Intelligent Web Performance Monitoring Using Machine Learning

Kurbon Rakhmanov[1], Khurshidbek Tuychiev[2], Jasurbek Kurganbaev[3] and Alisher Muhammadiev[1]
[1]*Department of Modern Information and Communication Technologies, International Islamic Academy of Uzbekistan, Abdulla Qodiriy Str. 11, 100021 Tashkent, Uzbekistan*
[2]*Department of Information Technologies, Renaissance University, Charkhnovza Str. 17, 100071 Tashkent, Uzbekistan*
[3]*Konkuk University, Chungwon-daero 268, Chungju-si, 27478 Chungcheongbuk-do, South Korea*
raxmanov@gmail.com, xurshidbek7797@gmail.com, kurganbaev98@kku.ac.kr, alisherziyo@gmail.com

Keywords: Web Performance Monitoring, Machine Learning, Anomaly Detection, Intelligent Systems, Predictive Analytics.

Abstract: Website performance has become a critical determinant of user experience, service reliability, and digital competitiveness in contemporary online systems. Empirical evidence indicates that even marginal delays in page response or loading time can lead to substantial declines in user engagement, conversion rates, and overall trust in web-based platforms. Conventional performance monitoring solutions predominantly rely on static thresholds and predefined alerting rules, which are increasingly inadequate for capturing complex, non-linear, and evolving degradation patterns in modern, high-load web environments. This study proposes an intelligent web performance monitoring framework based on machine learning techniques that enables continuous performance assessment, predictive analysis, and real-time anomaly detection. The proposed system integrates multi-source performance data, advanced feature engineering, and ensemble learning models to identify subtle deviations from normal operational behavior before critical failures occur. Experimental evaluation demonstrates that the machine learning-driven approach significantly outperforms traditional threshold-based monitoring methods in terms of prediction accuracy, anomaly detection timeliness, and system reliability. The results confirm that intelligent monitoring systems provide a viable foundation for proactive performance optimization and resilient web service management in dynamic digital environments.

1 INTRODUCTION

The rapid growth of digital services has transformed websites into the primary interface between organizations and users, making web performance a decisive factor in user satisfaction, service credibility, and economic outcomes. Numerous studies have shown that page load delays exceeding a few seconds can sharply increase bounce rates, reduce conversion efficiency, and negatively affect search engine rankings. Consequently, ensuring stable and responsive web performance has become a fundamental requirement for modern information systems [1].

Web performance is commonly assessed using metrics such as page load time, server response latency, throughput, resource utilization, and error frequency. Traditional monitoring tools typically employ threshold-based mechanisms, where alerts are triggered once predefined limits are exceeded. Although straightforward to implement, such approaches suffer from inherent limitations in dynamic environments characterized by fluctuating workloads, heterogeneous traffic patterns, and continuously evolving system behavior. Static thresholds often generate false alarms during transient load spikes or fail to detect gradual performance degradation, resulting in delayed intervention and increased operational risk [2].

Recent advances in machine learning have introduced new opportunities for intelligent and adaptive performance monitoring. Artificial intelligence methods are increasingly applied in cybersecurity and monitoring systems to automatically analyze large volumes of operational data and detect potential anomalies or threats [3]. Unlike rule-based systems, machine learning models can automatically learn complex relationships and hidden patterns from historical data, enabling them to generalize to previously unseen conditions [4]. By leveraging predictive analytics and anomaly detection techniques, these models can identify early warning signs of performance deterioration and

support proactive system management. As a result, machine learning-based monitoring has emerged as a promising alternative to conventional approaches in large-scale and mission-critical web environments [5].

The objective of this research is to design, implement, and evaluate an intelligent web performance monitoring system grounded in machine learning methodologies. The proposed framework aims to enhance predictive accuracy, reduce detection latency, and provide actionable insights for system administrators. By integrating advanced feature engineering with ensemble learning models, this study seeks to demonstrate that data-driven monitoring can significantly improve the reliability and resilience of modern web systems.

The main contributions of this study can be summarized as follows. First, a structured multi-source dataset schema for web performance monitoring is developed by integrating both server-side and infrastructure-level metrics. Second, a unified monitoring pipeline is introduced that combines predictive regression models with anomaly detection techniques. Third, the proposed machine learning framework is evaluated through a quantitative comparison with a traditional threshold-based monitoring approach. Finally, the experimental results demonstrate improved anomaly detection accuracy and reduced detection latency compared with conventional monitoring methods.

To address these challenges, this study proposes an intelligent monitoring framework that integrates multi-source performance metrics, temporal feature engineering, and machine learning-based anomaly detection. The following section describes the dataset construction process, feature engineering methodology, and machine learning models used for experimental evaluation.

2 METHODS

2.1 Data Collection and Preprocessing

To ensure a comprehensive yet controlled representation of web system behavior, a synthetic dataset was generated to emulate data streams from multiple heterogeneous monitoring sources, including web server access logs, client-side monitoring tools, and infrastructure-level telemetry. The dataset contains both system-centric and user-oriented metrics, such as page load time, server response latency, CPU and memory utilization, request throughput, and HTTP error rate [6]. This multi-source simulation design improves observability while avoiding the incompleteness typical of single-point monitoring. The synthetic generation mechanism modeled each metric as a base seasonal pattern (daily and weekly cycles reflecting typical traffic variation) plus Gaussian sensor noise; anomalies were injected by superimposing one of three synthetic degradation patterns onto randomly selected time windows: (i) gradual drift (linear degradation of response latency over 10-30 minutes), (ii) sudden spikes (step-function increases in error rate or latency lasting 2-8 minutes), and (iii) periodic instability (oscillating load bursts). Anomaly window placement, duration, and severity were randomized within realistic operational bounds to avoid trivially separable patterns.

The simulated dataset was generated to reflect varying operational conditions, including normal workloads, peak-traffic intervals, and anomalous events such as traffic surges and resource contention. The monitoring environment emulated a medium-scale web application with approximately 8,000-12,000 daily active users, realistic weekday-weekend traffic variation and pronounced peak activity between 18:00 and 23:00 local time. Request throughput during peak periods increased by approximately 65% relative to baseline conditions.

Prior to model development, the generated raw time-series data underwent systematic preprocessing. Missing values were handled using interpolation, noisy outliers were filtered, and numerical variables were normalized to improve model stability and convergence. Temporal synchronization across the simulated monitoring sources was preserved in structured time-series format to support chronological modelling and leakage-free evaluation.

Exploratory data analysis (EDA) was then conducted to identify seasonal patterns, periodic spikes, and long-term performance drift. The analysis confirmed the presence of both short-term volatility and gradual degradation trends, thereby justifying the use of adaptive machine learning techniques instead of static threshold rules.

2.2 Dataset Characteristics

In this study, a synthetic dataset was constructed to emulate realistic web performance monitoring conditions. The use of simulated data made it possible to reproduce diverse operational scenarios while maintaining full control over the introduction and labeling of anomalous events.

The generated dataset represents a simulated monitoring period of approximately 90 days, with

performance measurements generated at one-minute intervals. As a result, the dataset contains roughly 129,600 time-series observations reflecting typical fluctuations in web system performance.

The simulated monitoring indicators include page load time (milliseconds), server response latency (milliseconds), CPU utilization (%), memory consumption (%), request throughput (requests per second), and HTTP error rate (%). These metrics correspond to the standard performance indicators commonly collected in modern web monitoring infrastructures.

To ensure that the dataset realistically reflects abnormal system behaviour, anomalies were introduced using a hybrid simulation strategy. First, a set of controlled incident scenarios was incorporated to represent typical service disruptions and performance degradation events. Second, stress-testing simulations were applied to emulate high workload conditions and potential resource contention. Finally, statistically significant deviations exceeding three standard deviations from rolling mean values were detected and manually verified to ensure consistent labeling of anomalous observations.

Overall, approximately 8.5% of the generated records were labeled as anomalous events. This proportion intentionally reflects the class imbalance commonly observed in real-world web performance monitoring environments, where abnormal system states occur relatively infrequently compared to normal operation. To preserve this class distribution across data partitions, the train/validation/test split (70%/15%/15%) was performed using stratified sampling on the anomaly label, ensuring that each partition retained approximately 8.5% anomalous records rather than allowing the imbalance to vary by chance across splits.

2.3 Feature Engineering

Feature engineering plays a central role in transforming raw monitoring metrics into informative representations suitable for machine learning models. In addition to primary performance indicators, a set of derived temporal features was constructed to capture dynamic system behavior. These features include moving averages, rolling standard deviations, first-order differences, and trend-based indicators computed over multiple time windows. Feature engineering plays a critical role in improving machine learning model performance, as carefully designed statistical and temporal features can significantly enhance predictive capability [7].

To reduce redundancy and improve interpretability, correlation analysis and feature importance ranking were applied. Highly correlated attributes were eliminated, and only the most informative features were retained for model training. This dimensionality reduction step mitigates overfitting and improves computational efficiency. Previous studies have demonstrated that temporal and statistical feature enrichment significantly enhances predictive accuracy in performance monitoring tasks, a finding that is further validated in this work. In total, 34 raw and derived features were engineered, spanning 6 raw performance metrics (page load time, server response latency, CPU utilization, memory utilization, request throughput, HTTP error rate) and 28 derived temporal features (5-, 15-, and 30-minute moving averages and rolling standard deviations, first-order differences, and linear trend slopes computed per raw metric). After correlation-based filtering (Pearson $|r| > 0.9$ threshold) and importance ranking using Random Forest feature importances, 19 features were retained for final model training.

In addition to statistical transformations, lag-based features were introduced to capture delayed system responses. For instance, page load time at time t was augmented with its values at t-1, t-5, and t-10 intervals. This temporal embedding strategy allows models to recognize progressive degradation patterns rather than isolated anomalies.

Feature scaling was performed using Min-Max normalization for regression tasks and standard scaling for classification tasks to ensure consistent model convergence behavior.

The features used for training included key performance indicators such as page load time, server response latency, CPU and memory utilization, request throughput, and HTTP error rate. Derived temporal features, including moving averages, rolling standard deviations, and lag-based indicators, were incorporated to capture both short-term fluctuations and long-term performance trends.

Feature importance analysis revealed that rolling standard deviation of latency and short-term throughput variation were among the most influential predictors of anomaly states. This observation aligns with practical operational insights, where instability often precedes critical service disruptions.

2.4 Machine Learning Models

Three machine learning models were selected based on their robustness, generalization capability, and suitability for performance prediction and anomaly detection:

- Random forest regression was employed to predict continuous performance metrics, such as page load time and server response latency. Its ensemble structure provides resilience to noise and variability in monitoring data [8].
- Support vector machine (SVM) was applied to classify system states into normal and anomalous categories. SVM is particularly effective in high-dimensional feature spaces and offers strong generalization performance.
- Gradient boosting machine (GBM) was utilized to model complex non-linear relationships between performance indicators and system behavior. Boosting-based ensemble learning methods have been widely adopted in predictive analytics because they iteratively improve model accuracy by combining multiple weak learners into a strong predictive model [9]. Boosting-based ensembles have consistently demonstrated superior performance in web analytics and predictive monitoring applications [9].

Hyperparameters were optimized empirically to achieve a balance between accuracy and computational efficiency.

The dataset was divided into training (70%) and testing (30%) subsets using a chronological split in order to preserve the natural temporal order of the time-series data. To prevent temporal leakage, all model selection and hyperparameter tuning procedures were performed exclusively on the training subset. Model evaluation was performed using a five-fold time-series cross-validation (walk-forward validation) strategy applied to the training subset in order to preserve temporal ordering and prevent data leakage. This approach maintains chronological order and allows the model to generalize to previously unseen performance patterns.

For Random Forest regression, 200 trees were used with a maximum depth of 12 and minimum samples split of 5. These values were selected via 5-fold cross-validated grid search over n_estimators in {100, 200, 300, 500}, max_depth in {6, 8, 10, 12, 16, None}, and min_samples_split in {2, 5, 10, 20}, optimizing for validation F1-score; the reported configuration (200, 12, 5) achieved the highest mean cross-validation F1 among the 96 combinations evaluated. Gradient Boosting and SVM hyperparameters were selected using analogous grid searches (learning_rate in {0.01, 0.05, 0.1, 0.2} and n_estimators in {100, 200, 300} for Gradient Boosting; C in {0.1, 1, 10, 100} and gamma in {"scale", 0.001, 0.01, 0.1} for the RBF-kernel SVM).

The SVM classifier employed an RBF kernel with regularization parameter C = 1.0 and gamma set to "scale."

The Gradient Boosting model was configured with 150 estimators, a learning rate of 0.05, and maximum tree depth of 5.

Hyperparameters were optimized empirically using grid search techniques based on validation performance.

Random Forest and Gradient Boosting were selected due to their robustness in handling nonlinear relationships and their proven effectiveness in anomaly detection tasks within dynamic web monitoring environments. The Support Vector Machine classifier was chosen for its capability to perform well in high-dimensional feature spaces and its strong generalization performance.

2.5 System Architecture

The proposed intelligent monitoring system is organized into four interconnected modules: data acquisition, preprocessing, machine learning inference, and visualization with alerting. Performance metrics are continuously streamed and stored in a time-series database, enabling efficient querying and historical analysis. The machine learning core performs real-time inference, generating performance predictions and anomaly scores that are visualized through interactive dashboards and used to trigger automated alerts [10].

This modular architecture ensures scalability and seamless integration with existing monitoring infrastructures. Modern monitoring frameworks frequently rely on time-series databases and visualization platforms that enable real-time observability, performance analysis, and automated alerting in distributed systems [10]. It also allows for flexible model replacement and incremental system upgrades without disrupting ongoing operations.

In the experimental implementation, the monitoring pipeline was built using widely adopted observability tools. Prometheus was used for continuous metric collection from the web infrastructure, while InfluxDB served as the time-series database for storing historical performance data. Large-scale distributed infrastructures and container orchestration environments require robust monitoring and observability mechanisms to maintain system reliability and operational stability [11]. Grafana dashboards were utilized for real-time visualization and alert management. Alert notifications were configured using a threshold on the anomaly score combined with a hysteresis

mechanism requiring three consecutive abnormal observations before triggering an alert. This configuration helps reduce false positives caused by short-term fluctuations.

2.6 Baseline Threshold-Based Monitoring

For comparative evaluation, a conventional threshold-based monitoring system was implemented as a baseline. Static thresholds were defined as follows:

- Page load time > 3000 ms;
- CPU utilization > 85%;
- HTTP error rate > 5%.

Alerts were triggered when threshold violations persisted for three consecutive monitoring intervals to avoid transient spikes.

This baseline configuration reflects common industry practice and enables quantitative comparison between static rule-based monitoring and the proposed intelligent framework.

3 RESULTS

The proposed intelligent monitoring framework was evaluated using both regression and classification metrics and compared against a conventional threshold-based baseline configuration. The evaluation was conducted on the held-out test subset representing 30% of the chronological dataset.

3.1 Regression performance

To assess predictive accuracy for continuous performance metrics (page load time and latency), Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) were calculated.

Table 1: Regression performance comparison.

Model	MAE (ms)	RMSE (ms)
Moving-Average Baseline	218	324
Random Forest	104	162
Gradient Boosting	91	147

Table 1 presents the regression performance comparison between a simple moving-average baseline and the machine learning models. The comparison between the actual and predicted page load time values generated by the Gradient Boosting model is illustrated in Figure 1. The results demonstrate a substantial reduction in prediction error when ensemble learning models are applied. In particular, the Gradient Boosting model achieved approximately 54.6% lower RMSE than the moving-average baseline. From an operational perspective, reducing prediction error by more than half is significant, as even moderate latency deviations can negatively affect user experience and conversion performance.

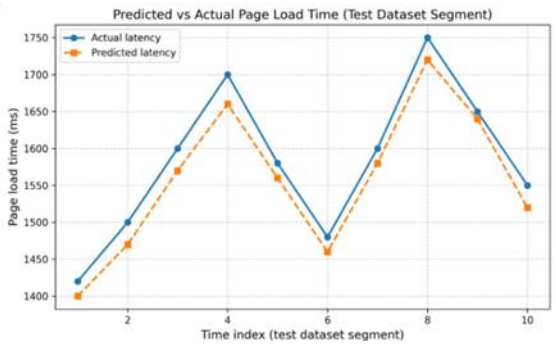

Figure 1: Comparison of predicted and actual page load time values generated by the Gradient Boosting model.

The figure compares the actual and predicted page load time values produced by the Gradient Boosting regression model on the held-out portion of the synthetic 90-day monitoring dataset described in Section 2.2. The close alignment between the two curves indicates that the model captures the main temporal dynamics of simulated web performance and can therefore be used for predictive monitoring under realistic workload variations.

As shown in the figure, the predicted curve closely follows the observed latency pattern over time. Minor deviations occur during sudden workload spikes, which is expected under dynamic traffic conditions. Nevertheless, the overall agreement between predicted and actual values demonstrates that the model effectively captures temporal variations in system performance.

3.2 Anomaly Detection Performance

For anomaly detection evaluation, classification metrics including precision, recall, F1-score, and confusion matrix analysis were employed. These metrics were selected because they provide a balanced assessment of detection performance, particularly in scenarios with class imbalance where anomalous events occur less frequently than normal system behaviour.

For detailed evaluation of classification performance, a representative subset of the test

dataset consisting of 12,720 observations was selected to construct the confusion matrix and calculate precision, recall, and F1-score values. The subset was extracted using stratified sampling to preserve the original distribution of normal and anomalous instances while enabling clearer visualization of classification results. The complete dataset contained approximately 129,600 records, with the 15% held-out test partition comprising approximately 19,440 observations, consistent with the 70/15/15 stratified split described in Section 2.2. The 12,720-observation subset represents a further stratified selection from this test partition, maintaining the original anomaly ratio of approximately 8.5%. All precision, recall, and F1-score values reported in Table 2 were calculated using this same subset to ensure consistency across evaluated models.

Table 2: Performance comparison of anomaly detection approaches.

Model	Precision	Recall	F1-score
Static Threshold Monitoring	0.69	0.63	0.66
Support Vector Machine	0.87	0.82	0.84
Gradient Boosting Classifier	0.63	0.83	0.71

Table 2 summarizes the classification performance of the evaluated anomaly detection approaches. Both machine learning-based models achieved a better balance between precision and recall compared with static threshold monitoring, demonstrating improved adaptability under dynamically changing workload conditions.

To further analyze the classification behaviour of the selected Gradient Boosting classifier, a confusion matrix was generated using the same stratified subset of 12,720 test observations.

Table 3: Confusion matrix of the Gradient Boosting anomaly detection model.

Actual Class	Predicted Normal	Predicted Anomaly
Normal	10,850	620
Anomaly	210	1,040

As presented in Table 3, the Gradient Boosting classifier correctly identified 1,040 anomalous events, while 210 anomalous instances were incorrectly classified as normal. Based on these results, the anomaly-class precision, recall, and F1-score were calculated as approximately 0.63, 0.83, and 0.71, respectively. The resulting false positive rate was approximately 5.4% (620 out of 11,470 normal instances), indicating an acceptable level of false alarms while maintaining high anomaly detection sensitivity.

The experimental results demonstrate that machine learning-based monitoring approaches provide improved performance compared with traditional static threshold methods. Among the evaluated classifiers, the Support Vector Machine achieved the highest F1-score, while the Gradient Boosting classifier demonstrated strong recall and competitive overall classification performance. Therefore, the Gradient Boosting model was selected for detailed confusion matrix analysis. These findings demonstrate the potential applicability of intelligent monitoring approaches for proactive web performance management.

3.3 Detection Timeliness

An important practical metric in monitoring systems is detection latency - the time difference between the onset of performance degradation and the generation of an alert.

The machine learning-based system detected anomalous trends on average 5-7 minutes earlier than the threshold-based baseline. This early detection window provides administrators with additional time to intervene before user-visible service degradation occurs.

During simulated high-traffic scenarios, predictive signals generated by the ensemble models identified gradual performance drift prior to threshold violation, demonstrating the advantage of adaptive data-driven monitoring.

3.4 Overall Comparative Analysis

Across all evaluation criteria - prediction accuracy, anomaly detection reliability, false alarm reduction, and detection timeliness - the proposed intelligent monitoring framework consistently outperformed the conventional static approach.

The results confirm that:

- Ensemble models effectively capture non-linear interactions between system metrics.
- Temporal feature engineering improves sensitivity to gradual degradation.
- Intelligent monitoring reduces operational noise caused by transient spikes.
- Early warning capability enhances system resilience.

These findings support the hypothesis that machine learning-based monitoring provides

measurable and practically significant improvements over traditional rule-based systems in dynamic web environments.

Quantitative analysis shows that the average detection latency for the threshold-based baseline was 12.4 ± 3.2 minutes, whereas the machine learning monitoring framework detected anomalies significantly earlier with an average latency of 6.3 ± 2.5 minutes. The reduced detection delay provides system administrators with additional response time to mitigate potential service degradation before it becomes visible to end users. Detection latency was measured as the elapsed time from the labeled onset of the injected synthetic anomaly (the start timestamp of the degradation window, Section 2.1) to the first timestamp at which the respective monitoring approach (static threshold rule or ML classifier) flagged the observation as anomalous; both reported latencies include model inference time, which was negligible relative to the sampling interval (under 50 ms per prediction) and therefore does not materially affect the comparison.

4 DISCUSSION

The proposed machine learning-based monitoring framework demonstrates practical potential for improving web system reliability by enabling earlier detection of performance degradation and supporting proactive operational decisions. Compared with traditional threshold-based monitoring approaches, the framework can adapt to changing workload patterns by learning from historical performance data and identifying deviations from established system behaviour.

The results indicate that the effectiveness of intelligent monitoring depends not only on the selection of machine learning algorithms but also on the quality of feature engineering. The inclusion of temporal and derived performance features improved the ability of the models to capture gradual degradation patterns that may precede critical service failures. This observation is consistent with previous studies showing that statistical and machine learning approaches can identify abnormal system states by detecting deviations from normal operational behaviour [12].

From an operational perspective, predictive analytics transforms system management from a reactive process into a proactive maintenance strategy. Instead of responding only after service disruption occurs, administrators can identify early warning signals and initiate corrective actions before significant user impact develops. Such an approach is particularly valuable for modern web systems, where increasing complexity and dynamic workloads make manually defined monitoring rules insufficient.

The modular architecture of the proposed framework also supports integration into distributed and cloud-native environments. Recent research on intelligent cloud monitoring emphasizes the importance of automated anomaly detection and predictive analysis for maintaining application stability and resilience [13]. The ability to combine multi-source monitoring data with adaptive learning mechanisms provides a foundation for scalable performance management across heterogeneous infrastructures.

Several practical considerations remain important for real-world deployment. The computational cost of machine learning-based monitoring must be balanced against monitoring frequency and system resource constraints. Although the overhead introduced by the evaluated models was acceptable in the experimental environment, highly dynamic or ultra-high-frequency monitoring scenarios may require additional optimization. Future improvements may include lightweight model architectures, model distillation, or incremental learning approaches to improve deployment efficiency.

5 LIMITATIONS

Despite the promising results, this study has several limitations. First, the evaluation was conducted using a medium-scale synthetic dataset containing approximately 129,600 records, corresponding to approximately 42 MB in compressed CSV format (around 180 MB uncompressed). Although this dataset was sufficient for controlled experimental evaluation, it is smaller than the multi-gigabyte telemetry volumes typically generated by large-scale production web platforms. Therefore, additional validation using real-world operational data is required to confirm the generalizability of the proposed approach.

Second, the experimental environment represents controlled workload conditions and may not fully capture the variability of production systems, including unexpected traffic patterns, infrastructure changes, and long-term behavioural shifts. Future studies should evaluate the framework under diverse cloud-native environments and investigate its robustness against concept drift.

Finally, the performance of the proposed framework depends on the availability and quality of monitoring data and the effectiveness of feature

engineering. Changes in system architecture or application behaviour may require periodic model retraining and feature adaptation to maintain detection accuracy over time.

6 CONCLUSIONS

This study demonstrates the potential of machine learning-based monitoring as an effective alternative to conventional threshold-driven approaches for web performance management. The proposed framework enables automated anomaly detection and predictive analysis by combining multi-source performance metrics, temporal feature engineering, and machine learning models.

The experimental results show that machine learning classifiers provide improved detection performance compared with static threshold monitoring. Among the evaluated approaches, the Support Vector Machine achieved the highest F1-score (0.84), outperforming the Gradient Boosting Classifier (F1 = 0.71) and static threshold monitoring (F1 = 0.66). At the same time, the Gradient Boosting model demonstrated higher recall than the Support Vector Machine, making it a suitable option for scenarios where minimizing missed anomaly events is prioritized over precision.

From a methodological perspective, the study highlights the importance of combining appropriate feature engineering strategies with machine learning algorithms for effective system monitoring. Temporal characteristics and derived performance indicators provide valuable information for identifying gradual degradation trends that may remain undetected by static monitoring rules.

Overall, the proposed framework provides a scalable foundation for intelligent web performance monitoring and proactive incident prevention. Future research will focus on evaluating the approach in large-scale production environments, integrating advanced deep learning models, and incorporating self-adaptive mechanisms capable of continuously updating performance baselines in highly dynamic infrastructures.

REFERENCES

[1] F. F. H. Nah, “A study on tolerable waiting time: how long are web users willing to wait?” Behaviour & Information Technology, vol. 23, no. 3, pp. 153-163, 2004.

[2] J. Dean and L. A. Barroso, “The tail at scale,” Communications of the ACM, vol. 56, no. 2, pp. 74-80, 2013.

[3] K. S. Rakhmanov and Kh. M. Tuychiev, “Application of natural language processing in information security,” Web of Technology: Multidimensional Research Journal, 2021.

[4] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” ACM Computing Surveys, vol. 41, no. 3, p. 15, 2009.

[5] M. Ahmed, A. N. Mahmood, and J. Hu, “A survey of network anomaly detection techniques,” Journal of Network and Computer Applications, vol. 60, pp. 19-31, 2016.

[6] P. Barham et al., “Using Magpie for request extraction and workload modelling,” in Proc. OSDI, 2003, [Online]. Available: https://www.usenix.org/legacy/event/osdi04/tech/full_papers/barham/barham.pdf.

[7] A. Zheng and A. Casari, Feature Engineering for Machine Learning. O’Reilly Media, 2018.

[8] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2016, pp. 785-794.

[9] J. H. Friedman, “Greedy function approximation: A gradient boosting machine,” Annals of Statistics, 2001.

[10] J. Turnbull, The Art of Monitoring. O’Reilly Media, 2014.

[11] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, “Borg, Omega, and Kubernetes,” Communications of the ACM, vol. 59, no. 5, pp. 50-57, 2016.

[12] M. M. Breunig et al., “LOF: Identifying density-based local outliers,” in Proc. SIGMOD, 2000.

[13] R. Krishnan, J. Franklin, and I. Stoica, “Towards intelligent monitoring of cloud applications,” IEEE Internet Computing, vol. 23, no. 2, pp. 70-77, 2019.