

ANALYZING THE EFFECTIVENESS OF DECISION TREE AND NAÏVE BAYES CLASSIFIERS IN CLOUD COMPUTING ENVIRONMENTS

Muhammad Asim Shahid^{*1}, Humera Azam², Imran³^{*1}College of Computing and Software Engineering, Ziauddin University, Karachi, Pakistan²Department of Computer Science, University of Karachi, Karachi, Pakistan³Department of Computer Science, Bahria University, Karachi, Pakistan^{*1}asim.shahid@zu.edu.pk, ²humera.azam@uok.edu.pk, ³imranali.bukc@bahria.edu.pkDOI: <https://doi.org/10.5281/zenodo.17854624>**Keywords**

Cloud computing, Fault classification and prediction, Machine learning algorithm, Weibull distribution

Article History

Received: 13 February 2025

Accepted: 10 March 2025

Published: 26 March 2025

Copyright @Author

Corresponding Author: *

Muhammad Asim Shahid

Abstract

CC has been growing rapidly in recent years due to its cost-effectiveness and efficient resource management. With this huge growth in the last ten years, many organizations have shifted their operations online for better reach, flexibility, and visibility. This research will do a comparison of two different machine learning techniques: Decision Tree and Naive Bayes Tree, based on precision and error detection. The output shows that NB Tree performed better with 97.05% correct predictions for an 80/20 split, 96.09% for a 70/30 split, and 96.78% for a 10-round cross-validation. But NB Tree is slower and took approximately 1.01 seconds because it involves more complexity in its algorithm. On the other side, the accuracy of the Decision Tree (J48) is comparatively low, at 96.78%, 95.95%, and 96.78% for the splits mentioned above, while taking significantly lesser time of 0.11 seconds. NB Tree outperformed J48 by about 0.9% in terms of precision, whereas J48 completed the analysis in about 9 seconds faster than NB Tree.

INTRODUCTION**Research Background & Motivation**

With the exponential growth in CC, hundreds of companies have implemented CC in the last ten years to promote accessibility, mobility, and sharability. More organizations are now moving workloads to the online environment where service providers manage the technical configuration to reduce operating costs and underutilization of assets. In these environments, multiple users share computing resources comprising software applications, physical devices, and virtual platforms. The main architectural layers in cloud systems include IaaS, SaaS, and PaaS. The manifestation of defects at these layers

can significantly impact durability, which makes early detection and removal critical for better performance. All customer resources are migrated to the cloud provider who becomes responsible for performing data processing activities. While smart devices offer value-added services, they also facilitate on-demand accessibility based on user needs from anywhere in the world.

These data are fitted using the Weibull distribution, which is often used in reliability studies for representing time to failure. Its capability for modeling different shapes of failure rates also allows it to show increasing or decreasing risk trends with time, thus enabling the

identification of aging-related degradations or initial defects.

Machine learning has great potential for the study of building energy use as data become more available and methodological developments continue to improve. The field, at the intersection of statistics and computation, is critically important for data analysis and smart system deployment. Approaches are generally divided into supervised and unsupervised learning. Supervised models are trained on examples with labeled inputs paired with expected outputs; this allows the model to make predictions based on input data. Unsupervised learning uses unlabeled data, allowing for the discovery of hidden structures or patterns within the data. Primary challenges for the latter paradigm involve clustering similar items and identifying associations between those items.

Research Problems

To enhance the impact of the article, it is important to find the gaps in previous work. A critical review of related previous work points out several deficiencies: [7]

- Consistency remains an issue in decision tree CC models that is yet to be resolved and warrants scholarly attention [7].
- The literature suggests that the improvement of cloud node reliability requires a machine learning-based approach, though other approaches exist.
- A new approach is required for improving accuracy, reducing the rate of errors, and ensuring stability in the performance of machine learning in cloud systems [9].

Research Goals

This research paper intends to utilize machine learning to enhance the accuracy and reduce forecasting errors, hence improving the performance of the service in cloud computing. Precisely, the following are the aims of this paper:

- Research the degree to which decision trees can enhance reliability metrics in CC configurations.

- Identify the main factors that influence CC reliability and assess the predictive accuracy of decision trees for these factors.

- Assess the various machine learning approaches to predict reliability issues in cloud environments by comparing decision trees with Naive Bayes. When the examined models are compared, the focus is on practical performance rather than on relative theoretical strengths. Because one approach may be stronger than another based on data structure or pattern of system behavior, comparing both in the same conditions often generates more instructive insights into real-world applicability.

Research Scope

Compliant behavior was recorded during the specified study periods:

- Data collection would be done for the application of machine learning in fault tolerance within cloud computing. The main dataset will include information about the failure of virtual machines and the repair actions taken accordingly.
- Machine learning techniques are applied in this field to improve the accuracy and reduce errors in fault forecasting. The supervised learning approaches provide the models with an accurate identification of patterns, leveraging the examples for which the outcome is already known. Training these systems enables them to make sound predictions based on previously labeled data.

Literature Review

Shahid et al. [10] argue that CC represents a major paradigmatic shift, connecting isolated systems into extensive networks. Companies like IBM, Amazon, and Yahoo—as well as Google—offer cloud services on the global level, allowing users to access applications and utilities according to their needs, thus eliminating the need for a local setup within one's computer.

Shahid et al. [2] review how cloud platforms support sharing computing resources such as programs, machines, and code across multiple users. The key architectural layers include IaaS, PaaS, and SaaS. For assured dependability, error

handling is implemented at the software layer to handle different kinds of failures in these tiers.

Lang et al. [11] present that deep learning, a class of machine learning, is based on artificial neural networks to build representations of data in layers, which improves performance on image classification, speech recognition, object detection, and document classification. WekaDeeplearning4j is a tool implemented in Weka, which provides an easy-to-use interface for users to train complex networks, such as convolutional or recurrent ones, by using the Deeplearning4j engine. It also provides utilities for the preprocessing of textual and visual inputs, including GPU usage.

Wang et al. [12] apply a multinomial naïve Bayes (MNB) classifier at every leaf of a decision tree, introducing a new model they call MNBTree. Unlike NBTree, here features are split into zero and nonzero entries, making the structure binary. It does not aim to optimize the classification performance at each node but instead uses information gain to make it build trees faster. By employing a multiclass strategy, an enhanced version-namely MMNBTree-has been developed, attaining significant uplift in predictive performance. Their empirical studies on various benchmark text classification datasets have shown that both MNBTree and MMNBTree work very efficiently.

Feng et al. [13] propose the application of machine learning to estimate the compressive strength of concrete. This technique constructs a robust predictor by aggregating several less accurate models through adaptive boosting, hence improving accuracy in an iterative series of corrections. The procedure updates the weights depending on the previous errors, placing increasing emphasis on the more difficult examples.

Pei et al. [14] propose a system called CBA-net, which integrates CNN, BiLSTM, and attention mechanisms. It uses HDBSCAN to cluster historical fault records by location and time. Given the fact that this method captures temporal patterns with ease, it reached a precision of 93% in fault identification and showed great performance in timing estimates, with an RMSE

as low as 0.031. Due to rapid convergence during training, this approach enhances detailed failure forecasting in large-scale computing environments.

The work of Shrestha & Mahmood [15] delves into methods that accelerate the training process while gaining precision in the model within deep learning. While focusing on standard techniques, they discuss the underlying mathematics, limitations, extant progress, and real-world applications. Various network architectures are studied, including convolution-based systems, feedback-driven models, skip-connections, generative frameworks, and decision-oriented agents.

Bildosola et al. [16] present a pragmatic decision-support instrument for the adoption of clouds, using structured questionnaires to capture data while guiding early-stage cloud endeavors, especially those based on SaaS models. This method supports executives in creating empirically-based Cloud Roadmaps. Instead of assumptions, testing in the real world was done through a small-scale pilot with regional companies to evaluate awareness of CC and also to determine industries that are potentially and implementably viable.

Mesbahi et al. [17] propose WekaDeeplearning4j-a tool within Weka that enables deep learning through a graphical user interface. Instead of writing code, the user interacts graphically while the Deeplearning4j backend carries out the training of complex models, including convolutional and recurrent networks. The system preprocesses textual and visual inputs in advance, with the possibility of using GPU support for computation where available.

Patel & Prajapati [18] argue that machine learning is the process of letting systems develop skills themselves by exposing them to a lot of data instead of explicitly encoding the fixed programs. Within this area, decision trees stand out because they work on a principle that makes decisions by partitioning data according to features. The models are found across domains like medical diagnosis, information retrieval, document processing, and quantitative research. Due to performance concerns, over time, several variants

of the underlying method were developed. Three popular variants, namely ID3, with the entropy measures, its successor C4.5, which refines splits using gain ratio, and CART-known for binary partitions with Gini impurity-are considered in this paper.

Tanha et al. [19] note that some machine learning methods, among them semi-supervised learning, rely on both labeled and unlabeled data. Conventional decision tree learners, though, cannot support effective self-training due to their limited ability to estimate prediction probabilities. The authors investigate a number of enhancements, such as grafting, distance-based measures, Naive Bayes trees, and embedding Laplace correction without pruning. They scale these enhancements up to decision tree ensembles, which outperform the improved single-tree variants, encouraging further development.

Qiu et al. [20] review current machine learning work dealing with high dimensional data. From state-of-the-art techniques, the authors consider new notions currently under assessment. Those range from methods to learn informative representations, train deep networks, share tasks across systems, run processes in parallel, transfer knowledge between domains, select pivotal samples during training, and successfully make use of kernel functions. The paper goes on to discuss challenges arising when such models have to be deployed on large-scale data, after which

options with greater detail to minimize such issues are provided.

Problem Statement

Decision trees are seeing more use in cloud systems, yet effects on stability aren't clear. This work explores their role in boosting reliability measures within cloud setups. Looking at strengths and weaknesses, it seeks to offer useful guidance for providers aiming to increase uptime while lowering risks - using actual cases and tested examples [17].

Research Methodology

The present part explains classification methods, research setup, ways to gather information, approaches for analyzing data alongside suggested models. Each aspect is covered clearly without necessary details.

Study Design

This part presents how faults are grouped and the model used for predictions. Main data sets serve to develop the model, besides supporting fault detection and labeling. The aim is finding which machine learning classifiers give higher accuracy in forecasting, along with solid performance during testing on varied groups, while also reducing mistakes in predicting faults.

This part also outlines how data was created and gathered during the study. As seen in Fig. 1, the setup of the research model is displayed.

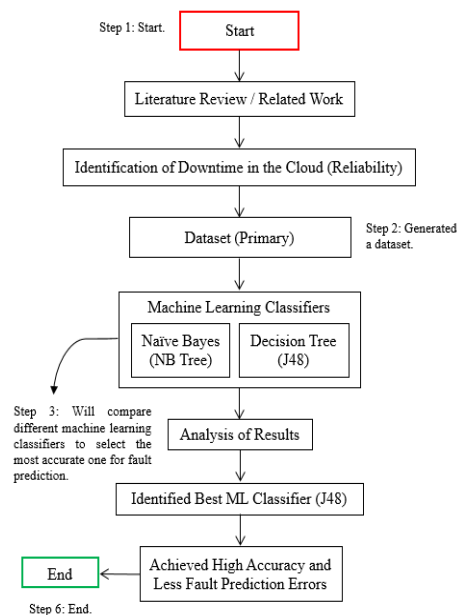


Fig. 1 An illustration of the research framework from an implementation perspective.

Generated Primary Data

The Weibull distribution plays an essential role in predicting how long systems last before failing. Unlike the exponential model, it handles changing failure rates, capturing initial defects as well as aging effects. Its flexibility allows rising or falling

risk patterns over time [9]. Researchers applied this model to simulate data using varied settings from the Java platform. Parameter details are collected in Table 1, whereas the resulting dataset appears in Table 2.

Table 1. An Overview of the Main Factors that Produced the Data

User	Port No	Host No	Network Host	Distribution
1	32	384	Ram, Storage, Bandwidth, & MIPS	The Weibull distribution models a system's failure rate over time, showing both rising and falling rates.

Table 2. A Concise Overview of the Primary Dataset

FHID	HFTIME	LFT	DIS	FTIME/RTIME	STATUS
456	1	38455	Weibull	10568	Failure
456	1	38455	Weibull	12464	Repair
170	2	24432	Weibull	7236	Failure
170	2	24432	Weibull	7236	Repair
224	2	18201	Weibull	18036	Failure
224	2	18201	Weibull	18036	Repair
335	1	15468	Weibull	22894	Failure
335	1	15468	Weibull	22894	Repair

Table 2 presents the main data set - 700 instances featuring six characteristics: FHID, HFT, LFT, Dis, FTime/RTime, along with Status; these cover numeric and categorical values.

Machine Learning Classifiers

This research detects issues by applying different machine learning methods. While using algorithms like J48 and NBTree, it sorts faults into groups for better forecasting.

- **Decision Tree (J48):** J48 is widely applied in areas like pattern recognition, image analysis, or machine learning. This method runs a sequence of checks - each evaluating a numeric feature against a set threshold. In contrast to

neural networks, these trees offer more transparent reasoning behind decisions. They're mainly used for sorting data into categories, which makes them a frequent choice within data mining tasks. Every node stands for traits that need classification [21]; meanwhile, edges reflect possible trait values. Formula 1 and formula 2 present measures for disorder and useful splits [22].

$Entropy(S) = \sum_{i=1}^c P_i \log 2^{P_i}$	(1)
$Gain(S, A) = \sum_{v \in S_V(A)} \frac{S_v}{S} Entropy(S_v)$	(2)

- **Naïve Bayes Tree:** NBTree is a type of supervised machine learning that uses Bayes' rule to work out conditional likelihoods. It's often applied in sentiment classification tasks. Instead of guessing, it estimates how likely each label fits the given text. Then, it picks the one with highest score [23]. First, merge some texts per category based on how likely word n appears in class m given its frequency w [23].

$$p(m) \propto \pi_m \prod_{n=1}^{|v|} p(n|m) w_n \quad (3)$$

Establishing Parameters for the Machine Learning Classifier

The setup of different ML classifiers, along with their value settings, appears in Table 3 - this configuration aims at predicting faults using accuracy and class distinctions.

Table 3. Establishing Parameters for the Machine Learning Classifier

ML Classifiers	Establishing Parameters	Values
Decision Tree (J48)	Batch size	100
	Binary splits	False
	Collapse tree	True
	Confidence factor	0.25
	Debug	False
	Do not check capabilities	False
	Do not make the split point the actual value	False
	Min num obj	2
	Num decimal places	2
	Num folds	3
	Reduced error pruning	False
	Save instance data	False

	Seed	1
	Subtree raising	True
	Unpruned	False
	Use la place	False
	Use MDL correction	True
	Set the cache mode	Memory
	Data queue size	0
	Resume	False
	Preserve file system cache	False
	Number of GPUs	1
	Size of per-fetch buffer for multiple GPUs	24
	Model parameter averaging frequency	10
	Batch size	100
	Debug	False
	Do not check capabilities	False
	Num decimal places	2
	Seed	1
Naïve Bayes Tree (NBTree)	Batch size	100
	Debug	False
	Do not check capabilities	False
	Num decimal places	2

Experiments & Results

This part shows findings from analyzing data through different machine learning approaches - such as NBTree or J48 - and includes a confusion matrix for evaluation.

This research seeks accurate results while reducing mistakes in forecasting faults through standard ML methods - using established approaches instead of complex models - to improve reliability without extra complexity.

Using 80/20 splits, alongside 70/30 and tenfold validation methods, NBTree together with J48

processed the main dataset, yielding acceptable classification outcomes. For assessing performance, metrics such as validation reliability, prediction mistakes, also category precision were computed via formulas 4 through 14. Minor differences emerged in output scores when comparing NBTree against J48 within the original data. Still, J48 performs better regarding time efficiency. Analysis included the Remove Percentage Filter, set up in WEKA version 3.8.6.

Accuracy

$$\frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

The definition of accuracy in EQ 4 is provided as follows.

True-Positive Rate

$$\frac{TP}{TP + FN} \quad (5)$$

The recall, also known as the true positive rate, is defined in Equation 5 as follows.

True-Negative Rate =

$$\frac{TN}{TN + FP} \quad (6)$$

The true negative rate in Equation 6 is defined as follows.

Precision =

$$\frac{TP}{TP + FP} \quad (7)$$

The definition of accuracy in EQ 7 is as follows.

$$\text{False-Positive Rate} = \frac{FP}{TN + FP} \quad (8)$$

The false positive rate is defined in Equation 8 as follows.

$$\text{MCC} = \frac{FP}{TN + FP} \quad (9)$$

The Matthews correlation coefficient is defined in Equation 9 as follows.

$$\text{F-Measure} = \frac{2PPV \times TPR}{PPV + TPR} \quad (10)$$

The F-measure is defined in Equation 10 as follows.

- The root mean square error (RMSE) measures differences between predicted and observed values in a model, often applied in evaluation tasks [24].
- The Mean Absolute Error (MAE) measures how far apart two continuous values are [24].
- Relative absolute error divides overall mistake size by that of a basic prediction method, making it comparable across cases [25].
- The relative squared error (RSE) adjusts the overall squared mistake using division by that of a basic prediction method [25].

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (11)$$

The RMSE is defined in equation 11.

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (12)$$

The MAE is defined in Equation 12.

$$E_i = \frac{\sum_{j=1}^n |P_{(ij)} - T_j|}{\sum_{j=1}^n |T_j - \bar{T}|} \quad (13)$$

The Relative Average Error (RAE) is defined in equation 13 as follows.

$$E_i = \sqrt{\frac{\sum_{j=1}^n (P_{(ij)} - T_j)^2}{\sum_{j=1}^n (T_j - \bar{T})^2}} \quad (14)$$

The RSE is defined in equation 14 as follows.

Comparing Models Classification Based on a Primary Dataset

Preliminary findings show classifier performance on the STATUS category, drawn from the main data set via NBTree combined with J48.

The primary findings show NBTree outperforms 80/20 split with 97.05% accuracy, while also exceeding 70/30 at 96.09%, along with 10-fold CV scoring 96.78%, especially regarding predictive precision; still, its processing duration

hits 1.01 seconds - slightly inefficient despite strong performance.

In terms of accuracy for fault prediction, J48 comes in second place - reaching 96.78% using the 80/20 split, 95.95% with 70/30, while also scoring 96.78% under 10-fold cross-validation. Despite this, its processing time is only 0.11 seconds; making it notably fast.

Although the two classifiers differ by 9 seconds in processing speed, their performance gap in

accuracy and error detection - NBTree versus J48 - is just 0.9%. Despite this time mismatch, predictive outcomes remain nearly identical across both models.

A close look at how accurate the J48 and NBTree models are can be found here - results break down

performance per category (Repair or Failure), along with outcomes from the test subset. To verify these results further, check Figures 2 and 3 located in the Primary Dataset

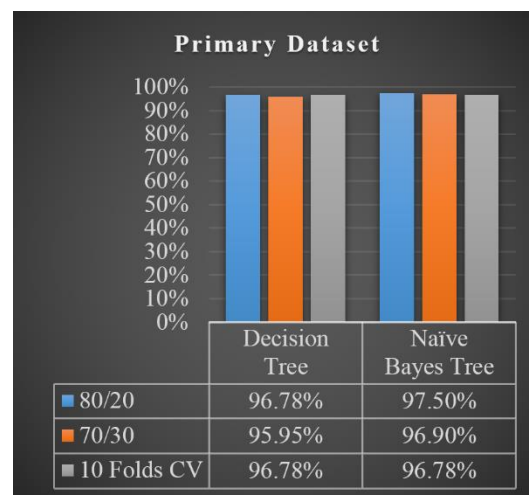


Fig. 2 Accuracy of the primary dataset on J48 & NBTree by class (failure/repair).

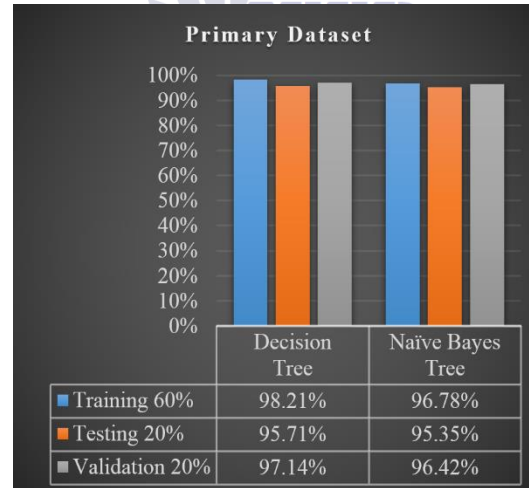


Fig. 3 Accuracy of the primary dataset for J48 and NBTree related to data validation (failure or repair).

The confusion matrix helps compute main evaluation scores - F-Measure, Accuracy, Precision, and Recall - not just through summation. It sorts outcomes by actual versus predicted labels. For accuracy and error detection, J48 and NBTree produced separate confusion tables; these appear

in Figure 4 and Figure 5. Of the two, NBTree achieves top accuracy but underperforms slightly in detecting faults within the central data collection. Still, its computational demand remains low - processing time was only 1.01 seconds, well under typical load.

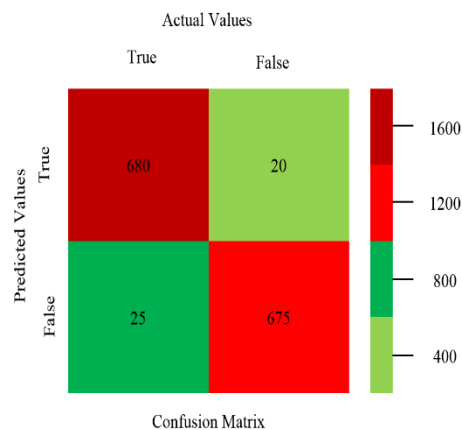


Fig. 4 Confusion matrix of the J48 classifier for accuracy and fault prediction using the primary dataset.

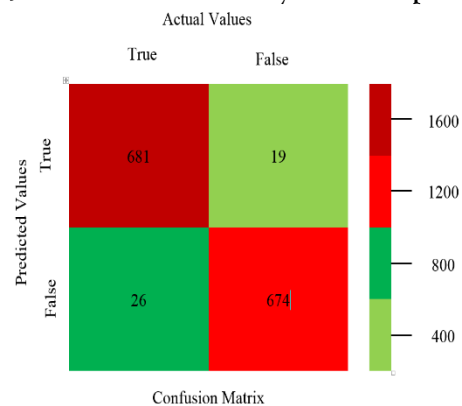


Fig. 5 Confusion matrix of the NBTree classifier for accuracy and fault prediction using the primary dataset.

The J48 method shows strong performance with minimal mistakes, finishing in 0.11 seconds; by contrast, NBTree requires an extra nine seconds although its accuracy improves by merely 0.9%. As shown in Figures 6 and 7, correct

classifications and incorrect ones - both positive and negative - are displayed clearly, pointing out where predictions fail and how outcomes differ from real labels.

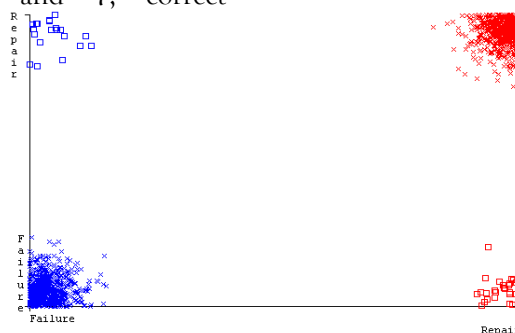


Fig. 6 Inaccuracies of the J48 classifier in accuracy and fault prediction are based on primary data.

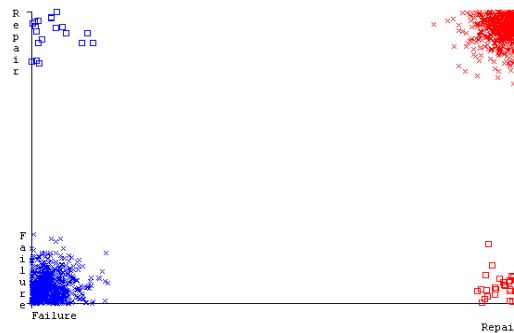


Fig. 7 Inaccuracies of the NBTree classifier in accuracy and fault prediction are based on primary data.

Discussion

This research sought to lower mistakes without sacrificing precision or consistency - using different methods to improve outcomes instead of relying on standard approaches.

The experiment is running again using WEKA. This tool groups various machine learning methods together, supporting tasks like cleaning data, grouping items, predicting values, finding patterns, detecting clusters, along with visual displays. Being open source, it's shared under GNU's public license. Performance stays strong during development of fresh ML setups. Users may apply its tools straight to datasets or embed them into custom Java programs [18].

The results indicate the NBTree model delivers top performance, reaching 97.05% accuracy on an 80/20 split; meanwhile, it scores 96.09% on a 70/30 setup - while achieving 96.78% through 10-fold validation. This approach also shows the smallest error frequency, coupled with a fast runtime of just 1.01 seconds.

In comparison, J48 shows the second-best accuracy - 96.78% under 10-fold cross-validation and 80/20 split; however, it drops to 94.95% for 70/30 - with processing time at just 0.9 seconds. Though NBTree edges ahead by 0.9% in precision, J48 matches closely on performance but uses nearly identical runtime.

Conclusion & Future Work

In this work, one researcher looked closely at earlier studies. Searches were done on platforms like Google Scholar, Web of Science, or Science Direct - key terms helped locate suitable papers. Terms involved ML classifiers; methods used for

spotting faults in CC also came up frequently. Some focused on widely-used models known to improve precision while lowering mistakes in forecasts.

- The Weibull distribution may be applied via a visual interface to create the primary data set for CloudSim.
- Although code may adapt itself to improve settings, stopping it during the process can prevent ideal results from being reached.
- High accuracy with few mistakes is possible via NBTree; yet more work must explore its computational demands. The goal here is comparing performance across methods.
- Deep learning models often predict outcomes more precisely while reducing mistakes. Still, they need substantial data to deliver dependable findings - small sets may fail them. If enough information is available, these approaches usually work better than classic machine learning alternatives.

REFERENCES

- [1] M. A. Shahid, M. M. Alam, and M. M. Su'ud, "Performance Evaluation of Load-Balancing Algorithms with Different Service Broker Policies for Cloud Computing," *Applied Sciences*, vol. 13, no. 3, p. 1586, Jan. 2023, doi: 10.3390/app13031586.

- [2] M. A. Shahid, N. Islam, M. M. Alam, M. S. Mazliham, and S. Musa, "Towards Resilient Method: An exhaustive survey of fault tolerance methods in the cloud computing environment," *Computer Science Review*, vol. 40, p. 100398, May 2021, doi: 10.1016/j.cosrev.2021.100398.
- [3] M. A. Shahid, M. M. Alam, and M. M. Su'ud, "A Systematic Parameter Analysis of Cloud Simulation Tools in Cloud Computing Environments," *Applied Sciences*, vol. 13, no. 15, Art. no. 15, Jan. 2023, doi: 10.3390/app13158785.
- [4] Sir Syed University of Engineering and Technology Karachi, Pakistan et al., "A Systematic Survey of Simulation Tools for Cloud and Mobile Cloud Computing Paradigm," *JISR-C*, vol. 20, no. 1, Jun. 2022, doi: 10.31645/JISRC.22.20.1.10.
- [5] M. A. Shahid, M. M. Alam, and M. M. Su'ud, "Achieving Reliability in Cloud Computing by a Novel Hybrid Approach," *Sensors*, vol. 23, no. 4, p. 1965, Feb. 2023, doi: 10.3390/s23041965.
- [6] L. Zhang et al., "A review of machine learning in building load prediction," *Applied Energy*, vol. 285, p. 116452, Mar. 2021, doi: 10.1016/j.apenergy.2021.116452.
- [7] Z. Ullah, F. Al-Turjman, L. Mostarda, and R. Gagliardi, "Applications of Artificial Intelligence and Machine learning in smart cities," *Computer Communications*, vol. 154, pp. 313–323, Mar. 2020, doi: 10.1016/j.comcom.2020.02.069.
- [8] A. Netti, Z. Kiziltan, O. Babaoglu, A. Sirbu, A. Bartolini, and A. Borghesi, "Antarex HPC Fault Dataset." Zenodo, Oct. 10, 2018. Accessed: Jul. 02, 2025. [Online]. Available: <https://zenodo.org/records/1453949>
- [9] "Weibull Distribution - an overview | ScienceDirect Topics." Accessed: Jul. 02, 2025. [Online]. Available: <https://www.sciencedirect.com/topics/physics-and-astronomy/weibull-distribution>
- [10] M. A. Shahid, N. Islam, M. M. Alam, M. M. Su'ud, and S. Musa, "A Comprehensive Study of Load Balancing Approaches in the Cloud Computing Environment and a Novel Fault Tolerance Approach," *IEEE Access*, vol. 8, pp. 130500–130526, 2020, doi: 10.1109/ACCESS.2020.3009184.
- [11] S. Lang, F. Bravo-Marquez, C. Beckham, M. Hall, and E. Frank, "WekaDeeplearning4j: A deep learning package for Weka based on Deeplearning4j," *Knowledge-Based Systems*, vol. 178, pp. 48–50, Aug. 2019, doi: 10.1016/j.knosys.2019.04.013.
- [12] S. Wang, L. Jiang, and C. Li, "Adapting naive Bayes tree for text classification," *Knowl Inf Syst*, vol. 44, no. 1, pp. 77–89, Jul. 2015, doi: 10.1007/s10115-014-0746-y.
- [13] D.-C. Feng et al., "Machine learning-based compressive strength prediction for concrete: An adaptive boosting approach," *Construction and Building Materials*, vol. 230, p. 117000, Jan. 2020, doi: 10.1016/j.conbuildmat.2019.117000.
- [14] X. Pei, M. Yuan, G. Mao, and Z. Pang, "Application of multivariate time-series model for high performance computing (HPC) fault prediction," *PLoS ONE*, vol. 18, no. 10, p. e0281519, Oct. 2023, doi: 10.1371/journal.pone.0281519.
- [15] A. Shrestha and A. Mahmood, "Review of Deep Learning Algorithms and Architectures," *IEEE Access*, vol. 7, pp. 53040–53065, 2019, doi: 10.1109/ACCESS.2019.2912200.
- [16] I. Bildosola, R. Río-Belver, E. Cilleruelo, and G. Garechana, "Design and Implementation of a Cloud Computing Adoption Decision Tool: Generating a Cloud Road," *PLoS ONE*, vol. 10, no. 7, p. e0134563, Jul. 2015, doi: 10.1371/journal.pone.0134563.
- [17] M. R. Mesbahi, A. M. Rahmani, and M. Hosseinzadeh, "Reliability and high availability in cloud computing environments: a reference roadmap," *Hum. Cent. Comput. Inf. Sci.*, vol. 8, no. 1, p. 20, Dec. 2018, doi: 10.1186/s13673-018-0143-8.

- [18] H. H. Patel and P. Prajapati, "Study and Analysis of Decision Tree Based Classification Algorithms," *ijcse*, vol. 6, no. 10, pp. 74–78, Oct. 2018, doi: 10.26438/ijcse/v6i10.7478.
- [19] J. Tanha, M. Van Someren, and H. Afsarmanesh, "Semi-supervised self-training for decision tree classifiers," *Int. J. Mach. Learn. & Cyber.*, vol. 8, no. 1, pp. 355–370, Feb. 2017, doi: 10.1007/s13042-015-0328-7.
- [20] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, "A survey of machine learning for big data processing," *EURASIP J. Adv. Signal Process.*, vol. 2016, no. 1, p. 67, Dec. 2016, doi: 10.1186/s13634-016-0355-x.
- [21] L. Breiman, "Bagging predictors," *Mach Learn*, vol. 24, no. 2, pp. 123–140, Aug. 1996, doi: 10.1007/BF00058655.
- [22] B. Charbuty and A. Abdulazeez, "Classification Based on Decision Tree Algorithm for Machine Learning," *JASTT*, vol. 2, no. 01, pp. 20–28, Mar. 2021, doi: 10.38094/jastt20165.
- [23] Department of Mathematics and Computer Science, University of Missouri-St. Louis, Missouri, USA., R. V. B. Vangara*, K. Thirupathur, Department of Computer Science, University of Bridgeport, Connecticut, USA., S. P. Vangara, and Department of Information Systems, Indiana Tech University, Indianapolis, USA., "Opinion Mining Classification using Naive Bayes Algorithm," *IJITEE*, vol. 9, no. 5, pp. 495–498, Mar. 2020, doi: 10.35940/ijitee.E2402.039520.
- [24] "GMD - Root-mean-square error (RMSE) or mean absolute error (MAE): when to use them or not." Accessed: Jul. 02, 2025. [Online]. Available: <https://gmd.copernicus.org/articles/15/5481/2022/>
- [25] "Relative Absolute Error." Accessed: Jul. 02, 2025. [Online]. Available: <https://www.gepsoft.com/GeneXproTools/AnalysesAndComputations/MeasuresOffit/RelativeAbsoluteError.htm>