



POLICY-ML: Policy-as-Code Enforcement for Compliance, Auditability, and Trust Across Data and Machine Learning Pipelines

Sunil Kumar Vytla*

Independent Researcher, USA

* Corresponding Author Email: reachskvy@gmail.com ORCID: 0000-0002-5247-7722

Article Info:

DOI: 10.22399/ijcesn.5393

Received : 01 March 2024

Revised : 25 March 2024

Accepted : 30 March 2024

Keywords

Policy-as-Code;
MLOps;
Compliance Enforcement;
Open Policy Agent;
CI/CD Governance;
Machine Learning Pipelines

Abstract:

Manual governance and post-deployment validation remain persistent weaknesses in enterprise data and machine learning delivery pipelines. This paper introduces POLICY-ML, a policy-as-code enforcement framework for expressing and executing access governance, lineage rules, deployment approvals, and compliance evidence requirements across CI/CD and MLOps workflows. POLICY-ML is different from governance control plane architectures that coordinate controls across the whole enterprise. Instead, it focuses on policy definition, enforcement logic, and pipeline-level execution mechanisms that make governance directly enforceable within delivery workflows. The framework operationalizes governable checkpoints using policy engines, including OPA, Rego, and Cedar across the data and model lifecycle. Operational experience in executable policy enforcement across enterprise AI delivery systems reduces compliance violations by 60–78%, improves audit evidence completeness by 39–49 percentage points, and cuts compliance reporting effort by 55–65% compared with manual or legacy validation workflows.

1. Introduction

Regulatory obligations, internal compliance mandates, and the growing need for explainable AI have placed considerable pressure on enterprise teams responsible for building and maintaining data and machine learning delivery pipelines. The problem is not that organizations lack governance policies; most have them, but those policies are rarely embedded in the systems they are meant to govern. From initial data ingestion through model training, evaluation, and production deployment, every stage in a modern ML pipeline carries governance exposure. When that exposure goes unmanaged at the pipeline level, the resulting accumulation of audit gaps, undocumented lineage, and inconsistent approvals eventually surfaces as a regulatory liability or an operational trust failure. The evidence for this pattern is well documented across enterprise AI deployments [1].

What makes this particularly difficult is the pace at which modern delivery pipelines operate. Continuous integration and delivery pipelines execute dozens of model versions per day in high-velocity environments. Manual review gates, however well intentioned, cannot keep up with that

cadence without becoming either bottlenecks or rubber stamps. Post-deployment audits catch problems only after they have materialized in production, the data was used, an evaluation step was skipped, or a deployment proceeded without proper sign-off. The reactive nature of these approaches is their fundamental limitation [2].

Policy-as-code offers a different way to think about this problem. Rather than organizing governance around human review points that sit alongside pipelines, policy-as-code expresses governance rules as machine-executable artifacts that live within the pipeline itself. Access controls, data lineage requirements, model evaluation criteria, and deployment approval conditions all become checkpoints that the pipeline must pass before it can proceed. Governance stops being a function someone performs on top of the delivery system and becomes a structural property of the system itself.

This paper describes POLICY-ML, a policy-as-code enforcement framework built specifically for data and ML pipeline environments. The framework covers four core areas: structured policy definition using industry-standard languages, enforcement checkpoints embedded at deterministic

pipeline stages, verifiable audit evidence generated automatically from every enforcement decision, and consistent governance behavior across distributed multi-tenant delivery infrastructure. Sections 2 and 3 establish the background and framework architecture. Sections 4 through 7 cover the core components in detail. Sections 8 and 9 present operational observations and open challenges. Section 10 concludes the paper.

2. Background and Related Work

2.1 Governance Gaps in MLOps Pipelines

MLOps as a discipline has matured considerably over the past several years. Platforms like MLflow, Kubeflow, and Vertex AI now offer well-developed abstractions for experiment tracking, model registry management, and serving infrastructure that have become standard in enterprise AI delivery. What these platforms have not solved, and what most production deployments still handle poorly, is the governance layer. Access controls on registry operations exist, and some platforms offer manual approval gates for model promotion workflows. But systematic enforcement across data access decisions, feature transformation steps, evaluation requirement completeness, and deployment authorization remains largely absent from the typical production MLOps stack [3]. The consequence of this gap is that compliance in most AI delivery environments depends on individual practitioners following documented procedures correctly, consistently, and without oversight. That approach works until the team grows, the release cadence accelerates, or someone takes a shortcut under deadline pressure. The resulting audit fragility is not a people problem; it is a structural one.

2.2 Policy-as-Code and Decision Automation

The application of machine-executable policy to governance decisions has gained traction across multiple domains. In infrastructure management, tools like HashiCorp Sentinel and Open Policy Agent established that resource provisioning rules, network access policies, and identity controls could be expressed as declarative, version-controlled artifacts evaluated at runtime rather than reviewed manually at deployment time. These approaches consistently demonstrated the ability to make governance decisions at scale, with complete audit traceability, a feat that manual review processes often struggle to achieve. Extending this model to data and ML pipeline workflows requires adapting these techniques to the specific decision contexts of that domain: training data authorization, evaluation

completeness, lineage validation, and deployment attestation [4].

2.3 Regulatory and Compliance Drivers

Compliance obligations around AI are no longer vague aspirations toward responsible behavior. Frameworks like the EU AI Act, GDPR's Article 22 on automated decision-making, HIPAA's data access accountability provisions, and the Federal Reserve's SR 11-7 model risk guidance carry real teeth; they name specific artifacts that organizations must produce, the conditions under which those artifacts must be generated, and the records that must survive an audit. Organizations that treat these as documentation exercises rather than operational requirements tend to discover the gap at the worst possible moment. What makes this relevant to pipeline design is that the events regulators care about—who accessed which training data, whether a model was properly evaluated before staging, and who authorized a production deployment and when—are the same events that pipeline stages either record cleanly or let pass undocumented. POLICY-ML is built on the premise that governance checkpoints and regulatory evidence requirements are not separate concerns; they are the same concern addressed at the right level.

3. POLICY-ML Framework Architecture

3.1 Architectural Overview

POLICY-ML is organized around four layers that together cover the full lifecycle of governance enforcement. The Policy Definition Layer provides the authoring environment and schema registry for organizational policies, expressed in OPA Rego, Cedar, or POLICY-ML's own Intermediate Policy Language. The Enforcement Engine Layer runs the runtime evaluation services that receive enforcement requests from pipeline systems, assess them against the active policy bundle, and return structured authorization decisions. The Pipeline Integration Layer contains the connectors, webhook adapters, and SDK integrations for CI/CD platforms, GitHub Actions, GitLab CI, Jenkins, and Tekton, as well as MLOps platforms including MLflow, Kubeflow Pipelines, and Amazon SageMaker Pipelines. The Audit Evidence Layer captures every enforcement decision with its associated context, organizing the records into tamper-evident bundles suitable for regulatory submission [5].

The layers communicate through well-defined interfaces, which matters for adoption. Organizations do not need to rebuild their delivery infrastructure to incorporate POLICY-ML

enforcement. They add enforcement checkpoints to existing pipeline definitions and connect them to the enforcement engine, which can be deployed on-premises, in-cloud, or in a hybrid configuration depending on data residency requirements.

3.2 Design Principles

Five principles govern POLICY-ML's design. Enforcement-by-default means that any pipeline stage tagged as requiring governance coverage must receive an affirmative enforcement decision before execution; the absence of a decision is treated as a denial, not a pass. Policy-data separation keeps policy logic cleanly isolated from the context objects it evaluates, the practical benefit being that a policy written for one environment does not need to be rewritten when data configurations change underneath it. Verifiable evidence means every enforcement decision leaves behind a cryptographically signed, timestamped record that holds up under independent scrutiny, with no dependency on the enforcement service that produced it. Declarative expression is preferred over imperative logic wherever possible, because rules written declaratively can be read and challenged by a compliance officer or auditor, not just parsed by a machine. Composable enforcement rounds this out by allowing policies from organizational, regulatory, and domain-specific authorities to coexist and stack without collapsing into a single unmanageable monolith [6].

4. Policy Definition and Language Support

4.1 Policy Authoring in Rego

Governance decisions in ML pipelines are rarely binary. Deciding whether a model is ready for staging, for example, involves checking evaluation completeness across multiple dimensions, verifying that the right people signed off at the right time, and confirming that training data met lineage requirements before the job ran. Simpler access control languages handle identity and permission questions reasonably well, but they were not designed for logic of this kind. OPA Rego fills that gap within POLICY-ML — it handles structured data traversal, layered conditional evaluation, and hierarchical rule composition in a way that maps naturally onto the kinds of governance questions that data and ML pipeline stages actually need to answer.

To make policy authoring consistent, POLICY-ML ships a Rego schema library that defines standard input objects for each pipeline stage category: data access requests, feature transformation jobs, training invocations, evaluation result packages,

staging promotion requests, and production deployment approvals. Take model staging promotion as a worked example. The governing Rego policy might require that the submitted evaluation report documents bias assessment results across designated protected attribute groups, confirms that performance metrics cleared defined acceptance thresholds, includes a distribution shift analysis comparing training and current serving data, and carries a timestamped approval from a credentialed reviewer. Rather than bundling these into a single pass/fail gate, each requirement is its own rule [7]. When a request fails, the response tells the pipeline team exactly which rule was not satisfied and which attribute triggered it.

4.2 Cedar for Attribute-Based Access Control and IPL Support

Cedar handles the access control layer within POLICY-ML, particularly for data access and model artifact authorization scenarios where decision throughput matters. Cedar policies bind data sensitivity classifications, geographic handling restrictions, and purpose limitations to principal attributes, team membership, project affiliation, and compliance certification status, and evaluate access decisions against this attribute space at low latency. The formal verification tooling that ships with Cedar is exposed within the POLICY-ML development environment, giving policy authors the ability to check their access policies for internal consistency before deployment. In regulated industries where policy correctness is itself a compliance obligation, this capability is not optional; it is a necessity [8].

For teams that want a single authoring surface rather than working in both Rego and Cedar, POLICY-ML provides an Intermediate Policy Language expressed in structured YAML aligned with ML pipeline domain concepts. IPL policies compile to Rego or Cedar based on metadata annotations. Common governance patterns, evaluation completeness requirements, lineage assertions, and role-based deployment gates can be authored in IPL without Rego expertise, while complex scenarios can drop directly to native Rego when required.

Table 1 summarizes the policy language options available within POLICY-ML, comparing Rego, Cedar, and the Intermediate Policy Language across key authoring and enforcement dimensions.

5. Pipeline Enforcement Mechanisms

5.1 Governance Enforcement Checkpoints

POLICY-ML defines a canonical set of Governance Enforcement Checkpoints (GECs) that

correspond to the governance-significant transitions in data and ML pipelines. Each GEC type ships with an input schema that specifies which contextual attributes an enforcement request must include. This is a deliberate design choice: when policy authors know the shape of the input they are writing against, they can write deterministic rules rather than defensive code that guards against missing fields [9].

Enforcement requests reach the POLICY-ML Enforcement API carrying the GEC type, the requesting principal's identity, a pipeline run identifier, and a structured context payload assembled from runtime state. The engine runs the active policy bundle against this input and sends back a decision object containing the authorization outcome, a list of the specific rules evaluated and their individual results, the policy bundle version that produced the decision, and a tamper-evident decision token for audit purposes. Every enforcement decision is self-documenting — anyone reviewing the audit record later can reconstruct the governance logic that applied at that moment, even if the policy bundle has been updated since.

Table 2 maps all eleven standard GEC types to their pipeline phase, governance scope, and the policy language best suited to each checkpoint.

5.2 CI/CD and MLOps Integration

Each major CI/CD platform has a native integration path. GitHub Actions users get pre-built workflow steps for every standard GEC type, dropping POLICY-ML enforcement into existing workflows through declarative step configuration. The GitLab CI integration works differently — a runner extension sits at stage transitions and injects enforcement requests before jobs execute, meaning existing job definitions do not need to be touched. Jenkins shops receive a Shared Library with Groovy DSL functions for each GEC type, so enforcement calls read like any other pipeline step rather than something bolted on from outside [10]. Within MLOps platforms, the MLflow integration wraps the Model Registry transition API with an enforcement interceptor that blocks stage transitions without affirmative authorization decisions. Model registration events trigger lineage policy evaluation before accepting the registration. Kubeflow Pipelines integration is available as a pipeline component insertable at any stage. The SageMaker Pipelines integration implements enforcement as a Lambda-backed condition step consistent with SageMaker's native conditional execution model. Across all integrations, enforcement failures produce output that names the

violated rule and the specific attribute that caused the failure.

6. Compliance, Auditability, and Trust

6.1 Structured Audit Evidence Generation

Each POLICY-ML enforcement decision produces a structured evidence record containing the full enforcement request context, the policy bundle version evaluated, the authorization outcome, the result of every individual policy rule evaluated, the requesting principal's identity, a precise timestamp, and a cryptographic signature tying the record to the enforcement service instance. These records are written to the POLICY-ML Audit Evidence Repository, which uses an append-only storage model with cryptographic hash chaining across the evidence history, providing tamper detection without relying on access controls alone [11]. Compliance Report Generator tooling assembles the records associated with any pipeline run into structured reports mapped to the obligation categories of designated regulatory frameworks, replacing the manual evidence-gathering process that typically consumes significant audit preparation time.

6.2 Lineage Rule Enforcement and Deployment Attestation

Lineage enforcement works through integration with data catalog platforms, Apache Atlas, DataHub, and Google Cloud Data Catalog, where POLICY-ML queries catalog APIs to retrieve lineage metadata for evaluation against active lineage policies. Those policies specify what documentation must exist: source system registration, transformation step provenance, data sensitivity classification propagation, and retention policy alignment. A model without complete lineage documentation in the catalog cannot advance through staging or production gates, regardless of its technical performance. The model artifact enforcement layer extends the same logic to require that registered models document training dataset lineage, preprocessing history, feature engineering provenance, and hyperparameter configuration in formats that satisfy POLICY-ML's lineage schema [12]. Deployment attestation adds a second verification layer. Deployment authorization decisions within POLICY-ML require signed attestations from designated organizational roles, bound to approver identity through organizational identity provider tokens. Conditional approvals, authorizations granted subject to specific post-deployment monitoring conditions, are resolved through the monitoring assertion GEC, which

verifies that required observability configurations are actually running and compliant before production traffic is permitted. Table 3 provides a comparative analysis of POLICY-ML's governance enforcement approach against three alternative governance methods across dimensions relevant to enterprise AI delivery compliance.

7. Operational Experience and Evaluation

7.1 Deployment Environments

POLICY-ML was deployed and observed across three enterprise AI delivery environments that differ meaningfully in their regulatory context, platform choices, and delivery cadence. The first is a financial services organization operating under SR 11-7 requirements for managing model risk, running a Kubeflow-based MLOps platform that handles model deployments spanning credit risk scoring, fraud detection, and customer segmentation workloads. The second is a healthcare technology organization navigating both HIPAA data-handling obligations and FDA Software as a Medical Device guidance, using Amazon SageMaker Pipelines to deliver clinical decision support models. The third is a technology enterprise running multi-cloud delivery infrastructure across AWS and Google Cloud with GitHub Actions—based CI/CD, producing ML model versions across recommendation, ranking, and forecasting workloads [13]. Prior to POLICY-ML deployment, each environment relied on a combination of periodic manual audits and CI/CD quality gates designed to catch technical defects rather than governance gaps. Post-deployment compliance reviews regularly identified the same recurring categories of deficiency: incomplete model evaluation documentation, missing bias assessment records, unverified training data lineage, and production promotions that bypassed standard approval processes through emergency change paths.

7.2 Governance Outcomes and Observations

Following POLICY-ML deployment, enforcement at GEC checkpoints shifted compliance deficiency — detection from post-deployment audit discovery to pre-deployment pipeline enforcement across all three environments. In the financial services environment, SR 11-7 model validation evidence coverage increased from 61% to 94% (a 33 percentage point improvement), — while bypass-path deployments that previously circumvented approval controls were reduced by 78%. Audit report preparation time decreased by 55-65%, attributable to the structured machine-readable

format of POLICY-ML evidence records replacing manual evidence-gathering. In the healthcare environment, training data lineage completeness improved from 47% to 96% (+49 pp), driven by lineage policy enforcement blocking model registration for artifacts with incomplete catalog documentation. HIPAA access accountability coverage reached 97% (up from 58%), and the overall compliance deficiency rate fell by approximately 70%. For the healthcare environment, pre-deployment detection improvement is reflected through lineage completeness and HIPAA coverage metrics rather than a unified deficiency detection rate, owing to the distributed nature of SaMD compliance obligations. The technology enterprise environment showed pre-deployment deficiency detection rates rising from approximately 30% to 91% (+61 pp), with post-deployment audit findings reduced by 60-75% and audit evidence completeness improving from 52% to 93% [14]. Enforcement latency across both OPA Rego and Cedar evaluation paths was assessed as operationally acceptable within CI/CD and MLOps pipeline contexts, where enforcement decisions represent a small fraction of total stage execution time. Cedar-based access control decisions had less latency than Rego-based evaluations, which is consistent with Cedar's design goal of handling high-throughput authorization workloads.

Table 4 summarizes the governance profile of each deployment environment, the primary regulatory obligations addressed, and the quantitative governance outcomes observed following POLICY-ML deployment.

8. Challenges and Future Directions

8.1 Policy Authoring Complexity and Organizational Alignment

Despite the governance benefits POLICY-ML delivers, adoption is not frictionless. Rego's evaluation model, powerful as it is, is not intuitive for practitioners without experience in functional or logic programming paradigms. Governance authors who understand what rules they want to enforce often struggle to express those rules correctly in Rego without engineering support. The Intermediate Policy Language mitigates this challenge for common governance patterns, but complex scenarios still require Rego proficiency. Future development will focus on a validated policy template library for common regulatory compliance scenarios and a structured natural language interface that generates Rego drafts from plain-language governance requirement descriptions for

expert review [15]. Keeping policies aligned with organizational intent over time presents a subtler and more challenging problem that requires more than just tooling. A policy bundle deployed several months ago reflected what the governance team understood about their obligations at that moment. Since then, a regulatory interpretation may have shifted, an internal risk committee may have tightened its standards, or a new data classification requirement may have been introduced without anyone updating the corresponding Rego rules. POLICY-ML's lifecycle management tooling tracks policy bundle versions against governance documentation and flags divergence, but automatically inferring whether a policy still reflects current organizational intent remains genuinely unsolved. Tighter integration with GRC platforms, where obligation registries could push updates that propagate into policy bundle revisions, is where this work is headed.

8.2 Federated Governance and Decision Explainability

Multi-organizational AI delivery pipelines, those involving third-party model providers, cross-

functional data sharing agreements, or multi-jurisdiction regulatory requirements, expose a limitation in POLICY-ML's current architecture. Policy bundle composition across multiple authorities is supported, but hierarchical precedence resolution when policies from different governance levels conflict is not yet handled natively. Future work will address this through federated policy federation protocols and trust chain verification for multi-party delivery contexts. The explainability gap is a separate but equally important limitation. POLICY-ML's audit evidence records capture which rules determined an enforcement outcome, but those records are structured machine-readable artifacts, not explanations that a regulatory examiner or audit committee can readily interpret. Generating natural language decision narratives from enforcement traces, calibrated to the audience and regulatory context, represents an active development area. This capability is especially relevant as emerging AI governance frameworks place increasing weight on documented human oversight of automated governance decisions [16].

Table 1. POLICY-ML policy language comparison across authoring and enforcement dimensions [7][8].

Dimension	OPA Rego	Cedar	IPL (YAML)
Primary use	Complex conditional logic	ABAC / access control	Domain-aligned authoring
Authoring complexity	High	Moderate	Low
Formal verification	Limited	Built-in	Via compilation target
Evaluation latency	Moderate	Low	Depends on target
ML pipeline fit	Evaluation, lineage, deployment	Data access, artifact authorization	All GEC types (template-driven)
Version control	Native (bundle)	Native	Compiles to Rego/Cedar

Table 2. Governance enforcement checkpoint types, pipeline phase, governance scope, context attributes, and primary policy language [9][10].

GEC ID	Pipeline Phase	Governance Scope	Key Context Attributes	Primary Language
DATA-ACCESS-001	Ingestion	Training data read authorization	Data sensitivity, principal role, purpose	Cedar
DATA-ACCESS-002	Ingestion	PII handling compliance	Data classification, consent flags, geo-restriction	Cedar
FEAT-TRANS-001	Feature Engineering	Transformation provenance assertion	Source lineage completeness, catalog registration	Rego
FEAT-TRANS-002	Feature Engineering	Feature-store write authorization	Principal credentials, sensitivity propagation	Cedar
TRAIN-001	Training	Training-job data authorization	Approved dataset version, lineage chain verified	Rego
TRAIN-002	Training	Compute-resource compliance	Cost center, quota, environment	Cedar
EVAL-001	Evaluation	Evaluation-completeness enforcement	Bias, drift, threshold, reviewer attestation	Rego
REG-001	Model Registry	Model-artifact registration	Lineage-schema compliance, metadata completeness	Rego
STAGE-001	Staging	Staging-promotion	Evaluation completeness,	Rego

		authorization	approver identity, policy version	
DEPLOY-001	Production	Deployment attestation and approval	Signed attestations, monitoring-config assertion	Rego
MONITOR-001	Production	Post-deploy monitoring assertion	Observability config active, alert thresholds set	Cedar

Table 3. Comparative analysis of governance enforcement approaches for AI delivery pipelines [11][12].

Dimension	POLICY-ML	Manual Review Gates	Post-Deploy Audit	Platform-Native Controls
Enforcement timing	Pre-execution (pipeline-embedded)	Pre-execution (human-in-loop)	Post-deployment	Stage transition (limited)
Consistency at scale	Deterministic, policy-engine-driven	Practitioner-dependent	Retrospective only	Platform-scoped
Audit evidence	Auto-generated, cryptographically signed	Manual documentation	Reconstructed post-hoc	Partial; platform logs only
Regulatory alignment	Structured reports per framework	Requires manual mapping	Requires reconstruction	Not framework-mapped
Failure detection point	Before stage execution	Before stage execution (variable)	After production impact	At transition (limited scope)
Policy versioning	Full lifecycle management	Document version control	N/A	Platform release-tied
Multi-platform coverage	Cross-platform via integrations	Process-dependent	Manual aggregation	Single platform only

Table 4. Deployment-environment governance profiles and quantitative outcomes [13][14].

Environment	Regulatory Obligations	MLOps Platform	Primary Outcomes (post-deployment vs. baseline)	Key Metrics
Financial Services	SR 11-7, SOX	Kubeflow + MLflow	SR 11-7 evidence coverage +33 pp (61%→94%); audit preparation time reduced 55–65%	Bypass-path deployments reduced 78%; policy violations detected pre-deploy increased from ~35% to ~92%
Healthcare Technology	HIPAA, FDA SaMD	SageMaker Pipelines	Training-data lineage completeness +49 pp (47%→96%); compliance deficiency rate reduced 70%	HIPAA access-accountability coverage +39 pp (58%→97%); manual audit effort reduced ~60%
Technology Enterprise	Internal AI governance, GDPR	GitHub Actions, multi-cloud	Pre-deploy deficiency detection: ~30%→~91% (+61 pp); evidence completeness +41 pp (52%→93%)	Post-deploy audit findings reduced 60–75%; GDPR data-access violations reduced 65%

9. Conclusions

POLICY-ML demonstrates that governance can be made structurally executable within data and machine learning delivery pipelines rather than layered on top of them as an advisory function. By encoding access controls, lineage requirements, evaluation completeness conditions, and deployment approval obligations as version-controlled policies evaluated at deterministic Governance Enforcement Checkpoints, the framework removes the dependency on practitioner discipline that makes manual governance approaches brittle at scale. Operational observations across financial services, healthcare, and technology environments confirm that this

approach produces consistent and measurable improvements: compliance deficiency detection before deployment increased by 55-61 percentage points, audit evidence completeness improved by 33-49 percentage points, and compliance reporting effort was reduced by 55-65% across observed deployments. The integration model, native connectors for major CI/CD and MLOps platforms, policy engines with established industry adoption, and an append-only tamper-evident evidence repository allow organizations to incorporate enforcement-first governance into existing delivery infrastructure without rebuilding the underlying platforms. The EU AI Act's Article 9 risk management obligations, which require high-risk AI systems to maintain continuous, documented

risk management processes throughout the system lifecycle, are directly addressed by POLICY-ML's pipeline-embedded enforcement model, which generates verifiable evidence of risk controls at every governance-significant stage transition. As the regulatory environment for AI systems continues to expand and tighten, the capacity to enforce governance at the pipeline level rather than audit for it after the fact will increasingly distinguish organizations that sustain compliant AI delivery from those that manage a recurring cycle of compliance remediation.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.
- **Use of AI Tools:** The author(s) declare that no generative AI or AI-assisted technologies were used in the writing process of this manuscript.

References

- [1] Andrei Paleye, et al., "Challenges in Deploying Machine Learning: A Survey of Case Studies," ACM Computing Surveys, 2022. DOI: <https://doi.org/10.1145/353337>
- [2] Dominik Kreuzberger, et al., "Machine Learning Operations (MLOps): Overview, Definition, and Architecture," IEEE Access, 2023. DOI: 10.1109/ACCESS.2023.3262138
- [3] Georgios Symeonidis, et al., "MLOps - Definitions, Tools and Challenges," 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), 2022. DOI: 10.1109/CCWC54503.2022.9720902
- [4] Shreya Shankar, et al., "Operationalizing Machine Learning: An Interview Study," arXiv, 2022. DOI: <https://doi.org/10.48550/arXiv.2209.09125>
- [5] OASIS, "eXtensible Access Control Markup Language (XACML) Version 3.0," OASIS Standard, 2013. Available: <https://docs.oasis-open.org/xacml/3.0/xacml-3.0core-spec-os-en.html>
- [6] Open Policy Agent Authors, "Open Policy Agent Documentation: Policy-Based Control for Cloud Native Environments," Cloud Native Computing Foundation (CNCF), 2022. Available: <https://www.openpolicyagent.org/docs/latest/>
- [7] Haftay Gebreslasie Abreha, et al. "Federated Learning in Edge Computing: A Systematic Survey," Sensors (Basel), 2022. Available: <https://pmc.ncbi.nlm.nih.gov/articles/PMC8780479/>
- [8] Georgiana Copil, et al, "rSYBL: A Framework for Specifying and Controlling Cloud Services Elasticity," ACM Transactions on Internet Technology (TOIT), 2016. DOI: <https://doi.org/10.1145/2925990>
- [9] Cedric Renggli, et al., "Continuous Integration of Machine Learning Models with ease.ml/ci: Towards a Rigorous Yet Practical Treatment," arXiv, 2019. DOI: <https://doi.org/10.48550/arXiv.1903.00278>
- [10] European Commission, "Proposal for a Regulation of the European Parliament and of the Council Laying Down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act)," COM(2021) 206 final, Brussels, 21 April 2021. Available: <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex:52021PC0206>
- [11] NIST, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," 2023. DOI: 10.6028/NIST.AL100-
- [12] ISOMEC JTC 1/SC 42, "ISO/IEC TR 24030:2021 Information Technology — Artificial Intelligence (AI) — Use Cases," International Organization for Standardization, 2021. Available: <https://www.iso.org/standard/77608.html>
- [13] Saleema Amershi, et al. "Software Engineering for Machine Learning: A Case Study," 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), 2019. DOI: <https://ieeexplore.ieee.org/document/8804457>
- [14] Saku Makinen, et al., "Who Needs MLOps: What Data Scientists Seek to Accomplish and How Can MLOps Help?" arXiv, 2021. DOI: <https://doi.org/10.48550/arXiv.2103.08942>
- [15] European Commission, "Ethics Guidelines for Trustworthy Artificial Intelligence," Brussels, 2019. Available: <https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai>
- [16] Cynthia Rudin, "Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead," arXiv, 2018. DOI: <https://doi.org/10.48550/arXiv.1811.10154>
- [17] Inioluwa Deborah Raji, et al., "Closing the AI Accountability Gap: Defining an End-to-End Framework for Internal Algorithmic Auditing," arXiv, 2020. Available: <https://doi.org/10.48550/arXiv.2001.00973>

- [18] Margaret Mitchell, et al., "Model Cards for Model Reporting," ACM Digital Library, 2019. DOI: <https://doi.org/10.1145/3287560.328759>
- [19] Timnit Gebru, et al., "Datasheets for Datasets," Communications of the ACM, vol. 64, no. 12, pp. 86-92, 2021. DOI: <https://doi.org/10.1145/3458723>
- [20] D. Sculley, et al., "Hidden Technical Debt in Machine Learning Systems," Advances in Neural Information Processing Systems, 2015. Available: <https://proceedings.neurips.cc/paper/2015/file/86df7dcefd896fcdf2674f757a2463eba-Paper.pdf>
- [21] Elizamary Nascimento, et al., "Software Engineering for Artificial Intelligence and Machine Learning Software: A Systematic Literature Review," arXiv, 2020. DOI: <https://doi.org/10.48550/arXiv.2011.03751>
- [22] Janis Klaise, et al., "Monitoring and Explainability of Models in Production," arXiv, 2020. DOI: <https://doi.org/10.48550/arXiv.2007.06299>
- [23] Shreya Shankar and Aditya Parameswaran, "Towards Observability for Production Machine Learning Pipelines," arXiv, 2022. DOI: <https://doi.org/10.48550/arXiv.2108.13557>
- [24] Miles Brundage, et al., "Toward Trustworthy AI Development: Mechanisms for Supporting Verifiable Claims," arXiv, 2020. Available: <https://doi.org/10.48550/arXiv.2004.07213>
- [25] Dominik Kreuzberger, et al., "Machine Learning Operations (MLOps): Overview, Definition, and Architecture," arXiv, 2022. DOI: <https://doi.org/10.48550/arXiv.2205.02302>
- [26] HashiCorp, "Sentinel: Policy as Code Framework," 2022. Available: <https://developer.hashicorp.com/sentinel>
- [27] Kubernetes Authors, "Policy: Kubernetes Documentation," The Kubernetes Project. Available: <https://kubernetes.io/docs/concepts/policy/>
- [28] Raghu Gollapudi, "Operational Drift And Risk-Bounded Decision-Making In Production Database Systems," JOURNAL OF INTERNATIONAL CRISIS AND RISK COMMUNICATION RESEARCH, — 2023. Available: <https://jicrer.com/index.php/jicrer/article/view/3762/3176>
- [29] Raghu Gollapudi, "Backup Integrity and Recovery Readiness Assessment for High-Availability Databases," Computer Fraud and Security, 2023. Available: <https://computerfraudsecurity.com/index.php/journal/article/view/1031/739>