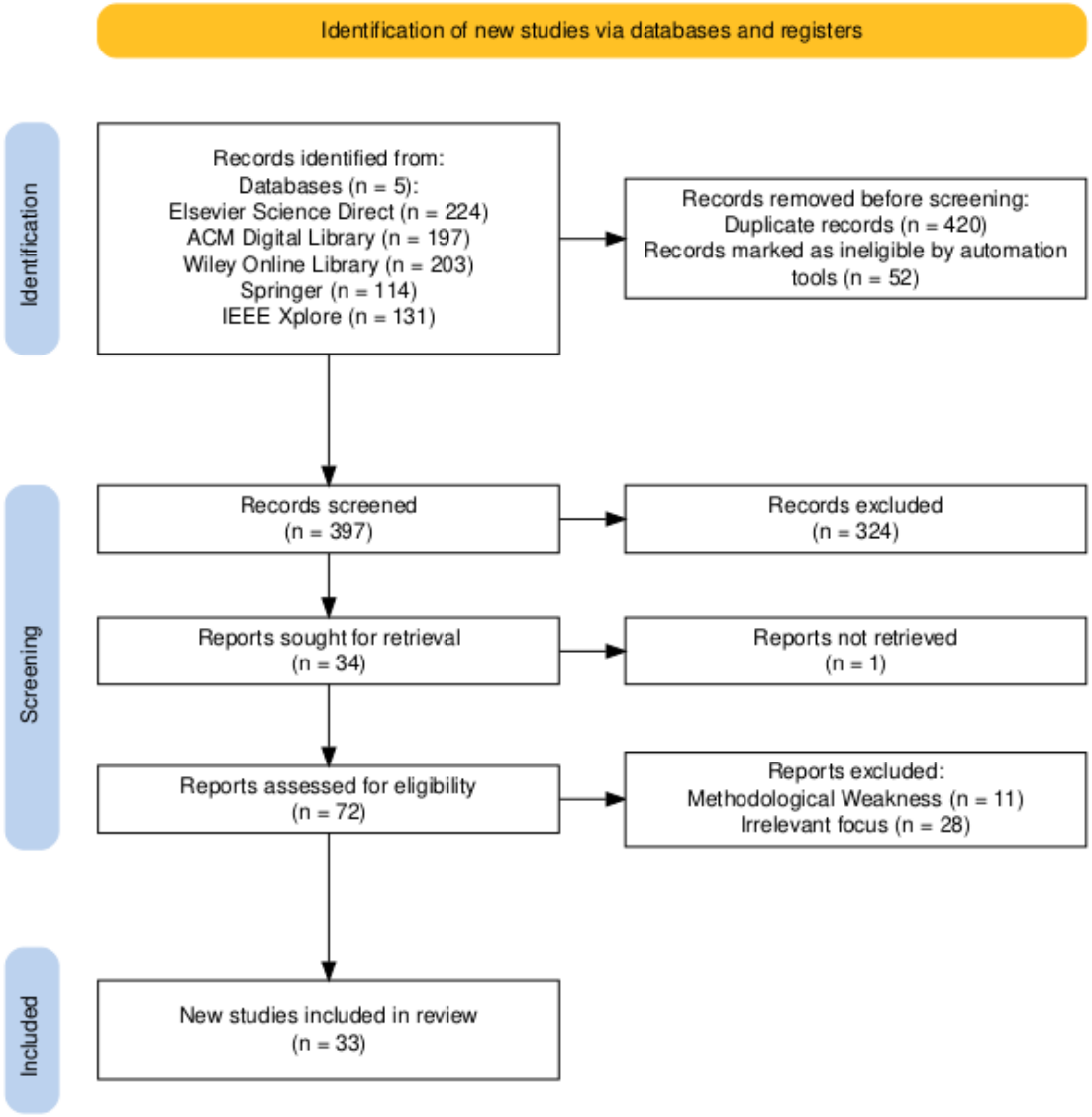

Figure 6. PRISMA paper screening process (adapted from [47])

2.4. Data extraction and data management

To maintain consistency, data were extracted into a standardised, pilot-tested Google Sheets form. The extraction fields were:

1) Bibliographic information: authors, title, venue, publication year, publication type.
2) Context: deployment stages covered, problem motivation, sector (public, private, unspecified).
3) ML methodology: model family, learning task, feature categories, training data source and size.
4) Assessment: performance measures, baseline procedures, reported improvement, statistical testing.
5) Operationalisation: integration pattern (for example, CI/CD plugin), degree of automation, and tools mentioned.
6) Limitations: problems and threats to validity declared by the primary study authors.

Extraction was performed by two reviewers, and a third reviewer verified a random sample of 30% of the extractions. Discrepancies were resolved in consensus meetings. The complete extraction dataset for the 33 studies, together with the per-study characteristics summarised in Supplementary Table S1, is available from the corresponding author on request. Sector was coded from explicit statements about the organisational origin of the data or the deployment context studied; where a paper named no sector, it was coded Unspecified rather than inferred, which is why 17 of 33 studies carry that label.

2.5. Quality evaluation (risk of bias)

A purpose-built checklist informed by previous software engineering SLRs was used to evaluate methodological quality and the risk of bias in the selected studies. Six criteria were rated on a 0 to 2 scale (No / Partial / Yes):

1) QC1: clarity of research objectives and questions.
2) QC2: suitability of research design and data collection.
3) QC3: detailed description of the ML approach.
4) QC4: rigour of evaluation, including comparison with credible baselines.
5) QC5: discussion of limitations and threats to validity.

6) QC6: code or data availability supporting verification.

The maximum attainable score was 12. The per-criterion scores were re-derived from the full text of every study during revision and are reported in full in Supplementary Table S1. Across the 33 studies the mean total score was 5.5 (SD = 3.2): 7 studies rate as low risk of bias (total 9 to 12), 12 as moderate (5 to 8) and 14 as high (0 to 4). The corpus is therefore methodologically heterogeneous, and its aggregate risk of bias is substantial rather than low; alongside a core of rigorous empirical studies, a considerable share of the included papers are narrative or conceptual contributions with limited evaluation. Supplementary Table S1 tabulates each study's score on every criterion together with its principal declared weaknesses and an overall risk rating, so that readers can trace exactly where bias is concentrated. The most frequent deductions arose under QC6, since public code and data remain the exception rather than the rule in this literature (22 of 33 studies score 0), followed by QC5, as roughly half the corpus declares no limitations at all. These weaknesses are treated as findings in their own right in Sections 3.3 and 3.4 and motivate the evaluation-rigour items of the research agenda.

2.6. Data synthesis methods

The extracted data were synthesised using a convergent mixed-methods approach.

1) Quantitative synthesis: for RQ1 (models and stages) and RQ3 (performance), descriptive statistics (frequencies and percentages with the denominator of 33 studies) were computed and visualised through bar charts and tables.
2) Qualitative synthesis: for RQ2 (design), RQ4 (context) and RQ5 (challenges), thematic analysis was performed through open coding, grouping of codes into categories, and refinement into general themes, with thematic saturation confirmed in reviewer discussion.
3) Translational synthesis: a synthesised case illustration of rollout and rollback decisions (Section 4) was assembled by combining design patterns, model architectures and reported issues from multiple studies into a single illustrative cloud-native scenario.

All 33 included studies enter the descriptive frequency counts with equal weight and with study type recorded per study; the counts are therefore claims about the literature's attention, not about empirical effect sizes, and no conceptual or secondary entry is

treated as empirical validation. Wherever a claim depends on measured performance, it draws only on the 17 primary empirical studies. Including reviews also risks double counting evidence that appears in primary form elsewhere; two safeguards apply. Each included paper is counted exactly once in every frequency count, and the model counting rule distinguishes families a secondary study merely reviews from families a primary study trains, so review-internal corpora are never merged into the counts.

2.7. Reproducibility of this review

For independent verification, the review makes the following artefacts explicit. Table 4 lists each reproducibility item and where it can be found. This review was not registered in a prospective registry, and no separate protocol was published; the a priori protocol is available from the corresponding author. The protocol was defined before screening began; the search strings for all five databases appear in Table 1; the PRISMA 2020 checklist is provided as Supplementary File S2; the identities, metadata and quality scores of all 33 included studies are tabulated in Supplementary Table S1; and the full extraction dataset is available from the corresponding author on request. Baselines used by the included studies are reported per study in Supplementary Table S1 and synthesised in Section 3.4. No new software was produced for this review, so a code repository is not applicable.

Table 4. Reproducibility checklist for this review

Item	Location / status
Review protocol defined a priori	Section 2; available from the corresponding author on request
PRISMA 2020 checklist	Supplementary File S2 (separate file)
Full search strings per database	Table 1
Inclusion and exclusion criteria	Section 2.2 (IC1-IC4; EC1-EC5)
List of the 33 included studies with metadata and quality scores	Supplementary Table S1 (separate file)
Data extraction form fields	Section 2.4
Inter-rater reliability	Screening kappa = 0.82; full-text decisions consensus-validated; per-study quality scores reported in full in Supplementary Table S1
Baselines used by the included studies	Supplementary Table S1 and Section 3.4

Item	Location / status
Extraction dataset	Available from the corresponding author on request
Analysis code	Not applicable (no new software was produced)

2.8. Limitations of the review methodology

Although the review was performed rigorously, several limitations should be acknowledged.

1) Database coverage: the search was limited to five large databases and peer-reviewed literature, which may have missed relevant studies in smaller venues or the grey literature (for example, industry technical reports), possibly under-representing the industrial state of practice.
2) Search string specificity: although extensive, the string may not capture every emerging terminology (for example, "AI-powered DevOps"), so omissions are possible.
3) Temporal scope: the area is highly dynamic, and studies published after September 2025 fall outside this review, which portrays the field up to that date.
4) Corpus quality: with a mean quality score of 5.5 of 12 and 14 of 33 included studies at high risk of bias, the evidence base is weak and heterogeneous, and every quantitative pattern reported here should be read with that constraint in view.
5) Subjectivity in synthesis: the qualitative thematic analysis and the conceptual case study, although moderated by multiple reviewers, inherently involve interpretive judgement.

3. RESULTS AND DISCUSSION

3.1. Overview of selected papers

The corpus comprised 33 manuscripts: 23 of 33 (70%) journal articles and 10 of 33 (30%) conference contributions. Table 5 shows the yearly distribution of publications. Output has risen since 2018, with pronounced growth between 2023 and 2025, when 19 of the 33 articles appeared. This pattern is consistent with descriptions of machine learning for

deployment as an emergent, rapidly expanding area. Of the 33 included studies, 17 are primary empirical studies, 11 are secondary reviews or surveys and 5 are conceptual proposals.

Table 5. Publication year and venue type distribution of the papers

Year	Conference papers	Journal papers	Total
2018	2	0	2
2019	1	2	3
2020	1	0	1
2021	1	4	5
2022	2	1	3
2023	1	5	6
2024	1	7	8
2025	1	4	5
Total	10	23	33

The included studies were published across a broad range of peer-reviewed venues, including IEEE outlets (3 of 33: IEEE Access and IEEE Transactions on Software Engineering), five IEEE-affiliated conference proceedings, four Springer outlets, and a long tail of independent journals; no single venue dominated the corpus. No single research group dominated the corpus. Each author appeared in only one paper within the final dataset, indicating that effort in this area is distributed across many independent teams rather than concentrated in a few institutions. For readers who do not require the methodological detail that follows, Table 6 condenses the review into a single summary, pairing each research question with its headline finding and the quantitative evidence behind it. The subsections then present each result in full.

Table 6. The review at a glance: headline findings by research question

RQ	Headline finding	Quantitative evidence
RQ1	Effort concentrates on autoscaling and build prediction, using predominantly tree-based models	12 of 33 (36%) autoscaling; 9 of 33 (27%) build prediction; tree-based models 16 of 33 (48%)
RQ2	Code metrics dominate model inputs, while data sources and validation designs are often unreported	Code metrics 17 of 33 (52%); unspecified data sources 11 of 33 (33%); unspecified splitting 19 of 33 (58%)
RQ3	Gains over baselines are consistently claimed but rarely statistically substantiated	Reported gains of 3 to 40 percentage points; statistical significance tests or confidence intervals in only 1 of 33 (3%)
RQ4	No public sector or African institutional study appears in the corpus	0 of 33 public sector; 16 of 33 (48%) private sector; 0 of 33 African evidence
RQ5	Explainability, concept drift and dataset scarcity are the binding constraints	Explainability 16 of 33 (48%); drift 11 of 33 (33%); dataset constraints 24% of coded challenge mentions

3.2. RQ1: which ML models are used, and at what deployment stages?

3.2.1. Deployment stages

Figure 7 maps the papers to phases of the deployment pipeline. Autoscaling is the single most studied stage, appearing in 12 of 33 papers (36%), where models scale computational resources dynamically during deployment. Build prediction follows in 9 of 33 papers (27%), covering build success prediction, build duration prediction and build prioritisation. Rollback decisions and anomaly detection each appear in 6 of 33 papers (18%). Deployment risk prediction appears in 5 of 33 papers (15%), and test selection or prioritisation in 3 of 33 papers (9%). Notably, 13 of 33 articles (39%) take a broader view, studying continuous integration and continuous deployment (CI/CD) or DevOps pipeline optimisation without confining themselves to one narrowly defined stage. Percentages sum to more than 100 because several studies examine multiple phases.

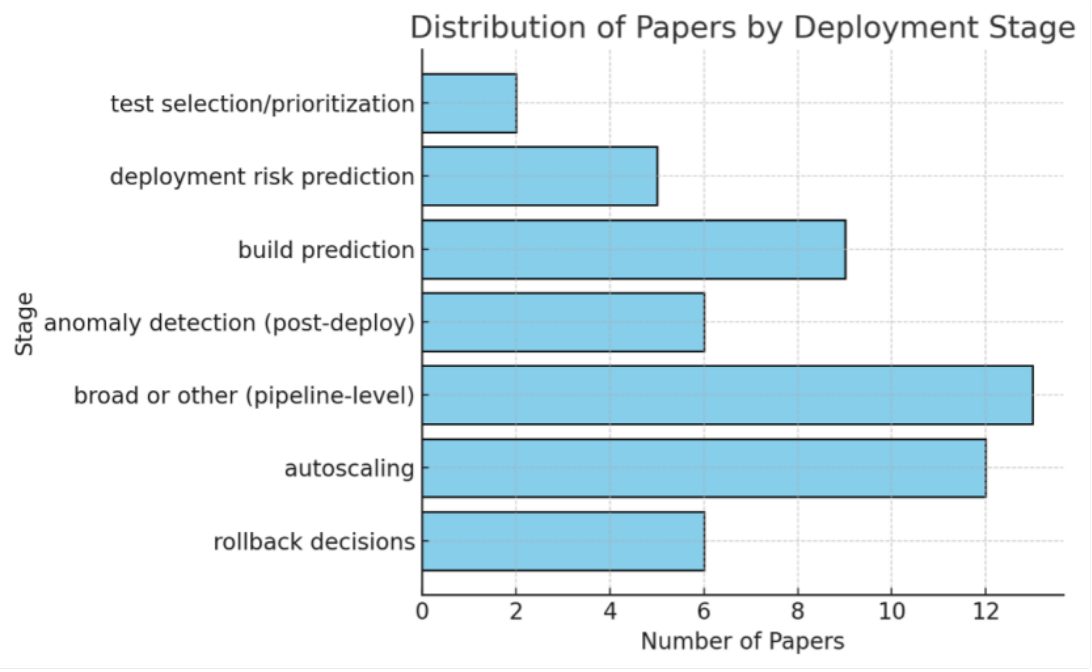

Figure 7. Distribution of papers by deployment stage

The bar chart in Figure 7 shows the distribution of the reviewed studies across deployment stages. Most of the literature concentrates on pipeline-level optimisation and autoscaling, followed by build prediction, anomaly detection and rollback decision-making, with smaller proportions addressing deployment risk prediction and test selection.

3.2.2. ML model families

Figure 8 summarises the machine learning model families named in the recorded model and baseline descriptions of the 33 studies, counting a family wherever a study trains it, evaluates it as a comparator, or, in the case of secondary studies, explicitly reviews it; a family attributed to a secondary study therefore reflects models that study reviews or surveys, not models it trains, and conceptual proposals contribute only the families they explicitly specify. Tree-based methods are the most common family, named in 16 of 33 studies (48%) overall: Random Forests in 12 of 33 (36%), Decision Trees in 7 of 33 (21%) and gradient boosting methods such as XGBoost and AdaBoost in 5 of 33 (15%). Support Vector Machines (SVMs) appear in 9 of 33 papers (27%), the most frequently named individual model after Random Forests. Neural networks of various forms appear in 11 of 33 studies (33%), comprising generic feed-forward or extreme learning machine designs in 7 of 33 (21%), recurrent or LSTM models in 3 of 33 (9%) and convolutional networks in

2 of 33 (6%). Reinforcement learning features in 7 of 33 studies (21%), applied to autoscaling and sequential decision-making. Classical statistical methods include logistic regression in 4 studies, Naive Bayes in 3, and Bayesian networks and k-nearest neighbours in 1 each. Explicit ensemble frameworks are named in 2 of 33 (6%), and Transformer or large language model architectures in 1 of 33 (3%). Individual family counts sum to more than the family-union figure because a study naming several tree-based methods contributes once to the union but to each individual count; the union and the individual counts are therefore reported separately and are not additive.

The dominance of tree-based methods is not accidental, and interpreting it matters for deployment reliability. Deployment telemetry is overwhelmingly tabular, moderately sized and heavily imbalanced, and on such data tree ensembles are hard to beat: they require little feature scaling, tolerate missing values, and expose comparatively interpretable decision logic that operators can interrogate before trusting a release decision [19]-[22]. Deep architectures, by contrast, repay their complexity only where large sequential or multimodal corpora exist, which explains their concentration in latency-trace anomaly detection rather than build prediction. It follows that the field's model choices are being driven by data shape and by the operational need for explainable decisions, not merely by fashion, and any benchmark effort for this area should therefore privilege tabular, imbalanced, drift-prone datasets over the large clean corpora typical of mainstream ML benchmarks.

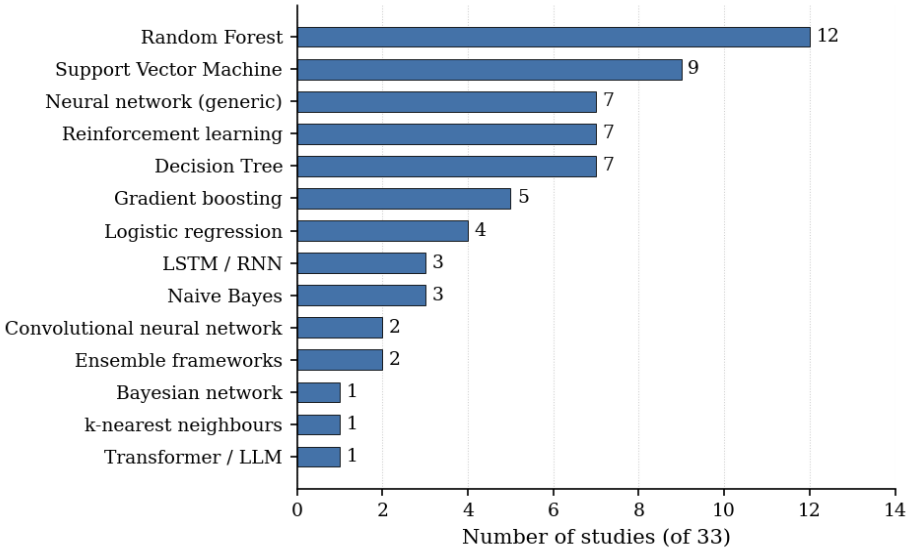

Figure 8. Machine learning models used across papers

The bar chart in Figure 8 shows the distribution of model families across the reviewed studies. Random Forests and SVMs lead, followed by Decision Trees, neural networks and reinforcement learning, reflecting a preference for interpretable and well-established algorithms in deployment research.

3.2.3. Task types

Most papers combine several task types. Classification is the most common, found in 27 of 33 studies (82%), where models predict binary or multi-class outcomes such as deployment success or failure, build success or failure, or risk level. Anomaly detection appears in 16 of 33 publications (48%), uncovering unusual trends in deployment metrics or system behaviour. Regression and forecasting appear in 14 of 33 studies (42%), estimating continuous variables such as build duration, resource demand or performance measures. Reinforcement learning appears in 10 of 33 studies (30%), mainly to learn policies for autoscaling and adaptive deployment decisions. The prevalence of classification reflects both the availability of labelled historical deployment data and the direct usefulness of categorical predictions in deployment decision-making. The broader trend aligns with research on ML across software engineering processes and quality estimation tasks [27], [28]. The bar chart in Figure 9 presents the distribution of machine learning task types in the analysed studies. Classification is the most common activity, with anomaly detection, regression and forecasting, and reinforcement learning following, demonstrating a wide range of ML use in deployment pipelines.

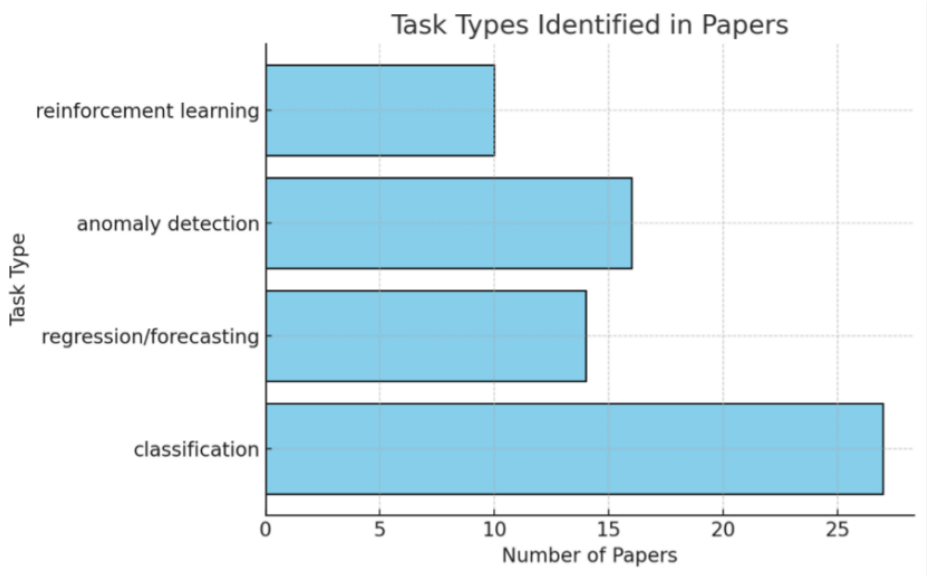

Figure 9. Task types identified in reviewed papers

3.3. RQ2: how are models designed, trained and integrated?

3.3.1. Features

Figure 10 categorises the features used across the papers. Code metrics appear most frequently, in 17 of 33 papers (52%), including lines of code changed, number of files modified, complexity metrics and code churn. Test metrics appear in 6 of 33 papers (18%), covering coverage, execution time and failure rates. Historical deployment data appear in 5 of 33 papers (15%), capturing previous outcomes, deployment frequency and time since the last deployment. Developer and team metrics such as experience and team size appear in 4 of 33 papers (12%), and system metrics in only 1 of 33 (3%). Notably, 14 of 33 papers (42%) describe features in general or unspecified terms. Most papers (64%) combine multiple feature types rather than relying on a single category, so percentages again sum to more than 100. The bar chart in Figure 10 describes the input features used in the surveyed studies. Code metrics dominate, with unspecified or combined features next, followed by test metrics, historical deployment data, developer and team metrics, and a low rate of system metrics, reflecting the heterogeneity of data sources used to train models.

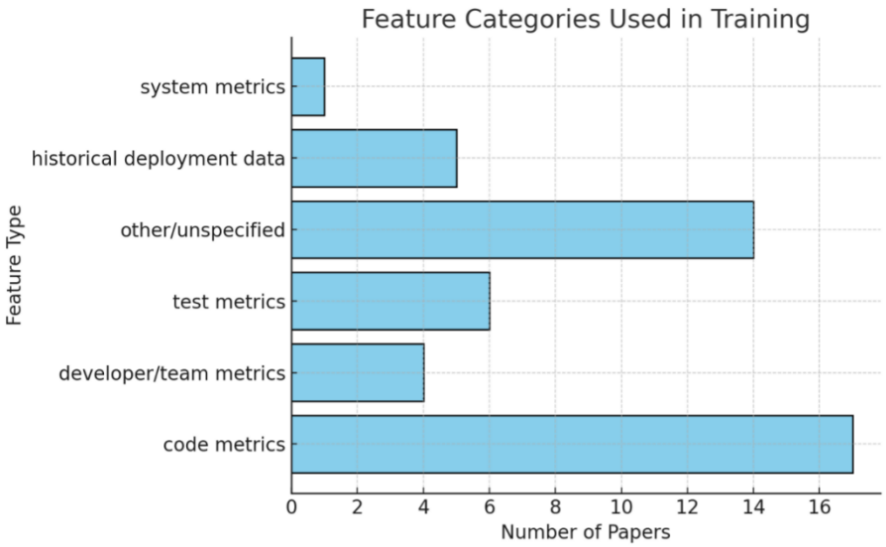

Figure 10. Feature categories used in model training

3.3.2. Training data and evaluation

Data sources vary. Company-internal data are used in 16 of 33 papers (48%), open-source projects in 5 of 33 (15%), both internal and open-source in 2 of 33 (6%), and simulated or synthetic data in 8 of 33 (24%). These categories are not mutually exclusive: a study may draw on several source types, so the counts sum to more than 33. Notably, 11 of 33 papers

(33%) do not clearly specify their data sources. Class or data imbalance is explicitly reported as a challenge in 18 of 33 papers (55%). Among handling strategies, SMOTE or other sampling techniques appear in 2 papers and ensemble-style approaches in 1; most papers acknowledge the imbalance but do not detail remediation.

On validation design, 8 of 33 studies (24%) used cross-validation, 4 of 33 (12%) used time-based splits and 2 of 33 (6%) used random splits, while the remaining 19 of 33 papers (58%) either did not specify their splitting strategy or adopted alternative arrangements. The most common evaluation metrics were precision (14 of 33 studies, 42%), accuracy (13 of 33, 39%), recall and F1-score (11 of 33 each, 33%), AUC/ROC (8 of 33, 24%) and mean absolute error (2 of 33, 6%). Only 1 of 33 papers (3%) reported explicit statistical significance tests or confidence intervals, a shortfall that recurs throughout this review.

This metric inconsistency deserves interpretation rather than mere description. It arises because deployment tasks are heterogeneous (a rollback trigger and a build-time forecast cannot share one headline metric), because no shared benchmark dataset exists against which metric conventions could stabilise, and because confidentiality prevents the data sharing that would allow replication. The practical consequence for deployment reliability is direct. Without common metrics, comparable baselines and uncertainty estimates, a practitioner cannot tell whether a published model would outperform a simple rule in their own pipeline, and reported accuracy figures on imbalanced deployment data can be actively misleading when a trivial always-predict-success classifier already scores highly. Evaluation discipline is therefore not an academic nicety in this field; it is a precondition for safe adoption.

3.3.3. Integration patterns

Table 7 records how the machine learning models are integrated into deployment pipelines. CI/CD plugin-style integration is most common (17 of 33 papers, 52%), followed by standalone tools (10 of 33, 30%) and monitoring dashboards (9 of 33, 27%). Some contributions remain purely conceptual without implementation detail (2 of 33, 6%), and others leave integration unspecified (10 of 33, 30%). Regarding human interaction, most studied systems retain human supervision (24 of 33 papers, 73%), with the model recommending and a human operator deciding. Only 5 of 33 papers (15%) describe fully automated deployments, and 4 of 33 (12%) do not specify the automation level. Recent

work on AI-based CI/CD and deployment optimisation describes similar plugin-centred, workflow-integrated implementation patterns [23], [25].

Table 7. Integration patterns and tools

Category	Subcategory / tool	Number of papers (of 33)
Integration type	CI/CD plugin	17 (52%)
	Standalone tool	10 (30%)
	Monitoring dashboard	9 (27%)
	Not implemented	2 (6%)
	Unspecified	10 (30%)
Human involvement	Human-in-the-loop	24 (73%)
	Fully automated	5 (15%)
	Unspecified	4 (12%)
Platforms and tools	TensorFlow	11 (33%)
	MLflow	7 (21%)
	Kubernetes	5 (15%)
	Kubeflow	5 (15%)
	PyTorch	1 (3%)
	Other / unspecified	15 (45%)
Tool multiplicity	Single tool	12 (36%)
	Multiple tools	10 (30%)

Note: a study may adopt several integration types, involvement modes and tools, so counts within each category sum to more than 33.

The most frequently mentioned tools and platforms are TensorFlow (11 of 33 papers, 33%), MLflow (7 of 33, 21%), Kubernetes (5 of 33, 15%) and Kubeflow (5 of 33, 15%). PyTorch

is mentioned in 1 of 33 papers (3%), and 15 of 33 (45%) use alternative tools or leave the stack unspecified. About 10 of 33 articles (30%) document simultaneous use of more than one tool or platform, underlining the complexity of modern deployment infrastructure. Related optimisation work similarly emphasises platform-level automation in microservices and CI/CD environments [24].

3.4. RQ3: what measurable gains do models deliver?

3.4.1. Performance metrics and baseline comparisons

All 33 papers report performance metrics and baseline comparisons of some kind. The most common measures are precision (14 of 33, 42%), accuracy (13 of 33, 39%), recall (11 of 33, 33%), F1-score (11 of 33, 33%) and AUC/ROC (8 of 33, 24%), with a few studies using custom, deployment-specific measures. Reported performance varies widely across studies and tasks, reflecting differences in problem complexity, dataset characteristics and baseline definition. Qualitative improvements over baselines are consistently claimed, and quantitative percentage gains were recorded during data extraction; however, the diversity of contexts, metrics and baselines prevents exact cross-paper comparison. Common baselines include random selection, rule-based heuristics and simpler ML models. Reported gains range from modest (3 to 10 percentage points above strong baselines) to large (20 to 40 percentage points above weak baselines such as random selection), with the size of the improvement strongly dependent on baseline strength, task difficulty and dataset traits. Only 1 of 33 articles (3%) reports statistical significance tests or confidence intervals, which severely limits confidence in the claimed gains.

3.4.2. Business and operational impact

Several studies record business and operational impact [35], [36], [38], though the reported figures are eclectic and context-specific. The main impact types are temporal efficiencies such as shorter build and deployment times [35], financial savings such as lower cloud resource costs from improved autoscaling [39], [40], and reliability improvements such as fewer deployment failures and greater uptime. Quantitative business impact is often expressed qualitatively or in broad ranges, which prevents the extraction of representative benchmarks. In turn, the absence of standardised business measures remains a real deficiency when justifying value beyond technical performance indicators.

3.4.3. Consistency across contexts and performance decay

The included studies note that results vary along contextual dimensions, including project size, deployment frequency, team size and system complexity. As a rule, larger and more active projects exhibit different patterns from smaller, less active ones. Concept drift or model degradation over time is explicitly recognised in 11 of 33 papers (33%), generally as a theoretical risk that models decay without retraining; few studies measure this degradation or propose specific mitigation. The scarcity of longitudinal production studies means performance decay remains, in most cases, a matter of theory rather than evidence.

3.5. RQ4: public sector versus private sector applications

The analysis reveals a pronounced sectoral imbalance in the literature on ML for software deployment. Of the 33 included studies, 16 (48%) explicitly focus on private sector contexts, primarily technology firms, SaaS vendors, e-commerce platforms and commercial startups, while the remaining 17 of 33 (52%) either lack sectoral specificity or provide insufficient contextual detail for definitive classification. No study (0 of 33) directly addresses a public sector deployment context. Two studies touch adjacent notions of 'public' infrastructure - one evaluates fault prediction on the publicly available NASA defect dataset and another optimises autoscaling for public cloud platforms - but neither engages a government or public administration setting. None of the 33 studies reports evidence from an African institution.

Although 19 of 33 studies (58%) acknowledge governance or compliance considerations, and 9 of 33 each (27%) discuss privacy constraints or organisational risk tolerance, these reflections remain generic and decontextualised, rarely distinguishing the legal, operational or infrastructural realities of public versus private institutions. Substantive discussion of legacy system integration, rigid procurement cycles, long approval chains and interoperability, all well-documented barriers in public sector IT modernisation [14], is notably absent from the corpus.

Because no included study engages a public sector deployment context, the distinctive requirements of that setting must be drawn from the public administration literature rather than from the corpus itself: stricter regulatory regimes such as data sovereignty and auditability requirements, multi-stakeholder decision-making, prolonged

procurement timelines, and definitions of deployment success that extend beyond uptime or cost to include policy compliance, service equity and public accountability [13], [14].

Meanwhile, the private sector literature concentrates on high-velocity, cloud-native domains such as web services, mobile applications and scalable SaaS products, where rapid iteration and failure tolerance are culturally and technically embedded. By comparison, critical public sector domains such as health information systems, tax and benefits administration, justice platforms, digital education infrastructure and national service portals are absent from the reviewed corpus. This omission is not incidental. These domains operate under heightened regulatory scrutiny, lower failure tolerance, greater public visibility and complex socio-technical constraints, all of which may deter both industry investment and academic inquiry. Consequently, current research offers limited insight into how ML-driven deployment tools can be adapted, validated or responsibly deployed in precisely the contexts where equitable, reliable and transparent digital services are most needed. These counts reflect the combined principal and targeted searches (Section 2.1).

The African dimension sharpens the contrast further. Deployment in many African public institutions must contend with constrained and intermittent bandwidth, heterogeneous legacy estates accumulated across successive donor-funded projects, thin observability tooling, a shortage of specialised MLOps skills within the civil service, and procurement rules that were written for physical goods rather than continuously evolving software. Private sector assumptions such as elastic cloud capacity, complete telemetry and permissive experimentation simply do not hold. It follows that ML-for-deployment techniques intended for these settings must be lightweight enough to run on modest infrastructure, explainable enough to satisfy audit and oversight bodies, and conservative enough to operate safely where a failed release can suspend access to identity verification, health insurance validation or vehicle registration for an entire population.

3.6. RQ5: gaps, risks and open challenges

3.6.1. Technical challenges

All 33 papers record technical challenges. The recurring themes are explainability and interpretability, concept drift and model decay, dataset limitations, and feature

engineering complexity. Explainability and interpretability are noted in 16 of 33 papers (48%), which specifically report the need for models whose predictions human operators can understand, especially where operators must trust and justify a model's decision. Concept drift and model decay are cited in 11 of 33 papers (33%) as a risk that performance erodes as deployment patterns change, though most studies acknowledge this without quantifying it. Dataset-related constraints, including the absence of public deployment datasets, the lack of standardised benchmarks and the difficulty of sharing deployment data owing to confidentiality, recur throughout the corpus and account for 24% of the coded challenge mentions shown in Figure 11. Several studies also stress that effective feature engineering requires substantial domain knowledge of deployment processes, and that automated feature learning for this context remains under-studied. Figure 11 summarises the prevalence of the principal technical problems in the reviewed studies. Explainability and interpretability were cited most often (48%), followed by concept drift and model decay (33%), dataset constraints (24%) and feature engineering complexity (18%).

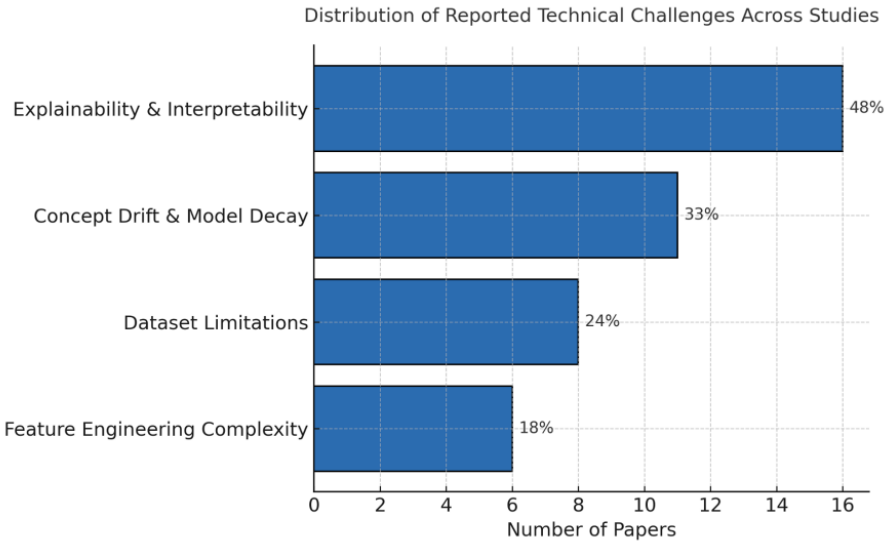

Figure 11. Reported technical challenges in ML-based deployment

3.6.2. Evaluation and operational challenges

Every paper report evaluation and operational difficulties. The studies rely on divergent evaluation measures, which frustrates comparison, and no shared agreement exists on baseline performance for deployment-focused tasks. Reproducibility remains a concern, as many studies provide neither public code nor datasets, limiting independent

verification. Although several papers acknowledge threats to validity, most say little about external validity or the transferability of findings beyond their settings. Operational realities such as infrastructure cost, long-term maintenance burden and the practical difficulty of integrating models into production pipelines are consistently under-reported. Moreover, most evaluations are retrospective rather than prospective, which is insufficient to substantiate true effectiveness in a live deployment setting.

3.6.3. Ethical and safety concerns

Most papers touch on ethical considerations only at the surface, with little analysis of how such concerns can be addressed in practice. Common issues nonetheless emerge across the corpus. Safety risks arise when models decide wrongly, especially in automated rollback or autoscaling. Accountability remains unresolved, since it is unclear who bears responsibility when an ML-driven deployment decision causes a failure. Fairness concerns appear because models trained on historical data may reproduce or amplify existing biases. Many papers additionally note the absence of clear audit trails and limited explainability, both of which matter for regulatory compliance in transparency-critical environments.

These observations connect directly to the wider global debate on trustworthy artificial intelligence in the public sector, and the connection deserves to be made explicit. Explanation techniques such as local surrogate models were proposed precisely because users must be able to interrogate why a prediction was made before acting on it [48], and assurance frameworks for the ML lifecycle argue that safety cases, not accuracy scores, are the appropriate currency for high-stakes automation [38]. Surveys of AI adoption in public administration report that trust, accountability and organisational governance, rather than algorithmic capability, are the binding constraints on uptake [13], [14], while studies of real-world ML deployment document how opaque models erode operator confidence [8] and how MLOps practice struggles to institutionalise auditability [26]. It follows that for government deployment pipelines, explainability and auditability are not desirable extras but preconditions of legitimacy: a rollback decided by a model must be explainable to an auditor general as readily as to an engineer. Yet the reviewed corpus offers no predefined framework for evaluating ethical risk in deployment automation, and closing that gap is itself a research task of the first order.

3.6.4. Priority research gaps and future directions

Research gaps and future directions are discussed throughout the corpus, and several cross-cutting issues stand out. 10 of 33 studies (30%) call for standardised benchmark datasets and consistent evaluation protocols so that results become comparable across deployment tasks. There is a severe shortage of public sector research, confirming the RQ4 finding. Although many papers acknowledge the importance of explainability, very few propose or assess methods tailored to deployment decisions. Concept drift is widely recognised, yet systematic techniques for detecting and adapting to drift in deployment settings remain limited. Most studies emphasise model performance alone, with little exploration of how humans and ML systems should collaborate during deployment decisions. The literature also tends to optimise isolated pipeline stages rather than the end-to-end workflow. Finally, reporting practice needs improvement, since many papers lack statistical significance tests, confidence intervals, clear baseline descriptions and sufficient reproducibility detail. These gaps represent opportunities to make ML-based deployment tools more reliable, trustworthy and widely adopted across diverse organisational contexts, and Section 5.4 organises them into a staged agenda. To close the results, Table 8 pairs each research question with its evidence-based finding and the agenda element derived from it, so that review evidence and agenda recommendations remain clearly distinguishable.

Table 8. Research questions, evidence-based findings and their agenda derivations

RQ	Evidence-based finding (Sections 3.2 to 3.6)	Agenda derivation (Section 5.4)
RQ1	Effort concentrates on autoscaling (12 of 33) and build prediction (9 of 33); the tree-based family is named in 16 of 33	Short term: lightweight tree-ensemble and hybrid rule-plus-ML pilots on accessible stages
RQ2	Code metrics dominate inputs (17 of 33); data sources unspecified in 11 of 33; validation design unspecified in 19 of 33	Medium term: shared benchmarks, open datasets and standardised reporting
RQ3	Gains over baselines are claimed widely, but only 1 of 33 reports significance tests or confidence intervals	Short to medium term: statistical rigour, uncertainty quantification and credible baselines

RQ4	No public sector study and no African institutional evidence in the corpus; 16 of 33 explicitly private sector	Long term: context-aware deployment support for African public institutions, beginning with targeted empirical pilots
RQ5	Explainability (16 of 33), drift (11 of 33) and dataset constraints (24% of coded mentions) dominate reported challenges	Short term: explainable ML pilots; medium term: drift-aware retraining and telemetry-sharing agreements

The middle column reports coded review evidence; the right column is the conceptual agenda derived from it and is not itself an empirical finding.

4. A SYNTHESISED CASE ILLUSTRATION: ML-ASSISTED ROLLOUT AND ROLLBACK DECISIONS IN CLOUD-NATIVE PIPELINES

This section provides a synthesised case illustration constructed from patterns observed across the included studies. It is not an empirical case from a single organisation but a synthesis of common deployment strategies. When rolling out new releases in cloud-native systems, the decision to deploy or roll back is made in real time and driven by the need to reduce downtime, financial loss and reputational damage [12], [40], [43]. The section examines how machine learning models predict the performance of new releases from historical data such as deployment logs, build metrics and runtime error signals.

4.1. Design and training of the model

Across the reviewed studies, two model types recur for predictive rollout decisions and are adopted here as one illustrative configuration, not as the demonstrated best solution: the tree-based ensemble XGBoost and recurrent neural networks (LSTM) [11], [12], [41], [49].

1) XGBoost models consume deployment metadata such as build success rates, regression rates, code churn, hour and day of week, and historical outcomes [12], [41], producing a binary forecast of how a release is likely to fare.
2) LSTM networks consume sequential telemetry, sampled at ten-second intervals, to watch for irregularities during the test phase of a rollout [40], and can detect when a release is beginning to destabilise and predict deterioration before it reaches production scale [49].

In the synthesised design, data are partitioned into 70% training, 15% validation and 15% test sets with cross-checking. XGBoost feature attributions identified recent failures and slow build times as the strongest signals, while the LSTMs surfaced latency spikes that precede deployment instability.

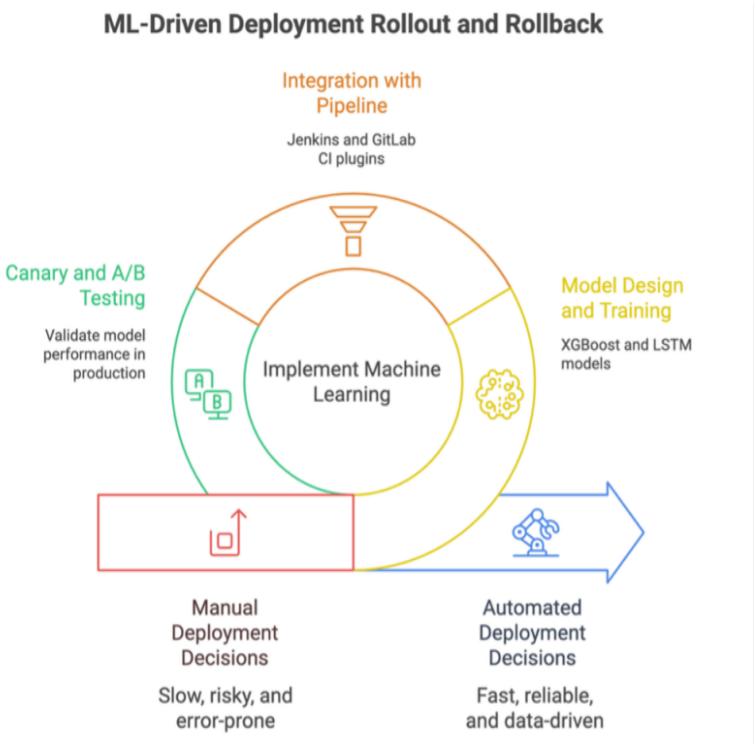

Figure 12. Model design and training

Figure 12 contrasts traditional deployment decision-making, which is slow, error-prone and high-risk, with the ML-driven alternative. The pipeline comprises four stages. Model design and training uses XGBoost and LSTM models trained on deployment history. Integration with the pipeline packages these models as containerised services rolled out via Jenkins or GitLab CI. The central core makes real-time decisions on whether to continue the rollout or revert. The canary and A/B testing phase verifies correct behaviour through progressive traffic checks from 10% to 50% to 100%, culminating in automated deployment decisions.

4.2. ML integration into the deployment pipelines

Established CI/CD tools such as Jenkins, GitLab CI and GitHub Actions integrate the ML components into the deployment pipelines [37], [41]. In these arrangements a model-generated score determines the next action in the workflow. A score above 0.8 allows

the pipeline to proceed with a canary deployment. Scores below 0.5 are routed to a human reviewer, who decides whether to continue the rollout or return the build for further checks.

4.3. Testing for validation with canary and A/B deployments

Canary deployments route 10%, 25%, 50% and finally 100% of traffic to the new version when new releases are introduced [50]. Progression to the next level occurs automatically if the model predicts a failure likelihood below 0.2% and monitoring metrics such as error rate, response time and memory usage stay within limits [40]. A/B testing has been used to verify the effectiveness of predictive rollback experimentally [51]. Control groups received conventional fixed-threshold rollouts, while treatment groups relied on ML to control rollouts and reverse them on early signs of trouble. In one 30-day controlled comparison synthesised from the SDLC-wide review by Shafiq et al. [11], the ML-assisted configuration recorded 28 to 35 fewer rollbacks, detected anomalies 40% faster and increased the frequency of successful deployments by 15% relative to a human-driven baseline. These figures originate from individual studies conducted in particular settings; they are illustrative single-context results and do not describe the measured performance of one validated system.

4.4. Business and operational impacts

Individual studies synthesised in the lifecycle assurance survey by Ashmore et al. [38] also report a predictive rollback arrangement reducing average recovery time from failure to about 13 minutes and lifting service availability by seven per cent [38]; again, these are single-context results rather than outcomes of a common validated system. From the perspective of operators and end users, false rollbacks fell and confidence in deployment outcomes rose [33]. The result was less on-call burden for operations teams and a faster release cadence [52].

4.5. Limitations and lessons

Despite these successes, the studies encountered recurring difficulties, including the need to retrain periodically as deployment patterns changed and imbalanced data requiring techniques such as oversampling. Making the decision process understandable was a further problem, addressed through SHAP values and attention heatmaps that show how individual features influence the rollback decision. Some organisations

declined to collect the detailed telemetry the system requires, citing privacy and compliance concerns.

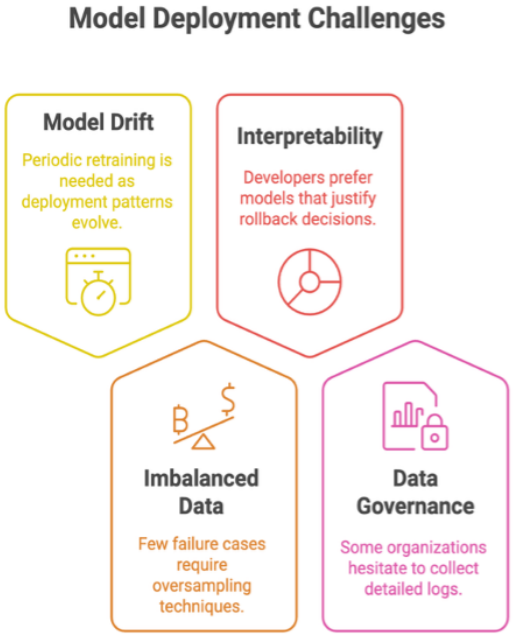

Figure 13. Challenges with model deployments

As Figure 13 shows, four principal obstacles arise when operating the XGBoost and LSTM models. Model drift, where usage patterns change over time, demands periodic retraining and sustained maintenance investment. The need for transparent decision-making is met through techniques such as SHAP values and attention heatmaps. Imbalanced data, the scarcity of rare failures in training sets, is mitigated with oversampling techniques such as SMOTE so that models better predict the failures that do occur. Data governance resistance, arising from privacy and compliance concerns about recording detailed system behaviour, restricts the available data and limits the demonstrated benefits.

4.6. Findings

The synthesised evidence indicates that a combination of XGBoost and LSTM is one configuration reported for predicting the likelihood of software and application errors, and that such predictive models are being merged with canary and A/B testing to create a streamlined continuous deployment process. This fusion, sometimes described as predictive automation, offers DevOps teams in cloud-native environments a practical way to accelerate deployment while sharply reducing operational risk.

5. DISCUSSION

For many African countries, large-scale digital government platforms such as identity systems, tax portals, health insurance registries, and passport or visa services are increasingly central to service delivery. Failures in these deployments can undermine public trust, widen digital divides and interrupt access to essential services. Yet none of the studies in this corpus examines such systems, and none reports evidence from African institutions. This absence is troubling, because deployment conditions in African public sectors are marked by resource constraints, heterogeneous legacy systems and stringent data governance obligations that differ substantially from those in the global North. In African public institutions, a failed deployment can suspend digital identity authentication, driver and vehicle registration or health insurance verification, so the absence of ML-based deployment studies in these settings leaves a major gap in both research and practice.

5.1. The state of ML for deployment

Broader mapping studies characterise ML-for-deployment as one strand of a wider, uneven interaction between AI techniques and software engineering practice [29]. On this analysis, ML in software deployment is a promising but under-developed research area. Effort concentrates on autoscaling and build prediction, where historical CI/CD data are relatively accessible and prediction targets map cleanly onto operational decisions. The prominence of tree-based and other classical models further indicates that deployment datasets tend to be tabular, noisy and imbalanced rather than large multimodal corpora, which makes interpretability and data efficiency first-order design considerations. Related work on CI/CD optimisation, deployment automation, API deployment and the broader ML-in-software-engineering practice reports the same limits on evaluation rigour, reproducibility and viable integration [23]-[25], [27], [28], [30].

5.2. The public sector research gap

The absence of public sector applications from the corpus is remarkable, but it must be read as a research and publication gap rather than proof that no such systems exist in practice: relevant deployments may go unpublished because of confidentiality obligations, vendor-implemented and vendor-locked systems, weak links between government IT units and academic venues, or simple lack of incentive to publish

operational work. Public institutions operate under stricter security, procurement, auditability and compliance policies, and often depend on legacy systems that do not yield the rich observability data many private sector studies assume. These conditions raise the cost of experimentation and demand explainable, auditable, human-supervised deployment support. Techniques effective in high-velocity commercial cloud settings may therefore not transfer to public service platforms without modification for governance needs, risk appetite and infrastructure constraints. Related public administration research confirms that AI adoption is conditioned by organisational and governance tensions and that legacy systems remain a principal obstacle to digital transformation [13], [14].

Why does the gap persist? Three explanations are proposed. They are not coded from the included studies; they are inferred by reading the review findings against the public administration literature, and Table 9 lists the sources supporting each. The first is funding structure: applied deployment research requires sustained access to production pipelines, and public bodies rarely have discretionary budgets for experimental infrastructure, while research funders reward novel algorithms over unglamorous operational studies [13]. The second is infrastructural: legacy estates generate sparse, inconsistent telemetry, so the labelled datasets on which supervised deployment models depend often do not exist in government settings, and confidentiality rules prevent sharing what does exist [14]. The third is a policy and incentive barrier: procurement frameworks seldom recognise ML-assisted tooling as a procurable category, institutional review processes are slow, and civil service career structures offer little reward for publishing negative or operational results. It follows that the under-representation is structural rather than accidental, and that remedies must address funding instruments, data access mechanisms and procurement policy simultaneously; without this explanatory framing, the complete absence of public sector evidence reported under RQ4 would remain descriptive rather than actionable.

Table 9. Inferred explanations for the public sector gap and their supporting sources

Inferred explanation	Supporting public administration and adoption sources
Funding structure rewards algorithmic novelty over operational deployment studies	[13], [14]
Infrastructural telemetry poverty and confidentiality limits in legacy government estates	[15], [18], [53]
Procurement, policy and incentive barriers, including goods-oriented procurement and cautious government cloud adoption	[14], [17]

5.3. Practical recommendations

Based on these findings, we provide tailored recommendations for the stakeholder groups who shape deployment practice. For researchers: priority should shift from theoretical or retrospective studies toward empirical work grounded in real production environments. Longitudinal case studies that track model performance over time would illuminate concept drift, maintenance burden and operational decay. Shared datasets and standardised evaluation benchmarks specific to deployment tasks are needed for cumulative knowledge. Attention should turn to under-explored pipeline phases such as canary analysis, rollback logic and post-deployment monitoring, where ML could add substantial value but remains under-studied. Crucially, researchers must confront the marked absence of public sector contexts by designing studies that engage government IT systems, legacy infrastructure and regulatory realities. Interpretability and human-ML collaboration should be treated as core design requirements, and all empirical reporting should include statistical significance measures and confidence intervals.

For practitioners: adoption should begin pragmatically. Teams are advised to start with simple, interpretable baselines such as rule-based thresholds or statistical control charts before investing in complex models, ensuring that added sophistication is justified by measurable gains. Published performance figures should be read with caution, since their relevance depends on deployment context, data quality and architecture. The operational

realities of production ML, including monitoring overhead, retraining cycles and integration with observability stacks, must be budgeted for. Given the stakes, human oversight should be retained for critical rollouts, particularly where failure could affect user safety, data integrity or service continuity. Regular retraining on recent deployment data should be built into workflows to counter concept drift. Comprehensive logging of model inputs, predictions and human overrides is essential for debugging, auditing and institutional trust.

For tool builders: the focus must be usability, transparency and operational sustainability. Tools should integrate smoothly with widely used CI/CD platforms such as Jenkins, GitLab CI and GitHub Actions so adoption does not require workflow upheaval. Interpretability should be designed into the interface rather than added later. Clear documentation of a model's limitations, such as sensitivity to data shifts or behaviour under rare conditions, helps users set realistic expectations. Because deployment environments change quickly, tools must support rapid retraining, versioning and model rollback, and built-in monitoring and alerting should detect degradation early and trigger human review before failures occur. This recommendation aligns with recent studies on optimising CI/CD workflows and AI-enhanced DevOps processes [23], [25].

For policymakers, especially in the public sector: this review underscores an urgent opportunity to modernise government software practice through targeted support for ML-augmented deployment. Funding should be directed to applied research addressing the specific constraints of public institutions, including legacy systems, strict procurement rules and data sovereignty requirements. Secure, privacy-preserving mechanisms for sharing anonymised deployment data across agencies could relieve the data scarcity that currently hampers public sector innovation. Clear policy guidance is needed on the responsible use of ML in deployment decisions, covering risk thresholds, human oversight and success metrics that extend beyond uptime to equity, accessibility and policy compliance. Where possible, public procurement should explicitly require interpretability, auditability and documentation from vendors, so that deployed systems remain transparent, contestable and aligned with public accountability standards.

For African public institutions, a practical starting point is a minimal deployment telemetry set that supports decision models without touching citizens' personal data:

build status, release timestamp, error rate, latency, rollback status, incident ticket identifiers and manual override logs. Collected at the pipeline and platform level, these signals are operational rather than personal, which simplifies lawful processing; where request-level traces are needed, aggregation, pseudonymisation and short retention windows keep collection within data protection obligations [53]. Ethical and safe collection further requires explicit data governance arrangements: a named controller within the institution, documented purposes limited to deployment assurance, access controls audited under the institution's cybersecurity obligations, and procurement clauses that oblige vendors to expose deployment telemetry to the contracting authority rather than locking it away.

These arrangements sit within an increasingly concrete African legal landscape. South Africa's Protection of Personal Information Act (POPIA), fully in force since 2021, conditions how public bodies process and share operational data that can identify individuals [53]; Kenya's Data Protection Act 2019, Nigeria's Data Protection Act 2023 and Ghana's Data Protection Act 2012 (Act 843) impose comparable duties. Public procurement rules and national cybersecurity directives add further obligations on hosting, incident reporting and cross-border data flows, and cautious government cloud adoption shapes what deployment infrastructure is even available [17]. Cybersecurity statutes add a further layer: South Africa's Cybercrimes Act 19 of 2020 and Kenya's Computer Misuse and Cybercrimes Act 2018 impose system-security and incident-handling duties on public bodies, while procurement statutes such as Ghana's Public Procurement Act 663 of 2003 govern how deployment tooling, hosting and telemetry access can be contracted from vendors. Deployment telemetry programmes designed within these constraints from the outset are far more likely to survive audit and scale beyond pilots.

5.4. Future research directions: a staged agenda

In light of the gaps identified under RQ5, the research directions are organised here as a staged roadmap rather than an undifferentiated list, so that researchers, institutions and funders can see what is achievable now and what requires longer horizons. Short term (one to two years). The immediate priorities are pilot deployments of explainable ML in selected government projects, using post hoc explanation techniques [48] and human-in-the-loop gating so that every automated recommendation remains auditable;

the development of lightweight models, favouring tree ensembles and distilled or quantised variants that run on the modest, sometimes offline infrastructure typical of resource-constrained institutions; and hybrid designs that layer ML scoring over existing rule-based deployment checks, so that the rules provide a verifiable safety floor while the model adds predictive refinement. Uncertainty quantification, in which models report their own confidence and defer low-confidence cases to human review, belongs in this horizon as well.

Medium term (two to four years). The field should build public-private and public-academic partnerships for ML deployment research, pairing government pipelines with university and industry expertise under data protection agreements; establish shared benchmark datasets and evaluation protocols for deployment tasks, ideally including anonymised public sector telemetry; investigate transfer learning so that models trained in one project or organisation can be adapted cheaply to another, reducing the data burden on individual institutions; and treat deployment as a sequential decision problem for safe reinforcement learning, with reward functions and exploration strategies constrained for safety [31].

Long term (five years and beyond). The destination is context-aware deployment frameworks tailored to African governance structures, in which procurement constraints, audit obligations, legacy interoperability and service equity metrics are first-class design inputs rather than afterthoughts. Achieving this requires a shift from correlation to causal reasoning about deployment failure, end-to-end optimisation across the whole pipeline instead of isolated stages, systematic study of robustness against adversarial manipulation of ML safety checks, and, most importantly, in-depth empirical case studies of ML adoption inside African and other resource-constrained public institutions, so that technical solutions are fitted to local governance systems, infrastructures and public service missions.

The quality profile of the corpus also tempers the confidence that can be placed in its findings, and by extension in the agenda derived from them. With a mean quality score of 5.5 of 12 and 14 of 33 studies rated at high risk of bias, mainly through absent baselines, unreported validation designs and closed data, the quantitative patterns reported in Section 3 should be read as indicative rather than settled. The agenda is constructed to

be robust to this uncertainty: its short- and medium-term elements target precisely the practices whose absence produces the risk, namely credible baselines, statistical testing, open datasets and shared benchmarks, so that a stronger evidence base is a direct product of pursuing the agenda itself. A simple sensitivity check reinforces this: excluding the 14 high-risk studies leaves 19, among which the sectoral absence is unchanged (0 public sector studies) and tree-based dominance persists (12 of 19), so the review's central claims do not rest on its weakest evidence.

5.5. Limitations of this review

This review has several limitations. The search covered major digital libraries but not specialty venues or grey literature. The query centred on deployment terminology and may have missed pertinent work expressed differently. Papers on MLOps, the deployment of ML models themselves, were excluded to preserve focus on general software deployment, and that demarcation is admittedly a judgement call. Publication bias may under-represent negative results. The field is dynamic, and publications since the search may fill some named gaps. Finally, sectoral classification of some papers as public or private may contain errors. Notwithstanding these shortcomings, the review offers a faithful overview of the discipline and identifies its key trends and gaps.

6. CONCLUSION

This paper is a systematic review and an agenda-setting study. Its purpose has been to map, quantitatively and critically, how machine learning is applied to software deployment, and to establish where the evidence does and does not reach. The evidence base of 33 peer-reviewed studies from 2018 to 2025 shows a field with genuine promise but real immaturity: models can estimate deployment success and add value across diverse deployment activities, yet evaluation practice, operational integration, explainability and real-world adoption all remain problematic. Three findings anchor the agenda. Within the peer-reviewed ML-for-deployment corpus reviewed here, no study directly addresses a public sector deployment context and none reports evidence from an African institution; effort clusters on autoscaling and build prediction with tree-based models dominant; and statistical rigour is nearly absent, with only one study reporting significance tests or confidence intervals. These are statements about the selected academic corpus: relevant industry or government practice may exist without having

been published, and the absence of evidence should not be read as evidence that no such systems operate in practice. The corpus is also high-risk and heterogeneous, with 14 of 33 included studies rated at high risk of bias, which limits the confidence that can be placed in the agenda derived from it.

The under-representation of the public sector remains a significant research gap given the importance of government IT systems and their distinctive constraints in data governance, legacy environments and regulatory adherence, and Section 5.2 has argued that its causes are structural, spanning funding, infrastructure and policy. Closing it requires targeted empirical studies conducted inside African public institutions, and it requires the field to adopt statistical testing, open datasets and shared benchmarks as preconditions for cumulative progress, with longitudinal validation in production environments rather than one-off laboratory evaluations. A note on scope: this study identifies the gap and proposes the agenda; the agenda is derived from synthesis and is not itself validated, and no deployment framework is developed or evaluated here. The design, implementation and empirical assessment of ML-assisted deployment support tailored to African public institutions are reserved for follow-up work, for which the present review provides the evidential foundation. The immediate next empirical step is concrete: pilot ML-assisted deployment monitoring, built on the minimal telemetry set of Section 5.3, in one African public digital service, and report the results whatever they show.

ACKNOWLEDGMENTS

The authors would like to express their sincere gratitude to Dr. Filistea Naude, the Department of Computer Science Librarian at the University of South Africa (UNISA), for her valuable assistance in conducting the extensive literature search and initial retrieval of studies from major academic databases. Her contribution significantly enhanced the quality and depth of this research. The authors also acknowledge the use of Napkin AI in generating several of the illustrative images featured in this article. The tool played a helpful role in producing clear figures that complement the paper's explanations.

REFERENCES

[1] H. Krasner, "The cost of poor software quality in the US: a 2020 report," Consortium for Information and Software Quality (CISQ), 2021. [Online]. Available: https://www.it-cisq.org/the-cost-of-poor-software-quality-in-the-us-a-2020-report/. Accessed: Sep. 15, 2025.

[2] M. Shahin, M. A. Babar, and L. Zhu, "Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices," *IEEE Access*, vol. 5, pp. 3909-3943, 2017, doi: 10.1109/ACCESS.2017.2685629.

[3] J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Upper Saddle River, NJ, USA: Addison-Wesley, 2010.

[4] P. Narang and P. Mittal, "Performance assessment of traditional software development methodologies and DevOps automation culture," *Eng. Technol. Appl. Sci. Res.*, vol. 12, no. 6, pp. 9726-9731, 2022, doi: 10.48084/etasr.5315.

[5] G. Kim, P. Debois, J. Willis, and J. Humble, *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. Portland, OR, USA: IT Revolution Press, 2016.

[6] H. Kheder, "An innovative approach to enhancing software development efficiency through agile methodologies," *Kufa J. Eng.*, vol. 16, no. 2, pp. 328-343, 2025, doi: 10.30572/2018/kje/160220.

[7] H. Qian, "Software development efficiency through automated deployment systems," *J. Sustain. Policy Pract.*, vol. 1, no. 3, pp. 99-109, Sep. 2025, doi: 10.71222/pzvfqm21.

[8] R. Paleyes, R.-G. Urma, and N. D. Lawrence, "Challenges in deploying machine learning: a survey of case studies," *ACM Comput. Surv.*, vol. 55, no. 6, pp. 1-29, 2022, doi: 10.1145/3533378.

[9] M. Mannan, S. Mustafa, and A. Hussain, "A maturity level framework for practicing machine learning operations in CI/CD for software deployment," *J. Indep. Stud. Res. Comput.*, vol. 22, no. 1, 2024, doi: 10.31645/jisrc.24.22.1.4.

[10] U. K. Durrani et al., "A decade of progress: a systematic literature review on the integration of AI in software engineering phases and activities (2013-2023)," *IEEE Access*, vol. 12, pp. 171185-171204, 2024, doi: 10.1109/ACCESS.2024.3488904.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

[11] S. Shafiq, A. Mashkoor, C. Mayr-Dorn, and A. Egyed, "A literature review of using machine learning in software development life cycle stages," *IEEE Access*, vol. 9, pp. 140896-140920, 2021, doi: 10.1109/ACCESS.2021.3119746.

[12] P. Kumar, M. Nadeem, and M. Shameem, "Machine learning based predictive modeling to effectively implement DevOps practices in software organizations," *Autom. Softw. Eng.*, vol. 30, no. 2, Art. no. 21, 2023, doi: 10.1007/s10515-023-00388-8.

[13] R. Madan and M. Ashok, "AI adoption and diffusion in public administration: a systematic literature review and future research agenda," *Gov. Inf. Q.*, vol. 40, no. 1, Art. no. 101774, Jan. 2023, doi: 10.1016/j.giq.2022.101774.

[14] Z. Irani, R. M. Abril, V. Weerakkody, A. Omar, and U. Sivarajah, "The impact of legacy systems on digital transformation in European public administration: lesson learned from a multi case analysis," *Gov. Inf. Q.*, vol. 40, no. 1, Art. no. 101784, Jan. 2023, doi: 10.1016/j.giq.2022.101784.

[15] T. Mawela, N. M. Ochara, and H. Twinomurinzi, "E-government implementation: a reflection on South African municipalities," *South Afr. Comput. J.*, vol. 29, no. 1, pp. 147-171, 2017, doi: 10.18489/sacj.v29i1.444.

[16]United Nations Department of Economic and Social Affairs, *E-Government Survey 2024: Accelerating Digital Transformation for Sustainable Development*. New York, NY, USA: United Nations, 2024. [Online]. Available: https://publicadministration.un.org/egovkb. Accessed: Sep. 15, 2025.

[17] F. Mohammed, O. Ibrahim, and N. Ithnin, "Factors influencing cloud computing adoption for e-government implementation in developing countries: instrument development," *J. Syst. Inf. Technol.*, vol. 18, no. 3, pp. 297-327, 2016, doi: 10.1108/JSIT-01-2016-0001.

[18] S. M. Mutula and J. Mostert, "Challenges and opportunities of e-government in South Africa," *Electron. Libr.*, vol. 28, no. 1, pp. 38-53, 2010, doi: 10.1108/02640471011023360.

[19] S. Gupta and A. Chug, "An optimized extreme learning machine algorithm for improving software maintainability prediction," in *Proc. 11th Int. Conf. Cloud Comput. Data Sci. Eng. (Confluence)*, Noida, India, 2021, pp. 829-836, doi: 10.1109/Confluence51648.2021.9377196.

[20] Y. N. Soe, P. I. Santosa, and R. Hartanto, "Software defect prediction using random forest algorithm," in *Proc. 12th South East Asian Tech. Univ. Consortium (SEATUC)*, Yogyakarta, Indonesia, 2018, pp. 1-5, doi: 10.1109/SEATUC.2018.8788881.

[21] V. Kumar, P. Kumari, A. Chatterjee, and D. P. Mohapatra, "Software fault prediction using random forests," *Smart Innov. Syst. Technol.*, vol. 194, pp. 95-103, 2019, doi: 10.1007/978-981-15-5971-6_10.

[22] A. Aggarwal, K. S. Dhindsa, and P. K. Suri, "Usage patterns and implementation of random forest methods for software risk and bug predictions," *Int. J. Innov. Technol. Explor. Eng.*, vol. 8, no. 9S, pp. 927-932, 2019, doi: 10.35940/ijitee.i1150.0789s19.

[23] S. R. Dileepkumar and J. Mathew, "Optimizing continuous integration and continuous deployment pipelines with machine learning: enhancing performance and predicting failures," *Adv. Sci. Technol. Res. J.*, vol. 19, no. 3, pp. 108-120, 2025, doi: 10.12913/22998624/197406.

[24] N. Charankar, "Microservices and API deployment optimization using AI," *Int. J. Recent Innov. Trends Comput. Commun.*, vol. 11, no. 11, pp. 1090-1095, 2023, doi: 10.17762/ijritcc.v11i11.10618.

[25] S. Ali and D. Puri, "Optimizing DevOps methodologies with the integration of artificial intelligence," in *Proc. 3rd Int. Conf. Innov. Technol. (INOCON)*, Bangalore, India, 2024, pp. 1-5, doi: 10.1109/INOCON60754.2024.10511490.

[26] D. Kreuzberger, N. Kuhl, and S. Hirschl, "Machine learning operations (MLOps): overview, definition, and architecture," *IEEE Access*, vol. 11, pp. 31866-31879, 2023, doi: 10.1109/ACCESS.2023.3262138.

[27] H. Abubakar, M. S. Obaidat, A. Gupta, P. Bhattacharya, and S. Tanwar, "Interplay of machine learning and software engineering for quality estimations," in *Proc. Int. Conf. Commun. Comput. Cybersecur. Informat. (CCCI)*, Sharjah, United Arab Emirates, 2020, pp. 1-6, doi: 10.1109/CCCI49893.2020.9256507.

[28] O. Borges, M. Lima, J. Couto, B. Gadelha, T. Conte, and R. Prikladnicki, "ML@SE: what do we know about how machine learning impact software engineering practice?," in *Proc. 17th Iberian Conf. Inf. Syst. Technol. (CISTI)*, Madrid, Spain, 2022, pp. 1-6, doi: 10.23919/CISTI54924.2022.9820309.

[29] L. Berardinelli et al., "Model driven engineering, artificial intelligence, and DevOps for software and systems engineering: a systematic mapping study of synergies and challenges," *ACM Trans. Softw. Eng. Methodol.*, early access, Art. no. 3759454, 2025, doi: 10.1145/3759454.

[30] J. R. Valdivia, J. Gamboa-Cruzado, and P. de la Cruz Velez de Villa, "Systematic literature review on machine learning and its impact on APIs deployment," *Comput. Sist.*, vol. 27, no. 4, pp. 1107-1124, 2023, doi: 10.13053/cys-27-4-4371.

[31] X. Lu, Y. Yu, and M. Pan, "Reinforcement learning-based auto-scaling algorithm for elastic cloud workflow service," *Lect. Notes Comput. Sci.*, vol. 13148, pp. 303-310, 2022, doi: 10.1007/978-3-030-96772-7_28.

[32] A. Dearle, "Software deployment, past, present and future," in *Proc. Future Softw. Eng. (FOSE '07)*, Minneapolis, MN, USA, 2007, pp. 269-284, doi: 10.1109/FOSE.2007.20.

[33] N. S. Thomas and S. Kaliraj, "An improved and optimized random forest-based approach to predict software faults," *SN Comput. Sci.*, vol. 5, 2024, doi: 10.1007/s42979-024-02764-x.

[34] D. S. Battina, "An intelligent DevOps platform research and design based on machine learning," *Int. J. Innov. Eng. Res. Technol. (IJIERT)*, vol. 6, no. 3, 2019.

[35] O. C. Oyeniran, A. O. Adewusi, A. G. Adeleke, L. A. Akwawa, and C. F. Azubuko, "AI-driven DevOps: leveraging machine learning for automated software deployment and maintenance," *Eng. Sci. Technol. J.*, vol. 4, no. 6, pp. 728-740, 2023, doi: 10.51594/estj.v4i6.1552.

[36] S. Joshi, "A review of generative AI and DevOps pipelines: CI/CD, agentic automation, MLOps integration, and LLMs," *Int. J. Innov. Res. Comput. Sci. Technol.*, vol. 13, no. 4, pp. 1-14, 2025, doi: 10.55524/ijircst.2025.13.4.1.

[37] A. Hrusto, E. Engstrom, and P. Runeson, "Towards optimization of anomaly detection in DevOps," *Inf. Softw. Technol.*, vol. 160, Art. no. 107241, 2023, doi: 10.1016/j.infsof.2023.107241.

[38] R. Ashmore, R. Calinescu, and C. Paterson, "Assuring the machine learning lifecycle: desiderata, methods, and challenges," *ACM Comput. Surv.*, vol. 54, no. 5, pp. 1-39, 2021, doi: 10.1145/3453444.

[39] H. Qiu et al., "AWARE: automate workload autoscaling with reinforcement learning in production cloud systems," in *Proc. 2023 USENIX Annu. Tech. Conf. (USENIX ATC 23)*, Boston, MA, USA, 2023.

[40] Y. Gari, D. A. Monge, E. Pacini, C. Mateos, and C. Garcia Garino, "Reinforcement learning-based application autoscaling in the cloud: a survey," *Eng. Appl. Artif. Intell.*, vol. 102, Art. no. 104288, 2021, doi: 10.1016/j.engappai.2021.104288.

[41] E. Enemosah, "Enhancing DevOps efficiency through AI-driven predictive models for continuous integration and deployment pipelines," *Int. J. Res. Publ. Rev.*, vol. 6, no. 1, pp. 871-887, 2025, doi: 10.55248/gengpi.6.0125.0229.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

[42] F. Nogueira, J. C. B. Ribeiro, M. A. Zenha-Rela, and A. Craske, "Improving La Redoute's CI/CD pipeline and DevOps processes by applying machine learning techniques," in *Proc. 11th Int. Conf. Qual. Inf. Commun. Technol. (QUATIC)*, Coimbra, Portugal, 2018, pp. 282-286, doi: 10.1109/QUATIC.2018.00050.

[43] H. Bruneliere et al., "AIDOaRt: AI-augmented automation for DevOps, a model-based framework for continuous development in cyber-physical systems," *Microprocess. Microsyst.*, vol. 94, Art. no. 104672, 2022, doi: 10.1016/j.micpro.2022.104672.

[44] T. O. Kolawole and A. Fakokunde, "Machine learning algorithms in DevOps: optimizing software development and deployment workflows with precision," *Int. J. Res. Publ. Rev.*, vol. 6, no. 1, pp. 247-264, 2025, doi: 10.55248/gengpi.6.0125.0209.

[45] B. Kitchenham, "Procedures for performing systematic reviews," Keele University, Keele, U.K., Technical Report TR/SE-0401, 2004.

[46] M. J. Page et al., "The PRISMA 2020 statement: an updated guideline for reporting systematic reviews," *BMJ*, vol. 372, Art. no. n71, 2021, doi: 10.1136/bmj.n71.

[47] N. R. Haddaway, M. J. Page, C. C. Pritchard, and L. A. McGuinness, "PRISMA2020: an R package and Shiny app for producing PRISMA 2020-compliant flow diagrams, with interactivity for optimised digital transparency and open synthesis," *Campbell Syst. Rev.*, vol. 18, no. 2, Art. no. e1230, 2022, doi: 10.1002/cl2.1230.

[48] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?': explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. (KDD)* (KDD '16), San Francisco, CA, USA, 2016, pp. 1135-1144, doi: 10.1145/2939672.2939778.

[49] H. Muthukrishnan, V. Viradia, and D. Yadav, "Unified AI and ML framework in DevSecOps practices, solving real-world problems," in *SoutheastCon 2025*, Concord, NC, USA, 2025, pp. 1250-1257, doi: 10.1109/SoutheastCon56624.2025.10971458.

[50] S. Malhotra, A. Elsayed, R. Torres, and S. Venkatraman, "Evaluate canary deployment techniques using Kubernetes, Istio, and Liquibase for cloud native enterprise applications to achieve zero downtime for continuous deployments," *IEEE Access*, vol. 12, pp. 87883-87899, 2024, doi: 10.1109/ACCESS.2024.3416087.

[51] S. Bollineni, "AI-driven automation in DevOps: explore how artificial intelligence and machine learning can enhance automation in DevOps," *J. Artif. Intell. Mach. Learn. Data Sci.*, vol. 2, no. 1, pp. 1295-1298, 2024, doi: 10.51219/jaimld/satyadeepak-bollineni/296.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

[52] N. G. Camacho, "Unlocking the potential of AI/ML in DevSecOps: effective strategies and optimal practices," *J. Artif. Intell. Gen. Sci.*, vol. 2, no. 1, pp. 91-100, 2024, doi: 10.60087/jaigs.v2i1.99.

[53] C. Staunton, K. Tschigg, and G. Sherman, "Data protection, data management, and data sharing: stakeholder perspectives on the protection of personal health information in South Africa," *PLOS ONE*, vol. 16, no. 12, Art. no. e0260341, 2021, doi: 10.1371/journal.pone.0260341.

[54] A. Gupta, "A predictive framework for managing DevOps practices using machine learning models," *Iconic Res. Eng. J.*, vol. 4, no. 9, 2021.

[55] B. Gajbhiye, A. Aggarwal, and S. Jain, "Automated security testing in DevOps environments using AI and ML," *Int. J. Res. Publ. Semin.*, vol. 15, no. 2, 2024.

[56] F. Binbeshr and M. Imam, "Comparative analysis of AI-driven security approaches in DevSecOps: challenges, solutions, and future directions," in *Proc. 29th Int. Conf. Eval. Assess. Softw. Eng. (EASE 2025)*, 2025.

[57] P. Liang, Y. Wu, Z. Xu, S. Xiao, and J. Yuan, "Enhancing security in DevOps by integrating artificial intelligence and machine learning," *J. Theory Pract. Eng. Sci.*, vol. 4, no. 2, 2024.

[58] G. Kambala, "Intelligent software agents for continuous delivery: leveraging AI and machine learning for fully automated DevOps pipelines," *Iconic Res. Eng. J.*, 2024.

[59] M. Bagherzadeh, N. Kahani, and L. Briand, "Reinforcement learning for test case prioritization," *IEEE Trans. Softw. Eng.*, vol. 48, no. 8, pp. 2836-2856, 2022, doi: 10.1109/TSE.2021.3070549.