Majestic Journal of Information Technology
Volume 1, 2026, **pp.** 1-22, **Article ID:** CM2601102004

Research Article

Explainable Forecasting of Anomalous Operational States Using Multivariate Cloud Telemetry

Abdul Ghafoor Memon[1, *]

[1]*Emaan Institute of Management Sciences, Karachi, Pakistan*

Article History

Received: 01 April, 2026 Revised: 09 September, 2026 Accepted: 16 September, 2026 Published: 28 September, 2026

Abstract:

Introduction: Unexpected anomalous states in cloud-based IT services can create substantial operational risk, motivating early-warning analytics that anticipate telemetry degradation before it is observed. This study presents an explainable framework for forecasting anomalous operational states from multivariate cloud telemetry.

Methodology: The Cloud Resource Usage Dataset contains 14,400 observations across 10 user streams and 1,440 one-minute timestamps. Because confirmed outage records are unavailable, the anomaly label is treated as a proxy for possible service degradation, making the task anomaly-risk forecasting rather than validated outage prediction. After data-quality correction, 31 telemetry-derived predictors were constructed without anomaly labels or workload type. A 12-step history predicted the exact label three steps ahead, representing a nominal three-minute horizon. Chronological splitting within each user stream produced 4,170 test sequences with 341 positives (8.18%). Logistic Regression, Random Forest, XGBoost, and LSTM were evaluated at a fixed 0.50 threshold. LSTM achieved F1 = 0.1421, PR-AUC = 0.0845, recall = 0.5513, and false-alert rate = 0.5529, while Logistic Regression achieved the highest PR-AUC (0.0874).

Results: Bootstrap testing found no significant LSTM advantage over Logistic Regression or XGBoost, although F1 significantly exceeded Random Forest. SHAP identified CPU-memory correlation, disk-I/O variance, disk-I/O lag 3, memory lag 3, and CPU slope as leading contributors. Episode-onset analysis showed 54.38% coverage across 320 episodes.

Conclusion: The results support proof-of-concept explainable anomaly-risk forecasting, although weak discrimination and the short horizon require cautious interpretation and independent production-scale validation.

Keywords: Cloud telemetry, anomaly forecasting, AIOps, LSTM, explainable Artificial Intelligence.

1. INTRODUCTION

Cloud-based IT services are a cornerstone of today's digital infrastructure. They can run mission-critical applications in the financial, healthcare, e-commerce, media streaming, and enterprise IT sectors [1, 2]. Cloud elasticity is becoming a cornerstone of cloud computing, helping organisations become scalable, available, and cost-effective [3]. This dependency also increases the risk of disruption; if service is interrupted for any reason, it can result in financial losses, service level agreement (SLA) breaches, reputational damage, and a loss of user trust [4, 5]. As cloud systems grow in size, scope, and complexity, service availability is increasingly a critical operational reliability concern.

Outages are unlikely to be caused by one single failure point in a cloud environment. They tend to result from pervasive resource pressure, compute workload bursts, and poor interactions among compute, memory, storage, and network elements [6]. However, many operational monitoring systems still use hard-coded threshold alerts that trigger only when a metric crosses a threshold [7]. This reactive approach often fails because it is too late to prevent failure. Fixed thresholds also struggle to adapt to dynamic workloads and heterogeneous service behaviour, resulting in too many false alerts and too few alerts for critical events. These constraints drive the need for smarter, more predictive methods of cloud reliability management.

[*]Address correspondence to this author at Emaan Institute of Management Sciences, Karachi, Pakistan; E-mail: deancs@emaan.edu.pk

Cite as: A. G. Memon, "Explainable forecasting of anomalous operational states using multivariate cloud telemetry," *Maj J Infor Tech.*, vol. 1, p. 1–22, 2026, Article Id: CM2601102004.

This study takes a different perspective on service outages, defining outage risk as a time-varying process rather than a series of one-off, discrete events. Unusual cloud activity can manifest as slow changes in telemetry data, such as CPU saturation, increasing memory pressure, network performance fluctuations, or growing disk activity [8]. These indicators can happen before the service is interrupted and can thus be modelled as potential service outage indicators. For this reason, outage-risk management is considered a forecasting problem in this context, not just a task to be done after an incident.

Early-warning systems that predict outage risk have benefits over traditional fault-detection systems. Operators can use predictive analysis to anticipate and address cumulative stress and progressive degradation before users notice, which could limit downtime and operational impact [9]. This view aligns with trends in AIOps and site reliability engineering towards data-driven decisions that support incident response, rather than a reactive approach [10]. But this is not possible without models that are reliable and learn temporal dependencies in high-dimensional operational data in a reliable and interpretable way.

Many papers have focused on anomaly detection and failure diagnosis in cloud systems, but there are several limitations. First, most investigations rely on a single evaluation metric, such as CPU utilisation or memory utilisation, rather than characterising relationships across resource dimensions [11]. This is not an accurate representation, because outages often occur when multiple metrics are stressed, not just when one spikes. Second, most of the literature focuses on post-incident anomaly detection; that is, it detects abnormal behaviour only after service degradation has occurred. These can be diagnostically valuable, but offer little warning for prevention. Third, anomaly detection does not necessarily equal outage forecasting; detecting an anomaly does not mean a service disruption.

Many black-box machine learning and deep learning models also lack transparency. In operational environments, trustworthy predictions and actions are hard to achieve, particularly when alerts can trigger automatic or semi-automatic mitigation [12]. If models are not explainable, reliability engineers may be less likely to adopt them, and even when they are correct, they may be less useful. These issues highlight the need for multivariate, explainable outage-risk predictive frameworks.

Cloud monitoring and AIOps knowledge has been maturing, but a gap remains. Very few publicly available frameworks still support explainable forecasting of IT service outage risk. Numerous studies use operational data that are unavailable, which limits comparisons between studies. Further, cloud telemetry is often underused, and lagged effects, rolling trends, and escalation patterns that lead to degradation are generally poorly reported. The field also needs to address rare-event prediction under class imbalance and control false alerts. The fusion of these capabilities, such as forecasting, cross-resource telemetry fusion, temporal learning, and explainability, is therefore not thoroughly explored.

This study aims to address these gaps by achieving three objectives. First, it builds a predictive analytics model to estimate future abnormal service operation states based on cloud telemetry data, using abnormal cloud service indicators only as surrogate measures of possible service degradation. Second, it incorporates CPU, memory, disk I/O, network I/O, lagged values, rolling statistics, and slopes; it also includes relationships between resources not derived from labels, while excluding workload type from the principal predictor set to minimise proxy leakage. Third, SHAP values provide interpretable predictions so that reliability engineers can investigate the factors driving a prediction. The novelty is not algorithmic but methodological integration: several well-known methods of modelling, evaluation, and explainability are integrated, and the model's strength and robustness are assessed while accounting for data imbalance. The contribution introduces a leakage-controlled forecasting workflow that combines multivariate telemetry processing, temporal feature engineering, imbalance-aware assessment, robustness validation, and operational interpretation.

This contribution is presented as an application of an AIOps framework rather than a new learning algorithm. These techniques – lagged features, rolling statistics, LSTM forecasting, and SHAP explanations – are well-known. Still, the innovation is combining them into a leakage-controlled, multivariate anomaly-risk forecasting pipeline that leverages an open dataset, imbalance-aware evaluation, false-alert analysis, episode-onset warning-coverage analysis at a preselected forecast window, and explainability. The study is not only anomaly-tagged, but it also forecasts these operational states instead of just detecting them as anomalies; it evaluates 31 different predictors derived from the telemetry signal data, but under class imbalance, it compares linear, bagged-tree, gradient-boosting, and sequence models, and interprets the resulting risk scores using SHAP. Workload type is not included in the main set of predictive features because it is highly correlated with the dataset label; it is included in the diagnostic and sensitivity analysis.

The operational interpretation is intentionally limited to early anomaly warning. We use an exact three-step-ahead target, and the data sampling interval has a median of 60 seconds, resulting in a nominal three-minute forecast horizon. The latency of mean model-only inference was less than 0.001ms per sample for the Logistic Regression model, and around 0.028ms, 0.003 ms, and 0.037ms per sample for Random Forest, XGBoost, and LSTM, respectively. These values are based on model inference only; a pipeline with live data would also require telemetry ingestion, processing, communication, orchestration, and possible human approval. Therefore, the three-minute anomaly forecast should not be treated as a three-minute warning before a customer-visible outage.

2. LITERATURE REVIEW

2.1. IT Service Reliability and Outage Analysis

In large-scale distributed and cloud systems, researchers have extensively studied IT service reliability and availability,

as the coordinated behaviour of heterogeneous infrastructure components affects cloud service availability [13]. Previous studies list resource exhaustion, workload peaks, cascading failures, configuration errors, and software bugs as some of the most common causes of cloud-service outages. Cloud environments differ from monolithic systems because interdependencies among compute, storage, and networking layers can cause localised stresses to cascade rapidly into service-level degradations [14]. Hence, outages often result from accumulated operational pressures rather than a single failure.

Traditional site reliability engineering (SRE) processes monitor key performance indicators (KPIs), establish service-level objectives (SLOs) [15], and react to static or dynamic alerts. These practices improve observability and incident response but remain largely reactive. Monitoring techniques typically detect abnormal behaviour only after a set threshold is exceeded, so they fail to anticipate the onset of degradation. Furthermore, when a cloud-based system contains elastic and variable workloads, thresholds may be difficult to model because normal operating ranges change across services and time periods [8, 16]. Thus, traditional reliability monitoring is not very effective at predicting failure before it becomes operationally relevant.

2.2. Anomaly Detection in Cloud and Distributed Systems

Previous studies around cloud failure management have mainly focused on anomaly detection. Common statistical tools include z-score analysis, moving averages, and control charts, which are simple and easy to understand [17]. These methods are generally stationary or distributionally regular; however, this assumption does not hold for cloud telemetry. To overcome these limitations, researchers have proposed machine learning techniques such as clustering, isolation forests, autoencoders, and probabilistic graphical models to find nonlinear patterns in high-dimensional monitoring data.

In many cases, machine learning substantially improves anomaly-detection accuracy, but most existing approaches are, by nature, retrospective [18]. They detect abnormal behaviour only after it occurs, so they offer limited value in proactively preventing outages [19]. Another limitation is that these models may be confused by benign workload variation, leading to more false positives [20]. Importantly, an anomaly isn't necessarily an outage precursor, as some anomalies are short-lived and self-resolving. These limitations show the need to complement detection models with prediction models to estimate the likelihood of service failure.

2.3. Predictive Analytics and AIOps

AIOps is a paradigm change from reactive monitoring to predictive and automated IT operations. It is similar to predictive maintenance in industrial systems, using sensor data to predict equipment failure before it occurs. The same principle holds in cloud environments: If cloud resource usage is treated as a temporal forecasting problem, the data can be used to detect early indicators of service degradation.

While the concepts align, predictive modelling in IT operations faces several technical challenges. Outage events are extremely rare, leading to class imbalance that may bias standard learning algorithms [21]. Operational data are also noisy due to workload variability, deployment activity, and measurement artefacts. Cloud environments are not stationary, and new configurations and workload profiles can cause temporal drift [22]. Deep learning models like recurrent neural networks and temporal convolutional networks can learn sequential dependencies [23], but they lack interpretability and operational trust in AIOps. Predictive analytics thus supplements, rather than replaces, existing reliability practices.

2.4. Multivariate Operational Telemetry

One shortcoming often noted in previous studies is the focus on a single resource or narrowly defined telemetry sources. Studies analysing CPU, memory, or network usage alone may not capture the interdependencies among cloud resources that drive complex degradation patterns. When workloads generate compounded resource contention, several resources are likely to be affected by service degradation: CPU, memory, disk I/O, and network [24]. These interactions can make predictions difficult and limit the ability to generalise for different workload types.

Metrics for compute, memory, storage, and networking resources have attracted interest in a growing number of recent studies on cross-resource telemetry fusion [5]. These methods provide a more holistic view of system health by accounting for cross-resource relationships and temporal dynamics. However, many current studies still focus on anomaly detection or root-cause analysis without considering future outage-risk forecasting. Multi-source models may be more complicated, complicating feature selection, model understanding, and computation. These trade-offs encourage a feature-engineering strategy that balances predictive power with operational ease.

2.5. Explainable AI for IT Operations

A critical need for using machine learning in IT operations is explainability. Reliability engineers need to know not only what a model predicts, but also why it predicts that. If the predictive model's outputs feed into mitigation decisions, the model's behaviour becomes opaque, and operators may lose trust in it and be unable to judge whether an alert reflects credible operational risk [12]. Explainable AI has thus emerged as a crucial demand in AIOps research and practice.

Other approaches like SHAP and LIME can explain the contribution of individual features to predictive risk and link these explanations to operational metrics such as disk contention, memory pressure, and CPU saturation, enhancing actionability [25]. However, previous research usually applies explainability at the end of static modelling [26, 27]. This limits the immediate usefulness of XAI in real-world IT service management.

2.6. Summary and Identified Gap

In conclusion, although significant progress has been made in cloud monitoring, anomaly detection, and cloud diagnostic analysis, gaps remain in forecasting regarding explainability and reproducibility. Much of the literature uses detection-

oriented methods that offer little warning, and class imbalance, temporal drift, and limited interpretability remain problematic for predictive analytics. Cross-resource telemetry fusion has become a key factor; however, many studies focus on future-risk forecasting without providing operational interpretations.

Proprietary datasets also limit reproducibility and comparative analysis. Hence, evaluation of explainable, multivariate outage-risk prediction based on publicly accessible heterogeneous operational data remains underdeveloped. To address this, this work introduces an integrated framework that integrates temporal predictive modelling, multivariate feature engineering, imbalance-aware evaluation, and explainable decision support. It therefore contributes in practice and methodology, not by offering novelty in any individual element such as lag features, rolling statistics, LSTM, or SHAP.

2.7. Novelty Positioning Against Existing Literature

The study's novelty lies in the integrated forecasting workflow rather than in any individual algorithmic component. The workflow combines multivariate telemetry, temporal feature engineering, class-imbalance handling, validation-based model selection, episode-onset warning-coverage analysis at a prespecified horizon, false-alert evaluation, robustness testing, and SHAP explanations in a single anomaly-risk forecasting study. This positioning avoids overstating algorithmic novelty while clarifying the practical methodological contribution.

3. METHODOLOGY

3.1. Dataset Description

This research used the open Cloud Resource Usage Dataset for Anomaly Detection. The final implementation included 14,400 observations, 10 user streams, and 1,440 unique timestamps from 2025-07-01 00:00:00 to 23:59:00, with a median sampling interval of 60 seconds. Each observation includes CPU usage, memory usage, disk I/O, network I/O, workload type, user identifier, and an anomaly label. In the public dataset description, anomaly labels are defined as signals of resource overuse, including hidden anomalies, but without a precise, deterministic formula for generating labels. Thus, this manuscript does not infer the provenance of the label beyond the published description. Empirically, 1,257 labels were observed in Crypto Mining, and all of them were positive, which accounts for 85.7435% of all the observed labels, and the rule Crypto Mining with CPU usage at least 80% matched 99.9931% of the observed labels with one mismatched label. Workload type was therefore an unusually strong proxy for the label and was omitted from the main set of predictive features and examined individually in a sensitivity analysis.

The data set does not include records of confirmed production outages, but it could be used to control an environment to determine whether temporal telemetry patterns can be predicted for abnormal operational states. The goal of this research is not to directly predict outages visible to the customer, but to assess the potential of an explainable early-warning system that uses a middle layer of abnormal system behaviour as a warning signal.

We mathematically represented the dataset as a multivariate time series.

$$\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^{T}, \quad (1)$$

The vector of observations at time t is called x_t, and the corresponding label of anomaly at time t is called y_t. The original analysed variables have no missing values during the observation period, so no pre-observation period and no imputation are needed before feature engineering. While there are 14400 rows, these are divided across 10 parallel user streams, each covering only 1440 different one minute timestamps, which has implications as noted in the Limitations section below.

3.2. Outage Proxy Definition

Availability reports for confirmed IT service outages are hard to come by since outages are often a matter of confidentiality and related to in-house operations. An anomaly label is used as a proxy for this, so that the study includes only the potential causes of a service disruption rather than service disruption itself. This is important to note, because the model is predicting the probability of anomaly events that are precursors to a production outage, and not the actual production outage. This is in line with the AIOps literature that indicates that abnormal telemetry can last for a long time before service degradation or service unavailability occurs.

The principal modelling target was the exact anomaly label three observations ahead rather than a rolling-OR or sustained-anomaly proxy. For user stream u and anchor time t, let $z(u,t)$ denote the 31-feature engineered predictor vector and $y(u,t)$ denote the observed anomaly label. The primary forecasting target was $y(u,t+3)$, implemented by shifting the anomaly label by three observations within each user stream. Each classical model therefore used $z(u,t)$ to predict $y(u,t+3)$. For the LSTM, the 12-step input sequence consisted of $z(u,t-11), z(u,t-10), \ldots, z(u,t)$, and this complete sequence was paired with the same target $y(u,t+3)$. The $t+3$ target alignment was completed before model-ready observations were counted and before LSTM sequence generation; therefore, the three-step horizon was not subtracted again when calculating the 987 development sequences or 417 held-out sequences per user.

A secondary sustained-anomaly sensitivity analysis was conducted to examine whether conclusions changed when a positive state required at least k consecutive anomaly labels. For k=2, the held-out set contained 42 positive observations (1.0072%) forming 21 episodes. For k=3 and k=5, no positive held-out observations or episodes remained; therefore, precision, recall, F1, PR-AUC, false-alert rate, and warning coverage were not estimable for those settings. The sustained definitions were treated as sensitivity analyses only and did not replace the exact primary target.

3.3. Exploratory Data Analysis

The statistical characteristics, variation, and correlations of cloud resource metrics were explored first through exploratory data analysis before the model was built. It was not an

inferential objective, but rather a diagnostic objective: the analysis was used to understand the distributional skewness, variance, and extreme values to guide in feature engineering and model selection. Cloud telemetry data exhibit this kind of skewness and variance because workloads are inherently bursty and elastic.

Workload heterogeneity was also taken into account, due to different types of services generating different resource-utilisation patterns. For instance, a workload that uses a lot of CPU computations can cause the CPU to become overloaded, while a data-driven workload might require more memory and disk I/O. Inter-dimensional relationship of resources was explored to find correlation in the behaviour which could signify cascaded stress. The results of these observations warranted use of multivariate features and cross-resource dependencies in the predictive model.

3.4. Data Preprocessing

Before feature extraction and model training, telemetry validity checks were applied. The implementation identified 23 CPU values below 0%, one CPU value above 100%, and 285 negative Disk I/O values. CPU usage was clipped to the physically valid 0–100% range and negative Disk I/O values were clipped to zero before feature engineering and standardisation. After cleaning, CPU ranged from 0 to 100 and Disk I/O from 0 to 69.70. Continuous predictors were then standardised using means and standard deviations estimated exclusively from the training partition, with the same training-derived parameters applied to held-out data to prevent leakage.

$$x'_{t,i} = \frac{x_{t,i} - \mu_i}{\sigma_i}, \tag{2}$$

where $x_{t,i}$ denotes the observed value of feature i at time t, while μ_i and σ_i denote the mean and standard deviation of feature i, respectively, calculated exclusively from the training partition. The same training-derived parameters were subsequently applied to the validation and test partitions to prevent information leakage.

Temporal splitting was performed chronologically and separately within each of the 10 user streams after feature engineering and exact t+3 target alignment. Each original user stream contained 1,440 observations; the 12-observation rolling feature calculations made the first 11 positions unavailable, while the t+3 target shift made the final three positions unavailable, leaving 1,426 model-ready anchor–target observations per stream. The first 998 model-ready observations formed the development partition and the final 428 observations formed the held-out test partition. For classical-model hyperparameter selection, the first 898 development observations were used for fitting and the final 100 observations were used for chronological validation. For the LSTM, 12-step sequences were first generated across the complete 998-observation development partition, producing 998 − 12 + 1 = 987 development sequences per user; these were then split chronologically into the first 888 fitting sequences and the final 99 validation sequences. Within the held-out partition, 428 − 12 + 1 = 417 complete test sequences were available per user, producing 4,170 common aligned held-out sequences across the 10 user streams. No LSTM sequence crossed a user-stream or development/test boundary; because the LSTM fitting/validation split was applied after development-sequence construction, adjacent fitting and validation sequences could share historical input observations. Although validation targets occurred strictly after fitting targets, shared historical inputs near the fitting/validation boundary may introduce limited dependence between the two subsets; future work should evaluate a purged temporal validation gap that removes all shared input history.

3.5. Feature Engineering

The purpose of feature engineering was to include both the current state of the system and its history. Lagged features were created from past observations of these continuous resource metrics (CPU usage, memory usage, disk I/O, and network I/O) in each user stream, which allowed the models to feed in the latest historical resource behaviour.

$$\mu_{t,i}^{(w)} = \frac{1}{w}\sum_{j=0}^{w-1} x_{t-j,i}\,, \tag{3}$$

Rolling statistical features were derived from sliding windows, for example, rolling means and rolling variances, and slope features were derived to identify short-term trends in the resources.

$$\sigma_{t,i}^{2(w)} = \frac{1}{w}\sum_{j=0}^{w-1} (\,x_{t-j,i} - \mu_{t,i}^{(w)})^2, \tag{4}$$

CPU_Burst was calculated *via* the implemented first difference rule to detect a sudden CPU increase, and CPU_Memory_Corr was computed as the 12-observation rolling Pearson correlation of CPU and memory use. These features are able to capture short time periods of CPU spikes and cross-resource dependence without any workload type and label information. These designed attributes combined transformed raw telemetry into a representation that was appropriate for predictive learning. The rolling means, variances, and CPU–memory correlation were calculated using a 12-observation rolling window, as was the implementation parameter ROLLING_WINDOW = 12. The 12 observation rolling window summarised around 12 minutes of the past telemetry.

3.6. Predictive Models

Four predictive models were compared using a shared chronological hold-out design. Logistic Regression was used to get the linear baseline. Non-linear tree-based interactions were modelled by Random Forest. XGBoost is also added as a more robust baseline gradient boosting model that can take advantage of engineered features such as lag, rolling, slope, burst, and cross-resource features without any memory of the sequences. The LSTM received 12-step sequences and modelled temporal dependencies directly through gated recurrent states. Including XGBoost and engineered temporal predictors creates a stronger comparison than evaluating the LSTM only against static linear

and bagged-tree baselines. The amount of historical information was not identical across model classes. Each classical model received a single 31-feature vector at anchor time t, whose longest engineered rolling features used raw telemetry from $t-11$ through t, corresponding to an effective history of up to 12 observations. The LSTM received 12 consecutive engineered vectors from $t-11$ through t; because the earliest vector in that sequence itself contained a 12-observation rolling summary, the LSTM could indirectly access raw telemetry as far back as $t-22$, corresponding to an effective history of up to 23 observations. The comparison should therefore be interpreted as a comparison of complete forecasting pipelines rather than as a controlled architecture-only comparison under matched temporal receptive fields.

$$\hat{p}_{t+h} = P(y_{t+h} = 1 \mid \mathbf{x}_t) = \sigma(\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_t) \quad (5)$$

$$\sigma(u) = \frac{1}{1+e^{-u}} \quad (6)$$

where x_t is the engineered feature vector constructed from the telemetry observations up to time t, β_0 is the intercept, β is the vector of the model coefficients, and p ̂(t+h) is the estimate of the probability of an anomalous operational state at time t+h. Logistic Regression offers a simple yet interpretable baseline model that is computationally efficient, but has the limitation of a linear decision function which does not support more complex interactions and sequential escalation processes.

To capture the interactions of nonlinear features, an ensemble of decision trees, the Random Forest Classifier, was used. The model does not include nonlinear interactions, but each engineered instance is modelled separately, which is not as flexible as direct modelling of sequential interactions. Therefore, a long short-term memory (LSTM) network was chosen to model explicitly the temporal dynamics. The LSTM can hold the information from previous time steps by using gated memory mechanisms, which can learn the escalation trajectories that could be precursors to outages.

In addition to the four principal predictive models, static CPU/memory threshold alerting and a user-specific EWMA CPU score were implemented as exploratory operational reference baselines on the same common aligned held-out set. The static reference generated a positive score when CPU usage exceeded 75% or memory usage exceeded 75%. The EWMA reference used a 12-observation CPU EWMA with adjust=False, calculated the CPU residual relative to the EWMA, scaled the residual by its population standard deviation, and transformed it to a 0–1 score using the logistic function. These operational references were retained separately from the four-model architectural comparison.

3.7. Implementation Details

Experiments were conducted in Python using NumPy and Pandas for preprocessing, scikit-learn for Logistic Regression and Random Forest, XGBoost for gradient boosting, PyTorch for the LSTM, and SHAP for explanation. Classical-model hyperparameters were selected from 11 prespecified candidate configurations using chronological validation PR-AUC as the sole selection criterion. Logistic Regression evaluated C ∈ {0.1, 1.0, 10.0}, and C = 0.1 was selected with validation PR-AUC=0.089369. Random Forest evaluated four configurations: 200 trees with depth 6 and minimum leaf size 2; 200 trees with depth 10 and minimum leaf size 2; 300 trees with depth 6 and minimum leaf size 4; and 300 trees with depth 10 and minimum leaf size 4. For all Random Forest candidates, min_samples_split = 2, max_features = sqrt, and class_weight = balanced were fixed. The selected Random Forest used 300 trees, maximum depth 6, and minimum leaf size 4, achieving validation PR-AUC = 0.091260.

XGBoost evaluated four prespecified configurations: 200 estimators with depth 3 and learning rate 0.03; 200 estimators with depth 5 and learning rate 0.05; 400 estimators with depth 3 and learning rate 0.03; and 400 estimators with depth 5 and learning rate 0.05. Subsample=0.90, colsample_bytree = 0.90, min_child_weight = 2, reg_alpha = 0, and reg_lambda = 1 were fixed across candidates. The selected configuration used 200 estimators, maximum depth 3, and learning rate 0.03, with validation PR-AUC=0.078695. The XGBoost positive-class weight was 10.031941 during validation search and 10.138393 after refitting on the complete development partition.

The LSTM architecture was prespecified rather than selected through a separate hyperparameter search. It contained one LSTM layer with 64 hidden units and dropout=0.20, and was trained using Adam with learning rate=0.001, training batch size=64, a maximum of 50 epochs, and early-stopping patience of five epochs. Training batches were not shuffled, validation used batch size 256, and early stopping monitored weighted binary cross-entropy validation loss with a minimum required improvement of 1×10^{-4}. The positive-class weight was 10.044776, calculated from 8,076 negative and 804 positive LSTM fitting sequences. The final training run stopped after six epochs and retained the checkpoint with the lowest validation loss of 1.214379.

Experiments were executed in a Google Colab Python 3 runtime using CPU execution only; the recorded implementation reported 'Device: cpu', and random seeds were fixed at 42 for NumPy, Python's random module, and PyTorch. Exact transient Colab package-version and hardware-instance metadata were not preserved in the saved run; consequently, the reported computational environment documents the execution platform, device class, libraries, and randomisation controls without claiming exact software-build reproducibility. The workflow loaded and validated telemetry; corrected physically implausible values; constructed 31 telemetry-derived predictors; generated the exact t+3 target within each user stream; performed user-specific chronological fitting, validation, and held-out splitting; generated 12-step LSTM sequences without crossing user or development/test boundaries; fitted the four predictive models; evaluated all models at a fixed 0.50 classification threshold on the common 4,170-sequence held-out set; and conducted feature ablation, workload-inclusion sensitivity, Gaussian-noise robustness, workload-specific evaluation, temporal robustness, sustained-anomaly sensitivity, paired-bootstrap testing, SHAP analysis,

episode-onset warning-coverage analysis at the prespecified horizon, prediction-stability analysis, and inference-latency benchmarking.

3.8. Robustness and Statistical Validation Protocol

To strengthen internal reliability, Gaussian-noise, workload-specific, temporal, workload-inclusion, sustained-anomaly, and paired-bootstrap analyses were applied using held-out data. The principal models were not retrained during Gaussian-noise, workload-specific, or temporal robustness evaluation. For Gaussian noise, perturbations were applied to the four cleaned raw telemetry variables in the held-out period and the lag, rolling, slope, burst, and correlation features were recomputed before scoring. Paired bootstrap testing used 1,000 identical resamples across models for F1 and PR-AUC comparisons. Table **1** summarises the purpose, procedure, and reported outputs of the robustness and statistical validation analyses used to assess the reliability of the predictive models.

The measurement noise, logging delays, and the slight variation of telemetry that occurs in real cloud monitoring systems were taken into account by including the Gaussian noise test. The workload-specific test was added to ensure that the forecasting model was not effective for only one workload type. The temporal robustness test has been added as workloads on clouds may vary over time and a practical outage prediction model should be stable across the various parts of the monitoring horizon. In combination, these tests offer controlled evidence of the robustness of the models, yet recognise that further external generalisation to production scale is required.

3.9. Statistical Significance Testing

A paired bootstrap significance test with 1,000 iterations was carried out to determine if the differences in performances between the LSTM model and the baseline models were statistically significant. Given the imbalanced nature of the outage forecasting task, F1 and PR-AUC were used as the statistics, as they are more suitable metrics for the task. All test instances were held out chronologically in the same manner and ground-truth labels were the same across all models. The pair bootstrap results are used here as an approximate within-sample uncertainty estimate, not as evidence from independent observations in time, since adjacent forecasting windows are not completely independent.

For each bootstrap iteration, test-set prediction windows were sampled with replacement. The same bootstrap sample was applied to the LSTM, Random Forest, XGBoost, and Logistic Regression predictions. For each model, the selected performance statistic was calculated as:

$$S_m^{(b)} \tag{7}$$

where $S_m^{(b)}$represents either the F1 or PR-AUC of model $m$during bootstrap iteration b. The performance difference between the LSTM and each baseline model was then computed as:

$$\Delta^{(b)} = S_{LSTM}^{(b)} - S_{baseline}^{(b)} \tag{8}$$

The 95% confidence interval was obtained from the 2.5th and 97.5th percentiles of the bootstrap distribution of $\Delta^{(b)}$. Statistical significance was calculated using a two-sided empirical bootstrap test:

$$p = 2 \times \min\left[P(\Delta^{(b)} \leq 0), P(\Delta^{(b)} \geq 0)\right] \tag{9}$$

A difference was considered statistically significant when the 95% confidence interval did not include zero and the two-sided bootstrap p-value was below 0.05. This procedure ensured that significance was computed from paired model-performance differences on identical test samples, rather than from an unpaired comparison.

3.10. Evaluation Metrics

Since anomaly events are imbalanced, precision, recall and F1 were prioritised in the performance evaluation:

$$\text{Precision} = \frac{TP}{TP + FP}, \text{Recall} = \frac{TP}{TP + FN}, F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{10}$$

PR-AUC was used to evaluate ranking performance across decision thresholds under class imbalance, while precision, recall, F1, and false-alert rate were calculated at the prespecified 0.50 classification threshold. The threshold was fixed before held-out evaluation and was not optimised using test labels. The 0.50 threshold was retained as a prespecified common operating point to ensure direct comparability across models and to avoid post-hoc threshold optimisation on the held-out test set; threshold optimisation for deployment-specific precision–recall trade-offs was outside the scope of this proof-of-concept evaluation. Operational suitability was additionally examined through episode-onset warning coverage at the prespecified forecast horizon, prediction stability, and inference latency.

Table **2**(**a**) summarises the training configuration used to maintain temporal realism and controlled optimisation. The window size captures short-term escalation trends, while the forecast horizon reflects a practical early-warning requirement. Chronological splitting reduces leakage risk, and early stopping helps control overfitting. Table **2**(**b**) shows that the final 31-feature predictor set integrates current-state telemetry with lagged, rolling, slope, burst, and cross-resource variables while excluding workload type and all label-derived information.

For each user stream u, $CPU(u,t), Memory(u,t), Disk(u,t),$ and $Network(u,t)$ represented the cleaned CPU usage, memory usage, disk I/O, and network I/O values at observation t, respectively. The four current-state predictors were therefore defined directly as $CPU(u,t), Memory(u,t), Disk(u,t),$ and $Network(u,t)$. $Resource_{Saturation(u,t)}$ was assigned a value of 1 when $CPU(u,t) > 75$ or $Memory(u,t) > 75$, and 0 otherwise. For each resource variable r, the three lag features were $Lag1(r,u,t) = r(u,t-1), Lag2(r,u,t) = r(u,t-2), and\ Lag3(r,u,t) = r(u,t-3)$. The 12-observation rolling mean was calculated as the arithmetic mean of observations from $t-11$ through t, while rolling variance was the sample variance of those same 12 observations using $ddof = 1$. The slope feature was calculated as

$Slope(r,u,t) = r(u,t) - r(u,t-1)$. $CPU\_Burst(u,t)$ was assigned 1 when $CPU(u,t) - CPU(u,t-1) > 2$ and 0 otherwise. $CPU\_Memory\_Corr(u,t)$ was the Pearson correlation between the 12 paired CPU and memory observations from $t-11$ through t.

CPU_Burst was implemented as a binary indicator of an abrupt increase in CPU usage and was defined as $CPU\_Burst(u,t) = 1$ when $CPU(u,t) - CPU(u,t-1) > 2$, and $CPU\_Burst(u,t) = 0$ otherwise. CPU_Memory_Corr represented the Pearson correlation between CPU and memory usage over the same 12-observation rolling window, using the paired observations from $t-11$ through t. Thus, $CPU\_Memory\_Corr(u,t) = Corr[CPU(u,t-11:t), Memory(u,t-11:t)]$. Overall, the predictor set consisted of four current-state telemetry variables, one Resource_Saturation indicator, 12 lagged predictors, four rolling means, four rolling variances, four first-difference slope predictors, one CPU_Burst indicator, and one CPU_Memory_Corr feature, giving a total of 31 predictors.

3.11. Explainability Framework

To support interpretability, SHapley Additive exPlanations (SHAP) were calculated for the final LSTM using GradientExplainer. Fifty development sequences were selected deterministically at evenly spaced indices across the 9,870 available LSTM development sequences and used as the SHAP background/reference set, while the first 200 held-out sequences were explained. GradientExplainer was executed with nsamples=50, and no held-out sequence was used in the background set. The sample size was selected as a computationally efficient approximation of the training distribution while preserving representative temporal feature variation. The resulting attribution array had shape (200, 12, 31), corresponding to sequences, time steps, and input features. Global feature importance was calculated as the mean absolute SHAP value averaged across all 200 explained sequences and all 12-time steps for each feature. A local sequence-level explanation was also produced by aggregating the 12 time-step attributions for an individual held-out sequence.

The anomaly-risk forecasting pipeline is summarised in Fig. (**1**). Core resource telemetry is validated, cleaned, transformed to user-specific temporal predictors, and chronologically split into a fitting, a validation and a held-out period. The type of work is also included in the context and for diagnostic and subgroup analysis, but not in the key predicting feature subset. Imbalance-sensitive as well as service-reliability measures are used to evaluate the predictive performance, and SHAP is employed to interpret the predictions of LSTM.

4. RESULTS

4.1. Correlation and Resource Interaction Analysis

The Cloud Resource Usage Dataset is summarised in Table **3**. The dataset is 14,400, time indexed observations with no missing data. The raw anomaly label is imbalanced – 1257 positives (8.7292%). This is a description of the prevalence in the forecasting sample, as opposed to the aligned held-out forecasting prevalence of 341/4,170 (8.1775%), which reduces the number of eligible forecasting samples with the removal of lag/rolling construction, t+3 horizon, chronological partitioning, and complete-sequence requirements.

The raw anomaly-label rate is 8.7292% (1,257/14,400). Before modelling, 23 negative CPU observations and one CPU observation above 100% were clipped to the physically valid 0–100% range, while 285 negative Disk I/O observations were clipped to zero. The final primary forecasting target was the exact anomaly indicator $t+3$; after alignment, the commonly held-out evaluation set contained 4,170 forecasting sequences with 341 positive targets (8.1775%).

The correlation analysis indicates that CPU usage is the strongest resource-level correlate of anomalous operational behaviour in this dataset. As seen in Fig. (**2**), CPU usage is highly correlated with the anomaly label, indicating that this dataset is likely to have anomalous system states when there is high CPU usage. This correlation cannot be seen as a cause-and-effect relationship, as the anomaly-generating process could be partly threshold-dependent. Positive moderate correlations are observed between memory usage and both CPU usage and disk I/O, suggesting a coupled resource stress under high workload.

Network I/O shows relatively lower correlations with CPU usage and the anomaly label. This suggests that network stress alone is less strongly associated with anomalous states in this dataset, or that its effect is indirect through compute and memory constraints. Overall, the correlation structure supports the modelling decision to use cross-resource telemetry rather than relying on a single metric.

From an operational perspective, the result suggests that single-resource thresholds may be insufficient for early-warning systems. Reliability engineers are more likely to benefit from correlated risk indicators that capture compounded subsystem stress, provided that these indicators are validated against production incident records.

4.2. Workload Characteristics

Fig. (**3**). shows substantial workload heterogeneity: Web Service and Database Query account for the largest numbers of observations, followed by Video Streaming, Crypto Mining, and Backup. Each of the 10 user streams contains all five workload categories, so workload type is time-varying rather than static within a user. Because all raw positive anomaly labels occur within Crypto Mining observations, workload type is treated as a potential label proxy and is excluded from the principal predictive model.

The workload mix exposes the models to different resource-utilisation profiles, but the extreme association between workload type and the raw anomaly label means that workload identity cannot be treated as an ordinary predictive feature without risking an artificially easy task. Consequently, the principal results use the 31-feature workload-excluded specification, while a separate workload-inclusion sensitivity analysis is reported to quantify the effect of explicitly adding workload encoding.

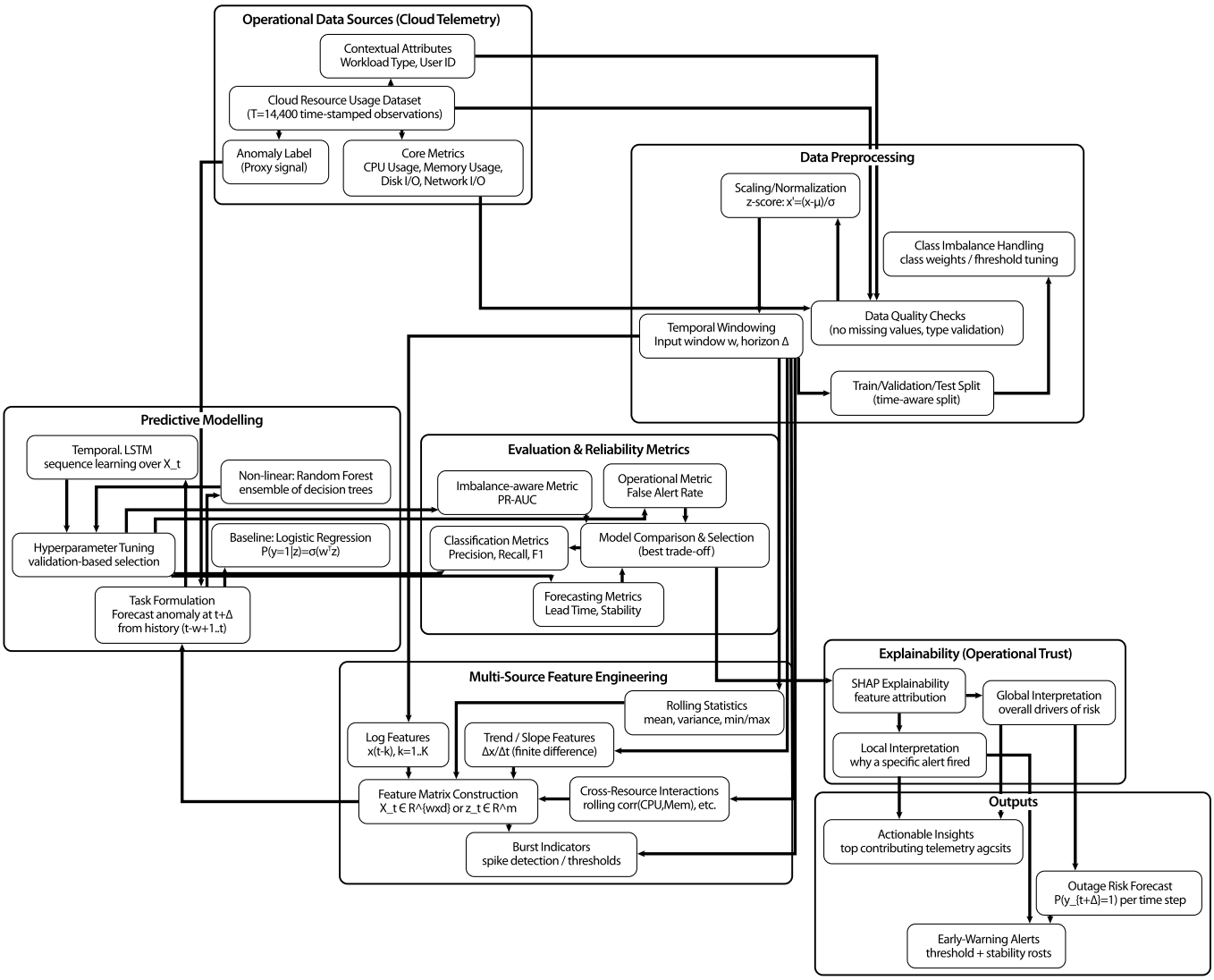

Fig. (1). Predictive analytics workflow for anomaly-risk forecasting. The framework combines multivariate cloud telemetry, leakage-controlled preprocessing, temporal feature engineering, predictive modelling, robustness evaluation, SHAP-based explainability, and early-warning risk estimates.

Operationally, the dataset contains heterogeneous service profiles within every user stream, but this should not be interpreted as evidence of generalisation to unseen enterprise workloads. The observed label structure and one-day time horizon require validation on independent production telemetry before workload-level generalisation can be claimed.

4.3. Resource Behaviour and Anomaly Patterns

Distributional analysis reveals non-stationary and nonlinear behaviour of the resource metrics. The utilisation of the CPU is right skewed, with the majority of the observations in the middle range of utilisation and a noticeable tail at the higher end as seen in Fig. (**4**). The overlap of this tail region is found with the anomaly labelled instances, indicating instability behaviour like threshold in the dataset in high CPU load.

In practice, this result means that it is possible to have instability without having reached the saturation point in the CPU. While it is possible that predictive modelling could be able to capture pre-threshold escalation patterns earlier, more than proxy anomaly labels alone are needed to confirm this using production outage records.

The data in Fig. (**5**), further illustrates high variability and outliers on the CPU, memory and disk I/O, as observed during bursty workloads and elastic scaling effects in cloud environments. The I/O dispersion is moderate for disk I/O, and comparatively low for network I/O. Outliers found throughout the different resource dimensions suggest that anomalous behaviour is influenced by high utilisation, but also by bursts and volatility. I believe that these patterns warrant the presence of rolling statistics, slope features and burst indicators.

Table 1. Robustness and statistical validation protocol.

Test	Purpose	Procedure	Reported Output
Gaussian noise stress test	Assess stability under measurement noise	Add zero-mean Gaussian noise equal to 10% of the training-set standard deviation to raw held-out CPU, memory, disk I/O, and network I/O; reapply physical bounds; recompute all engineered features; score without retraining.	Precision, recall, F1, PR-AUC, false-alert rate, ΔF1, ΔPR-AUC
Workload-specific robustness test	Assess variation across workload categories	Partition the common aligned held-out set by workload type. Workload type is used only for subgroup evaluation and is excluded from the principal predictor set.	N, positive cases, precision, recall, F1, PR-AUC, false-alert rate
Temporal robustness test	Assess short-horizon stability across time	Split the 4,170 aligned held-out forecasting sequences into non-overlapping early (2,080) and late (2,090) segments and score the same fitted models at the same threshold.	Positive rate, precision, recall, F1, PR-AUC, false-alert rate
Paired bootstrap significance test	Assess whether model differences exceed sampling variation	Use 1,000 paired bootstrap resamples of identical held-out indices for LSTM versus Logistic Regression, Random Forest, and XGBoost.	Observed difference, 95% CI, two-sided *p*-value, significance

Table 2(a). Model training configuration.

Parameter	Value
LSTM sequence length	12 engineered time steps
Forecast target	Exact anomaly label at t+3
Nominal horizon	3 minutes (median sampling interval 60 s)
Development/Test split	70% / 30% chronological within each user stream
Validation split	Classical models: first 898 of 998 development observations for fitting and final 100 for validation; LSTM: 987 sequences generated from all 998 development observations, then first 888 sequences for fitting and final 99 for validation.
Aligned held-out test set	4,170 sequences; 341 positives (8.1775%)
Predictor count	31 telemetry-derived features; workload and label-derived variables excluded
Decision threshold	0.50 fixed before held-out evaluation
Logistic Regression	C=0.1; lbfgs; L2; class_weight=balanced; max_iter=2000
Random Forest	300 trees; max_depth=6; min_samples_split=2; min_samples_leaf=4; max_features=sqrt; class_weight=balanced
XGBoost	200 estimators; learning_rate=0.03; max_depth=3; subsample=0.90; colsample_bytree=0.90; min_child_weight=2; reg_alpha=0; reg_lambda=1; scale_pos_weight=10.138393
LSTM	1 LSTM layer; 64 hidden units; dropout=0.20; Adam learning rate=0.001; training batch size=64; maximum 50 epochs; validation weighted BCE monitored for early stopping; minimum improvement=1×10^{-4}; patience=5; final run stopped after 6 epochs.
Weighted BCE positive weight	10.044776, calculated from 8,076 negative and 804 positive LSTM fitting sequences.
Random seed	42
Classical-model tuning	Validation PR-AUC only; selected LR C=0.1 (0.089369), RF 300/depth 6/leaf 4 (0.091260), XGBoost 200/depth 3/lr 0.03 (0.078695)
Rolling feature window	12 observations, corresponding to approximately 12 minutes at the median 60-second sampling interval.

Table 2(b). Complete 31-feature predictor set.

Feature Group	Exact Variables Included
Current state	CPU_Usage; Memory_Usage; Disk_IO; Network_IO; Resource_Saturation
CPU temporal	CPU_Usage_lag1; CPU_Usage_lag2; CPU_Usage_lag3; CPU_Usage_mean; CPU_Usage_var; CPU_Usage_slope
Memory temporal	Memory_Usage_lag1; Memory_Usage_lag2; Memory_Usage_lag3; Memory_Usage_mean; Memory_Usage_var; Memory_Usage_slope
Disk temporal	Disk_IO_lag1; Disk_IO_lag2; Disk_IO_lag3; Disk_IO_mean; Disk_IO_var; Disk_IO_slope
Network temporal	Network_IO_lag1; Network_IO_lag2; Network_IO_lag3; Network_IO_mean; Network_IO_var; Network_IO_slope
Burst/cross-resource	CPU_Burst; CPU_Memory_Corr

Table 3. Dataset summary and descriptive statistics.

Metric	CPU Usage (%)	Memory Usage (%)	Disk I/O	Network I/O	Anomaly Label
Count	14,400	14,400	14,400	14,400	14,400
Mean	35.562	44.153	12.631	7.994	0.087
Std. Dev.	19.211	15.300	9.503	6.460	0.282
Minimum	0.00	3.99	0.00	0.53	0
25th Percentile	24.158	34.130	7.110	4.510	0
Median (50th)	31.340	41.630	10.680	5.310	0
75th Percentile	39.630	49.702	14.882	6.530	0
Maximum	100.00	97.14	69.70	42.40	1

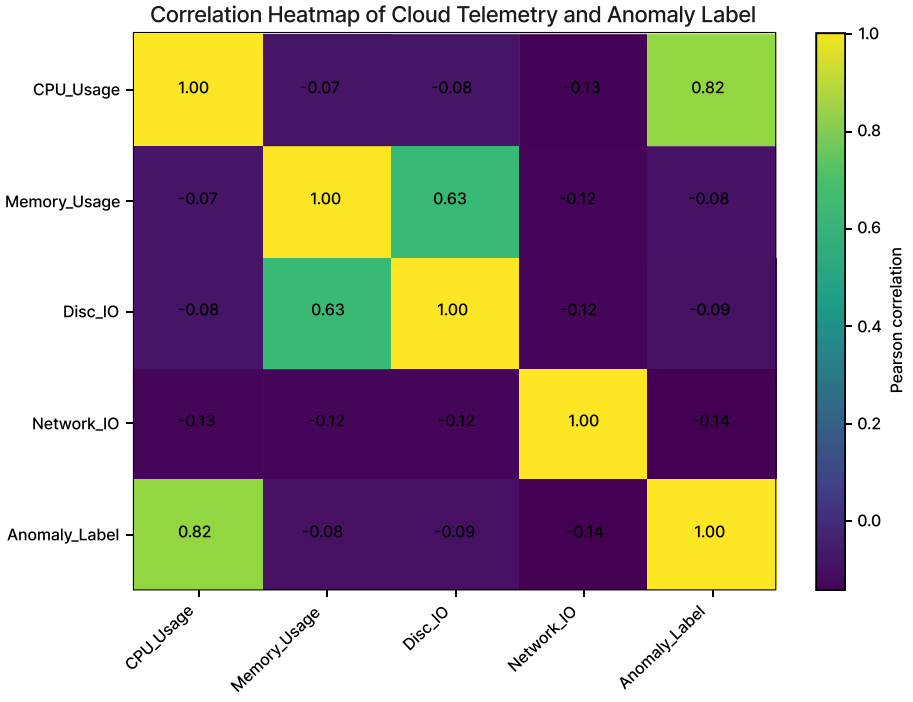

Fig. (2). Pearson correlation heatmap showing relationships between cloud resource metrics and anomaly labels. The x-axis and y-axis represent telemetry variables, while colour intensity represents correlation coefficients ranging from negative to positive association.

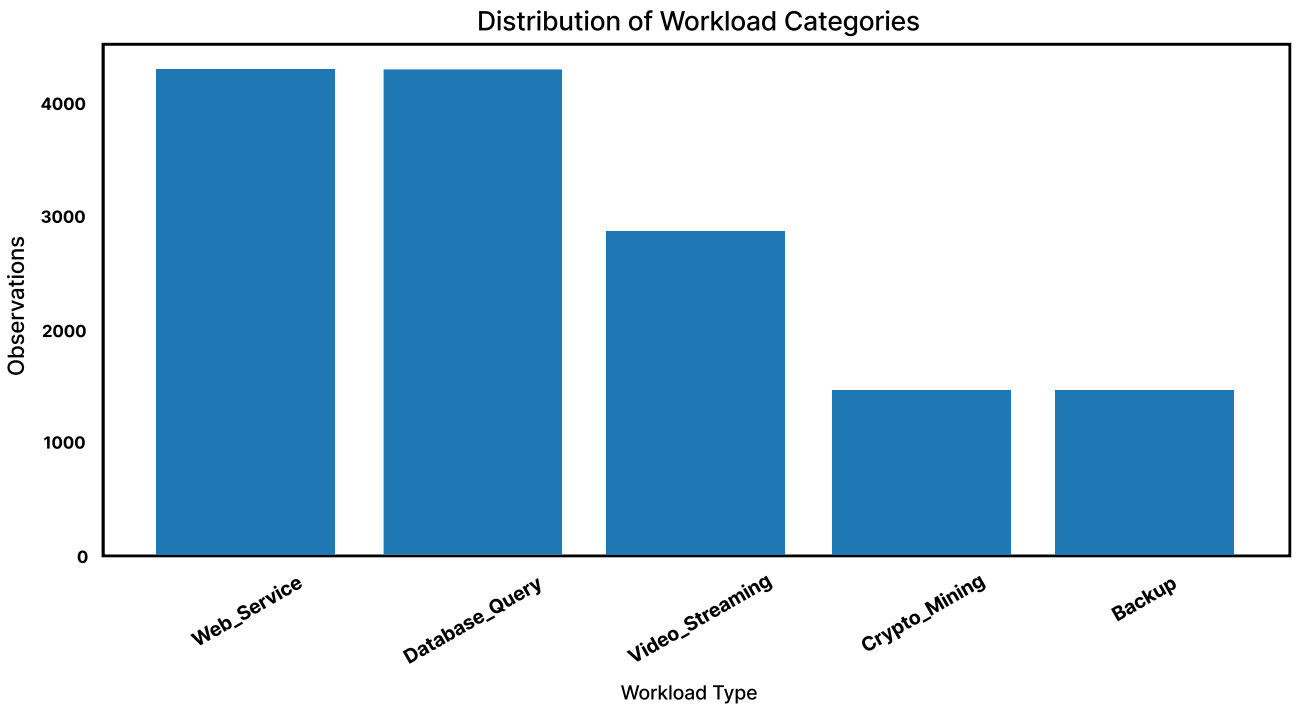

Fig. (3). Distribution of workload categories within the cloud telemetry dataset. The x-axis represents workload categories, while the y-axis represents the number of observations associated with each workload type.

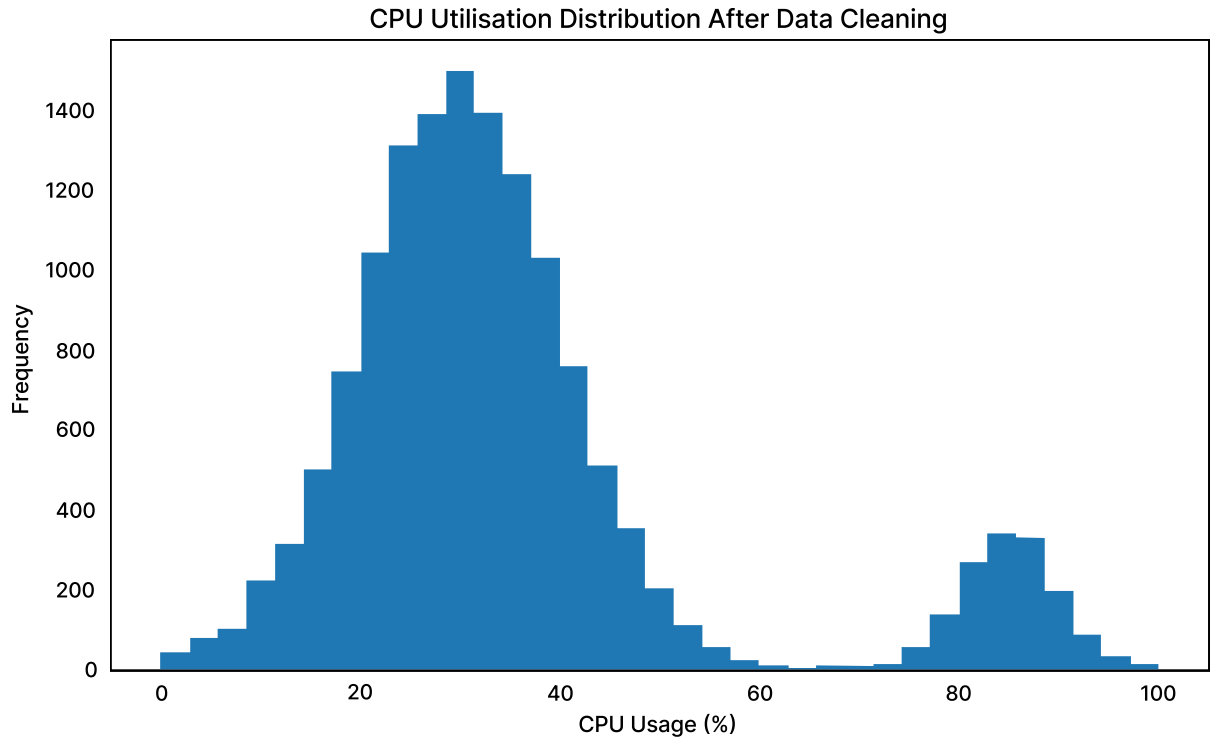

Fig. (4). Histogram showing CPU utilisation distribution across telemetry observations. The x-axis represents CPU utilisation values (%), and the y-axis represents observation frequency.

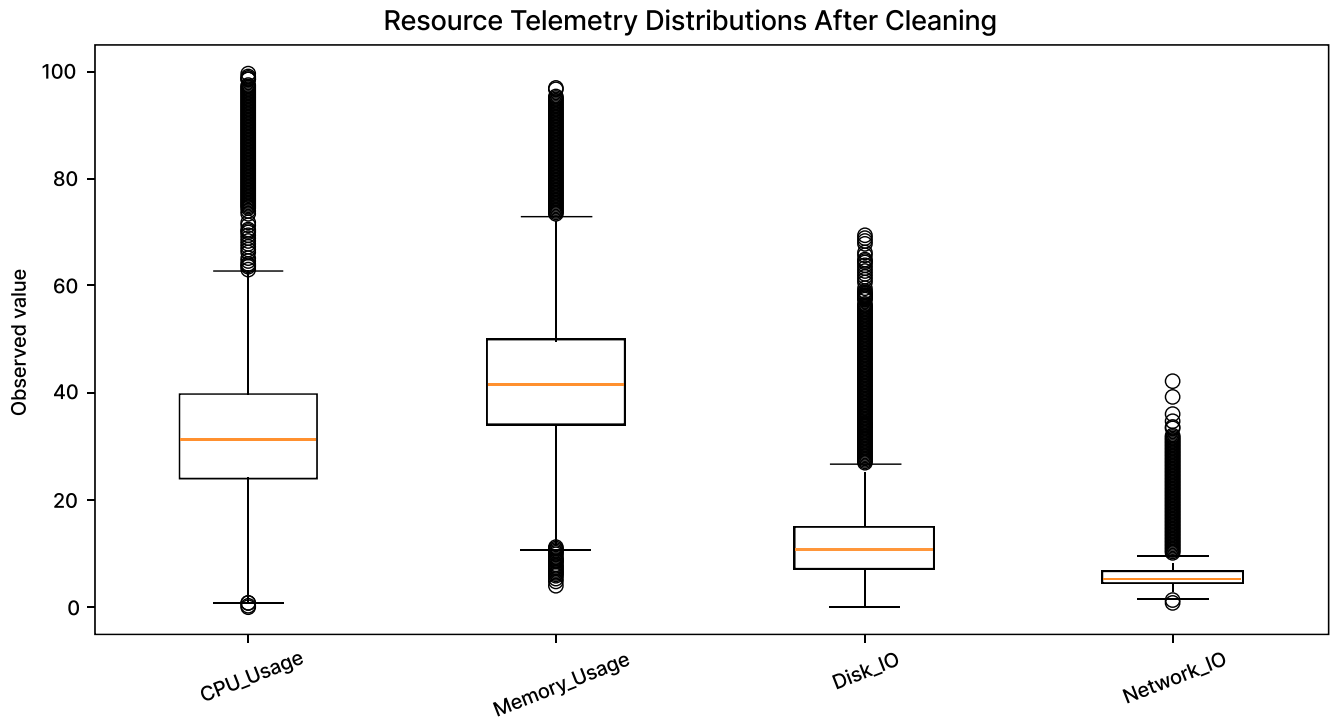

Fig. (5). Box plot comparison of CPU usage, memory usage, disk I/O, and network I/O distributions. The x-axis represents cloud resource metrics, while the y-axis represents observed utilisation values and variability.

A service-dependability point of view, a detection of volatility caused by bursts is crucial for determining if a workload's volatility is benign or part of sustained degradation. This separation can decrease the number of alerts that are not necessary and still remain sensitive to any ongoing instability.

4.4. Model Performance

There is little predictive discrimination between models in all four models on the corrected exact t+3 target. LSTM had the best performance in terms of F1 (0.1421) and recall (0.5513) and Logistic Regression had the best PR-AUC (0.0874). Random Forest was the most conservative model, with a false-alert rate of 0.1029, and recall of 0.0968. These results should be considered in the context of the held out positive prevalence of 8.1775% and not as proof of good production ready prediction.

An increased false-alert rate can lead to alarm fatigue and may result in an inefficient use of resources since engineers have to respond to many false alarms. Valuing false alarms reduction may help to make more effective incident response and reduce the cognitive load, but the impact on the operation of such reductions should be validated in production environments.

The results in Fig. (**6**). demonstrate the different trade-offs between false alerts and sensitivity; LSTM has the highest F1 and Recall, Logistic Regression has the highest PR-AUC and Random Forest has the lowest False Alert rate. To compare the performance of Logistic Regression, Random Forest, XGBoost and LSTM on the same 4,170 aligned held-out sequences with fixed threshold of 0.50, Table **4**(**a**) is presented. The results indicate generally weak predictive performance across all four models, reflecting the difficulty of correctly identifying the positive class. LSTM achieved the highest recall (0.5513) and F1-score (0.1421), suggesting superior sensitivity, although this was accompanied by the highest false-alert rate (0.5529). Logistic Regression produced a comparable F1-score (0.1408) and achieved the highest PR-AUC (0.0874), indicating slightly better overall precision–recall discrimination. XGBoost recorded the highest precision (0.0837) but lower recall (0.3460). Random Forest showed the lowest false-alert rate (0.1029), but its substantially reduced recall (0.0968) demonstrates a conservative detection strategy that misses many positive cases.

Table **4**(**b**) results show that its contribution varies across models rather than producing a consistent improvement. For Logistic Regression, inclusion slightly increased PR-AUC from 0.0874 to 0.0890 and reduced the false-alert rate, but precision, recall, and F1 declined. Random Forest benefited most in precision, recall, and F1, with F1 increasing from 0.0859 to 0.1000, although PR-AUC marginally decreased. XGBoost also improved modestly across precision, recall, F1, and PR-AUC after workload inclusion. In contrast, LSTM performance weakened, with recall falling from 0.5513 to 0.4927 and F1 from 0.1421 to 0.1378. Overall, workload information appears more useful for tree-based models than for the linear and recurrent architectures.

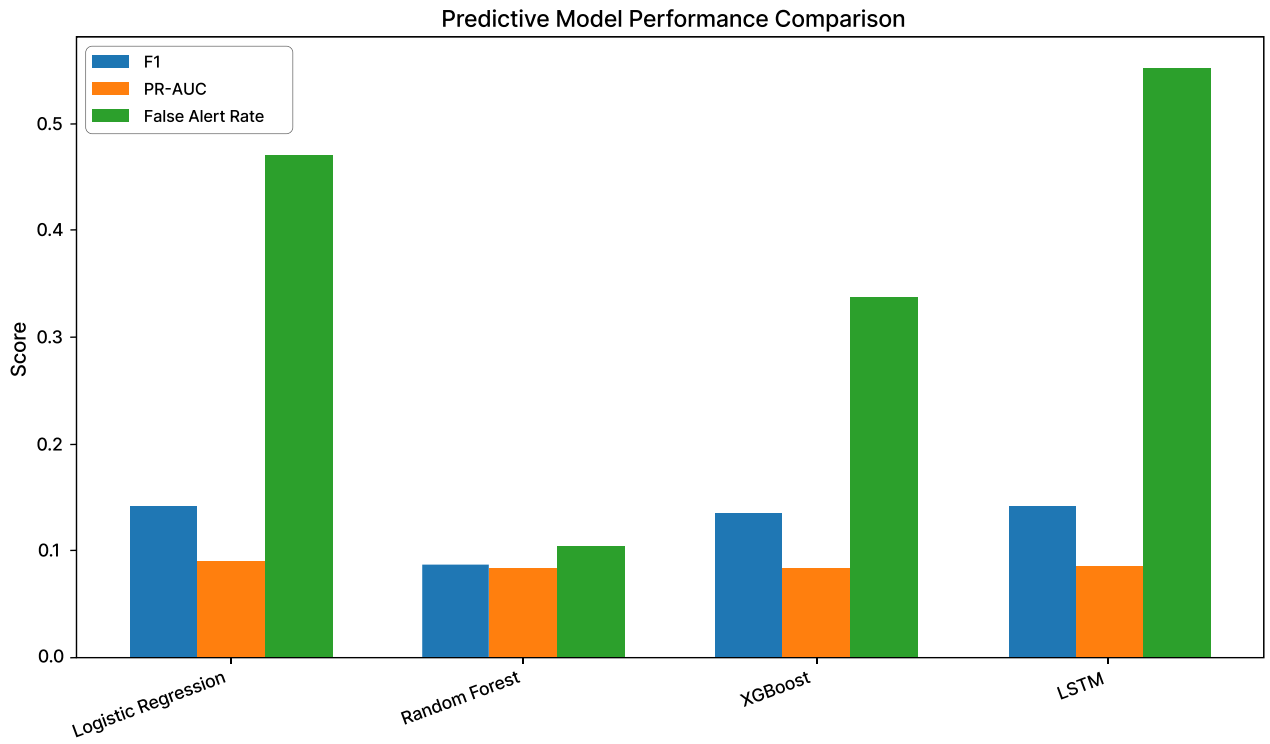

Fig. (6). Comparative held-out performance of Logistic Regression, Random Forest, XGBoost, and LSTM using the corrected exact t+3 target. Bars show F1, PR-AUC, and false-alert rate from the final implementation.

Table 4(a). Model performance comparison.

Model	Precision	Recall	F1-Score	PR-AUC	False-Alert Rate
Logistic Regression	0.0827	0.4751	0.1408	0.0874	0.4696
Random Forest	0.0773	0.0968	0.0859	0.0825	0.1029
XGBoost	0.0837	0.3460	0.1349	0.0822	0.3372
LSTM	0.0816	0.5513	0.1421	0.0845	0.5529

Table 4(b). Workload-feature inclusion sensitivity.

Model	Workload Feature	Precision	Recall	F1	PR-AUC	False-Alert Rate
Logistic Regression	Excluded	0.0827	0.4751	0.1408	0.0874	0.4696
Logistic Regression	Included	0.0789	0.4487	0.1343	0.0890	0.4662
Random Forest	Excluded	0.0773	0.0968	0.0859	0.0825	0.1029
Random Forest	Included	0.0888	0.1144	0.1000	0.0818	0.1045
XGBoost	Excluded	0.0837	0.3460	0.1349	0.0822	0.3372
XGBoost	Included	0.0858	0.3578	0.1384	0.0847	0.3395
LSTM	Excluded	0.0816	0.5513	0.1421	0.0845	0.5529
LSTM	Included	0.0801	0.4927	0.1378	0.0853	0.5040

4.5. Feature Ablation Analysis

An ablation study was conducted to determine whether predictive performance was driven mainly by the LSTM architecture or by engineered temporal features. Feature groups were removed systematically while the model architecture was kept constant. This design isolates the contribution of lag features, rolling statistics, and cross-resource interactions.

Table **5** shows small, non-monotonic ablation effects. Removing lag features increased F1 from 0.1421 to 0.1442, and removing rolling statistics increased F1 to 0.1449 and PR-AUC to 0.0893. The raw-metrics-only configuration produced F1=0.1385. Accordingly, the ablation results do not demonstrate a universal predictive gain from engineered temporal features; their principal value in this dataset may instead lie in representation diversity, cross-resource characterisation, and interpretability rather than consistent improvement in aggregate F1 or PR-AUC.

4.6. Robustness and Statistical Validation Results

Robustness was evaluated quantitatively using Gaussian-noise perturbation, workload-specific subgroup analysis, temporal-segment evaluation, and paired bootstrap resampling. All robustness tests used the held-out chronological test partition, and the trained Logistic Regression, Random Forest, XGBoost, and LSTM models were not re-estimated during robustness testing. Table **6** presents Gaussian-noise robustness results, Table **7**(**a**) reports workload-specific performance, Table **7**(**b**) presents temporal-segment robustness, Table 7(c) reports sustained-anomaly sensitivity, and Table 8 presents the paired-bootstrap comparisons between the LSTM and the baseline models.

4.6.1. Gaussian-Noise Robustness Test

For feature j, the perturbed test-set value was calculated as:

$$x'_{t,j} = x_{t,j} + \epsilon_{t,j}, \epsilon_{t,j} \sim \mathcal{N}\left(0, \alpha^2 \sigma_j^2\right) \quad (10)$$

where $\alpha = 0.10$ and σ_j denotes the training-set standard deviation of feature j. Zero-mean Gaussian noise equal to 10% of the feature-specific training standard deviation was added independently to the raw held-out CPU usage, memory usage, disk I/O, and network I/O values. Physical bounds were then reapplied and all lag, rolling, slope, burst, and cross-resource features were recomputed from the perturbed telemetry before scoring the already fitted models.

Gaussian-noise perturbation produced only small performance changes. F1 changed by +0.0042 for Logistic Regression, -0.0082 for Random Forest, +0.0006 for XGBoost, and -0.0028 for LSTM. PR-AUC changes were -0.0008, +0.0032, +0.0054, and +0.0002, respectively. These results indicate limited sensitivity to the specified 10% raw-telemetry perturbation, although the low clean-test discrimination remains the more important limitation.

4.7. Workload-Specific Robustness

There were 4170 forecasting observations in the common aligned chronological test set. Workload type was not a part of the principal predictor matrix in the evaluation of each observation, but was only evaluated based on the workload category known at the time of the observation. The numbers of positives in the subgroups add up to 4,170 as do the numbers of positives in the main held-out evaluation set.

For each of the various workloads, performance was different; but none of the workloads experienced the very high positive prevalence that was previously observed in the misaligned target. The percentages of positive cases for Crypto Mining, Database Query, and Database Construction were 31/399, 115/1,293, and 1,105/11,400, respectively. LSTM F1 was found to be between 0.1254 and 0.1498 across workloads, and Logistic Regression achieved a value of 0.1625 in Database Query. Random Forest was generally conservative. No universal superiority of any model is demonstrated in these results, which indicate that there is some variation in workload.

4.8. Temporal Robustness

The common aligned held-out forecasting sequences were split into two non-overlapping temporal subsets: 2080 sequences were in the early time period, and 2090 sequences were in the late time period. There were no significant differences in the positive prevalence between segments (0.0822 and 0.0813). The same fitted models and the same 0.50 thresholds were used in both segments. The performance estimates for the early and late prediction times for all four predictive models are reported in Table **7**(**b**).

Minimal change was observed in the short timeframe results. Logistic Regression F1 changed from 0.1395 to 0.1422, Random Forest from 0.0833 to 0.0885, XGBoost from 0.1344 to 0.1353, and LSTM from 0.1382 to 0.1460. LSTM PR-AUC changed from 0.0838 to 0.0874. Such differences are not a sign of a short horizon collapse, but the set of data is still quite short, around one day, to evaluate long-term concept drift.

The results are controlled evidence of short-term temporal stability for the data that is available. In real-world cloud environments, software configurations, traffic patterns, infrastructure, and workload characteristics can evolve over a longer period, so they don't rule out the possibility of longer-term concept drift.

In the k = 3 and k = 5 cases, there were no positive held-out observations or episodes, so the precision, recall, F1, PR-AUC, false-alert rate, and warning coverage were not estimable.

Table **7**(**c**) indicates that the sustained-anomaly definition significantly lowered prevalence of positive events. With k = 2, there were only 42 positive observations and 21 episodes left to test the predictive performance of all four models, and with k = 3 and k = 5 there were none of the positive held-out cases to assess the models reliably.

4.9. Paired Bootstrap Confidence Intervals

The commonly held-out predictions were subject to a paired bootstrap significance test with 1,000 repetitions. On F1 and PR-AUC, LSTM was compared with Logistic Regression and Random Forest with the same resampled indices.

Bootstrap distributions were computed and the 95% confidence intervals are based on the 2.5th and the 97.5th percentiles. Empirical two-sided p-values were computed with a finite-sample correction to avoid reporting zero *p*-values.

Table 5. Feature ablation study results.

Feature Configuration	Precision	Recall	F1-Score	PR-AUC
Full Model (All 31 Features)	0.0816	0.5513	0.1421	0.0845
Without Lag Features	0.0838	0.5161	0.1442	0.0846
Without Rolling Statistics	0.0845	0.5073	0.1449	0.0893
Without Cross-Resource Features	0.0787	0.4223	0.1327	0.0801
Raw Metrics Only	0.0886	0.3167	0.1385	0.0849

Table 6. Gaussian-noise robustness results for all predictive models.

Model	Precision	Recall	F1-Score	PR-AUC	False-Alert Rate	ΔF1	ΔPR-AUC	Condition
Logistic Regression	0.0852	0.4868	0.1450	0.0866	0.4657	+0.0042	-0.0008	Gaussian noise
Random Forest	0.0716	0.0850	0.0777	0.0858	0.0982	-0.0082	+0.0032	Gaussian noise
XGBoost	0.0852	0.3314	0.1355	0.0877	0.3171	+0.0006	+0.0054	Gaussian noise
LSTM	0.0800	0.5396	0.1393	0.0847	0.5526	-0.0028	+0.0002	Gaussian noise

Table 7(a). Workload-specific performance for all predictive models.

Workload	Model	N	Positive	Precision	Recall	F1	PR-AUC
Backup	Logistic Regression	444	37	0.0792	0.4324	0.1339	0.1101
Backup	Random Forest	444	37	0.0000	0.0000	0.0000	0.0893
Backup	XGBoost	444	37	0.0894	0.2973	0.1375	0.0953
Backup	LSTM	444	37	0.0810	0.4595	0.1377	0.0857
Crypto Mining	Logistic Regression	399	31	0.0674	0.5806	0.1208	0.0702
Crypto Mining	Random Forest	399	31	0.0513	0.1290	0.0734	0.0788
Crypto Mining	XGBoost	399	31	0.0757	0.4516	0.1296	0.0700
Crypto Mining	LSTM	399	31	0.0699	0.6129	0.1254	0.0815
Database Query	Logistic Regression	1,293	115	0.0968	0.5043	0.1625	0.1050
Database Query	Random Forest	1,293	115	0.0926	0.1304	0.1083	0.0944
Database Query	XGBoost	1,293	115	0.0892	0.3739	0.1441	0.0966
Database Query	LSTM	1,293	115	0.0870	0.5391	0.1498	0.0973
Video Streaming	Logistic Regression	821	66	0.0826	0.4242	0.1383	0.0786
Video Streaming	Random Forest	821	66	0.0435	0.0303	0.0357	0.0799
Video Streaming	XGBoost	821	66	0.0885	0.2576	0.1318	0.0846
Video Streaming	LSTM	821	66	0.0813	0.5606	0.1420	0.0755
Web Service	Logistic Regression	1,213	92	0.0759	0.4565	0.1302	0.0984
Web Service	Random Forest	1,213	92	0.0876	0.1304	0.1048	0.0794
Web Service	XGBoost	1,213	92	0.0773	0.3587	0.1272	0.0802
Web Service	LSTM	1,213	92	0.0809	0.5761	0.1419	0.0946

Paired bootstrap resampling showed no statistically significant difference between LSTM and Logistic Regression for F1 (difference 0.0013, 95% CI [-0.0141, 0.0159], $p = 0.836$) or PR-AUC (difference -0.0029, 95% CI [-0.0140, 0.0062], $p = 0.528$). LSTM exceeded Random Forest significantly for F1 (difference 0.0562, 95% CI [0.0299, 0.0842], $p = 0.002$), but not for PR-AUC (difference 0.0020, $p = 0.682$). LSTM and XGBoost were statistically indistinguishable for F1 (difference 0.0072, $p = 0.488$) and PR-AUC (difference 0.0023, $p = 0.724$) (Table **8**).

The conclusion is thus not generalisable that the LSTM is superior. The only difference that was statistically significant among the tested differences was LSTM *vs.* Random Forest (F1). The differences between LSTM and all three baselines (random, most recent, and most popular) were not statistically significant at 0.05. Overall, the robustness analyses indicate moderate temporal sensitivity for short time horizons, moderate perturbation, and large differences in alerting trade-offs between models and workloads, and limited discrimination. These analyses are helpful to the internal diagnostic process but cannot replace longer-duration, independently labelled production telemetry.

4.10. Explainability Results

The final SHAP GradientExplainer analysis was successfully carried out on 200 test sequences of LSTM, resulting in an attribution array of shape (200, 12, 31). Global mean absolute SHAP importance, averaged across all 200 sequences and 12 time steps, ranked CPU_Memory_Corr first (0.004278), followed by Disk_IO_var (0.002954), Disk_IO_lag3 (0.002615), Memory_Usage_lag3 (0.002381), CPU_Usage_slope (0.002136), and CPU_Usage_lag3 (0.001908). The explanation thus suggests a representation of a mixed cross-resource and temporal mechanism, not just a mechanism dominated by CPU saturation.

Fig. (**7**) shows that both cross-resource dependence and lagged/variability terms contribute to the LSTM risk score. CPU_Memory_Corr was the strongest global feature, while disk variance, disk lag 3, memory lag 3, and CPU slope also ranked highly. These attributions improve transparency but should be interpreted as model explanations for this dataset rather than causal evidence of operational failure mechanisms.

Once production validated, these feature attributions could enable directed operational investigation, suggesting whether a forecast is related to correlated CPU-memory behaviour, variability of storage, or recent changes of time. This present study does not require automatic mitigation from SHAP values as it does not target an actual outage label, but rather an anomaly proxy.

4.11. Episode-Onset Warning Coverage at the Prespecified Horizon

In this case, since the forecasting goal was set as t+3, then the forecasts were awarded for the prespecified nominal three-minute horizon. For each user stream, the positive aligned targets were clustered into contiguous episodes of maximal length such that there was one negative target between positive targets. Advance-warning credit was only given to the prediction that was made before an episode hit; there was at most one prediction per episode that could be given advance warning; there was no duplication of advance warning credit for predictions that occurred at the same time; and predictions after an episode hit did not count as advance warning. Each set was constructed using exactly the same episode onsets as the held-out set, with 174 onsets correctly predicted in the held-out set, resulting in 54.375% episode-onset coverage. This analysis therefore focuses on coverage of the onset of episodes, rather than modelling an empirical distribution of lead times. The mean standard deviation of the rolling prediction was 0.018.

The interpretation of operational actionability must be taken with great care. The mean model-only inference latency for Logistic Regression was less than 0.001ms per sample, while it was around 0.028ms per sample, 0.003ms per sample and 0.037ms per sample for Random Forest, XGBoost, and LSTM, respectively. Benchmarking of latency was conducted with one warm up inference call, and 30 full test-set timing calls, for Logistic Regression, Random Forest and XGBoost and 20 full test-set timing calls for LSTM, with each call scoring all 4170 held out samples aligned by their order in the training set. Finally, this implies that model inference is negligible compared to the nominal three-minute horizon, but data ingestion, feature construction, transport, orchestration and, potentially, human decision time would be part of the end-to-end production latency.

Fig. (**8**) shows LSTM risk probabilities for a single user stream held-out for the given example. The average standard deviation of the rolling prediction results was 0.018 across the test data. The forecasting target was set as t+3, meaning that all credited warnings were for the specified nominal 3 minute time horizon and all 174 of the 320 episode onsets were correctly warned. This 54.375% onset coverage is a significant improvement over the previously misaligned estimate and does not represent a guarantee of operational warning for a high proportion of positive events, therefore a nominal horizon is not to be read as an operational horizon.

The results of robustness analyses demonstrated only moderate degradation from Gaussian measurement noise and minor differences between the early and late temporal segments. However, when analysing the workloads, there were variations in performance between service profiles and no single model was consistently better in each category of workload. This is, therefore, evidence of controlled test-set robustness, and not evidence of an overall LSTM advantage or evidence of production-scale generalisation.

5. DISCUSSION

5.1. Interpretation of Findings

The final results show that the LSTM predictions are mainly caused by both cross-resource dependence and temporal telemetry, not only by CPU usage. The most influential feature was CPU_Memory_Corr, followed by Disk_IO variance, Disk_IO lag 3, Memory_Usage lag 3 and CPU slope. Although CPU is still relevant and the corrected explanation can't be used

to make the claim that current CPU usage or resource saturation is explicitly dominating the predictive mechanism everywhere [13, 24, 25].

In addition to CPU-related features, variables related to memory, disk and network are also included. The relative SHAP values should not be interpreted as causal mechanisms but as associations for this model and dataset. A multivariate pattern allows for analysis of correlated patterns of subsystem behaviour, but no independent data with incident labels would be available for determining if these patterns of behaviour always precede real degradation of service [5, 24].

5.2. Model Comparison Insights

There is no clear winner for the corrected comparison. The best PR-AUC (0.0874) was obtained by Logistic Regression, while the best F1 (0.1421) and recall (0.5513) was obtained by LSTM. Random Forest had the lowest false-alert rate (0.1029) with very low recall (0.0968) and XGBoost was in the middle tier. No significant differences were found except for the F1 score of LSTM which was better than Random Forest in Bootstrap testing. The PR-AUC values are relatively close to the positive prevalence of 8.1775%, suggesting that the discrimination in the models is not large.

Table 7(b). Temporal robustness results for all predictive models.

Segment	Model	N	Positive Rate	Precision	Recall	F1	PR-AUC
Early	Logistic Regression	2,080	0.0822	0.0816	0.4795	0.1395	0.0854
Early	Random Forest	2,080	0.0822	0.0751	0.0936	0.0833	0.0807
Early	XGBoost	2,080	0.0822	0.0835	0.3450	0.1344	0.0831
Early	LSTM	2,080	0.0822	0.0793	0.5380	0.1382	0.0838
Late	Logistic Regression	2,090	0.0813	0.0838	0.4706	0.1422	0.0961
Late	Random Forest	2,090	0.0813	0.0794	0.1000	0.0885	0.0871
Late	XGBoost	2,090	0.0813	0.0840	0.3471	0.1353	0.0834
Late	LSTM	2,090	0.0813	0.0838	0.5647	0.1460	0.0874

Table 7(c). Sustained-anomaly sensitivity analysis.

k	Model	Positive Prevalence	Recall	F1	PR-AUC	False-Alert Rate	Episodes	Warning Coverage
2	Logistic Regression	0.010072	0.4762	0.0237	0.0135	0.3937	21	0.5238
2	Random Forest	0.010072	0.0476	0.0430	0.0128	0.0119	21	0.0476
2	XGBoost	0.010072	0.1429	0.0155	0.0130	0.1756	21	0.1905
2	LSTM	0.010072	0.5238	0.0238	0.0306	0.4319	21	0.5238
3	All models	0.000000	N/E	N/E	N/E	N/E	0	N/E
5	All models	0.000000	N/E	N/E	N/E	N/E	0	N/E

Table 8. Paired bootstrap comparison of LSTM performance against baseline models.

Comparison	Metric	Observed Difference	95% Confidence Interval	Two-Sided Bootstrap *p*-value	Significant
LSTM - Logistic Regression	F1	0.0013	[-0.0141, 0.0159]	0.836	No
LSTM - Logistic Regression	PR-AUC	-0.0029	[-0.0140, 0.0062]	0.528	No
LSTM - Random Forest	F1	0.0562	[0.0299, 0.0842]	0.002	Yes
LSTM - Random Forest	PR-AUC	0.0020	[-0.0086, 0.0127]	0.682	No
LSTM - XGBoost	F1	0.0072	[-0.0141, 0.0309]	0.488	No
LSTM - XGBoost	PR-AUC	0.0023	[-0.0103, 0.0144]	0.724	No

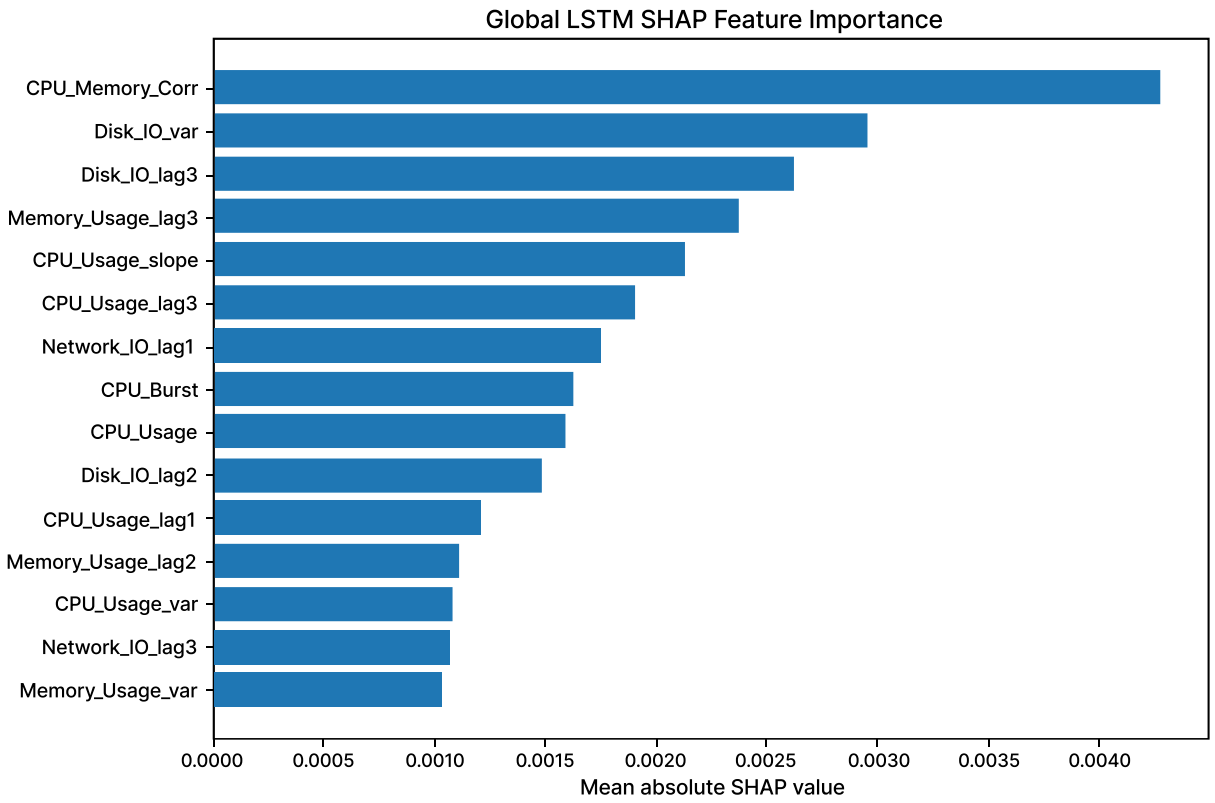

Fig. (7). Global LSTM SHAP feature importance from the final implementation. Bars report mean absolute GradientExplainer SHAP values aggregated across 200 held-out sequences and all 12-time steps.

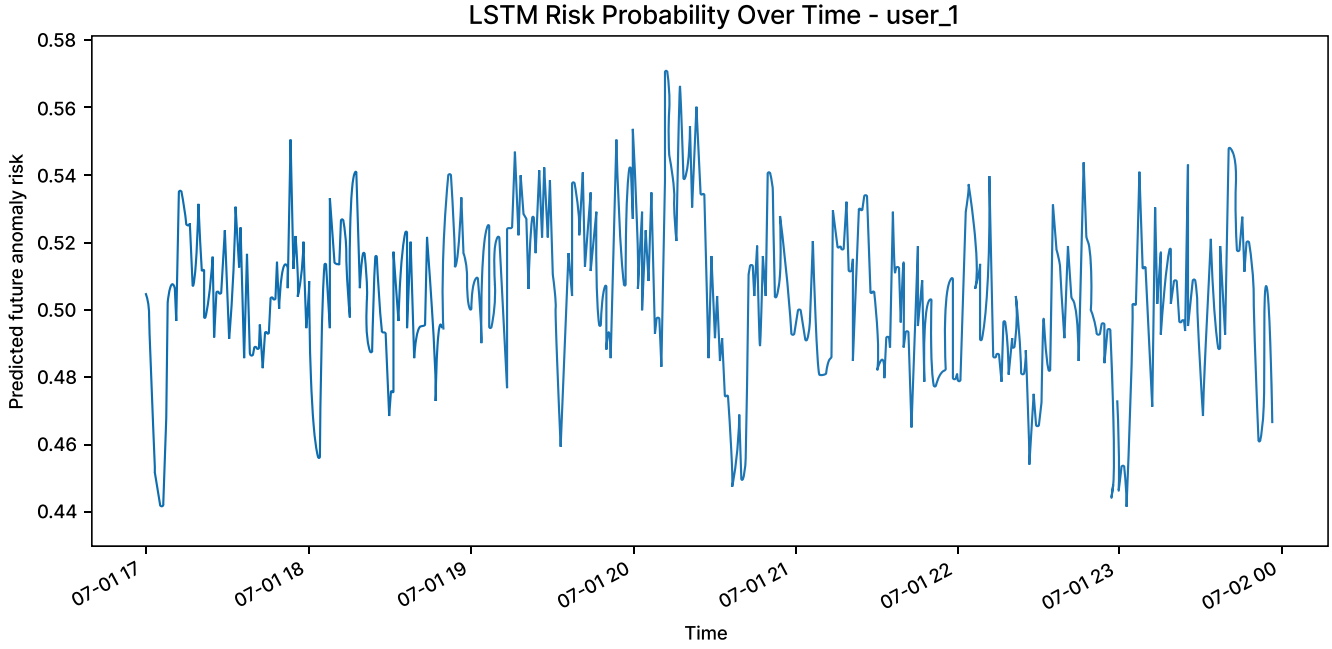

Fig. (8). LSTM predicted future anomaly-risk probability over time for one held-out user stream, generated by the final implementation.

5.3. Operational Implications

The results show a potential additional monitoring signal for use through the use of predictive analytics, but the corrected results are not sufficiently discriminative to be used for autonomous operational intervention. A nominal three-minute horizon and 54.375%-episode onset warning coverage indicates that some episodes of anomalies can be anticipated and many others won't be warned. Any future scaling, workload redistribution, or deployment review should therefore rely on more robust validation against known incidents, and make explicit operational cost/benefit thresholds.

Random Forest performed the best at the lowest false alarm rate (0.1029), however at the same time having the lowest recall (0.0968) showing the trade-off between conservatively alerting and missing anomalous states. Logistic Regression and LSTM yielded significantly higher recall as well as very high false-alert rate. Alert burden and missed-event costs should thus be taken into account in the operational model selection. It should be noted that future research could also evaluate probability calibration using the calibration curve, the Brier score, the expected calibration error, or the calibration slope/calibration intercept to test the reliability of the predictiveness of the probability of an event.

5.4. Comparison with Prior Work

The present study is distinct from previous studies in three practical ways, but not in terms of a new algorithmic approach. First, the temporal nature of anomaly is considered as an antecedent event, rather than a final event, in analysis, which changes the focus from anomaly detection to outage-risk forecasting [18, 19]. This difference is pertinent because there is a time delay for the detection-oriented methods. Second, it is a multivariate approach instead of a single metric method, which enables the representation of dependencies between dimensions of resources in the cloud [5, 11, 19, 24].

Third, the framework is integrated into the predictive process, making explainability an intrinsic part of it. SHAP is used to explain which features are driving LSTM predictions while the evaluation also indicates false-alert behaviour, coverage at the episode level, temporal robustness and workload sensitivity. The contribution is rather an actual, explainable and forecasting AIOps workflow than a new algorithm or evidence for production-ready anomaly prediction.

LIMITATIONS AND FUTURE DIRECTIONS

There are some limitations in this study that limit the degree of its interpretability. Most importantly, labels for anomalies are not necessarily labels for confirmed outages by the customer, but rather only for overuse or abnormal behaviour that is hidden from the customer [20]. The public dataset description does not give a precise deterministic rule for generating the labels and the last audit revealed an unusually high correlation between the label and the workload type: there are 1257 raw positives in the "Crypto Mining" workload type, and an empirical rule "Crypto Mining, CPU ≥ 80%" accounted for all but one of those positives. Type of workload was therefore not included in the main models, but this label structure is still a significant disadvantage for the dataset-validity. This restriction is not remedied by the label wording, excluding features from the model or tuning the model alone, as this restriction is due to the structure of the available labels.

Another constraint is that the predictive task could be a reflection in part of the structure of the anomaly-label generation process and not of actual operational degradation forecasting. The empirical Crypto Mining plus CPU ≥ 80% rule generates almost all the labels found in the anomalous data, and thus the models could be trained on a subset of the telemetry characteristics that are a result of this rule, rather than a set of independent characteristics that precede service failures. Thus, the predictive performance should not be taken as an indicator of the outage forecasting skill of the general type. The framework should be validated in future research with independently labelled data sets of confirmed incidents, service degradation or events impacting customers.

There are two factors that limit external validity: the dataset's provenance and the period of time it covers. It includes 14,400 rows, but, for the final implementation, it is possible to count only 10 user streams and 1,440 distinct one-minute timestamps, covering about 24 hours. This is too short of a time period to observe weekly or seasonal cycles, long-term concept drift, architectural changes, rare outage precursors or changing workload regimes. The independent time coverage is not compensated by overlapping windows. Because of the same temporal dependence, the ordinary paired bootstrap could also be a bias estimator of uncertainty compared with a block-bootstrap or user-stratified time-series resampling procedure in the future this validation should be conducted using dependence-aware resampling. Another modelling limitation is that the architectures did not receive the same amount of historical receptive field, and future benchmarking should ensure the same amount of historical information for both architectures to distinguish architecture from the amount of historical information. A more robust validation should, therefore, be done with a much larger production or independent public dataset with known incident/outage records. Also not simulated in the dataset is a service dependency, deployment changes, multi-tenant interference, and event responses by humans in real-world enterprise environments.

In addition to the external validation, it is important to have internal diagnostics that are strong, such as Gaussian-noise testing, workload-specific evaluation, workload-inclusion sensitivity, temporal segmentation, sustained-anomaly sensitivity, and paired bootstrap testing. The low held out PR-AUC values (0.0822-0.0874) are just a little higher than the positive prevalence (0.0818), meaning discrimination is weak. These results are therefore to be regarded as evidence to support proof-of-concept and not as evidence to support the reliability of deployment.

The data set employed in this study provides well-controlled evaluation of predictive modelling techniques but does not

provide a full representation of enterprise cloud environments. Therefore, the reported results should be taken as proof of the feasibility of an explainable forecasting workflow and not as an indicator of deployment readiness.Future work efforts should consider adding more modalities of data, such as system logs, traces, alerts, and IT service management (ITSM) incident information. Incorporating quantitative telemetry together with system logs, traces, alerts and textual incident histories, can help better discriminate between benign anomaly and true precursors of service degradation. Continual learning and online learning methods for adapting predictive models to non-stationarity due to new workloads, infrastructure changes, and changing service behaviours should also be studied.

Another promising approach for modelling the propagation of localised operational stress in interconnected microservices is by means of service-dependency graphs. Graph-aware temporal models may enable to model localised stress propagation through service topologies and improve prediction and root cause analysis. These extensions, in combination with external validation on production scale telemetry and human-in-the-loop evaluation by reliability engineers, might give greater confidence in the operational applicability.

CONCLUSION

This paper explored explainable forecasting of anomalous operational states by multivariate cloud telemetry. The dataset does not contain outage records confirmed by the customer, but rather anomaly labels, thus the empirical task is the forecasting of anomalies (which can represent possible degradation) and the result is considered a proxy indicator of possible degradation. LSTM achieved the highest F1 (0.1421) and recall (0.5513), Logistic Regression achieved the highest PR-AUC (0.0874), XGBoost achieved F1 = 0.1349 and PR-AUC = 0.0822, and Random Forest achieved F1 = 0.0859 and PR-AUC = 0.0825. A significant LSTM advantage over RF was discovered only for F1 *via* bootstrap analysis while all LSTM differences were non-significant for the other three metrics. SHAP analysis showed that CPU–memory correlation and disk/memory temporal features are the most important features to the LSTM, while episode analysis achieved 54.375% coverage at the prespecified t+3 nominal horizon.

The key contribution is a documented AIOps workflow integrating telemetry cleaning, 31-feature leakage-controlled temporal engineering, user-specific chronological splitting, validation-only model selection, exact three-step-ahead forecasting, workload-excluded principal modelling, robustness analysis, sustained-anomaly sensitivity, paired bootstrap comparison, SHAP explainability, episode-level warning analysis, stability measurement, and inference-latency benchmarking. The approximately one-day temporal coverage, weak PR-AUC values, strong workload-label association, and absence of confirmed outage labels prevent claims of production outage prediction or long-term generalisation; larger independent datasets with transparent label provenance and real incident records are required before operational adoption can be established.

LIST OF ABBREVIATIONS

CPU	=	Central Processing Unit
ITSM	=	IT Service Management
KPIs	=	Key Performance Indicators
LSTM	=	Long Short-Term Memory
PR-AUC	=	Precision-Recall Area Under the Curve
SLA	=	Service Level Agreement
SLOs	=	Service-Level Objectives
SRE	=	Site Reliability Engineering
SHAP	=	Shapley Additive exPlanations

AUTHOR'S CONTRIBUTION

The author A.G.M. contributed to the theoretical framework, methodology, data analysis, results interpretation and writing up of the manuscript.

ETHICAL APPROVAL & INFORMED CONSENT

Not applicable.

AVAILABILITY OF DATA AND MATERIALS

The data will be made available on reasonable request by contacting the corresponding author [A.G.M.].

FUNDING

None.

CONFLICT OF INTEREST

The author serves as a member of the journal's Editorial Board. To ensure an unbiased review process, the author was not involved in the peer review or editorial decision-making of this manuscript, which was handled independently by another editor.

ACKNOWLEDGEMENTS

Declared none.

DECLARATION OF AI

During the preparation of this manuscript, the author utilized ChatGPT to support language enhancement and editorial refinement. The author thoroughly evaluated, verified, and revised all AI-assisted content and accepts full responsibility for the accuracy, originality, and integrity of the final manuscript.

REFERENCES

[1] T. R. Merlo, F. Fard, and S. Hawamdeh, "Cloud Computing's Impact on the Digital Transformation of the Enterprise: A Mixed-Methods Approach," *Sustainability,* vol. 17, no. 13, Art. no. 5755, 2025, https://doi.org/10.3390/su17135755

[2] V. Gharibvand *et al.*, "Cloud based manufacturing: A review of recent developments in architectures, technologies, infrastructures, platforms and associated challenges," *Int J Adv Manuf Technol.,* vol. 131, pp. 93-123, 2024, https://doi.org/10.1007/s00170-024-12989-y

[3] A. Q. Khan, M. Matskin, R. Prodan, C. Bussler, D. Roman, and A. Soylu, "Cost modelling and optimisation for cloud: a graph-based approach," *J Cloud Comput.,* vol. 13, no. 1, Art. no. 147, 2024, https://doi.org/10.1186/s13677-024-00709-6

[4] F. Qazi, D. Kwak, F. G. Khan, F. Ali, and S. U. Khan, "Service Level Agreement in cloud computing: Taxonomy, prospects, and challenges," *Internet of Things.,* vol. 25, Art. no. 101126, 2024, https://doi.org/10.1016/j.iot.2024.101126

[5] J. Soldani and A. Brogi, "Anomaly detection and failure root cause analysis in (micro) service-based cloud applications: A survey," *ACM Comput. Surv.*, vol. 55, no. 3, Art. no. 59 pp. 1-39, 2022, https://doi.org/10.1145/3501297

[6] S. Ahuja, V. H. Lopez Chalacan, and H. Resendez, "Performance Variability in Public Clouds: An Empirical Assessment," *Information.,* vol. 16, no. 5, Art. no. 402, 2025, https://doi.org/10.3390/info16050402

[7] F. Voutsas, J. Violos, and A. Leivadeas, "Mitigating alert fatigue in cloud monitoring systems: A machine learning perspective," *Comput Netw.,* vol. 250, Art. no. 110543, 2024, https://doi.org/10.1016/j.comnet.2024.110543

[8] H. He, X. Li, P. Chen, J. Chen, M. Liu, and L. Wu, "Efficiently localizing system anomalies for cloud infrastructures: a novel dynamic graph transformer based parallel framework," *J Cloud Comput.,* vol. 13, Art. no. 115, 2024, https://doi.org/10.1186/s13677-024-00677-x

[9] T. N. Tengku Asmawi, A. Ismail, and J. Shen, "Cloud failure prediction based on traditional machine learning and deep learning," *J Cloud Comput.,* vol. 11, Art. no. 47, 2022, https://doi.org/10.1186/s13677-022-00327-0

[10] J. Diaz-De-Arcaya, A. I. Torre-Bastida, G. Zárate, R. Miñón, and A. Almeida, "A joint study of the challenges, opportunities, and roadmap of mlops and aiops: A systematic survey," *ACM Comput Surv.*, vol. 56, no. 4, Art. no. 84, pp. 1-30, 2023, https://doi.org/10.1145/3625289

[11] A. Hussien Ali, H. Almisbahi, E. Alkayal, and A. Almakky, "Enhancing Real-Time Anomaly Detection of Multivariate Time Series Data via Adversarial Autoencoder and Principal Components Analysis," *Electronics.,* vol. 14, no. 15, Art. no. 3141, 2025, https://doi.org/10.3390/electronics14153141

[12] S. Ali *et al.* "Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence," *Inf Fusion.,* vol. 99, Art. no. 101805, 2023, https://doi.org/10.1016/j.inffus.2023.101805

[13] S. Deng *et al.* 'Cloud-Native Computing: A Survey From the Perspective of Services,' *Proceedings of the IEEE*, vol. 112, no. 1, pp. 12–46, 2024, https://doi.org/10.1109/JPROC.2024.3353855

[14] L. M. Barata, S. Sequeira, E. Lopes, P. R. Inácio, and M. M. Freire, "Anomaly detection and root-cause identification in microservices: a survey," *Cluster Comput.*, vol. 29, Art. no. 309, 2026, https://doi.org/10.1007/s10586-026-06095-9

[15] B. Treynor Sloss, S. Nukala, and V. Rau, 'Metrics That Matter,' *Commun ACM.*, vol. 62, no. 4, pp. 88, 2019, https://doi.org/10.1145/3303874

[16] S. Alharthi, A. Alshamsi, A. Alseiari, and A. Alwarafy, "Auto-scaling techniques in cloud computing: Issues and research directions," *Sensors,* vol. 24, no. 17, Art. no. 5551, 2024, https://doi.org/10.3390/s24175551

[17] T. Lu, L. Wang, and X. Zhao, "Review of anomaly detection algorithms for data streams," *Appl Sci.,* vol. 13, no. 10, Art. no. 6353, 2023, https://doi.org/10.3390/app13106353

[18] H. Almajed, A. Alsaqer, and A. Albuali, 'Towards Effective Anomaly Detection: Machine Learning Solutions in Cloud Computing,' *Int J Adv Comput Sci Appl.*, vol. 16, no. 2, 2025, https://doi.org/10.14569/IJACSA.2025.01602132

[19] F. Wang, Y. Jiang, R. Zhang, A. Wei, J. Xie, and X. Pang, "A survey of deep anomaly detection in multivariate time series: taxonomy, applications, and directions," *Sensors*, vol. 25, no. 1, Art. no. 190, 2025, https://doi.org/10.3390/s25010190

[20] AJ. Qiu, H. Shi, Y. Hu, and Z. Yu, "Enhancing anomaly detection models for industrial applications through SVM-based false positive classification," *Appl Sci.,* vol. 13, no. 23, Art. no. 12655, 2023, https://doi.org/10.3390/app132312655

[21] M. Altalhan, A. Algarni, and M. T.-H. Alouane, "Imbalanced data problem in machine learning: A review," *IEEE Access,* vol. 13, pp. 13686-13699, 2025, https://doi.org/10.1109/ACCESS.2025.3531662

[22] T. Mehmood, S. Latif, N. S. M. Jamail, A. Malik, and R. Latif, "LSTMDD: an optimized LSTM-based drift detector for concept drift in dynamic cloud computing," *PeerJ Comput Sci.,* vol. 10, pp. e1827, 2024, https://doi.org/10.7717/peerj-cs.1827

[23] B. Lim and S. Zohren, "Time-series forecasting with deep learning: a survey," *Philos Trans R Soc A.,* vol. 379, no. 2194, Art. no. 20200209, 2021, https://doi.org/10.1098/rsta.2020.0209

[24] V. Medel, U. Arronategui, O. Rana, J. Á. Bañares, and R. Tolosana-Calasanz, "Modeling and characterizing service interference in dynamic infrastructures," *IEEE Access.,* vol. 11, pp. 21387-21403, 2023, https://doi.org/10.1109/ACCESS.2023.3250606

[25] Y. Gebreyesus, D. Dalton, D. De Chiara, M. Chinnici, and A. Chinnici, "AI for automating data center operations: model explainability in the data centre context using shapley additive explanations (SHAP)," *Electronics.,* vol. 13, no. 9, Art. no. 1628, 2024, https://doi.org/10.3390/electronics13091628

[26] J. Kim, H. Maathuis, and D. Sent, "Human-centered evaluation of explainable AI applications: a systematic review," *Fron Artif Intell.,* vol. 7, Art. no. 1456486, 2024, https://doi.org/10.3389/frai.2024.1456486

[27] J. Tang, Y. Liu, X. Huang, Z. Jiang, and F. Wu, "Explainable AI for post-hoc and pseudo-post-hoc predictive maintenance of governor valve actuators," *Sci Rep.,* vol. 15, no. 1, Art. no. 39504, 2025, https://doi.org/10.1038/s41598-025-23346-8