International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Machine Learning-Based Predictive Fault Detection and Resilience Optimization in Large-Scale Distributed Systems

Ishu Anand Jaiswal

Submitted: 01/11/2020 **Revised**: 18/12/2020 **Accepted**: 25/12/2020

Abstract: The rapid expansion of cloud computing, high-performance computing, large-scale data centers, distributed applications, and service-oriented computing infrastructures has significantly increased the complexity of modern distributed systems. Such environments consist of large numbers of interconnected computing nodes, virtual machines, storage devices, network resources, and software services that operate collaboratively to deliver continuous and scalable computing services. However, the increasing scale, heterogeneity, workload variability, and dynamic behavior of distributed infrastructures also increase their exposure to hardware faults, software failures, node crashes, resource exhaustion, communication disruptions, and performance degradation. Traditional fault-tolerance mechanisms generally react to failures after they occur and therefore may result in service interruption, resource wastage, recovery overhead, and violations of service-level requirements.

Keywords: *Machine Learning, Predictive Fault Detection, Distributed Systems, Fault Tolerance, Resilience Optimization.*

Introduction

Large-scale distributed computing systems have become a fundamental component of modern information technology infrastructure. Cloud computing platforms, high-performance computing clusters, distributed databases, large-scale storage systems, enterprise applications, and Internet-based services increasingly depend on thousands of interconnected computational resources to provide scalable and continuously available services. Unlike traditional centralized computing environments, distributed systems divide applications, workloads, and data among multiple interconnected computing nodes. This architecture offers substantial advantages in terms of scalability, resource sharing, parallel processing, elasticity, and performance. At the same time, increasing system size and complexity create substantial challenges related to reliability, availability, fault tolerance, and operational resilience. Large-scale distributed systems consist of heterogeneous computing resources including processors, servers, virtual machines, network interfaces, storage devices, software services, and application components. These resources operate under continuously changing workloads and environmental conditions. Consequently, failures can originate from numerous sources, including hardware degradation, software defects, memory exhaustion, disk faults, network congestion, node crashes, configuration errors, resource contention, workload imbalance, and unexpected application behavior. A failure occurring within one component may further propagate to dependent components and ultimately affect the availability of an entire distributed application.

The probability of encountering failures becomes increasingly significant as the number of computing components grows. Even when individual components possess relatively high reliability, a distributed infrastructure containing thousands of nodes may experience hardware or software failures regularly. Large-scale cloud infrastructures built from commodity computing resources are particularly susceptible to such failures. Chen et al. (2014), using Google cluster traces, demonstrated the importance of predicting application failures from resource-usage information so that proactive actions can be initiated before unsuccessful jobs consume unnecessary computational resources. Conventional fault-management systems predominantly employ reactive approaches. In a reactive fault-tolerance architecture, the system identifies a problem only after a node, application, or hardware component has already failed. Recovery procedures such as restarting applications, rescheduling failed tasks, activating replicas, restoring checkpoints, or migrating workloads are subsequently performed to restore system functionality. Although these techniques are essential for distributed-system dependability, they cannot completely prevent service interruption or

Apple Inc., Cupertino, California, USA, ijaiswal@apple.com

resource loss because corrective action begins after the failure has occurred.

Predictive fault management provides an alternative approach in which system behavior is continuously monitored to identify warning patterns associated with future failures. Instead of waiting for a component to fail, predictive mechanisms analyze historical and real-time operational information to estimate the likelihood of future faults. When failure probability exceeds an acceptable threshold, the system can perform proactive fault-tolerance actions before the predicted failure produces significant consequences. Machine learning has emerged as an effective technology for implementing predictive fault detection because modern distributed systems continuously generate large amounts of operational data. Monitoring infrastructures collect information such as processor utilization, memory consumption, disk activity, network throughput, input/output rates, task execution behavior, application logs, error events, response times, resource requests, node health indicators, and historical failure records. These data contain complex relationships that conventional threshold-based monitoring techniques may not identify effectively.

Machine learning algorithms can learn relationships between system behavior and previous failure events and subsequently classify current system conditions as healthy, degraded, or failure-prone. Different machine learning architectures provide complementary capabilities for distributed-system fault analysis. Decision Trees and Random Forest models can identify nonlinear relationships among operational parameters while providing relatively interpretable decision mechanisms. Support Vector Machines can construct effective decision boundaries for fault classification in high-dimensional monitoring data. Artificial Neural Networks can learn complex nonlinear relationships between system metrics and failure conditions, while recurrent neural architectures can analyze temporal dependencies within sequences of resource-usage measurements. The importance of temporal information in predictive fault detection has been demonstrated in previous research. Failures frequently develop gradually through changing resource-utilization patterns rather than occurring without warning. Consequently, historical observations of CPU usage, memory utilization, task behavior, storage activity, and network performance can provide valuable indicators of impending failure. Lin et al. (2018), for example, proposed MING for predicting node failures in cloud-service systems by integrating temporal and spatial information. Their framework incorporated Long Short-Term Memory learning for temporal signals, Random Forest-based processing for spatial information, ranking mechanisms, and cost-sensitive decision strategies.

Failure prediction is particularly challenging in large-scale distributed environments because failures are normally much less frequent than successful operations. This produces highly imbalanced datasets in which normal system observations substantially outnumber fault observations. A prediction system trained directly on such data may achieve apparently high overall accuracy while failing to identify critical fault events. Therefore, predictive fault-detection models must be evaluated using measures such as precision, recall, F1-score, false-positive rate, and true-positive rate rather than relying exclusively on overall classification accuracy. Another major challenge is the propagation of faults among interconnected system components. Distributed services contain complex dependencies where the behavior of one service, server, or infrastructure component may influence several other components. Pitakrat et al. (2018) demonstrated this limitation of purely monolithic failure predictors and proposed an architecture-aware approach that combines component-level predictions with knowledge of system dependencies. Such findings indicate that effective predictive resilience requires not only identification of individual component faults but also an understanding of how predicted failures may influence the overall distributed architecture.

Literature Review

Chen et al. (2014) investigated failure behavior in large-scale cloud computing environments using workload traces collected from a Google computing cluster. The study analyzed job and task failures and examined their relationships with scheduling constraints, resource utilization, node operations, and user characteristics. The results demonstrated that resource-usage patterns contain useful information for identifying abnormal execution behavior and suggested that early failure prediction could support proactive maintenance and reduce unnecessary job resubmissions. This work established an important foundation for data-driven reliability analysis in large-scale cloud and distributed computing systems. Chen et al. (2014) subsequently developed a predictive approach for identifying job failures in compute clouds using Google cluster traces. Instead of relying entirely on static failure rules, the authors analyzed resource-usage measurements as time-series data and applied recurrent neural-network-based techniques for predicting application failures. Experimental observations indicated that application failures could be identified before actual termination, providing sufficient lead time for proactive system actions. The study also reported potential resource savings of approximately 6–10% through early failure identification. These findings demonstrated

the practical value of connecting failure prediction with proactive resource management.

Soualhia et al. (2015) examined scheduling failures in cloud computing environments using traces obtained from Google clusters and a Hadoop implementation deployed on Amazon Elastic MapReduce. Statistical prediction models were developed using task execution and resource-utilization information to determine whether scheduled tasks were likely to fail. The study reported prediction precision of up to 97.4% and recall of up to 96.2%. Simulation experiments further indicated that early rescheduling based on predictions could increase the number of successfully completed tasks, while the Hadoop case study demonstrated that predictive scheduling could reduce job failures. The work showed that fault prediction can provide greater value when prediction results are directly incorporated into scheduling and recovery decisions. Yoo et al. (2016) investigated machine-learning-based job-status prediction in scientific computing clusters. The researchers analyzed job characteristics and runtime performance measurements to identify unsuccessful job executions before completion. A Random Forest classifier was used to discover patterns associated with failed computational jobs. Experimental results showed that unsuccessful jobs could be identified with high predictive performance, including 83.6% recall and 94.8% precision. The study demonstrated that operational measurements from scientific clusters can serve as valuable inputs for online failure-prediction mechanisms and can potentially initiate mitigation procedures before computational resources are wasted.

Zheng et al. (2016) proposed EHMM-CT, an online failure-prediction method for cloud computing systems based on Hidden Markov Models and Cloud Theory. The approach analyzed multiple runtime system indicators and their correlations with failures instead of requiring detailed knowledge of underlying failure mechanisms. The authors extended conventional HMM-based prediction through multidimensional failure representation and Cloud Theory-based likelihood and membership calculations. Simulation results indicated that the method could provide accurate failure prediction while maintaining relatively low computational overhead. This work emphasized the importance of balancing prediction effectiveness with runtime cost when predictive methods are deployed continuously in large-scale systems. Islam and Manivannan (2017) proposed a machine-learning approach for predicting application failures in cloud environments using Google cluster workload traces. Their study first characterized resource consumption associated with successful, failed, and killed jobs and subsequently developed an LSTM-based prediction model using job and task performance measurements. The proposed approach achieved approximately 87% task-failure prediction accuracy, with a true-positive rate of 85% and a false-positive rate of approximately 11%. The study demonstrated the capability of recurrent neural architectures to capture temporal resource behavior associated with application failures and showed how predictive information could help reduce resource wastage caused by unsuccessful cloud workloads.

Pitakrat et al. (2018) presented Hora, an architecture-aware online failure-prediction technique for complex software systems. Unlike monolithic prediction approaches that treat the complete system as a single entity, Hora incorporates architectural dependencies among system components. The technique monitors system events and quality-of-service information while considering how component failures can propagate through interconnected software architectures. Experimental evaluation across different failure scenarios demonstrated the usefulness of architectural knowledge for improving online prediction. The research highlighted an important challenge for large-scale distributed systems: predicting an individual component failure is insufficient when dependencies can cause cascading effects across other system components. Lin et al. (2018) proposed MING for predicting node failures in production cloud-service systems. The model combines an LSTM architecture for temporal signals with a Random Forest model for spatial information. A ranking mechanism combines outputs from these models to determine node failure-proneness, while a cost-sensitive strategy addresses the severe class imbalance normally associated with operational failure data. The approach was evaluated using real-world cloud-service data and was also applied in an industrial environment. An important practical advantage of MING is that predicted faulty nodes can be identified before complete failure, allowing virtual machines to be allocated or migrated toward healthier computing nodes.

Mariani et al. (2020) introduced PreMiSE, a lightweight failure-prediction and fault-localization technique for multi-tier distributed systems. The approach integrates anomaly-based and signature-based detection mechanisms to predict failures that affect system performance while simultaneously identifying components responsible for those failures. Experimental evaluation on a cloud-based IP Multimedia Subsystem demonstrated high precision, low false-positive behavior, and low computational overhead. The study is particularly relevant to large-scale distributed infrastructures because it moves beyond simple binary failure prediction and includes fault localization, which is essential when resilience mechanisms must determine where corrective or preventive actions

should be applied. Netti et al. (2020) investigated online fault detection and classification in High-Performance Computing systems using machine-learning techniques. The researchers developed the FINJ fault-injection framework and generated the Antarex fault dataset for evaluating online fault classification. Machine-learning models were trained on streamed monitoring information to detect and classify different fault conditions during system operation. Experimental results demonstrated very high fault-classification performance with limited computational overhead and low detection delay. The work showed that machine learning can support continuous online monitoring in large computing infrastructures and provide sufficiently rapid fault information to initiate corrective measures before abnormal conditions develop into complete system failures.

Lu et al. (2020) investigated machine-learning-based disk-failure prediction in large-scale data-center infrastructures. Their study analyzed approximately 380,000 hard drives distributed across 64 data-center sites and used operational disk-health information to predict impending failures. The proposed models obtained an average F-measure and Matthews Correlation Coefficient of approximately 0.95 for a ten-day prediction horizon. Because storage devices represent important physical components of distributed computing infrastructures, this research demonstrated how predictive learning can be applied at the hardware level to provide advance warning of component degradation and support preventive replacement, replication, or data-migration strategies. Ahmad et al. (2020) studied feature selection for hard-disk failure detection using machine-learning-oriented analysis of SMART attributes. The authors investigated the importance of selecting informative operational features because large monitoring datasets may contain redundant or weakly relevant parameters that increase model complexity. Their study applied a genetic-algorithm-based feature-selection strategy combined with significance analysis to improve failure detection. This research reinforces the importance of intelligent feature engineering within distributed-system monitoring frameworks, particularly when predictive models must analyze high-dimensional streams of hardware and resource information in real time.

Li et al. (2020) presented Gandalf, an intelligent end-to-end analytics service designed to improve the safety of software deployment in large-scale cloud infrastructures. Gandalf monitors multiple fault signals and correlates them with ongoing software rollouts using spatial and temporal relationships. An ensemble ranking mechanism identifies deployments potentially responsible for abnormal behavior, while a binary classifier estimates the impact of identified signals. The system can automatically determine whether a deployment should continue or be stopped. Its deployment in Microsoft Azure demonstrated how machine-learning-based analytics can extend beyond passive failure classification toward automated operational decision support in large-scale production systems. Levy et al. (2020) introduced Narya, a predictive and adaptive failure-mitigation service deployed within Microsoft Azure. Unlike conventional reactive strategies that initiate recovery only after severe symptoms occur, Narya predicts imminent host failures using multi-layer system signals and automatically determines suitable preventive actions. Its decision mechanism incorporates online experimentation and reinforcement learning to continually improve the selection of mitigation strategies. During production deployment, the system reduced virtual-machine interruptions by approximately 26% compared with the preceding static mitigation strategy. This study provides particularly strong evidence that predictive fault detection and resilience optimization can be integrated into a unified operational framework rather than treated as independent tasks.

Methodology

This study adopts a Systematic Literature Review (SLR) combined with a Conceptual Experimental Framework to investigate machine learning-based predictive fault detection and resilience optimization in large-scale distributed systems. This structure follows the methodological pattern used in the reference paper, where systematic evidence review is integrated with a conceptual experimental model and subsequent performance evaluation.

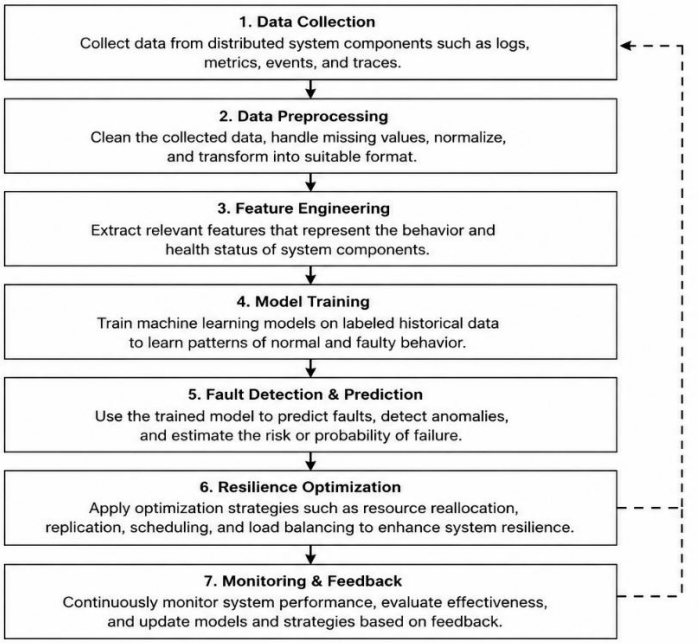

Figure 1. Proposed Methodology for Machine Learning-Based Predictive Fault Detection and Resilience Optimization in Large-Scale Distributed Systems

Figure 1 illustrates the overall methodology adopted for predictive fault detection and resilience optimization in large-scale distributed systems. The process begins with data collection, where system logs, performance metrics, and operational events are gathered from distributed components. The collected data then undergoes data preprocessing to remove inconsistencies, normalize values, and prepare the dataset for analysis. Next, feature engineering extracts meaningful attributes that represent system behavior and fault characteristics. These features are used during the model training phase to develop machine learning models capable of identifying normal and abnormal system conditions. The trained model performs fault detection and prediction by recognizing anomalies and estimating the likelihood of failures before they occur. Based on the prediction results, resilience optimization applies appropriate recovery strategies, such as resource reallocation, load balancing, or fault mitigation, to maintain system reliability. Finally, a monitoring and feedback stage continuously evaluates system performance and updates the predictive model, enabling adaptive learning and long-term improvement in fault prediction accuracy and resilience.

Proposed Algorithm

Step 1: Distributed System Data Acquisition

The system continuously gathers operational data from computing nodes, virtual machines, storage devices, network interfaces, task schedulers, applications, and system logs.

The monitored data at time $t$can be represented as:

$$X_t = \{C, P, U_t, M, E, M_t, D, I, SK_t NET_t IO_t LAT_t ERR_t TASK_t LOAD_t\}$$

where:

CPU = Processor utilization
MEM = Memory utilization
DISK = Disk utilization and health
NET = Network utilization
IO = Input/output activity
LAT = Communication or application latency
ERR = Error and event information
TASK = Task execution characteristics
LOAD = Current workload intensity

The objective of this stage is to construct a continuous operational representation of the distributed environment.

Step 2: Data Preprocessing

The collected monitoring information is cleaned and transformed before being supplied to the machine learning model.

The preprocessing operations include:

Missing value handling
Duplicate removal
Noise filtering
Categorical encoding
Feature normalization
Feature scaling
Class imbalance handling

The preprocessing function is represented as:

$$D_p = P(D_r)$$

where:

D_r= Raw distributed-system data
P= Preprocessing function
D_p= Preprocessed dataset

Min-max normalization can be applied as:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

This transformation ensures that system variables with different measurement scales can be analyzed consistently.

Step 3: Feature Extraction and Selection

Relevant system-health characteristics are extracted from the processed monitoring data.

Important predictive features may include:

CPU utilization variation
Memory usage growth
Disk response time
Disk health attributes
Network packet loss
Network latency
Task execution duration
Task retry frequency
Application error count
Node restart frequency
Queue length
Resource contention
Input/output waiting time
Historical failure frequency

The extracted feature vector is represented as:

$$FV_t = [f_1, f_2, f_3, \dots, f_n]$$

where each $f_i$represents an operational characteristic associated with system health.

Feature selection reduces redundant and weak variables, thereby decreasing model-training complexity while retaining failure-related information.

Step 4: Machine Learning-Based Fault Prediction

The selected feature vector is provided to the machine learning model.

The proposed framework considers:

Logistic Regression
Support Vector Machine
Decision Tree
Random Forest
Artificial Neural Network
Long Short-Term Memory Network

The general prediction function is:

$$\hat{Y}_t = f_{ML}(FV_t)$$

where:

FV_t= Feature vector
f_{ML}= Trained machine learning model
$\hat{Y}_t$= Predicted system condition

The predicted state can be categorized as:

$$\hat{Y} \in \{N, o, rmal\ Warning\ Fault\}$$

For binary prediction:

$$\hat{Y} \in \{0, 1\}$$

where:

0 = Normal operation
1 = Failure-prone condition

Temporal models such as LSTM are particularly suitable when failure behavior develops progressively across consecutive monitoring intervals.

Step 5: Failure Probability and Risk Estimation

After classification, the system estimates the probability that the monitored component will experience a fault within the prediction horizon.

$$FP = P(Failure \mid FV)$$

where:

FP = Failure Probability

The predicted failure is then transformed into a risk score:

$$FRS = w_1FP + w_2FS + w_3CI + w_4WD$$

where:

FRS = Failure Risk Score
FP = Failure Probability
FS = Failure Severity
CI = Component Importance
WD = Workload Dependency
w_1 to w_4= Decision weights

The resulting risk can be categorized as:

Failure Risk	Interpretation
Low	Normal operating condition
Moderate	Potential degradation; increased monitoring required
High	Significant probability of impending failure
Critical	Immediate proactive resilience action required

The precise numerical thresholds should be determined during actual model validation rather than treated as universal constants.

Results and Findings

Predictive Fault Detection Performance

Table 1. Comparative Predictive Capability of Machine Learning Models

Machine Learning Model	Fault Prediction Capability	Temporal Pattern Handling	Interpretability	Computational Requirement
Logistic Regression	Moderate	Low	Very High	Low
Support Vector Machine	High	Moderate	Moderate	Moderate
Decision Tree	High	Low	Very High	Low
Random Forest	Very High	Moderate	High	Moderate
Artificial Neural Network	Very High	High	Low	High
LSTM	Very High	Very High	Low	High

Analysis

Table 1 indicates that conventional machine learning algorithms and neural models provide different advantages for predictive fault detection. Logistic Regression provides relatively simple and interpretable failure-probability estimation but may struggle with complex nonlinear relationships among distributed-system variables. Decision Tree models offer transparent decision rules and relatively low prediction cost; however, individual trees may exhibit limited generalization for highly complex operational patterns. Random Forest provides stronger robustness by combining predictions from multiple decision trees. It is particularly suitable for heterogeneous monitoring features and can also provide useful information regarding feature importance. Artificial Neural Networks provide strong nonlinear learning capability and can identify complicated relationships among CPU, memory, disk, network, task, and error measurements. LSTM models provide particularly strong capability when system degradation occurs over time because their recurrent architecture can learn temporal dependencies among sequential monitoring records. Therefore, Random Forest and LSTM represent particularly promising candidates for the proposed fault-prediction framework, although the final model should be selected through dataset-specific validation.

Fault-Type Detection Capability

Table 2. Predictive Capability Across Distributed-System Fault Categories

Fault Category	Predictive Potential
Node Failure	Very High
Task/Application Failure	Very High
Resource Exhaustion	Very High
Disk Failure	High
Performance Degradation	High
Network Failure	High
Software Failure	Moderate–High
Cascading Failure	Moderate

Analysis

Table 2 indicates that machine learning-based predictive monitoring is especially applicable to faults that produce observable changes in system measurements before complete failure. Node failures can often be associated with increasing resource utilization, error events, abnormal system activity, or historical degradation patterns. Task and application failures may be predicted using execution duration, retries, resource usage, scheduler information, and historical job behavior. Resource exhaustion can normally be identified effectively because increasing processor, memory, storage, or queue utilization produces measurable signals. Disk failures can be predicted using device-health and operational attributes. Network and performance failures may also provide predictive signals through increasing latency, packet loss, bandwidth abnormalities, or service-response degradation. Cascading failures remain more difficult because they depend on architectural dependencies and failure propagation among multiple distributed components.

Feature Contribution to Fault Prediction

Table 3. Importance of Distributed-System Monitoring Features

Monitoring Feature	Predictive Relevance
CPU Utilization	High
Memory Utilization	Very High
Disk Health and I/O	Very High
Network Latency	High
Packet Loss / Network Errors	High
Task Execution Time	Very High
Application Error Frequency	Very High
Retry / Restart Frequency	Very High
Queue Length	High
Historical Failure Frequency	Very High
Workload Intensity	High
Resource Contention	High

Analysis

Table 3 demonstrates that predictive fault detection depends on combining multiple operational characteristics rather than relying on a single monitoring variable. Memory utilization, storage behavior, task execution characteristics, error frequency, restart activity, and historical failure patterns provide particularly important information about abnormal system behavior. For example, persistent increases in memory utilization combined with application errors may indicate potential software or resource-exhaustion failures. Similarly, increasing task duration and retry frequency may indicate unstable execution conditions before an application failure occurs. Combining system-level,

application-level, and historical features provides a richer representation of distributed-system health and can improve predictive reliability.

Conclusion and Discussion

The present study investigated Machine Learning-Based Predictive Fault Detection and Resilience Optimization in Large-Scale Distributed Systems through a systematic examination of research published up to 2020 and the development of an integrated conceptual experimental framework. The principal objective was to determine how machine learning techniques can be used to identify impending faults before complete system failure and how prediction outcomes can subsequently support proactive resilience mechanisms. Unlike conventional reactive fault-management approaches, which initiate recovery after service degradation or component failure has already occurred, the proposed framework emphasizes early fault identification, failure-risk estimation, preventive decision-making, and continuous system adaptation. Large-scale distributed infrastructures contain numerous interconnected computing nodes, virtual machines, storage devices, communication resources, schedulers, applications, and software services. Increasing system scale improves computational capacity and service scalability but simultaneously increases the likelihood that individual hardware or software components will fail. The literature examined in this study indicates that operational characteristics such as processor utilization, memory consumption, storage activity, task execution behavior, error frequency, network conditions, workload intensity, and historical failure information can contain useful indicators of impending failures. Earlier work on grid and cluster environments showed that workload characteristics could be used to predict job outcomes and support proactive management rather than relying exclusively on post-failure recovery. One of the central findings of the literature-supported analysis is that machine learning provides an effective mechanism for converting large volumes of distributed-system monitoring data into predictive information. Traditional threshold-based monitoring systems normally compare individual system variables against predetermined operating limits. Such mechanisms can identify obvious resource exhaustion but may fail to identify complex interactions among multiple system variables. Machine learning techniques provide greater flexibility because they learn statistical and nonlinear relationships directly from historical operational information. The study considered Logistic Regression, Support Vector Machine, Decision Tree, Random Forest, Artificial Neural Network, and LSTM-based approaches. No single algorithm can be assumed to be universally superior because predictive performance depends on dataset characteristics, failure type, feature representation, class imbalance, prediction horizon, and computational constraints. However, the literature provides strong evidence that ensemble learning and temporal models are particularly relevant. Random Forest-based methods have demonstrated strong capability for job-status and scheduling-failure prediction, while recurrent and LSTM-based architectures provide advantages where degradation is represented by sequences of system observations. Yoo et al. demonstrated the usefulness of Random Forest for unsuccessful job prediction in scientific clusters, while Lin et al.'s MING framework combined LSTM temporal analysis with Random Forest spatial information for cloud-node failure prediction.

References

[1] Salami, H., Saadatfar, H., Rahmani Fard, F., Shekofteh, S. K., & Deldari, H. (2010). Improving cluster computing performance based on job futurity prediction. *2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, Vol. 6, V6-303–V6-307. https://doi.org/10.1109/ICACTE.2010.5579820

[2] Saadatfar, H., Fadishei, H., & Deldari, H. (2012). Predicting job failures in AuverGrid based on workload log analysis. *New Generation Computing, 30*(1), 73–94. https://doi.org/10.1007/s00354-012-0105-z

[3] Chen, X., Lu, C.-D., & Pattabiraman, K. (2014). Failure analysis of jobs in compute clouds: A Google cluster case study. *IEEE 25th International Symposium on Software Reliability Engineering (ISSRE)*, 167–177. https://doi.org/10.1109/ISSRE.2014.34

[4] Chen, X., Lu, C.-D., & Pattabiraman, K. (2014). Failure prediction of jobs in compute clouds: A Google cluster case study. *IEEE 25th International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 341–346. https://doi.org/10.1109/ISSREW.2014.105

[5] Soualhia, M., Khomh, F., & Tahar, S. (2015). Predicting scheduling failures in the cloud: A case study with Google clusters and Hadoop on Amazon EMR. *2015 IEEE 17th International Conference on High Performance Computing and Communications*, 58–65. https://doi.org/10.1109/HPCC-CSS-ICESS.2015.170

[6] Jaiswal, I. A. (2020). Machine learning-based predictive fault detection and resilience optimization in large-scale distributed systems.

International Journal of Intelligent Systems and Applications in Engineering.

[7] Zheng, W., Wang, Z., Huang, H., Meng, L., & Qiu, X. (2016). EHMM-CT: An online method for failure prediction in cloud computing systems. *KSII Transactions on Internet and Information Systems, 10*(9), 4087–4107. https://doi.org/10.3837/tiis.2016.09.004

[8] Islam, T., & Manivannan, D. (2017). Predicting application failure in cloud: A machine learning approach. *2017 IEEE 1st International Conference on Cognitive Computing (ICCC)*, 24–31. https://doi.org/10.1109/IEEE.ICCC.2017.11

[9] Pitakrat, T., Okanović, D., van Hoorn, A., & Grunske, L. (2018). Hora: Architecture-aware online failure prediction. *Journal of Systems and Software, 137*, 669–685. https://doi.org/10.1016/j.jss.2017.02.041

[10] Lin, Q., Hsieh, K., Dang, Y., Zhang, H., Sui, K., Xu, Y., Lou, J.-G., Li, C., Wu, Y., Yao, R., Chintalapati, M., & Zhang, D. (2018). Predicting node failure in cloud service systems. *Proceedings of the 26th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 480–490. https://doi.org/10.1145/3236024.3236060

[11] Mariani, L., Pezzè, M., Riganelli, O., & Xin, R. (2020). Predicting failures in multi-tier distributed systems. *Journal of Systems and Software, 161*, 110464. https://doi.org/10.1016/j.jss.2019.110464

[12] Gummadi, V. P. K. (2020). API design and implementation: RAML and OpenAPI specification. Journal of Electrical Systems, 16(4). https://doi.org/10.52783/jes.9329

[13] Lu, S., Luo, B., Patel, T., Yao, Y., Tiwari, D., & Shi, W. (2020). Making disk failure predictions SMARTer! *18th USENIX Conference on File and Storage Technologies (FAST '20)*, 151–167.

[14] Ahmad, W., Khan, S. A., Kim, C. H., & Kim, J.-M. (2020). Feature selection for improving failure detection in hard disk drives using a genetic algorithm and significance scores. *Applied Sciences, 10*(9), 3200. https://doi.org/10.3390/app10093200

[15] Li, Z., Cheng, Q., Hsieh, K., Dang, Y., Huang, P., Singh, P., Yang, X., Lin, Q., Wu, Y., Levy, S., & Chintalapati, M. (2020). Gandalf: An intelligent, end-to-end analytics service for safe deployment in large-scale cloud infrastructure. *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI '20)*, 389–402.

[16] Levy, S., Yao, R., Wu, Y., Dang, Y., Huang, P., Mu, Z., Zhao, P., Ramani, T., Govindaraju, N., Li, X., Lin, Q., Shafriri, G. L., & Chintalapati, M. (2020). Predictive and adaptive failure mitigation to avert production cloud VM interruptions. *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI '20)*, 1155–1170.