**Journal of Computer Science and Technology Studies**
ISSN: 2709-104X
DOI: 10.32996/jcsts
Journal Homepage: www.al-kindipublisher.com/index.php/jcsts

JCSTS
AL-KINDI CENTER FOR RESEARCH AND DEVELOPMENT

**| RESEARCH ARTICLE**

# Streaming Telemetry Analytics for AI-Driven Service Performance Prediction in Large-Scale Microservices Architectures

**Manisha Konda[1]✉ and Kamalakar Reddy Singi[2]**
*[1]Senior Analyst, Analytics Starcom (Publicis Groupe) manishakonda1999@gmail.com*
*[2]Senior Software Engineer Valparaiso University kamalakarreddy.singi@valpo.edu*
**Corresponding Author**: Manisha Konda, **E-mail**: manishakonda1999@gmail.com

**| ABSTRACT**
Distributed systems today, such as cloud infrastructures, microservice architectures, and hybrid cyber-physical networks, are increasingly difficult to monitor with respect to fault detection and diagnostics because of their large size, complexity, and dynamics. Telemetry data, a record of detailed operational metrics and event logs, is important for understanding how a system behaves and for proactively maintaining it. The paper describes an AI-based telemetry system capable of predicting service performance in a microservices system. The architecture leverages streaming telemetry, such as system metrics, logs, and traces, from cloud-native setups. The data are cleaned and normalized, and their features are extracted using Principal Component Analysis (PCA) to enhance data quality and reduce dimensionality. To learn temporal patterns of dependencies and identify anomalies in service behavior, an LSTM-based deep learning model is employed. A privacy protection layer is a safety measure that ensures the secure handling of sensitive information. The presented model achieves 94% accuracy with excellent predictive performance, low detection time, and few false positives. The comparative analysis reveals that the framework is superior to existing solutions and can be used to reliably monitor microservice performance in real time.

**| KEYWORDS**
Microservices Architecture, Service Performance Prediction, Telemetry Data, Artificial Intelligence, Machine Learning, Privacy Preservation, Real-Time Monitoring.

**| ARTICLE INFORMATION**
**ACCEPTED:** 01 June 2026 **PUBLISHED:** 03 July 2026 **DOI:** 10.32996/jcsts.2026.8.8.6

## 1.Introduction

Over the last ten years, distributed software systems have become more popular due to the growing need for Internet of Things applications and advancements in cloud infrastructure [1][2]. As a new architectural style that encourages the use of fine-grained services and collaboration across cloud nodes, microservice architecture (MSA) has arisen in response to this need and in reaction to advancements in service-oriented architecture (SOA) [3][4][5]. Software applications that use SOA or monolithic style often encounter issues with availability, fault tolerance, scalability, and maintainability [6][7]. The architecture of microservices is gaining interest in both academia and business[8], and it is often contrasted with monolithic design [9][10][11]. Many of the findings of these research publications contradict each other in terms of performance of various designs. Network devices are producing an increasing amount of data, including control, statistics, and user data[12][13].

Telemetry data is essential for maintaining the integrity of spacecraft systems and guaranteeing the success of spacecraft missions. Therefore, to guarantee the safe functioning of spacecraft subsystems, it is essential to promptly identify and notify of any unexpected occurrences connected to their functionality [14]. There have been a number of approaches to anomaly identification in spacecraft telemetry data monitoring that have been developed in the last few years [15][16]. After many decades, the commercial and military telemetry groups reaped the rewards of their investment in these standards.

The microservices architecture is radically different in approach compared to traditional monolithic applications, as it involves a system of distributed services, each of them encapsulating a specific business function or capability [17][18][19]. It has become very popular because of its flexibility, scalability, and adaptability regarding the handling of complex software demands across

various industries. Recent developments in artificial intelligence (AI) offer bright solutions to such problems. Reinforcement learning (RL), predictive analytics (PA), and evolutionary algorithms (EA) are some of the AI methods that promise more advanced resource optimization methods [20][21][22]. The benefit of RL is that systems can learn optimal policies for allocating resources through interactions with their environment and feedback on the outcomes of their actions. Artificial intelligence (AI) is a tool that can potentially offer solutions to these challenges by automating and improving the creation of microservice architectures, especially in new software development [23][24]. Machine learning (ML), and natural language processing (NLP) are used as AI techniques to help with critical design activities, including requirement analysis, service identification, and architectural decision-making [25][26]. There has been a convergence of telemetry data with data mining, ML, and DL over the past ten years. This partnership aims to implement new automated techniques (on-board) and semi-automated techniques (at the ground station). Some of the space systems' health monitoring processes can be automated using these methods[27][28]. Many DM, ML, and DL techniques have been proposed in recent years to address telemetry anomaly detection and health monitoring in aerospace systems.

The key contributions of this research are as follows:

- Design of a real-time AI-driven telemetry analytics framework for service performance prediction in large-scale microservices architectures using streaming metrics, logs, and traces.
- Integration of a Kubernetes-native observability pipeline, leveraging container orchestration environments (Kubernetes clusters with Prometheus and OpenTelemetry) for real-world cloud deployment compatibility[29].
- Development of a hybrid deep learning model (PCA + LSTM) enhanced with a privacy-preserving layer, enabling accurate, scalable, and secure performance prediction with reduced dimensionality and temporal dependency learning.

### *1.1 Motivation and Novelty of the Study*

Large-scale microservice architectures are being used more and more in modern cloud-native applications to provide scalability, flexibility, and rapid deployment, but they are distributed and highly dynamic and therefore difficult to ensure that service performance is consistent. Conventional methods of monitoring with fixed limits and reactive alarms sometimes fail to notice small-scale anomalies or anticipate the existence of performance decreases in real-time especially when workloads fluctuate, and there exist complicate inter-service relationships. In the meantime, microservice-based systems continuously produce vast amounts of telemetry data, such as CPU utilization, memory usage, network throughput, and latency, and manual analysis is not feasible, and traditional methods are not powerful enough to discover nonlinear temporal relationships. Recent changes in artificial intelligence and deep learning promise to do this prediction of performance in advance based on incoming data, but many current systems do not scale to deploy at large scale or respond to performance in real time.

The novelty and validity of this study are the sophistication and changeability of the microservices architecture and the necessity to address the issue with an artificial intelligence-based telemetry framework. The proposed algorithm is a combination of live stream telemetry logs and an AI-based controlled layered model to predict service performance accurately and detect anomalies by detecting temporal dependencies in system behavior. It is a state-of-the-art with regards to privacy, unlike alternative methods; it merges end-to-end pre-processing, PCA-based feature reduction, and a privacy-preserving layer. The design enables the high-volume telemetry data to be analyzed appropriately, quick and scalable and with low overhead on monitoring. As the existing approaches usually prove to be unwieldy, fail to capture the time-related features, or cause excessive expenses, the proposed framework provides a critical and proactive answer to the problem of performance prediction by making system stability, downtime reduction, and the efficient management of the cloud-native microservices environment.

### *1.2 Organization of the Paper*

The paper is structured in the following way: Section II provides a review of related literature on AI-based telemetry analytics frameworks in Microservices Performance. Section III explains the methodology, i.e. data pre-processing, model development and evaluation. In Section IV, the results of the experiments and comparison of the models are given. Section V wraps up with fundamental conclusions and recommendations about future research on enhancing the efficiency of commercial computing systems.

## 2. Literature Review

The literature shows the enhancement of real-time performance, fault detection, scalability, and efficiency through the use of AI-driven telemetry, predictive analytics, and microservices architectures, exposing gaps in predictive performance modeling and edge deployment.

S. O. Awodele et al. (2026) introduce RCASage, an AI-augmented inference engine for autonomous root cause analysis in microservices. LSTM/NLP anomaly detection, and an Autonomous Inference Engine integrating GNNs with Neural Granger Causality. RCASage minimizes the MTTR by more than 90 percent, offers clear causal explanations with XAI, and addresses DevOps with JIT defect predictions [30].

M. K. Gaddam (2025) proposes a scalable observability architecture specifically designed to support AI-driven microservices, that is, the natural opacities and dynamics of AI workloads. The framework proposed provides an opportunity to achieve better visibility over the whole AI pipeline, including data ingestion and model inference by combining fine-grained telemetry with trace correlation, anomaly detection, and model-centric metrics. The methodology shows enhanced root cause analysis, decreased system downtime, and heightened openness of model performance behaviour. The architecture is validated through a real-world deployment on a Kubernetes-based AI platform, showing minimal performance overhead while maintaining high fidelity in observability signals [31].

G. Dkmak et al. (2025) introduce an innovative unsupervised method of microservices anomaly detection the Night Watch algorithm. The methodology eliminates significant limitations of existing approaches through temporal considerations and multi-source information. Depending on the size of the training set, their findings show that the Night Watch algorithm greatly enhances recall (39% improvement) and precision (92% improvement). These results show that the algorithm is capable of improving real-time anomaly detection in microservice setup [32].

B. Barua and M. S. Kaiser (2024) create a system for real-time travel reservations using a microservices architecture and prescriptive analytics. By decoupling components, the system achieves modularity, scalability, and fault tolerance. ML models optimize demand forecasting, dynamic pricing, and performance, improving response time, throughput, transaction success, and prediction accuracy. This allows making data-based decisions in time and increasing operational efficiency, which is a blueprint for other multifaceted, data-driven applications [33].

D. K. Pentyala (2024) suggests an AI-based system of fault identification in cloud-optimized data engineering systems, using the techniques of machine learning models to process telemetry, logs and performance indicators in real-time. The framework covers system failures predicting anomalies and optimizing resources utilization minimizing down time and enhancing reliability. Quantitative evaluation reveals it is much better, e.g., it reduces the fault detection latency by 78% and resource efficiency by 65% compared to traditional monitoring plans, which demonstrates the extent to which AI-enabled predictive analytics can be used to monitor cloud services[34].

G. P. Menaud (2023) suggests a hybrid ML-based system that entails the combination of past trace statistics and instant telemetry data to predict and optimize performance. The architecture employs supervised learning to make predictions and reinforcement learning to optimize adaptively. Their analysis and suggested model reveal that with different workloads, ML performs much better in terms of flexibility and efficiency of microservices and decreases the latency and enhances the throughput more than 30 times in simulated deployments[35].

Table I provides a summary of the methods, key results, advantages, constraints, and prospective studies of the recent research and reveals the gaps in predictive analytics and scalable monitoring with the help of microservices.

**Table 1. Research Gaps in AI-Driven Telemetry and Microservices Performance Prediction**

| Author (Year) | Methodology | Key Findings & Advantages | Limitations | Future Work |
|---|---|---|---|---|
| S. O. Awodele et al. (2026) | RCASage: AI-augmented inference engine; three-stage pipeline (telemetry ingestion, LSTM/NLP anomaly detection, Autonomous Inference Engine with GNN + Neural Granger Causality) | Reduces MTTR by >90%; identifies true root causes; integrates Explainable AI (XAI); bridges DevOps via JIT defect prediction | Focused primarily on root cause analysis rather than direct service performance prediction; requires extensive telemetry data | Extend to predictive performance optimization for microservices; integrate edge computing telemetry |
| M. K. Gaddam (2025) | Scalable observability architecture for AI-driven microservices; telemetry, trace correlation, anomaly detection, model-centric metrics; deployed on Kubernetes | Improved visibility across AI pipeline; minimal overhead; reduces downtime; enhances model transparency | Lacks explicit predictive performance modeling; limited evaluation metrics beyond observability | Extend framework to predictive analytics for latency/failure forecasting; incorporate AI-based performance prediction models |
| G. Dkmak et al. (2025) | Night's Watch algorithm: unsupervised anomaly detection in microservices; multi-source data + temporal features | Precision up to 92%, recall up to 39%; reduces false positives; enhances real-time anomaly detection | Focused only on anomaly detection, not root cause analysis or predictive performance; recall is moderate | Integrate predictive models for service degradation; enhance recall and scalability for large microservices systems |
| B. Barua & M. S. Kaiser (2024) | Microservices architecture with predictive analytics; ML models for demand forecasting, dynamic pricing, and system performance optimization | Improves response time, throughput, transaction success; scalable and modular; supports real-time analytics; enhances customer satisfaction | Case study limited to travel reservation systems; does not address cross-service dependencies or telemetry-driven anomaly detection | Extend to AI-driven microservices monitoring and real-time failure prediction; test in other domains like healthcare and industrial systems |

| | | | | |
|---|---|---|---|---|
| D. K. Pentyala (2024) | AI-driven fault identification in cloud-optimized data engineering systems; ML models for telemetry, logs, and performance metrics | Reduces fault detection latency by 78%; increases resource efficiency by 65%; improves reliability | Focused on fault detection, not predictive service performance; limited to cloud infrastructure, not microservices | Integrate with real-time performance prediction in microservices; extend to multi-layered distributed architectures |
| G. P. Menaud (2023) | Hybrid ML framework: historical trace analysis + real-time telemetry; supervised learning for prediction, reinforcement learning for adaptive optimization | Reduces latency and increases throughput by >30%; improves adaptability under varying workloads | Simulated deployment; lacks real-world important microservices validation | Apply in real-world large-scale microservices; integrate edge computing telemetry; include multi-service failure prediction |

## 3. Methodology

The proposed framework is to be used to predict the performance of AI-driven services in the microservices architecture of large scale as illustrated in Fig. 1. The system is deployed to a Kubernetes-based cloud-native environment, with microservices being deployed in the form of containerized workloads. Every service is an independent pod, which is orchestrated by Kubernetes, which supports dynamic scaling, load balancing, and self-healing in fluctuating workload scenarios. The framework begins with gathering of telemetry data where operational metrics such as CPU utilization, memory utilization, network throughput and service latency are continuously emitted by distributed microservices. Prometheus is used to gather telemetry data in terms of metric scraping and OpenTelemetry is used to provide distributed tracing and logging, which keeps the service interactions in the cluster visible end-to-end. The resulting telemetry streams are then worked upon at the data pre-processing stage that involves cleaning of data, feature extraction by application of Principal Component Analysis (PCA) and feature scaling through application of Min-Max normalization in order to achieve a homogeneous representation of the features.. A privacy-preserving layer is also built to anonymize sensitive identifiers within the telemetry streams and then proceed to further analysis. The generated data are then subjected to the AI analytics layer, where the temporal feature encoding and anomaly detection are achieved through the assistance of DL techniques to identify the abnormal behavior of the system and predict service degradation. Lastly, a feedback and model optimization system should be continuous to allow the system to respond to the changing telemetry patterns.

### *3.1 Dataset Description*

The data available in this research is streaming telemetry data that is gathered in cloud-native microservices setups. The data contains system metrics like CPU usage, memory usage, network throughput and service latency, application logs and distributed traces. In Kubernetes-based systems, the monitoring systems can be used to obtain these telemetry signals including OpenTelemetry and Prometheus.

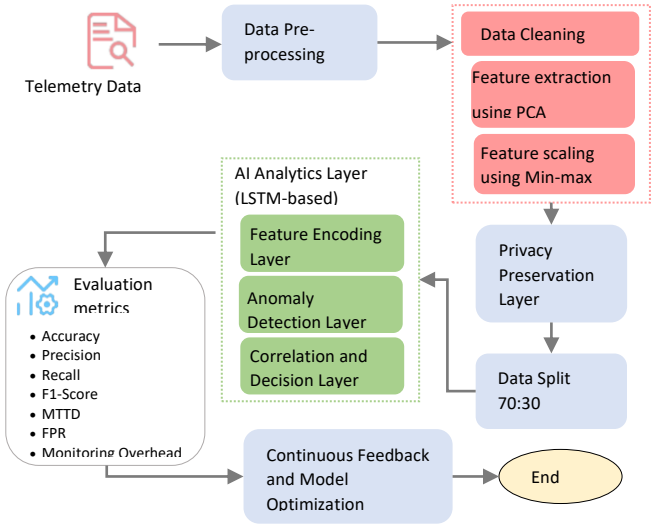


**Figure 1. AI-Based Telemetry Analytics Framework for Microservices Performance**

### *3.2 Data Preprocessing*

The telemetry data has been collected and initially pre-processed to guarantee the quality and consistency of the data. This phase involves a number of data processing steps.

- **Data cleaning:** Telemetry records that are invalid, duplicate, or inconsistent are eliminated, in order to permit consistent observations.
- **Noise filtering:** The use of smoothing methods to minimize spike anomalies, transient peaks and monitoring artifacts in such measurements as CPU usage, latency, and network throughput.
- **Removal of corrupted records:** Discovery and deletion of telemetry records that lack timestamps or do not contain full metric values or have corrupted logs to ensure data integrity.
- **Consistency verification:** Checking the telemetry times and range of metrics to keep the value in reasonable working ranges.
- **Data standardization:** Organizing the cleaned telemetry data into a standard form to extract features easily and then analyze the data using ML.

Pre-processing is important to make sure that abnormal spikes and monitoring artifacts do not adversely affect the learning of models.

### *3.3 Feature Extraction*

Use of feature extraction helps to minimize redundancy in the attributes of telemetry but maintain the most informative attributes of the data set. Correlated metrics, like the use of CPU, consumption of memory, network throughput, and service latency, are found in the telemetry streams in large-scale microservices environments. These associations are capable of raising the complexity of computing and lowering the effectiveness of ML models. Hence, PCA is used as a dimensionality reduction method to condense the initial telemetry feature space into a smaller collection of independent components. PCA determines directions of maximum variance in data and retains significant components, which practically eliminates redundancy and preserves the required patterns for predicting service performance using AI.

### *3.4 Feature Scaling using Min-max*

Normalization of range of telemetry attributes in a microservices dataset is done using feature scaling. All features are then scaled with min-max normalization to range of 0-1. The change guarantees equal input of features and the learning efficiency of the model. The Min–Max normalization formula is defined in Equation (1):

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \qquad (1)$$

Here, X is original feature value, $X_{min}$ and $X_{max}$ are minimum and maximum values of feature, and $X'$ represents the normalized value after scaling.

### *3.5 Privacy Preservation Layer*

To safeguard sensitive operational data, the framework incorporates a privacy-preserving layer before AI processing. Telemetry data can include identifiable data, such as service instance identifiers or infrastructure addresses. Consequently, hash-based pseudonymization is considered to be an anonymization approach that conceals sensitive identifiers. This layer provides secure data processing whilst losing the analytical value of telemetry signals.

After pre-processing, the data is separated into prediction model training/test sets to evaluate prediction model. The telemetry data used in this study is in 70:30 ratio, 30 percent of the data is allocated to testing and evaluation, and 70 percent to training.

### *3.6 AI Analytics Layer*

The AI analytics layer is the central element of the suggested framework to predict service performance degradation in large-scale microservice architectures. This layer has three sublayers:

#### *3.5.1 Feature Encoding Layer*

The feature encoding layer transforms the telemetry data which has been processed into structured sequential inputs that are accepted by LSTM network. Distributed microservice telemetry, such as CPU usage, memory usage, service latency, and network traffic, is arranged in time-ordered sequences reflecting the behavior of a system over time[36]. This sequential form helps the LSTM to capture patterns and time dependencies to be ready to achieve effective learning and prediction of performance.

#### *3.5.2 Anomaly Detection Layer*

An LSTM network is used as the anomaly detection layer to learn the past system behavior and detect abnormal deviations in the telemetry patterns that are signs of performance degradation. LSTMs are time-series models that are also suitable because they are able to capture long-term dependencies by using internal memory structures to model them. The functions of LSTM

such as forget gate, input gate, candidate cell state, cell state update, output gate, and hidden state are described collectively in Equations (2) to (7), and make the model effective in terms of capturing the temporal relationships and detecting anomalies:

$$forget\ gate = f_t = \sigma\big(W_f.[h_{t-1}, x_t] + b_f\big) \quad (2)$$

$$Input\ gate = \ i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \quad (3)$$

$$Cell\ Candidate = \tilde{C}_t = \tanh(W_c.[h_{t-1}, x_t] + b_c) \quad (4)$$

$$Cell\ State\ update = \ C_t = f_t.C_{t-1} + i_t.\tilde{C}_t \quad (5)$$

$$Output\ gate = \ o_t - \sigma(W_o.[h_{t-1}, b_o] + b_o) \quad (6)$$

$$hidden\ state = h_t = o_t.tanh(C_t) \quad (7)$$

The LSTM network through these gated operations captures time relationships in telemetry signals and detects abnormal deviations in service performance metrics.

### *3.5.3 Correlation and Decision Layer*

The final prediction on system performance conditions is produced by the correlation and decision layer. This layer interprets the representations learned by the LSTM and checks the correlation between various telemetry metrics, in order to decide whether the system is functioning normally or there is a performance degradation in the system. This layer compares the trends of telemetry patterns that are observed between various monitoring attributes to generate final anomaly warnings and service performance forecasts. These forecasts can be used to proactively monitor through the detection of possible failures of the services that might cause a major effect on the operations of the systems.

### *3.7 Performance Assessment Metrics*

The standard ML metrics are used to evaluate the performance of the proposed framework. In order to understand these metrics, one should define the key terms:

- **TP (True Positives):** This represents the number of service performance degradations that are predicted as an anomaly, and are correct.
- **FP (False Positives):** The number of normal system behaviors that are falsely predicted to be anomalous.
- **FN (False Negatives):** These are service performance degradations that are predicted to be normal.
- **TN (True Negatives):** The count of behaviors of the normal system that were rightly forecasted as normal.
-

The performance measures using these are:

***Accuracy:*** The ratio of properly identified measurements is the statistic used to assess the system's efficiency; it is given in Equation (8):

$$Accuracy = \frac{TP+TN}{TP+FN+FP+TN} \quad (8)$$

***Precision:*** The accurately anticipated positive instances are measured by this metric. Equation (9) measures it as the ratio of the properly categorized values (TP) to the total forecasted values (TP + FP):

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP+FP}} \quad (9)$$

***Recall :*** Sensitivity is a measure that takes into account both the number of items in the dataset (TP + FN) and the proportion of properly categorized values (TP/FN). Therefore, Equation (10) shows the formula:

$$Recall = \frac{TP}{TP+FN} \quad (10)$$

***F1 score :*** This statistic, which goes by many names than just "F Measure," is weighted average of two metrics: recall and precision. Equation (11) yields a minimum of 0, and a maximum of 1, indicating the classifier's optimal performance:

$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \qquad (11)$$

**MTTD (Mean Time to Detection):** The average time taken by the system to detect anomalies or performance degradations, with lower values indicating faster detection. Mathematically, represented in Equation (12):

$$\text{MTTD} = \frac{Total\ Detection\ Time\ for\ All\ Events}{Number\ of\ Detected\ Events} \qquad (12)$$

**False Positive Rate (FPR):** The proportion of normal system behaviors incorrectly flagged as anomalies. A more dependable monitoring system is indicated by a lower FPR. Formulated in Equation (13).

$$FPR = \frac{FP}{FP + TN} \qquad (13)$$

**Monitoring Overhead:** The ratio of extra system resources used by the monitoring framework, which is efficient in a large-scale microservices setting with the formula presented in Equation (14).

$$\text{Overhead } (\% = \frac{\text{Monitoring Resource Usage}}{Total\ System\ Resource} \times 100 \qquad (14)$$

This measure guarantees the suggested framework's continued effectiveness in extensive microservices environments.

### *3.8 Continuous Feedback and Model Optimization*

The last step of the methodology involves a continuous feedback to improve the performance of the model over time. The events of the real system are compared with the predictions of the AI model with a perspective of identifying the errors in prediction. The model can be retrained according to the feedback with new telemetry information gathered hence giving it the ability to adapt to varying system behaviors. It is a constant learning process that will result in the framework being highly predictive in a dynamic microservices environment.

## 4. Result Analysis and Discussion

These findings show that the PCA-based dimensionality reduction and LSTM-based temporal modeling are effective in enhancing feature representation and sequential dependency learning in a microservice telemetry environment of Kubernetes. All experiments have been carried out in Python on Scikit-Learn and an edge computer system using an ARM-based processor and 16 GB RAM that have been optimized for low-latency inference. Since Table II shows that the balance between predictive accuracy and operational efficiency is very high, this implies the applicability of the proposed framework to real-time microservices monitoring with low detection latency and controlled overhead.

**Table 2. Predictive Performance of the Proposed Model in Microservices-Based Service Performance Forecasting**

| Metric | Proposed AI Framework |
|---|---|
| Accuracy | 0.94 |
| Precision | 0.93 |
| Recall | 0.90 |
| F1-Score | 0.91 |
| MTTD (min) | 5.8 |
| False Positive Rate | 0.07 |
| Monitoring Overhead | 21% |

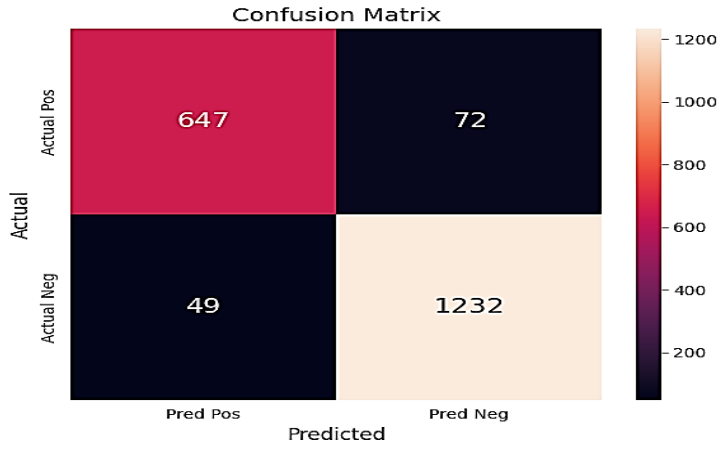


**Figure 2. Confusion Matrix of the Proposed Model**

The confusion matrix in Fig. 2 is well balanced and there is a high proportion of correct classifications in the main diagonal, which demonstrates a high separability between the normal and degraded service states. The relatively small values of false positives (72) and false negatives (49) confirm the strength of the model to reduce the error of misclassification, which is important to ensure the reliability of cloud-native monitoring systems.

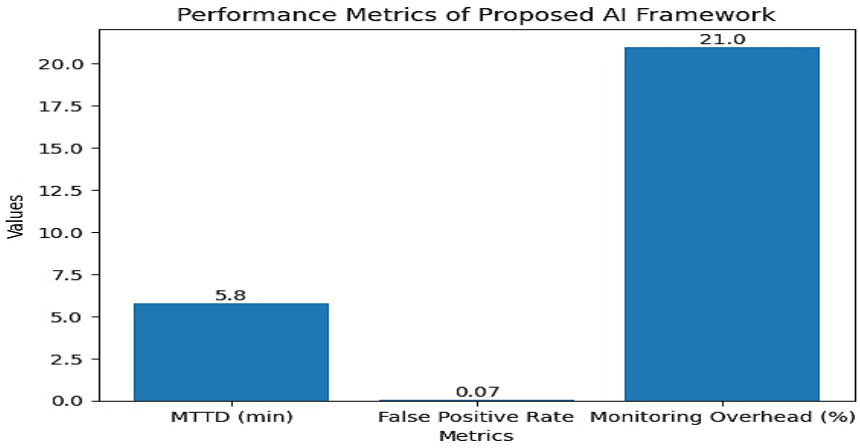


**Figure 3. Performance of the Proposed AI Framework**

Fig. 3 shows the efficiency of the proposed framework in operations with a low Mean Time to Detect (5.8 minutes) and makes it possible to respond practically to service degradation in dynamic microservices environments. The fact that the false positive rate (0.07) is low guarantees a high level of alertness whereas the monitoring overhead (21%), is a practical trade-off between computational cost and predictive performance in edge-enabled applications.

### *4.1 Comparative Analysis*

The comparative outcomes indicate that the proposed system is the most effective in terms of all evaluation measures compared to the existing baseline strategies. According to Table III, the proposed model outperforms classic machine learning baselines, such as Random Forest (RF), Gradient Boosting (GB), and Isolation Forest. RF and GB have moderate precision and recall, but cannot discover temporal dependencies in telemetry data streams. Isolation Forest, which is an unsupervised approach, has lower recall because it has a low ability to distinguish complex performance degradation patterns. However, the suggested LSTM-based model demonstrates better and balanced performance, being able to capture the temporal relationships and increase the accuracy of service performance prediction in microservices environments significantly.

**Table 3. Comparative Analysis of Service Performance Across Microservices Architectures Using Telemetry Data**

| Ref. | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| RF[37] | - | 85 | 78 | 81 |
| GB[38] | - | 89 | 86 | 87.5 |
| Isolation forest[39] | 90.19 | 78.42 | 60.35 | 68.59 |
| Proposed | 94.0 | 93.0 | 90.0 | 91.0 |

The proposed AI-based telemetry model demonstrates obvious superiority to the existing solutions, providing more robust predictive ability, reliability, and detection. Compared to the methods that have been previously used, the proposed model demonstrates a reduced or partial level of achievement on the key metrics, but it is more balanced and robust, which basically reduces the number of errors in addition to the possibility to detect the issue and use resources more efficiently. This highlights its relevance to predicting the performance of microservices accurately and on-demand within a real-time environment.

### *4.2 Discussion*

The experimental findings confirm the usefulness of the suggested AI-based telemetry framework to enhance predictive accuracy and real-time responsiveness of microservices environments, and the results have demonstrated classification accuracy, precision, recall and F1-score of 90 to 94 percent. Moreover, Confusion Matrix verification shows that the system is extremely strong, and a significant percentage of correct classifications and a small percentage of misclassifications are obtained, which means that it is possible to discriminate between normal and degraded service states successfully. The fact that the mean time to detect anomalies is low (5.8 minutes) indicates that this framework is capable of near real-time anomaly detection that is essential to dynamic cloud-based services. The low false positive rate (0.07) also provides assurances of reliability with minimal unnecessary alerts, and the 21% monitoring overhead provides a reasonable balance between system performance and resource utilization. The performance of the new model outperforms all previous studies in terms of each of the key metrics, which demonstrates the value of combining existing pre-processing methods (e.g., PCA and normalization), privacy preservation and LSTM-based temporal learning methods for accurate, timely service performance prediction.

The proposed framework has major benefits and implications for the real-world implementation of very large-scale microservices architectures, including providing the ability to monitor systems proactively, detect performance degradation quickly, and ensure improved operational reliability by using intelligent analysis of telemetry data. Implementation of privacy-preserving mechanisms will also make it ideal for use in sensitive cloud environments where data security is of utmost importance[40]. Despite these benefits, several limitations must also be addressed. First, because LSTM models tend to use computationally intensive computations, they may run into challenges with regard to scalability and performance on extremely large or high-frequency telemetry streams. Secondly, although acceptable, monitor overheads may still affect systems that are constrained with respect to available resources; for example, edge systems. Thirdly, the ability of the model to perform is dependent upon having high-quality and diverse training data, which could limit its ability to generalize across differing kinds of infrastructure or workloads that have not already been experienced. In future research, work may involve lightweight architecture exploring, adaptive learning techniques and cross-domain validation to improve scaling, efficiency and robustness of such frameworks.

## 5. Conclusion and Future Scope

Telemetry data is very heterogeneous, and it is produced in very large amounts, thus, it is hard to analyze without sophisticated analytical models. To conclude, the paper presents a telemetry model based on AI that predicts service performance in microservice large-scale architecture. The framework utilizes streaming telemetry data and other powerful pre-processing algorithms like cleaning, normalization, and PCA-based feature extraction. The LSTM-based model has the ability to capture the temporal dependencies and identify anomalies in the system behaviour. Moreover, a privacy-saving layer will provide safe data processing in a manner that does not interfere with the analysis. The presented framework has a high predictive accuracy, low false positives and can be detected at a high speed with a satisfactory monitoring overhead. Its superiority in comparison to other current methods has been confirmed, and it is therefore appropriate for monitoring and management of microservices in real time in the cloud native environments. The suggested AI-powered telemetry model has major implications in real-life cloud-native systems, especially on Kubernetes-based production systems like financial services, e-commerce systems, healthcare systems, and IoT systems. The system allows performance degradation to be caught proactively, reducing downtime, improving service-level agreement (SLA) compliance, and the user experience in applications where latency is a concern. Moreover, the privacy-sensitive telemetry processing is integrated into the framework, which means that it can be applied to industries that work with sensitive information related to operations or users, and it will not violate the data security principles, while being highly accurate in terms of analysis. Future work will focus on enhancing scalability through lightweight deep learning architectures such as transformers, enabling adaptive real-time learning, and improving edge–cloud collaboration for distributed intelligence in microservice ecosystems.

**Funding**: Please add: "This research received no external funding" or "This research was funded by NAME OF FUNDER, grant number XXX" and "The APC was funded by XXX".
**Conflicts of Interest**: Declare conflicts of interest or state "The authors declare no conflict of interest."
**ORCID iD (if any)**
**Publisher's Note**: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers.

**References**

[1] M. Parikh, A. A. Soni, S. M. Shah, and A. R. Jha, "Big Data Workload Profiling for Energy-Aware Cloud Resource Management," Jan. 2026, doi: 10.48550/arXiv.2601.11935.

[2] V. K. Sharma, "Cloud Computing IoT: 5G Focused IoT with Cloud Solutions," *Int. J. AI, BigData, Comput. Manag. Stud.*, vol. 6, no. 3, 2025, doi: 10.63282/3050-9416.IJAIBDCMS-V6I3P103.

[3] M. R. R. Deva, "DevOps and Continuous Delivery Adoption: Trends, Challenges, and Best Practices in Modern Software Development Life Cycle," *Int. J. Adv. Res. Comput. Sci.*, vol. 16, no. 4, pp. 118–124, Aug. 2025, doi: 10.26483/ijarcs.v16i4.7306.

[4] S. R. Sirikonda, "Reducing SRE Toil via Safe Autonomous Remediation in Cloud-Native Systems," *Am. J. Technol.*, vol. 5, no. 3, pp. 30–49, Mar. 2026, doi: 10.58425/ajt.v5i3.511.

[5] S. R. Chanthati, "Architecting Next-Gen Financial Systems with AI and Cloud-Native Microservice," in *Conference: 4th IEEE World Conference on Applied Intelligence and Computing (AIC-2025) IEEE Conference*, IEEE, 2025, p. july.

[6] G. Maddali, "An Efficient Bio-Inspired Optimization Framework for Scalable Task Scheduling in Cloud Computing Environments," *Int. J. Curr. Eng. Technol.*, vol. 15, no. 03, May 2025, doi: 10.14741/ijcet/v.15.3.4.

[7] A. Parupalli and H. Kali, "An In-Depth Review of Cost Optimization Tactics in Multi-Cloud Frameworks," *Int. J. Adv. Res. Sci. Commun. Technol.*, vol. 3, no. 5, pp. 1043–1052, Jun. 2023, doi: 10.48175/IJARSCT-11937Q.

[8] S. Jain and D. Jain, "Artifact Comparison Analyzer: Evaluating Microservice Build Metrics for Performance and Efficiency Improvements," in *2026 IEEE International Conference on AI Engineering and Innovations (AIEI)*, Jamshedpur, India: IEEE, 2026, pp. 1–6, March. doi: https://doi.org/10.1109/AIEI69164.2026.11497468.

[9] A. K. Padhy, C. Medicherla, B. Vulugundam, C. Kulkarni, T. P. Patel, and S. Shivam, "Latency-Optimized Microservices Orchestration for Real-Time E-Commerce in Multi-Cloud Environments," in *2025 International Conference on Computer and Applications (ICCA)*, IEEE, Dec. 2025, pp. 1–6. doi: 10.1109/ICCA66035.2025.11430930.

[10] T. P. Patel, "Adaptive Token Routing for Heterogeneous LLM Inference in Edge-Cloud Continuum," in *SoutheastCon 2026*, 2026, pp. 1–7. doi: 10.1109/SoutheastCon63549.2026.11476596.

[11] M. R. C. Mukkolakkal, "InfraLLM: A Generic Large Language Model Framework for Production-Grade Microservice Auto-Scaling in Cloud Infrastructure," *Int. J. Sci. Res. Mod. Technol.*, vol. 4, no. 11, pp. 113–123, 2025, doi: 10.38124/ijsrmt.v4i11.1023.

[12] H. P. Cyril and S. Kumara, "DevSecOps-Driven Security Integration in the Software Development Lifecycle Using CI/CD Pipelines," in *2026 IEEE 5th International Conference on AI in Cybersecurity (ICAIC)*, IEEE, Feb. 2026, pp. 1–6. doi: 10.1109/ICAIC67076.2026.11395737.

[13] R. Sinha, R. Gulati, A. Muttayane, K. Arunachalam, and J. Smith, "Network latency time measurement using DNS and web server messages," 11245606, 2022

[14] S. K. Chintagunta, "Survey of Containerization , Orchestration , and CI / CD Integration on DevOps in Modern Software Development," *Int. J. Curr. Eng. Technol.*, vol. 13, no. 6, pp. 610–618, 2023, doi: 10.14741/ijcet/v.13.6.14.

[15] S. K. Davuluri, V. Challagulla, V. Mudapaka, and U. Konka, "AI-Driven DevOps in Telecommunications: Bridging Predictive Analytics with Continuous Delivery for Network Agility," in *2025 IEEE International Conference and Expo on Real Time Communications at IIT (RTC)*, Oct. 2025, pp. 1–4. doi: 10.1109/RTC66985.2025.11211551.

[16] V. Ramamoorthi, "AI-Enhanced Performance Optimization for Microservice-Based Systems," *J. Adv. Comput. Syst.*, vol. 4, no. 9, pp. 1–7, 2024, doi: 10.69987/JACS.2024.40901.

[17] K. K. Mohammed, "The Future is Cloud : Modernizing Big Data for the Cloud Era," *Int. J. Sci. Res. Eng. Trends*, vol. 11, no. 5, pp. 1–5, 2025, doi: 10.5281/zenodo.17339856.

[18] J. B. Mehta, "Predictive Quality Engineering in Distributed Data Platforms Using Machine Learning," in *2026 IEEE International Systems Conference (SysCon)*, IEEE, Apr. 2026, pp. 1–6. doi: 10.1109/SysCon66367.2026.11503610.

[19] C. A. D. V. A Jahnavi A Kachhia, B Amit V Patel, "Design and Performance Analysis of Different Feeding Techniques With Micro-strip Patch Antenna," *Int. J. Adv. Technol. Eng. Res.*, vol. 5, no. 3, pp. 37–41, May, 2015.

[20] J. E. Kofi, "Data-Driven Cloud Workload Optimization Using Machine Learning Modeling for Proactive Resource Management," *Int. J. Artif. Intell. Data Sci. Mach. Learn.*, vol. 6, no. 4, pp. 27–37, 2025, doi: 10.63282/3050-922x.ijeret-v6i4p104.

[21] V. K. Bollu, "Threat Landscape in Artificial Intelligence Systems: Taxonomy, Attack Vectors and Security Implications,"

*World J. Adv. Res. Rev.*, vol. 29, no. 1, pp. 285–294, 2026, doi: 10.30574/wjarr.2026.29.1.0007.

[22] V. Methuku, S. Kamatala, P. Naayini, and P. R. Vontela, "From Ethical Principles to Technical Safeguards: A Unified Framework for Safe and Human-Centered Artificial Intelligence," *Am. Int. J. Comput. Sci. Technol.*, vol. 4, no. 5, pp. 26–34, Sep. 2022, doi: 10.63282/3117-5481/AIJCST-V4I5P103.

[23] S. Kilaru, "Automated ETL Intelligence: Metadata-Orchestrated Framework with Rule-Based Heuristics for Monitoring and Reporting," *Int. J. Inf. Electron. Eng.*, vol. 3, no. 6, p. 14, 2013.

[24] R. N. Rajendran, D. K. Rai, S. K. Anumula, and S. Agrawal, "Zero Trust Security Model Implementation in Microservices Architectures Using Identity Federation," in *2025 2nd International Conference on Recent Trends in Electrical, Electronics and Computing Technologies (ICRTEECT)*, 2025, pp. 1–6. doi: 10.1109/ICRTEECT67512.2025.11448625.

[25] S. Singamsetty, "An Intelligent Framework for Secure and Fair Cloud Resource Distribution," in *2025 7th International Conference on Innovative Data Communication Technologies and Application (ICIDCA)*, IEEE, Oct. 2025, pp. 686–690. doi: 10.1109/ICIDCA66325.2025.11280502.

[26] B. Krishnan, S. Perla, S. Maddela, and R. Lingam, "Adaptive Multi-Cloud Infrastructure for CRM Analytics: Real-Time ML and Data Sync with LLMs," in *2025 IEEE 3rd Global Conference on Wireless Computing and Networking (GCWCN)*, IEEE, Nov. 2025, pp. 1–8. doi: 10.1109/GCWCN66157.2025.11448404.

[27] M. A. Obied, F. F. M. Ghaleb, A. E. Hassanien, A. M. H. Abdelfattah, and W. Zakaria, "Deep Clustering-Based Anomaly Detection and Health Monitoring for Satellite Telemetry," *Big Data Cogn. Comput.*, vol. 7, no. 1, 2023, doi: 10.3390/bdcc7010039.

[28] A. Katangoori, "The Role of Big Data in Advancing Artificial Intelligence: Methods and Case Studies," *Int. J. Artif. Intell. Mach. Learn.*, vol. 6, no. 1, pp. 37–54, Jan. 2026, doi: 10.51483/IJAIML.6.1.2026.37-54.

[29] H. N. Dholariya, "Human-in-the-Loop AI for Cloud Data Engineering: The Collaborative Intelligence Architecture (CIRA) for Regulated Industries," *J. Inf. Syst. Eng. Manag.*, vol. 11, no. 1, pp. 883–898, Jan, 2026.

[30] S. O. Awodele *et al.*, "AI-Driven Root Cause Analysis Framework for Distributed Microservices Architectures," *Multidiscip. J. Eng. Technol. Sci.*, vol. 3, no. 1, pp. 8–17, 2026.

[31] M. K. Gaddam, "Architecting Observability for AI-Driven Microservices at Scale," in *2025 3rd International Conference on Intelligent Cyber Physical Systems and Internet of Things (ICoICI)*, IEEE, Sep. 2025, pp. 1830–1838. doi: 10.1109/ICoICI65217.2025.11252857.

[32] G. Dkmak, B. Can, O. Sevinc, C. B. Egeli, F. Baday, and B. Cetintav, "AI-Driven Anomaly Detection in Cloud-Native Microservices: The Night's Watch Algorithm," *Appl. Sci.*, vol. 15, no. 23, p. 12762, Dec. 2025, doi: 10.3390/app152312762.

[33] B. Barua and M. S. Kaiser, "Microservices-Based Framework for Predictive Analytics and Real-time Performance Enhancement in Travel Reservation Systems," 2024. doi: 10.48550/arXiv.2412.15616.

[34] D. K. Pentyala, "Artificial Intelligence for Fault Detection in Cloud-Optimized Data Engineering Systems," *Int. J. Soc. Trends*, vol. 2, no. 4, pp. 147–160, 2024.

[35] G. P. Menaud, "Machine Learning-Based Performance Prediction and Optimization in Microservices-Oriented Artificial Intelligence Systems," *Int. J. Mach. Intell.*, vol. 1, no. 5, pp. 1–6, 2023.

[36] S. A. Pushkala, "Financial Fraud Identification Using Graph Neural Network And LSTM With Autoencoder-Based Data Refinement," *J. Int. Cris. Risk Commun. Res.*, vol. 9, no. 1, 2026, doi: 10.63278/jicrcr.vi.3615.

[37] T. Ali, R. Iqbal, N. M. Ansari, T. Tariq, and A. A. Rafique, "Ai-Powered Anomaly Detection in Software Logs: a Machine Learning Approach for Proactive Fault Diagnosis and Self-Healing Systems," *Spectr. Eng. Sci.*, vol. 3, no. 3, pp. 302–322, 2025.

[38] R. Malaiyalan, "AI/ML-Driven Microservices Architecture for Scalable Cloud Computing Applications," *World J. Multidiscip. Stud.*, vol. 3, no. 3, pp. 28–35, 2026.

[39] H. Ge *et al.*, "SRdetector: Sequence Reconstruction Method for Microservice Anomaly Detection," *Electronics*, vol. 14, no. 1, p. 65, Dec. 2024, doi: 10.3390/electronics14010065.

[40] A. Gupta, "What Is The Right Security Posture? A Perspective on Cloud Computing Security Threats and Risk Assessment," *Int. J. Emerg. Res. Eng. Technol.*, vol. 4, no. 4, pp. 120–127, December, 2023, doi: 10.63282/3050-922X.IJERET-V4I4P112.