







A Proactive Approach to Fault Tolerance Using Predictive Machine Learning Models in Distributed Systems







Mohd Haroon¹, Zeeshan Ali Siddiqui^{2*}, Mohammad Husain³, Arshad Ali³ and Tameem Ahmad⁴

¹Department of Computer Science and Engineering, Integral University, Lucknow, India, ²Department of Computer Science and Engineering, University of Lucknow, Lucknow, India; ³Faculty of Computer and Information System, Islamic University of Madinah, Saudi Arabia; ⁴Department of Computer Engineering, Aligarh Muslim University, Aligarh, India

E-mail/Orcid Id:

MH,  mharoon@iul.ac.in,  <https://orcid.org/0000-0001-7967-7302>; ZAS,  zeealis@gmail.com,  <https://orcid.org/0000-0002-0049-2362>;

MH,  dr.husain@iu.edu.sa,  <http://orcid.org/0000-0001-7312-9567>; AA,  a.ali@iu.edu.sa,  <https://orcid.org/0000-0001-5625-0867>;

TA,  tameemahmad@zhcet.ac.in,  <https://orcid.org/0000-0001-9802-5713>

Article History:

Received: 26th Apr., 2024

Accepted: 21st Oct., 2024

Published: 30th Oct., 2024

Keywords:

Cloud computing, distributed systems, preventive maintenance, proactive fault tolerance, random forest machine learning models

How to cite this Article:

Mohd Haroon, Zeeshan Ali Siddiqui, Mohammad Husain, Arshad Ali, and Tameem Ahmad (2024). A Proactive Approach to Fault Tolerance Using Predictive Machine Learning Models in Distributed Systems. *International Journal of Experimental Research and Review*, 44, 208-220.

DOI:

<https://doi.org/10.52756/ijerr.2024.v44spl.018>

Abstract: In the era of cloud computing and large-scale distributed systems, ensuring uninterrupted service and operational reliability is crucial. Conventional fault tolerance techniques usually take a reactive approach, addressing problems only after they arise. This can result in performance deterioration and downtime. With predictive machine learning models, this research offers a proactive approach to fault tolerance for distributed systems, preventing significant failures before they arise. Our research focuses on combining cutting-edge machine learning algorithms with real-time analysis of massive streams of operational data to predict abnormalities in the system and possible breakdowns. We employ supervised learning algorithms such as Random Forests and Gradient Boosting to predict faults with high accuracy. The predictive models are trained on historical data, capturing intricate patterns and correlations that precede system faults. Early defect detection made possible by this proactive approach enables preventative remedial measures to be taken, reducing downtime and preserving system integrity. To validate our approach, we designed and implemented a fault prediction framework within a simulated distributed system environment that mirrors contemporary cloud architectures. Our experiments demonstrate that the predictive models can successfully forecast a wide range of faults, from hardware failures to network disruptions, with significant lead time, providing a critical window for implementing preventive measures. Additionally, we assessed the impact of these pre-emptive actions on overall system performance, highlighting improved reliability and a reduction in mean time to recovery (MTTR). We also analyse the scalability and adaptability of our proposed solution within diverse and dynamic distributed environments. Through seamless integration with existing monitoring and management tools, our framework significantly enhances fault tolerance capabilities without requiring extensive restructuring of current systems. This work introduces a proactive approach to fault tolerance in distributed systems using predictive machine learning models. Unlike traditional reactive methods that respond to failures after they occur, this work focuses on anticipating faults before they happen.

Introduction

In distributed systems, fault tolerance is crucial for maintaining reliability and availability, particularly as these systems developed in additionally complex and

increasingly critical industries like finance, healthcare, and cloud computing. The intricate nature, diversity and large number of components in distributed systems make them vulnerable to various types of failures (Kirti et al.,



2024a). To address this vulnerability, fault tolerance mechanisms are implemented to ensure continuous operation and correct functionality despite the occurrence of faults. As distributed systems grow in complexity, the importance of fault tolerance increases. In finance, for instance, uninterrupted service is paramount for transaction processing and real-time trading platforms. In healthcare, reliable systems are crucial for diagnostics, patient management, and treatment monitoring. Cloud computing services must ensure high availability to support a wide range of applications and services used by millions of users globally (Pal et al., 2023; Kumar et al., 2023; Swarnalatha et al., 2024; Zou et al., 2024). The diverse and distributed components of these systems can experience hardware failures, software bugs, and network issues. Hardware failures might include server crashes or disk failures, while software bugs could lead to unexpected system behaviour. Network issues can result in communication breakdowns between nodes (Lu et al., 2024). Each of these failures can potentially disrupt the entire system if not managed effectively.

Fault tolerance mechanisms, such as redundancy, replication, checkpointing, and failover strategies, are implemented to mitigate these risks. Redundancy is the process of making duplicates of important parts so that, should one fail, another can take over. Replication ensures that multiple copies of data or services are maintained across different nodes, preventing data loss and service interruption (Bessani et al., 2014; Sun et al., 2018). Checkpointing periodically saves the system's state, allowing it to roll back to a known good state in case of failure (Elnozahy et al., 2002; Gossman, et al., 2024). Failover mechanisms enable automatic switching to backup systems when primary systems fail, ensuring continuity of operations. Moreover, modern approaches to fault tolerance are incorporating predictive analytics and machine learning to preemptively identify potential issues. Through the examination of past data and identification of trends that predate malfunctions, these systems are able to implement remedial measures prior to malfunctions, hence augmenting dependability and accessibility. To ensure the dependability and availability of distributed systems (Siddiqui and Haroon, 2023), fault tolerance is a basic necessity. As these systems play increasingly pivotal roles in various critical sectors, the implementation of robust fault tolerance mechanisms becomes ever more important to ensure continuous, correct operation even in the face of inevitable failures. Various approaches are employed to achieve fault tolerance, each designed to address specific types of failures and meet particular system requirements. Here's

an overview of common fault tolerance techniques in distributed systems.

Replication

In distributed systems, replication is a failure tolerance approach that involves maintaining multiple copies of data or services across different nodes. This approach ensures that if one node fails, the system can continue to function using the replicated data from another node. Replication can be carried out synchronously, where updates are immediately propagated to all replicas, ensuring consistency but potentially adding latency. Conversely, asynchronous replication allows for quicker operations but may result in temporary inconsistencies.

Common replication strategies include active replication, where all replicas handle requests simultaneously, and passive replication, where a primary replica manages requests and updates the backups. Replication enhances system reliability ((Siddiqui and Haroon, 2024)), availability, and load balancing, but it also requires careful management to address challenges such as data consistency, network overhead, and storage costs. Replication in distributed systems is a fundamental technique for ensuring data availability, reliability, and fault tolerance. Mathematical formulas for replication typically involve determining the number of replicas, understanding quorum requirements, and assessing the trade-offs between availability and consistency (Garg, 2022). In quorum-based replication, a read-or-write operation requires approval from a certain number of replicas (quorum) to ensure data consistency. This approach is often used in distributed databases and consensus algorithms (Hasan and Zeebaree, 2024).

The following conditions must be met to ensure consistency, the number of replicas, n , represents how many copies of a piece of data or service exist across different nodes:

- $W+R>n$
- $W>n/2$
- $R>n/2$

Availability measures the proportion of time that the system can successfully respond to requests. With replication, the availability of the system improves as more replicas are added, but this can be subject to the number of available replicas (Eckart et al., 2008). If the system requires k out of n replicas to be available for operation, the availability can be calculated as:

$$A=1-(1-p)^k$$

Redundancy

Redundancy is a critical concept in fault-tolerant systems, where extra components or systems are added to take over in case of failure. Redundancy ensures continued operation and improves the system’s reliability (Bandari, 2020). Redundancy is a fault tolerance technique in distributed systems that involves duplicating critical components or functions to prevent system failures. By maintaining multiple copies of hardware, software, or network paths, redundancy makes sure that if one part fails, another can take over without any problems., allowing the system to continue operating without interruption. There are various types of redundancy, including hardware redundancy, software redundancy, and information redundancy. Hardware redundancy involves the use of additional physical components, such as extra servers or power supplies. Software redundancy entails running multiple instances of software applications on different systems. Information redundancy involves duplicating data across multiple storage devices. While redundancy significantly enhances system reliability and availability, it also increases costs and complexity. Therefore, careful design and management are required to balance these trade-offs effectively.

Series Redundancy:

The system fails if any single component fails. Series redundancy is a fault tolerance technique where multiple redundant components are arranged in a sequential manner to ensure system reliability. In this configuration, each component in the series must function correctly for the overall system to operate. If one component fails, the next in line takes over to maintain continuity. This technique is commonly used in scenarios where high reliability is critical, such as in power supply systems and communication networks. Series redundancy ensures that the failure of a single component does not lead to total system failure, thereby increasing the system's overall reliability. However, it can introduce higher latency and complexity, as each component must be capable of seamlessly taking over the task of the failed one. Effective monitoring and maintenance are crucial to managing the increased complexity and ensuring that all components are functioning correctly. Series redundancy can significantly enhance fault tolerance but requires careful design to avoid single points of failure and ensure smooth transitions between components.

System Reliability:

The reliability of a series system with n components is the product of the reliability of each component Ri

$$R_s = R_1 \times R_2 \times \dots \times R_n = \prod_{i=1}^n R_i$$

In a parallel configuration, the system continues to function as long as at least one component is operational. This configuration improves overall system reliability. The reliability of a parallel system with n components is calculated by the probability that at least one component does not fail (Power and Kotonya, 2018).

$$R_p = 1 - \prod_{i=1}^n (1 - R_i)$$

In k-out-of-n redundancy, the system is operational as long as at least k out of n components are functioning. This is a common setup in fault-tolerant systems.

The reliability of a k-out-of-n system can be calculated using the binomial probability formula:

$$R_{k,n} = \sum_{i=k}^n \binom{n}{i} R^i (1 - R)^{n-i}$$

Where $\binom{n}{i}$ is the binomial coefficient and R is the reliability of each component.

For a 2-out-of-3 system where each component has a reliability of 0.95:

$$R_{2,3} = \binom{3}{2} * .95^2 * (1 - .95) + \binom{3}{3} * .95^3$$

$$R_{2,3} = 3 * .95^2 * .05 + 1 * .95^3$$

$$R_{2,3} = .135375 + .857375 = .99275$$

Redundancy affects the mean time to failure (MTTF) and mean time to repair (MTTR) of a system. For a series system, the system’s MTTF is lower because any single failure will cause the system to fail (Kochhar and Jabanjalin, 2017). Figure 1 displays the system model for fault tolerance.

$$MTTF \text{ series} = \frac{1}{\sum_{i=1}^n \frac{1}{MTTF}}$$

Consensus Algorithms

Consensus algorithms are fundamental in distributed systems to ensure that multiple nodes can agree on a common state or decision, even in the presence of faults. The mathematical models for consensus algorithms typically revolve around ensuring properties such as safety (no two nodes decide on different values) and liveness (all non-faulty nodes eventually decide on a value) (Polze et al., 2011).

Byzantine Fault Tolerance is crucial when dealing with nodes that may fail or act maliciously. The objective is to achieve consensus despite up to f Byzantine faults among N nodes.

- Consensus can be achieved if and only if $N \geq 3f + 1$
- The quorum size Q must satisfy $Q > N + f/2$
- Prepare phase: $Q \text{ prepare} \geq 2f + 1$
- Commit phase: $Q \text{ commit} \geq 2f + 1$

Ensures no two honest nodes commit different values.

For any two nodes n_i, n_j : where $v_i=v_j$
 Ensures all non-faulty nodes eventually commit a value.

$$\exists t: \forall n_i \in N - f$$

Paxos

A family of protocols for achieving consensus in a network of unreliable processors. Paxos is designed for environments where nodes can fail (crash fault tolerance)

Acceptors promise not to accept any proposal numbered less than n

The proposer sends an acceptance request for proposal number n and value v to the quorum. Acceptors accept the proposal if it is the highest number they have seen.

$$Q \text{ accept} > N/2$$

Once a quorum of acceptors has accepted a proposal, it is considered decided.

$$\text{Safety: } Q \text{ accept} \cap Q \text{ prepare} \neq \emptyset$$

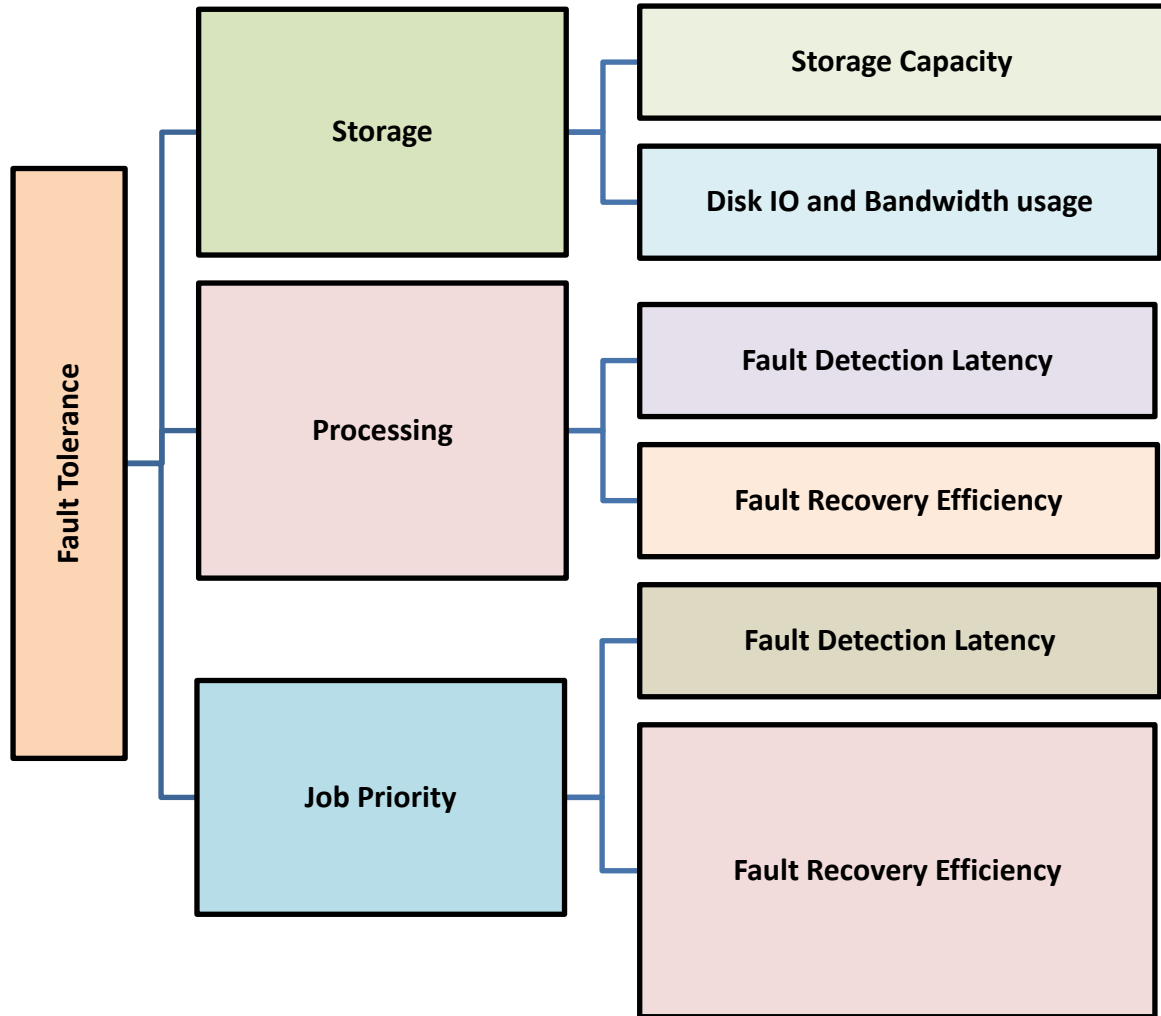


Figure 1. System Model for fault tolerance.

Table 1. Algorithm properties analysis.

Algorithms	Fault Model	Nodes Required (N)	Quorum Size (Q)	Properties
PBFT	Byzantine	$N \geq 3f+1$	$Q > N+f/2$	Safety, Liveness, Tolerates faults
Paxos	Crash	$N \geq 2f+1$	$Q > N/2$	Safety, Liveness
Raft	Crash	$N \geq 2f+1$	$Q > /2N$	Safety, Liveness

but do not act maliciously (Mukwevho and Celik, 2018).

A proposer sends a prepared request with proposal number n to a quorum Q of acceptors. Acceptors respond with the highest-numbered proposal they have accepted.

$$Q \text{ prepare} > N/2$$

Raft:

A consensus algorithm designed to be easier to understand and implement than Paxos. Raft simplifies consensus by dividing the problem into leader election, log replication and safety.

A node becomes a leader if it receives votes from a majority of nodes.

The leader wins if it receives votes $\geq N/2+1$.

The leader replicates log entries to a majority of followers. $Q \log \geq N/2+1$ (Khan and Haroon, 2022).

A log entry is committed if it is stored on a majority of nodes and the leader who created it is still in power. Entry e is committed if $\exists Q:e$ is in the logs of Q nodes and the leader's term is valid. The properties of PBFT, Paxos, and Raft algorithms are analysed in Table 1.

These mathematical models and properties help in designing, analyzing and validating consensus algorithms in distributed systems, ensuring they can achieve agreement and maintain functionality despite faults.

Research Gap:

The objective of this research is to address the pressing necessity for fault tolerance in distributed systems, which are becoming increasingly complex and vital for industries such as finance, healthcare, and cloud computing. As these systems continue to expand, the risk of various failures (e.g., hardware, software, and network) grows, necessitating robust fault tolerance mechanisms. The study aims to explore and enhance existing fault tolerance techniques such as replication, redundancy, and checkpointing and introduces predictive machine learning models to provide a proactive approach for mitigating potential failures. By employing predictive analytics, the research seeks to anticipate failures before they occur, thus improving the reliability and availability of distributed systems.

Traditional fault tolerance techniques primarily focus on reactive measures, addressing system failures only after they have occurred. There is a lack of research on proactive fault tolerance, especially leveraging predictive machine learning models to pre-emptively identify and mitigate faults before they impact the system. While fault tolerance mechanisms are well-researched, their impact on system performance, particularly in terms of downtime reduction, mean time to recovery (MTTR), and real-time responsiveness, has not been sufficiently studied.

The proposed solution offers insights into how proactive fault tolerance can enhance system reliability while minimizing performance trade-offs. We utilize supervised learning algorithms like Random Forests and Gradient Boosting (Yadav et al., 2024) to achieve high-accuracy fault predictions.

Related Work

Distributed systems are the backbone of modern computing infrastructures, powering cloud services, large-scale applications, and enterprise solutions. They provide the scalability, flexibility, and resilience required to handle vast amounts of data and serve millions of users. However, the inherent complexity and interdependencies within these systems make them susceptible to various types of faults and failures. Hardware malfunctions, software bugs, network disruptions, and resource contention are just a few examples of issues that can jeopardize system reliability and availability (Srivastava et al., 2013).

Traditional fault tolerance techniques in distributed systems typically involve reactive measures, such as redundancy, failover mechanisms, and post-failure recovery processes (Veer and Bhardwaj, 2024; Obadia et al., 2014). While effective in many cases, these approaches often address problems only after they have impacted the system, leading to service interruptions, data loss, and increased operational costs. In critical applications, even brief downtimes can have significant repercussions, from financial losses to reputational damage (Kalaskar and Thangam, 2023).

To overcome the limitations of reactive fault tolerance, there is a growing need for proactive strategies that can predict and prevent failures before they occur. This shift from reaction to anticipation is driven by the advancements in machine learning (Mondal et al., 2023) and artificial intelligence, which offer powerful tools for analyzing and interpreting complex data patterns. By harnessing these technologies, we can develop systems capable of detecting early signs of potential issues and taking preemptive actions to avoid faults (Gururaj et al., 2023a). Table 2 shows the analyses of state-of-the-art techniques of fault tolerance.

Table 2. Critique of state-of-the-art techniques.

References	Technique	Finding
Dhingra and Gupta, 2017	SVM, Decision Trees, Random Forest, Fault Prediction, Preemptive Migration	Machine learning can predict faults early, reducing system downtime and operational costs.
Yang et al., 2023	LSTM, CNN, Hybrid Models, Predictive Fault Management, Edge Node Replication	Machine learning can predict faults early, reducing system downtime and operational costs.
Lima et al., 2021	Neural Networks, Bayesian Networks, Predictive Maintenance, Task Migration	Neural networks effectively predict system failures; proactive task migration minimizes impact.
Bharany et al., 2022	Various ML Algorithms (Survey), Fault Detection, Resource Redundancy	Ensemble methods generally outperform individual models in predicting system faults.
Karadayi et al., 2020	K-means, DBSCAN, LSTM, Anomaly Detection, Preemptive Action	ML-based anomaly detection enhances early fault detection in IoT systems.
AlOrbani et al., 2021	Reinforcement Learning, Dynamic Resource Allocation, Load Balancing	Reinforcement learning optimizes resource allocation.
Chakrabarty et al., 2019	Linear Regression, Gradient Boosting, Fault Prediction, Redundant Scheduling	Predictive models anticipate failures, improving system reliability and uptime.
Al Qassem et al., 2023	Random Forest, Logistic Regression, Predictive Fault Management, Auto-scaling	Random forest models are effective in predicting cloud system faults; auto-scaling improves robustness.
Seba et al., 2024	LSTM, CNN, Hybrid Models, Predictive Fault Management, Edge Node Replication	Combining temporal and spatial data in hybrid models enhances fault tolerance in edge computing.
Ren, 2021	Ensemble Methods (Bagging, Boosting), Predictive Maintenance, Hot Standby Redundancy	Ensemble learning methods significantly reduce system downtime through effective fault prediction.
Singh and Singh, 2023	Feature selection and deep learning	Deep learning classifiers (CNN, FFNN, RBN) improve fault prediction rate.

Methodology

The primary challenge in implementing proactive fault tolerance in distributed systems is the ability to accurately predict faults in a timely manner. This requires sophisticated models that can process vast amounts of real-time data, identify subtle anomalies, and forecast impending failures with high precision. Additionally, these models must be integrated seamlessly into existing systems, providing actionable insights without introducing significant overhead or complexity (Venkataraman, 2023).

Traditional fault detection methods often rely on predefined rules or simple statistical techniques, which may not capture the dynamic and non-linear nature of modern distributed systems. As a result, there is a pressing need for more advanced approaches that leverage the predictive power of machine learning. These approaches must be capable of learning from historical data, adapting to evolving system behaviours and

providing early warnings that enable preventive maintenance and fault mitigation (Gururaj et al., 2023a).

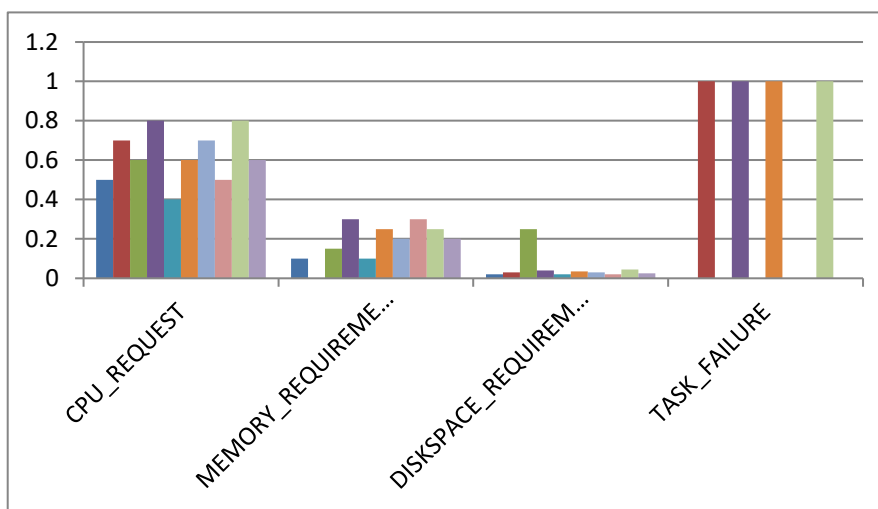
Developing Predictive Models: Designing and training machine learning models to predict a range of system faults based on historical and real-time data (Haloi and Chanda, 2024). This involves exploring various algorithms, including supervised learning and deep learning techniques, to identify the most effective models for different types of faults.

Real-Time Fault Prediction: Implementing a real-time monitoring and prediction system that can analyze incoming data streams, detect anomalies, and forecast potential failures. This system should provide sufficient lead time for operators to take corrective actions before faults occur (Gururaj et al., 2023b).

Integration and Scalability: Ensuring that the predictive fault tolerance framework can be integrated with existing distributed system architectures without significant modifications. The solution should be scalable to handle large volumes of data and adaptable to diverse operational environments (Fox and Brewer, 1999).

Table 3. Dataset for fault tolerance experimental validation.

Time	Job_Id	Task Index	Machine Id	Event Type	Priority	CPU Request	Memory Requirement	Diskspace Requirement	Task Failure
1	1001	1	3001	0	0	0.5	0.1	0.02	0
2	1002	2	3002	1	1	0.7	.2	0.03	1
3	1003	1	3003	0	2	0.6	0.15	0.25	0
4	1004	2	3004	2	3	0.8	0.3	0.04	1
5	1005	1	3005	0	1	0.4	0.1	0.02	0
6	1006	2	3006	1	2	0.6	0.25	0.035	1
7	1007	1	3007	0	3	0.7	0.2	0.03	0
8	1008	2	3008	2	0	0.5	0.3	0.02	0
9	1009	1	3009	0	2	0.8	0.25	0.045	1
10	1010	2	3010	1	1	0.6	0.2	0.025	0

**Figure 2. Task failure with respect to CPU Req, Memory Req, Disk Space Req.**

Result and Discussion

Conduction of extensive experiments to validate the accuracy and effectiveness of the predictive models are present in this section. This includes testing the system in simulated and real-world distributed environments, measuring its impact on system reliability, and comparing it with traditional fault tolerance methods (Hien, 2023). Table 3 displays the dataset.

In the table, the Event type attributes have various numerical values, here 0 means submit an event, 1 means scheduled event, and 2 means evict event. The priority attribute, 0 means the lowest priority, and increasing numerical value means the priority of the job is also increasing. Task failure 0 means no failure and 1 means the task failed. Figure 2 displays the task failure concerning CPU Req, Memory Req, and Disk Space Req.

Pre-process the Data

Read the data set and examine its contents. After reading the data set, the next step is to understand the structure, features, and target variables, Handle missing values, encode categorical variables, and normalize numerical features if necessary. In the next step Split the data into features and target variables. For the training purpose here the random forest (Swarnalatha et al., 2024) machine learning techniques have been applied. We can split the data set into several subsets of the data, build the decision tree of all subsets of the data, and then we can employ the approach of random forest to categorize the, the new example will show the fault or not. Random Forest is an ensemble machine learning algorithm that combines the predictions of multiple decision trees to improve accuracy and reduce overfitting. It is particularly effective for classification tasks, such as predicting task failures in distributed systems (Tiwari et al., 2024).

Randomly samples subsets of the training data with replacement to train each decision tree. Table 4 and Table 5 display the sample subset.

Similarly the entropy of task index calculated Entropy of task index (Figure 4).

Table 4. Samples subsets of the training data.

Time	Task Index	CPU Request	Task Failure
1	1	.5	0
2	2	.7	1
3	1	.6	0
4	2	.8	1
5	1	.4	0

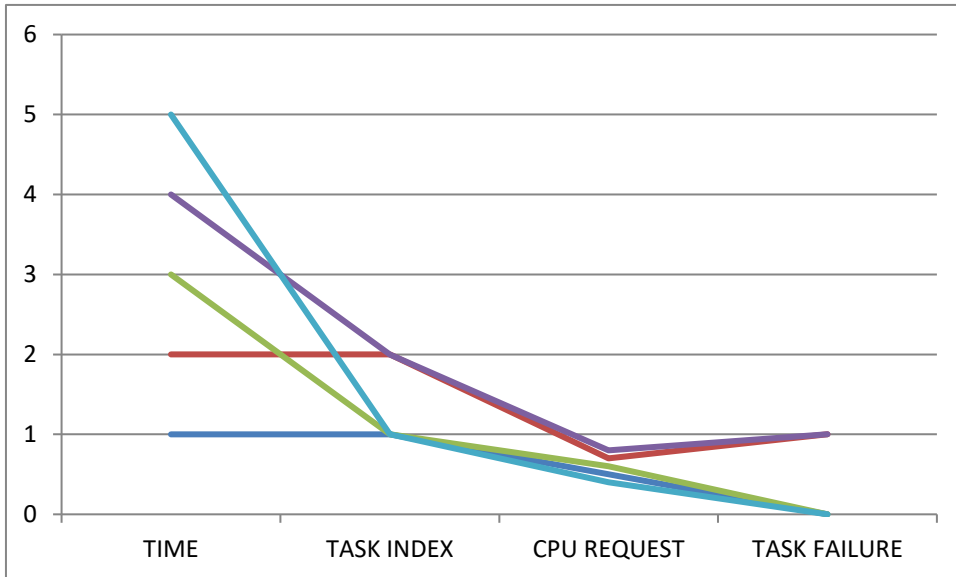


Figure 3. Data subset of Result for ensemble model 1.

In the decision tree the target attribute is the task failure, the main concern over here is to find out the root node. In the above data set the feature is Time, Task index, CPU request, and task failure. In the five examples, we found three examples are the negative example and two are the positive example. The entropy of the entire data set is calculated by the given mathematical model (Kirti et al., 2024b). Figure 3 shows the Data subset of the Result for ensemble model 1. Figure 3 shows the data subset of the Result for ensemble model 1. Figure 4 and 5 demonstrate the decision tree 1 and 2.

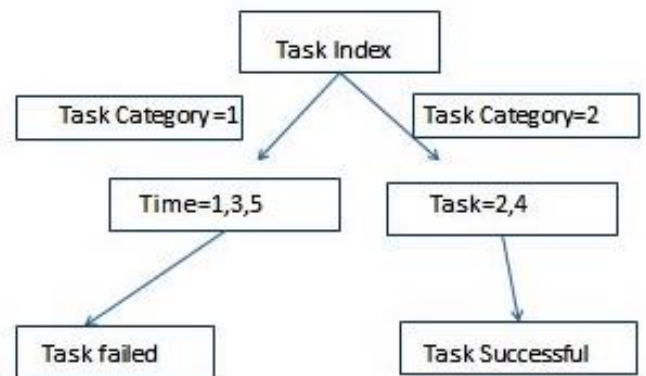


Figure 4. Decision tree 1.

Table 5. Samples subsets of the training data.

Time	Task Index	CPU Request	Task Failure
6	2	0.6	1
7	1	0.7	0
8	2	0.5	1
9	1	0.8	0
10	2	0.6	1

We have another decision tree of above data set

$$\begin{aligned}
 \text{Entropy } (S) &= -\frac{2}{5} \log_2 \left(\frac{2}{5} \right) - \frac{3}{5} \log_2 \left(\frac{3}{5} \right) \\
 &= \left(-\frac{2}{5} * -.397 - \frac{3}{5} * -.221 \right) \\
 &= -4 * -.397 - .6 * -.221 \\
 &= .2914
 \end{aligned}$$

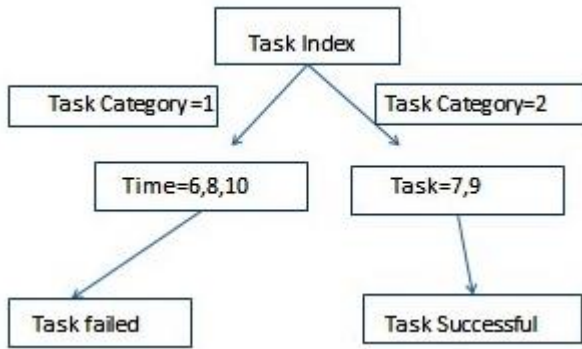


Figure 5. Decision tree 2.

After the ensemble learning, the new example is classified according to the target attribute. The example is given in the table (Sifat et al., 2024; Al-Dulaimy et al., 2022). Table 6 displays the predicted task failure. Figure 6 shows the comparative model between actual or predicted task failure.

Table 6. Predicted Task Failure.

Time	Job_Id	Task Index	Machine Id	Event Type	Priority	Cpu Request	Memory Req	Diskspace Req	Task Failure	Predicted Task Failure
1	1001	1	3001	0	0	0.5	0.1	0.02	0	0
2	1002	2	3002	1	1	0.7	.2	0.03	1	1
3	1003	1	3003	0	2	0.6	0.15	0.25	0	0
4	1004	2	3004	2	3	0.8	0.3	0.04	1	1
5	1005	1	3005	0	1	0.4	0.1	0.02	0	1
6	1006	2	3006	1	2	0.6	0.25	0.035	1	0
7	1007	1	3007	0	3	0.7	0.2	0.03	0	0
8	1008	2	3008	2	0	0.5	0.3	0.02	0	0
9	1009	1	3009	0	2	0.8	0.25	0.045	1	1
10	1010	2	3010	1	1	0.6	0.2	0.025	0	1

The confusion matrix of above data set is given below

Predicted → Actual ↓	TP	TN
	FP	1
FN	4	4

The accuracy of the model is calculated using the formula $(TP+TN)/(TP+TN+FP+FN)$, which results in 70%. This means the model correctly predicted 70% of the outcomes. The miss classification rate is defined as $(FN+FP)/(TP+TN+FP+FN)$, indicating the portion of incorrect predictions, while the false positive rate (FPR) is 0.33, and the true positive rate (TPR) is 0.75. Precision, which measures how many of the predicted positives are correct, is also 0.75. A random forest ensemble-based method was used by Lan and Li (2008), which enhanced the model's fault prediction. Because random forests are robust in forecasting

outcomes in complicated systems and can manage noisy data, they are frequently used for fault tolerance.

The use of sophisticated machine learning models for early failure detection is crucial, according to recent research on fault tolerance in distributed systems. Because of its intricacy, distributed systems are vulnerable to errors that could seriously interrupt operations. According to research, fault-tolerant systems ought to strive for prompt fault detection and self-recovery methods in addition to accuracy. To enhance defect detection without depending on centralized models, methods like federated learning, which handle data in a decentralized fashion, are being investigated. Hybrid models and deep learning are also important for increasing fault tolerance, according to recent studies. Techniques that combine ensemble approaches like random forests and recurrent neural networks (RNNs) for

instance offer greater defect detection rates and flexibility to changing contexts. Predictive maintenance and real-time monitoring are also being used by fault tolerance models nowadays to identify abnormalities early on, save downtime, and maximize resource allocation. Therefore, fault detection rates and system reliability can be greatly increased by incorporating contemporary methods like real-time monitoring and deep learning into conventional fault tolerance models like the random forest.

Conclusion

The application of predictive machine learning models significantly enhances fault tolerance in distributed systems by proactively addressing potential issues. Our analysis focused on employing Random Forest, a robust ensemble learning algorithm, to predict task failures within a distributed environment. The Random Forest model exhibited high accuracy in forecasting task failures, thereby substantially decreasing the likelihood of unexpected system downtimes. By analysing historical data, the Random Forest model identifies patterns and

anomalies that typically precede faults. This capability allows for timely interventions, as the model can signal potential issues before they develop into critical failures. The proactive nature of this fault tolerance approach facilitates preemptive maintenance and resource reallocation; further minimizing system downtime and enhancing overall reliability. Early detection of potential failures empowers system administrators to take corrective actions before issues escalate. For instance, if the model predicts a hardware component is likely to fail, administrators can replace the component during a scheduled maintenance window, rather than waiting for it to fail and cause an unscheduled outage. Similarly, if the model identifies an application likely to experience a software fault, administrators can deploy patches or redistribute workloads to mitigate the impact. By employing a Random Forest model to anticipate and stop task failures, the suggested technique increases fault tolerance in distributed systems. This proactive strategy improves resource management, strengthens maintenance plans, and decreases system downtime. It enables prompt actions before problems worsen, which results in cost savings and improved dependability. All things considered, the approach guarantees improved scalability and performance in big, complicated systems. The Random Forest model may struggle with highly dynamic environments where new failure patterns emerge rapidly, limiting its ability to adapt. It also requires significant historical data for accurate predictions, which may not always be available. Additionally, the model's complexity can lead to higher computational costs in large-scale systems.

Conflict of Interest

The authors declare no conflict of interest.

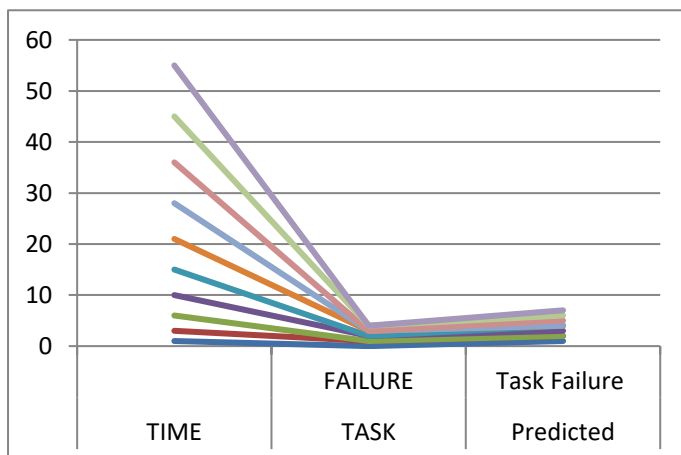


Figure 6. Comparative model in between task failure actual or task failure Predicted.

References

- Al-Dulaimy, A., Sicari, C., Papadopoulos, A. V., Galletta, A., Villari, M., & Ashjaei, M. (2022, September). Tolerancer: A fault tolerance approach for cloud manufacturing environments. *In 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1-8.
<https://doi.org/10.1109/ETFA52439.2022.9921606>
- Al Qassem, L. M., Stouraitis, T., Damiani, E., & Elfadel, I. A. M. (2023). Proactive random-forest autoscaler for microservice resource allocation. *IEEE Access*, *11*, 2570-2585.
<https://doi.org/10.1109/ACCESS.2023.3234021>
- AlOrbani, A., & Bauer, M. (2021). Load balancing and resource allocation in smart cities using reinforcement learning. *In 2021 IEEE International Smart Cities Conference (ISC2)*, pp. 1-7.
<https://doi.org/10.1109/ISC253183.2021.9562941>
- Bandari, V. (2020). Proactive Fault Tolerance Through Cloud Failure Prediction Using Machine Learning. *ResearchBerg Review of Science and Technology*, *3*(1), 51-65. Retrieved from <https://researchberg.com/index.php/rrst/article/view/54>
- Bessani, A., Sousa, J., & Alchieri, E. E. (2014, June). State machine replication for the masses with BFT-SMART. *In 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 355-362.
<https://doi.org/10.1109/DSN.2014.43>
- Bharany, S., Badotra, S., Sharma, S., Rani, S., Alazab, M., Jhaveri, R. H., & Gadekallu, T. R. (2022). Energy efficient fault tolerance techniques in green cloud computing: A systematic survey and taxonomy. *Sustainable Energy Technologies and Assessments*, *53*, 102613.
<https://doi.org/10.1016/j.seta.2022.102613>
- Chakrabarty, N., Kundu, T., Dandapat, S., Sarkar, A., & Kole, D. K. (2019). Flight arrival delay prediction using gradient boosting classifier. *In Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2018*, *2*, 651-659.
https://doi.org/10.1007/978-981-13-1498-8_57
- Dhingra, M., & Gupta, N. (2017). Comparative analysis of fault tolerance models and their challenges in cloud computing. *International Journal of Engineering & Technology*, *6*(2), 36-40.
<https://doi.org/10.14419/ijet.v6i2.7565>
- Eckart, B., Chen, X., He, X., & Scott, S. L. (2008).

- Failure prediction models for proactive fault tolerance within storage systems. In *2008 IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems*, pp. 1-8. <https://doi.org/10.1109/MASCOT.2008.4770560>
- Elnozahy, E. N., Alvisi, L., Wang, Y. M., & Johnson, D. B. (2002). A survey of rollback-recovery protocols in message-passing systems. *ACM Computing Surveys (CSUR)*, *34*(3), 375-408. <https://doi.org/10.1145/568522.56852>
- Fox, A., & Brewer, E. A. (1999, March). Harvest, yield, and scalable tolerant systems. In *Proceedings of the seventh workshop on hot topics in operating systems*, pp. 174-178. <https://doi.org/10.1109/HOTOS.1999.798396>
- Garg, S. (2022). Task resource usage of Google Cluster Usage Trace dataset [Data set]. Zenodo. <https://doi.org/10.5281/zenodo.6979672>
- Gossman, M. J., Nicolae, B., & Calhoun, J. C. (2024). Scalable I/O aggregation for asynchronous multi-level checkpointing. *Future Generation Computer Systems*, *160*, 420-432, ISSN 0167-739X. <https://doi.org/10.1016/j.future.2024.06.003>
- Gururaj, H. L., Flammini, F., Swathi, B. H., Nagaraj, N., & Ramesh, S. K. B. (2023a). Fault tolerance of network routers using machine learning techniques. In *Big Data Analytics and Intelligent Systems for Cyber Threat Intelligence*, pp. 253-274. River Publishers, eBook ISBN 9781003373384.
- Gururaj, H. L., Flammini, F., Swathi, B. H., Nagaraj, N., & Ramesh, S. K. B. (2023b). Machine Learning Techniques for Fault Tolerance Management. In *Computational Intelligence for Cybersecurity Management and Applications*, pp. 83-100. CRC Press, eBook ISBN 9781003319917.
- Haloi, R., & Chanda, D. (2024). Performance Analysis of KNN, Naïve Bayes, and Extreme Learning Machine Techniques on EEG Signals for Detection of Parkinson's Disease. *International Journal of Experimental Research and Review*, *43*(Spl Vol), 32-41. <https://doi.org/10.52756/ijerr.2024.v43spl.003>
- Hasan, D., & Zeebaree, S. R. (2024). Proactive Fault Tolerance in Distributed Cloud Systems: A Review of Predictive and Preventive Techniques. *Indonesian Journal of Computer Science*, *13*(2). <https://doi.org/10.33022/ijcs.v13i2.3808>
- Hien, P. T. (2023). Adaptive Fault Tolerance Mechanisms for Enhancing Service Reliability in Cloud Computing Environments. *Eigenpub Review of Science and Technology*, *7*(1), 252-265. Retrieved from <https://studies.eigenpub.com/index.php/erst/article/view/35>
- Kalaskar, C., & Thangam, S. (2023). Fault tolerance of cloud infrastructure with machine learning. *Cybernetics and Information Technologies*, *23*(4), 26-50. <https://doi.org/10.2478/cait-2023-0034>
- Karadayi, Y., Aydin, M. N., & Öğrenci, A. S. (2020). Unsupervised anomaly detection in multivariate spatio-temporal data using deep learning: early detection of COVID-19 outbreak in Italy. *IEEE Access*, *8*, 164155-164177. <https://doi.org/10.1109/ACCESS.2020.3022366>
- Khan, W., & Haroon, M. (2022). An efficient framework for anomaly detection in attributed social networks. *International Journal of Information Technology*, *14*(6), 3069-3076. <https://doi.org/10.1007/s41870-022-01044-2>
- Kirti, M., Maurya, A. K., & Yadav, R. S. (2024a). Fault-tolerance approaches for distributed and cloud computing environments: A systematic review, taxonomy and future directions. *Concurrency and Computation: Practice and Experience*, *36*(13), e8081. <https://doi.org/10.1002/cpe.8081>
- Kirti, M., Maurya, A. K., & Yadav, R. S. (2024b). A Fault-tolerant model for tuple space coordination in distributed environments. *Concurrency and Computation: Practice and Experience*, *36*(1), e7884. <https://doi.org/10.1002/cpe.7884>
- Kochhar, D., & Jabanjalin, H. (2017). An approach for fault tolerance in cloud computing using machine learning technique. *International Journal of Pure and Applied Mathematics*, *117*(22), 345-351. <https://api.semanticscholar.org/CorpusID:195063043>
- Kumar, A., Dutta, S., & Pranav, P. (2023). Supervised learning for Attack Detection in Cloud. *Int. J. Exp. Res. Rev.*, *31*(Spl Volume), 74-84. <https://doi.org/10.52756/10.52756/ijerr.2023.v31spl.008>
- Lan, Z., & Li, Y. (2008). Adaptive fault management of parallel applications for high-performance computing. *IEEE Transactions on Computers*, *57*(12), 1647-1660. <https://doi.org/10.1109/TC.2008.90>
- Lima, A. L. D. C. D., Aranha, V. M., Carvalho, C. J. D. L., & Nascimento, E. G. S. (2021). Smart predictive maintenance for high-performance computing systems: a literature review. The

- Journal of Supercomputing*, 77(11), 13494-13513.
<https://doi.org/10.1007/s11227-021-03811-7>
- Lu, L. T., Zhu, S. L., Wang, D. M., & Han, Y. Q. (2024). Distributed adaptive fault-tolerant control with prescribed performance for nonlinear multiagent systems. *Communications in Nonlinear Science and Numerical Simulation*, 138, 108222.
<https://doi.org/10.1016/j.cnsns.2024.108222>
- Mondal, S., Nag, A., Barman, A. K., & Karmakar, M. (2023). Machine Learning-based maternal health risk prediction model for IoMT framework. *International Journal of Experimental Research and Review*, 32, 145–159.
<https://doi.org/10.52756/ijerr.2023.v32.012>
- Mukwevho, M. A., & Celik, T. (2018). Toward a smart cloud: A review of fault-tolerance methods in cloud systems. *IEEE Transactions on Services Computing*, 14(2), 589-605.
<https://doi.org/10.1109/TSC.2018.2816644>
- Obadia, M., Bouet, M., Leguay J., Phemius K. and Iannone L. , (2014) Failover mechanisms for distributed SDN controllers, 2014 International Conference and Workshop on the Network of the Future (NOF), Paris, France, 2014, pp. 1-6.
<https://doi.org/10.1109/NOF.2014.7119795>
- Polze, A., Tröger, P., & Salfner, F. (2011, March). Timely virtual machine migration for pro-active fault tolerance. In *2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*, pp. 234-243.
<https://doi.org/10.1109/ISORCW.2011.42>
- Power, A., & Kotonya, G. (2018, June). A microservices architecture for reactive and proactive fault tolerance in IoT systems. In *2018 IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pp. 588-599.
<https://doi.org/10.1109/WoWMoM.2018.8449789>
- Pal, R., Pandey, M., Pal, S., & Yadav, D. (2023). Phishing Detection: A Hybrid Model with Feature Selection and Machine Learning Techniques. *Int. J. Exp. Res. Rev.*, 36, 99-108.
<https://doi.org/10.52756/ijerr.2023.v36.009>
- Ren, Y. (2021). Optimizing predictive maintenance with machine learning for reliability improvement. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part B: Mechanical Engineering*, 7(3), 030801.
<https://doi.org/10.1115/1.4049525>
- Seba, A. M., Gameda, K. A., & Ramulu, P. J. (2024). Prediction and classification of IoT sensor faults using hybrid deep learning model. *Discover Applied Sciences*, 6(1), 9.
<https://doi.org/10.1007/s42452-024-05633-7>
- Siddiqui, Z. A., & Haroon, M. (2023). Analysis of Challenges for Blockchain Adoption in Enterprise Distributed Applications. *International Journal on Recent and Innovation Trends in Computing and Communication*, 11(8s), 474–482.
<https://doi.org/10.17762/ijritcc.v11i8s.7228>
- Siddiqui, Z. A., & Haroon, M. (2024). Ranking of components for reliability estimation of CBSS: an application of entropy weight fuzzy comprehensive evaluation model. *International Journal of System Assurance Engineering and Management*, pp. 1-15. <https://doi.org/10.1007/s13198-024-02263-5>
- Sifat, M. M. H., & Das, S. K. (2024). Proactive and Reactive Maintenance Strategies for Self-Healing Digital Twin Islanded Microgrids Using Fuzzy Logic Controllers and Machine Learning Techniques. *IEEE Transactions on Power Systems*. <https://doi.org/10.1109/TPWRS.2024.3408096>
- Singh, D. P., & Singh, S. K. (2023). Precision fault prediction in motor bearings with feature selection and deep learning. *Int. J. Exp. Res. Rev*, 32, 398-407. <https://doi.org/10.52756/ijerr.2023.v32.035>
- Srivastava, S., Haroon, M., & Bajaj, A. (2013, September). Web document information extraction using class attribute approach. In *2013 4th International Conference on Computer and Communication Technology (ICCCT)*, pp. 17-22. <https://doi.org/10.1109/ICCCT.2013.6749596>
- Sun, S., Yao, W., & Li, X. (2018). DARS: A dynamic adaptive replica strategy under high load Cloud-P2P. *Future Generation Computer Systems*, 78, 31-40. <https://doi.org/10.1016/j.future.2017.07.046>
- Swarnalatha, K., Narisetty, N., Rao Kancharla, G., & Bobba, B. (2024). Analyzing Resampling Techniques for Addressing the Class Imbalance in NIDS using SVM with Random Forest Feature Selection. *International Journal of Experimental Research and Review*, 43(Spl Vol), 42–55. <https://doi.org/10.52756/ijerr.2024.v43spl.004>
- Tiwari, R. G., Haroon, M., Tripathi, M. M., Kumar, P., Agarwal, A. K., & Jain, V. (2024) A System Model of Fault Tolerance Technique in Distributed System and Scalable System Using Machine Learning. In *Software-Defined Network Frameworks*, pp. 1-16. CRC Press, eBook ISBN 9781003437482.
- Veer, A. S., & Bhardwaj, S. (2024, February). An

- Adaptive Storage Switching Algorithm for Fault-Tolerant Network Attached Storage systems. In *2024 2nd International Conference on Computer, Communication and Control (IC4)*, pp. 1-7. <https://doi.org/10.1109/IC457434.2024.10486061>
- Venkataraman, N. (2023). Proactive fault prediction of fog devices using LSTM-CRP conceptual framework for IoT applications. *Sensors*, 23(6), 2913. <https://doi.org/10.3390/s23062913>
- Yadav, P., Bhargava, C. P., Gupta, D., Kumari, J., Acharya, A., & Dubey, M. (2024). Breast Cancer Disease Prediction Using Random Forest Regression and Gradient Boosting Regression. *International Journal of Experimental Research and Review*, 38, 132–146. <https://doi.org/10.52756/ijerr.2024.v38.012>
- Yang, Y., Mei, J., Zhang, Z., Long, Y., Liu, A., Gao, Z., & Rui, L. (2023). Lightweight Fault Prediction Method for Edge Networks. *IEEE Internet of Things Journal*. <https://doi.org/10.1109/JIOT.2023.3333293>
- Zou, Y., Yang, L., Jing, G., Zhang, R., Xie, Z., Li, H., & Yu, D. (2024). A survey of fault tolerant consensus in wireless networks. *High-Confidence Computing*, 4(2), 100202. <https://doi.org/10.1016/j.hcc.2024.100202>

How to cite this Article:

Mohd Haroon, Zeeshan Ali Siddiqui, Mohammad Husain, Arshad Ali, and Tameem Ahmad (2024). A Proactive Approach to Fault Tolerance Using Predictive Machine Learning Models in Distributed Systems. *International Journal of Experimental Research and Review*, 44, 208-220.

DOI : <https://doi.org/10.52756/ijerr.2024.v44spl.018>



This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.