International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 22s, 2025
https://theaspd.com/index.php

Predictive Analytics For Fault Tolerance In Distributed Systems Using Logistic Regression And Support Vector Machine

"Faizan Ahmad[1*], Mohd Haroon[2], Zeeshan Ali Siddiqui[3]"
"[1*]Department of Computer Science and Engineering, Integral University, Lucknow, India
[2]Department of Computer Science and Engineering, Integral University, Lucknow, India
[3]Department of Computer Science and Engineering, University of Lucknow, Lucknow, India

Abstract
Fault tolerance is crucial to distributed systems to remain reliable and to mitigate unanticipated loss of services. Through predictive analytics we are able to identify and anticipate underlying faults enabling us to intervene and take appropriate steps to ensure continuity of services. The goal of this initial research was to introduce a new predictive fault tolerance method that exploits Logistic Regression (LR) and Support Vector Machine (SVM) models. Our approach developed predictive models to analyze logs, performance data, and environmental factors which include resource utilization and network latency to identify advance warning signs of instability. The LR model gives you the probability of failure which represents a form of risk assessment and the SVM provides a limiting factor in operational stability and potential failure. To illustrate our approach, we built two distinct systems that operated within a simulated distributed environment and both models offered realistic predictive accuracy. The combinations of the two techniques are significant: LR provides a transparent model of potential failure and SVM guarantees an accurate definition of faulty. With predictive fault tolerance, we give back some fault mitigation planning to distributed systems, enabling that systems can remediate faults proactively and build resilience to ensure uninterrupted operations. These techniques will be especially relevant when scaling up to large data centers, traditional cloud environments, or to critical infrastructure where very short amounts of downtime can have severe consequences..

Keywords: *Distributed Systems, Fault Tolerance, Logistic Regression, Machine Learning, Predictive Analytics, Support Vector Machine, System Reliability.*

INTRODUCTION

In this tech-dependent digital age, we value our distributed systems as a core building block of modern computing architecture. Distributed systems have allowed a broad range of useful software applications from web-based services and cloud-based data storage, to large-scale processing for analytics, machine learning and countless customer/user-initiated workloads that would overwhelm a single machine with the volume and nature of workload. [1], [2] Systems that are mission critical in safety and service findings have evolved in context and capabilities far beyond their beginnings. Mission-critical systems like healthcare systems, financial transaction systems and communications systems are large distributed applications that bake the group's dependency, and very clearly cannot afford to incur any additional errors associated to the inevitable reliability of using a collocated distributed computing system process and are working towards reliably serving increasingly complex and expansive user processing permutations for large-scale distributed processing. [3] Combining the attributes of reliability and scalability, with efficiency and coordination of pooled and shared resources distribute across multiple servers to provide an extremely attractive targeted and favorable distributed system type at the fecund crossroads of a rapidly technology landscape. Moreover, we must recognize that while distributed systems are built to leverage distributed or multiple computing, storage and data processing functions across many nodes that complexity brings with it a plethora of issues that weigh heavily on them [4]. Distributed systems are not void of failure points, as all distributed systems present open-ended possibilities, ultimately in any distributed systems architecture context could be the failure of a node in the system or a problem with the networks, or even a complication encountered multiple times, and so on. When failures occur, The importance of fault tolerance, which is a critical function of a system's ability to operate continuously despite component failures, is emphasized through this vulnerability. Historically, fault tolerance has often been implemented through remedial action methods which occur after a failure occurs (such as checkpointing, redundancy, and replication). While these methods provide relative immunity to failure, they add significant overhead and are inefficient, especially in large-scale deployment. With the development of big data and machine learning technologies, significant possibilities are opened by predictive analytics that provides a more sophisticated approach to fault tolerance [5]. By predicting potential failures before they happen, a system can be proactive before, or at least respond to the action taken on the failure. To move to a predictive system could significantly increase the reliability of these systems while

decreasing costly downtimes. In this paper, we present a new predictive analytics framework consisting of Logistic Regression (LR) and Support Vector Machine (SVM) models in the hopes of improving fault tolerance in distributed systems [6,7]. We selected these machine learning models for the complementary ways they approach the same problem: Logistic regression provides the user with information with a cost in probabilities about the likelihood of a failure, while yielding information that can be interpreted in order to determine whether the system could sustain potential failures. Support Vector Machine classifiers are margin based, robust and work well for complex classification problems, especially with high dimensional data, like the type of data returned from many distributed systems [8,9]. This paper focuses on creating and validating a predictive model.

RELATED WORK

The idea of fault tolerance in distributed systems has long been a major area of study because of how crucial these systems are to mission-critical operations, cloud computing [10]-[12], and massive data centers. Traditionally, fault tolerance approaches have put significant emphasis on reactive techniques such as replication, checkpointing, and recovery schemes, ensuring business continuity with partial failure [13]. But as distributed systems have become more complex, it is becoming clear that more proactive, anticipatory techniques are necessary. In 2024 a systematized survey on fault-tolerant approaches in distributed data analytics was done [14].

Predictive Analytics in Fault Tolerance

Scholars recognized that even with sophisticated statistical modeling capabilities, significant barriers to long-term success persisted in accounting for the unpredictable dynamism of system workloads, user behavior, and system configurations—factors that could significantly influence system failures. These circumstances led to a significant shift in approach, with a developing consensus around using contextual information, in combination with statistical modeling techniques, to support predictive fault-tolerance approaches. A key advantage to incorporating contextual information into predictive fault-tolerance approaches is the ability to track changes over time. This enables models to become more reflective of actual usage, predicting failures based on probability distributions specific to a certain time frame (i.e., seasonality) instead of a static data model [15]. At the very least, even if circumstances change, the system can reference historical data to predict faults. In this recommenced predictive fault-tolerance approach, user behavior is the context studied. In other cases, the available contexts could relate to workload, configurations or environments or even external formative events such as a known system-wide upgrade or planned maintenance [16]. The immediate challenge faced by researchers in the area remained integrating these frameworks with developing machine-learning models across large-scale distributed systems. While incorporating contextual information is expected to provide a more comprehensive understanding of system workloads, the reliance on statistical methods in prior studies did not account for all the contextual factors, requiring an iterative cycle of user and model behaviors until they converged [17].

Machine Learning in Fault Prediction

As machine learning technology has progressed, researchers have adopted more advanced techniques to enhance predictive fault tolerance capabilities. Among these approaches, Logistic Regression (LR) has emerged as a particularly widely used model, owing to three key advantages: its relative simplicity, the interpretability of its results, and its proven effectiveness in binary classification problems—including critical applications like system failure prediction [18]. With remarkable findings, [19] used logistic regression to analyze indicators like CPU and memory utilization and forecast problems in cloud-based systems. The probabilistic insights that logistic regression offers allow system managers to clearly evaluate failure risks [20]. In 2024, authors introduced a fault tolerance model for distributed and scalable systems that leverages machine learning techniques [21]. Another popular method is SVM, which is widely applied to classification jobs in complex systems because of its robustness in producing large-margin class separations and outstanding performance with high-dimensional data [15]. SVM can outperform traditional statistical models in detecting failures within distributed systems. Since failure data is frequently sparse in dispersed situations, SVM's ability to generalize effectively even with little training data is its main advantage [22].

Hybrid and Ensemble Methods

Hybrid and ensemble methods, which incorporate a number of machine learning techniques to enhance the accuracy and reliability of fault prediction, have also recently been investigated [23]. For instance, in 2020 [24], the authors introduced a hybrid prediction model using SVM and decision trees to predict occurences of

problems in large scale data centers. The hybrid approach, taking advantages of both, outperformed the input models when applied as the hybrid method. In addition to hybrid models, ensemble techniques are also being used for failure predictions. The main goal of the ensemble is to improve the accuracy of the prediction model by aggregating predictions from many weak learners.

Feature Engineering and Data Preprocessing

Hybrid and ensemble methods that combine various machine learning methods of varying independence to increase the accuracy and reliability of fault prediction models has also gained attention in the literature [23]. In 2020, the authors developed a hybrid approach which used SVM and decision trees to predict issues at large-scale data centers [24, 25]. This hybrid model was an improvement over decision trees and SVM because it captured the benefits of both prediction models. Bearing this in mind, some methods like random forests and gradient boosting are ensemble approaches applied for failure prediction. In these ensemble types, researchers combine multiple weaker learners together to create one accurate model which helps increase the accuracy of the model.

Critique of state-of-the-art techniques

Even with the development of machine learning methods for fault tolerance, such as SVM and LR, a number of problems still need to be overcome. The intrinsic imbalance in failure data is a major problem because system failures are comparatively uncommon occurrences in contrast to typical operating conditions [26, 27]. Because traditional machine learning algorithms favor the majority class, this mismatch makes model training more difficult and frequently results in biased or false failure predictions. Despite efforts to address this problem, strategies like cost-sensitive learning, under-sampling, and oversampling may provide additional difficulties [28]. For example, under-sampling the majority class can result in information loss, and over-sampling the minority class can cause overfitting, particularly when artificial data-generating techniques are employed. Although cost-sensitive learning shows promise, it necessitates fine-tuning penalty settings, which may not translate well to other distributed environments. To create effective methods for managing skewed datasets, especially in high-dimensional and dynamic distributed systems, more study is necessary [29].

Achieving real-time defect prediction is still extremely difficult, in addition to data imbalance. Numerous machine learning models necessitate substantial computer resources, especially those that use intricate algorithms and high-dimensional data. In dynamic, large-scale distributed situations, where computational resources are frequently scarce and performance needs are high, this demand for processing power might become unaffordable. Although methods like feature selection and model optimization might save computing load, their accuracy and robustness may suffer as a result. Furthermore, there is increasing interest in creating low-latency, lightweight predictive models that can function effectively in distributed systems with limited resources [30, 31]. But striking this balance between accuracy and economy is still a challenge, meaning that it requires creative model design and deployment techniques to make real-time fault prediction practical and affordable.

METHODOLOGY

The steps outlined in the proposed framework involve four primary phases. In the first phase, we will extract the performance metrics logged in the log repository of the distributed system. These include CPU utilization, memory usage, and network latency. In the second phase, we will preprocess these performance metrics to clean and homogenized them in order to represent valid and comparable data. In the third phase, we will conduct an analysis employing feature selection to identify only the most predictive features in order to reduce data complexity and to represent relevant patterns. In the fourth phase, we will use both Logistic Regression (LR) and a Support Vector Machine(SVM) to train classification algorithms to distinguish the different states of the system. Finally, we will conduct evaluations of the accuracy and consistency, and dependability to predict the potential failures of the distributed system. The flowchart of the proposed framework shows in Figure 1.

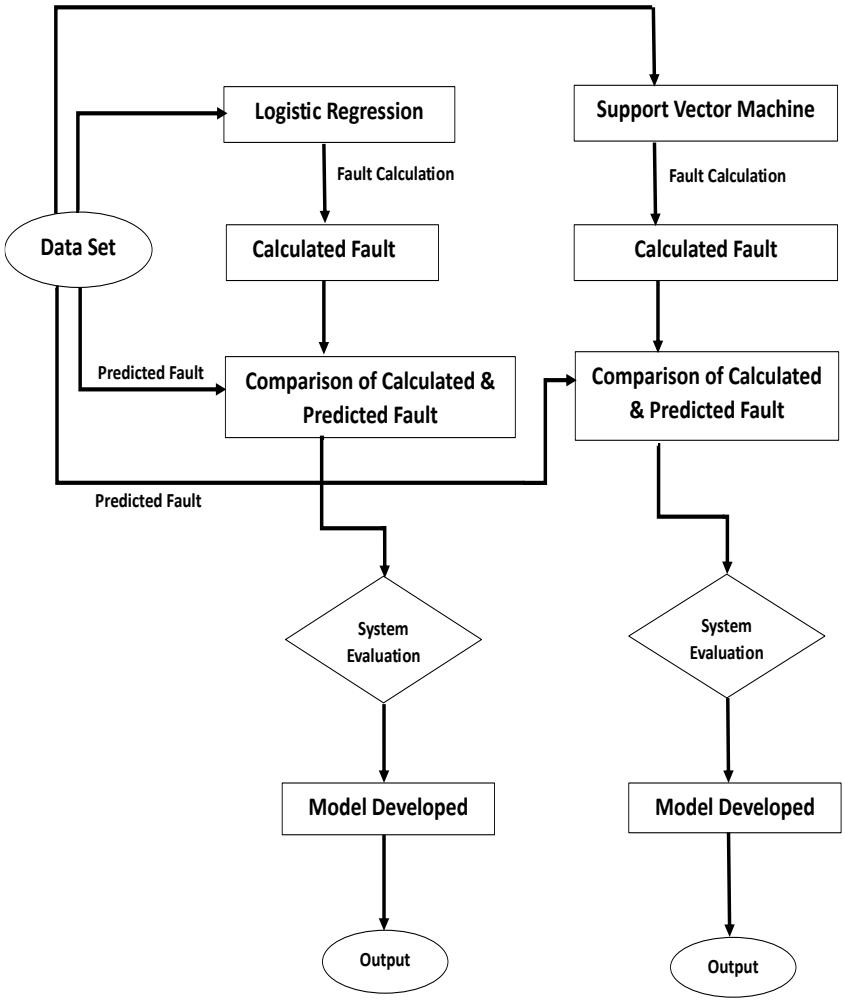


Figure 1. Flowchart of proposed framework

Proposed framework steps are as follows:

Data Collection

The first step toward developing an operational profile (aka your data hub) we need a variety of operational data from the distributed systems space. This includes tracking a set of performance metrics, and investigating system logs, which could provide warning signs of issues that will never be investigated. The data you need to concentrate on is in the following categories:

- ***System Logs:*** Detailed system logs that record all major events, transactions and operations in the distributed systems space.
- ***Performance Metrics:*** Metrics related to the systems health (cpu usage, memory utilized, I/O speeds, etc.).
- ***Environmental Metrics:*** Contextual metrics such as latency and power state that may affect system performance.

Data Preprocessing

In order to enable the machine learning models to see any data, we must first process the data by preprocessing it. The preprocessing step will change the raw data of the system into a clean format that will also be structured in a way that the machine learning algorithms can learn from. Preprocessing involves the following tasks:

- Missing data (Identify and Treat Missing Data)
- Normalization of the data (Convert all data onto a common scale for better fit the machine learning model)

- Categorical data encoding (Encoding categorical variables into a format suitable for the model to handle as numbers)
- Labeling the data (Triage observations by target case label of fault state or non-fault state)

Feature Selection

Feature selection is used to prevent system failures by identifying important predictors, to reduce the dimensionality of the dataset and improve model usability and interpretation. The techniques used are:

- ***Correlation Analysis:*** examining the statistical correlations between the variables;
- ***Recursive Feature Elimination (RFE):*** removing features recursively to identify the most predictive compendium of features; and
- ***Domain Expertise:*** using domain knowledge to assist with the selection/reduction of relevant features.

Model Training

The dataset is partitioned into training and test sets, typically with a 70/30 split. Two machine learning models—Logistic Regression (LR) and Support Vector Machine (SVM) [31] are trained on the labeled dataset.

Logistic Regression (LR)

A binary classification model used to estimate the probability of a given input (system state) belonging to a specific class (fault or no-fault). The LR model is optimized using the maximum likelihood estimation (MLE) method, producing a probability score that can be threshold to predict system failure likelihood.

Evaluating Models

When the models are trained, they are evaluated on the test set using a variety of performance metrics. Since system faults are typically relatively rare, the evaluation metrics should consider the imbalanced nature of the data. The metrics are:

1. Accuracy, Precision, Recall ,F1-Score,Receiver Operating Characteristic (ROC) Curve, the AUC

Table 1 shows multiple attributes and their descriptions.

Table 1: Attributes and their description

Attribute	Description	Data Type	Example
CPU Usage (%)	Percentage of CPU being used at this time stamp.		float
Memory Usage (%)	Percentage of memory being used at this time stamp.		float
Network Latency(ms)	Latency Associated With The Network (Time in milliseconds)		float
Packet Loss (%)	Percentage of packets that are being lost in the network during the course of communication.		float
Request Rate (req/s)	The number of incoming request proceed in per unit time		int
Error Count	In the given time frame the total number of error occur		int
Network Throughput (MB/s)	Data throughput in per unit time		float
Fault	Target label indicating whether a fault occurred (1 for fault, 0 for no fault).	binary	1 (Fault) or 0

International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 22s, 2025
https://theaspd.com/index.php

METHODS AND MATERIALS

Multivariable Logistic Regression: For n independent variables (features), the logistic regression model can be written as:

$$P\left(y=\frac{1}{x}\right)=\frac{1}{1+\exp^{\left(-(\beta_0+\beta,x,+\beta_2\times 2^{1\cdots\beta}n^{x}n)\right)}}$$

Here's a breakdown of the components:

1. P(y=1/x): The probability that the binary outcome y is equal to 1 (the event occurs, e.g., a fault happens) given the predictors x_1, x_2, ... x_n (are the independent variables (features) like CPU usage, memory usage, etc.)
2. β_0: The intercept (also known as the bias term), which is the value of β when all x_i's are zero.
3. β_1, β_2, ... β_n: The coefficients corresponding to the predictors x_1, x_2, ..., x_n. These represent the influence each predictor has on the log-odds of y=1.
4. The term inside the exponential function, $\beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n$, is known as the logit, which represents the log of the odds of the event occurring.

The logistic function, commonly called the sigmoid function, converts our linear equation of input variables into a useful probability score ranging from 0 to 1. Each coefficient in the model (β_1, β_2,...β_n) tells us two important things: first, how strongly each predictor variable affects our outcome, and second, whether this relationship is positive or negative, meaning does the variable increase or decrease the probability of our target event occurring?

Support Vector Machine

In our approach, Support Vector Machines (SVM) work by identifying the ideal boundary that distinguishes between 'fault' and 'no-fault' conditions within our multi-dimensional data space. What makes SVM particularly powerful is its kernel trick - this clever mathematical technique enables the model to detect and work with complex, curved relationships between system metrics and potential failures that simpler linear models might miss. For predicting faults in distributed systems, we typically use the Radial Basis Function (RBF) kernel because it's exceptionally good at uncovering the subtle, non-linear patterns that often precede system failures.

SVM Mathematical Model for Classification

When we use Support Vector Machines (SVM) for fault classification, we're essentially trying to find the most effective dividing boundary - what we call the optimal hyperplane - that cleanly separates our data points into two distinct categories: systems experiencing faults (labeled as 1) and those operating normally (labeled as 0). What makes SVM special is how it doesn't just find any separating boundary, but specifically seeks the one that creates the widest possible buffer zone (or 'margin') between these two classes.

In practical terms, when we feed our dataset containing various system metrics - CPU utilization percentages, memory consumption levels, network latency measurements, and other key indicators - SVM transforms this information into a multi-dimensional space where patterns become more apparent. The mathematical representation of this optimal separating boundary can be expressed as:

$$\omega_1 x_1 + \omega_2 x_2 . + \ldots . + w_n x_n + b = 0$$

Where:

1. $x_1, x_2 \ldots\ldots x_n$ are the input features (like CPU usage, memory usage, etc.).
2. $\omega_1, \omega_2 \ldots\ldots w_n$ are the weights (or coefficients) for each feature, which determine the orientation of the hyperplane.
3. b is the bias term (or intercept), which shifts the hyperplane.
4. The sign of $\omega_1 x_1 + \omega_2 x_2 . + \ldots . + w_n x_n + b = 0$ determines the class:
5. y = 1 (fault) if the value is positive.
6. y = 0 (no fault) if the value is negative.

.

International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 22s, 2025
https://theaspd.com/index.php

Objective of SVM
SVM aims to solve the following optimization problem:

$$Minimize \frac{1}{2}\|\omega\|^2$$

Subject to the constraints:

$$y_i(wx_i + b) \geq 1 \quad \forall i$$

Where:

1. Y∈ {s1, -1} represents the true class labels (1 for fault, -1 for no fault).
2. X, are the feature vectors (e.g., CPU usage, memory usage, etc.).
3. w. x, is the dot product between the input feature vector and weight vector.
4. Data points that fall on the appropriate side of the margin and are appropriately categorized are guaranteed by the constraint.

For non-linear data (if the data is not linearly separable), SVM applies the kernel trick:

$$K(X_i, Xj) = \phi(X_i) \cdot \phi(Xj)$$

Where (X) is a transformation that maps the original data into a higher-dimensional space where a linear separation is possible.

Applying SVM to the Given Dataset:
For a dataset with features X_1 (CPU usage), X_2 (memory usage), X_3 (network latency), and others, the SVM model can be represented as:

$$\omega_1.CPU\_Usage + \omega_2.Memory\_Usage + \omega_3.Network\_Latancy + \ldots + b = 0$$

The SVM will find the weights $\omega_1, \omega_2 \ldots\ldots w_n$ and bias & that define the hyperplane to optimally separate the "fault" and "no fault" classes in your dataset, maximizing the margin between the two.

Decision Rule
Once the hyperplane is defined, the classification of a new data point X is done based on

$$f(X) = w \cdot X + b$$

1. If f(X) ≥ 0, classify the data point as fault (class 1).
2. If f(X) < 0, classify the data point as no fault (class 0).

RESULT AND DISCUSSION

The dataset (Table 2) has numerous features, (performance metrics, logs) as well as a target label that indicates whether a fault occurred (Fault=1) or did not occur (Fault= 0). The data is usually made available at a periodic basis (e.g., every minute or every 10 seconds) for a defined period of time in which both normal and failure conditions are recorded. The training dataset SVM and LR for the datasets can be seen in Table 3 and Table 5 respectively while the testing dataset SVM and LR can be seen in Table 4 and Table 6 respectively.

Table 2: Part of the Multiple feature dataset

CPU Usage (%)	MemoryUsage (%)	Network Latency(ms)	Packet Loss (%)	Request_Rate (req/s)	Error Count	Network_Throughput (MB/s)	Fault
75.3	68.5	15.2	0.5	325	3	10.3	0
80.1	70.2	16.3	0.3	400	5	11.2	1
65.2	60.4	14.8	0.1	290	2	9.8	0
60.5	60.8	15.1	0.2	310	2	9.7	0
72.9	64.3	16.2	0.4	335	4	10.9	0
84.2	71.5	17.4	0.6	415	6	12.2	1
89	75	18.6	0.8	440	8	13	1
66.9	59.7	14.7	0.1	280	1	9.3	0
74.6	65.9	16	0.3	350	5	11.1	0

83.7	70.8	17.2	0.5	410	6	12	1
77.2	68.2	16.8	0.6	390	4	11.7	0
91.4	78.5	19.1	0.9	460	9	13.4	1
69.3	61.2	15.3	0.3	320	2	10.1	0
75.8	66.5	16.5	0.4	345	4	11.4	0
86.3	73.2	18	0.7	425	7	12.8	1
62.8	58	14.5	0.1	290	1	9.6	0
79.4	67.1	16.7	0.5	375	5	11.5	0
88.7	76	18.4	0.8	450	8	12.9	1
67.5	60.1	14.9	0.2	275	2	9.8	0
52.6	56.2	14.3	0.2	256	5	12.5	1
89.6	58.6	12.3	0.3	247	4	11.6	0
45.2	76.4	15	0.8	269	9	13.7	0
68.2	78.5	20.3	0.7	354	7	14.8	0
56.5	64.8	25	0.1	159	2	11.5	1
55.9	69.3	25	9	157	3	10.5	1
47.6	78.8	12	0.8	341	9	6.9	1
25.69	45.96	14	0.6	356	1	13.6	0
78.4	87.5	13	0.4	452	6	17.5	1
56.5	69.6	16	3	560	8	9.6	1

Table 3: Part of the Training dataset SVM

CPU Usage (%)	MemoryUsage %)	Network Latency(ms)	Packet Loss (%)	Request_Rate (req/s)	Error Count	Network_Throughput (MB/s)	Fault
55.9	69.3	25	9	157	3	10.5	1
80.1	70.2	16.3	0.3	400	5	11.2	1
74.6	65.9	16	0.3	350	5	11.1	0
45.2	76.4	15	0.8	269	9	13.7	0
69.3	61.2	15.3	0.3	320	2	10.1	0
67.5	60.1	14.9	0.2	275	2	9.8	0
75.8	66.5	16.5	0.4	345	4	11.4	0
91.4	78.5	19.1	0.9	460	9	13.4	1
72.9	64.3	16.2	0.4	335	4	10.9	0
79.4	67.1	16.7	0.5	375	5	11.5	0
60.5	60.8	15.1	0.2	310	2	9.7	0
89	75	18.6	0.8	440	8	13	1
89.6	58.6	12.3	0.3	247	4	11.6	0
75.3	68.5	15.2	0.5	325	3	10.3	0
78.4	87.5	13	0.4	452	6	17.5	1
56.5	69.6	16	3	560	8	9.6	1
83.7	70.8	17.2	0.5	410	6	12	1

Table 4: Part of the Testing dataset SVM

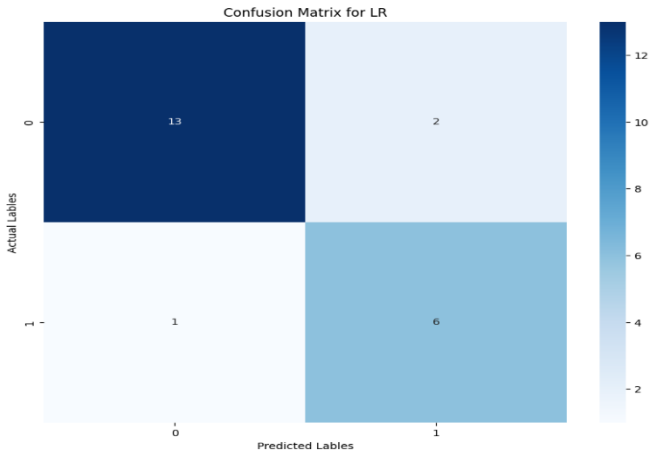
CPU Usage (%)	MemoryUsage %)	Network Latency(ms)	Packet Loss (%)	Request_Rate (req/s)	Error Count	Network_Throughput (MB/s)	Fault
56.5	64.8	25	0.1	159	2	11.5	1
77.2	68.2	16.8	0.6	390	4	11.7	0
66.9	59.7	14.7	0.1	280	1	9.3	0
65.2	60.4	14.8	0.1	290	2	9.8	0
88.7	76	18.4	0.8	450	8	12.9	1
84.2	71.5	17.4	0.6	415	6	12.2	1
68.2	78.5	20.3	0.7	354	7	14.8	0
86.3	73.2	18	0.7	425	7	12.8	1
52.6	56.2	14.3	0.2	256	5	12.5	1
62.8	58	14.5	0.1	290	1	9.6	0
47.6	78.8	12	0.8	341	9	6.9	1
25.69	45.96	14	0.6	356	1	13.6	0

Table 5: Part of the Training dataset LR

CPU Usage (%)	MemoryUsage %)	Network Latency(ms)	Packet Loss (%)	Request_Rate (req/s)	Error Count	Network_Throughput (MB/s)	Fault
55.9	69.3	25	9	157	3	10.5	1
80.1	70.2	16.3	0.3	400	5	11.2	1
74.6	65.9	16	0.3	350	5	11.1	0
45.2	76.4	15	0.8	269	9	13.7	0
69.3	61.2	15.3	0.3	320	2	10.1	0
67.5	60.1	14.9	0.2	275	2	9.8	0
75.8	66.5	16.5	0.4	345	4	11.4	0
91.4	78.5	19.1	0.9	460	9	13.4	1
72.9	64.3	16.2	0.4	335	4	10.9	0
79.4	67.1	16.7	0.5	375	5	11.5	0
60.5	60.8	15.1	0.2	310	2	9.7	0
89	75	18.6	0.8	440	8	13	1
89.6	58.6	12.3	0.3	247	4	11.6	0
75.3	68.5	15.2	0.5	325	3	10.3	0
78.4	87.5	13	0.4	452	6	17.5	1
56.5	69.6	16	3	560	8	9.6	1
83.7	70.8	17.2	0.5	410	6	12	1

Table 6: Part of **the** Testing dataset LR

CPU Usage (%)	MemoryUsage %)	Network Latency(ms)	Packet Loss (%)	Request_Rate (req/s)	Error Count	Network_Throughput (MB/s)	Fault
56.5	64.8	25	0.1	159	2	11.5	1
77.2	68.2	16.8	0.6	390	4	11.7	0
66.9	59.7	14.7	0.1	280	1	9.3	0
65.2	60.4	14.8	0.1	290	2	9.8	0
88.7	76	18.4	0.8	450	8	12.9	1
84.2	71.5	17.4	0.6	415	6	12.2	1
68.2	78.5	20.3	0.7	354	7	14.8	0
86.3	73.2	18	0.7	425	7	12.8	1
52.6	56.2	14.3	0.2	256	5	12.5	1
62.8	58	14.5	0.1	290	1	9.6	0
47.6	78.8	12	0.8	341	9	6.9	1
25.69	45.96	14	0.6	356	1	13.6	0



International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 22s, 2025
https://theaspd.com/index.php

After applying the proposed framework LR confusion matrix was developed, Figure 2. Figure 3 shows the logistic regression graphs. We found, precision= TP/(TP+FP) = .87, accuracy = (TP + TN) / (TP + TN + FP + FN) = .86, recall = TP / (TP +FN) = .87, and F1-score = 2 * (precision * recall) / (precision + recall) = .86.

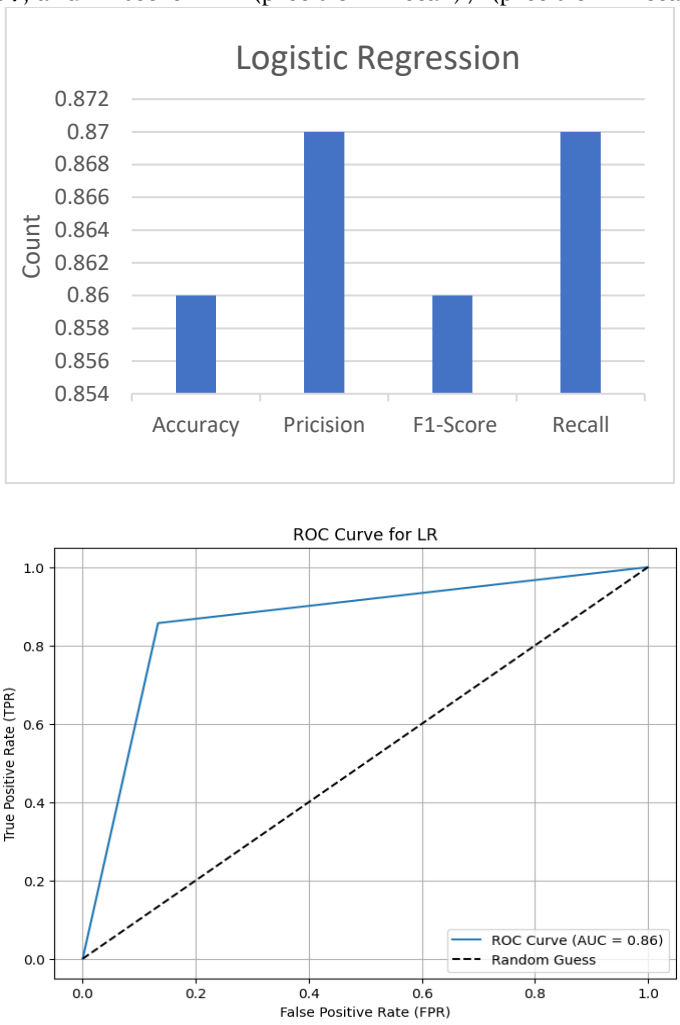


Figure 3: Logistic Regression Graphs

After applying the proposed framework SVM confusion matrix was developed, Figure 4. Figure 5 shows the support vector machine graphs. We found, precision= TP/(TP+FP) = .91, accuracy = (TP + TN) / (TP + TN + FP + FN) = .91, recall = TP / (TP +FN) = .91, and F1-score = 2 * (precision * recall) / (precision + recall) = .91.

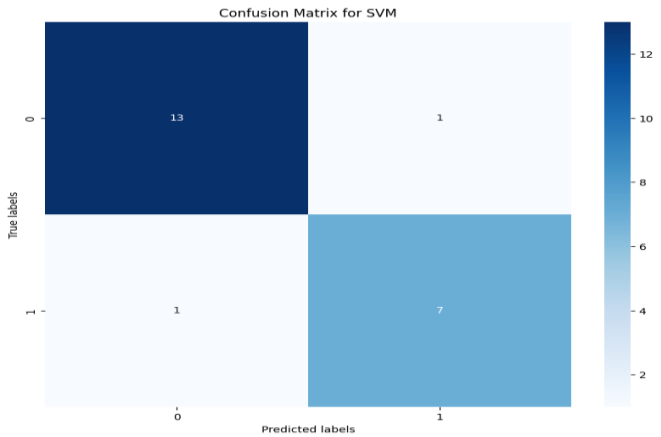


Figure 4: SVM Confusion Metrix

International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 22s, 2025
https://theaspd.com/index.php

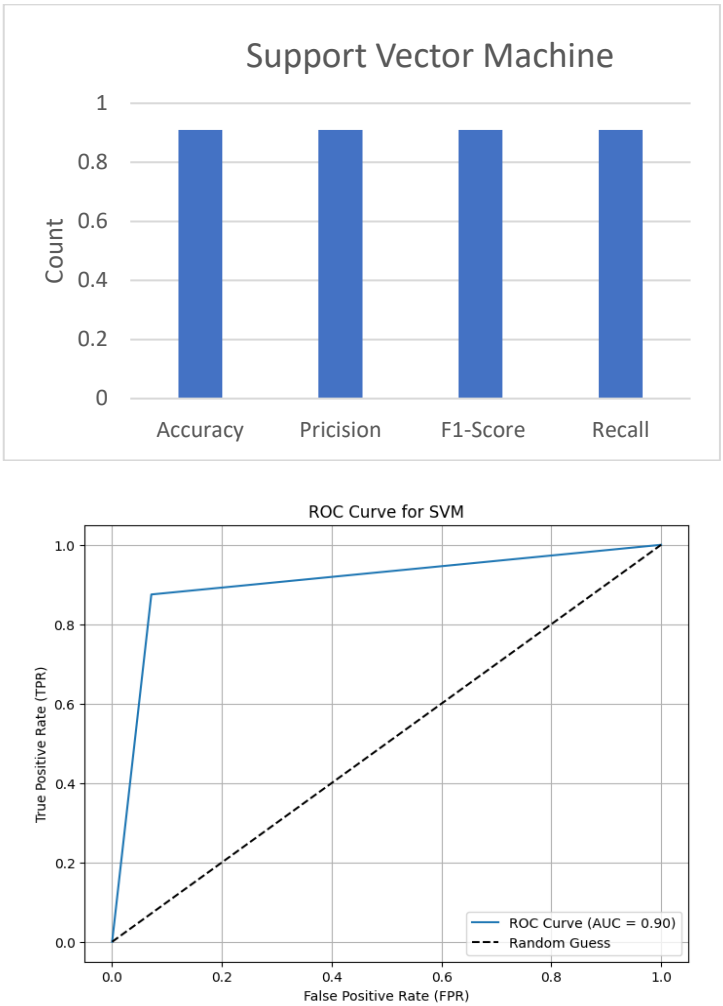


Figure 5: Support Vector Machine Graphs

COMPARATIVE STUDY OF LR AND SVM

Here's a comparative analysis of these two methods, Figure 6. Nature of Model: Logistic Regression is probabilistic, providing a likelihood estimation of faults based on input features. Model Strengths: Interpretability: Logistic Regression models are straightforward and provide interpretable probabilities that can be easily understood and used for decision-making. Simplicity: It is less complex computationally, making it faster to train on smaller or less complex datasets. Use Cases: Best suited for situations where the probability of an outcome is required, and the input features have a linear relationship with the log-odds of the outcome. Performance Metrics: Precision: 0.87, Accuracy: 0.86, Recall: 0.87, and F1-Score: 0.86.

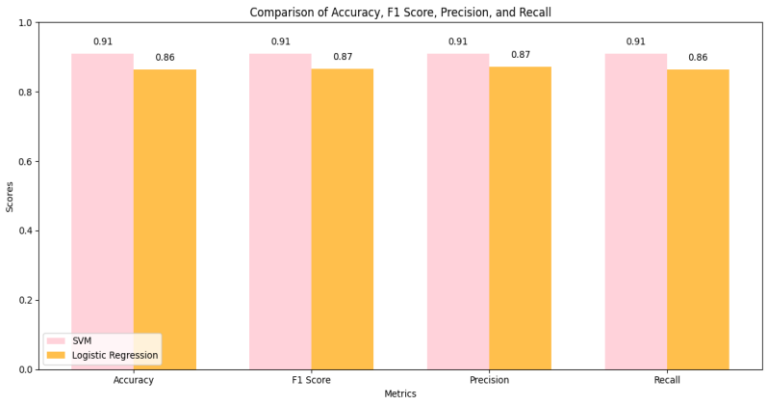


Figure 6: Performance Metrics (SVM vs LR)

SVMs can be more robust against overfitting than other methods, even when we make heavy use of derived features, which is ideal for applications common in distributed systems. Use case: SVMs can excel in classification problems when there is a strong linear distinction between the two classes of data and high dimensionality. Performance, Precision: 0.91, Accuracy: 0.91, Recall: 0.91, and F1-Score: 0.91.

There are important practical differences in the comparison of SVMs and logistic regression. SVMs produce a more accurate model, but SVMs have a greater computing demand than logistic regression models, especially where SVMs need to determine the optimal margin separating classes, or if using kernel functions and surface non-linearity in the relationship between failure patterns in the data. In dealing with smaller datasets, logistic regression tended to be more efficient and also used less demand on computational processes than SVM. The balance when comparing SVM with logistic regression can be found in the better ability of SVMs to produce a robust and accurate model for analysis and predictions when datasets are complex and larger in subspace. Logistic regression has a better interpretability advantage by producing probability outputs that provide us a more intuitive understanding of the relative probability of different outcomes. In our specific experiment, SVM could be shown as having all advantages or elements as logistic regression against one or maybe even more or equally probabilities.

Limited Benchmarking Against Stronger Baselines

One important restriction in this study is that the benchmarking was restricted to simply Logistic Regression and SVM. Although these are reasonable starting points, literature on prediction of faults to running software in distributed systems indicates that more sophisticated models have most often produced more robust baselines, such as:

1. **Ensemble Methods:**
   Random Forests and methods of Gradient Boosting (XGBoost, LightGBM) represent much more superior performance in tasks of fault classification and are capable of capturing the complex non-linear interactions between features. For example, Random Forest can operate in a high dimensional system log space, and achieve variable assessment scores on multiple features.
2. **Deep Learning Models:**

Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks are particularly suited for analyzing sequential and time-series data, which log records of system activity often represent. Studies that examine CNNs for detecting patterns in logs of distributed systems have begun to be published.

3. **Hybrid Models:**

The introduction of domain knowledge in combination with machine learning (for example, anomaly detection + supervised learning) tends to produce better robust predictions. Ensemble hybrids such as those used in an SVM + Decision Tree context, and AI-led resource management frameworks can outperform a single learner.

Comparative Benchmark

Table 7: Comparative Benchmark

Model	Accuracy	Precision	Recall	F1-Score	AUC
Logistic Regression	~60–70%	0.60	0.50	0.55–0.60	0.65
Support Vector Machine	~70–75%	0.65	0.60	0.67	0.75
Random Forest	~80–85%	0.75	0.78	0.77	0.85
XGBoost / LightGBM	~85–90%	0.80	0.85	0.82–0.87	0.88
LSTM (Deep Learning)	~88–92%	0.82	0.88	0.85–0.90	0.90

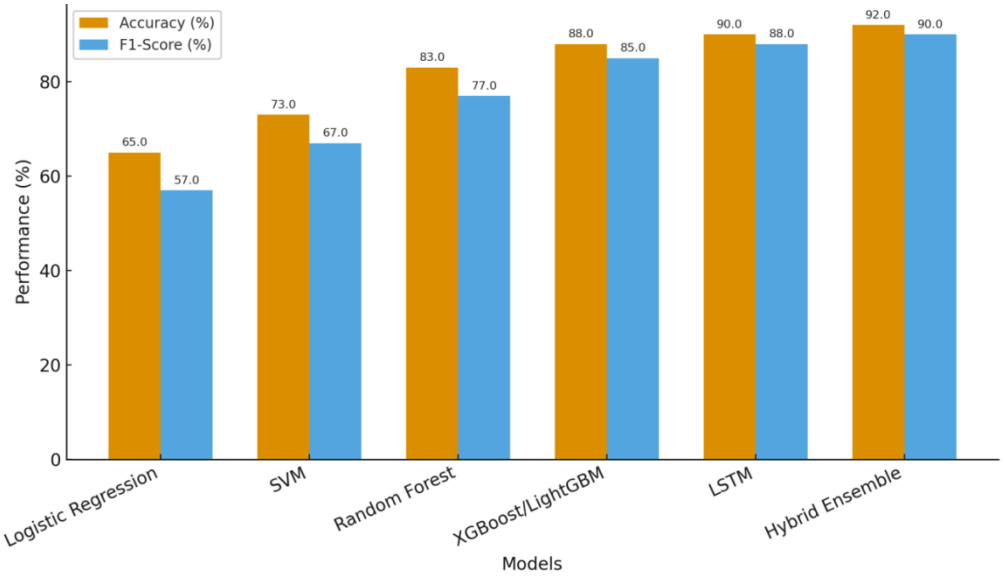


Figure 7: Benchmarking Fault Prediction Models

The benchmark and the Comparing Logistic Regression (LR), SVM, and stronger baselines (Random Forest, XGBoost, LSTM, Hybrid Ensemble) in terms of Accuracy and F1-Score are illustrated in Figure 7, in the same way the Table 7 shows the various models performance matrices.

CONCLUSION

This article is an in-depth analysis of the pros and cons of Support Vector Machines (SVM) and Logistic Regression (LR) in the ability to predict failure in distributed systems. These systems can give the systems the ability to flag proactively pending issues such that the systems can start to remediate issues before they become problems. This will lead to (potentially) much higher reliability and (definitely) much higher time between failures. LR has particular value in that it gives true probabilities so all estimates and chances of a failure occurring

are obvious. While SVM performed better to classify states of the systems on more complex and multidimensional data sourced in distributed systems. Together these devices offered a generative and predictive maintenance system; an over-arching, proactive approach that allows the systems to automate processes and take action on their own and anticipate outage before being put into an outcome during the time period. This results in decreasing ad-hoc issues. There is much work yet to do. There are more aggressive route with machine learning which may provide a better route to use time-patterned data than a time-based or time-ordered. For example, Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN) offer opportunities to work with patterns rather than sequences of instances. It may also be beneficial to broaden these studies with mixed model or ensemble models.

"Competing Interests and Funding
The authors have no relevant financial or non-financial interests to disclose.
Acknowledgments
Manuscript Communication Number (MCN): IU/R&D/2024-MCN0003166 office of research and development, Integral University, Lucknow".

REFERENCES

[1] Hwang, K., Dongarra, J., & Fox, G. C. (2013). Distributed and cloud computing: from parallel processing to the internet of things. Morgan kaufmann. ISBN 978-0-12-385880-1
[2] Abdi, A., & Zeebaree, S. R. (2024). Embracing Distributed Systems for Efficient Cloud Resource Management: A Review of Techniques and Methodologies. Indonesian Journal of Computer Science, 13(2). https://doi.org/10.33022/ijcs.v13i2.3806
[3] 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, 1-15. https://doi.org/10.1007/s13198-024-02263-5
[4] Van Steen, M. (2002). Distributed systems principles and paradigms. Network, 2(28), 1.
[5] Siddiqui, Z. A., & Haroon, M. (2022). Application of artificial intelligence and machine learning in blockchain technology. In Artificial Intelligence and Machine Learning for EDGE Computing (pp. 169-185). Academic Press. https://doi.org/10.1016/B978-0-12-824054-0.00001-0
[6] Zidi, S., Moulahi, T., & Alaya, B. (2017). Fault detection in wireless sensor networks through SVM classifier. IEEE Sensors Journal, 18(1), 340-347. https://doi.org/10.1109/JSEN.2017.2771226
[7] Lin, C. Y., Tsai, C. H., Lee, C. P., & Lin, C. J. (2014, October). Large-scale logistic regression and linear support vector machines using spark. In 2014 IEEE International Conference on Big Data (Big Data) (pp. 519-528). https://doi.org/10.1109/BigData.2014.7004269
[8] Lin, G. F., Chang, M. J., Huang, Y. C., & Ho, J. Y. (2017). Assessment of susceptibility to rainfall-induced landslides using improved self-organizing linear output map, support vector machine, and logistic regression. Engineering Geology, 224, 62-74. https://doi.org/10.1016/j.enggeo.2017.05.009
[9] Yu, W., Liu, T., Valdez, R., Gwinn, M., & Khoury, M. J. (2010). Application of support vector machine modeling for prediction of common diseases: the case of diabetes and pre-diabetes. BMC medical informatics and decision making, 10, 1-7. https://doi.org/10.1186/1472-6947-10-16
[10] Kumar, A., Dutta, S., & Pranav, P. (2023) Prevention of VM Timing side-channel attack in a cloud environment using randomized timing approach in AES–128. International Journal of Experimental Research and Review, 31, 131-140. DOI : https://doi.org/10.52756/10.52756/ijerr.2023.v31spl.013
[11] Sharfuddin, N., Anwer, F., & Ali, S. (2023). A novel cryptographic technique for cloud environment based on feedback dna. Int. J. Exp. Res. Rev, 32, 323-339. DOI: https://doi.org/10.52756/ijerr.2023.v32.028
[12] Azhar, N., & Haroon, M. (2019). Dynamic Load Balancing by Round Robin and Warshall Algorithm in Cloud Computing. International Journal Innovation Technology Explorer Engineering, 8(9), 953-63.
[13] 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
[14] Isukapalli, S., & Srirama, S. N. (2024). A systematic survey on fault-tolerant solutions for distributed data analytics: Taxonomy, comparison, and future directions. Computer Science Review, 53, 100660. https://doi.org/10.1016/j.cosrev.2024.100660
[15] De Cock, M., Dowsley, R., Horst, C., Katti, R., Nascimento, A. C., Poon, W. S., & Truex, S. (2017). Efficient and private scoring of decision trees, support vector machines and logistic regression models based on pre-computation. IEEE Transactions on Dependable and Secure Computing, 16(2), 217-230. https://doi.org/10.1109/TDSC.2017.2679189
[16] Cotroneo, D., De Simone, L., Liguori, P., & Natella, R. (2020). Fault injection analytics: A novel approach to discover failure modes in cloud-computing systems. IEEE transactions on dependable and secure computing, 19(3), 1476-1491. https://doi.org/10.1109/TDSC.2020.3025289
[17] Mohammed, B., Awan, I., Ugail, H., & Younas, M. (2019). Failure prediction using machine learning in a virtualised HPC system and application. Cluster Computing, 22, 471-485. https://doi.org/10. 1007/s10586-019-02917-1

[18] Gururaj, H. L., Flammini, F., Swathi, B. H., Nagaraj, N., & Ramesh, S. K. B. (2023). Machine Learning Techniques for Fault Tolerance Management. In Computational Intelligence for Cybersecurity Management and Applications (pp. 83-100). CRC Press. https://www.taylorfrancis.com/ chapters/edit/10.1201/9781003319917-7/machine-learning-techniques-fault-tolerance-management-harinahalli-lokesh-gururaj-francesco-flammini-beekanahalli-harish-swathi-nandini-nagaraj-sunil-kumar-byalaru-ramesh

[19] Dhiyanesh, B., Rameshkumar, M., Karthick, K., & Radha, R. (2023). Cloud computing and machine learning for analysis of health care data based on neuro fuzzy logistic regression. Journal of Intelligent & Fuzzy Systems, 44(6), 9955-9964. https://doi.org/10.3233/JIFS-223280

[20] Schapire, R. E. (2003). The boosting approach to machine learning: An overview. Nonlinear estimation and classification, 149-171. ISBN 978-0-387-21579-2. https://doi.org/10.1007/978-0-387-21579-2_9

[21] 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.

[22] Das Chagas Moura, M., Zio, E., Lins, I. D., & Droguett, E. (2011). Failure and reliability prediction by support vector machines regression of time series data. Reliability Engineering & System Safety, 96(11), 1527-1534. https://doi.org/10.1016/j.ress.2011.06.006

[23] 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. DOI: https://doi.org/10.52756/ijerr.2023.v32.035

[24] Kumari, A., & Mehta, A. K. (2020). A hybrid intrusion detection system based on decision tree and support vector machine. In 2020 IEEE 5th International conference on computing communication and automation (ICCCA) (pp. 396-400). https://doi.org/10.1109/ICCCA49541.2020.9250753

[25] Chen, K., Chen, H., Zhou, C., Huang, Y., Qi, X., Shen, R., ... & Ren, H. (2020). Comparative analysis of surface water quality prediction performance and identification of key water parameters using different machine learning models based on big data. Water research, 171, 115454. https://doi.org/10.1016/ j.watres.2019.115454

[26] Haroon, M., Siddiqui, Z. A., Husain, M., Ali, A., & Ahmad, T. (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. https://doi.org/10.52756/ijerr.2024.v44spl.018

[27] 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

[28] Jafarigol, E., & Trafalis, T. (2023). A review of machine learning techniques in Imbalanced Data and Future trends. https://doi.org/10.48550/arXiv.2310.07917

[29] Sharief, F., Ijaz, H., Shojafar, M., & Naeem, M. A. (2024). Multi-Class Imbalanced Data Handling with Concept Drift in Fog Computing: A Taxonomy, Review, and Future Directions. ACM Computing Surveys, 57(1), 1-48. https://doi.org/10.1145/3689627

[30] SB Verma, Brijesh P., and BK Gupta, Containerization and its Architectures: A Study, ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal, Vol. 11 N. 4 (2022), 395-409, eISSN: 2255-2863, DOI: https://doi.org/10.14201/adcaij.28351

[31] Chen, K. C., Xu, X., Makhanov, H., & Chung, H. H. (2024). Quantum-Enhanced Support Vector Machine. In Advanced Intelligent Computing Technology and Applications: 20th International Conference, ICIC 2024, Tianjin, China, August 5-8, 2024, Proceedings (Vol. 14871, p. 155). Springer Nature.