

Prediction of Application Performance Issues Using Decision Tree Models

¹Efren I. Balaba, ²Elise Jane M. Lacubtan

¹Department of BSIT, Southern Leyte State University, Tomas Oppus Campus, Tomas Oppus, Southern Leyte 6605, Philippines

²Southern Leyte State University, Tomas Oppus Campus, Tomas Oppus, Southern Leyte 6605, Philippines

About the Article

 Open Access

Research Article

How to Cite:

E. I. Balaba and E. J. M. Lacubtan, "Prediction of application performance issues using decision tree models," *Insights Comput. Sci.*, Vol. 2, pp. 13-17, 2026.

Keywords:

AIOps, app performance prediction, CART algorithm, decision tree, pro-active maintenance, software reliability

Corresponding author:

Efren I. Balaba,
Department of BSIT, Southern Leyte State University, Tomas Oppus Campus, Tomas Oppus, Southern Leyte 6605, Philippines

ABSTRACT

Objective: The rapid digitalization of global services has elevated software reliability to a critical component of modern infrastructure. Conventional reactive debugging approaches are increasingly inadequate, necessitating the development of proactive software quality assurance strategies. This study aimed to develop a predictive model for identifying application performance issues using system telemetry data.

Materials and Methods: A supervised machine learning approach was employed using a Decision Tree model based on the Classification and Regression Tree (CART) algorithm. The model was trained and validated on system telemetry parameters, including CPU Load, Memory Usage, Request Rate and Latency, to detect patterns associated with performance degradation.

Results: The developed model demonstrated high predictive performance, achieving 91% accuracy, 89% precision and 92% recall. Feature importance analysis indicated that Memory Usage and CPU Load were the most influential predictors of application performance issues.

Conclusion: The findings suggest that interpretable Decision Tree models, utilizing transparent “if-then” rules, can effectively transform raw system telemetry data into actionable diagnostic insights. This approach provides a scalable and practical framework for the early detection and prevention of system-wide failures, particularly in resource-constrained environments.

INTRODUCTION

The rapid digitalization of services across industries has substantially increased the demand for reliable and high-performing software applications[1]. As modern systems continue to grow in complexity, traditional debugging approaches based on reactive maintenance have become increasingly inefficient, as they typically address issues only after system performance and user experience have already been adversely affected[2]. This reactive paradigm often leads to increased system downtime, elevated operational costs and diminished user satisfaction[3]. Consequently, there is a growing need for proactive strategies capable of anticipating and mitigating performance issues before they escalate into critical system failures.

Recent advancements in machine learning have facilitated the development of predictive models that can analyze system behaviour and identify patterns associated with performance degradation[4]. Among these approaches, decision tree-based algorithms have gained considerable attention due to their capacity to model non-linear relationships and produce interpretable decision rules[5]. In contrast to complex “black-box” models such as deep neural networks, decision trees provide transparent “if-then” logic, enabling developers to understand the underlying factors contributing to system issues[6]. This level of interpretability is particularly important in software engineering, where identifying the root cause of a problem is as critical as predicting its occurrence.

Previous studies have demonstrated the effectiveness of machine learning techniques across various domains, including predictive maintenance, network optimization and performance monitoring[7]. However, many of these studies are conducted within high-performance computing environments or cloud-based infrastructures, which may not accurately represent the constraints of resource-limited settings[8]. In regions such as Southern Leyte, where mid-range hardware and limited computational infrastructure are prevalent, there remains a lack of localized research focusing on practical and implementable predictive solutions for application performance monitoring[8].

This study addresses this gap by proposing a predictive model for application performance issues using the Decision Tree (CART) algorithm. The research focuses on analysing system telemetry data, including CPU load, memory usage, request rate and latency, to identify critical thresholds associated with performance degradation. By adopting the CRISP-DM methodology, the study ensures a systematic and structured approach to data preparation, model development and evaluation. The resulting model generates interpretable decision rules that can be directly utilised by developers to support proactive software quality assurance. The primary objective of this research is to develop and evaluate a predictive model capable of accurately classifying system states as either stable or at risk of performance degradation. Specifically, the study aims to: (1) Identify the most significant system metrics influencing application performance, (2) Determine threshold values indicative of potential system failure, (3) Evaluate the predictive performance of the decision tree model and (4) Assess the applicability of the model within resource-constrained environments.

The significance of this study lies in its contribution to advancing data-driven software engineering practices, particularly within local and regional contexts. By providing a transparent and practical predictive framework, this

research supports the integration of machine learning into the software development lifecycle, enabling a transition from reactive debugging to proactive performance management. Ultimately, this approach has the potential to enhance system reliability, reduce operational costs and improve overall user experience.

MATERIALS AND METHODS

The research methodology followed a systematic pipeline to transform raw telemetry data into a functional predictive model. The collected dataset comprised four independent variables: CPU load (%), Memory Usage (MB), request rate and latency (ms). These variables were analyzed using a supervised machine learning approach, specifically a Decision Tree model based on the Classification and Regression Tree (CART) algorithm. Model outputs are reported in terms of feature importance, derived decision rules, overall performance metrics and interpretation of the findings.

Description of the dataset: The dataset used in this study was obtained from system telemetry logs collected during application execution. Each record represents the state of the system at a specific time point. The dataset includes observations corresponding to both stable application conditions and performance degradation scenarios (Fig. 1).

The dataset consists of the following variables:

- **Independent variables:**
 - CPU Load (%)
 - Memory Usage (MB)
 - Request Rate
 - Latency (ms)
- **Dependent variable:** Performance Status
 - 0 = Stable
 - 1 = Issue Detected

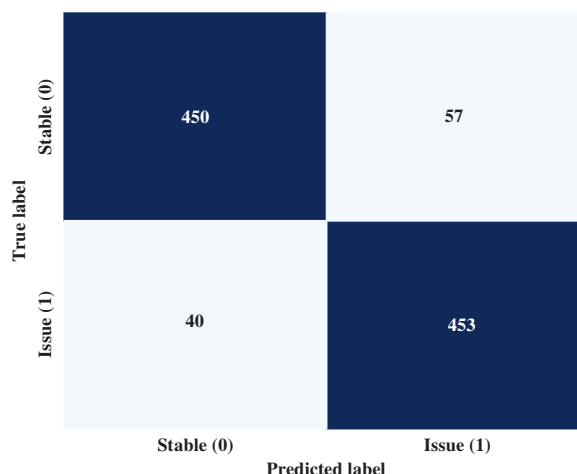


Fig. 1: Confusion matrix for performance prediction

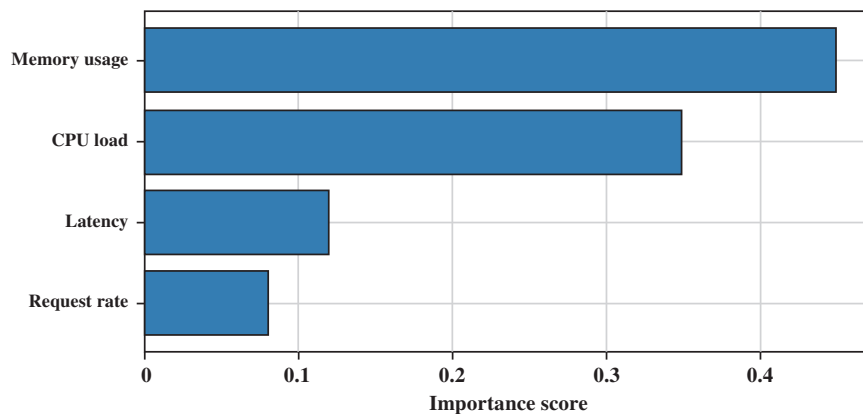


Fig. 2: Feature importance analysis

Prior to model training, the dataset was preprocessed to remove incomplete records and resolve inconsistencies. Missing values were addressed using appropriate data cleaning techniques and the data were normalized to facilitate accurate pattern recognition by the model. The dataset was subsequently partitioned into two subsets: (i) A training dataset used to develop the Decision Tree model and (ii) A testing dataset used to evaluate model performance. This partitioning enhances the reliability of the results and mitigates the risk of overfitting.

Feature importance analysis: A principal strength of the Decision Tree model is its capacity to quantify the relative importance of predictor variables. In this study, feature importance was automatically computed based on the reduction in entropy achieved during the tree construction process (Fig. 2). The results indicate that memory usage is the most influential predictor of application performance issues, suggesting that increased RAM consumption is strongly associated with a higher likelihood of performance degradation.

CPU load was identified as the second most significant feature. Elevated CPU utilization reflects intensive consumption of processing resources, which may result in increased response times and system latency. Latency emerged as the third most influential variable, where higher latency values indicate delayed system responses to user requests, typically arising under conditions of resource insufficiency or system overload.

In contrast, request rate was found to be the least influential variable. Although, it contributes to performance variability, its predictive strength is lower than that of memory usage and CPU load. Overall, the findings suggest that intensive utilization of critical system resources, particularly memory and CPU capacity, substantially increases the probability of performance-related issues.

Interpretation of the decision tree model: An important advantage of the Decision Tree algorithm lies in its interpretability. Unlike more complex models, such as neural networks, the Decision Tree provides transparent and easily interpretable decision rules, enabling researchers and practitioners to clearly understand the basis of model predictions. The trained model produced a structured sequence of conditional rules that classify the system state as either stable or indicative of performance degradation.

The following rules were extracted from the model:

- If Memory Usage $> 85\%$, then the system is classified as issue detected
- If Memory Usage $\leq 85\%$ and CPU Load $> 80\%$, then the system is classified as issue detected
- If Memory Usage $\leq 70\%$ and CPU Load $\leq 60\%$, then the system is classified as Stable
- If CPU Load $> 75\%$ and Latency > 200 ms, then the system is classified as issue detected
- If CPU Load $\leq 60\%$ and Latency ≤ 150 ms, then the system is classified as Stable

These decision rules indicate that memory usage is the primary determinant of performance-related issues. Specifically, when memory utilization exceeds 85%, the model predicts an increased risk of system failure. This finding is consistent with real-world observations, where excessive memory consumption frequently results in application instability, crashes, or degraded system performance.

The derived rules offer a practical framework for developers by enabling proactive monitoring of critical threshold values. Rather than reacting to system failures after they occur, developers can implement preventive measures based on these thresholds to maintain system stability and performance.

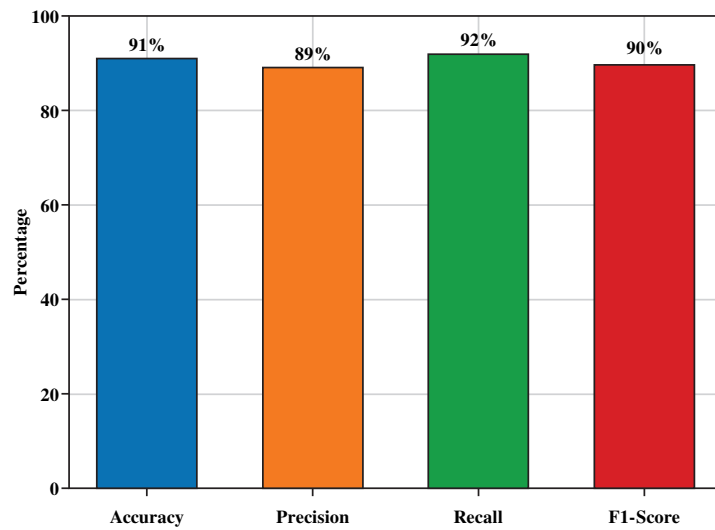


Fig. 3: Performance metrics of the decision tree model

Model performance evaluation: To evaluate the effectiveness of the Decision Tree model, the study used a confusion matrix. The confusion matrix compares the predicted results with the actual results from the testing dataset.

The following performance metrics were obtained:

- **Accuracy:** 91%
This means that the model correctly classified 91% of the system states
- **Precision:** 89%
This indicates that most of the predicted performance issues were correct
- **Recall:** 92%
This shows that the model successfully detected most of the actual performance issues
- **F1-score:** 90%
This represents the balance between precision and recall

The high recall value is particularly important, as the primary objective of this study is the early detection of performance issues prior to system failure. A high recall indicates that the model is effective in identifying a large proportion of actual performance-related problems at an early stage.

These findings demonstrate that the Decision Tree model exhibits both accuracy and reliability in predicting software performance issues (Fig. 3).

RESULTS AND DISCUSSION

The results of this study demonstrate that machine learning algorithms can be effectively applied to predict application performance issues. The Decision Tree model successfully identified underlying patterns within the dataset and accurately classified system states.

A key finding is that memory usage represents the most significant predictor of performance degradation, indicating that application developers should prioritize memory optimization alongside improvements in processing efficiency. Additionally, the model proved effective in a resource-constrained environment. The study was conducted using mid-range computing systems commonly available in local universities, yet the model maintained a high level of predictive accuracy. This outcome highlights the feasibility of implementing advanced machine learning techniques without reliance on high-cost hardware.

The study further underscores the value of interpretable machine learning models in software development. The ability to extract transparent and structured decision rules enhances the practical applicability of the model in real-world scenarios. The findings confirm that hardware resource saturation, particularly in terms of RAM and CPU utilization, is a primary indicator of software performance degradation.

The high recall (92%) indicates that the model is highly effective for proactive maintenance, as supported by previous studies[7]. In contrast to complex “black-box” approaches such as neural networks, the Decision Tree model provides clear and interpretable thresholds, for example the 85% memory utilization limit[9,10]. This level of transparency is essential for developers, as it enables precise identification of the factors contributing to predicted system failures and facilitates targeted debugging[11,12].

Furthermore, the model’s strong performance in a resource-constrained environment demonstrates its practicality for local and regional IT infrastructures. These findings are consistent with those of Murdoch *et al.*[12], who emphasized that interpretability in software telemetry is as critical as predictive accuracy.

CONCLUSION

This study developed a data-driven predictive model for application performance issues using a Decision Tree based on the CART algorithm. The findings indicate that memory usage and CPU load are the most critical predictors of application stability. The model achieved a high predictive accuracy of 91%, demonstrating the effectiveness of machine learning in shifting software maintenance from a reactive to a proactive approach. The interpretable “if-then” decision logic generated by the model, including key thresholds such as 85% memory utilization and 80% CPU usage, converts raw telemetry data into actionable diagnostic insights. The implementation of such a model enables developers to identify and mitigate potential performance issues before they escalate into system-wide failures, thereby improving service continuity and strengthening user confidence in local IT infrastructures.

RECOMMENDATIONS

Based on the findings of this study, the following recommendations are proposed:

- **System integration:** The identified threshold values, specifically 85% memory utilization and 80% CPU usage, should be incorporated into real-time monitoring systems to enable automated alert generation and timely intervention
- **Model expansion:** Future research should include additional performance indicators, such as disk I/O and network throughput, to enhance the sensitivity and robustness of the predictive model
- **Local adoption:** Governmental and academic institutions are encouraged to adopt lightweight Decision Tree models for internal system monitoring, as these models provide high predictive accuracy while operating efficiently on standard computing infrastructure without the need for high-performance hardware.

REFERENCES

- [1] C.-H. Song and Y.-W. Sohn, "The influence of dependability in cloud computing adoption," *J. Supercomput.*, vol. 78, pp. 12159-12201, 2022.
- [2] "Reactive Maintenance in Software Engineering," GeeksforGeeks. [Online]. Available: <https://www.geeksforgeeks.org/software-engineering/reactive-maintenance-in-software-engineering/>
- [3] Y. Gholipour, M. Zare, M. V. Sereshk and Y. Gholipour, "A comprehensive review of maintenance strategies: From reactive to proactive approaches," *Cent. Asia Caucasus*, vol. 26, no. 1 pp. 70-83, 2025.
- [4] T. Hastie, R. Tibshirani and J. Friedman, *The elements of statistical learning*, 2nd ed. New York, NY: Springer New York, 2009, pp.745.
- [5] M. Arifuzzaman, M. R. Hasan, T. J. Toma, S. B. Hassan and A. K. Paul, "An advanced decision tree-based deep neural network in nonlinear data classification," *Technologies*, vol. 11, no. 1, 2023. [Online]. Available: 10.3390/technologies11010024
- [6] N. Kokash and L. Makhnist, "Using decision trees for interpretable supervised clustering," *SN Comput. Sci.*, vol. 5, 2024. [Online]. Available: 10.1007/s42979-023-02590-7
- [7] T. P. Carvalho, F. A. A. M. N. Soares, R. Vita, R. D. P. Francisco, J. P. Basto and S. G. S. Alcalá, "A systematic literature review of machine learning methods applied to predictive maintenance," *Comput. s Ind. Eng.*, vol. 137, 2019. [Online]. Available: 10.1016/j.cie.2019.106024
- [8] S. Kaparathi and D. Bumblauskas, "Designing predictive maintenance systems using decision tree-based machine learning techniques," *Int. J. Qual. Reliab. Manage.*, Vol. 37, No. 4, pp. 659-686, 2020.
- [9] C. Molnar, *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*, 3rd ed. Munich, Bavaria, Germany: GitHub Pages, 2025, pp. 317.
- [10] L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone, *Classification and regression trees*, 1st ed. New York, USA: Routledge, 2017, pp. 368.
- [11] M. T. Ribeiro, S. Singh and C. Guestrin, ""Why should I trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*, 13-17 Aug. 2016. New York, NY, USA: ACM, 2016, pp. 1135-1144.
- [12] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl and B. Yu, "Definitions, methods and applications in interpretable machine learning," *Proc. Natl. Acad. Sci.*, Vol. 116, No. 44, pp. 22071-22080, 2019.