Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 3, Issue 11, 2025

OPTIMIZING FAULT TOLERANCE IN CLOUD COMPUTING WITH MACHINE LEARNING: A COMPARATIVE STUDY OF DEVELOPING AND INNOVATIVE HYBRID APPROACH

Muhammad Asim Shahid[*1], Imran[2], Adnan Alam Khan[3]

[*1]*College of Computing and Software Engineering, Ziauddin University, Karachi, Pakistan*
[2]*Department of Computer Science, Bahria University, Karachi, Pakistan*
[3]*Department of Computer Science, DHA Suffa University, Karachi, Pakistan*

[*1]asim.shahid@zu.edu.pk

DOI: https://doi.org/10.5281/zenodo.17836921

Keywords
Cloud computing, Delta check pointing, Fault tolerance, Fault classification and prediction, Machine learning algorithms, Weibull distribution

Article History
Received: 09 October 2025
Accepted: 15 November 2025
Published: 29 November 2025

Corresponding Author: *
Muhammad Asim Shahid

Abstract

Cloud computing is one of the most rapidly growing areas of the computer industry, providing a wide variety of advantages and opportunities for users. As a result of their rapid growth during the past decade, many businesses have transitioned to cloud services so they can gain better access to their data, scale quickly to meet customer requirements, and have improved visibility into the operations of their company. In addition to these benefits, there are many challenges associated with cloud computing, including resource allocation, security, availability, privacy, quality of service, data management, performance compatibility, and fault tolerance. Fault tolerance (also known as FT) is the property of a system that enables the system to continue serving its intended purpose even in the presence of one or more faults. There are many challenges associated with FT, including heterogeneity, the absence of well-defined standards, automation problems, and reliability of downtime for a cloud service, recovery points, recovery time objectives, and management of cloud workloads. Machine learning (ML) algorithms such as AdaBoostM1, Bagging, Decision Tree (also called J48), Deep Learning (also called Dl4jMLP), and Naive Bayes Tree (NB Tree) have been incorporated into the proposed study to improve the accuracy of predicting faults occurring in cloud computing environments and to reduce the number of errors associated with predicting faults, as well as to improve the reliability of cloud computing. In addition, it uses an error tolerance strategy called delta-check pointing. Virtual Machines (VMs) were used to create data used to determine the best machine learning classifier. Then, in order to create accurate data and reduce the errors in prediction associated with the repair or failure of these Virtual Machines, the original data for this study was divided into an 80/20 split and a 70/30 split. Another technique used for this study was 10-fold cross validation. In the area of Machine Learning, the Naive Bayes Tree (NB Tree) classifier provided the highest accuracy with the lowest number of errors. The accuracy for the 80/20 Split was 97.05%, for the 70/30 Split was 96.09%, and for the 10-Fold Cross Validation was 96.78%, according to the original dataset used in this study. The execution time for the NB Tree classifier was 1.01 seconds which is significantly longer than the second-best classifier, Decision Tree (J48).

The accuracy of Decision Tree (J48) as ranked is 96.78% for the 80/20 Split, 95.95% for the 70/30 Split, and 96.78% for 10-fold cross-validation which places it second overall. One distinguishing feature of J48 compared to the NB Tree is its lower algorithmic complexity (0.11 seconds for J48 versus 1.01 seconds for NB Tree) and fewer errors in prediction for J48. Although the accuracy and error prediction between J48 Decision Tree and NB Tree differ by only 0.90%, there exists a nine-second operational complexity in time between them as well. Consequently, we decided to modify our J48 Decision Tree algorithm based upon the findings from my analysis. My recommendation for using this method is because it generates the best accuracies and least predictions when applied to the datasets I analyzed: 97.05% when I performed an 80/20 split; 96.42% for a 70/30 split; and 97.07% after performing a 10-fold cross-validation.

INTRODUCTION

Research background & motivation

The introduction of cloud computing as a new Information Technology delivery model has resulted in it being accepted as one of the leading business models for delivering IT Applications, Components and Infrastructure. The Cloud – where the infrastructure is managed by the provider – has become more attractive to users because of the savings it provides as well as the ability for effective resource allocation [1], [2]. There are Five Core Characteristics of Cloud Computing: 1) on-demand self-service 2) broad network access 3) pooled resources 4) rapid elasticity 5) measurable services. There are Four Different Deployment Models: 1) Hybrid Cloud 2) Public Cloud 3) Community Cloud 4) Private Cloud. The Cloud Service Model comprises Three Service Models: 1) Software as a Service (SaaS) 2) Platform as a Service (PaaS) 3) Infrastructure as a Service (IaaS) [3]. Users access Cloud Computing via the internet and pay for the resources they consume. All Resources utilized by the user are the responsibility of the Cloud Provider that has processed them [4]. Deficiencies in Cloud Infrastructure Environments will greatly impact Cloud Resource Reliability. Identifying and Correcting Possible Weaknesses within Cloud Infrastructure Will Facilitate Improvements in Reliability and Operational Efficiency for Cloud Computing [5].

Cloud security is faced with the difficulty of Fault Tolerance (FT). Several authors have recognized this issue and have made attempts at providing cloud solutions and developing FT frameworks, but their efforts do not seem to have been successful enough to create a viable solution. FT is defined as "the ability of a system to continue to perform correctly when at least one of its internal components is not working properly" [5]. There are four categories of faults found within a fault-tolerant system: transient faults, intermittent faults, persistent faults, and Byzantine faults [6], [7]. Accordingly, techniques associated with the fault tolerance of computer systems can be classified into three primary categories: Reactive Approach (RAMs), Proactive Approach (PRMs), and Resilient Approach (RMs). Some of the techniques that fall within this classification system include [8]:

In [9] various techniques have been proposed to increase the level of reliability of systems and help them continue to operate as intended when there are errors and/or faults present. In a distributed computing environment, one of the primary obstacles to increasing the reliability of a System is maintaining customer trust and satisfaction while limiting the loss of revenue; therefore, cloud computing must be scalable and resilient in order to be of any value at all.

When discussing reliability in the context of cloud computing, the definition is "the system's ability to provide the correct service (function)" during a certain period of time and under certain defined environmental conditions. While there are numerous methods available for improving the reliability of Cloud Services, no extensive investigation of the various options exists to date [10]. As such, improving cloud service reliability

will be one of the more important areas of CC [11].

The dataset used for this research will be based on the Weibull Distribution; a Model frequently used to determine how long a system will continue to operate before it fails. The Weibull Distribution is considered more effective than the Exponential Distribution for this purpose because it provides for both increasing and decreasing reliability curves based on the changing likelihoods of a system's failure. Additionally, it is used to illustrate these two different forms of reliability (or failure mode) for both early and late stages of a system's operating history [12].

The use of technology development has allowed for advancements in data collection, therefore Machine Learning (ML) is one area of potential for using ML to analyze and optimize energy systems within buildings. The ML community continues to grow as it is intersected by both statisticians and computer scientists. ML has become an important component of the data science and AI (Artificial Intelligence) communities [13]. In recent years, ML has significantly reduced the recovery time of Random Access Memory (RAM) systems down to milliseconds when these systems experience a fault. As the design and architecture of these systems evolve, so do the methods for recovering from faults. Recently, there has been growing interest in RSMs (Rapid-Self-Modifying Systems) for providing trusted disaster recovery through improved means for recovering from power outages. RSMs include the capability to monitor the current system state, respond to customer inquiries regarding equipment failures, and adapt following the identification of errors or successful predictions. The application of ML to RSMs provides a means for of a system to improve through learning, adapting and developing the ability to recover following the occurrence of fault or failure events. The continual desire to find predictive techniques, develop tools for learning from faulty actions, and monitoring the performance of a system while responding to customer needs has enabled the evolution of the RSM concept. The application of ML to RSMs greatly enhances the capacity of these systems to "learn" and be more flexible than what was previously possible [7].

Materials and methods

Literature review

According to Shahid et al. [14], Cloud Computing (CC) has become a major development over the past few years. The CC industry has turned previously isolated Distributed Computing systems into Global Networks allowing customers from anywhere in the world to access services provided by enterprises such as IBM, Amazon and Microsoft without the need for any software installation on their computers. CC also has various issues including Security, Compliance with Regulations, Governance, Fault Tolerance, Interoperability, Data Protection and Availability. Cloud architectures use multiple servers as opposed to a single server. All of these servers are used together by several customers who run applications on them. There are three main layers that comprise most Cloud Infrastructure: Infrastructure as a Service (IaaS), Software as a Service (SaaS) and Platform as a Service (PaaS). Any errors may occur at any of these layers, recovery mechanisms have been developed for the Software Layer in order that Cloud Service Providers may operate without interruptions.

Gupta et al. [15] proposed that Distributed Grid Computing and Fault Tolerance (FT) is an extremely difficult area of research and development. Because many of these systems use a significant amount of computing power, solutions to complex problems may take considerable time to develop. However, with the flexibility offered by CC, many unforeseen tasks may arise and can result in a defective product. The success of CC in maintaining reliability and resilience depends on how well Defects are dealt with and managed. FT in Computer Systems can be split into two categories; Hardware Fault Tolerance and Software Fault Tolerance.

Mukwevho and Celik [16], [17] suggest that rather than completely stopping services when a disaster occurs (such as server failure), "cloud downtime" will be redefined as preserving

conference status within storage; some major outages at one data centre may negatively affect other organisations dependent upon it, therefore cloud service providers must ensure that every organisation's Service Level Agreements (SLA) are met.

Edemo [18] notes that this approach serves as an effective response to continuing projects; CP ensures a continued failure-free status of the application following completion. This approach is employed at the library, application and device level.

Feng et al. [19] describe a machine-learning predictive method of concrete compressive strength. Their method uses a combination of weaker predictors that have been combined to produce a strong model connecting input and output data through increased boosting.

Butt et al. [20] offer a review article on the security risks and solutions associated with Cloud Computing (CC) that use machine learning methods. The authors analyse several CC security risk mitigation approaches based on Supervised, Unsupervised, Semi-supervised and Reinforcement Learning methods, evaluating the strengths and weaknesses of each and recommending areas for future research to enhance CC security.

According to Pei and his colleagues [21] the Multidimensional Fusion (CNN-BiLSTAM-Attention) (CBA)-net provides a new way of doing fault prediction, and is able to accurately learn from previous fault logs an understanding of both where in the world and when faults record occurred. To preprocess the data before fault predictions are made, it uses a novel data preprocessing technique which employs Hierarchical Density Based Clustering of Applications with Noise (HDBSCAN). This new technique allows the fault prediction model to have an increased sensitivity to time series characteristics; and to effectively extract local features. Pei et al's. study found that the Root Mean Square Error (RMSE) for the estimated times at which a fault will occur was 0.031, and the average success rate of finding the nodes involved (where the fault occurred) was 93% accurate, making this model a viable tool for accurately predicting faults in very large supercomputing systems where it would be difficult to directly monitor those faults with physical monitoring equipment.

The research by Shrestha & Mahmood [22], investigated optimization techniques to minimize training time as well as maximizing accuracy when using deep learning. This research investigates the underlying mathematics of the most commonly used training techniques, as well as the advantages and disadvantages of these training techniques, and gives examples of how these techniques can be applied to many different applications. The research also explored a number of different types of deep learning models; convolutional, recurrent, residual, reinforcement learning, and variational autoencoders models.

Lang et al. [23] deep Learning is a sub-theme in machine learning that uses artificial neural networks to create multi-layer arrays or representations of data, providing major improvements in areas such as document classification, object detection, image-processing speech recognition, image-classification. WekaDeeplearning4j is an application of Weka's GUI-based, allowing users to easily create Weka's deep-learning applications using Deeplearning4j as the backend engine by training and building deep neural networks, Convolutional Neural Networks (CNN) and recurrent neural networks (RNN) using GPU on text or image-based datasets.

Wang et al. [24] using a multinomial naive Bayes classifier at each leaf of a decision tree, Wang et al. [24] created the Multinomial Naive Bayes Tree (MNBTree), which is a binary-based version of its predecessor NBTree. The MNBTree builds a binary tree using a split down from zero and non-zero values, and improves the speed of constructing MNBTree through the use of information gain. In order to improve the classification performance, we have developed the Multiclass Multinomial Naive Bayes tree (MMNBTree), and a comparison of results from employing the proposed MNBTree and MMNBTree against existing text classification

datasets shows the superior performance of both models.

In their paper, Bildosola et al. [25] developed a cloud computing adoption decision-supporting tool, specifically for SaaS solutions, through a diagnostic application framework consisting of predefined questions for users to sort and accumulate the required data and generate a customised Cloud Roadmap for operating in the cloud environment. Bildosola and his colleagues conducted a pilot study with businesses located in the local area to gather their cloud computing knowledge and identify promising industries and effective tools for those industries to be utilised when determining the most appropriate cloud service provider.

Zhang et al. [26] conducted research on how to incorporate feature weighting techniques into text classifiers when using the naive Bayes algorithm. The many feature weighting techniques that have previously been proposed often have drawbacks concerning increasing the complexity of the model and the processing time to use them, while they do not consistently provide significant improvements in the performance of naive Bayes classification algorithms. Nevertheless, feature weighting in general continues to be a key topic of research in machine learning, and there have been significant contributions to this area from academic researchers. Zhang and his colleagues have also shown that there exist simple and effective feature weighting methods that can be easily implemented into naive Bayes text classifiers.

Liu et al. [27] state that failure detectors are very important to maintain high availability in distributed systems, especially when they are used in complex multi-application systems. The majority of existing research on accrual failure detectors has been conducted, however, many of these fail in cloud service environments. The Weibull Distribution Failure Detector is a newly developed technology that can be used in cloud computing and is designed to use the Weibold Distribution function to adjust to the changing environment of a cloud computing network. Through experimental testing, it has been shown that the Weibull Distribution Failure Detector is significantly faster and more accurate than other failure detection systems, particularly in scenarios of uncertainty such as those found in cloud computing.

Madni et al. [28] have researched cloud computing as an infrastructure capable of handling large task loads; however, scheduling jobs on cloud computing platforms is NP-complete and, therefore, heuristic methods for solving job scheduling problems must be used. Depending on their operating assumptions, it is difficult to determine the best-suited algorithm for the particular assignment being scheduled. In order to remedy this condition, six different rule-based heuristic algorithms have been developed that can be applied to both homogenous and heterogeneous environments. The performance of these algorithms: First Come First Served (FCFS), Minimum Completion Time (MCT), Minimum Execution Time (MET), Maxmin, Min-min, and Sufferage, were compared using various performance metrics, including cost, makespan, throughput, and imbalance.

Tanha et al. [29] also proposed that machine learning algorithms can be trained on both labeled and unlabeled data, where one approach for semi-supervised learning is decision trees learners along with self-training. 1) Decisions Trees However, the standard approach to learning decision trees is not sufficient since it cannot derive accurate probability values for predictions. There are some works available in literature, which tries to enhance the decision tree algorithm like grafted tree [11], Distance-based tree [12], Naive-Bayes Tree [13], Laplace correction and no pruning combination [14]. These improvements have also been extended to ensembles of decision trees, outperforming modified decision trees and leading to a significant progress.

Patel & Prajapati [30] stated that learning machines is a task of making computers to make decision by using the predefined data sets, without a particular programming. Decision trees are one of the important methods in this area, widely applied for many applications such as search engine, healthcare and statistical analysis etc. Various algorithms of decision tree,

including ID3, C4. 5 and CART have been devised for their effectiveness and precision. These three of algorithms are studied in this work and their practical use is discussed.

Mesbahi et al. [31] present WekaDeeplearning4j, a Weka-based software interface that facilitates deep learning using a graphical user interface. Using Deeplearning4j as the backend, the system trains deep neural networks (including convolutional and recurrent architectures) and provides utilities to preprocess both text and image data, supporting also GPU acceleration.

In the work [32], Attallah et al. propose an approach focused on making cloud computing infrastructure more available and reliable based on acting on CPU failures at virtual machines (VM). This method monitors the CPU utilization and reacts to high CPU usage either by migrating the VM to another host, or rebalancing workloads on the current host, enabling the transfer.

K. Devi [33] proposes virtual machines fault tolerance (VMFT) as a way to achieve reliability in the ability of the system to survive failures, reducing service times and therefore increasing dependability. The criterion for the selection of the best VM is the one that produces correct results in the minimum time. A cloud simulation tool has been utilized to implement VMFT, while the reliability of a single-node VM is measured using its program execution time; therefore, the most reliable VM is the one which provides the results within the expected execution time.

Problem statement

Reliability is a key metric, which may be different for successive computation steps. Cloud computing should provide consistent performance to meet the reliability requirements. A basic approach to achieving this is primary backup replication in fault-tolerant software. While cloud computing is gaining momentum, little is known about its impact on system stability. The role of decision trees in improving the metrics for reliability within cloud infrastructures is also not well understood. This work aims to examine the pros and cons of decision trees in enhancing reliability during cloud computing [31].

Addressing fault tolerance through machine learning model designs will be crucial for high accuracy, reduced fault predictions, and dependable operations of virtual machines.

Methodology for Conducting Research

In this section, the methodology suggested was primarily concerned with the implementation of the method. In addition, the section describes in detail how the design of the study will be developed, how data will be generated and then processed into usable information, how data will be analyzed, and finally, how the analytical techniques will be structured.

Structure of the Research Design

This shows how the research design for the current study will be created. The following machine learning techniques will be employed:

Application of Machine Learning

This research will use supervised learning methods for improved accuracy and reduced failure prediction error rates through the use of labelled datasets for classification of data and predicting outcomes.

Fault Tolerance (FT) Methodology

This study is exploring the use of FT to detect and correct virtual machine (VM) failures in Cloud Computing (CC) and the use of D-CP to decrease the impact of VM failure on service requests.

Reliability

The ability of a cloud computing (CC) system to perform tasks reliably for a defined period, within certain specified circumstances. Reliability is achieved in this competitive environment through FT techniques and the combined use of machine learning to assure flawless operation of all virtual machines.

Research Methodology

We created a flowchart to help clarify the flow of Research. We used data obtained through the

Weibull distribution method to use as primary data.

Through machine learning (ML) and Dynamic Change Point (D-CP) methods, we developed a robust and precise solution with high accuracy, minimal fault prediction errors, and increased reliability in Cloud Computing represented in Figure 1. This novel approach has created a framework for the future development of this platform.

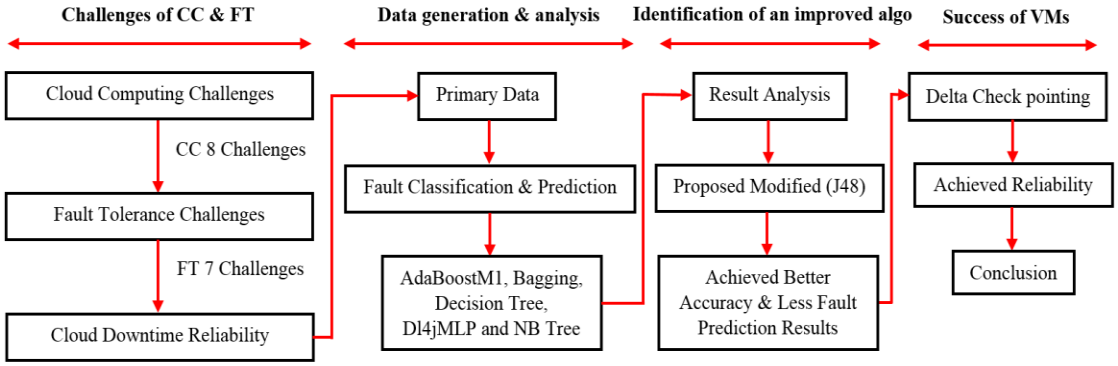


Fig 1. An illustration depicting the implementation view of the research framework.

Generated primary data

The data we generated was based on the Weibull Distribution. The Weibull Distribution is often used when performing Time-To-Failure analysis on reliability studies. This is because it allows users to model situations with "non-constant" failure rates (Early & Wear Out). Therefore, this means you can create two different types of failure curves (increasing/Decreasing) with the Weibull Distribution [12].

We employed multiple parameters on the Java® platform to develop our dataset. The following table summarizes all of the relevant factors that influenced our dataset.

Table 1. A summary of the key factors that influenced the development of the dataset.

Users	Port No	Host No	Network Host	Distribution
2	32	384	Bandwidth, Storage, RAM, and MIPS	The Weibull distribution includes failure rate functions that can either increase or decrease.

Table 2. A brief summary of the primary dataset.

Attributes	Attributes Names				Attributes Type	Instances
7	1.	FHID	2.	HFTIME	Numeric and Nominal	1400
	3.	LFT	4.	DIS		
	5.	DISHT	6.	FTIME/RTIME		
	7.	STATUS				

Mathematical equations for data analysis techniques

AdaBoostM1

$$\Pr_{i\sim w1}[h_f(x_i \neq y_i)] \leq \sum_{t=1}^{T} \sqrt{1 - 4Y_t^2} \quad (1) \quad [34]$$

Bagging

$$Q(j|x) = P(\emptyset(x, L) = j \quad (2) \quad [35]$$

Decision tree (J48)

$$\text{Entropy}(S) = \sum_{i=1}^{c} P_i \log 2^{P_i} \quad (3) \quad [36]$$

$$\text{Gain}(S, A) = \sum_{v \in} \frac{S_v}{S}_{V(A)} \text{Entropy}(S_v) \quad (4) \quad [36]$$

Deep learning (Dl4jMLP)

$$J(0) = \frac{1}{m}\sum_{i=1}^{m} L\left(y_i^{\theta}, y_i\right) \quad (5) \quad [37]$$

Naïve bayes tree

$$p(m) \infty \pi_m \prod_{n=1}^{|v|} p(n|m) w_n \quad (6) \quad [38]$$

Modification of the decision tree (J48)

Although the original J48 method has the highest amount of errors in its predictions and the lowest rate of accuracy, it will be used with a change in methodology by creating a changed J48 classification tree to provide improved accuracy and reduce the number of errors. A diagram illustrating this enhanced classifier can be found in the following figure (Figure 2). The primary dataset used to increase accuracy and decrease instances of fault prediction errors.

High accuracy and low fault prediction errors have been evaluated for \(G_{(F_2)} \in G_{(F_2)} G_{(F_2)}^+ \) and \(G_F \in G_F G_F^+ \) using an objective function. Fault prediction is achieved with high precision and reduced errors through the use of objective functions and algorithm parameters. The split point is not a true number when the confidence factor parameter is between 0.52 and 0.5.

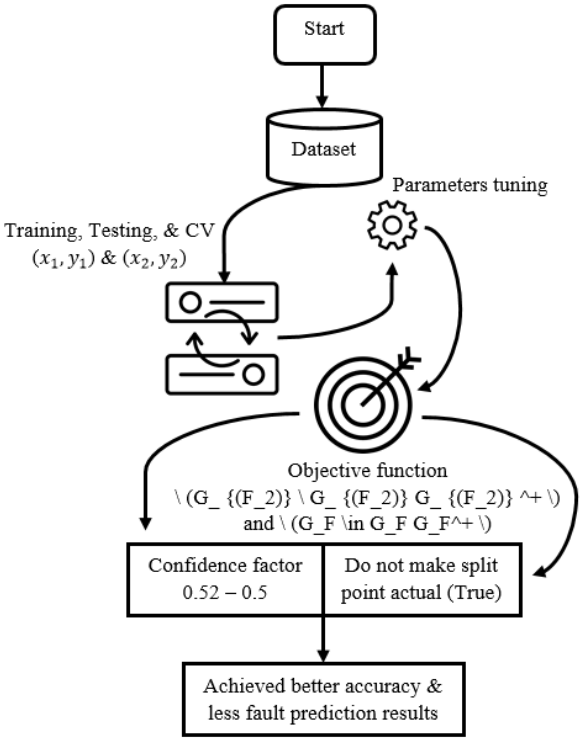


Fig 2. Displays the Modified Decision Tree (J48) classifier's block diagram.

Equations 7 through 9 present the mathematical formulas for the modified J48 decision tree classifiers.

$$(x_1, y_1) \text{ and } (x_2, y_2) \tag{7}$$

Training, testing, and cross-validation procedures are outlined in Equation 7.

$$G_{F_2} - \in G_{F_2}\ G_{F_2}{}^+ \in \ \& \ G_F - \in G_F\ G_F{}^+ \in \tag{8}$$

Equation 8 specifies the objective function.

$$CI = \bar{x} \pm Z . \frac{s}{\sqrt{n}} \tag{9}$$

Equation 9 defines the confidence factor without using the actual value of the split point as previously stated.

Delta checkpointing

The D-CP technique enables virtual machines to maintain multiple 'snapshots' of their running state (checkpoints) by taking an image of its memory. However, if many CPs were to be sent out over a data center's existing network, the amount of data would rapidly outgrow the capacity of the network. D-CP resolves this problem by creating new CPs with only the differences (modified pages) compared to previous checkpoints. All of the previously saved portions of the virtual machine's full state (base systems) are stored after the very first CP. The time from one CP to another is the period that the technique is defined by (the D-CP interval) [39].

An explanation of the D-CP algorithm

According to the design and implementation of the Customer Process (CP) System, the CP System also allows for the duration of a CP

interval to adjust throughout the customer's project lifecycle. The next CP interval is determined using the task's failure history associated with the virtual machine (VM) used to perform the tasks. For example, when there is a history of poor task performance, the length of the next CP interval will be shorter; however, if there is a positive history of successful task performances, then it will allow a longer CP interval to be assigned. The parameters associated with Equations (10) - (15) were derived by the development of the D-CP Method.

τ_{ji}
The time it takes to execute task j on virtual machine i (10)

$\tau\Upsilon_{ji}$
The remaining time required for task j to complete on virtual machine i (11)

$F_i(x1)$
The likelihood of failure for virtual machine i (12)

$F_i(x0)$
Probability of no failure of VM i (13)

h
CP interval (14)

Z
Number of failures during the task execution (15)

Experiments and results

This chapter provides an overview of all findings, classifications, comparisons, performance evaluations, etc., associated with each algorithm (AdaBoostM1, Bagging, J48, DL4J MLP, NB Tree) through the use of the confusion matrix and graphical representations; and will include the results obtained using the J48 algorithm (the lead algorithm of this research) as its primary focus. This research includes a comparison of the algorithms being used as standard machine learning classifiers and those that include FT Delta-CP, with an emphasis on high accuracy, reduced fault prediction, and improved fault tolerance of the algorithms involved in this research.

As part of the analysis of the machine learning classifiers, FT Delta-CP was also included as part of the machine learning classifiers that would be used prior to using FT Delta-CP to perform the prediction of faults in electrical equipment. This analysis will provide insight into how the fault-tolerant (FT) Delta-CP can assist in the overall reduction of faults predicted and improve fault tolerance for the classifiers used in this research. The classifiers used were AdaBoostM1, Bagging, J48, Dl4j MLP, and NB Tree). Each classifier performed sufficiently to provide that the goal of the study had been met, through accurate predictions; and the results for comparisons between the classifiers (AdaBoostM1, Bagging, J48, Dl4jMLP and NB Tree) for accuracy, fault prediction, and validation of the data by class were compared.

Configured an ML classifier to simulate the highest possible level of accuracy and minimum level of fault prediction

WEKA is a free to use Java technology created at the University of Waikato, New Zealand and is currently licensed under the GNU General Public License [40], developing through a series of successive versions. WEKA provides many machine learning algorithms and other tools for data preprocessing, including sampling and

discretizing [41]. Table 3 contains further details on the setting for the experiment.

Table 3. Description of experiment configuration.

Setup	
Processor	Intel (R) Core (TM) i7 3770 CPU @ 3.40 GHz
RAM	8.00 GB
Windows	8.1 Platform
IDE	WEKA (3.9.6)
Weka Interface	Explorer
Java Version	JDK 25

Comparison of multiple classification techniques utilizing a primary dataset

Classifier predictions utilizing; AdaBoostM1, Bagging, J48, Dl4jMLP, NBTrees, on the STATUS category from primary dataset. Results on primary data suggest NBTree yields both the greatest accuracy (80/20: 97.05%); (70/30: 96.09%); (10-fold: 96.78%) and the lowest percentage of incorrect predictions however corresponds to a low complexity (1.01 sec).

The second highest classifier is J48 with the accuracy (80/20: 96.78); (70/30: 95.95%); (10-fold: 96.78%) with a better complexity (0.11 sec). NBTree and J48 differ in accuracy by only 0.9% but differ in complexity by 9 seconds. A detailed comparison of each classifier's accuracy for AdaBoostM1, Bagging, J48, Dl4jMLP, and NBTrees can be found in this article.

In addition to accuracy by category (R/F) this article includes a demonstration of the predictions on the test dataset. For enhanced validation refer to Figures 3 & 4 located in the Primary Dataset.

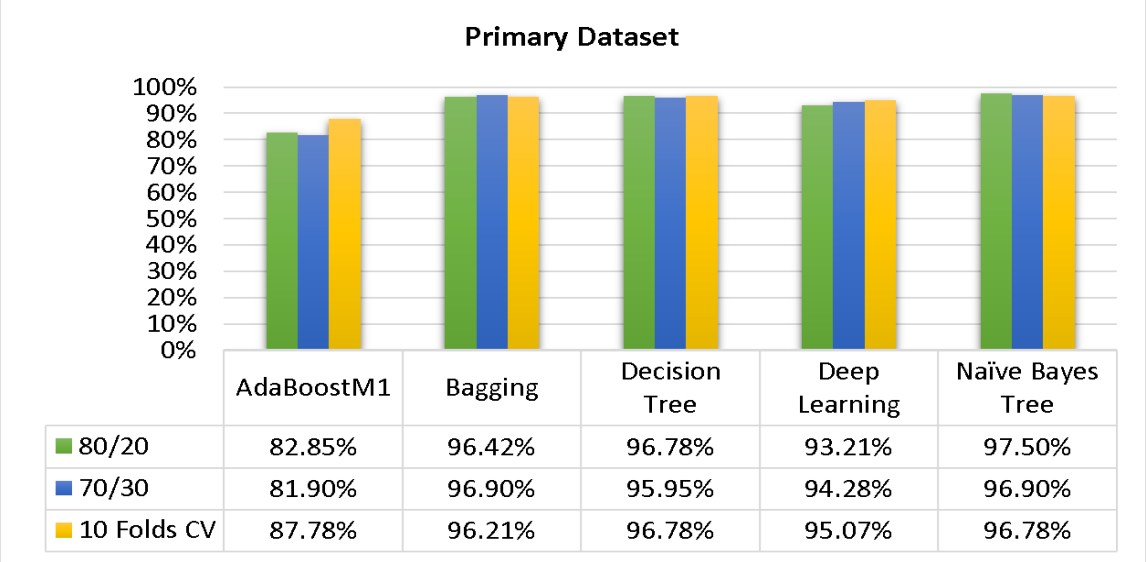
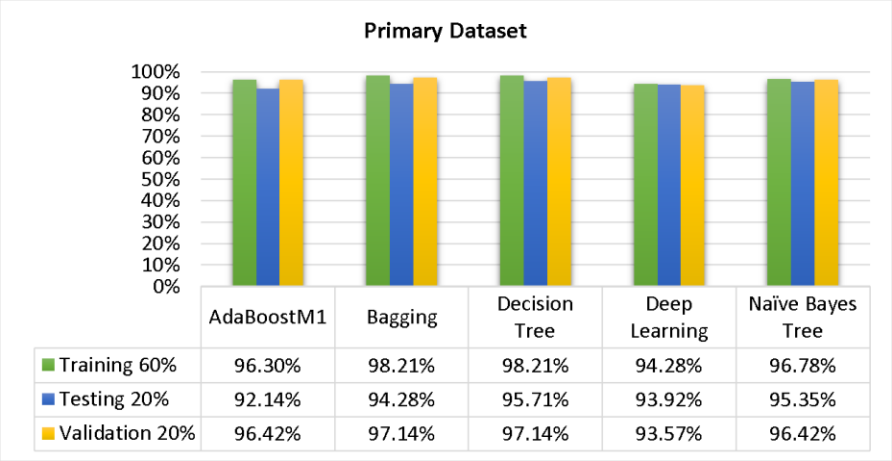


Fig 3. Accuracy of ML classifiers by class (failure/repair).



	AdaBoostM1	Bagging	Decision Tree	Deep Learning	Naïve Bayes Tree
Training 60%	96.30%	98.21%	98.21%	94.28%	96.78%
Testing 20%	92.14%	94.28%	95.71%	93.92%	95.35%
Validation 20%	96.42%	97.14%	97.14%	93.57%	96.42%

Fig 4. Accuracy of ML classifiers by class (failure/repair) in relation to DV outcomes.

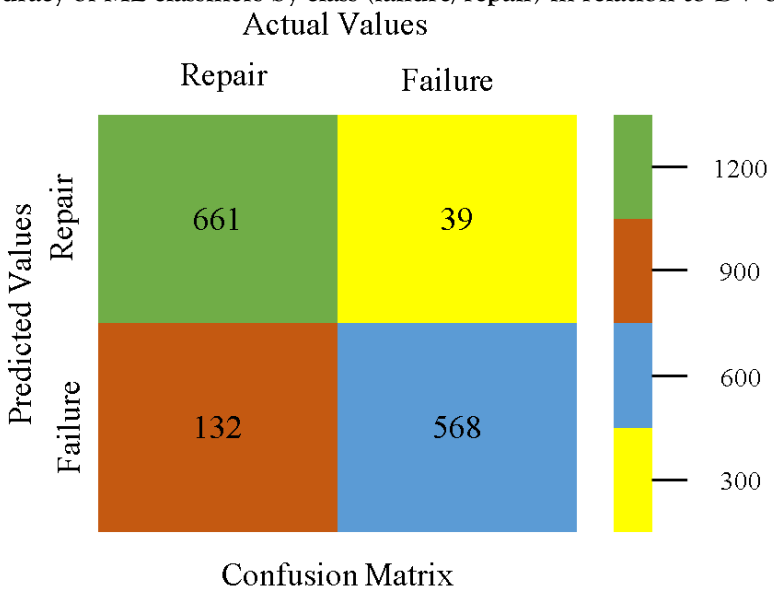


Fig 5. Confusion matrix for the AdaBoostM1 classifier regarding accuracy and fault prediction.

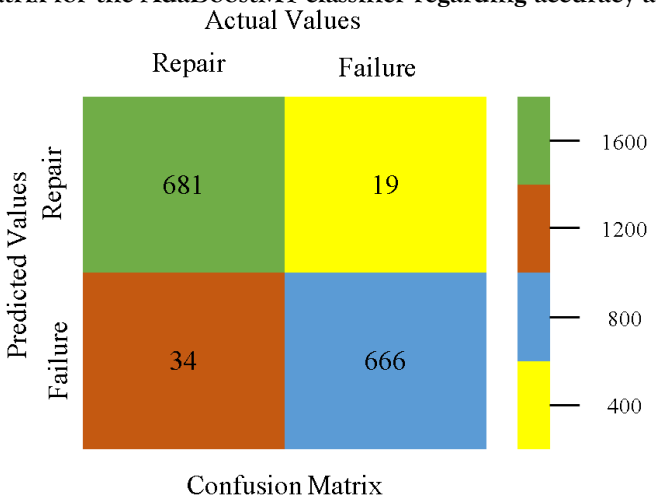


Fig 6. Confusion matrix for the Bagging classifier regarding accuracy and fault prediction.

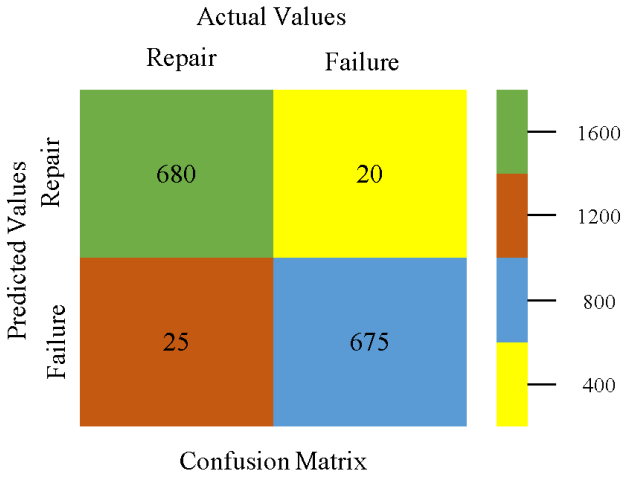


Fig 7. Confusion matrix for the J48 classifier regarding accuracy and fault prediction.

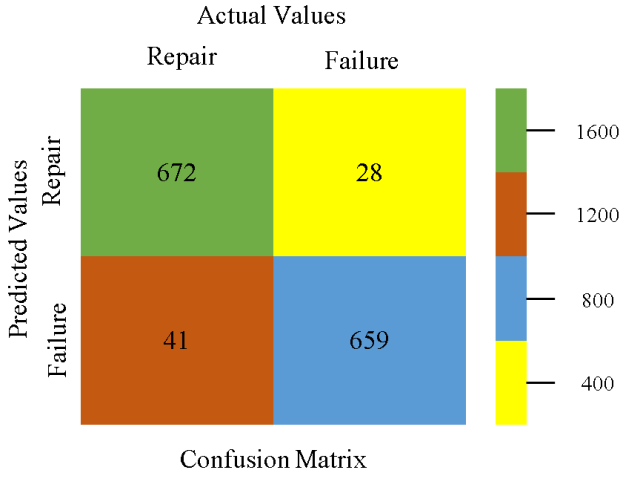


Fig 8. Confusion matrix for the Dl4jMLP classifier regarding accuracy and fault prediction.

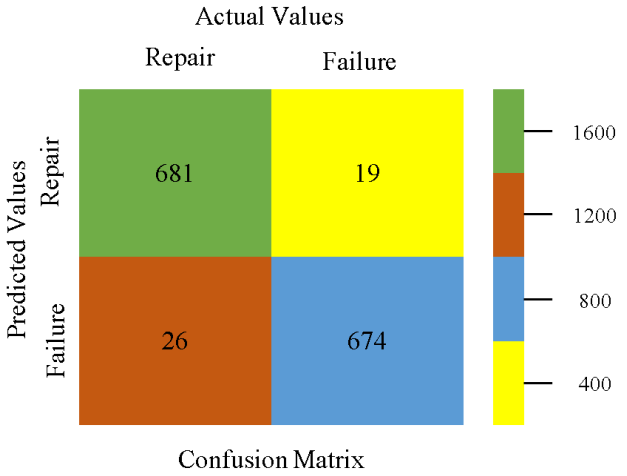


Fig 9. Confusion matrix for the NBTree classifier regarding accuracy and fault prediction.

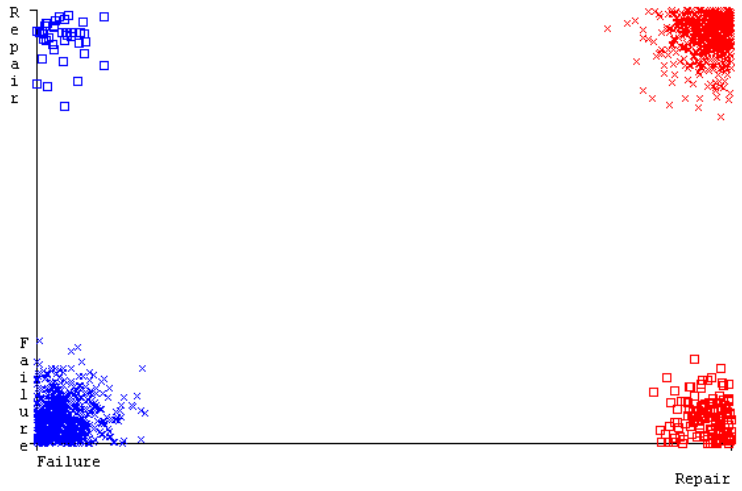


Fig 10. AdaBoostM1 classifier errors regarding accuracy and fault prediction.

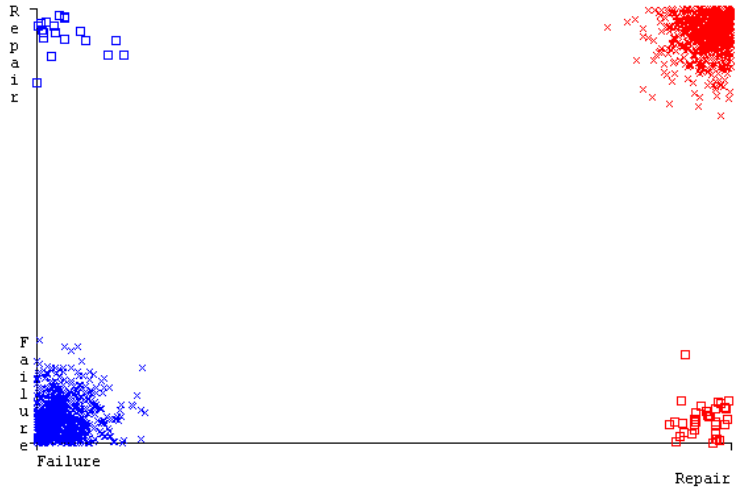


Fig 11. Bagging classifier errors regarding accuracy and fault prediction.

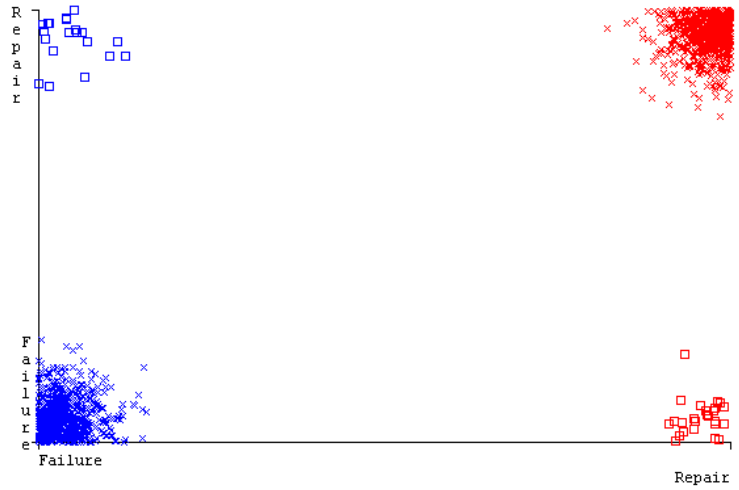


Fig 12. J48 classifier errors regarding accuracy and fault prediction.

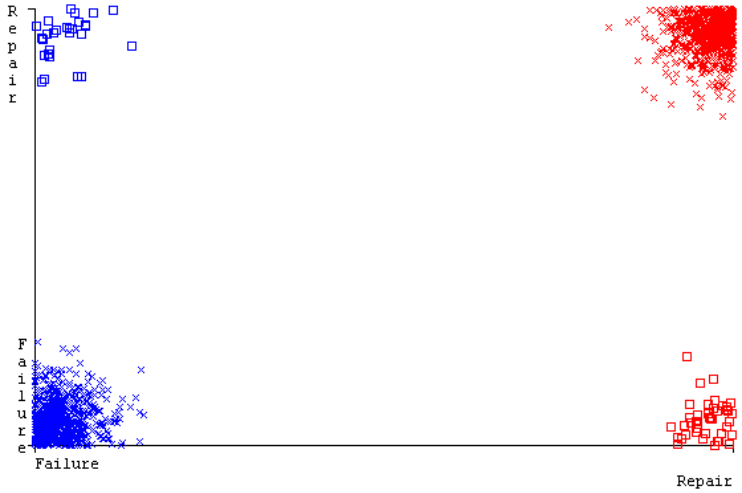


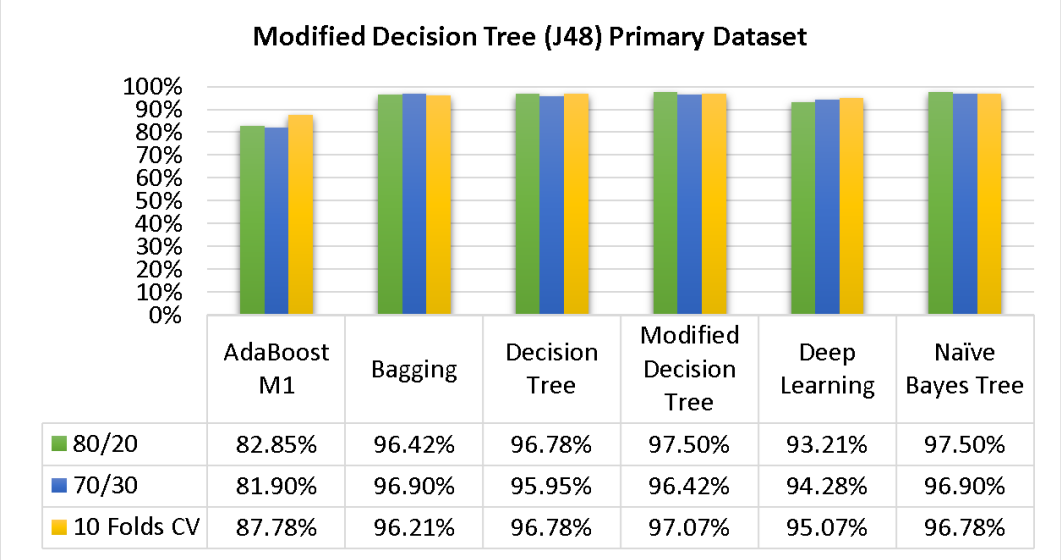
Fig 13. Dl4jMLP classifier errors regarding accuracy and fault prediction.

Fig 14. NBTree classifier errors regarding accuracy and fault prediction.

Results from a modified J48 decision tree

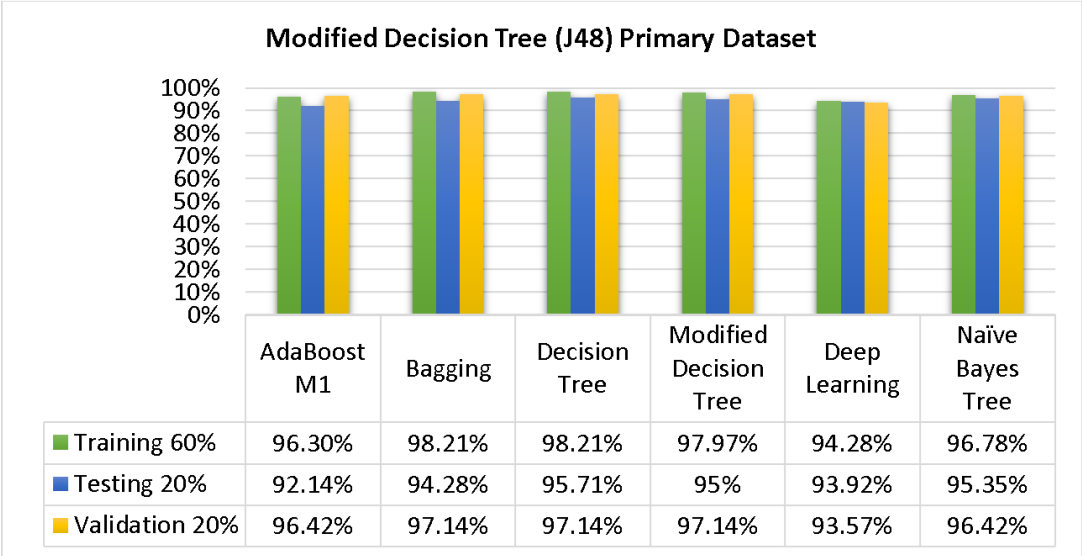
Figures 15 and 16 summarize the significant components of categorization of the datasets. The modified J48 classification Model is the model that produces both the most accurate prediction as well as the least number of errors when making predictions. It produced accuracy ratings of 97.07%, 97.05%, and 96.42% using 10-fold cross-validation, 80/20 splitting, and 70/30 splitting respectively. When the modified J48 algorithm was updated, its time complexity was reduced to 0.02 seconds after the update.

To summarize the comparison between AdaBoostM1, Bagging, J48, Dl4jMLP, and NBTree on the Primary Dataset using accuracy for each class (Repair/Failure) and for validating test-split predictions, see Figures 15 & 16.



	AdaBoost M1	Bagging	Decision Tree	Modified Decision Tree	Deep Learning	Naïve Bayes Tree
80/20	82.85%	96.42%	96.78%	97.50%	93.21%	97.50%
70/30	81.90%	96.90%	95.95%	96.42%	94.28%	96.90%
10 Folds CV	87.78%	96.21%	96.78%	97.07%	95.07%	96.78%

Fig 15. Comparison of ML classifiers with Modified J48 accuracy by class, (failure/repair).



	AdaBoost M1	Bagging	Decision Tree	Modified Decision Tree	Deep Learning	Naïve Bayes Tree
Training 60%	96.30%	98.21%	98.21%	97.97%	94.28%	96.78%
Testing 20%	92.14%	94.28%	95.71%	95%	93.92%	95.35%
Validation 20%	96.42%	97.14%	97.14%	97.14%	93.57%	96.42%

Fig 16. Comparison of ML classifiers with Modified J48 accuracy by class about DV findings.

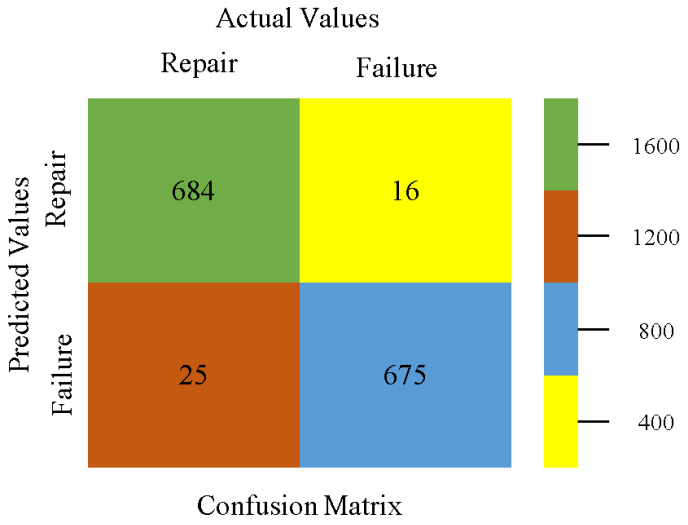


Fig 17. Confusion matrix of the modified J48 classifier for accuracy and fault prediction.

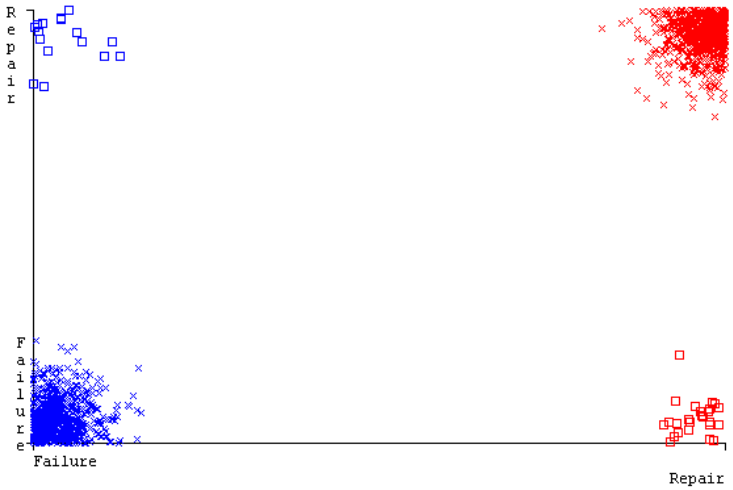


Fig 18. Classifier errors of the Modified J48 for accuracy & fault prediction.

Discussion

Reliability is the first step of fault tolerance (FT) models for cloud computing (CC), so obtaining reliable FT Models with ML can be difficult because ML has been minimally exploited as a method to support the reliability of FT Models. For a model to be dependable, ML techniques need to be combined with FT techniques to provide high levels of accuracy and lowest fault rates.

The study objective was to provide high levels of accuracy, low numbers of prediction errors, and greater reliability through the D-CP method for development of the J48 Classifier. The D-CP methodology simplifies the research process for the creation of the J48 Classifier. The model produced by the J48 Classifier was implemented on the base data set and provided a higher predictive defect identification performance and accuracy than the other classifiers examined in the study.

The classifier outputs were compared against the predictions of the classifiers: AdaBoostM1, Bagging, J48, Dl4jMLP, and NBTree. The main factor in the comparison was high predictive

accuracy, along with low prediction error rates, in evaluating a model's reliability. The findings of the J48 classifier served as a basis for employing the Delta Checking technique to reduce the number of virtual machine (VM) failures due to prediction errors.

In conclusion, this research is innovative and has a tremendous impact on the methods used to achieve high levels of accuracy and low rates of faults for reliability. The D-CP methodology employs varying compression parameters for the model, while the J48 classification processes use an iterative approach to set the parameters for the D-CP Method; thus providing an entirely new methodology.

Conclusions and future work

The goal of this study is to establish a method for improving the prediction of faults in CC by comparing the accuracy, error rates and dependability of traditional methods with Machine Learning approaches and Fault Tolerant (FT) strategies. The Study developed its primary Dataset based on the Weibull Distribution, which is a popular Reliability model that can handle Non-Constant Failure Rates. The Weibull Distribution successfully simulates the failure characteristics of both the Burnout and Wearout phenomena.

The results of this study provided strong evidence that the development of the primary dataset produced a better performing dataset for the refinement of the Machine Learning Method applied in this study. The Machine Learning Method significantly increased the Reliability of the FT process through Increased Accuracy, Decrease in Fault Prediction Error rate, and hence increased confidence in the Reliability of the FT process.

The study concludes that the most accurate classifying algorithm for the Primary Data was the NBTree Classifier with an overall accuracy of 97.05% when using the 80/20 Split, 96.09% using the 70/30 Split, and an accuracy of 96.78% using the 10-Fold Cross Validation Method. The NBTree Classifier also had the Lowest Fault Prediction Rate (0.02) with an Average Execution Time of 1.01 seconds.

The Machine Learning Method with the Next Best Classifying Algorithm, J48, had an accuracy of 96.78% using both the 80/20 and 10-Fold Split methods, and an accuracy of 94.95% for the 70/30 Split method. Like the NBTree Classifier, the J48 Classifier executed in a slightly shorter duration, 0.90 seconds, and maintained a Low Fault Prediction Rate (0.04). The difference in Accuracy (0.90%) and Execution Time (0.90 seconds) between the two classifiers was marginal.

REFERENCES

Sunyaev, A. Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies; Springer International Publishing: Cham, Switzerland, 2020. doi:10.1007/978-3-030-34957-8

Shahid MA, Alam MM, Su'ud MM. Performance Evaluation of Load-Balancing Algorithms with Different Service Broker Policies for Cloud Computing. Applied Sciences. 2023;13: 1586. doi:10.3390/app13031586

Anjum N, Chowdhury MR. International Journal of Advanced Research in Computer and Communication Engineering. SSRN Journal. 2024 [cited 2 Oct 2025]. doi:10.2139/ssrn.4847308

Applied Sciences | Free Full-Text | A Systematic Parameter Analysis of Cloud Simulation Tools in Cloud Computing Environments. [cited 22 Feb 2024]. Available: https://www.mdpi.com/2076-3417/13/15/8785

Shahid MA, Islam N, Alam MM, Mazliham MS, Musa S. Towards Resilient Method: An exhaustive survey of fault tolerance methods in the cloud computing environment. Computer Science Review. 2021;40: 100398. doi:10.1016/j.cosrev.2021.100398

Arabnejad H, Pahl C, Estrada G, Samir A, Fowley F. A Fuzzy Load Balancer for Adaptive Fault Tolerance Management in Cloud Platforms. In: De Paoli F, Schulte S, Broch Johnsen E, editors. Service-Oriented and Cloud Computing. Cham: Springer International Publishing; 2017. pp. 109–124. doi:10.1007/978-3-319-67262-5_9

Mukwevho MA, Celik T. Toward a Smart Cloud: A Review of Fault-Tolerance Methods in Cloud Systems. IEEE Trans Serv Comput. 2021;14: 589–605. doi:10.1109/TSC.2018.2816644

Jain P. A DYNAMIC PROCESS FOR FAULT TOLERANCE TECHNIQUES IN CLOUD COMPUTING (DPFT). J. Gujrat Res. Soc. 2019;21.

Khaldi M, Rebbah M, Meftah B, Smail O. Fault tolerance for a scientific workflow system in a Cloud computing environment. International Journal of Computers and Applications. 2020;42: 705–714. doi:10.1080/1206212X.2019.1647651

Mesbahi MR, Rahmani AM, Hosseinzadeh M. Reliability and high availability in cloud computing environments: a reference roadmap. Hum Cent Comput Inf Sci. 2018;8: 20. doi:10.1186/s13673-018-0143-8

Zhou A, Wang S, Cheng B, Zheng Z, Yang F, Chang RN, et al. Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization. IEEE Trans Serv Comput. 2017;10: 902–913. doi:10.1109/TSC.2016.2519898

Weibull Distribution - an overview | ScienceDirect Topics. [cited 2 Oct 2025]. Available: https://www.sciencedirect.com/topics/physics-and-astronomy/weibull-distribution

Zhang L, Wen J, Li Y, Chen J, Ye Y, Fu Y, et al. A review of machine learning in building load prediction. Applied Energy. 2021;285: 116452. doi:10.1016/j.apenergy.2021.116452

Shahid MA, Islam N, Alam MM, Su'ud MM, Musa S. A Comprehensive Study of Load Balancing Approaches in the Cloud Computing Environment and a Novel Fault Tolerance Approach. IEEE Access. 2020;8: 130500–130526. doi:10.1109/ACCESS.2020.3009184

Gupta V, Kaur BP, Jangra S. An efficient method for fault tolerance in cloud environment using encryption and classification. Soft Comput. 2019;23: 13591–13602. doi:10.1007/s00500-019-03896-6

Preemptive Fault Tolerance in DDS based Distributed System using Application Migration. IJRASET. 2020;8: 963–970. doi:10.22214/ijraset.2020.29240

Goundar Sam, Bhardwaj AD. Efficient Fault Tolerance Cloud Environments. International Journal of Cloud Applications and Computing 2018;8 20-31 doi: 10.4018/IJCAC.2018070102

Edemo MK. DEVELOPING FAULT TOLERANCE ARCHITECTURE FOR REAL-TIME SYSTEMS OF CLOUD COMPUTING.

Feng D-C, Liu Z-T, Wang X-D, Chen Y, Chang J-Q, Wei D-F, et al. Machine learning-based compressive strength prediction for concrete: An adaptive boosting approach. Construction and Building Materials. 2020;230: 117000. doi:10.1016/j.conbuildmat.2019.117000

Butt UA, Mehmood M, Shah SBH, Amin R, Shaukat MW, Raza SM, et al. A Review of Machine Learning Algorithms for Cloud Computing Security. Electronics. 2020;9: 1379. doi:10.3390/electronics9091379

Pei X, Yuan M, Mao G, Pang Z. Application of multivariate time-series model for high performance computing (HPC) fault prediction. Ijaz MF, editor. PLoS ONE. 2023;18: e0281519. doi:10.1371/journal.pone.0281519

Shrestha A, Mahmood A. Review of Deep Learning Algorithms and Architectures. IEEE Access. 2019;7: 53040–53065. doi:10.1109/ACCESS.2019.2912200

Lang S, Bravo-Marquez F, Beckham C, Hall M, Frank E. WekaDeeplearning4j: A deep learning package for Weka based on Deeplearning4j. Knowledge-Based Systems. 2019;178: 48–50. doi:10.1016/j.knosys.2019.04.013

Wang S, Jiang L, Li C. Adapting naive Bayes tree for text classification. Knowl Inf Syst. 2015;44: 77–89. doi:10.1007/s10115-014-0746-y

Bildosola I, Río-Belver R, Cilleruelo E, Garechana G. Design and Implementation of a Cloud Computing Adoption Decision Tool: Generating a Cloud Road. Suleman H, editor. PLoS ONE. 2015;10: e0134563. doi:10.1371/journal.pone.0134563

Zhang L, Jiang L, Li C, Kong G. Two feature weighting approaches for naive Bayes text classifiers. Knowledge-Based Systems. 2016;100: 137–144. doi:10.1016/j.knosys.2016.02.017

Liu J, Wu Z, Wu J, Dong J, Zhao Y, Wen D. A Weibull distribution accrual failure detector for cloud computing. Song H, editor. PLoS ONE. 2017;12: e0173666. doi:10.1371/journal.pone.0173666

Madni SHH, Abd Latiff MS, Abdullahi M, Abdulhamid SM, Usman MJ. Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment. Choo K-KR, editor. PLoS ONE. 2017;12: e0176321. doi:10.1371/journal.pone.0176321

Tanha J, Van Someren M, Afsarmanesh H. Semi-supervised self-training for decision tree classifiers. Int J Mach Learn & Cyber. 2017;8: 355–370. doi:10.1007/s13042-015-0328-7

Patel HH, Prajapati P. Study and Analysis of Decision Tree Based Classification Algorithms. ijcse. 2018;6: 74–78. doi:10.26438/ijcse/v6i10.7478

Mesbahi MR, Rahmani AM, Hosseinzadeh M. Reliability and high availability in cloud computing environments: a reference roadmap. Hum Cent Comput Inf Sci. 2018;8: 20. doi:10.1186/s13673-018-0143-8

Attallah SMA, Fayek MB, Nassar SM, Hemayed EE. Proactive load balancing fault tolerance algorithm in cloud computing. Concurrency and Computation. 2021;33: e6172. doi:10.1002/cpe.6172

Suguna S, Devi K. VMFT: VIRTUAL MACHINE FAULT TOLERANCE IN CLOUD COMPUTING.

Santos SGTDC, De Barros RSM. Online AdaBoost-based methods for multiclass problems. Artif Intell Rev. 2020;53: 1293–1322. doi:10.1007/s10462-019-09696-6

Breiman L. Bagging predictors. Mach Learn. 1996;24: 123–140. doi:10.1007/BF00058655

Charbuty B, Abdulazeez A. Classification Based on Decision Tree Algorithm for Machine Learning. JASTT. 2021;2: 20–28. doi:10.38094/jastt20165

Guo Y, Cao X, Liu B, Gao M. Solving Partial Differential Equations Using Deep Learning and Physical Constraints. Applied Sciences. 2020;10: 5917. doi:10.3390/app10175917

Department of Mathematics and Computer Science, University of Missouri-St. Louis, Missouri, USA., Vangara* RVB, Thirupathur K, Department of Computer Science, University of Bridgeport, Connecticut, USA., Vangara SP, Department of Information Systems, Indiana Tech University, Indianapolis, USA. Opinion Mining Classification u sing Naive Bayes Algorithm. IJITEE. 2020;9: 495–498. doi:10.35940/ijitee.E2402.039520

Amoon M. Adaptive Framework for Reliable Cloud Computing Environment. IEEE Access. 2016;4: 9469–9478. doi:10.1109/ACCESS.2016.2623633

Merlini D, Rossini M. Text categorization with WEKA: A survey. Machine Learning with Applications. 2021;4: 100033. doi:10.1016/j.mlwa.2021.100033

Bilal A, Hasany SMN, Pitafi AH. Effective modelling of sinkhole detection algorithm for edge-based Internet of Things (IoT) sensing devices. IET Communications. 2022;16: 845–855. doi:10.1049/cmu2.12385