**Bulletin of Electrical Engineering and Informatics**
Vol. 15, No. 3, June 2026, pp. 2505~2514
ISSN: 2302-9285, DOI: 10.11591/eei.v15i3.11905


# Performance analysis of classification models to determine the health status of edge computing devices

**Ricardo Yauri[1,2], Nora Bertha La Serna Palomino[2]**
[1]Facultad de Ingeniería, Universidad Tecnológica del Perú, Lima, Perú
[2]Facultad de Ingeniería de Sistemas e informática, Universidad Nacional Mayor de San Marcos, Lima, Perú

**Article Info**

***Article history:***

Received Nov 19, 2025
Revised Mar 20, 2026
Accepted Apr 19, 2026

***Keywords:***

Deep neural network
Edge computing
Embedded algorithms
Hardware constraints
Model evaluation
TinyML

**ABSTRACT**

Artificial intelligence (AI) has contributed to the development of autonomous systems in the healthcare field by integrating machine learning models, whose evaluation on resource-limited hardware devices is important to ensure their efficiency. This research evaluates the performance of classification models in edge computing (EC) systems, considering metrics such as accuracy, latency, memory consumption, and energy efficiency on low-power microcontrollers using TinyML techniques. The processes involved include the development, implementation, and testing of algorithms on embedded hardware using differentiated preprocessing techniques and the validation of hypotheses through statistical analysis. The results show that the decision tree (DT) model is more efficient in terms of prediction time and energy consumption, while random forests (RFs) stand out for their greater accuracy. Furthermore, memory analysis reveals that models based on fully connected neural networks are more efficient in terms of RAM usage. This provides guidelines for selecting algorithms in resource-constrained environments.



***Corresponding Author:***

Ricardo Yauri
Facultad de Ingeniería, Universidad Tecnológica del Perú
Cal. Natalio Sanchez Nro. 125, Lima, Perú
Email: c24068@utp.edu.pe

## 1. INTRODUCTION

The growth of technologies related to edge computing (EC) and artificial intelligence (AI) has allowed the development of intelligent systems for applications in the health area, contributing to the prediction of diagnosis where the evaluation of the behavior of prediction models [1], [2] in the devices is critical [3]. In this context, EC emerges as an alternative for the integration of AI algorithms in hardware devices with limited resources, considering metrics such as energy consumption [4]. For example, during the COVID-19 pandemic, many patients with high-risk signs [5], [6] were isolated without receiving continuous health monitoring [7], [8].

There are problems related to continuous health monitoring, related to constant readings and medical intervention, affecting the quality of patient treatment [3], [9]. In this context, data transfer in internet of things (IoT) solutions generates high latency times and energy consumption [2], [10], which is critical in applications such as the detection of arrhythmias in patients [7], [11], making it necessary to improve the scalability of hardware devices with integrated algorithms [4]. In relation to the technological problem, there is a need to integrate sensors and neural network models into mobile devices [9], [12], making it necessary to evaluate the behavior of the models [8], [13]. Furthermore, the integration of AI requires the evaluation of accuracy, resource consumption and response time [6], [14].

The literature review describes research that seeks to improve the continuous measurement of physiological variables [9], [15] where some works propose cloud, fog and EC architectures to optimize decision making [5], [8] related to the prediction of heart rate and arrhythmias [11], [13]. In addition, other applications use integrated devices such as the ESP32 with AI [16], [17], where prediction models are used [7], [12]. The methodology applied in these studies includes the evaluation of prediction models such as long short-term memory (LSTM) and random forest (RF) [9], [13], use of large data sets, for their implementation in microcontrollers [4], [11], without considering hardware limitations and identification of trends in health systems with machine learning [3], [12], and the development of an adaptive system that optimizes the configuration of hardware and software [4], [12]. Furthermore, some studies using ESP32-based platforms focus on accuracy but do not describe a comparison between different models. Previous research trends have shown federated learning approaches and adaptive IoT systems for health monitoring that improve data privacy.

The objective is to evaluate the performance of classification models in an embedded EC system for vital signals. The use of TinyML techniques for IoT in healthcare has highlighted the importance of lightweight models for implementation in embedded systems, often prioritizing hardware resource efficiency. The results are important in healthcare scenarios such as monitoring patients with IoT medical devices in resource-constrained environments where latency and power consumption are relevant. The main contributions of this research are:

- Comparative evaluation of machine learning models support vector machines (SVM), RF, and decision trees (DT), multilayer perceptron (MLP), and deep neural networks (DNN), on hardware.
- Evaluation of performance metrics, including accuracy, latency, and memory consumption.
- Statistical validation of model performance differences.
- Implementation guidelines for model selection in IoT applications.

## 2. PROPOSED SYSTEM

This section describes the development, implementation, and testing of AI algorithms on a hardware device using EC techniques. In this context, we address the adaptation of a dataset through differentiated preprocessing techniques, where sensor data are preprocessed to obtain the most relevant features (Figure 1). Hypothesis validation is then performed using statistical processes to verify the efficiency of the models, RAM and FLASH memory, prediction time, and power consumption (Figure 2).

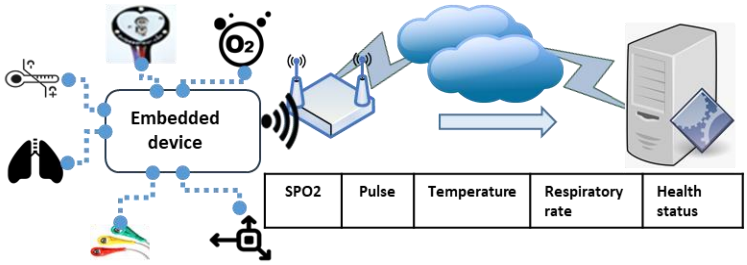


Figure 1. Classification workflow

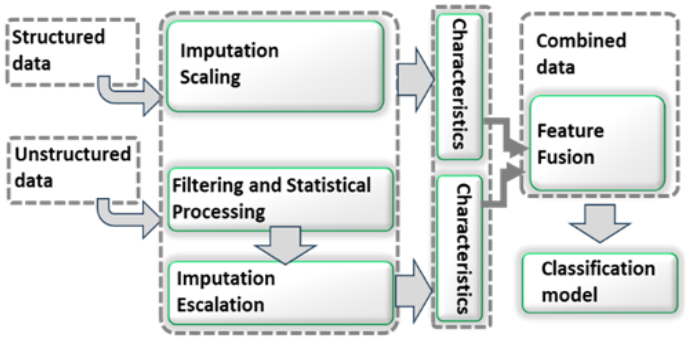


Figure 2. Classification process

### 2.1. Data sources and health-status definition

Considering the literature review, the CVD-Vital-Signs database [18], [19] and the database generated from the NEWS2 health status categorization [20], [21] were selected (NEWS2 score is widely used in clinical settings to assess patient deterioration based on vital signs). Both datasets include key physiological variables such as oxygen saturation ($SpO_2$), body temperature, respiratory rate, and heart rate. In the case of the NEWS2 scale, this was created to create a representative dataset of vital signs with three severity output categories (0=normal or stable, 1=mild, and 2=critical). After applying the balancing strategy, an adjusted number of samples per class was obtained to reduce bias. The integration of the datasets was achieved by selecting common physiological variables such as heart rate, respiratory rate, oxygen saturation, and temperature. Normalization was then applied to standardize the scale of these variables.

### 2.2. Model families and hyperparameter optimization

A process model is proposed for the evaluation of embedded AI models on an edge computing device (ECD), where several preprocessing stages are integrated with the aim of adapting both structured and unstructured data for training and testing. This approach has been developed from previous research in the field of embedded systems for EC [18], [19] and TinyML applications [20], [21], making use of recognized methodologies such as KDD and CRISP-DM [22], [23] (Figure 3). The training and testing stages consider data preprocessing, data division, training, validation, classification, and testing.

### 2.3. Process model for embedded artificial intelligence evaluation

Feature extraction is necessary using the differentiated preprocessing technique (in this approach, different preprocessing methods are applied to the features to improve model performance) due to its low to medium complexity and simplicity (it requires less computational power and is easy to interpret).

To implement the signal classification processes, programming scripts were developed in Python and C++ for the hardware device. Libraries such as TensorFlow Lite, TensorFlow [24], [25], and Micromlgen were used for deployment on the hardware device. The mechanisms and techniques used in each stage are described:

- Loading and exploration of the dataset: an exploratory data analysis was performed, and the dataset has 16,637 records.
- Data preprocessing: the data are adapted for use in the training and validation stages of machine learning models by performing data cleaning. For unbalanced classes, the least frequent subsampling technique is used as illustrated in Figure 4. Data preprocessing included data cleaning, removal of missing values, normalization with min-max scaling, and feature selection.
- Data splitting and algorithm selection: in this stage, the data is split into training, validation, and test sets, with a proportion of 64%, 16%, and 20%, respectively, using the Scikit-learn library.
- Training: model training is performed by adjusting the parameters of each algorithm, optimizing hyperparameters using techniques such as GridSearchCV and RandomizedSearchCV (Figure 5).

### 2.4. Implementation of the model for deployment in ESP32

During the validation stage, different hyperparameter settings are used for each model with tabular data. Among the machine learning-based models, the RF-based model performed best, with an accuracy of 0.9482 and an F1-score of 0.948, followed by the DT-based model (0.938). On the other hand, the SVM model showed the lowest performance, with an accuracy of 0.9166. Among the NN-based models, the MLP model ranked third in accuracy, with 0.9245 (Table 1).

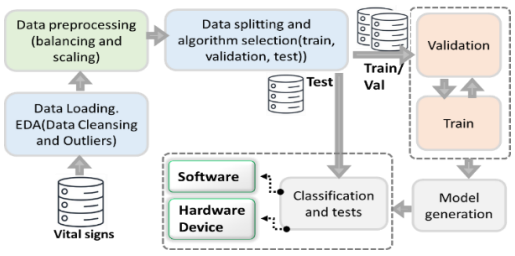


Figure 3. Process model for embedded AI evaluation

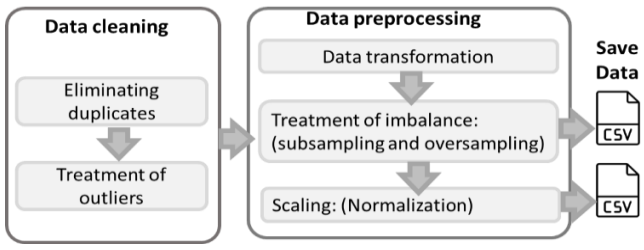


Figure 4. Data preprocessing

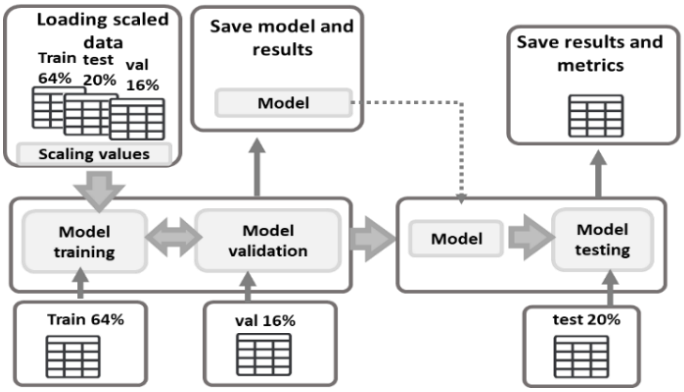


Figure 5. Training process

Table 1. Metrics summary with validation data in the development environment

| Model | Dataset | Configuration | Accuracy | F1-score |
|---|---|---|---|---|
| SVM | DD4 | 'C': 100, 'gamma': 5.0, 'kernel': 'rbf' | 0.9166 | 0.9166 |
| DT | DD7 | 'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2 | 0.938 | 0.9381 |
| RF | DD5 | 'max_depth': 15, 'min_samples_leaf': 2, 'min_samples_split': 3, 'n_estimators': 30 | 0.9482 | 0.9482 |
| MLP | DD3 | 'layers': [32, 16], 'activation': 'relu', 'learning_rate': 0.01, 'batch_size': 16, 'epochs': 20 | 0.9245 | 0.9243 |
| DNN | DD4 | 'layers': [128, 64, 32, 16, 8], 'activation': 'relu', 'learning_rate': 0.001, 'batch_size': 16, 'epochs': 20 | 0.92 | 0.9198 |

## 3. RESULTS

Each of the models generated in the previous stage are evaluated using test data in the development environment and on the hardware device that implements the hardware device using EC techniques.

### 3.1. Models in the development environment

After training and validation, the models' performance is evaluated using the test dataset, under similar deployment conditions as on the hardware device. According to the metrics summary, Table 2 shows the best models for each algorithm type, where the RF model shows the best overall accuracy (0.9504), followed by the DT model (0.9450) and the MLP model (0.9303).

Table 2. Metrics summary with test data in the development environment

| Model | Dataset | Configuration | Accuracy | F1-score |
|---|---|---|---|---|
| SVM | DD3 | 'C': 100, 'gamma': 5.0, 'kernel': 'rbf' | 0.9213 | 0.9211 |
| DT | DD8 | 'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2 | 0.945 | 0.945 |
| RF | DD3 | 'max_depth': 15, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 30 | 0.9504 | 0.9504 |
| MLP | DD3 | 'layers': [32, 16], 'activation': 'relu', 'learning_rate': 0.01, 'batch_size': 16, 'epochs': 20 | 0.9303 | 0.9301 |
| DNN | DD8 | 'layers': [128, 64, 32, 16, 8], 'activation': 'relu', 'learning_rate': 0.001, 'batch_size': 16, 'epochs': 20 | 0.9231 | 0.9231 |

The behavior evolution graph shows that the most accurate models are used, showing that both training and validation accuracy increased steadily, indicating convergence. However, as more training data is used, the validation accuracy approaches the training accuracy, indicating that overfitting does not occur (Figures 6 and 7). The results explicitly show that the RF algorithm achieves the highest predictive performance, while DT is more efficient in relation to computational cost.

### 3.2. Evaluation of models on the hardware device

The evaluation focuses on verifying whether the model maintains efficient performance considering hardware constraints using a section of the test data. To do this, the models and scaled data (along with the scaling values used) are imported and then converted to hardware-compatible formats (PHW1). The hardware metrics were obtained using a multimeter (in the case of electrical current), the "micros" program function (for latency measurement), and compiler options (to obtain memory usage).

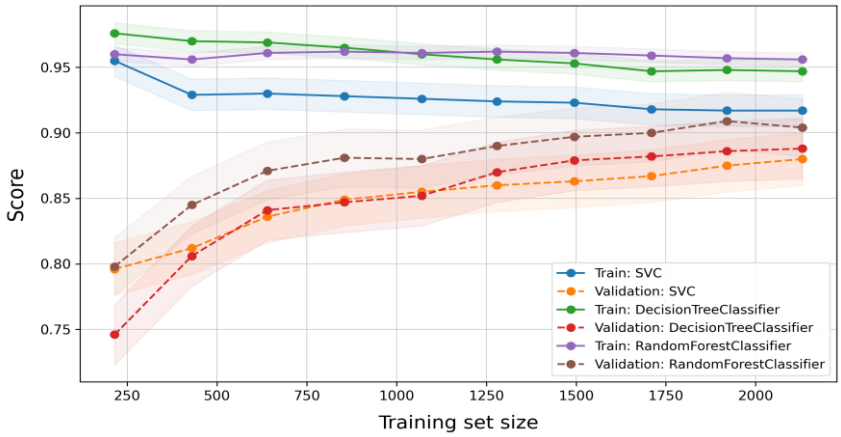


Figure 6. Training and validation curves with convergence of machine learning models

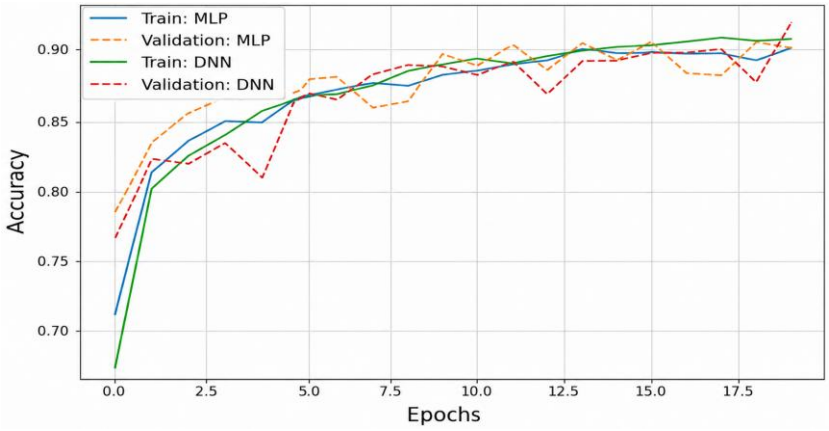


Figure 7. Training curves and validation of neural network-based models with inferior performance compared to DT-based models

The models are exported using the micromlgen 1.1.28 version and TensorFlow lite 2.17 version software libraries that translate the models from Python to C++ and are imported from the ECD to obtain performance metrics (PHW2). A conversion stage for unscaled data (NE/ES Process) is then considered, which is then input into the hardware. To facilitate reproducibility, a workflow is shown that details the loading of models, obtaining metrics, data preparation, conversion, and hardware inference (Figure 8).

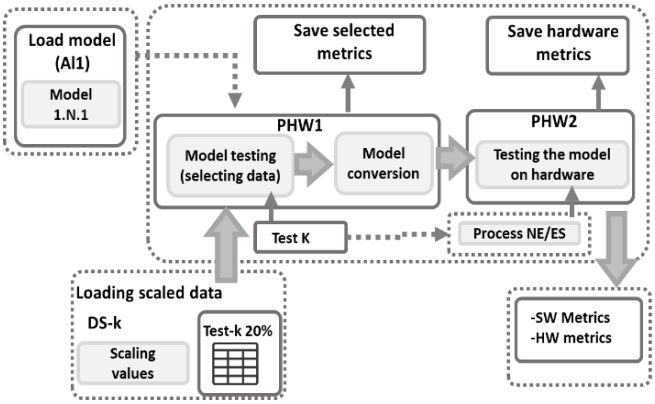


Figure 8. Processes to determine the behavior of models on the hardware device

Comparing the evaluated models (Table 3), it is observed that, in terms of accuracy, the RF and DT models stood out with 96.6% and 95.6% respectively, while the F1-score of the RF models was the highest, reaching 97%, indicating a balance between accuracy and sensitivity. On the other hand, the neural network-based models achieved good results with accuracies of 94.6% and 94.3%, but their F1-score was slightly lower.

Table 4 shows differences in the models' behavior when running on hardware. RF again gives the best performance, with a high accuracy of 96.6%, followed by the MLP-based model. On the other hand, lighter models, such as DT, have slightly lower accuracy (95.6%). SVM, on the other hand, showed the lowest impact on hardware, with an accuracy of 91%. During the evaluations, each test on the hardware device was repeated 10 times, and the metrics obtained correspond to the average values. In addition, 95% confidence intervals were calculated for inference latency, power consumption, and memory usage to quantify data variability and perform statistical tests.

Table 3. Average metrics with selected test data in the software

| Model | Configuration | Accuracy | F1-score |
|---|---|---|---|
| SVM | C': 100, 'gamma': 5.0, 'kernel': 'rbf' | 0.9466 | 0.9532 |
| DT | max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2 | 0.9566 | 0.9598 |
| RF | max_depth': 15, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 30 | 0.9666 | 0.9707 |
| MLP | layers': [64, 32], 'activation': 'relu', 'learning_rate': 0.01, 'batch_size': 16, 'epochs': 20 | 0.94666 | 0.9524 |
| DNN | layers': [64, 32, 16, 8], 'activation': 'relu', 'learning_rate': 0.01, 'batch_size': 32, 'epochs': 20 | 0.94333 | 0.95 |

Table 4. Average of the metrics on the hardware device with the test data

| Model | Configuration | Accuracy |
|---|---|---|
| SVM | C': 100, 'gamma': 5.0, 'kernel': 'rbf' | 0.91 |
| DT | max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2 | 0.956 |
| RF | max_depth': 15, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 30 | 0.96601 |
| MLP | layers': [64, 32], 'activation': 'relu', 'learning_rate': 0.01, 'batch_size': 16, 'epochs': 20 | 0.957 |
| DNN | layers': [64, 32, 16, 8], 'activation': 'relu', 'learning_rate': 0.01, 'batch_size': 16, 'epochs': 20 | 0.942 |

## 4. DISCUSSIONS

The results show trade-offs between model performance and resource consumption. While RF models offer greater accuracy, this requires more memory. DT models, on the other hand, are less accurate but more efficient. Neural network models (MLP and DNN) are more adaptable to patterns but require more processing power. Furthermore, energy efficiency, memory usage, and real-time processing are critical factors for monitoring in embedded systems.

### 4.1. Comparison of software and hardware environment

In the development environment, the RF model (96.6%) maintains a value like that obtained in the hardware device. Furthermore, the DT models also maintain their value of 95.66% in both the hardware and development environments, indicating greater adaptability to device limitations and showing the least change between the two environments.

Overall, the results indicate a trade-off between accuracy and computational efficiency. Although the RF model has the highest accuracy, the DT model offers the best balance between performance and hardware resource consumption, making it more suitable for applications on hardware devices. This suggests the importance of selecting models not only based on predictive performance but also considering hardware limitations (Figure 9).

### 4.2. Operation on the hardware device

This section shows the behavior of the hardware device related to power consumption, latency, time used to make predictions, the size of the models in memory, FLASH and RAM. While it is noted above that the models maintained acceptable accuracy (around 90%) on the hardware device, there is a notable difference in the case of inference time and power consumption. The DT-based model offers a reasonable balance between accuracy and computational efficiency by consuming the least amount of electrical current (39.3 mA) and inference time (8.665 us), as shown in Table 5, considering TN as true negative, TP as true positive and us as microseconds.

On the other hand, in the case of RF, despite their adequate performance in the software environment in terms of accuracy, they are observed to consume more memory resources and inference time than all the models. The DT achieves the best energy efficiency, while more complex models, such as RF and DNN, require significantly more resources. Finally, in terms of power consumption, neural network-based models are the most energy intensive, as shown in Figure 10.

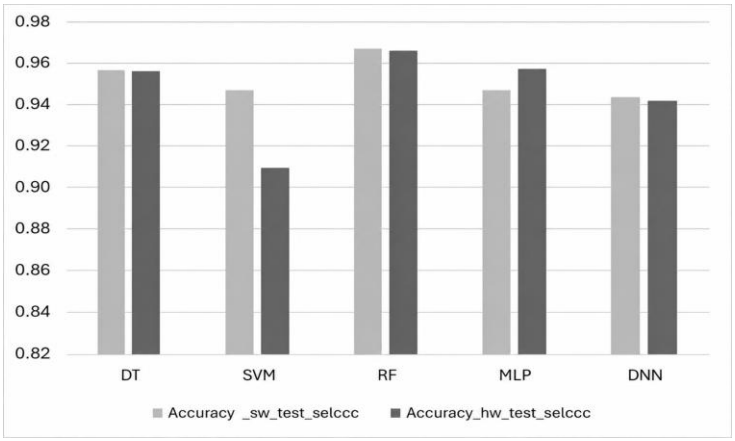


Figure 9. Comparison of model performance in development and hardware environments

Table 5. Average operating features on the hardware device

| Model | Current (mA) | TN (us) | TP (us) | RAM (%) | FLASH (%) |
|---|---|---|---|---|---|
| SVM | 40.3 | 4.8 | 73.02 | 82.22 | 46.1523 |
| DT | 39.3 | 4.818 | 8.665 | 6.9 | 6.771 |
| RF | 41.46 | 4.821 | 720.2 | 100 | 100 |
| MLP | 59.6 | 4.677 | 123.2358 | 6.838 | 9.942 |
| DNN | 57.8 | 4.692 | 409.502 | 7.305 | 10.495 |

From an application perspective, the results suggest that models like DTs are suitable for continuous monitoring of portable devices, where energy consumption and inference time are critical. On the other hand, models like RF are more suitable when greater accuracy is needed on devices with fewer hardware resources. The energy efficiency of DTs stems from their simple structure and low computational complexity. Unlike neural network methods, DTs require fewer arithmetic operations during inference.

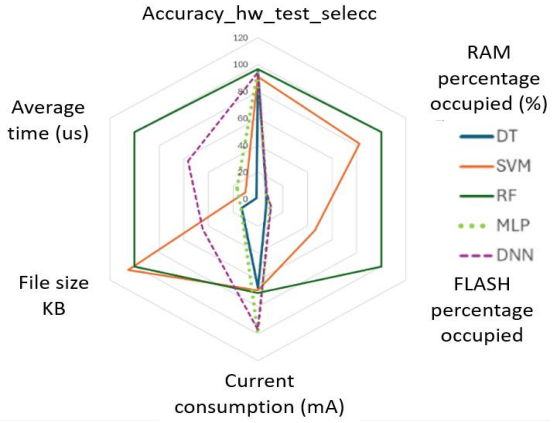


Figure 10. Comparative graph of data obtained during the evaluation on the hardware

### 4.3. Hypothesis testing

The statistical procedure was structured considering the assumptions of normality and homogeneity of variances using the Shapiro-Wilk and Levene tests. Based on these results, parametric or non-parametric statistical tests were selected, and if significant differences were detected, post-hoc multiple comparison tests. To test the hypothesis, the following steps were taken for each characteristic to be evaluated regarding the behavior of the classification models in the ECD:

- Selection of learning algorithms.
- Calculation of the metrics to be evaluated: for each model, multiple sample groups are considered (10 samples corresponding to model variations for each of the 5 algorithms).

- Definition of hypotheses: a null hypothesis (H0) and an alternative hypothesis (H1) for each of the characteristics: H0: there is no difference between the evaluated features of the different models and H1: there is a significant difference between the evaluated features of the different learning models.
- Verification of assumptions. Two assumptions are verified: data normality and homogeneity of variances. To do this, Shapiro-Wilk and Levene's statistical tests are performed to verify homogeneity of variances.
- Determination of differences between models. Based on the previous results, the type of parametric (e.g., ANOVA) or nonparametric (Kruskal-Wallis) test is selected.
- Multiple comparison tests: if the previous result indicates significant differences, multiple comparison tests such as the Tukey test or the Bonferroni test are performed. As an example of the hypothesis evaluation procedure, the results obtained for evaluating electricity consumption: Shapiro-Wilk test: the electricity consumption data follows a normal distribution; Levene's test: the variances of the electricity consumption data are equal across groups.

The evaluated metrics are shown in Table 6, which considers computational efficiency, a critical aspect in hardware with limited resources. The table presents the statistical results obtained with non-parametric tests (Kruskal-Wallis). In particular, the DT model stands out for its lower prediction time, power consumption, and FLASH memory usage. The MLP model has lower RAM consumption, while models such as RF and SVM show higher resource requirements. In the case of RAM and FLASH, their null hypothesis was rejected, while it was not for accuracy.

Table 6. Statistical summary of performance metrics for evaluated models

| Metric | Test used | p-value | Significant differences | Best model | Key findings |
|---|---|---|---|---|---|
| Prediction time | Kruskal-Wallis | 8.37E-09 | Yes | DT | DT is the fastest (~8.67 µs), followed by SVM and MLP. DNN and RF are the slowest. |
| Accuracy | Kruskal-Wallis | 0.1358 | No | RF | No statistical differences; all models show similar performance. |
| RAM memory | Kruskal-Wallis | 7.76E-10 | Yes | MLP, followed by DT | MLP uses less RAM; RF and SVM consume more. |
| FLASH memory | Kruskal-Wallis | 1.08E-09 | Yes | DT, followed by MLP | DT is the most storage-efficient; RF and SVM use more memory. |
| Current consumption | Kruskal-Wallis | 1.27E-08 | Yes | DT, followed by SVM and RF | DT is the most efficient (~39.3 mA); DNN and MLP consume more energy. |
| Model size | Kruskal-Wallis | 7.43E-09 | Yes | DT, followed by MLP | DT has the smallest size (~59.7 KB); RF and SVM are the largest. |

### 4.4. Deployment guidelines

Based on the tests performed, recommendations are established for model selection in healthcare settings. If high accuracy is required with low-power hardware, RF models can be used. Regarding energy efficiency, latency, and memory, DT models offer a better balance between performance and computational cost. Neural network-based models (MLP and DNN) are suitable for applications requiring lower accuracy, but their energy consumption should be evaluated. SVMs are less adaptable to hardware limitations.

The results can be related to monitoring scenarios, where access to real-time information is critical. In this case, DT are suitable. However, in hospital settings with limited computing resources, complex models can be used for high accuracy. Furthermore, more sensitive models are preferred when classification results are critical.

## 5. CONCLUSION

This paper makes a contribution by integrating comparative processes, hardware-considered analysis, and statistical validation of machine learning models within a TinyML-based computing environment. The results can serve as a basis for the design of vital signs monitoring systems in remote patient monitoring applications or IoT-based wearable devices. The prediction time of the models in hardware was evaluated, finding that the DT model offers the shortest prediction time and has significant differences in performance compared to models such as the DNN and RF. Regarding the accuracy assessment when evaluating the models in hardware, it is recommended to consider other relevant aspects for decision-making, such as prediction time or memory consumption. Although the statistical tests did not identify significant differences between the models, the visual comparison highlights the RF model, thanks to its highest average accuracy (0.96601), followed by the MLP models.

These results confirm that memory-related metrics (RAM and FLASH) present statistically significant differences, unlike accuracy, where no significant differences were found. In this case, MLP stands out as the most efficient model in terms of RAM usage. Regarding FLASH memory usage, the alternative hypothesis is confirmed, with the DT and MLP models emerging as the best options for limited hardware devices. In the case of the datasets used in this study (CVD-Vital-Signs and NEWS2), these were integrated and preprocessed for research purposes, and the new version generated is available from the authors upon request of readers. While the RF model achieved the highest accuracy, the DT model is best suited for implementation due to its lower power consumption, shorter inference time, and reduced computational complexity.

A limitation is that the evaluation was only performed on one ESP32, which prevents the results from being generalized. Furthermore, the dataset is based on existing data and does not account for the variability of real-world data. Future research could focus on integrating classification models for monitoring health-related parameters, including wearable devices, and validating their performance in real-world scenarios. Additionally, cloud architectures could be integrated for large-scale applications.

## FUNDING INFORMATION

Authors state no funding involved.

## AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

| Name of Author | C | M | So | Va | Fo | I | R | D | O | E | Vi | Su | P | Fu |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Ricardo Yauri | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Nora Bertha La Serna Palomino | | ✓ | | | ✓ | ✓ | | | ✓ | ✓ | ✓ | ✓ | ✓ | |

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

## CONFLICT OF INTEREST STATEMENT

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author, Ricardo Yauri, upon reasonable request.

## REFERENCES

[1] K. V. K. Stephen *et al.*, "IOT-Based Generic Health Monitoring with Cardiac Classification Using Edge Computing," *Journal of Internet Services and Information Security*, vol. 13, no. 2, pp. 128–145, May 2023, doi: 10.58346/JISIS.2023.I2.008.

[2] P. Gupta, A. V. Chouhan, M. A. Wajeed, S. Tiwari, A. S. Bist, and S. C. Puri, "Prediction of health monitoring with deep learning using edge computing," *Measurement: Sensors*, vol. 25, p. 100604, Feb. 2023, doi: 10.1016/j.measen.2022.100604.

[3] O. Oyebode, J. Fowles, D. Steeves, and R. Orji, "Machine Learning Techniques in Adaptive and Personalized Systems for Health and Wellness," *International Journal of Human-Computer Interaction*, vol. 39, no. 9, pp. 1938–1962, May 2023, doi: 10.1080/10447318.2022.2089085.

[4] M. A. Scrugli, D. Loi, L. Raffo, and P. Meloni, "An Adaptive Cognitive Sensor Node for ECG Monitoring in the Internet of Medical Things," *IEEE Access*, vol. 10, pp. 1688–1705, 2022, doi: 10.1109/ACCESS.2021.3136793.

[5] S. Ghosh and A. Mukherjee, "STROVE: spatial data infrastructure enabled cloud–fog–edge computing framework for combating COVID-19 pandemic," *Innovations in Systems and Software Engineering*, vol. 20, no. 4, pp. 727–743, Dec. 2024, doi: 10.1007/s11334-022-00458-2.

[6] F. Quiñonez-Cuenca *et al.*, "Evaluation of AIoT performance in Cloud and Edge computational models for mask detection," *Ingenius*, vol. 2022, no. 27, pp. 32–47, Dec. 2022, doi: 10.17163/ings.n27.2022.04.

[7] J. P. Tincopa, P. Vela-Anton, C. U. Quispe-Juli, and A. Arostegui, "Development of an IoT Device for Measurement of Respiratory Rate in COVID-19 Patients," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 4,

pp. 77–82, 2022, doi: 10.14569/IJACSA.2022.0130409.
[8] Y. Abadade, N. Benamar, M. Bagaa, and H. Chaoui, "Empowering Healthcare: TinyML for Precise Lung Disease Classification," *Future Internet*, vol. 16, no. 11, p. 391, Oct. 2024, doi: 10.3390/fi16110391.
[9] Y. Xiang, S. Li, and P. Zhang, "An exploration in remote blood pressure management: Application of daily routine pattern based on mobile data in health management," *Fundamental Research*, vol. 2, no. 1, pp. 154–165, Jan. 2022, doi: 10.1016/j.fmre.2021.11.006.
[10] V. K. Quy, N. V. Hau, D. V. Anh, and L. A. Ngoc, "Smart healthcare IoT applications based on fog computing: architecture, applications and challenges," *Complex and Intelligent Systems*, vol. 8, no. 5, pp. 3805–3815, Oct. 2022, doi: 10.1007/s40747-021-00582-9.
[11] Z. Jia *et al.*, "TinyML Design Contest for Life-Threatening Ventricular Arrhythmia Detection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 1, pp. 127–140, Jan. 2024, doi: 10.1109/TCAD.2023.3309744.
[12] S. Hassantabar, J. Zhang, H. Yin, and N. K. Jha, "MHDeep: Mental Health Disorder Detection System Based on Wearable Sensors and Artificial Neural Networks," *ACM Transactions on Embedded Computing Systems*, vol. 21, no. 6, pp. 1–22, Nov. 2022, doi: 10.1145/3527170.
[13] K. Owaki, Y. Kanda, and H. Kimura, "Heart Rate Control System for Walking with Real-Time Heart Rate Prediction," *IEICE Transactions on Electronics*, vol. E107-C, no. 11, pp. 501–505, Nov. 2024, doi: 10.1587/transele.2023ESS0002.
[14] J. Hyysalo *et al.*, "Smart mask – Wearable IoT solution for improved protection and personal health," *Internet of Things (Netherlands)*, vol. 18, p. 100511, May 2022, doi: 10.1016/j.iot.2022.100511.
[15] M. Humayun, A. Alsirhani, F. Alserhani, M. Shaheen, and G. Alwakid, "Transformative synergy: SSEHCET—bridging mobile edge computing and AI for enhanced eHealth security and efficiency," *Journal of Cloud Computing*, vol. 13, no. 37, pp. 1–21, Feb. 2024, doi: 10.1186/s13677-024-00602-2.
[16] O. Lawal, S. A. V. Shajihan, K. Mechitov, and B. F. Spencer, "Edge Integration of Artificial Intelligence into Wireless Smart Sensor Platforms for Railroad Bridge Impact Detection," *Sensors*, vol. 24, no. 17, Aug. 2024, doi: 10.3390/s24175633.
[17] R. Yauri, R. Espino, and A. Castro, "Evaluation of an Indoor Location System Using Edge Computing and Machine Learning Algorithms," *International Journal of Online and Biomedical Engineering*, vol. 20, no. 4, pp. 4–17, Mar. 2024, doi: 10.3991/ijoe.v20i04.46771.
[18] D. Situnayake, "Machine Learning on Mobile and Edge Devices with TensorFlow Lite," Infoq, 2020. https://www.infoq.com/presentations/tensorflow-lite. (Accessed Nov. 09, 2021).
[19] P. Lea and an O. M. C. Safari, *IoT and Edge Computing for Architects - Second Edition*, Packt Publishing, 2020.
[20] H. Zhou, X. Zhang, Y. Feng, T. Zhang, and L. Xiong, "Efficient human activity recognition on edge devices using DeepConv LSTM architectures," *Scientific Reports*, vol. 15, no. 13830, pp. 1–18, Apr. 2025, doi: 10.1038/s41598-025-98571-2.
[21] A. Khatoon, W. Wang, M. Wang, L. Li, and A. Ullah, "TinyML-enabled fuzzy logic for enhanced road anomaly detection in remote sensing," *Scientific Reports*, vol. 15, no. 20659, pp. 1–20, Jul. 2025, doi: 10.1038/s41598-025-01981-5.
[22] G. M. Iodice, *TinyML Cookbook*, O'Reilly Media, Inc., 2022.
[23] A. Purbasari, F. R. Rinawan, A. Zulianto, A. I. Susanti, and H. Komara, "CRISP-DM for Data Quality Improvement to Support Machine Learning of Stunting Prediction in Infants and Toddlers," in *Proceedings - 2021 8th International Conference on Advanced Informatics: Concepts, Theory, and Application, ICAICTA 2021*, Sep. 2021, pp. 1–6, doi: 10.1109/ICAICTA53211.2021.9640294.
[24] R. D. Marco, F. Di Nardo, A. Rongoni, L. Screpanti, and D. Scaradozzi, "Real-Time Dolphin Whistle Detection on Raspberry Pi Zero 2 W with a TFLite Convolutional Neural Network," *Robotics*, vol. 14, no. 5, May 2025, doi: 10.3390/robotics14050067.
[25] Z. Lv *et al.*, "Efficient Deployment of Peanut Leaf Disease Detection Models on Edge AI Devices," *Agriculture (Switzerland)*, vol. 15, no. 3, p. 332, Feb. 2025, doi: 10.3390/agriculture15030332.

## BIOGRAPHIES OF AUTHORS

**Ricardo Yauri** is a Master of Science in Electronic Engineering with a mention in Biomedical. He is an associate professor at Universidad Nacional Mayor de San Marcos (UNMSM) and a Ph.D. student in Systems Engineering. He is also a professor at Universidad Tecnológica del Perú and Universidad Peruana del Norte. He has participated as a teacher in courses oriented to the Internet of Things and applications in home automation and the Cisco Academy for IoT. He was a researcher at INICTEL-UNI in the Embedded Systems and Internet of Things research group. He developed research projects on the implementation of low-consumption IoT devices that involve inference techniques, machine learning algorithms. He can be contacted at email: C24068@utp.edu.pe and boncer99@gmail.com.

**Nora Bertha La Serna Palomino** is a Ph.D. in Computer Science from the Universidad del País Vasco, Spain, obtained in March 1998. She received her Master of Science in Computer Science from the University of Cantabria, Spain, in July 1990, and her Bachelor of Science in Computer Science from the National University of San Marcos in Lima, Peru, in 1992. She is a Full Professor at the National University of San Marcos (UNMSM) at both the undergraduate and graduate levels. Her research areas include software engineering and computer science. She is the Principal Investigator of funded research projects and the author of articles published in specialized and indexed journals. She also serves as an advisor for undergraduate, Master's, and doctoral theses. She can be contacted at email: nlasernap@unmsm.edu.pe.