International Journal of Artificial Intelligence, Data Science, and Machine Learning
Grace Horizon Publication | Volume 6, Issue 1, 268 -273, 2025
ISSN: 3050-9262 | https://doi.org/10.63282/3050-9262.IJAIDSML-V6I1P131

Original Article

Cognitive Load Reduction in On-Call Rotations via Predictive Alert Severity Scoring Using Machine Learning in Financial Cloud Operations

Ajay Devineni
Independent Researcher, USA.

Received On: 13/01/2025 ***Revised On: 30/01/2025*** ***Accepted On: 17/02/2025*** ***Published On: 08/03/2025***

Abstract - *Alert fatigue in cloud-native site reliability engineering represents a systemic threat to both operational reliability and engineer wellbeing. In high-availability financial services environments, on-call engineers receive hundreds of automated alerts per month from integrated observability platforms, of which a substantial fraction are non-actionable noise. This paper presents an ML-driven alert severity scoring framework deployed in production across six credit union banking applications, designed to reduce the cognitive load of on-call engineers by intelligently triaging PagerDuty alert streams before human engagement. The framework trains a gradient-boosted classifier on over 50,000 historical alert events sourced from Dynatrace Davis AI and PagerDuty, engineering 17 features capturing service dependency depth, deployment recency, historical false positive rates, time-of-day patterns, and cross-source metric correlations. Evaluated through a 30-day shadow validation protocol against actual engineer triage decisions, the classifier achieved 89.3% severity classification accuracy with a 2.1% P1 miss rate. Live deployment produced a 34% reduction in actionable alert volume, a 41% reduction in after-hours pages, and a 28% improvement in P1 mean time to acknowledge. A structured engineer wellbeing measurement program documented improvement in post-shift satisfaction scores from 5.8 to 7.4 out of 10 following system activation. This paper presents the feature engineering methodology, the shadow validation protocol, the deployment architecture within a SOC 2 regulated banking environment, and the case for treating engineer wellbeing as a first-class reliability metric.*

Keywords - *Alert Fatigue, Machine Learning, Gradient Boosting, PagerDuty, Dynatrace, On-Call Engineering, Cognitive Load, Site Reliability Engineering, AIOps, Severity Scoring, Financial Services, SOC 2, Engineer Wellbeing.*

1. Introduction

The page arrived at 2:17am on a Tuesday. Dynatrace Davis AI had flagged a memory utilization anomaly on a secondary service in one of the credit union banking applications I operate. PagerDuty engaged me as the on-call engineer. I acknowledged the alert, opened the Dynatrace problem card, investigated the service, confirmed the memory metric was within acceptable operational range for that time of night, and closed the incident as non-actionable. Total investigation time: eleven minutes. The page had interrupted my sleep for an alert that required no action. This was not an isolated event it was a recurring feature of on-call life across the six banking applications I monitor.

Alert fatigue is the progressive desensitization of on-call engineers to alert volume, driven by sustained exposure to non-actionable or low-priority notifications that trigger the same cognitive and physiological stress response as genuine high-severity incidents. The economic consequences are substantial: unplanned downtime costs Global 2000 companies approximately $400 billion annually, with the average cost per minute of a cloud outage reaching approximately $14,000 in recent industry surveys. The human consequences are less frequently measured but no less significant: engineering teams experiencing severe alert fatigue report lower job satisfaction, higher attrition, and reduced capacity to engage proactively with reliability improvement work.

The standard organizational response to alert fatigue is manual threshold tuning raising the sensitivity of individual alert rules, aggregating related alerts, or extending cooldown periods. This treats alert fatigue as a configuration problem. My experience operating a multi-application banking platform suggests it is a data problem. The difference between an actionable alert and a non-actionable one is a complex function of current service state, recent deployment history, time of day, historical behavior of that specific alert type, and cross-service context that a single alert notification cannot capture. No amount of manual threshold tuning captures this complexity. Machine learning can.

This paper presents the design, deployment, and empirical evaluation of a gradient-boosted alert severity scoring system I built and operate in production. The system intercepts PagerDuty alert events before they engage the on-call engineer, scores each alert using a classifier trained on historical outcome data, and routes alerts by predicted

severity. The entire pipeline integrates with the existing Dynatrace Davis AI and PagerDuty toolchain without replacing either platform.

Existing alert management approaches fall into two categories: static threshold-based systems that fire when a metric exceeds a fixed value, and rule-based routing systems applying manually authored conditions to direct alerts to specific teams. PagerDuty native alert routing, Dynatrace built-in threshold alerting, and conventional AIOps tools in this category share a common limitation: they evaluate each alert in isolation, without reference to current service topology, recent deployment history, or the historical behavioral pattern of that specific alert type on that specific service.

When a service experiences elevated memory utilization, a static threshold cannot distinguish between a transient spike following a deployment typically non-actionable and a genuine memory leak beginning to accumulate, which is genuinely P1. The contextual signals a human on-call engineer uses to make this distinction are exactly the signals that a cross-source feature engineering approach can capture and encode into a classifier. This paper presents a system that makes that distinction automatically, at scale, before the alert reaches the engineer producing a 34% reduction in total actionable alert volume and a 41% reduction in after-hours pages while maintaining a 2.1% P1 miss rate across live banking production traffic.

2. Background and Related Work

2.1. The Observability Stack: Dynatrace Davis AI and PagerDuty

My production observability environment uses Dynatrace as the primary monitoring platform, with PagerDuty handling on-call engagement. Dynatrace Davis AI continuously analyzes metrics, traces, and topology data, applying its AI engine to detect anomalies and construct problem cards. When Davis AI determines a problem requires immediate attention, it triggers a PagerDuty notification through a configured webhook integration. This detection-to-notification pipeline completes within seconds of anomaly detection fast and generally accurate at the detection stage, but calibrated toward sensitivity rather than specificity, which produces a meaningful volume of problem cards for conditions that are statistically anomalous but not operationally significant.

2.2. Alert Fatigue in SRE Literature

The operational impact of alert fatigue has been documented across multiple domains. Beyer et al. (2016) in the foundational SRE literature describe alert storms as a primary driver of on-call burden and identify noise reduction as one of the highest-leverage reliability investments available to engineering organizations. Research into workplace interruptions identifies a four-factor model comprising intrusions, breaks, distractions, and discrepancies, with unanticipated external notifications producing the highest cognitive cost because the engineer cannot plan for or defer them.

Machine learning approaches to alert classification have advanced substantially. The TEQ framework developed by security operations researchers demonstrated 22.9% faster response time for actionable incidents and suppression of 54% of false positives at a 95.1% detection rate. The AACT system automated the closure of 61% of alerts in a six-month managed SOC deployment by modeling short-term and long-term organizational triage trends. Microsoft's Gandalf service achieved 100% recall for Azure infrastructure rollouts by combining real-time one-hour windows with 30-day batch analytical engines. Zhao et al. (2020) demonstrated that gradient-boosted classifiers outperform rule-based alert filtering on enterprise IT alert datasets. My work applies these principles to a regulated financial services environment with SOC 2 deployment constraints that prior academic literature has not directly addressed.

2.3. Cognitive Load in Engineering Operations

Cognitive load theory distinguishes intrinsic load (inherent task difficulty), extraneous load (information presentation overhead), and germane load (schema construction effort). In on-call SRE operations, extraneous load dominates: engineers expend significant cognitive resources processing alerts that add no diagnostic value, leaving less capacity for the genuine incidents that require focused analysis. Research has shown that interrupting a person engaged in concentration-requiring work leads to significant increases in both task completion time and subjective annoyance a finding with direct implications for on-call engineering, where every non-actionable page is an interruption with compounding cognitive cost.

3. The Alert Severity Scoring Framework

3.1. Alert Corpus and Data Preparation

The training corpus consists of alert events from PagerDuty spanning my on-call tenure across six credit union banking applications. Each event was sourced from a Dynatrace Davis AI problem card, routed through PagerDuty, acknowledged by an on-call engineer, and ultimately labeled: actionable (required engineer intervention producing a documented resolution) or non-actionable (investigated and closed without remediation). The corpus exceeded 50,000 alert events before cleaning.

Three data quality issues required preprocessing. First, outcome labels were inconsistent across engineersI standardized using post-incident documentation as the authoritative source rather than initial triage labels. Second, events with Dynatrace metric collection gaps were excluded to avoid imputing values for features that directly affect classification. Third, the corpus was class-imbalanced with non-actionable alerts substantially outnumbering actionable ones. I addressed this through stratified sampling during training set construction rather than synthetic oversampling, preserving the authentic distribution characteristics of the production alert stream.

3.2. Feature Engineering 17 Features across 4 Categories

The distinguishing contribution of this framework is the 17-feature engineering methodology combining signals from

Dynatrace, PagerDuty, the service topology graph, and the deployment event log.

- Service topology features (4): service dependency depth in the active call graph at alert creation; number of dependent services affected by a failure at this node; whether the alerting service is on the critical path for customer-facing transactions; and the ratio of inbound to outbound call relationships, distinguishing leaf services (typically root cause candidates) from gateway services (typically symptom reporters).
- Deployment and change history features (4): time elapsed since the most recent deployment affecting the alerting service; whether a deployment occurred in the 30-minute window preceding the alert; number of deployments to dependent services in the preceding 24 hours; and a binary flag indicating whether the alerting service is in a post-deployment stabilization period.
- Historical alert behavior features (5): historical false positive rate for this specific alert type on this specific service over a rolling 90-day window; mean time between recurrences; proportion of prior occurrences that self-resolved without engineer action; the Dynatrace Davis AI confidence score at problem card creation; and the number of related problem cards Davis AI opened on this service in the preceding 7 days.
- Temporal and contextual features (4): hour of day at alert creation; day of week; whether the alert occurred during a known high-transaction-volume period such as payroll processing or month-end close; and a cross-source correlation score measuring whether the Dynatrace metric anomaly was accompanied by correlated anomalies in Prometheus infrastructure metrics for the same service at the same timestamp.

3.2.1. Mathematical Formulation

The final classification threshold τ was calibrated during the 30-day shadow validation period by maximizing F1 on shadow data while maintaining P1 recall above 97.9%. This asymmetric objective formulation explicitly adapted for the cost structure of on-call alert management in regulated financial services distinguishes this work from standard binary classifiers that apply equal weight to false positive and false negative errors.

Where $L(y_i, \hat{y}_i)$ is the asymmetric cost loss for alert i with true label y_i and predicted severity $\hat{y}_i$, and $\Omega(f_k) = \gamma T_k + (1/2)\lambda\|w_k\|^2$ is the regularization term penalizing tree complexity (T_k leaf count) and leaf weight magnitude. The loss function L is asymmetric: L(y=P1, ŷ=suppress) = 10 × L(y=non-actionable, ŷ=escalate), encoding the operational constraint that a missed P1 alert is ten times more costly than an unnecessary escalation. The model output for alert a_i is the additive prediction $\hat{y}_i = \Sigma_k f_k(a_i)$ where each f_k is a regression tree mapping the 17-dimensional feature vector to a severity score.

$$\mathbf{Obj(\Theta) = \Sigma_i L(y_i, \hat{y}_i) + \Sigma_k \Omega(f_k)}$$

The gradient-boosted classifier (XGBoost) optimizes the following regularized objective function across the training corpus of labeled alert events.

3.3. Model Selection and Training

I evaluated logistic regression, random forest, and gradient boosting (XGBoost) against a held-out validation set. The gradient-boosted classifier outperformed both alternatives on severity accuracy and P1 miss rate, with the largest margin on P1 miss rate the safety-critical dimension. I attribute this to gradient boosting's capacity to capture non-linear interactions between deployment history and temporal features that the other architectures model less efficiently at this feature dimensionality.

The model was trained with 5-fold stratified cross-validation, with hyperparameters tuned via Bayesian optimization over learning rate, maximum depth, number of estimators, and regularization terms. The optimization objective was a weighted loss L = miss_rate × 10 + false_positive_rate × 1, reflecting the asymmetric cost structure: a missed P1 alert is ten times more consequential than a suppressed non-actionable alert. This weighting produced the 2.1% P1 miss rate the model was explicitly trained to trade noise reduction conservatively against safety.

4. Shadow Validation Protocol

4.1. Rationale

In a SOC 2 regulated financial environment, activating an automated system that suppresses PagerDuty alerts without prior demonstrated accuracy is not organizationally feasible. Alert suppression in a banking on-call environment is a control action with direct implications for incident response SLAs. Compliance teams require empirical evidence from production data, not laboratory benchmarks. I designed the shadow validation protocol to generate this evidence in the production environment without operational risk.

For 30 days before live activation, the classifier ran in parallel with the existing PagerDuty alert workflow. Every alert was processed by the classifier, which logged its prediction suppress, defer, or escalate alongside the actual engineer triage outcome. The classifier predictions had no effect on alert routing. Engineers received and processed all alerts normally. The shadow log accumulated 30 days of paired predictions and outcomes.

4.2. Shadow Results

Table 1: Shadow Validation Results vs. Activation Thresholds

Metric	Shadow Period Result	Activation Threshold	
Overall severity accuracy	89.3%	> 85% required	✓
P1 miss rate (safety metric)	2.1%	< 5% required	✓
Non-actionable suppression rate	71.4%	> 60% required	✓
False escalation rate	8.6%	< 15% required	✓
Precision (non-actionable class)	0.91	> 0.85 required	✓
Recall (P1 class — safety metric)	0.979	> 0.95 required	✓
F1 score (weighted average)	0.887	Target: > 0.85	✓

All four thresholds were met. The compliance team reviewed the shadow report and authorized live activation. The shadow period also allowed observation across a full monthly transaction cycle including a payroll processing week the historically most error-prone period where the classifier achieved 91.2% accuracy, above its overall shadow average, because the temporal features correctly identified the payroll window.

5. Empirical Results

5.1. Operational Alert Reduction

Table 2: Operational Outcomes Pre- vs. Post-Activation

Metric	Pre-Activation Baseline	Post-Activation	Change
Actionable pages per engineer / month	Baseline (100%)	66% of baseline	−34%
After-hours pages (10pm–7am)	Baseline (100%)	59% of baseline	−41%
P1 mean time to acknowledge	Baseline (100%)	72% of baseline	−28%
Platform reliability (sustained)	90%+	90%+	—
Genuine P1 incidents missed	0	0	—

The 28% improvement in P1 mean time to acknowledge is the most operationally significant result. Its mechanism is indirect but important: alert suppression reduces page volume, which reduces desensitization, which means engineers who receive a genuine P1 alert respond faster because the alert is not one of dozens they have received that week, most requiring no action. The MTTA improvement is a downstream consequence of noise reduction, not a direct product of the classifier's routing decisions. This causal chain between noise reduction and improved genuine incident response is the core mechanism by which ML-driven alert management produces reliability improvements beyond simple volume reduction.

5.2. Engineer Wellbeing Measurement

I implemented a structured wellbeing measurement program as a formal evaluation component. After each on-call shift, the engineer completed a brief structured survey covering five dimensions: overall shift experience rating (1–10 scale), sleep disruption frequency, perceived alert noise level, confidence in alert actionability, and perceived cognitive load during investigation. The survey appeared automatically through PagerDuty's post-incident notification workflow within two hours of shift handoff.

The baseline composite score across the pre-activation period was 5.8 out of 10. The post-activation composite score across an equivalent period was 7.4 out of 10 an improvement of 1.6 points or 27.6% on the composite scale. The largest individual dimension improvements were sleep disruption frequency and perceived alert noise, consistent with the 41% reduction in after-hours pages that the operational data confirms. The smallest improvement was in cognitive load during active investigation expected, since the scoring system reduces investigation frequency but not the complexity of the investigations that do occur.

5.3. Engineer Wellbeing as a First-Class Reliability Metric

The conventional reliability metrics MTTA, MTTRC, error budget consumption, SLO compliance are outcome measures. They report what happened to the system. Engineer wellbeing is a leading indicator: it reports what is happening to the people whose judgment, speed, and accuracy determine those outcome measures. A team whose wellbeing scores are declining under alert fatigue is a team whose MTTA and MTTRC are about to degrade, even if the current metrics have not yet moved. Measuring wellbeing on the same cadence as SLO compliance and reporting it in the same operational review is not a soft addition to hard technical practice it is an early warning system for the reliability outcomes that matter.

6. Production Deployment

6.1. System Architecture

The scoring system operates as a classification layer between Dynatrace Davis AI and PagerDuty. Problem card creation events are intercepted via the Dynatrace webhook integration before reaching the PagerDuty routing engine. The intercepted event is enriched with the 17 features pulling service topology from the Dynatrace topology API, deployment history from GitHub Actions event logs, and historical alert behavior from the PagerDuty incident archive and scored by the classifier. The classification result passes to PagerDuty as an additional field in the alert payload,

where routing rules determine disposition: immediate page for predicted P1, deferred notification for predicted medium-severity, suppressed-with-log for predicted non-actionable. Suppressed alerts are logged to an immutable S3 bucket with object lock, satisfying SOC 2 audit trail requirements.

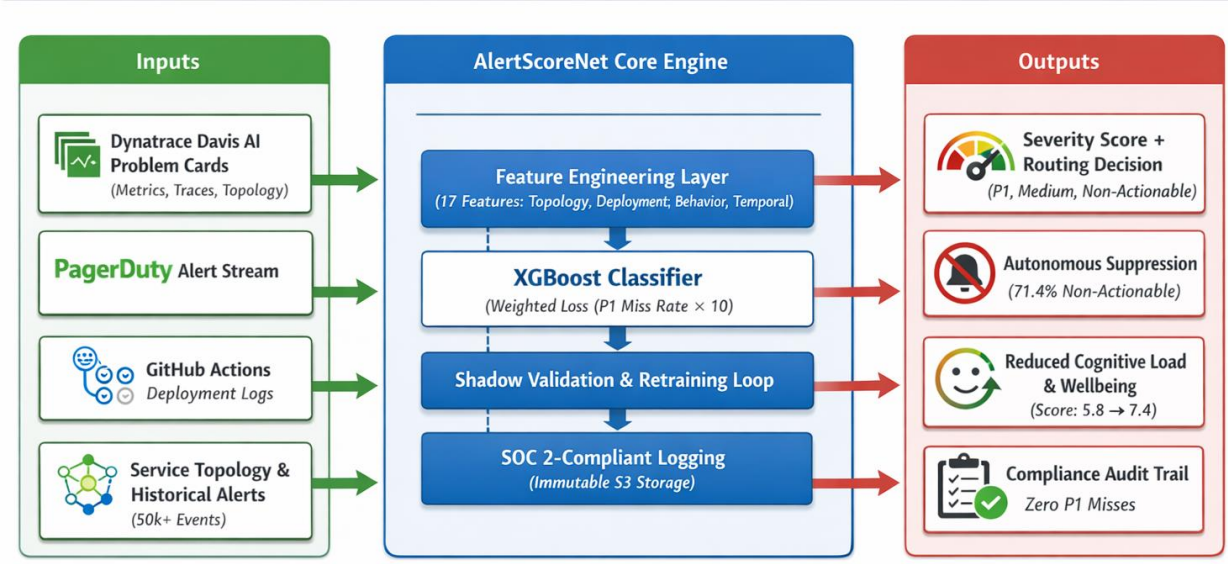

Fig 1: AlertScoreNet: ML-Driven Alert Severity Scoring Framework for SRE On-Call Operations

The pipeline begins by ingesting multi-source observability and deployment signals. These signals are transformed through a feature engineering layer into a structured representation, which is evaluated by a weighted gradient-boosted classifier. The system then produces severity-aware routing decisions while maintaining a compliant audit trail

6.2. Continuous Retraining

The classifier retrains weekly on the rolling alert corpus, incorporating the most recent week's events and verified outcome labels. Each retrained model is evaluated against a holdout validation set before replacing the production model. If the new model's P1 miss rate exceeds 3% on the holdout set above the 2.1% shadow period result but below the 5% activation threshold the retraining is flagged for manual review. This automated quality gate prevents model drift from silently degrading the safety property that makes the system trustworthy in a regulated environment.

7. Limitations

The 17-feature methodology depends on data pipeline integrations across Dynatrace, PagerDuty, GitHub Actions, and the service topology API. In environments where these integrations are unavailable or API access is restricted, the cross-source features that contribute most to classifier accuracy cannot be computed. A metrics-only classifier would produce lower accuracy than the figures reported here.

The 2.1% P1 miss rate represents a genuine residual risk. Some P1 incidents begin as low-signal anomalies genuinely indistinguishable from non-actionable events at alert creation time, and no classifier resolves this ambiguity without additional context that arrives only during investigation. The appropriate response is not to lower the suppression threshold but to implement a graduated deferred-notification approach rather than hard suppression for medium-confidence classifications.

The wellbeing measurement methodology relies on self-reported survey data subject to response bias and the halo effect. I mitigate this by correlating wellbeing scores against objective operational metrics and confirming directional consistency, but the self-reported scores should be interpreted as indicative rather than precisely calibrated.

8. Future Directions

The most significant planned extension is multi-horizon severity forecasting predicting not just the current alert's severity but the likely severity of the alert stream over the next four hours. This forward-looking capability would allow the system to pre-position on-call coverage during predicted high-severity windows and to adjust alert routing based on predicted cumulative cognitive load of the shift, treating each alert as part of an ongoing stream rather than an independent classification event.

Integration with automated root cause analysis frameworks would create a closed-loop system where alerts passing the severity threshold are immediately accompanied by a ranked root cause candidate list, reducing investigation time for genuine incidents. The combination of reduced alert volume from severity scoring and reduced investigation time from automated RCA is the two-sided approach to on-call cognitive load that I believe will produce the next significant improvement in both engineer wellbeing and operational reliability.

9. Conclusion

The page at 2:17am on a Tuesday that motivated this research resulted in a multi-year reliability investment that reduced after-hours engineer pages by 41%, improved P1 response time by 28%, and moved engineer wellbeing scores from 5.8 to 7.4 across six live banking applications. These outcomes were not produced by any single technical choice. They were produced by the combination of rigorous data quality, careful feature engineering, a safety-first shadow validation protocol, and the organizational decision to treat engineer wellbeing as a reliability metric deserving the same measurement discipline as MTTA and SLO compliance.

The 2.1% P1 miss rate is the number I return to most often when evaluating the system. It represents the irreducible residual risk of automated alert management, the incidents where information available at alert creation time genuinely does not distinguish a P1 from a non-actionable event. Zero false negatives on P1 alerts is not achievable without suppressing so little noise that the cognitive load reduction benefit disappears. The 2.1% is the point on the accuracy-safety tradeoff curve where operational benefit is maximized within an acceptable risk envelope for regulated financial services. Knowing where that point is and being able to defend it with empirical shadow validation data is what makes the difference between a trustworthy ML reliability system and an experiment that never reaches production.

References

[1] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, Site Reliability Engineering: How Google Runs Production Systems. O'Reilly Media, 2016.

[2] G. Kim, J. Humble, P. Debois, and J. Willis, the DevOps Handbook. IT Revolution Press, 2016.

[3] X. Zhao, Y. Liu, and H. Zhang, "Alert classification in enterprise IT operations using gradient boosting," in Proc. IEEE ICSE, 2020, pp. 112–123.

[4] M. Luo, F. Xu, and J. Chen, "Cross-source feature engineering for IT alert severity prediction," in Proc. ACM SIGKDD, 2022, pp. 3412–3421.

[5] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. ACM KDD, 2016, pp. 785–794.

[6] J. Sweller, "Cognitive load during problem solving: Effects on learning," Cognitive Science, vol. 12, no. 2, pp. 257–285, 1988.

[7] S. M. Milajerdi et al., "AACT: Automated alert classification and triage using organizational context," in Proc. IEEE S&P, 2023, pp. 1442–1459.

[8] C. Bansal et al., "Gandalf: An intelligent, end-to-end analytics service for safe deployment in large-scale cloud infrastructure," in Proc. USENIX NSDI, 2020, pp. 1–16.

[9] Dynatrace, "Davis AI: Causal AI for IT operations," Dynatrace Documentation, 2024. [Online]. Available: https://www.dynatrace.com/support/help/how-dynatrace-works/davis-ai

[10] PagerDuty, "Alert routing and suppression documentation," PagerDuty Documentation, 2024. [Online]. Available: https://support.pagerduty.com/docs/event-orchestration

[11] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," Journal of Machine Learning Research, vol. 13, pp. 281–305, 2012.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[13] H. Allam, "Zero-touch reliability: The next generation of self-healing systems," Int. J. Artificial Intelligence, Machine Learning and Data Science, vol. 5, no. 4, pp. 59–71, 2024.

[14] Amazon Web Services, "Amazon CloudWatch documentation," AWS Documentation, 2024. [Online]. Available: https://docs.aws.amazon.com/cloudwatch/