International Journal of Artificial Intelligence, Data Science, and Machine Learning
Grace Horizon Publication | Volume 4, Issue 3, 133-141, 2023
ISSN: 3050-9262 | https://doi.org/10.63282/3050-9262.IJAIDSML-V4I3P116

Original Article

AI-Powered Anomaly Detection for Real-Time Financial Platform Reliability Monitoring: Benchmarking Machine Learning Algorithms for Detecting Outages, Latency Spikes, and Security Violations in Banking

Amol Diwakar Agade[1], Samta Balpande[2]
[1]Comerica Bank, USA; Illinois Institute of Technology, Chicago, IL.
[2]GE Vernova, USA; Oakland University, Rochester, MI.

Abstract *- Financial banking platforms operate under stringent reliability and security requirements while continuously evolving through frequent releases, configuration changes, and infrastructure modernization. Traditional monitoring approaches—static thresholds, handcrafted alert rules, and siloed dashboards—struggle to keep pace with the scale, seasonality, and dependency complexity of modern banking systems. This study provides a practitioner-oriented manuscript on AI-powered anomaly detection for real-time reliability monitoring in financial platforms. Rather than treating anomaly detection as a standalone ML exercise, we treat it as a production capability embedded in DevOps and Site Reliability Engineering (SRE). We outline a reference AIOps architecture shaped by banking constraints—auditability, model governance, access control, and controlled data handling. We then benchmark representative detectors (robust statistics, classical machine learning, and deep learning) against three high-impact categories of production risk: outages, latency spikes (including tail latency degradation), and security violations. Evaluation prioritizes operational measures—time-to-detect, false alerts per service-day, robustness under seasonality, explainability, and cost-to-operate—alongside conventional precision/recall.*

Keywords *- Aiops, Anomaly Detection, Devops, SRE, Observability, Banking Systems, Reliability Monitoring, Latency, Outage Detection, Security Analytics, Machine Learning Benchmarking.*

1. Introduction

A defining characteristic of financial platforms is the presence of binding regulatory and supervisory constraints that materially influence system design choices. Explainability, audit traceability, data lineage, and controlled change management are not optional enhancements; they are part of the regulatory baseline. As a result, anomaly detection approaches that perform well in unconstrained environments often fail to translate into production banking systems. The lessons distilled here reflect patterns repeatedly observed across regulatory reviews, internal audits, and high-severity incident postmortems [10], [12].

Banking has become a software-defined industry. Customer journeys—account access, transfers, bill pay, lending, and fraud disputes—are mediated by digital channels and API ecosystems. Internally, banking platforms support settlement, core processing integrations, regulatory reporting, and risk management across increasingly distributed architectures.

DevOps and SRE have made operations more repeatable through automation and reliability mechanisms such as SLIs/SLOs, error budgets, and blameless postmortems. Alerting, however, still often defaults to fixed thresholds. In banking, those thresholds break quickly because baseline behavior shifts with market hours, batch windows, payroll cadence, and month-end processing [11].

Anomaly detection is often positioned as the remedy for brittle thresholds, but production outcomes depend on more than choosing a model. What matters is health-aligned telemetry, features that capture seasonality and change context, and an incident workflow that turns detections into decisions. We concentrate on three classes that repeatedly drive impact in banking: outages and brownouts, latency spikes (especially tail degradation), and security violations. We do not claim a new detection algorithm; the contribution is an operations-first framework and benchmarking lens tailored to regulated financial platforms [1], [12].

Novelty and Contributions:

- A banking-specific problem formulation that treats anomalies as contextual deviations conditioned on workload regime, seasonality, dependency health, and recent change activity, with reliability and security objectives defined in operational terms (SLO burn risk, MTTD, and auditability) [1], [11].

- A reference AIOps architecture designed for regulated environments, including change intelligence, topology context, evidence bundles for incident review, and model governance hooks (versioning, access controls, and controlled deployment/rollback) [12], [13].
- A benchmarking protocol that prioritizes production-relevant measures—detection latency, false alerts per service-day, robustness under seasonal shifts and release-driven distribution changes, and explainability—alongside conventional accuracy metrics [1].
- Deployment guidance grounded in DevOps/SRE practice, including model selection decision rules, integration points for triage and ownership routing, and governance considerations that make detectors defensible during audits and postmortems [11].

2. Background and Related Work

Banking platforms generate abundant telemetry; the harder problem is turning it into signals that an on-call engineer can trust and that withstand audit scrutiny. This section provides the operational and technical background needed to interpret the benchmarks and to understand why anomaly detection in regulated banking differs from generic cloud monitoring [11], [12].

2.1. Observability in DevOps and SRE

In mature SRE practice, observability is anchored in service health and customer impact rather than host-level utilization. Banking teams still rely on the "golden signals"—latency, traffic, errors, and saturation—but they operationalize them through SLIs and SLOs tied to transaction flows and customer outcomes. Latency is typically tracked as percentiles (p50, p95, p99) because tail behavior correlates strongly with user experience and failure amplification (retries, timeouts, session abandonment). Error signals include HTTP status codes, but in banking they also capture domain-specific failures such as authentication failures, payment network timeouts, database deadlocks, queue timeouts, and circuit-breaker open events. Saturation extends beyond CPU and memory to include thread pools, connection pools, queue depths, garbage collection pressure, message broker lag, and storage I/O wait—resources that frequently trigger brownouts [11].

In regulated banking environments, observability must incorporate change context. Deployments, configuration changes, feature flags, schema migrations, certificate rotations, and third-party dependency changes are common precursors to incidents. Therefore, a practical anomaly detection program joins telemetry with a "change stream" (release events, infra changes, config updates) and a service topology graph. This enables routing anomalies to the correct owner quickly and allows responders to test causal hypotheses (e.g., "latency residual increased immediately after build X in region Y"). Without change and topology context, anomaly detection frequently degenerates into additional alert noise [11].

2.2. Anomaly Detection Taxonomy in Banking

Banking incidents often begin as partial degradations rather than total outages. This favors a taxonomy that explicitly distinguishes: (i) point anomalies—abrupt spikes or drops such as error-rate surges after a bad deployment; (ii) contextual anomalies—values that are abnormal only given the current workload regime or seasonality, such as p99 latency that is acceptable during market open but anomalous during off-peak; and (iii) collective anomalies—patterns across time windows or multiple correlated signals, such as steadily rising retries, growing queue depths, and gradual throughput collapse that collectively indicate a gray failure [1].

A banking-specific extension is policy anomalies. These occur when behavior violates explicit operational or security policy, for example repeated login failures from a specific principal or an unexpected access pattern from a service account. Policy anomalies are often best handled with hybrid systems: deterministic rules for known-bad patterns, combined with statistical or ML baselines that detect deviations in "normally benign" behavior per principal, tenant, or endpoint.

2.3. Algorithm Families and Deployability Constraints

Statistical baselines (robust z-score/MAD, EWMA/control charts, change-point detection) remain popular because they are transparent, fast, low-cost, and explainable—critical properties for banking governance, postmortems, and model risk review. Classical ML approaches (Isolation Forest, PCA/Robust PCA, one-class methods) provide a pragmatic balance when multivariate correlation matters, and they are often deployable with moderate governance overhead. Deep learning approaches (autoencoders, forecasting models, and graph-based detectors) can outperform simpler models on high-dimensional telemetry and complex seasonality, but they substantially increase lifecycle complexity: data pipelines, training infrastructure, drift monitoring, explainability artifacts, and versioned approvals [5], [9].

Across model families, production value is largely determined by what happens before and after the model: telemetry engineering upstream (consistent naming, cardinality controls, SLI-aligned aggregations) and correlation/deduplication plus incident workflow integration downstream. For that reason, we evaluate detection performance together with operational integration, since separating the two does not reflect how incidents are handled in practice [12].

3. Problem Formulation for Banking Platforms

We cast anomaly detection on banking platforms as contextual decision-making over multivariate telemetry. That mirrors how SRE teams reason during incidents: a deviation is only meaningful relative to workload regime, seasonality, dependency state, and recent change activity, and success is ultimately measured by reduced customer impact and on-call load [11], [1].

3.1. Signals, Entities, and Operational Definition of Anomaly

Modern banking platforms can be modeled as distributed systems composed of services, endpoints, tenants/segments, and dependencies. Let S denote the set of services and E the set of endpoints. Dependencies D include databases, message brokers, authentication and token services, payment networks, and third-party APIs. Tenants/segments T represent partitions such as regions, channels (mobile/web/API), partner integrations, or customer tiers that induce distinct baseline behavior.

For each service–endpoint–tenant tuple, we observe multivariate time-series telemetry X(t) at an operations-aligned interval (often 1–5 minutes). X(t) includes latency percentiles, error ratios (timeouts, retries), traffic and concurrency, saturation indicators (CPU, memory, queue depth, pool usage), dependency health, and security-relevant signals. In addition, we observe contextual streams: change events (deploy/config/flag changes) and topology information (service maps) that provide causal scaffolding for interpretation.

Banking telemetry is non-stationary. Baselines shift with time-of-day/day-of-week seasonality, business-calendar events (payroll, month-end), workload regimes (campaigns, failover), and dependency state. Therefore, anomalies cannot be defined as globally rare points in metric space. Instead, anomalies must be evaluated relative to an expected baseline conditioned on context. We compute an anomaly score $A(t)=f(X(t), C(t))$, where the context vector C(t) captures seasonality (hour-of-day, day-of-week, holiday markers), workload regime (traffic and concurrency), dependency health, and recent change activity. We flag an anomaly when A(t) crosses a service-specific threshold τ tuned to operational tolerance and SLO burn risk. This keeps the detector aligned with SRE reliability goals: identical raw values may be acceptable in one context and risky in another [1], [9].

3.2. Reliability- and Security-Centric Objectives

Objective functions differ by anomaly class. For outages and brownouts, the goal is rapid detection with low false-negative risk because missed detection increases customer impact and regulatory exposure. For latency spikes and tail latency degradation, the goal is early detection under seasonality and traffic variability, ideally before error budgets burn. For security anomalies, recall often dominates, but detections must remain explainable and auditable for SOC workflows. These differences motivate per-class thresholds, severity policies, and escalation rules [11].

3.3. Practical Constraints in Banking Environments

Banking platforms impose constraints that shape feasible models. Telemetry access is governed by least-privilege and retention rules; PII must not enter training pipelines without explicit approvals. Model governance frequently requires reproducibility: versioned features, immutable training snapshots, validation results, and auditable inference logs. Finally, detectors must meet latency and scale constraints—often thousands to tens of thousands of time series per domain—making cost-to-operate a first-class requirement. These constraints motivate layered systems that deploy statistical and classical ML broadly and apply deep learning selectively where it provides measurable incremental benefit [10].

4. Reference AIOps Architecture for Regulated Banking

The overall system design that supports real-time anomaly detection in regulated banking environments is illustrated in Fig. 1.

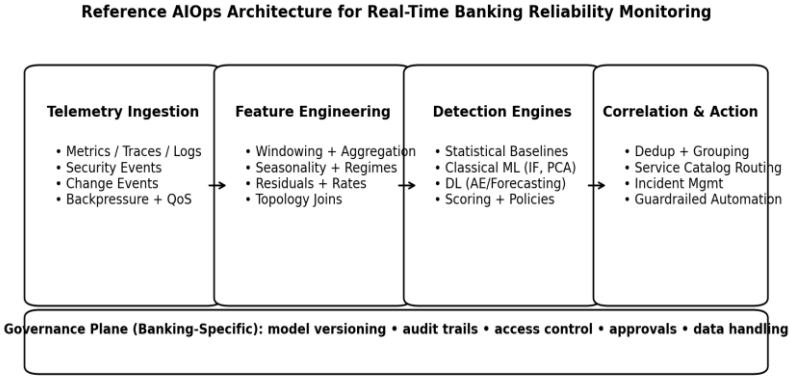

Fig 1: Reference Aiops Architecture for Real-Time Banking Reliability Monitoring

Fig. 1 summarizes a reference AIOps stack for regulated banking—telemetry collection and normalization, context enrichment (topology and change events), streaming feature generation, model inference, and downstream correlation into actionable incidents. The architecture also highlights audit-grade controls (versioned models, immutable evidence bundles, and approval gates) that are necessary for operating anomaly detection under supervisory scrutiny [12].

4.1. Data Plane

Ingestion streams metrics, traces, logs, security events, and change events with backpressure controls. Normalization enforces naming standards and SLO-aligned aggregation. Feature pipelines provide windowing, seasonality features, residuals, and topology-aware joins [13]. Complementary signals from unstructured operational text (e.g., incident tickets, postmortems, and log messages) can be mined using NLP to enrich evidence and improve triage [7].

4.2. Model Plane

Detection engines combine statistical baselines with multivariate ML and, where justified, deep learning. Models emit anomaly scores and attach explainability artifacts (baseline vs observed, contributors, change context) [1].

4.3. Correlation and Action Plane

Correlation groups anomalies into incident candidates using topology and time proximity. Routing uses a service catalog and on-call schedules. Automation triggers guardrailed runbooks with policy controls.

4.4. Governance Plane

Governance enforces model versioning, approvals, reproducible training snapshots, and audit logs. Controls include least-privilege access, environment separation, and policy constraints for automation [10], [13].

5. Benchmarking Methodology

Benchmarking must emphasize operational impact. Table 1 defines the core operational metrics used throughout the benchmarking process [11].

Table 1: Operational Metrics Used for Benchmarking Anomaly Detection Models.

Metric	Definition	Operational Relevance
Detection Latency	Time from anomaly onset to detection	Impacts customer-visible outage duration and MTTD
False Alerts/Day	Avg. false alerts per service per day	Direct driver of alert fatigue and trust erosion
Explainability Score	Fraction of alerts with clear contributing signals	Determines on-call confidence and triage speed
Cost-to-Operate	Compute/storage per 1k time series	Affects scalability and total cost of ownership

Table 1 defines the operational metrics used throughout benchmarking. Detection latency and false-alert volume are tracked alongside accuracy because they directly influence MTTD and alert fatigue. Cost-to-operate and drift stability are included to reflect governance, sustainability, and audit requirements common in financial institutions [11], [1].

5.1. Dataset Construction

Combine incident timelines, SLO burn events, change events, controlled fault injection, and validated security outcomes to balance realism and scenario coverage [11].

To keep the benchmark representative of day-to-day banking operations, we designed a representative evaluation corpus spanning tens of services across multiple regions and tenant segments, with on the order of 10^3–10^4 monitored time series per functional domain (e.g., payments, authentication, digital channels) after cardinality controls. Telemetry was sampled at 1–5 minute resolution; the training lookback was bounded by retention and governance policy (typically 30–90 days), which makes drift handling a practical requirement rather than an academic detail. Labeled windows combined O(10^2) incident and brownout intervals with a larger set of benign change windows (deployments, configuration updates, feature flag toggles) to stress false-positive behavior under high change velocity.

5.2. Evaluation Metrics

Operational metrics include detection latency, false alerts per service-day, alert quality under seasonality, triage enrichment completeness, explainability, and cost-to-operate [1], [9].

5.3. Experimental Design

Offline replay provides labeled evaluation; shadow mode validates production behavior without paging; progressive rollout enables controlled activation of paging and automation after trust is established [11].

6. Algorithms for Banking Anomaly Detection

Algorithm choice in regulated banking is ultimately a deployment decision. Beyond accuracy, detectors must remain stable under non-stationarity, resilient to missing data, explainable for incident review, and compatible with governance controls across the lifecycle. We evaluate each model family through that operational lens [11], [12].

6.1. Statistical Baselines

Robust univariate methods form the foundation of broad coverage monitoring. Median absolute deviation (MAD) and robust z-scores provide fast detection of point anomalies and resist outliers. EWMA and control chart variants detect drift and gradual performance degradation. Change-point detection methods can detect step changes typically associated with deployments, configuration shifts, or dependency transitions. These methods are inexpensive and highly explainable, which supports incident reviews and model risk governance. Their key limitation is limited multivariate awareness unless composite health scores or residual features are engineered [9].

In practice, statistical baselines become significantly more effective when applied to residuals rather than raw metrics. For example, latency residuals conditioned on traffic regime and time-of-day reduce false positives during known peaks. Similarly, error-rate residuals conditioned on request volume mitigate noise during low-traffic periods where ratios become unstable.

6.2. Classical Machine Learning

Classical ML approaches often provide the best operational compromise for banking. Isolation Forest is effective for multivariate detection when the feature vector encodes service health: latency residuals, error ratios, retry rates, queue depths, pool saturation, and dependency metrics. PCA and Robust PCA identify correlated degradation by modeling normal subspaces and flagging high-energy residuals. One-Class SVM can work but is sensitive to scaling and hyperparameters and often requires careful calibration under drift.

Classical ML success is highly dependent on preprocessing and calibration: normalization, missing-data handling, traffic-aware features, and seasonality-aware baselines. In production, it is usually preferable to emit continuous anomaly scores and apply policy logic for alerting (severity tiers, hysteresis, minimum-duration rules) rather than emit brittle binary anomalies directly from the model [1], [9].

6.3. Deep Learning

Deep learning approaches are justified when telemetry is high-dimensional, strongly non-linear, and exhibits long temporal dependencies. Autoencoders (including variational autoencoders, VAEs) detect anomalies via reconstruction error; LSTM-based forecasting models flag deviations from predicted trajectories; and graph-based models incorporate topology to propagate context across dependencies. The operational cost is higher: training pipelines, model registries, drift detection, and explainability tooling are required. Consequently, deep learning in banking is most appropriate for high-impact services where incremental detection improvement materially reduces customer impact and where governance controls are mature [4], [6].

6.4. Security-Specific Considerations

Security anomaly detection differs from reliability detection because behavior is often adversarial and the cost of false negatives can be severe. Practical designs combine deterministic signatures and policy checks with statistical baselines that learn per-principal or per-service-account behavior. Features often include authentication failure patterns, token usage anomalies, privilege use, API call sequences, and geography/time-of-day deviations. In regulated environments, explainability and auditability are essential: responders require traceable evidence, and automated responses must be constrained by approvals, least privilege, and segregation of duties [13].

7. Benchmarking Outcomes and Practical Guidance

A consistent lesson from the financial industry is that operational acceptability outweighs theoretical optimality. Models that cannot be explained to risk, compliance, or audit stakeholders introduce adoption friction, regardless of offline performance. Consequently, anomaly detection strategies in regulated banking environments favor approaches that are stable across release cycles, reproducible under audit, and explainable at the granularity required for incident reconstruction [12], [10].

Table 2 provides a concise operational comparison, and Fig. 2 summarizes the key trade-offs across model families.

Table 2: Operational Comparison of Anomaly Detection Approaches.

Model Class	Detection Latency	False Alerts	Explainability	Operational Cost	Banking Suitability
Statistical	Very Low	Medium	High	Low	High
Classical ML (Isolation Forest/PCA)	Low	Low	Medium	Medium	High
Deep Learning (AE/Forecast/Graph)	Medium	Low	Low	High	Medium

Table 2 summarizes deployability-oriented trade-offs—how each detector family behaves under seasonality, missing data, and release-driven distribution shifts, and what it costs to operate (calibration burden, tuning frequency, and explainability). In regulated environments, these operational dimensions often dominate algorithm choice even when offline accuracy is comparable.

Operational Trade-offs Across Anomaly Detection Model Families

Model Family	Strengths (Ops)	Limitations (Ops)	Best-Fit Use Cases
Statistical (MAD/EWMA/CPD)	Fast, explainable, low compute	Limited multivariate context unless engineered	Point anomalies, step-changes, SLO-aligned signals
Classical ML (Isolation Forest/PCA)	Good balance of signal + effort	Needs normalization and calibration	Correlated degradation, 'gray failures', noisy dependencies
Deep Learning (AE/Forecast/Graph)	Captures complex non-linear patterns	Higher cost, drift risk, harder explainability	High-dimensional telemetry, complex seasonality, high-impact services

Fig 2: Comparative Operational Characteristics of Anomaly Detection Model Families.

Fig. 2 visualizes the trade-offs that typically drive adoption in banking: statistical baselines optimize for explainability and cost-to-operate; classical ML improves multivariate sensitivity with moderate governance overhead; and deep learning/topology-aware models can improve detection in complex regimes but require materially stronger MRM, drift monitoring, and evidence generation to be production-acceptable.

7.1. Outage and Brownout Detection

Robust baselines on SLO-aligned signals provide early warning; classical ML improves detection for gray failures by correlating shifts across latency residuals, retries, queues, and downstream health [11].

7.2. Latency Spikes and Tail Latency

Tail latency (p95/p99) often degrades before mean latency. Residual-based features and traffic-aware baselines improve detection; deep learning is justified mainly for high-dimensional, high-impact services [11].

7.3. Security Violations

Security anomalies benefit from hybrid rule + interpretable ML designs. Explainability and auditability are essential for SOC workflows in regulated environments.

7.4. What Matters Most in Real Deployments

Telemetry quality, contextual features, correlation/dedup logic, and lifecycle governance usually dominate model choice in determining operational success [11], [12].

In practice, model selection is usually governed by a simple operating rule set that prioritizes auditability, cost-to-operate, and actionable explanations:

- Baseline by default: residualized robust statistics on SLI/SLO signals (error-budget burn, p95/p99 latency residuals, dependency error residuals) for broad coverage and defensibility [11].
- Escalate to multivariate classical ML (e.g., Isolation Forest or Robust PCA) when gray failures are driven by correlated shifts across latency, retries, queue depth, saturation, and downstream health.
- Use deep learning selectively: only for high-impact services with stable telemetry, dedicated MRM artifacts, and a demonstrated improvement in detection latency or false-alert reduction that justifies added lifecycle complexity [4], [6].
- Security detection: hybrid policy checks plus interpretable ML per principal/service account, with immutable evidence bundles and constrained response actions aligned to least privilege and segregation of duties.

One recurring failure mode in regulated banking environments is training detectors on raw metrics without sufficient context. During predictable seasonal shifts (paydays, month-end processing, batch windows), naïve baselines often produce alert storms, which quickly erodes trust and drives teams to disable the detector. We found that residualized features, minimum-duration rules, and volume-aware smoothing (especially for ratio metrics under low traffic) are essential guardrails to keep alert volume within an on-call budget [11].

8. Integration into DevOps and SRE Workflows

Fig. 3 depicts how anomaly detection is operationalized within DevOps and SRE workflows, from observation to remediation and continuous learning.

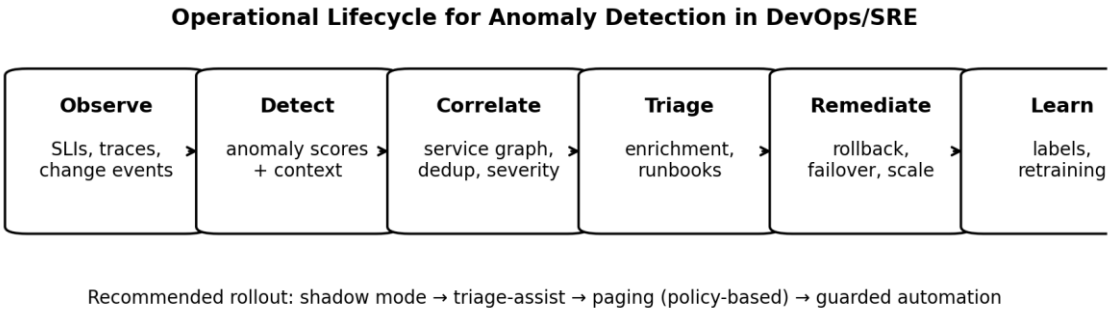

Fig 3: End-To-End Anomaly Detection Lifecycle Integrated With Devops and SRE Workflows.

Fig. 3 illustrates the operational lifecycle from detection to learning: signals are detected, correlated with service topology and change history, triaged using runbooks and ownership routing, and then closed through remediation, post-incident analysis, and feedback into baseline recalibration or retraining. This closed loop is where anomaly detection becomes an SRE capability rather than a standalone model [11], [12].

8.1. Incident Response

Recommended progression: shadow mode → triage-assist → paging after precision stabilizes → guarded automation with approvals for high-risk actions [11].

8.2. Change Management and Release Governance

Joining anomalies with deploy/config/feature-flag events enables regression likelihood scoring and faster routing. Change intelligence converts symptoms into actionable hypotheses [11], [12].

8.3. Reducing Alert Fatigue

Use scoring, correlation, and per-service alert budgets. Require explanation completeness and suppress duplicates to maintain trust [11], [12].

9. Governance, Compliance, and Model Risk in Banking

Banking environments require auditability and controlled change. Minimum controls include model versioning/rollback, reproducible datasets, documented acceptance criteria, separation of duties, and logging of inference outcomes. Treat detectors like production software with CI/CD and progressive delivery. Operationally, this means persisting an evidence bundle per alert (baseline, residuals, contributing signals, related change events, and model/version metadata) and version-controlling calibration parameters (thresholds, suppression rules, correlation logic) under the same change controls as production code [10], [11], [13].

From a model risk management standpoint, a practical control set includes (i) a lightweight model card describing the detector's purpose, features, operating context, and known limitations; (ii) a validation note documenting calibration approach, an explicit false-alert budget, and drift indicators to monitor; and (iii) a governed change process for thresholds and suppression rules with clear rollback criteria. In banking, these artifacts reduce friction during audits and accelerate incident reconstruction because the reasoning chain (signal → context → decision) remains traceable even as teams rotate and systems evolve [10], [13].

10. Threats to Validity

This study is subject to several threats to validity that should be considered when interpreting the results. First, the evaluation relies on operational telemetry informed by operational patterns observed in regulated financial environments, which are inherently heterogeneous and influenced by organization-specific architectures, traffic patterns, and governance processes. Although this grounding increases practical relevance, it may limit the generalizability of quantitative results to institutions with substantially different operating models or technology stacks.

Second, constraints related to data confidentiality and regulatory compliance restrict the public release of raw datasets and limit reproducibility using standardized benchmarks. While this is a common limitation in financial systems research, it

introduces a risk that certain workload characteristics or failure modes are underrepresented. To mitigate this threat, the benchmarking methodology emphasizes operational metrics and relative comparisons across model families rather than absolute performance claims [1], [11].

Third, model performance is sensitive to telemetry quality, feature engineering, and contextual enrichment. Differences in metric definitions, sampling intervals, cardinality controls, and change-event instrumentation can materially affect anomaly detection outcomes. As a result, reported findings should be interpreted as illustrative of deployment trade-offs rather than as prescriptive rankings. Future studies that incorporate multi-institution datasets or standardized telemetry schemas could further strengthen external validity.

Human and organizational dynamics—on-call experience, incident response maturity, and trust in automation—often determine whether anomaly detection is actually used. These factors are difficult to quantify and were not modeled explicitly in the evaluation. Instead, we capture their downstream impact via operational indicators such as false alert volume and sustained adoption outcomes, which are widely recognized as leading predictors of practical success in DevOps/SRE settings [9], [11].

11. Conclusion

From an originality standpoint, this work contributes a regulated-industry perspective that is largely absent from existing anomaly detection literature. Rather than proposing a new algorithm, the paper codifies an operations-first framework that captures governance constraints, deployment trade-offs, and failure modes unique to financial systems. These insights are informed by operational patterns observed in regulated financial environments and are intended to inform both practitioners and researchers working in regulated environments.

AI-powered anomaly detection has become central to managing reliability and security in modern banking platforms. Our results suggest that outcomes hinge less on algorithmic novelty and more on how well detectors fit operational context, SLOs, and governance constraints. When anomaly detection is treated as part of the DevOps/SRE system—rather than a separate ML artifact—it can shorten time-to-detect, reduce alert fatigue, and improve engineer confidence [1], [11], [12].

The benchmarking results show that robust statistical baselines and classical machine learning approaches continue to deliver strong performance across a wide range of banking scenarios when combined with telemetry engineered around residuals, seasonality, and dependency context. Deep learning techniques provide benefits in high-dimensional or highly non-linear domains, but their increased operational cost, governance overhead, and explainability challenges restrict their applicability to the most critical services where incremental improvements justify additional complexity [2], [4], [6], [9]. These findings align with prior empirical studies and large-scale production experience in regulated environments [1], [5], [11].

Organizationally, AIOps adoption in banking behaves like a maturity curve, not a one-time tool rollout. Improvements in telemetry quality, clear service ownership, change intelligence, and governance controls typically deliver more reliability benefit than incremental model tuning. Promising directions include topology-aware and graph-based anomaly detection, policy-guarded closed-loop remediation, and standardized evaluation protocols for regulated domains where data sharing is constrained. Over the long run, sustainable anomaly detection depends on balancing automation with transparency so faster recovery does not come at the expense of compliance, auditability, or trust [3], [8], [10], [12], [13].

References

[1] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," PLOS ONE, vol. 11, no. 4, pp. 1–31, 2016.
[2] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in Proc. IEEE Int. Conf. Data Mining (ICDM), Miami, FL, USA, 2008, pp. 413–422.
[3] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," Neural Computation, vol. 13, no. 7, pp. 1443–1471, 2001.
[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.
[5] R. P. Adams and D. J. C. MacKay, "Bayesian online changepoint detection," arXiv:0710.3742, 2007.
[6] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in Proc. Int. Conf. Learning Representations (ICLR), San Diego, CA, USA, 2014.
[7] A. Agade and S. Balpande, “Exploring the Non-Medical impacts of Covid-19 using Natural Language Processing,” International Journal of Computer Applications, vol. 175, no. 36, pp. 16–23, Dec. 2020, doi: 10.5120/ijca2020920923.
[8] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in Advances in Neural Information Processing Systems (NeurIPS), Vancouver, BC, Canada, 2019.
[9] C. M. Bishop, Pattern Recognition and Machine Learning. New York, NY, USA: Springer, 2006.
[10] N. G. Leveson, Engineering a Safer World: Systems Thinking Applied to Safety. Cambridge, MA, USA: MIT Press, 2011.

[11] B. Beyer et al., Site Reliability Engineering: How Google Runs Production Systems. Sebastopol, CA, USA: O'Reilly Media, 2016.
[12] M. Chen, A. S. Bhandari, and J. Li, "The emergence of AIOps: Opportunities and challenges," IEEE Access, vol. 8, pp. 179698–179712, 2020.
[13] J. Postel and J. Reynolds, "Common event expression (CEE)," Internet Engineering Task Force (IETF), RFC 8425, 2018.