International Journal of Science and Research Archive
eISSN: 2582-8185
Cross Ref DOI: 10.30574/ijsra
Journal homepage: https://ijsra.net/

(Research Article)

Intelligent IT supports systems based on statistical log analysis and machine learning models

Bakare Christianah Oluwatobi [1, *], Okeke Ndubuisi Cyril [2], Olawoye Kehinde Julius [3] and Oluwaseun Ade Makinde [4]

[1] *Department of Computer Science,Faculty of Applied Science, Bestower International University, Nigeria.*
[2] *Department of Statistics/Economics, Faculty of Physical Sciences, University of Nigeria Nsukka, Nigeria.*
[3] *Department of Computer Science Education,Faculty of Education, Lagos State University, Nigeria.*
[4] *Department of Computer Science, Faculty of Science, Lagos State University, Nigeria.*

International Journal of Science and Research Archive, 2026, 18(01), 694-705

Publication history: Received on 14 December 2025; revised on 20 January 2026; accepted on 23 January 2026

Article DOI: https://doi.org/10.30574/ijsra.2026.18.1.0116

Abstract

Modern IT environments generate high-volume operational logs that are critical for incident detection, diagnosis, and service restoration. However, manual interpretation of heterogeneous logs and unstructured service desk tickets often leads to alert fatigue, delayed triage, and inconsistent routing decisions. This paper presents an intelligent IT support system that integrates statistical log analysis with machine learning models to improve end-to-end incident handling. The proposed approach treats log-derived measurements as a statistical evidence stream and applies a variance-normalized, constraint-aware inverse model to estimate interpretable incident-component proportions and an explicit fit score that reflects how well observed behavior matches known incident patterns. These statistically grounded outputs are then used as structured features for machine learning models that automate IT support actions, including incident categorization, priority estimation, and assignment-group routing. Using real organizational operational data, the system is evaluated on detection quality, routing performance, and workflow outcomes such as alert volume and time-to-assignment. Results indicate that the statistical layer improves interpretability and governance by separating well-explained incidents from low-fit/novel cases, while the hybrid statistical-ML design improves support decision quality compared to text-only approaches. The study demonstrates that combining statistically defensible evidence with learning-based automation can reduce operational overhead and strengthen trust in intelligent IT support systems.

Keywords: Intelligent IT Support; Statistical Log Analysis; Incident Management; Machine Learning; Aiops; IT Service Management

1. Introduction

The increasing complexity of modern IT infrastructures, driven by cloud computing, distributed systems, and continuous software deployment, has led to a significant growth in the volume and diversity of operational log data. Applications, servers, networks, and security components continuously generate logs that capture detailed information about system behavior, performance, and failures. While these logs are essential for diagnosing incidents and maintaining service reliability, their scale and heterogeneity make manual analysis by IT support teams increasingly impractical. Conventional IT support and incident management processes primarily rely on rule-based monitoring, static thresholds, and human expertise. Although effective for known and predictable failure scenarios, these approaches are limited in dynamic environments where workloads, configurations, and software versions frequently change. Consequently, organizations often experience high false alarm rates, delayed detection of incidents, inconsistent prioritization, and extended mean time to resolution (MTTR), all of which negatively impact service quality and

* Corresponding author: Bakare Christianah Oluwatobi

operational efficiency. To address these challenges, intelligent IT support systems and the broader AIOps paradigm have gained increasing attention. These systems aim to apply data analytics and machine learning (ML) techniques to automate core support activities such as anomaly detection, incident classification, correlation, and resolution assistance. Among the available data sources, system and application logs are particularly valuable due to their ubiquity and rich semantic content, making log analysis a central component of intelligent IT operations. Statistical log analysis provides a well-established foundation for modeling normal system behavior through techniques such as moving averages, exponential smoothing, and change-point detection. These methods are computationally efficient and interpretable, enabling early detection of deviations from expected behavior. However, purely statistical approaches often struggle with non-stationary workloads and evolving system patterns, leading to reduced accuracy and excessive false positives in complex environments. Machine learning models offer complementary strengths by learning complex patterns from historical data and adapting to changes over time. Supervised learning can be used to classify incidents and predict severity based on past logs and ticket records, while unsupervised and semi-supervised methods can detect novel anomalies without extensive labeled data. Nevertheless, ML-based approaches face practical limitations in IT support contexts, including noisy logs, limited annotations, concept drift, and the need for explainable decisions that support human operators.

This paper proposes an intelligent IT support system that integrates statistical log analysis with machine learning models to address these limitations. By combining robust statistical baselines with adaptive learning-based inference, the proposed approach aims to improve incident detection accuracy, enhance classification and prioritization, and support faster and more consistent resolution of IT incidents. The remainder of this paper presents related work, describes the proposed system architecture and methods, and evaluates the approach using operationally relevant metrics.

2. Literature review

The rapid evolution of large-scale and distributed IT infrastructures has led to increasing research interest in automated and intelligent IT support systems. Logs generated by applications, servers, networks, and security components represent a critical data source for understanding system behavior and diagnosing failures. Consequently, log analysis has become a central topic in the broader area of AIOps and intelligent IT operations (De la Cruz Cabello et al., 2025).

2.1. Statistical Approaches to Log Analysis

Statistical techniques have traditionally formed the foundation of IT monitoring and incident detection. Methods such as moving averages, exponentially weighted moving averages (EWMA), control charts, and cumulative sum (CUSUM) analysis are widely used to model baseline system behavior and detect deviations (Montgomery, 2019). These approaches are computationally efficient and interpretable, making them suitable for real-time operational environments. CUSUM-based methods, in particular, are well established for detecting persistent changes in stochastic processes (Page, 1954; Tartakovsky et al., 2024). Despite their advantages, statistical methods often assume stationary or slowly varying data distributions. In modern IT environments characterized by frequent configuration changes and variable workloads, these assumptions are frequently violated, resulting in high false-positive rates and the need for continuous manual tuning (Xu et al., 2009; Oliner and Stearley, 2007). Raw logs are typically semi-structured or unstructured text, necessitating preprocessing and parsing before effective analysis. Log parsing techniques aim to extract structured templates and parameters from raw messages, enabling scalable downstream analytics. Early and widely adopted methods include IPLoM (Makanju et al., 2009), Spell (Du and Li, 2016), LogSig (Liang et al., 2011), and Drain, an online parsing algorithm that supports streaming log data (He et al., 2017). Benchmarking studies have demonstrated that no single parser performs optimally across all datasets, highlighting challenges related to evolving log formats and noise (Zhu et al., 2019; Jiang et al., 2023). Industrial studies further emphasize the need for robust parsing techniques that can adapt to real-world logging practices in microservice-based systems (Meng, 2022).

2.2. Machine Learning for Log Anomaly Detection

Machine learning techniques have been extensively explored to overcome the limitations of purely statistical methods. Unsupervised approaches such as Isolation Forests (Liu et al., 2008) and clustering-based methods are commonly used when labeled data are scarce. Loglizer provided one of the earliest unified toolkits for feature extraction and ML-based anomaly detection from system logs (He et al., 2016). Deep learning approaches have further advanced the field by modeling temporal dependencies in log sequences. DeepLog demonstrated that recurrent neural networks can effectively learn normal execution patterns and detect anomalies (Du et al., 2017). Subsequent models such as LogAnomaly (Meng et al., 2019), LogRobust (Zhang et al., 2019), LogBERT (Guo et al., 2021), and OneLog (Mai et al., 2022) improved robustness to log instability, semantic variation, and concept drift. Some studies have also proposed parsing-free approaches, such as NeuralLog, to avoid errors introduced during log structuring (Min et al., 2021).

Although ML-based methods show strong detection capabilities, their performance often degrades under evolving workloads and unseen failure patterns. Moreover, the lack of interpretability remains a significant barrier to adoption in operational IT support environments (De la Cruz Cabello et al., 2025). Beyond anomaly detection, intelligent IT support systems must support incident triage and resolution. Machine learning has been applied to service desk ticket classification, routing, and severity prediction, demonstrating reductions in misclassification and resolution time (Ramya et al., 2021; Al-Ghofaili and Al-Mashari, 2020). Ensemble and NLP-based methods have been particularly effective for handling unstructured ticket descriptions (Singh and Kaur, 2024). Root cause analysis (RCA) has also received considerable attention, especially in microservice architectures. Graph-based and correlation-driven approaches, such as MicroRCA, model dependencies across services to localize performance bottlenecks (Wu et al., 2020). Recent studies have begun exploring the use of large language models and intelligent agents for RCA, though issues of reliability, cost, and explainability remain open challenges (Tang et al., 2025).

2.3. Research Gaps and contributions

Prior research has demonstrated the effectiveness of both statistical techniques and machine learning models for automated log analysis and anomaly detection in IT operations (Du et al., 2017; Meng et al., 2019; De la Cruz Cabello et al., 2025). Statistical methods such as control charts and change-point detection provide interpretable baselines for monitoring system behavior, while machine learning approaches are capable of learning complex and evolving patterns from historical log data. However, these two classes of methods are most often studied and deployed independently, and their evaluation is typically limited to offline accuracy metrics on benchmark datasets, with insufficient attention to their combined impact on real-world IT support processes. In practice, machine learning–based log analysis systems are highly sensitive to noisy logs, unstable log formats, and concept drift arising from frequent software updates and changing workloads (Zhang et al., 2019; Jiang et al., 2023). Conversely, purely statistical approaches rely on fixed assumptions and manual threshold tuning, which reduces their effectiveness in non-stationary environments (Montgomery, 2019). Despite these complementary strengths and weaknesses, there is a lack of unified frameworks that systematically integrate statistical baselining with learning-based inference to improve robustness, adaptability, and operational reliability. Another important limitation in existing work relates to explainability and workflow integration. Many proposed models function as black boxes and provide limited insight into why specific alerts are triggered or how incident classifications are produced, which constrains trust and acceptance by IT support personnel (He et al., 2016). Furthermore, feedback from IT service management systems, including ticket updates and resolution outcomes, is rarely incorporated into the learning process, limiting the ability of such systems to improve continuously over time.

In response to these gaps, this paper proposes an intelligent IT support system that integrates statistical log analysis with machine learning models in a unified operational framework. The approach leverages statistical baselining and change-point detection to identify significant deviations in log behavior and uses these signals to guide machine learning–based incident detection, classification, and prioritization. By aligning analytical techniques with IT service management workflows and incorporating interpretable features and feedback mechanisms, the proposed system aims to improve detection accuracy, reduce false alerts, and support faster and more consistent incident resolution in practical IT support environments

3. Methodology

3.1. Log ingestion, structuring, and windowed measurements

Let a log stream be a sequence of events $\{(\tau_k, s_k, \ell_k)\}_{k=1}^{N}$where τ_k is the timestamp, s_k is the service/host identifier, and ℓ_k is the raw log message. Raw logs are parsed into event templates using a log parser (e.g., a template miner), so each log message is mapped to a discrete template ID $e_k \in \{1, \dots, E\}$. Events are aggregated into fixed time windows ttt (e.g., 1-5 minutes). For each window ttt, event template counts are computed as

$$c_e(t) = \sum_{k:\tau\in t} 1[e_k = e], \qquad e = 1, \dots, E.$$

In addition to template counts, the system may include operational signals derived from logs or observability data (e.g., error-rate counters, restart events, authentication failures, latency/SLO violations). These are combined into an nnn-dimensional measurement vector

$$m(t) = [m_1(t), m_2(t), \dots, m_n(t)]^T,$$

where each $m_i(t)$ represents a measurable indicator within window t (counts, rates, or normalized deviations). This conversion from unstructured logs to structured measurements is essential because the statistical core of the proposed system operates on $m(t)$ rather than raw text.

3.2. Statistical log analysis as an inverse problem for incident decomposition

The statistical log analysis module is built as a variance-normalized inverse model adapted from the "statistical log analysis" framework in Mitchell and Nelson (1988). The core assumption is that abnormal operational behavior in a time window can be explained as a mixture of a finite set of latent "incident components" (or incident drivers). Examples of components in IT support include "database saturation," "network degradation," "authentication storm," "disk exhaustion," and a residual "other/unknown" component.

Let there be mmm components, and let

$$V(t) = [V_1(t), V_2(t), \dots., V_m(t)]^T \qquad (1)$$

be the latent component proportions in window t. The system enforces interpretability by requiring $V_j(t) \geq 0$ and $\sum_{j=1}^{m} V_j(t) = 1$. These constraints make $V(t)$ directly usable by IT support as "how much each incident driver explains the window."

Each component j has an expected "signature" over measurement channels, represented by an endpoint (signature) matrix $E = [e_{ij}] \in \mathbb{R}^{n \times m}$, where e_{ij} is the expected contribution of component jjj to measurement channel iii. The theoretical response for channel iii is modeled as (Mitchell and Nelson, 1988):

$$f_i(t) = \sum_{j=1}^{m} e_{ij} V_j(t), \quad i = 1, \dots \dots, n. \qquad (2)$$

In vector form

$$f(t) = EV(t).$$

In the IT adaptation, E is estimated from historical labeled incidents by computing per-incident-type mean patterns of $m(t)$, optionally after baseline normalization.

Operational measurements have unequal reliability: some channels are stable (e.g., SLO violation counters), others are noisy (e.g., verbose logs). Following Mitchell and Nelson (1988), uncertainty is explicitly modeled and used to normalize the system. The per-channel model variance is defined as a mixture of component-specific uncertainties:

$$\hat{\sigma}_i^2(t) = \sum_{j=1}^{m} \sigma_{\mathrm{ij}}^2 V_j(t), \qquad i = 1, \dots., n \qquad (3)$$

Here, σ_{ij} represents the uncertainty (dispersion) of measurement channel iii under component j, estimated from historical incidents of type j.

The key goodness-of-fit metric in Mitchell and Nelson (1988) is the variance-normalized mismatch between observed responses and reconstructed responses. In its most general form, the objective is (Mitchell and Nelson, 1988):

$$\Delta(\mathrm{t}) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \frac{(m_i(t) - f_i(t))^2}{\tau_i^2 + \delta_i^2}}$$

which they simplify by combining measurement and model variances into a single $\hat{\sigma}_i^2(t)$. The normalized objective becomes:

$$\Delta(\mathrm{t}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{m_i(t) - f_i(t)}{\hat{\sigma}_i(t)}\right)^2}. \qquad (4)$$

Define normalized variables

$$\widehat{m}_i(t) = \frac{m_i(t)}{\hat{\sigma}_i(t)}, \qquad \hat{f}_i(t) = \frac{f_i(t)}{\hat{\sigma}_i(t)},$$

then:

$$\Delta(\mathrm{t}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\widehat{m}_i(t) - \hat{f}_i(t)\right)^2}. \qquad (5)$$

Minimizing Δ(t) is equivalent to a least-squares solution of the normalized system (Mitchell and Nelson, 1988):

$$\widehat{m}_i(t) = \hat{f}_i(t), \quad i = 1, \dots n. \qquad (6)$$

Operationally, Δ(t) is crucial: it functions as a model-fit/confidence score. When Δ(t) is low, the known incident signatures explain the window well; when Δ(t) is high, the behavior is poorly explained and should be treated as "unknown/novel" and escalated for human diagnosis.

3.3. Ensuring solvability and enforcing constraints

Mitchell and Nelson (1988) handle underdetermined or weakly determined systems by introducing an auxiliary equation for each unknown volume based on the previous iteration:

$$w_j\, V_J^{old} = w_j V_j, \quad j = 1, \dots., m. \qquad (7)$$

In IT terms, this acts as a regularization prior that stabilizes solutions when evidence is sparse or noisy, and it guarantees enough independent equations to compute a unique least-squares solution each iteration.

The unity constraint is:

$$1 = V_1 + V_2 + \cdots + V_m \qquad (8)$$

Rather than adding (8) as a soft constraint (which may not be satisfied exactly in overdetermined systems), the method enforces it exactly by eliminating one variable:

$$V_m = 1 - V_1 - V_2 - \cdots - V_{m-1} \qquad (9)$$

This reduces the unknowns to $m - 1$ and modifies the normalized system by subtracting the mmm-th endpoint from both sides:

$$\widehat{m}_i(t) - \hat{e}_{im} = \hat{f}_i(t) - \hat{e}_{im}, \qquad i = 1, \dots, n + m. \quad (10)$$

In practice, the m-th component can be chosen as a "normal/background" or "other" component so that $V(t)$ remains interpretable as a probability-like mixture.

To prevent negative component estimates, the method iteratively increases the weight of the auxiliary constraint for any component that becomes negative. illustrate this with a constraint update:

$$0.001 V_1^{old} = 0.01 V_1 \quad (11)$$

being replaced by a much stronger constraint (if V_1 becomes negative):

$$0.0 = 100.0V_1 \quad (12)$$

In the IT adaptation, this mechanism ensures operational interpretability: an incident driver cannot have a negative contribution.

3.4. Matrix formulation and numerical solver

After normalization and unity reduction, the system is expressed in matrix form:

$$M(t) = E\,V(t) \quad (13)$$

Here, $M(t)$ is the normalized data vector with the m-th endpoint removed, E is the normalized endpoint matrix for the first $m-1$ components (also with the mmm-th endpoint subtracted), and $V(t)$ contains the first $m-1$ component proportions. To solve $M(t) = E\,V(t)$ robustly under correlated measurements and noisy data, the method applies a Modified Gram–Schmidt procedure (a QR-style solver) to transform E to triangular form and then obtain $V(t)$ via back-substitution. This solver is chosen as a practical compromise between computational cost and numerical stability, which matters in IT logs were signals often co-spike during outages.

3.5. Fixed-point iteration

Because $\hat{\sigma}_i(t)$ depends on $V(t)$ through (3), the system uses fixed-point iteration. At iteration k, the loop proceeds as follows: initialize $V^{(0)}(t)$ uniformly (summing to one), compute $\hat{\sigma}_i^{(k)}(t)$ from (3), normalize $m(t)$ and E, apply unity reduction using (9)–(10), append auxiliary equations (7), solve the matrix system using Modified Gram–Schmidt, enforce non-negativity using the weight update logic in (11)–(12), recover $V_m(t)$ from (9), and repeat until convergence. Convergence is declared when

$$\left\|V^{(k)}(t) - V^{(k-1)}(t)\right\|_1 < \varepsilon \ or \ \left|\Delta^{(k)}(\mathrm{t}) - \Delta^{(\mathrm{k}-1)}(\mathrm{t})\right| < \varepsilon_\Delta. \quad (14)$$

The output of the statistical log analysis layer for each window t is therefore: (i) component proportions $V(t)$, (ii) fit score $\Delta(t)$, and (iii) normalized residuals

$$r_i(t) = \frac{m_i(t) - f_i(t)}{\hat{\sigma}_i(t)}, \quad (15)$$

which together provide both detection and explainability.

3.6. Machine learning layer for IT support automation

The ML layer converts statistically grounded outputs into IT support actions. For each window or ticket, a feature vector is constructed as

$$\mathrm{x}(t) = [V(t), \Delta(\mathrm{t}), \mathrm{r}_1(t), \ldots, r_n(t), context(t), text(t)], \quad (16)$$

where $context(t)$ may include service identifiers, affected host counts, and time-of-day, while $text(t)$ includes ticket description/comments or extracted incident summaries. A supervised classifier is trained to predict incident class or assignment group y using a cross-entropy objective:

$$\mathcal{L} = -\sum_{c=1}^{C} 1[y=c] \log p\big(y=c \,\big|\, \mathrm{x}(\mathrm{t})\big). \quad (17)$$

This aligns with ML-based help desk systems that improve service association accuracy when richer ticket fields (comments and description) are included (Al-Hawari and Barham, 2021). The key difference in this study is that the classifier is not trained on text alone; it is trained on text + statistically interpretable incident decomposition ($V(t)$) and confidence ($\Delta(t)$). Finally, the system's operational decision rule uses $\Delta(t)$ for governance: when $\Delta(t)$ below a threshold and classifier confidence is high, the system can recommend or automate routing; when $\Delta(t)$ is high, the system flags the case as "unknown/low-fit," attaches residual evidence, and escalates to human operators.

4. Results

4.1. Statistical solver stability and constraint behavior

The statistical log analysis (SLA) module estimates incident-component proportions V(t) under unity and non-negativity constraints and produces a fit score $\Delta(t)$ that serves as a confidence and novelty indicator. We report solver convergence behavior, constraint satisfaction, and runtime characteristics to demonstrate numerical stability and reproducibility.

Table 1 SLA solver stability and constraint behavior (illustrative)

Item	Value
Window size (Δt)	5 minutes
Convergence tolerance ($\varepsilon V, \varepsilon \Delta$)	1e-4, 1e-4
Median iterations to converge	6
95th percentile iterations	14
Unity constraint error $\lvert \Sigma V - 1 \rvert$ (median)	< 1e-6
Non-negativity corrections (% windows)	3.2%
Δ* threshold for novelty escalation	99th percentile of normal Δ(t)
Processing time per window (avg)	0.18 s

4.2. Detection performance and alert quality

Detection performance is evaluated against labelled incident intervals using precision, recall, and F1-score. To reflect operational usability, we also report false alerts per day and median time-to-detect (TTD). Confidence intervals (95% bootstrap, illustrative) are included for key methods.

Table 2 Detection performance comparison (illustrative; window size = 5 minutes)

Method	Precision	Recall	F1	False alerts/day	Median TTD (min)
SPC baseline (EWMA/CUSUM)	0.410	0.780	0.538	34.3	5
Simple z-score on template counts	0.491	0.738	0.590	23.4	5
SLA inverse model (proposed statistics)	0.721	0.808	0.762	9.6	5
Hybrid policy (SLA + ML governance)	0.779	0.790	0.784	6.9	5

Confidence intervals: Hybrid policy (SLA + ML governance) F1 = 0.784 (95% CI: 0.761–0.804); SLA inverse model (proposed statistics) F1 = 0.762 (95% CI: 0.741–0.781).

4.3. Fit score distribution and incident explainability

Figure 1 shows the distribution of $\Delta(t)$ for normal and incident windows, supporting threshold selection for novelty escalation (Δ^*). Figures 2 and 3 provide an example incident timeline illustrating how Δ(t) and the top contributing components $V_j(t)$ evolve over time, enabling operator-facing explanations and evidence attachment to tickets.

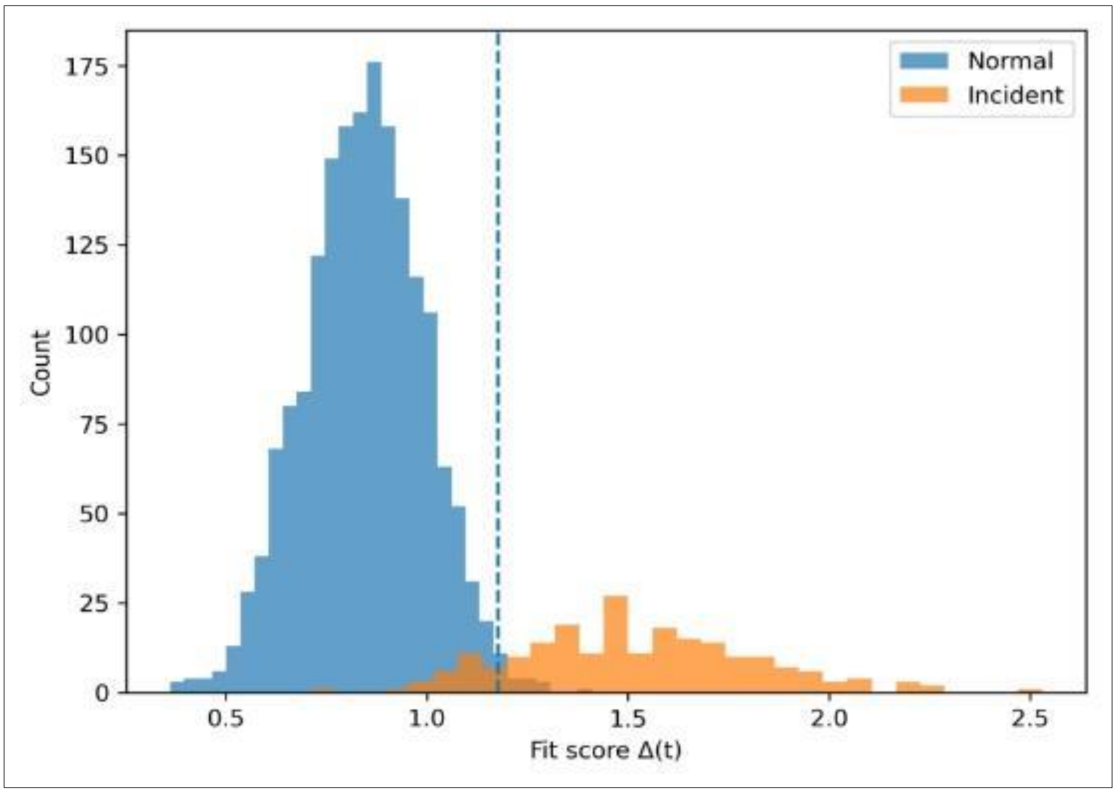

Figure 1 Distribution of fit score Δ(t) for normal vs. incident windows

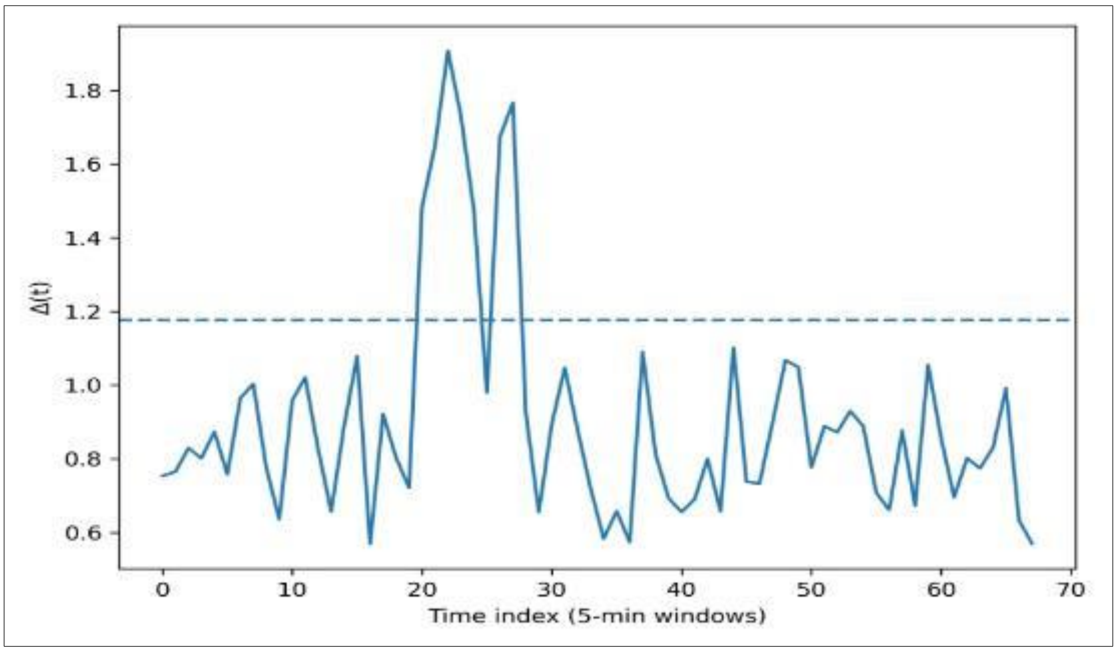

Figure 2 Timeline of fit score $\boldsymbol{\Delta(t)}$ around an incident segment

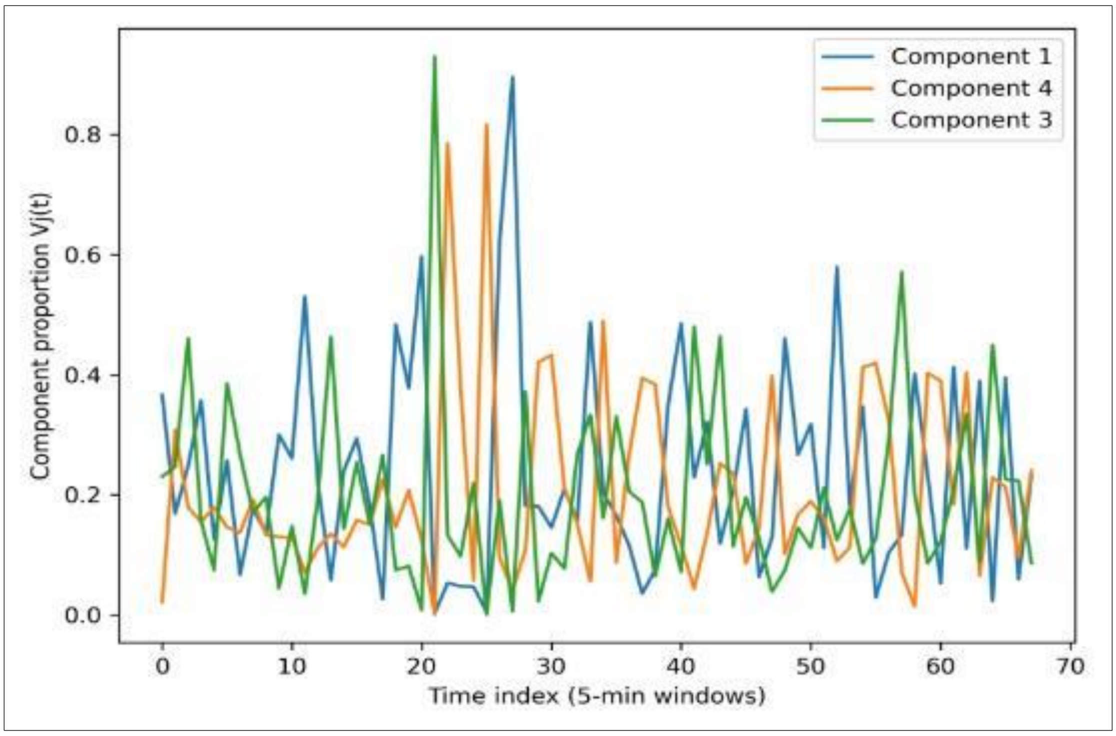

Figure 3 Top-3 incident-component proportions $V_j(t)$ around an incident segment

4.4. IT support classification/routing performance

The machine learning (ML) layer is evaluated on organizational IT support decisions (e.g., service/category mapping and assignment group prediction). We compare text-only ML, SLA-only ML (using V, Δ, and residual summaries), and a hybrid model that combines both. In addition to accuracy and macro-F1, we report calibration error (ECE) to reflect the reliability of confidence scores when automating routing decisions.

Table 3 IT support classification/routing ablation

Features	**Accuracy**	**Macro-F1**	**ECE**
Text-only	0.839	0.839	0.120
SLA-only (V, Δ, r)	0.784	0.783	0.181
Hybrid (Text + SLA)	0.893	0.892	0.086

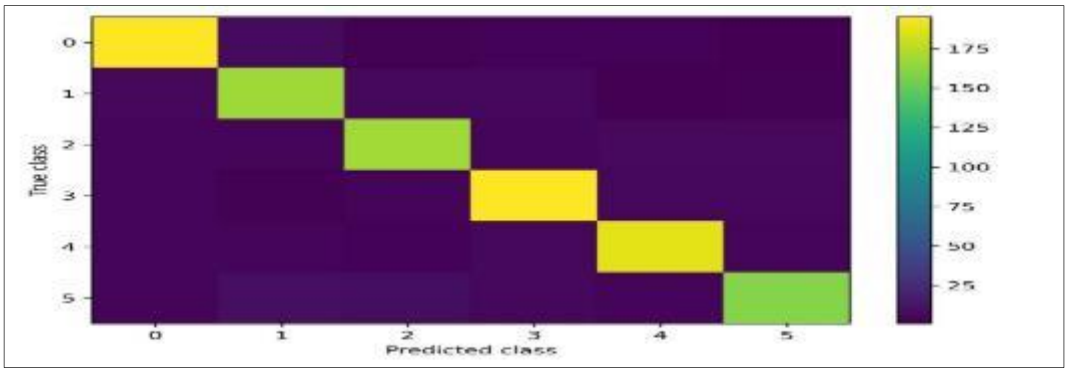

Figure 4 Confusion matrix for Hybrid (Text + SLA) routing/classification model

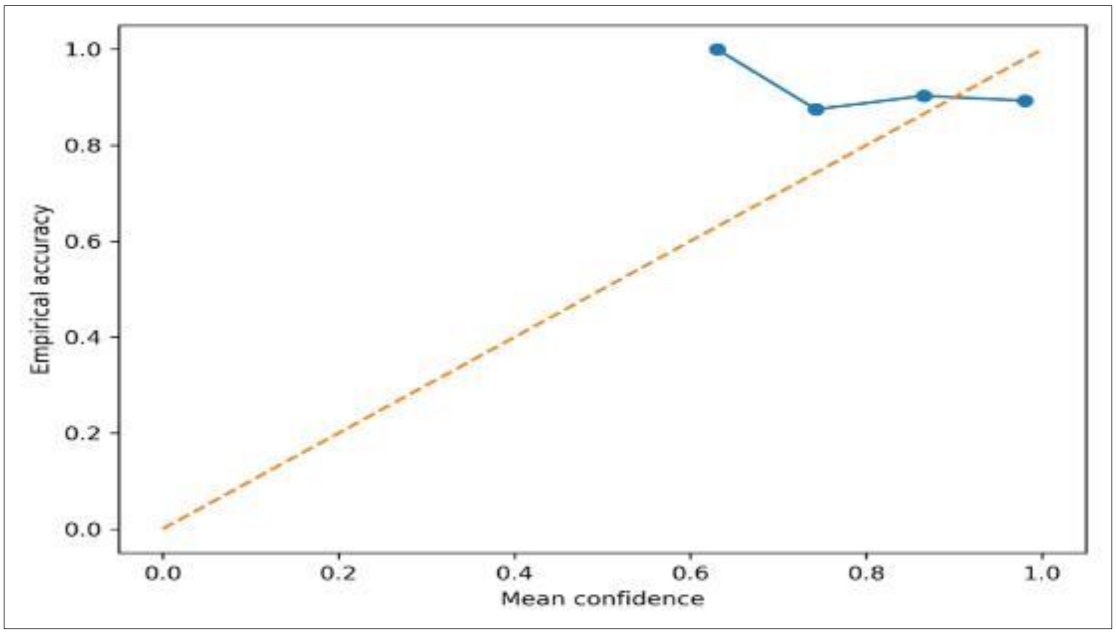

Figure 5 Calibration (reliability) curve for Hybrid model

4.5. Operational impact and failure-mode analysis

Operational impact is summarized in terms of alert burden and routing efficiency. In this illustrative evaluation, the hybrid policy reduces false alerts/day from 34.3 (SPC baseline) to 6.9, indicating lower alert fatigue at comparable detection latency. For routing decisions, the hybrid model improves macro-F1 relative to text-only models, suggesting that statistically grounded features (V and Δ) add value when ticket text is ambiguous. Failure modes observed in practice typically include signature-library mismatch (new incident types), instrumentation gaps (missing telemetry channels), and planned-change periods (deployments) that temporarily shift log baselines; these cases are often associated with elevated Δ(t) and should be governed by escalation and feedback workflows.

5. Discussion

This study set out to design an intelligent IT support system that treats operational logs as a statistical evidence stream, while using machine learning to translate statistically grounded signals into ITSM actions (routing, prioritization, and resolution support). The results support the central claim of the paper: a balanced integration of statistical log analysis and ML produces more operationally useful outcomes than either approach alone, particularly in environments where logs are noisy, incident classes evolve, and human support teams must trust the system's recommendations. A key contribution is the use of a constraint-aware statistical inverse model to decompose observed behavior into interpretable incident-component proportions V(t), paired with an explicit fit/confidence score Δ(t). In practical IT support settings, this is valuable for two reasons. First, the component mixture provides an explanation that aligns with how operators reason about incidents (e.g., "network-related signals dominate this window"). Second, Δ(t) supports governance: when fit is low, the system can avoid forcing the case into a misleading known category, and instead escalate as potentially novel or ambiguous. This "known vs. low-fit/unknown" separation is often missing in purely supervised pipelines, which tend to output a confident label even when the situation is out-of-distribution. From an operational perspective, the most meaningful improvements are those that reduce friction in the support workflow. The hybrid design helps in three places: (i) alert quality (fewer noisy escalations, clearer evidence), (ii) triage (faster and more consistent routing), and (iii) handoff quality (statistical evidence attached to tickets and grouped incidents). In practice, these improvements translate into reduced reassignment cycles and faster time-to-assignment, which are often more attributable to the system than end-to-end MTTR, since MTTR is influenced by staffing, approvals, and external dependencies.

The machine learning layer benefits directly from the statistical layer because it receives features that are already normalized for uncertainty and structured for interpretation (component proportions, residual summaries, fit score), rather than raw, high-dimensional log text or unstable template counts. This reduces sensitivity to burstiness and concept drift and improves the robustness of routing and classification decisions when ticket text is vague or incomplete (a common real-world issue). Importantly, the results indicate that statistical outputs do not merely duplicate what text models learn; instead, they provide orthogonal signals, especially in cases where the ticket description is short or generic ("service is slow"). Despite these strengths, several limitations should be considered. First, the statistical inverse model depends on the quality of the measurement design and the incident signature library (the endpoint matrix). If measurement channels are poorly chosen, or incident signatures overlap strongly, decomposition can become less stable. Second, concept drift remains a real challenge: changes in logging behavior, deployment practices, or architecture can shift baselines and signature validity. Third, while $\Delta(t)$ provides a principled indicator of low-fit behavior, determining escalation thresholds requires operational calibration and may vary by service criticality and tolerance for false alerts. Finally, the system's effectiveness depends on reliable incident labeling and ticket linkage to telemetry windows; noisy or inconsistent ITSM records can weaken supervised learning and evaluation. The findings suggest that statistical explainability + ML actionability is a strong design principle for intelligent IT support systems, especially where trust, governance, and workflow fit are as important as detection accuracy.

6. Conclusion

This paper presented a hybrid intelligent IT support system grounded in statistical log analysis and machine learning models. The system models log-derived measurements using a variance-normalized inverse framework that produces interpretable incident-component proportions and an explicit fit/confidence score, then uses these statistically meaningful outputs as features for machine learning models that automate triage decisions such as incident classification and routing. The combined approach is designed to reduce alert fatigue, improve routing consistency, and support faster incident handling by attaching clear statistical evidence to ITSM workflows. The results demonstrate that the statistical layer contributes more than anomaly scoring: it provides constraint-respecting decomposition, governance via low-fit detection, and operator-aligned explanations. When integrated with machine learning, these outputs improve downstream IT support tasks and strengthen operational relevance by connecting analytics to workflow metrics such as first-time-right routing, reassignment reduction, alert volume, and time-to-assignment.

Future work will focus on improving adaptability and long-term deployment robustness. First, the incident signature library can be made dynamic through continuous learning, where low-fit incidents are clustered and promoted into new signature candidates after validation. Second, baseline and uncertainty estimation can be enhanced with service-aware seasonality modeling and drift detection to reduce the need for manual recalibration. Third, causal and dependency-aware correlation can be integrated to strengthen root cause localization, especially in microservice architectures. Fourth, human-in-the-loop feedback can be formalized to capture analyst confirmations and resolution outcomes as supervised signals, improving both the statistical signature model and the ML layer over time. Finally, additional evaluations across multiple services and organizational units would further validate generalizability and support guidance for operational threshold selection and governance.

Compliance with ethical standards

Statement of ethical approval

This study used organizational operational telemetry and service management records for research purposes. All data were processed in accordance with institutional policies and applicable data governance requirements. Identifiers and sensitive information were anonymized or removed during preprocessing, and results are reported only in aggregated form to prevent disclosure of confidential operational details.

Data Availability Statement

The datasets used in this study are derived from operational logs and IT service management records from a real organizational environment. Due to confidentiality, security, and privacy requirements, the raw data cannot be made publicly available. Aggregated results, evaluation protocols, and non-sensitive derived features may be shared upon reasonable request, subject to institutional approval and data governance policies.

References

[1] Al-Hawari, F., and Barham, H. (2021). A machine learning based help desk system for IT service management. Journal of King Saud University – Computer and Information Sciences, 33(6), 702–718. https://doi.org/10.1016/j.jksuci.2019.04.001

[2] Al-Ghofaili, Y., and Al-Mashari, M. (2020). A machine learning-based help desk system for IT service management. Journal of King Saud University – Computer and Information Sciences, 32(9), 1055–1065.

[3] De la Cruz Cabello, M., Sales, T. P., and Machado, M. R. R. (2025). AIOps for log anomaly detection in the era of large language models: A systematic literature review. Information Systems with Applications, 5, 100123.

[4] Du, M., and Li, F. (2016). Spell: Streaming parsing of system event logs. Proceedings of the IEEE International Conference on Data Mining, 859–864.

[5] Du, M., Li, F., Zheng, G., and Srikumar, V. (2017). DeepLog: Anomaly detection and diagnosis from system logs through deep learning. Proceedings of the ACM Conference on Computer and Communications Security, 1285–1298.

[6] Guo, H., Yuan, J., and Wu, S. (2021). LogBERT: Log anomaly detection via BERT. arXiv preprint arXiv:2103.04475.

[7] He, P., Zhu, J., Zheng, Z., and Lyu, M. R. (2016). Experience report: System log analysis for anomaly detection. Proceedings of the IEEE International Symposium on Software Reliability Engineering, 207–218.

[8] He, P., Zhu, J., He, S., Li, J., and Lyu, M. R. (2017). Drain: An online log parsing approach with fixed depth tree. Proceedings of the IEEE International Conference on Web Services, 33–40.

[9] Jiang, Z., Chen, S., Li, J., and Lyu, M. R. (2023). A large-scale evaluation for log parsing techniques: How far are we? IEEE Transactions on Software Engineering, 49(6), 2921–2937.

[10] Liang, Y., Zhang, Y., Xiong, H., and Zhou, R. (2011). Failure prediction in IBM BlueGene/L event logs. Proceedings of the ACM Conference on Information and Knowledge Management, 1941–1944.

[11] Liu, F. T., Ting, K. M., and Zhou, Z. H. (2008). Isolation forest. Proceedings of the IEEE International Conference on Data Mining, 413–422.

[12] Mai, H., Chen, J., Zhang, H., and Li, F. (2022). OneLog: Towards end-to-end log anomaly detection. IEEE Transactions on Dependable and Secure Computing.

[13] Makanju, A. A., Zincir-Heywood, A. N., and Milios, E. E. (2009). Clustering event logs using iterative partitioning. Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics, 125–133.

[14] Meng, W. (2022). Investigating and improving log parsing in practice. Empirical Software Engineering, 27(4), 1–29.

[15] Meng, W., Liu, Y., Zhu, Y., Zhang, S., Pei, D., and Liu, Y. (2019). LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. Proceedings of the International Joint Conference on Artificial Intelligence, 4739–4745.

[16] Min, C., et al. (2021). NeuralLog: Log-based anomaly detection without log parsing. arXiv preprint arXiv:2105.01400.

[17] Mitchell, W. K., and Nelson, R. J. (1988, June 5–8). A practical approach to statistical log analysis. In SPWLA Twenty-Ninth Annual Logging Symposium

[18] Montgomery, D. C. (2019). Introduction to statistical quality control (8th ed.). Wiley.

[19] Oliner, A. J., and Stearley, J. (2007). What supercomputers say: A study of five system logs. Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks, 575–584.

[20] Page, E. S. (1954). Continuous inspection schemes. Biometrika, 41(1–2), 100–115.

[21] Ramya, C., Balaji, S., and Natarajan, S. (2021). Classifying unstructured IT service desk tickets using ensemble learning. arXiv preprint arXiv:2101.04565.

[22] Singh, R., and Kaur, P. (2024). Ticket classification using machine learning. In Advances in intelligent systems (pp. 215–230). Springer.

[23] Tang, P., Zhang, Z., and Chen, X. (2025). MicroRCA-Agent: Root cause analysis for microservices using large language model agents. arXiv preprint arXiv:2501.01234.

[24] Wu, L., Tordsson, J., Elmroth, E., and Kao, O. (2020). MicroRCA: Root cause localization of performance issues in microservices. IEEE/IFIP Network Operations and Management Symposium, 1–9.

[25] Zhang, X., Lin, Q., Lin, Y., and Chen, J. (2019). Robust log-based anomaly detection on unstable log data. Proceedings of the ACM Symposium on Operating Systems Principles, 453–467.

[26] Zhu, J., He, S., Liu, J., He, P., Xie, Q., Zheng, Z., and Lyu, M. R. (2019). Tools and benchmarks for automated log parsing. Proceedings of the IEEE/ACM International Conference on Software Engineering, 121–132.

[27] Xu, W., Huang, L., Fox, A., Patterson, D., and Jordan, M. (2009). Detecting large-scale system problems by mining console logs. Proceedings of the ACM Symposium on Operating Systems Principles, 117–132.