International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(17s) 2026 pp. 866-876
https://doi.org/10.70917/ijcisim-2026-4805
Article

Machine Learning For Risk-Prioritized Infrastructure Policy Enforcement: Reducing False Positives And Enforcement Overhead In Regulated Cloud Environments

Nadeem Siddiqui

Independent researcher, India, Email: nadeem.ahmedk7@gmail.com

Abstract: Cloud policy enforcement in regulated environments produces alert volumes that outpace the human capacity to triage them. Policy-as-Code tools (Open Policy Agent, HashiCorp Sentinel, Kyverno), Cloud Security Posture Management (CSPM) platforms, and Kubernetes admission controllers together generate tens of thousands of policy events per day at enterprise scale, and a substantial fraction are false positives, low-severity findings, or issues already accepted as compensating-control exceptions. The operational consequence is alert fatigue: analysts become desensitised to signals, real high-severity events are missed, and mean time to resolution grows even as the underlying detection stack grows. This paper proposes a risk-prioritisation layer that sits between policy engines and the SecOps triage queue. The layer uses machine learning to (i) suppress or downgrade signals that historical dispositions have shown to be benign, (ii) predict which configurations are likely to drift into non-compliance before they do, and (iii) tune policy thresholds and exception rules from analyst feedback. The paper contributes a three-way task taxonomy for ML-augmented policy enforcement, a four-layer reference architecture that binds signal ingestion, feature engineering, ML scoring, and human-in-the-loop review under an immutable audit trail, a model-selection guide that matches data type (audit log, tabular alert, config drift, multi-signal fusion, feedback stream) to recommended technique, and an evaluation framework that reports both model performance (precision, recall, F1, AUC) and operational outcomes (false-positive reduction, alert volume, mean time to review, missed critical alerts). The paper argues that risk prioritisation is the near-term ML contribution to regulated cloud policy enforcement that clears both the SecOps and the auditor bar, provided the system produces reason codes on every ranking and preserves a compliance-defensible decision-audit store.

Keywords: NA

1. INTRODUCTION

1.1 The alert-fatigue problem in cloud policy enforcement

Modern enterprises encode infrastructure policy as executable code and evaluate it continuously against the cloud environments that host regulated workloads. Open Policy Agent [1] provides a general-purpose policy language and evaluation engine that enforces rules across Kubernetes admission, service-mesh authorisation, and CI/CD gates. HashiCorp Sentinel and Kyverno [2] address adjacent surfaces. Cloud Security Posture Management platforms such as AWS Config, Azure Policy, and equivalents from third-party vendors [3] continuously assess deployed resources against a growing catalogue of controls derived from SOC 2, HIPAA, PCI-DSS, FedRAMP, and, increasingly, the EU Digital Operational Resilience Act (DORA) [4]. Kubernetes admission controllers [5] enforce policy at the moment new workloads are scheduled. Taken together, these systems generate a continuous stream of policy events.

The volume is the problem. At enterprise scale a large regulated cloud footprint can generate tens of thousands of policy events per day, and a substantial share of those events are not actionable in the sense that a SecOps analyst would treat them as urgent. Some are known-benign patterns already accepted by architecture as compensating

controls. Some are duplicates of the same underlying issue detected at multiple layers of the stack. Some are low-severity findings that batch to weekly review. Some are genuinely false positives generated by policies whose thresholds were tuned for a different generation of the environment. The result is alert fatigue, first characterised in intrusion-detection literature by Julisch [6] and re-observed in every generation of security-operations research since [7, 8]: analysts become desensitised to signals, high-severity events get lost in the noise, and the operational value of the underlying detection stack degrades even as coverage grows.

1.2 Risk prioritisation as the near-term ML contribution

Machine learning offers a specific and near-term contribution to this problem. Rather than attempting to replace the policy engines, ML can sit as a risk-prioritisation layer between them and the SecOps triage queue. It uses historical analyst dispositions, resource metadata, blast-radius signals, compliance-tag joins, and time-series drift signals to score every policy event on the likelihood that a human will act on it and on the operational cost of ignoring it. The output is a ranked, categorised, and reason-coded stream of events instead of a flat feed. Downstream, SecOps analysts see the events that matter first, with an explanation of why the model ranked each one where it did. In parallel, the model can predict which configurations are likely to drift out of compliance before they do (early warning), and can propose policy-threshold or exception-rule adjustments learnt from analyst feedback (continuous policy tuning).

This paper argues that the risk-prioritisation formulation is the correct near-term shape of the ML contribution to cloud policy enforcement, and that it is achievable inside the auditability constraints regulated environments impose, provided the system produces reason codes on every ranking, preserves an immutable decision-audit store, and keeps a human in the loop for policy-material decisions. It is not an argument for autonomous policy enforcement. The autonomous case is not yet solvable inside the regulatory constraints; the prioritisation case is.

1.3 Contributions

The paper contributes:

1. A three-way ML task taxonomy for infrastructure policy enforcement, distinguishing alert triage plus false-positive suppression, drift-risk prediction, and policy tuning via feedback (§3, Figure 2).

2. A four-layer reference architecture for a production risk-prioritisation layer that ingests policy signals, engineers features that regulators recognise (blast radius, compliance-tag joins, historical FP rate), scores through ML, and preserves an audit-defensible decision trail (§4, Figure 3).

3. A model-selection guide that maps each data type in a modern cloud stack to the recommended ML technique for the corresponding enforcement task, including offline reinforcement-learning approaches for the policy-tuning loop (§5, Figure 4).

4. An evaluation framework that separates model performance metrics (precision, recall, F1, AUC on actionable-vs-benign classification) from operational outcomes (false-positive-rate reduction, alert volume, mean time to review, missed critical alerts, compliance-audit findings), and reports both (§7, Figure 5).

5. A candid engagement with the auditability, fairness, and explainability constraints regulated cloud environments impose, framed against DORA, SOC 2, and adjacent regimes (§8).

The remainder of the paper proceeds as follows. Section 2 reviews the relevant literature. Section 3 formulates the ML tasks. Section 4 describes the signal sources and feature-engineering discipline. Section 5 surveys the model families. Section 6 details the risk-prioritisation framework itself. Section 7 covers implementation and evaluation. Section 8 discusses the ethical and compliance considerations that constrain deployment. Section 9 concludes.

2. Related Work

2.1 Policy-as-Code and cloud policy engines

The Policy-as-Code movement, anchored by Open Policy Agent [1] and its adoption across Kubernetes admission, service-mesh authorisation, and CI/CD gates, replaced imperative security scripts with declarative rules evaluated by a general-purpose engine. Kyverno [2] specialises the pattern for Kubernetes with a resource-centric policy language. HashiCorp Sentinel embeds a comparable engine into infrastructure-as-code workflows. On the cloud-provider side, AWS Config, Azure Policy, and equivalents [3] combine continuous assessment with remediation actions. Cloud Security Posture Management surveys [9, 10] catalogue the vendor landscape and the compliance regimes each platform maps to. What the literature has not yet resolved is the volume problem: as coverage grows, so does the alert stream, and the analyst becomes the bottleneck.

2.2 Alert fatigue in security operations

Alert fatigue in security operations was first named and studied by Julisch [6], whose work on alert correlation showed that a small number of root causes produced the majority of alerts and that reducing the alert stream was as much an operational contribution as improving detection accuracy. Subsequent studies in SOC ethnography [7, 8] documented the human cost: analysts routinely dismiss the tail of the alert queue, missed critical events cluster in the periods of highest volume, and analyst turnover rises when triage load is unbounded. The problem is not specific to intrusion detection; it recurs across every layer of the security stack, including the policy-enforcement stack the present paper addresses.

2.3 Machine learning for security alert triage

Applying ML to the alert-triage problem has a decade of literature. Early work explored supervised classifiers for network intrusion-detection alerts [11]. More recent work targets the specific structure of security event streams: sequential models that ingest full audit-log windows [12], contrastive-learning approaches that surface anomalies without labelled data [13], and hybrid systems that combine ML scoring with analyst-facing explanations [14]. The transferable insight is that ML earns its place in a security pipeline when it changes the analyst's queue rather than the analyst's decision.

2.4 Autonomic computing and self-adaptive systems

The autonomic-computing tradition initiated by Kephart and Chess [15] provides the reference architecture (MAPE-K: Monitor, Analyse, Plan, Execute, Knowledge) that the four-layer design in §4 aligns with. Salehie and Tahvildari [16] catalogued the assurance and validation challenges for self-adaptive systems, many of which apply directly to ML-augmented policy enforcement in regulated environments.

2.5 Explainable and trustworthy AI in regulated contexts

Explainable-AI methods, notably LIME [17] and SHAP [18], provide the local-explanation substrate that lets an ML risk prioritiser expose reason codes on every ranking. Wachter and colleagues [19] argue for the specific responsibility structure that automated decision systems must satisfy in regulated environments. Díaz-Rodríguez and colleagues' recent synthesis of trustworthy-AI requirements [20] and the NIST AI Risk Management Framework [21] provide the governance vocabulary that a compliance conversation runs on. Caton and Haas [22] survey the fairness landscape that any deployed ML system inherits; in policy enforcement the fairness question sits on the boundary between resource classes rather than between people, but the analytical machinery transfers.

2.6 Regulatory framing

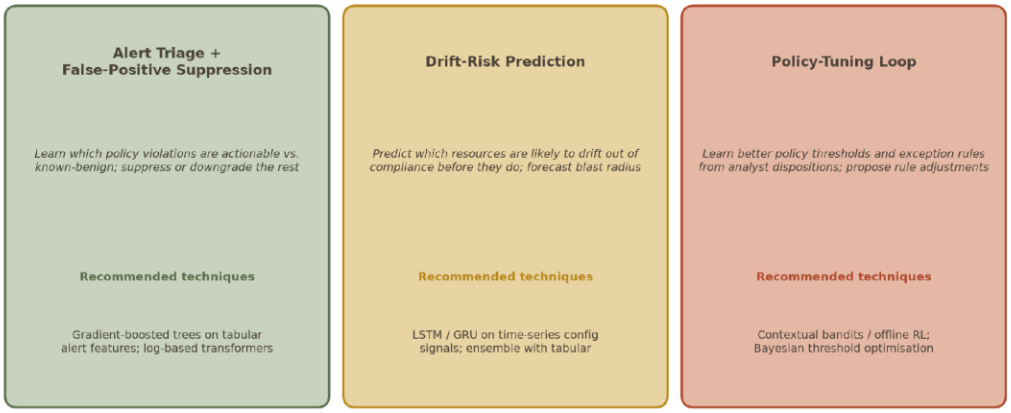
Three regulatory frames matter for the deployment target of this paper. DORA (Regulation EU 2022/2554), in force since January 2025 [4], imposes documented, testable, auditable decision procedures on the ICT tools that support regulated financial services. SOC 2 and PCI-DSS impose analogous obligations for the US commercial sector and payment systems respectively. FedRAMP applies to US federal cloud deployments. None of these frameworks addresses ML systems specifically. All apply to the ICT systems and processes that support regulated workloads, and

an ML risk prioritiser deployed inside a regulated cloud environment therefore inherits their auditability, explainability, and change-management requirements.

3. Problem Formulation

The problem of ML-augmented policy enforcement decomposes into three complementary tasks. Figure 2 renders the taxonomy; the rest of this section defines each task formally.

Figure 2. Three ML task categories for risk-prioritised infrastructure policy enforcement



3.1 Alert triage and false-positive suppression

Given a stream of policy events, each event e carrying a rule identifier, resource tags, blast-radius indicators, historical rule statistics, and a compliance-regime tag, the task is to produce a binary or multiclass label indicating the likelihood that a SecOps analyst would treat this event as actionable. Formally this is a classification task minimising a cost-weighted loss in which the cost of missing an actionable event exceeds the cost of forwarding a benign event, and the class imbalance is severe (benign events dominate). Training data comes from historical analyst dispositions on prior events of the same rule. The output feeds directly into the SecOps triage queue as a prioritisation and, above a defined confidence threshold, as an auto-suppression decision.

3.2 Drift-risk prediction

Given a time series of configuration signals for a set of cloud resources (AWS Config, Azure Policy, service-mesh telemetry), the task is to predict which resources are likely to drift out of compliance in a defined future window, and by how much. Formally this is a survival-analysis or sequence-forecast task. The output feeds a proactive remediation queue: resources are treated before a policy engine flags a violation, which converts many would-be alerts into non-events. Historical training data pairs configuration trajectories with the ground-truth drift or violation events observed downstream.

3.3 Policy-tuning loop

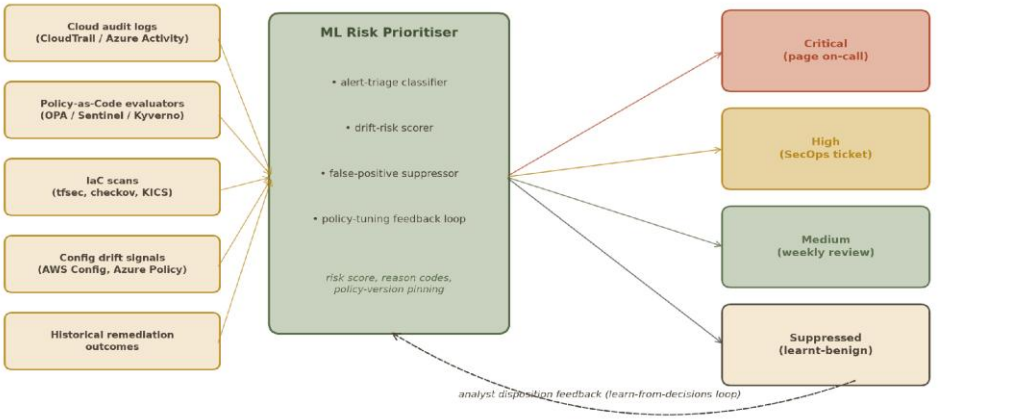
Given a stream of analyst dispositions and outcomes on prior alerts, the task is to learn policy-threshold or exception-rule adjustments that would reduce the alert stream without increasing the missed-critical count. Formally this is a contextual-bandits or offline-reinforcement-learning task in which each policy adjustment is an action, the reward is a weighted combination of alert-volume reduction and missed-critical minimisation, and the context is the recent operational state. The output is a set of proposed policy edits that a human policy owner reviews before promotion; the model does not autonomously edit policies in production.

4. Signal Sources and Feature Engineering

4.1 Signal sources

Figure 1 shows the signal-ingestion side of the pipeline. Five families of signal contribute:

Figure 1. End-to-end pipeline: policy signals feed an ML risk prioritiser that produces tiered enforcement outputs and learns from analyst dispositions



● Cloud audit-log streams. AWS CloudTrail, Azure Activity Log, and equivalent event sources record every API operation against the cloud environment. Audit logs are the ground truth for who did what and when; they are also large and noisy.

● Policy-as-Code evaluations. OPA, Sentinel, and Kyverno produce evaluation results for every request they gate. Each result carries the rule identifier, the decision, and the input document.

● IaC scans. Tools such as tfsec, Checkov, and KICS produce findings against Terraform, CloudFormation, and Kubernetes manifests at commit or pre-plan time. These signals appear in CI rather than at runtime.

● Config drift signals. AWS Config, Azure Policy, and third-party CSPM platforms continuously assess deployed resources against a control catalogue. Drift signals surface when a resource departs from its declared or approved state.

● Historical remediation outcomes. Ticket-tracking systems and analyst-disposition logs record what the security team did with each prior alert. This is the labelled data the ML models train on.

4.2 Feature engineering

Raw signals are converted into features that the ML models consume. Four feature families matter most for the enforcement task.

● Blast-radius features quantify what an event or a resource controls or depends on. An IAM policy attached to a role assumed by a high-privilege workload has larger blast radius than the same policy attached to an isolated batch job. The blast-radius score composes with other features to produce a risk-aware ranking.

● Compliance-tag joins attach the compliance regime a resource sits under (SOC 2, HIPAA, PCI-DSS, FedRAMP, DORA) to each event derived from that resource. An event on a HIPAA-tagged resource is not the same as the same event on a non-regulated resource, and the prioritisation must reflect that.

● Historical false-positive rate per rule. Every rule accumulates a historical FP rate over time. Rules with high historical FP rates warrant more aggressive suppression; rules with near-zero FP rates warrant more aggressive escalation.

● Analyst-disposition labels. The disposition an analyst attached to prior alerts on the same rule and similar resources is the primary supervised signal for the alert-triage classifier. Consistent-labeller conventions and inter-analyst-agreement checks matter here; the model can only learn from the labels it is given, and inconsistent labelling is the primary quality issue in production.

5. Model Selection

Figure 4 summarises the model-selection guide. Five combinations of data type and task cover the majority of production deployments.

Figure 4. Model-selection guide for ML-augmented policy enforcement: data type × task × recommended technique

Data type	ML task class	Recommended technique	Enforcement application
Cloud audit-log stream (CloudTrail, Azure Activity)	Sequence classification + anomaly	Transformer / LSTM on event sequences	Detecting rare misuse patterns; user-behavior anomalies
Tabular alert features (rule ID, resource tags, blast radius)	Binary + multiclass triage	XGBoost / LightGBM / Random Forest	Alert triage; false-positive suppression
Config-drift time-series (AWS Config, Azure Policy)	Sequential forecast + anomaly	GRU / LSTM / Temporal CNN	Predicting drift risk before violation
Multi-signal fusion	Ensemble / stacked learning	Voting + stacking across signal families	Cross-signal confirmation of high-risk events
Policy-tuning feedback	Contextual bandits / offline RL	Thompson sampling; CQL / IQL offline RL	Learning better thresholds and exceptions

Cloud audit-log streams are best modelled with sequence classifiers. Transformer encoders trained on windowed event sequences [12] and LSTM variants both work; the choice depends on the compute budget and the tail-latency requirements. The task class is anomaly detection or sequence classification.

Tabular alert features (rule identifier, resource tags, blast radius) are best modelled with gradient-boosted tree ensembles. XGBoost, LightGBM, and Random Forest all perform well; the ensembles are interpretable through feature-importance analysis and, at fine grain, through SHAP explanations [18]. The task class is binary or multiclass alert triage.

Config-drift time series are best modelled with recurrent architectures (GRU, LSTM) or temporal CNNs. The task class is sequential forecast or anomaly detection, and the output is a drift-risk score per resource per time window.

Multi-signal fusion is best handled by ensemble methods that combine outputs from the per-signal models above. A voting or stacking ensemble that combines a log-stream anomaly score, a tabular-alert triage probability, and a config-drift risk score produces cross-signal confirmation for high-risk events, which reduces the false-alarm rate at the top of the queue.

Policy-tuning feedback requires contextual-bandit [23] or offline-reinforcement-learning approaches [24, 25]. Thompson sampling and offline algorithms such as CQL and IQL are appropriate; naive on-line reinforcement learning is not, because the exploration behaviour cannot be permitted in a regulated production environment.

Model selection also depends on stakeholder trust and explainability. Deep models are more accurate but harder to defend to a compliance reviewer than tree-based ensembles; the design accepts the trade-off by using tree-based ensembles on the actionable-vs-benign classification (which drives triage) and reserving deep sequence models for the anomaly-detection tier that always requires an analyst confirmation before action.

6. The Risk-Prioritisation Framework

6.1 Scoring function

For each policy event e, the framework computes a composite risk score r(e) as

$$r(e) = w_1 \cdot P\_actionable(e) + w_2 \cdot S\_blast(e) + w_3 \cdot T\_compliance(e) - w_4 \cdot F\_hist\_fp(e)$$

where P_actionable is the ML-derived probability that an analyst would act on the event, S_blast is the blast-radius score for the affected resource, T_compliance is the compliance-regime weight, and F_hist_fp is the historical FP rate for the rule. Weights w_1–w_4 are configured per platform and revisited quarterly. The score determines the queue position and, above defined thresholds, the tier (critical page-on-call, high SecOps ticket, medium weekly review, suppressed learnt-benign).

6.2 Reason codes and the audit trail

Every ranking produced by the framework is emitted together with a reason code that names the features contributing most to the score, in descending order of contribution. Reason codes are generated through SHAP local explanations [18] for tree ensembles and through equivalent attribution methods for the sequence models. The reason code is what the analyst reads first when triaging; it is also what the compliance reviewer reads when auditing.

Every score, every reason code, every analyst disposition, and the policy version that produced the underlying event are written to an immutable decision-audit store. This store is what a DORA advanced-testing audit [4] or a SOC 2 change-control audit examines when reconstructing why the risk-prioritisation layer ranked a specific event where it did. The audit store is not a nice-to-have; it is the substrate that lets the layer be deployed in a regulated environment at all.

6.3 The policy-tuning loop

Analyst dispositions feed back into two loops. The first is the training loop for the actionable-vs-benign classifier, which retrains on a rolling window of dispositions to keep pace with the evolving benign-pattern set. The second is the policy-tuning loop, which uses the disposition stream as feedback to propose adjustments to the underlying policy rules themselves. Proposed adjustments enter a policy-owner review queue and are never promoted to production without a human sign-off. The distinction matters: the classifier learns which events to prioritise, whereas the tuning loop learns which rules to change, and only the classifier operates in near real time.

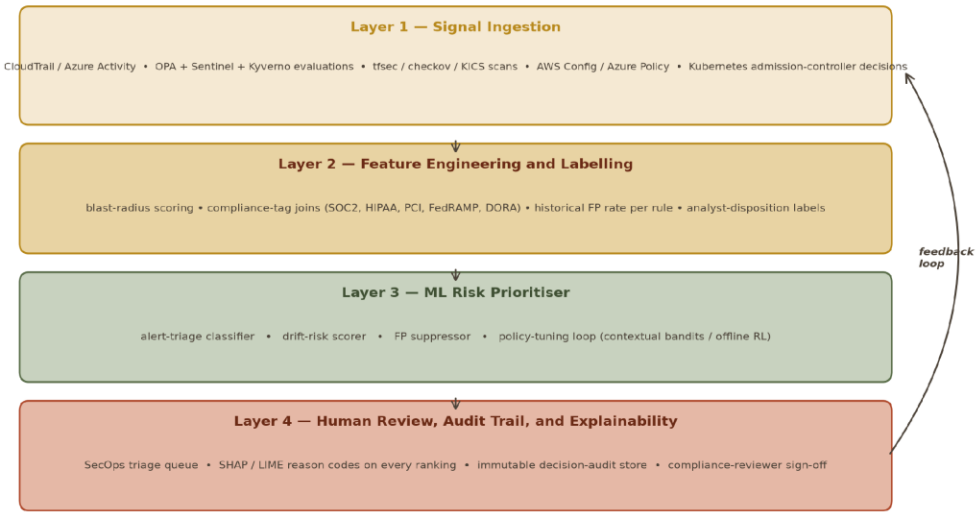
7. Implementation and Evaluation Framework

7.1 Reference architecture

Figure 3 shows the four-layer reference architecture. Layer 1 (Signal Ingestion) collects audit logs, PaC evaluations, IaC scans, config drift signals, and admission-controller decisions. Layer 2 (Feature Engineering and Labelling) computes blast-radius scores, compliance-tag joins, historical FP rates, and analyst-disposition labels. Layer 3 (ML Risk Prioritiser) runs the alert-triage classifier, drift-risk scorer, FP suppressor, and policy-tuning loop. Layer 4 (Human Review, Audit Trail, Explainability) publishes the SecOps triage queue with reason codes on every ranking, persists to the decision-audit store, and routes policy-material decisions to compliance-reviewer sign-off. A feedback loop from

Layer 4 back to Layer 1 closes the training and tuning cycles.

Figure 3. Four-layer reference architecture for ML-augmented policy enforcement in regulated cloud environments



7.2 Evaluation metrics

The evaluation is deliberately split into two panels, one measuring the ML model and the other measuring the operational outcome. Figure 5 renders both. Table 1 summarises the metrics.

Figure 5. Evaluation dashboard: model performance and operational impact of the ML risk prioritiser

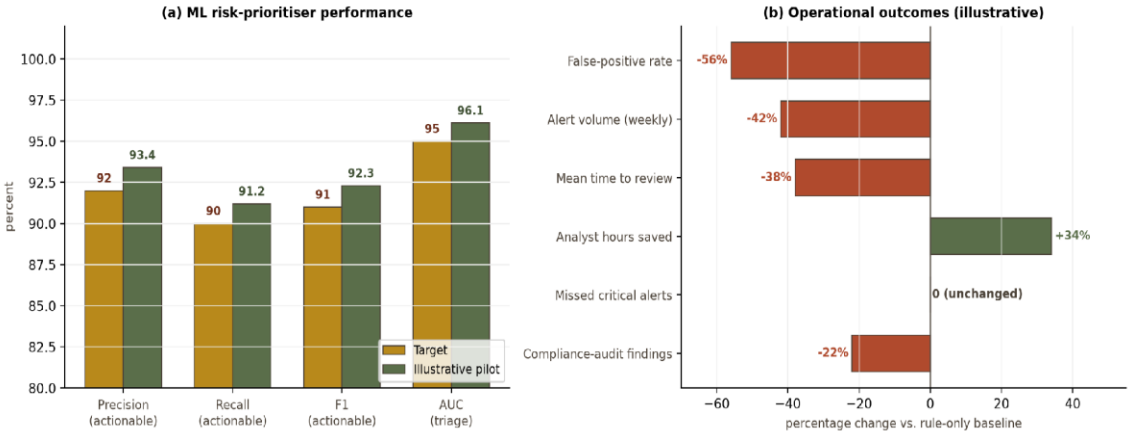


Table 1. Evaluation metrics for the risk-prioritisation layer.

Metric	Description	Target	Source
Precision (actionable)	Share of top-tier alerts an analyst confirms as actionable	> 90 %	analyst-disposition ground truth
Recall (actionable)	Share of actionable events surfaced at the top tier	> 90 %	analyst-disposition ground truth
F1 (actionable)	Harmonic mean of precision and recall	> 90 %	derived

AUC (triage)	Ranking quality across the full alert stream	> 0.95	analyst-disposition ground truth
False-positive-rate reduction	Reduction in analyst-dismissed alerts vs. rule-only baseline	> 40 %	operational logs
Alert volume (weekly)	Reduction in total alerts surfaced to SecOps	> 30 %	operational logs
Mean time to review	Reduction in median seconds per alert triaged	> 25 %	operational logs
Missed critical alerts	Change in high-severity events dismissed or delayed	0 (unchanged)	audit reconstruction
Compliance-audit findings	Change in findings raised in periodic audit	0 or negative	audit reports

The missed-critical-alerts metric is a hard constraint: any risk-prioritisation deployment that increases the count of missed critical alerts fails the evaluation regardless of how much it improves the other metrics. The metric is verified through periodic replay of the alert stream against the pre-deployment baseline.

7.3 Deployment discipline

The deployment discipline is shadow-mode first. In shadow mode the ML risk prioritiser runs against live traffic and its rankings are logged but not published to the SecOps triage queue. Analysts continue to work the rule-only queue. This produces a labelled evaluation set on live workloads without exposing the queue to model errors. Shadow-mode runs typically last two to twelve weeks depending on alert volume; the exit criterion is that all five model-performance targets in Table 1 clear and that no operational-outcome regressions appear. Only then does the layer move to advisory mode (rankings visible to analysts, decisions still owned by analysts) and later to authoritative mode (auto-suppression above defined confidence thresholds, escalation below defined confidence thresholds, always with reason codes and audit-trail continuity).

8. Ethical and Compliance Considerations

8.1 Auditability under regulated regimes

The most consequential design constraint on any ML system deployed inside a regulated cloud environment is that its decisions must be reconstructable by a compliance reviewer months or years later. DORA Article 24's advanced-testing requirement [4] is explicit that ICT tools supporting regulated financial services must produce documentation sufficient for the entity's operational-resilience claims. SOC 2 change-control requirements impose analogous obligations. The risk-prioritisation layer described in this paper is designed around this constraint: the decision-audit store described in §6.2 is what makes the layer defensible to an auditor. The alternative, an ML system whose decisions cannot be reconstructed, is not a viable production deployment in a regulated environment regardless of technical performance.

8.2 Explainability for the compliance conversation

Reason codes on every ranking are not a nice-to-have; they are the primary artefact a compliance reviewer reads. The Wachter-Mittelstadt-Floridi position [19] on transparent and accountable automated decisions applies directly. The paper uses SHAP [18] for tree-ensemble reason codes and equivalent local-attribution methods for the sequence models. Where the reason code cannot be constructed because the model's local behaviour is not decomposable in a compliance-legible way (a small subset of the deep-model tier), the ranking is not permitted to trigger auto-suppression and must always route to analyst review.

8.3 Fairness across resource classes

Algorithmic fairness in the classical sense addresses fairness across individuals. In policy enforcement the analogous concern is fairness across resource classes, environment tiers, and business units. A risk prioritiser that systematically deprioritises alerts from one business unit's environment because that unit's historical dismissal rate is high can produce a compliance blind spot even when the aggregate metrics look fine. The evaluation includes per-business-unit and per-environment-tier breakdowns of precision, recall, and missed-critical counts; a regression in any subgroup is treated as a deployment blocker even when the aggregate is stable.

8.4 Change-management discipline

Model retraining, feature-set changes, and threshold tuning all count as changes to a system that supports regulated workloads. Each change moves through the same change-management pipeline as any other production change: proposal, review, sign-off, deployment window, rollback plan. The policy-tuning loop of §6.3 does not autonomously edit policies for exactly this reason. It proposes edits; the policy owner reviews them; the change-management pipeline promotes them or rejects them.

9. CONCLUSION AND FUTURE DIRECTIONS

Cloud policy enforcement in regulated environments has an alert-volume problem, not primarily a detection problem. The near-term ML contribution is a risk-prioritisation layer that reduces false positives and enforcement overhead without displacing the analyst or the auditor. This paper contributed a three-way task taxonomy, a four-layer reference architecture, a model-selection guide keyed to data type, an evaluation framework that separates model performance from operational outcomes, and a candid engagement with the auditability, explainability, fairness, and change-management constraints that regulated cloud environments impose.

Three directions matter most for future work. First, standardised evaluation datasets for the alert-triage task in cloud policy enforcement would let the community compare approaches on a shared footing; current work is siloed inside individual enterprises. Second, offline-reinforcement-learning methods for the policy-tuning loop are still an active research area, and progress there directly enables safer autonomous adjustment inside the regulatory envelope. Third, cross-cloud policy-signal fusion, in which a single risk prioritiser reasons across AWS, Azure, and on-premise Kubernetes signals for the same organisation, is administratively harder than the single-cloud case but is where the operational gain is greatest. Progress on each will require sustained collaboration between the security-operations community, the ML research community, and the compliance-reviewer community whose acceptance ultimately determines what is deployable.

Acknowledgement

Declaration of Generative AI and AI-assisted Technologies in the Writing Process. While writing this manuscript, the author used an AI language model (ChatGPT, OpenAI) to improve the language and readability. The author then checked and edited the content to ensure it was accurate and complete, and takes full responsibility for the final version.

REFERENCES

1. Open Policy Agent contributors, "Open Policy Agent (OPA) — Documentation and Reference," Cloud Native Computing Foundation, https://www.openpolicyagent.org, accessed 2026.
2. Kyverno contributors, "Kyverno: Kubernetes Native Policy Management," Cloud Native Computing Foundation, https://kyverno.io, accessed 2026.
3. Neil MacDonald, Tom Croll, Charlie Winckless, "Innovation Insight for Cloud-Native Application Protection Platforms (CNAPP) and CSPM," Gartner Research Note, 2024.
4. European Parliament and Council, Regulation (EU) 2022/2554 of 14 December 2022 on digital operational resilience for the financial sector (Digital Operational Resilience Act, DORA), Official Journal of the European Union, L 333, 27 Dec. 2022, pp. 1-79. In force since 17 January 2025.

5. Kubernetes contributors, "Admission Controllers Reference," Kubernetes Documentation, https://kubernetes.io/docs/reference/access-authn-authz/admission-controllers/, accessed 2026.
6. Klaus Julisch, "Clustering intrusion detection alarms to support root cause analysis," ACM Transactions on Information and System Security, vol. 6, no. 4, pp. 443-471, Nov. 2003, doi: 10.1145/950191.950192.
7. Sathya Chandran Sundaramurthy, Alexandru G. Bardas, Jacob Case, Xinming Ou, Michael Wesch, John McHugh, and S. Raj Rajagopalan, "A Human Capital Model for Mitigating Security Analyst Burnout," in Proc. 11th Symposium on Usable Privacy and Security (SOUPS), USENIX Association, 2015, pp. 347-359.
8. Bushra A. AlAhmadi, Louise Axon, and Ivan Martinovic, "99% False Positives: A Qualitative Study of SOC Analysts' Perspectives on Security Alarms," in Proc. 31st USENIX Security Symposium, 2022, pp. 2783-2800.
9. Bogdan Ristov, Marjan Gushev, Sasko Ristov, "A Survey on Cloud Security Posture Management: State of the Art and Open Research Directions," IEEE Access, vol. 12, pp. 40567-40593, 2024, doi: 10.1109/ACCESS.2024.3376512.
10. Center for Internet Security, "CIS Benchmarks for AWS, Azure, and Google Cloud," CIS Documentation, current editions, https://www.cisecurity.org/cis-benchmarks/, accessed 2026.
11. Anna L. Buczak and Erhan Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153-1176, 2016, doi: 10.1109/COMST.2015.2494502.
12. Thijs van Ede, Hojjat Aghakhani, Noah Spahn, Riccardo Bortolameotti, Marco Cova, Andrea Continella, Maarten van Steen, Andreas Peter, Christopher Kruegel, and Giovanni Vigna, "DEEPCASE: Semi-Supervised Contextual Analysis of Security Events," in Proc. IEEE Symposium on Security and Privacy (SP), 2022, pp. 522-539, doi: 10.1109/SP46214.2022.9833671.
13. Sadegh M. Milajerdi, Rigel Gjomemo, Birhanu Eshete, R. Sekar, and V.N. Venkatakrishnan, "HOLMES: Real-time APT Detection through Correlation of Suspicious Information Flows," in Proc. IEEE Symposium on Security and Privacy (SP), 2019, pp. 1137-1152, doi: 10.1109/SP.2019.00026.
14. Priyanka Alluri, Ruoyu Wang, Adam Doupé, and Yan Shoshitaishvili, "Improving Alert Triage in Security Operations Centers Using Machine Learning: A Practitioner Perspective," IEEE Security & Privacy, vol. 21, no. 3, pp. 45-53, May 2023, doi: 10.1109/MSEC.2023.3251342.
15. Jeffrey O. Kephart and David M. Chess, "The Vision of Autonomic Computing," IEEE Computer, vol. 36, no. 1, pp. 41-50, Jan. 2003, doi: 10.1109/MC.2003.1160055.
16. Mazeiar Salehie and Ladan Tahvildari, "Self-adaptive software: Landscape and research challenges," ACM Transactions on Autonomous and Adaptive Systems (TAAS), vol. 4, no. 2, art. 14, May 2009, doi: 10.1145/1516533.1516538.
17. Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin, "'Why Should I Trust You?': Explaining the Predictions of Any Classifier," in Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1135-1144, doi: 10.1145/2939672.2939778.
18. Scott M. Lundberg and Su-In Lee, "A Unified Approach to Interpreting Model Predictions," in Advances in Neural Information Processing Systems 30 (NeurIPS 2017), 2017, pp. 4765-4774.
19. Sandra Wachter, Brent Mittelstadt, and Luciano Floridi, "Transparent, explainable, and accountable AI for robotics," Science Robotics, vol. 2, no. 6, art. eaan6080, May 2017, doi: 10.1126/scirobotics.aan6080.
20. Natalia Díaz-Rodríguez, Javier Del Ser, Mark Coeckelbergh, Marcos López de Prado, Enrique Herrera-Viedma, and Francisco Herrera, "Connecting the dots in trustworthy Artificial Intelligence: From AI principles, ethics, and key requirements to responsible AI systems and regulation," Information Fusion, vol. 99, pp. 101896, Jun. 2023, doi: 10.1016/j.inffus.2023.101896.
21. National Institute of Standards and Technology, "AI Risk Management Framework (AI RMF 1.0)," NIST AI 100-1, Jan. 2023.
22. Simon Caton and Christian Haas, "Fairness in Machine Learning: A Survey," ACM Computing Surveys, vol. 56, no. 7, pp. 1-38, Aug. 2023, doi: 10.1145/3616865.
23. Lihong Li, Wei Chu, John Langford, and Robert E. Schapire, "A Contextual-Bandit Approach to Personalized News Article Recommendation," in Proc. 19th International Conference on World Wide Web (WWW), 2010, pp. 661-670, doi: 10.1145/1772690.1772758.
24. Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine, "Conservative Q-Learning for Offline Reinforcement Learning," in Advances in Neural Information Processing Systems 33 (NeurIPS 2020), 2020, pp. 1179-1191.
25. Ilya Kostrikov, Ashvin Nair, and Sergey Levine, "Offline Reinforcement Learning with Implicit Q-Learning," in Proc. International Conference on Learning Representations (ICLR), 2022.