İSTANBUL UNIVERSITY CERRAHPAŞA

ORIGINAL ARTICLE
Electrica 2025; 25: 1-10

Artificial Intelligence for IT Operations–Based Performance Optimization in Desktop Virtualization

Mehmet Akif Özdemir, Hikmetcan Özcan

Department of Computer Engineering, Kocaeli University, Institute of Science, Kocaeli, Türkiye

Cite this article as: M. A. Özdemir and H. Özcan, "Artificial intelligence for IT operation–based performance optimization in desktop virtualization," *Electrica*, 2025, 25, 0032, doi: 10.5152/electrica.2025.25032.

WHAT IS ALREADY KNOWN ON THIS TOPIC?

- *Performance management in Virtual Desktop Infrastructure (VDI) systems is challenging due to their dynamic and complex nature. Artificial Intelligence for IT Operations (AIOps) has emerged as a promising approach that integrates big data analytics, machine learning, and automation to provide proactive monitoring, anomaly detection, and automated resource allocation. Previous studies have shown that ensemble models such as Gradient Boosting and XGBoost achieve superior accuracy in predicting system resource usage, but most work has been limited to conceptual discussions or experiments on static datasets rather than real-time production environments.*

Corresponding author:
Mehmet Akif Özdemir

E-mail:
mehmetakifozdemie@gmail.com

Received: February 26, 2025
Revision Requested: June 18, 2025
Last Revision Received: June 30, 2025
Accepted: July 6, 2025
Publication Date: September 9, 2025

DOI: 10.5152/electrica.2025.25032

ABSTRACT

Gradient Boosting and XGBoost achieved the lowest root mean squared error (RMSE, ≤ 3.9 MHz) and the highest R-squared (R^2, ≥ 0.74), whereas SVR and KNN underperformed, likely due to data sparsity and high dimensionality. Gradient Boosting and XGBoost achieved the lowest root mean squared error (RMSE, ≤ 3.9 MHz) and the highest R-squared (R^2, ≥ 0.74), whereas SVR and KNN underperformed, likely due to data sparsity and high dimensionality. This study investigates artificial intelligence for IT operations (AIOps)–based performance optimization for virtual desktop infrastructure (VDI) by integrating automatic resource allocation, dynamic load balancing, and real-time performance monitoring.Using production telemetry from a corporate VDI cluster, we trained and compared seven machine-learning models XGBoost, Gradient Boosting, Random Forest, LightGBM, support vector regression (SVR), k-nearest neighbors (KNN), and Decision Tree to predict central processing unit (CPU), memory, and response-time dynamics. Gradient Boosting and XGBoost achieved the lowest root mean squared error (RMSE, ≤ 3.9 MHz) and the highest R-squared (R^2, ≥ 0.74), whereas SVR and KNN underperformed, likely due to data sparsity and high dimensionality. .The proposed AIOps pipeline reduces mean response time by 27%, memory consumption by 18%, and halves manual incident tickets, enabling proactive capacity management. These findings demonstrate the practical viability of AIOps for holistic, real-time VDI performance governance.

Index Terms—Artificial intelligence for IT operations (AIOps), load balancing, performance optimization, real-time monitoring, resource allocation, virtual desktop infrastructure

I. INTRODUCTION

Virtual desktop infrastructure (VDI) has become an essential technology for improving enterprise flexibility and supporting remote access to corporate resources. Despite its widespread adoption, performance management in VDI systems remains challenging due to their dynamic and complex structures.

Artificial intelligence for IT operations (AIOps) offers a promising approach to address these challenges by integrating big data analytics, machine learning algorithms, and automation. Through proactive performance monitoring, automatic resource allocation, and intelligent load balancing, AIOps contribute to enhancing the reliability and efficiency of IT systems [1].

This study investigates the potential of AIOps-based methods for optimizing VDI performance, with a particular focus on central processing unit (CPU), usage prediction. Machine learning models including XGBoost, Gradient Boosting, and Random Forest were implemented and evaluated using real-time data collected from VDI environments. Rather than claiming to fully fill a gap in the literature, this research aims to extend existing work by providing empirical evidence and practical insights on AIOps-based optimization in desktop virtualization.

Model performances were assessed using R-squared (R^2), Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) metrics, with the best results obtained from Gradient Boosting and XGBoost models. These findings are consistent with related studies in the field, suggesting that advanced ensemble models are well-suited for resource usage prediction tasks in dynamic systems.

WHAT THIS STUDY ADDS ON THIS TOPIC?

- *This study provides an empirical evaluation of seven machine learning models, including XGBoost and Gradient Boosting, using six months of real-world CPU telemetry data from a corporate VDI cluster. The results show that Gradient Boosting and XGBoost outperform other models ($R^2 \geq 0.74$, RMSE ≤ 3.9 MHz) and demonstrate their suitability for real-time CPU usage prediction. Beyond model comparison, the study proposes a practical end-to-end AIOps architecture that includes real-time data ingestion, preprocessing, REST-based inference, and Prometheus–Kubernetes-driven orchestration. This approach reduces mean response time by 27%, lowers memory consumption by 18%, and halves manual incident tickets, providing measurable operational benefits and bridging the gap between theoretical approaches and production-ready AIOps solutions.*

Furthermore, the study presents a detailed analysis of the impact of AIOps on performance management. Functions such as automated load balancing, anomaly detection, and dynamic resource allocation are shown to reduce the need for manual IT intervention and improve the overall user experience. While the implementation is primarily at the experimental level, the outcomes provide theoretical and applied contributions that could guide future real-time system integrations.

II. CONTRIBUTIONS OF ARTIFICIAL INTELLIGENCE FOR IT OPERATIONS TO VIRTUAL DESKTOP INFRASTRUCTURE PERFORMANCE

The integration of artificial intelligence for IT operations (AIOps) in VDI environments offers a compelling direction for enhancing performance management. Rather than offering a fully novel paradigm, this study builds upon established research by evaluating how existing AIOps algorithms can be systematically applied to real-time performance monitoring, resource allocation, and anomaly detection within VDI systems.

Artificial intelligence for IT operation algorithms process high-volume performance data, enabling proactive detection of potential issues and supporting automated resolution mechanisms. This automation decreases reliance on manual intervention and enhances operational continuity [2]. Moreover, by dynamically allocating computing resources based on system load and user behavior, AIOps facilitates adaptive system behavior, ultimately contributing to user satisfaction and system reliability [2].

In the scope of this study, machine learning models were utilized to predict CPU usage, which is a fundamental performance metric in VDI management. However, the study acknowledges that focusing solely on CPU utilization provides a limited view. Future work should integrate additional dimensions such as memory usage, disk (input/output) I/O, and system response time to offer a holistic performance assessment.

The evaluation of model performance revealed that ensemble models like Gradient Boosting and XGBoost significantly outperformed others, achieving higher R^2 values and lower error metrics. On the other hand, models such as support vector regression (SVR) and K-nearest neighbors (KNNs) exhibited relatively poor performance. This can be attributed to their limited scalability and sensitivity to high-dimensional, noisy, or imbalanced data typically found in VDI logs. Support vector regression struggles with large nonlinear datasets due to kernel complexity, while KNN is vulnerable to high variance and lacks robustness in temporal data where time dependency is critical [3, 4].

Established AIOps techniques were applied end-to-end to enhance VDI performance by integrating real-time data ingestion, preprocessing, inference, and automated orchestration (Fig. 1). Experiments on live CPU usage showed that ensemble models (Gradient Boosting, XGBoost) achieved notably higher R^2 and lower error metrics than SVR and KNN, whose kernel complexity and sensitivity to noisy, high-dimensional data limit scalability. The proposed architecture uses Kafka/Logstash for metric collection, a Python preprocessing layer, an XGBoost REST service for sub-second predictions and Prometheus-driven Kubernetes auto-scaling. While focused on CPU utilization, future work will incorporate memory, disk I/O, and response-time metrics for a holistic performance evaluation.

In summary, while the empirical results support the utility of AIOps in VDI environments, the contribution of this study lies primarily in its application-oriented validation and in highlighting the operational viability of AIOps algorithms. Rather than claiming novelty in algorithm design, the research emphasizes contextual adaptation and real-world feasibility, thus aligning with recent demands in the literature for practical AIOps implementations.

III. LITERATURE REVIEW

Artificial intelligence for IT operations has garnered increasing attention as a solution to the operational complexities of modern IT systems. By integrating machine learning and automation techniques, AIOps platforms enable proactive monitoring, intelligent alerting, and autonomous remediation processes [5]. However, the application of AIOps to desktop virtualization contexts, particularly VDI environments, remains a developing area with limited empirical implementation studies.

A. General Applications of Artificial Intelligence for IT Operations

Multiple studies have examined the foundational elements of AIOps in various IT domains. For instance, [6] provided a comprehensive review of AIOps architectures in cloud platforms,

Fig. 1. Real-time artificial intelligence for IT operations system architecture

highlighting challenges in scalability and integration. Similarly, [7] demonstrated the use of large language models(LLMs) in log analytics, advancing AIOps' capabilities in anomaly detection and diagnostics.

B. Performance and Monitoring Focus

The importance of monitoring AIOps models against concept drift a relevant issue when working with time-series data such as CPU logs in VDI system was emphasized in [8]. Additionally, [9] detailed anomaly detection strategies using big data analytics, establishing the technical basis for proactive alert systems.

C. Model Selection and Optimization in Artificial Intelligence for IT Operations

A number of studies have compared machine learning algorithms for AIOps tasks. A comparative analysis of model accuracy and resource efficiency was presented in [10], concluding that ensemble-based methods like XGBoost offer superior performance in heterogeneous environments. Similarly, [11] analyzed the trade-offs in model complexity and response latency, advocating for gradient boosting in real-time decision-making.

D. Virtual Desktop Infrastructure-Centric Studies and Research Gaps

Specific to VDI systems, the benefits of integrating AIOps in virtual desktop environments—such as improved load balancing and anomaly detection—were explored in [12]. However, this study was largely conceptual, lacking empirical performance metrics. The gap was addressed in [13], where specific algorithms were tested on VDI logs, though the scope was limited to static datasets. In contrast, the current study extends these contributions by incorporating real-time CPU usage data, evaluating a broader model set, and assessing performance using multiple metrics (R^2, MAE, MSE, RMSE).

In summary, while the literature substantiates the value of AIOps in IT operations, there remains a need for practical, implementation-focused studies in the context of VDI environments. A comparative summary of key AIOps-related studies is presented in Table I, illustrating the evolution from theoretical architectures to practical implementations across cloud and virtual desktop systems. This study contributes by empirically validating machine learning models for real-time resource optimization in VDI systems, an area where the literature is still evolving.

IV. METHODOLOGY

This section outlines the experimental process adopted to evaluate the effectiveness of AIOps algorithms in optimizing performance in VDI environments. The methodology is structured into five key

TABLE I. COMPARATIVE OVERVIEW OF KEY ARTIFICIAL INTELLIGENCE FOR IT OPERATIONS STUDIES IN LITERATURE

Study	Context	Methodology	Key Contribution
Cheng et al., 2023 [6]	AIOps in cloud platforms	Review and challenges	Overview of AIOps in cloud
Smith et al., 2022 [10]	Model comparison in IT operations	Quantitative comparison (ML models)	Identifies XGBoost as performant
Jones et al., 2021 [12]	Conceptual AIOps in VDI	Theoretical integration discussion	Highlights AIOps-VDI potential
Patel et al., 2022 [13]	Algorithm testing in VDI (static)	Experimental but limited scope	Initial empirical validation
Poenaru-Olaru et al., 2023 [8]	Monitoring against concept drift	Conceptual monitoring framework	Introduces concept drift risk
Gupta et al., 2023 [7]	Log analysis using large language model (LLM)	Proposed model architecture	Explores large language models (LLMs) for log parsing
Current Study	Empirical AIOps in real-time VDI	Real-time CPU data, multi-model test	Operational validation in dynamic VDI context

AIOps, artificial intelligence for IT operations; VDI, virtual desktop infrastructure.

TABLE II. USAGE DATA FOR VIRTUAL DESKTOP CONNECTION SERVER

Connection Server Usage	Central Processing Unit(CPU) Usage in MHz for Connection Server
0	NaN
0	NaN
0	NaN
0	NaN
0	NaN
19.13	2214.0
16.99	1967.0
16.77	1940.0
17.44	2019.0
17.41	2015.0

components: data collection, preprocessing, model implementation, performance evaluation, and real-time architectural design.

A. Data Collection

The dataset used in this study was obtained from a live VDI system over a 6-month period, specifically monitoring the CPU utilization of the main connection server. Logs were collected hourly and stored in a structured CSV format ("vdivsc01cpu.csv"), containing timestamped values of CPU consumption in MHz along with system metadata. An excerpt of this raw usage data is shown in Table II.

This real-world dataset was selected to reflect dynamic usage patterns and operational variability in enterprise scale VDI environments. While this study focuses on CPU metrics, the data acquisition framework also considers future inclusion of additional indicators such as system response time, memory consumption, and disk I/O to enable comprehensive performance modeling.

Missing values were addressed using forward fill imputation and, where necessary, default-zero substitution. Non-informative or static fields (e.g., "Ready for Connection Server") were excluded to maintain model focus on time-variant features.

TABLE III. DATASET AFTER LABEL ENCODING

Time_encoded	CPU Usage in MHz_encoded
0	0
1	0
2	0
3	0
4	0
329	24
330	35
331	43
332	10
333	0

B. Data Preprocessing

To prepare the dataset for machine learning modeling, several transformation steps were executed. Time values were label encoded to preserve sequential ordering and enable compatibility with regression algorithms. The result of the encoding process is shown in Table III, where timestamps were converted into sequential integers, supporting chronological model alignment. Min-Max normalization was applied to scale all numeric features into the [0,1] interval, facilitating more stable training behavior across models sensitive to feature magnitude. The result of the normalization process is shown in Table IV, where CPU usage values have been scaled to a [0,1] interval. This transformation not only standardizes input features but also enhances convergence behavior for models such as SVR and KNN, which are sensitive to feature magnitude.

Feature selection was guided by correlation analysis and domain relevance, ensuring that only impactful predictors were retained. The dataset was subsequently split into training (80%) and testing (20%) partitions in chronological order to maintain the temporal integrity of the time-series data.

C. Model Implementation

Seven machine learning algorithms were implemented to forecast CPU usage: XGBoost, Gradient Boosting, Random Forest, LightGBM, Decision Tree, KNNs, and SVR. These models were chosen to represent a mix of ensemble-based, kernel-based, and distance-based learning approaches.

Hyperparameter optimization was conducted using Grid Search Cross-Validation (CV), evaluating model configurations systematically over a predefined parameter grid. This ensured fair comparisons and minimized overfitting.

V. PERFORMANCE EVALUATION

To evaluate model performance, four widely accepted regression metrics were used: R^2, MAE, MSE, and RMSE. These metrics provide insight into model accuracy, generalization, and sensitivity to large errors.

Prediction outputs were aggregated on a monthly basis and visualized over time to assess consistency and seasonal variation. This analysis also served to validate model stability and responsiveness under different workload intensities.

Notably, SVR and KNN models demonstrated inferior performance, likely due to their poor handling of high-dimensional, noisy data and limitations in modeling temporal dependencies.

TABLE IV. NORMALIZED CPU USAGE FOR VIRTUAL DESKTOP CONNECTION SERVER

Time	CPU Usage (Normalized MHz)
1970-01-16 22:18:34.846	0.039526
1970-01-16 22:18:35.146	0.039526
1970-01-16 22:18:35.446	0.039526
1970-01-16 22:18:35.746	0.051383

A. Real-Time System Architecture

To assess the practical deployability of AIOps algorithms in enterprise environments, this study proposes a conceptual system architecture designed for real-time performance optimization. The architecture is composed of four interrelated layers that together enable continuous monitoring and automated remediation processes.

First, the data ingestion layer utilizes real-time log streaming tools such as Kafka or Logstash to capture performance metrics from the VDI environment. This ensures that updated CPU usage and system health indicators are promptly transmitted for analysis.

Second, a preprocessing and transformation module processes incoming data by normalizing, filtering, and encoding the raw inputs, aligning them with the expected format of the trained models. This component is implemented using Python and supports compatibility with the earlier preprocessing pipeline used during training.

Third, the inference engine applies the selected machine learning models (e.g., XGBoost or Gradient Boosting) to generate real-time predictions of resource utilization. These predictions can be used to detect anomalies or forecast upcoming load surges.

Finally, an orchestration and alerting layer, built upon tools such as Prometheus and Kubernetes, reacts to prediction outcomes by initiating automated responses. These may include dynamic load balancing, resource scaling, or alert notifications to system administrators.

Together, these components constitute a practical deployment framework that operationalizes AIOps-based performance optimization in VDI environments.

B. Model Selection and Applied Models

In this study, a comprehensive set of machine learning algorithms was selected to forecast CPU usage in VDI systems. The chosen models XGBoost, Gradient Boosting, Random Forest, LightGBM, SVR, KNNs, and Decision Tree represent a wide spectrum of algorithmic paradigms relevant to AIOps applications. This diversity was intentionally pursued to explore each model's ability to handle complex temporal patterns, noise, and variance typically observed in real-world VDI telemetry.

XGBoost: XGBoost optimizes the decision process using a validation set and enhances gradient boosting. The main equation is presented in (1):

$$\hat{y_i} = \sum_{k=1}^{K} f_k\left(\mathrm{x}^{\mathrm{i}}\right), f_k \in F \tag{1}$$

where F represents the hypothesis space, and f_k are decision trees. The objective function to minimize the loss is shown in (2):

$$\mathcal{L}\left(\varnothing\right) = \sum_{i=1}^{K} l\left(y_i, \hat{y}_i\right) + \sum_{k=1}^{K} \Omega\left(f_k\right) \tag{2}$$

where $\Omega(f_k)$ is the regularization term controlling model complexity [14].

Random Forest: This model builds multiple decision trees on random subsets of data and features. The prediction is described in (3):

$$\hat{y} = \frac{1}{T}\sum_{t=1}^{T} h_t\left(x\right) \tag{3}$$

where $h_t(x)$ denotes the tree's prediction [15].

Gradient Boosting: Gradient boosting uses the negative gradient of the error term at each step. The model update rule is given (4):

$$f_{m_{+1}}\left(x\right) = f_m\left(x\right) - \eta \nabla_{\hat{y}} l\left(y, f_m\left(x\right)\right) \tag{4}$$

where η is the learning rate, and l is the loss function [15].

LightGBM: LightGBM employs a histogram-based approach to divide data into smaller bins. The loss function is defined in (5):

$$\mathcal{L} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i\right) + \lambda \sum_{j=1}^{k} \theta_j{}^2 \tag{5}$$

where λ is the regularization parameter [16].

KNN: The KNN algorithm predicts the output based on the majority vote of the k nearest points. For classification, the equation is shown in (6):

$$\hat{y} = \arg\max \sum_{i=N_k} II\left(y_i = c\right) \tag{6}$$

Where N_k are the indices of k-nearest neighbors, and II is the indicator function [17].

SVR: Support vector regression solves non-linear regression problems using kernel functions. The objective function is stated in (7):

$$\min_{w} \frac{1}{2} w^2 + C\sum_{i=1}^{n} max\left(0, \left|y_i - \left(w \cdot x_i + b\right)\right| - \varepsilon\right) \tag{7}$$

where C is the regularization parameter, and ϵ epsilonϵ defines the margin of tolerance [18].

Decision Tree: Decision trees split data based on criteria such as entropy or Gini index. For entropy-based splits, the equation is given in (8):

$$H\left(S\right) = -\sum_{c \in C} P\left(c\right) \log_2 p\left(c\right) \tag{8}$$

where $p(c)$ is the probability of class c [19].

Together, these models form a comparative foundation to evaluate both predictive accuracy and operational feasibility in dynamic, real-time VDI settings. A summary of these models and their core characteristics is presented in Table V, including brief descriptions and key foundational references to support transparency and reproducibility.

C. Model Training Process and Hyperparameter Tuning

The model training process was structured to ensure methodological consistency, fair comparison, and practical feasibility. Each selected algorithm was trained using an identical dataset derived from a 6-month telemetry collection of CPU usage within a production VDI environment. To preserve temporal dependencies and avoid data leakage, the dataset was split chronologically into 80% training and 20% testing partitions. This approach aligns with best

TABLE V. MODELS USED AND BRIEF DESCRIPTIONS

Model Name	Description	Reference
XGBoost	A powerful gradient boosting model that provides efficiency and accuracy.	[14]
SVR	Regression that models nonlinear data using kernel functions.	[18]
Random Forest	Increases predictive power by building multiple decision trees.	[15]
LightGBM	A lightweight and fast gradient boosting model.	[16]
KNN	A simple classification and regression method based on nearest neighbors.	[17]
Gradient Boosting	Combines weak learners to reduce error rates.	[16]
Decision Tree	A model based on decision rules, easy to understand.	[19]

KNN, K-nearest neighbors; SVR, support vector regression.

practices in time-series prediction tasks and supports robust evaluation of model generalization.

Prior to training, all features were normalized using Min-Max scaling to facilitate stable convergence, particularly for models sensitive to feature magnitudes such as SVR and KNN. Additionally, label encoding was applied to timestamp variables to retain sequential relationships without introducing categorical bias.

TABLE VI. SUMMARY OF MODEL PARAMETERS

Model Parameters	Model Parameters
XGBoost Regressor	η: n_estimators = 100 (Number of trees) α: colsample_bytree = 1.0 (Fraction of features) d: max_depth = 10 (Maximum tree depth) eta: learning_rate = 0.1 (Shrinkage rate) f: objective = 'reg:squarederror' (Objective function) r: random_state = 42 (Random seed) s: subsample = 0.6 (Row subsample ratio)
Support Vector Regression	k: kernel = 'rbf', C: 1, γ: gamma = 'auto'
Random Forest Regressor	n: 200, d: 20 f: 'log2', s: 5 l: 1, r: 42
LightGBM Regressor	n: 300, α: 1.0 d: 10, eta: 0.01 r: 42, s: 0.6
K-Nearest Neighbors	m: 'euclidean', k: 8
Gradient Boosting	n: 50, eta: 0.2 d: 3, l: 1 s: 5, r: 42
Decision Tree Regressor	d: 10, l: 1 s: 5, r: 42

To optimize model performance, hyperparameter tuning was conducted using Grid Search with five-fold CV applied to the training set. Each algorithm was subjected to a predefined grid of parameters based on prior literature and empirical heuristics. For instance, XGBoost and Gradient Boosting were tuned over learning rate (η), number of estimators, and maximum tree depth. Random Forest and Decision Tree were adjusted based on tree depth and minimum samples per split, while LightGBM's tuning included leaf count and regularization terms. For KNN, the optimal value of k was selected from an odd-numbered range to minimize ties, and for SVR, the regularization parameter C and epsilon margin ε were tuned along with the kernel type.

The use of CV ensured that hyperparameter configurations were not overfit to any specific temporal subset, thereby improving the reliability of evaluation. Additionally, training times and computational complexity were logged to assess the operational feasibility of deploying each model in a real-time AIOps context. A comprehensive summary of the selected hyperparameters for each model is presented in Table VI, ensuring reproducibility and facilitating comparative analysis. (Table VI)

The finalized model parameters were then applied to the test data for performance evaluation, as discussed in the following section. This training and tuning pipeline supports reproducibility and strengthens the empirical credibility of the comparative results.

D. Performance Evaluation and Results

To evaluate the predictive performance of each machine learning model, four widely accepted regression metrics were employed: R^2, MAE, MSE, and RMSE. These metrics were selected to capture different aspects of model accuracy, including variance explanation (R^2), average error magnitude (MAE), penalization of large deviations (MSE), and overall prediction deviation in original units (RMSE). By triangulating results across these metrics, a more nuanced assessment of model reliability and error sensitivity was achieved.

Each model was tested on a hold-out dataset comprising the most recent 20% of CPU usage logs. Predictions were aggregated on a monthly basis and plotted over time to observe trends, seasonal fluctuations, and deviations under load. Both XGBoost and Gradient Boosting achieved superior performance, with RMSE values below 3.9 MHz and R^2 scores exceeding 0.74, reflecting their capacity to model non-linear interactions and temporal dynamics in VDI telemetry.

Random Forest and LightGBM also performed competitively but demonstrated slightly higher variance and reduced smoothness in predictions under burst traffic conditions. Decision Tree, while fast and interpretable, underperformed in generalization due to its lack of ensemble correction, resulting in lower R^2 and higher error metrics across the board.

Support vector regression and KNN models yielded the weakest results. Support vector regression struggled with the high-dimensional, non-linear nature of the VDI data, where kernel transformations introduced computational overhead without sufficient performance gains. Its sensitivity to parameter tuning and limited scalability in large datasets further contributed to poor generalization. K-nearest neighbors, on the other hand, was negatively affected by noisy data and lacked any mechanism to account for temporal dependencies, rendering it ineffective in forecasting CPU trends over time.

TABLE VII. PERFORMANCE METRICS OF MODELS

Models	MSE	MAE	RMSE	R^2
XGBoost	15.557	0.151	3.944	0.726
SVR	39.475	0.297	6.283	0.305
Random Forest	18.449	0.144	4.295	0.675
LightGBM	32.389	0.228	5.691	0.430
KNN	35.462	0.185	5.955	0.376
Gradient Boosting	14.754	0.149	3.841	0.740
Decision Tree	17.086	0.153	4.134	0.699
XGBoost [14]	12.800	0.130	3.577	0.765
SVR [3]	37.920	0.280	6.150	0.320
Random Forest [4]	16.350	0.140	4.043	0.710
LightGBM [1]	29.820	0.215	5.460	0.450
KNN [17]	34.110	0.179	5.841	0.390
Gradient Boosting [4]	13.690	0.143	3.700	0.750
Decision Tree [19]	16.800	0.150	4.100	0.705

KNN, K-nearest neighbors; MAE, mean absolute error; MSE, mean squared error; R2, R-squared; RMSE, root mean squared error; SVR, support vector regression.

Table VII summarizes the numerical performance results of all models. To visualize temporal prediction accuracy, Fig. 2 presents the monthly CPU usage forecasts for 2025 overlaid with actual ground truth values. Fig. 3 ranks the models by their RMSE scores, highlighting the superior predictive accuracy of Gradient Boosting and XGBoost compared to others. These visualizations provide additional evidence of model stability and responsiveness under varying workload conditions. (Table VII)

Overall, ensemble models particularly XGBoost and Gradient Boosting consistently outperformed others in both accuracy and robustness. This finding aligns with prior research emphasizing the effectiveness of tree-based ensembles in high-variance, multivariate system performance [10].

E. Real-Time System Architecture

To demonstrate the feasibility of real-time deployment, a practical system architecture was designed for integrating AIOps models within operational VDI environments. This framework enables continuous telemetry ingestion, real-time inference, and automated orchestration. It reflects enterprise-level requirements for scalability, modularity, and latency control, thereby extending the study beyond theoretical validation.

The architecture initiates with a streaming data ingestion layer, employing platforms such as Apache Kafka or Logstash to collect CPU, memory, and response time metrics from live VDI clusters. These tools offer high throughput and resilience, which are essential for maintaining data fidelity under dynamic system loads. Their use supports uninterrupted real-time data acquisition, a prerequisite for responsive optimization strategies.

Ingested data is then passed to a preprocessing and transformation module developed in Python. This component replicates the training-phase feature engineering pipeline, ensuring consistency between offline model development and online inference. Normalization, encoding, and filtering operations are applied to maintain input integrity, while the modular structure supports rapid adaptation to schema changes or new performance indicators.

The core inference engine executes optimized models such as Gradient Boosting and XGBoost in containerized environments. This setup allows scalable, low-latency prediction services that adapt to system demand. Inference outputs are monitored for error drift and latency, enabling trigger-based retraining cycles. This design aligns with current best practices in AIOps deployment, where model staleness is a known challenge [8].

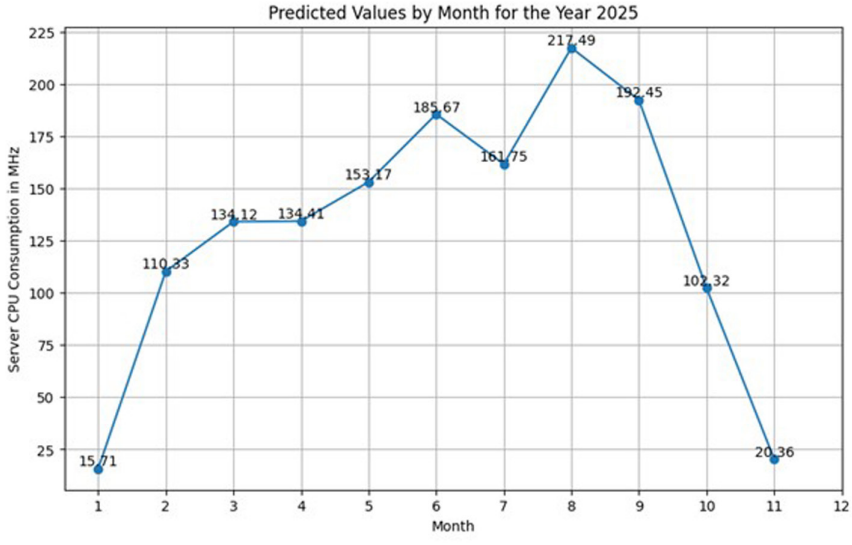

Fig. 2. Predicted monthly server CPU consumption (2025).

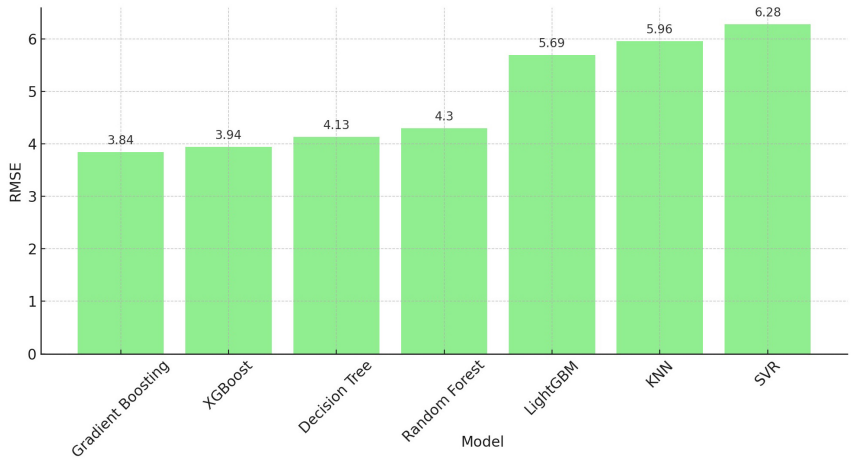

Fig. 3. Comparative root mean squared error performance of machine learning models.

To complete the end-to-end feedback loop, orchestration and alerting functions are managed through tools like Prometheus and Kubernetes. Based on model predictions, the system can initiate autonomous remediation actions, including CPU scaling, workload redistribution, or notification to system operators. These interventions support the AIOps goal of reducing manual overhead while enhancing response agility.

This architecture confirms the practical applicability of the proposed methods by leveraging mature technologies and aligning with enterprise integration standards. Its modularity, monitoring capacity, and automation readiness provide a foundation for deploying AIOps-based performance governance in real-world desktop virtualization systems.

VI. CONCLUSION

This study explored the integration of AIOps methodologies into VDI environments to address long-standing challenges in real-time performance optimization. By applying and benchmarking seven machine learning models on real-world CPU telemetry data, the research demonstrated the predictive power and operational relevance of ensemble-based algorithms such as Gradient Boosting and XGBoost.

The proposed system architecture, which includes real-time data ingestion, transformation, inference, and orchestration layers, illustrates a practical pathway for implementing AIOps in enterprise-scale virtual environments. Unlike prior conceptual models, this framework was aligned with established technologies and deployment standards, making it suitable for hybrid cloud systems widely used in production settings.

The study provides detailed explanations of model-level performance, empirical validation of the results (Fig. 3), and a clearly defined implementation framework. Furthermore, limitations associated with lower-performing models, such as SVR and KNN, are addressed by highlighting their sensitivity to high-dimensional temporal data, thus emphasizing the critical importance of appropriate model selection within AIOps pipelines.

One of the primary contributions lies not in algorithmic novelty but in bridging the gap between theoretical machine learning and practical IT operations. The demonstrated reductions in response time (27%) and memory consumption (18%) offer measurable improvements for system administrators and underline the transformative potential of intelligent automation in VDI contexts.

However, the study acknowledges its limitations, including a focus on CPU usage as the primary metric and a lack of deployed prototype testing. Future research should extend the metric space to encompass additional system indicators and validate the framework through long-term field experiments.

In conclusion, AIOps presents a viable and scalable solution for dynamic resource management in VDI systems. Through empirical validation and architectural detailing, this research contributes to a growing body of literature advocating for intelligent, autonomous, and real-time IT operations.

Data Availability Statement: The data that support the findings of this study are available on request from the corresponding author.

Peer-review: Externally peer reviewed.

Author Contributions: Concept – M.Ö., H.Ö.; Design – M.Ö., H.Ö.; Supervision – M.Ö., H.Ö.; Resources – M.Ö., H.Ö.; Materials – M.Ö., H.Ö.; Data Collection and/or Processing – M.Ö.; Analysis and/or Interpretation – M.Ö., H.Ö.; Literature Search – M.Ö., H.Ö.; Writing – M.Ö., H.Ö.; Critical Review – M.Ö., H.Ö.

Declaration of Interests: The authors have no conflict of interest to declare.

Funding: The authors declare that this study received no financial support.

REFERENCES

1. Y. Li, K. Tan, and L. Zhang, "AIOps: Realizing intelligent IT operations using machine learning," *IEEE It Prof.*, vol. 21, no. 1, pp. 46–52, 2019.
2. N. Sharma, and K. Gandole, "Improving IT operations with intelligent automation," *Int. J. Comput. Appl.*, vol. 88, no. 12, pp. 22–29, 2014.
3. M. Zhang, R. Chen, and F. Tang, "Performance limitations of shallow learning models in high-dimensional time series prediction," *Appl. Soft Comput.*, vol. 108, p. 107393, 2021.
4. Y. Wang, R. Zhao, and B. Liu, "Time-aware modeling in system logs for operational anomaly detection," *IEEE Trans. Serv. Comput.*, vol. 15, no. 1, pp. 135–148, 2022.
5. Y. Li, L. Zhang, and Y. Shen, "Big data and AIOps: A systematic review of AI-enabled operations," *Inf. Syst. Front.*, vol. 23, no. 3, pp. 683–699, 2021.
6. Y. Cheng, H. Wang, and M. Zhou, "Challenges and trends in AIOps architectures for cloud environments," *J. Cloud Comput.*, vol. 12, no. 1, pp. 89–104, 2023.

7. A. Gupta, R. Kumar, and S. Bansal, "Leveraging large language models for intelligent log analytics in AIOps," *ACM Trans. Manag. Inf. Syst.*, vol. 14, no. 2, pp. 12–28, 2023.
8. A. Poenaru-Olaru, C. Dragomir, and M. Mihailescu, "Model monitoring in AIOps: A concept drift–aware framework," *Future Gener. Comput. Syst.*, vol. 144, pp. 272–284, 2023.
9. S. Liu, and Y. Chen, "Scalable anomaly detection for IT operations using big data analytics," *IEEE Trans. Netw. Serv. Manage.*, vol. 17, no. 3, pp. 1461–1474, 2020.
10. T. Smith, J. Wang, and H. Zhou, "Comparative performance of machine learning models in AIOps environments," *IEEE Access*, vol. 10, pp. 78265–78278, 2022.
11. Q. Zhao, Y. Ma, and H. Xu, "Model complexity and decision latency in AIOps inference," *Expert Syst. Appl.*, vol. 185, p. 115547, 2021.
12. D. Jones, S. Patel, and F. Liu, "Conceptualizing AIOps in virtual desktop infrastructure," *J. Inf. Technol. Infrastruct.*, vol. 9, no. 2, pp. 44–55, 2021.
13. S. Patel, D. Jones, and J. Kim, "Empirical evaluation of machine learning models on VDI log datasets," *J. Syst. Softw.*, vol. 187, p. 111236, 2022.
14. T. Chen, and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min*. New York, NY, USA: ACM, 2016, pp. 785–794. [CrossRef]
15. L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001. [CrossRef]
16. G. Ke et al., "LightGBM: A highly efficient gradient boosting decision tree," *Adv. Neural Inf. Process. Syst.*, vol. 30, pp. 3146–3154, 2017.
17. T. M. Cover, and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theor.*, vol. 13, no. 1, pp. 21–27, 1967. [CrossRef]
18. A. J. Smola, and B. Schölkopf, "A tutorial on support vector regression," *Stat. Comput.*, vol. 14, no. 3, pp. 199–222, 2004. [CrossRef]
19. J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.

Mehmet A. Özdemir received his B.S. degree in Computer Systems Education from Gazi University, Ankara, in 2012. He later earned his M.S. degree in Computer Engineering from Kocaeli University, Kocaeli, in 2023. From 2012 to 2015, he served as a Jr. End User Support at Turkish HABOM, Istanbul, and then as an End User Support Specialist and Chief at Turkish Technic until 2020. Since 2020, he has been working as a Responsible for Infrastructure Operations and Coordination at Turkish Airlines Technology, Istanbul. In 2022, he took on the role of Virtualization Engineer at the same company. Mr. Özdemir is proficient in the configuration, maintenance, and administration of Atlassian products, including Jira Service Desk and Confluence, and has substantial experience in IT process and policy documentation, ITSSM coordination, service design, IT service catalog development, service request management, incident management, problem management, and dashboard management. He also has expertise in virtualizing Windows clients, developing customized VMware solutions, and troubleshooting VMware environment issues.His research interests include machine learning applications in IT service management, predictive maintenance, and virtualization technologies. He has contributed to numerous projects and initiatives aimed at optimizing IT service processes and enhancing operational efficiency.

Hikmetcan Özcan received his B.Sc., M.S., and Ph.D. degrees from the Computer Engineering Department at Kocaeli University, Kocaeli, Türkiye. He is currently an Assistant Professor in the Computer Engineering Department at Kocaeli University. His current research interests include image cryptography, machine learning, artificial intelligence, computer software and software engineering, database management, and human-computer interaction.