Online-Journals.org

iJOE International Journal of Online and Biomedical Engineering

iJOE | eISSN: 2626-8493 | Vol. 20 No. 4 (2024) | OPEN ACCESS

https://doi.org/10.3991/ijoe.v20i04.46771

PAPER

Evaluation of an Indoor Location System Using Edge Computing and Machine Learning Algorithms

Ricardo Yauri(✉), Rafael Espino, Antero Castro

Universidad Tecnológica del Perú, Lima, Perú

c24068@utp.edu.pe

ABSTRACT

The paper aims to evaluate precise location techniques with indoor devices using edge computing technologies, which are important for services such as smart homes and health. Despite their growing importance, indoor locations lack precise and standard methods, especially in complex environments. Solving this is being attempted through technologies such as reconfigurable surfaces and deep learning models, with attention to overcoming the challenges of indoor placement. The main objective of the study is to design a low-cost indoor location system using the ESP32 module and RSSI signals, integrated with embedded machine learning algorithms. The system to be developed will allow determining the location of objects or people with a location device through SSID signals from access points. The main objective is to evaluate the performance of three machine learning algorithms—random forest (RF), decision tree (DT) and support vector machine (SVM)—in the detection of four different locations (bathroom, kitchen, bedroom, and living room), involving the definition of system characteristics, data acquisition, the development of classifiers, and their integration in the ESP32 module to transmit location data wirelessly through the MQTT protocol. As a result of the evaluation, the DT model stands out for its efficiency under limited resource conditions during real-time implementation, but it may face challenges related to overfitting and resources at the implementation stage.

KEYWORDS

localization, machine learning, edge computing, Wi-Fi

1 INTRODUCTION

Currently, there are efforts focused on the precise location of nodes based on Internet of Things (IoT) technologies enabled for different communication networks,

Yauri, R., Espino, R., Castro, A. (2024). Evaluation of an Indoor Location System Using Edge Computing and Machine Learning Algorithms. *International Journal of Online and Biomedical Engineering (iJOE)*, 20(4), pp. 4–17. https://doi.org/10.3991/ijoe.v20i04.46771

Article submitted 2023-11-15. Revision uploaded 2024-01-12. Final acceptance 2024-01-13.

which is essential for a variety of location-based services, such as smart homes, smart healthcare, monitoring environments, personal navigation, and intelligent transportation. In addition, there is a lack of standardized and accurate positioning methods for indoor environments where mobile devices and IoT are currently being developed [1] [2].

Unlike outdoor positioning methods, where highly accurate techniques exist, equally reliable approaches are not available indoors. Indoor positioning, often based on WiFi fingerprints, although useful, still faces notable limitations that require more advanced and accurate solutions due to the lack of precision and generalization in indoor positioning systems that make use of WiFi fingerprints [3]. When localization is needed indoors and in multi-building and multi-story environments, building and story classification (BFC) functionality can be added along with latitude and longitude regression (LLR) in a wide three-dimensional space [4].

There are challenges to the use of received signal strength (RSSI), highlighting issues such as multipath, lack of line of sight, and variability in RSSI measurements [5]. This is related to the vulnerability of location-based services with IoT solutions due to attacks that exploit the RSSI and raises the need to detect and mitigate these attacks [6]. In some cases, the problem is tried to be solved with unconventional indoor solutions such as GPS [7] integrated into embedded devices such as edge computing solutions and tinyML, which are not effective indoors, while this could change with 5G technology [7].

During the literature review, articles have been found that focus on addressing the problem of precise location in indoor environments in the context of the Internet of Things (IoT). Some papers [1] [5] propose the use of reconfigurable smart surfaces (RSSI) to improve indoor location accuracy, highlighting the importance of this technology in applications such as smart homes and smart healthcare. On the other hand, in [2], they focus on improving interpretability in indoor positioning, using a deep learning model with an attention module to identify the most relevant access points. Both articles seek to overcome the limitations of traditional solutions in indoor environments and improve location accuracy.

Other studies are looking at how to make Wi-Fi-based indoor positioning systems more accurate and reliable. They suggest using a method called model stacking, which combines different models in a smart way. They use a technique called Bayesian optimization to figure out the best way to combine these models, aiming to improve accuracy and reliability. [3]. When there is a need for localization in indoor environments with multiple buildings and floors, federated learning (FL) can be used, addressing the challenges of a three-dimensional space [4]. On the other hand, other papers address challenges related to the accuracy and reliability of location in different technological contexts. In [6], he focuses on the detection of attacks based on the received signal strength (RSS) in IoT networks, proposing detection methods and evaluating their performance in real environments.

These location processes can be integrated with TinyML and neural networks, designing devices applied to communication and determining the location of objects [8], where the challenge lies in the integration of intelligence techniques in devices with hardware restrictions [9] [10]. In other cases, 5G technology is integrated to contribute to navigation in these environments through 5G reference signals

and machine learning [7] [11], while other solutions use RFID technologies [12]. Embedded technologies contribute to vehicle tracking solutions in smart cities [13] and drones [14].

For this reason, the following question arises: How is it possible to conduct the evaluation of an indoor location system using edge computing and machine learning algorithms? To answer this question, the general objective is defined as: Design a location system with a low-cost ESP32 module for indoor location based on RSSI signals and integration with embedded machine learning algorithms.

2 EDGE COMPUTING

It is a paradigm of data processing and application execution at the edge of a communication network, which means close to where the data is generated instead of sending it to a centralized data center or the cloud for processing [15] [16]. Instead of relying exclusively on remote servers for processing tasks, edge computing uses local computing resources, such as embedded devices, local servers, or network nodes, to perform faster and more efficient processing of data at the point of use [17].

2.1 ESP32 hardware module

It is a highly versatile and widely used microcontroller that belongs to the ESP8266/ESP32 family of products developed by the company Espressif Systems. The ESP32 has gained great popularity in the development community due to its Wi-Fi and Bluetooth connectivity capabilities, low cost, and extensive feature set [18] [19]. It offers a wide range of GPIO pins, sufficient processing power and memory, as well as integrated peripherals such as SPI, I2C, and UART. Furthermore, it is energy efficient and can be easily programmed through the Arduino environment [20] [21].

3 SYSTEM DEVELOPMENT

The system to be developed will allow determining the location of any object or person that has the location device and obtaining information from the SSID signals of the access point devices (see Figure 1). The objective of the system is to evaluate the behavior of three machine learning algorithms—RF, DT, and SVM—detecting four distinct locations: bathroom, kitchen, bedroom, and living room.

To ensure the success of the project, an incremental and agile development methodology was adopted. This choice arose from his adaptability to the dynamic nature of the project and his ability to provide early, incremental results. The agile approach allowed for a step-by-step evaluation of objectives, such as the integration of location systems. Rigorous tests and adjustments were conducted to validate the objective achievements and optimize of resources. In addition, this methodology was also designed to cover the definition of the signal, the architecture of hardware and software, as well as the wireless communication network.

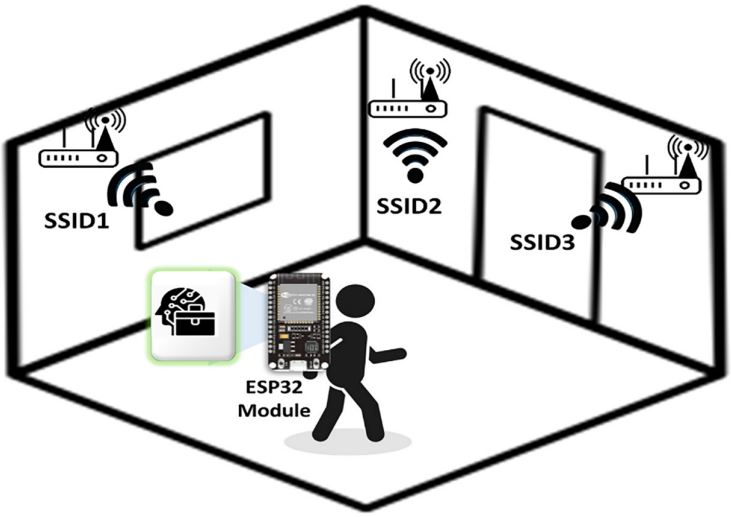

Fig. 1. Indoor location scheme

Initially, the scheme of specific characteristics of the positioning system is defined. Subsequently, data acquisition is performed using hardware, resulting in a data set and a C++ conversion algorithm. Next, three types of machine learning classifiers—SVM, RF, and DT—are developed and evaluated based on the acquired data, resulting in three models in the C++ language. The final phase involves embedding the machine learning algorithm into the ESP32 module, enabling wireless transmission of location data to an Internet panel via the MQTT protocol. This approach streamlines the development process, addresses the complexities of indoor positioning technology, and ensures iterative delivery of functional components (see Figure 2).

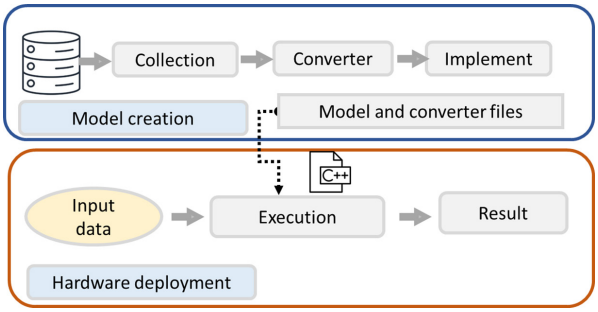

Fig. 2. Stages for system development

3.1 Definition of characteristics

The definition of characteristics is established based on the use of the RSSI of known Wi-Fi networks to characterize distinct locations. First, the RSSI of nearby Wi-Fi access points is collected, and if a network is out of range, its RSSI will be zero. Each location registers a number of networks with their respective RSSIs, creating a unique footprint identification for each place. Since not all networks are always visible, the collected data conforms to a sparse matrix, where most of the elements are zero, representing the absence of certain characteristics. Only the relevant elements, which are different from zero, indicate the RSSI values of the networks at each location, thus forming a distinctive representation.

The sparse nature of the data reflects that not all networks are present everywhere, and only non-null values are meaningful to characterize each location.

Below is an example of a matrix that could be generated based on the features of four distinct locations of a home (refer to Table 1). Each position is distinguished by a particular combination of networks and remains constantly detectable.

Table 1. SSID signal relationship matrix

Localization	AP 1	AP 2	AP 3	AP4	AP 5
Bathroom	0	–88	0	–88	–61
Kitchen	0	–72	–54	–80	0
Bedroom	–90	0	–64	0	–74
Hall	–53	0	0	0	0

3.2 Data collection

With the desired structure in our data, the next step involves obtaining it. To start, the ESP32 board is used for collection, performing periodic scans for identifiable networks, and recording their RSSI values in an encrypted JSON format. In this project, 20 registers are collected for each of the four locations for a few seconds, and then the information resulting from the serial output is saved in a text file. The code generated for the ESP32 module aims to acquire data from Wi-Fi networks (SSID) to later use them in training a location algorithm, as shown in Figure 3. Its operation is detailed below.

- Initial setup:
  - Serial communication is established, and the Wi-Fi module is configured in station mode.
  - The module disconnects from previous Wi-Fi networks.
- Main loop:
  - If there is a defined location, nearby Wi-Fi networks are scanned.
  - The location and data of each network (SSID and RSSI) are printed in JSON format.
  - A delay is introduced before repeating the process.
- Location settings:
  - If there is no location defined, it waits for the user to enter the location through the serial port.
  - If the command “scan {location}” is entered, that location is assigned to the corresponding variable.

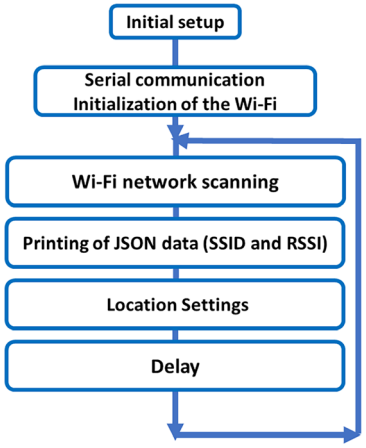

Fig. 3. Localization algorithm flowchart

After repeating the process for each location to be classified, something such as the result observed in Table 2 is obtained:

Table 2. Data acquired in Json format

Line	Data in Json
1	{"__location": "{cocina}", "MOVISTAR_E150": -72, "MAYRA": -79}
2	{"__location": "{cocina}", "MOVISTAR_E150": -64, "MAYRA": -82}
3	{"__location":"{cocina}", "MOVISTAR_E150":-67, "FA ALEJO": -89}
4	{"__location": "{cocina}", "MOVISTAR_E150": -70, "TP-Link": -91}

3.3 Feature converter

From the information acquired, a code in the C++ language is created capable of transforming the product of a Wi-Fi network search into a set of attributes that will be used as inputs for the categorization model. To conduct this process, a code generator tool is used that is based on the translation of machine learning algorithms. This tool, named "microm-legn," can be installed using the command "pip install-upgrade micro-mlgen."

The import of the "port_wifi_indoor_positioning" module is conducted, which has the purpose of providing functions and utilities for the generation of optimized code destined to implement indoor positioning models adapted for devices with limited resources, such as microcontrollers and embedded systems. The provided code (Table 3) allows converting the sample data (samples) into a suitable format, resulting in the following parameters:

- X: Represents the characteristics extracted from the samples, which will be used as inputs for the classification model. These characteristics are derived from data from scanned Wi-Fi networks.
- y: Corresponds to the labels or classes associated with the samples. These labels will indicate the location or position indoors.
- Classmap: It is a mapping that relates the labels with their corresponding numerical or categorical values.
- Converter_code: It is the code generated by the "micromlgen" library that will transform the samples into characteristics and labels suitable for the indoor positioning model. This code can include transformations, preprocessing, and model setup.

Finally, the print(converter_code) code will print to the console the code generated by the port_wifi_indoor_positioning function, which can later be used to implement the positioning model in the ESP32.

Table 3. Conversion instructions to generate features

Line	Conversion Python Instructions
1	pip install --upgrade micromlgen.
2	if __name__ == '__main__':
3	samples = ""
4	{"__location": "{cocina}", "MOVISTAR_E150": -72, "MAYRA": -79}
5	{"__location": "{cocina}", "MOVISTAR_E150": -64, "MAYRA": -82}
6	{"__location":"{cocina}", "MOVISTAR_E150":-67, "FA ALEJO": -89}
7	X, y, classmap, converter_code = port_wifi_indoor_positioning(samples)
8	print(converter_code)

The resulting transformation algorithm is saved in a file named "Converter.h," which contains the result of the object named "conver-ter_code." In this algorithm, a function called getFeatures () is incorporated, which is responsible for carrying out an analysis of Wi-Fi networks through a scanning process. For each detected network, the corresponding value is assigned to the feature vector, in case the network is recognized. At the end of this process, a vector of characteristics is obtained that could present a structure such as [2, 80, 0, 0, 0, 20, 0], where each component reflects the intensity of the RSSI signal of a network.

The following Figure 4 shows the tasks performed by the generator algorithm with a class called "WifiIndoorPositioning" that is found within the namespace (namespace) "Eloquent::Projects." The purpose of this class is to generate a feature vector from the data collected from nearby Wi-Fi networks. The class also includes a protected method called ssidToFeatureIdx(String ssid) that converts the SSID (network name) into a feature index to associate each network with its respective signal strength in the feature vector.

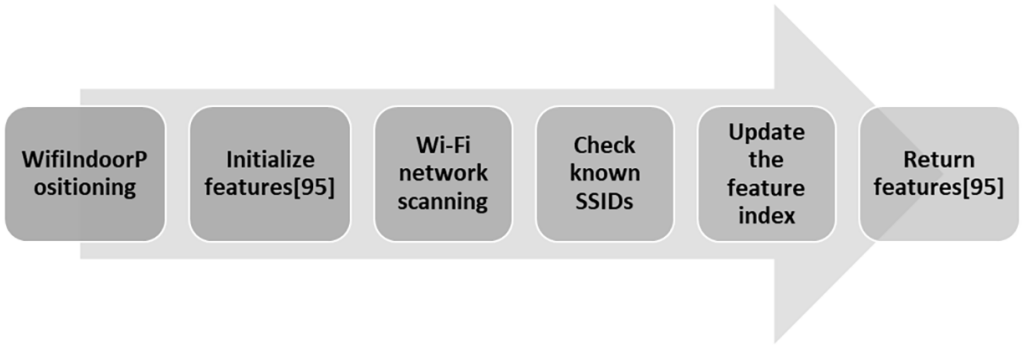

Fig. 4. Feature conversion algorithm diagram

3.4 Implementation of classification models

In this section, three classifier models are integrated with the feature vector at each of the registered locations. In the Python script, instead of printing the converter code, the classifier code is printed. The Python code performs a machine learning process using, for example, the decision tree algorithm with the scikit-learn library. First, the DecisionTreeClassifier class is imported, and an instance of this classifier is created. The model is then trained with input data X and class labels y. Finally, the port_wifi_indoor_positioning function is used to generate C++ code optimized for the microcontroller. Table 4 shows an example of the use of the instructions using the DT classifier.

Table 4. Instructions for implementing the classifier

Line	Instructions
1	pip install scikit-learn.
2	from sklearn.tree import DecisionTreeClassifier
3	from micromlgen import port_wifi_indoor_positioning, port
4	clf = DecisionTreeClassifier()
5	clf.fit(X, y)
6	print(port(clf, classmap = classmap))

The C++ language code implements three classifiers called "Classi-fierRF.h," "ClassifierRT.h" and "ClassifierSVM.h" to be imported from the embedded system. Classifiers contain two main functions: predict and predictLabel. The predict function takes a feature vector x and performs a series of comparisons on the feature values to determine the class to which it belongs. The predictLabel function converts the numeric class predicted by predict into a readable and understandable label, assigning names to the classes, such as "{bathroom}," "{kitchen}," "{domip}," and "{living room}."

3.5 Deployment of the embedded model

Both the file and the file are added to a programming project for the ESP32 module (see Figure 5). The new classification algorithm runs the scan and tells you what location you are in. The location information is sent to a remote server using the MQTT protocol for data display on a dashboard (see Figure 6).

Fig. 5. Deployment of the models in the ESP32 module

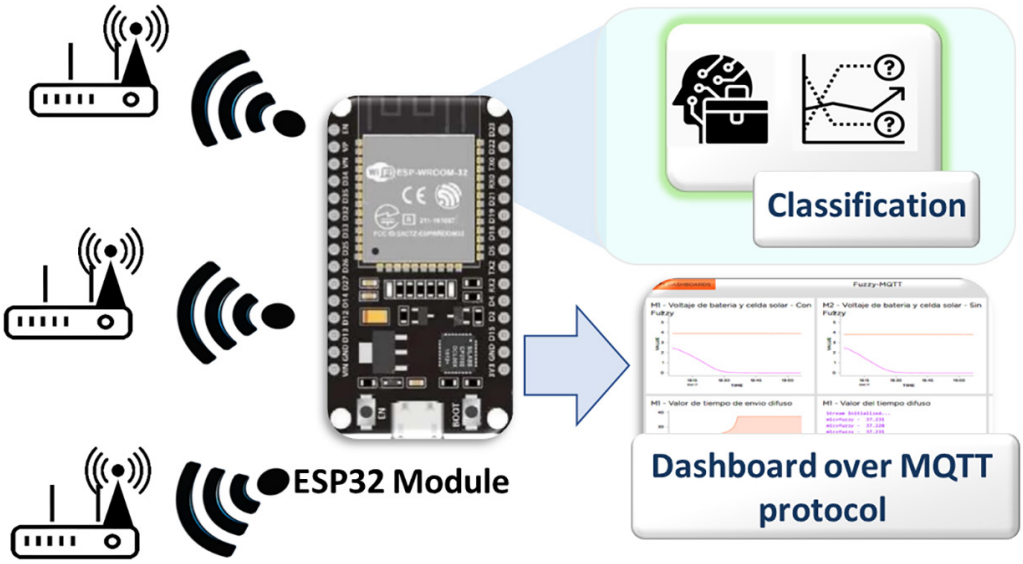

Fig. 6. Diagram of operation of the classification process for localization

4 RESULTS AND DISCUSSION

4.1 Training models in software

In this section, the evaluation of the three learning models is conducted during their training and testing stages with data science software. The choice of these three models allows for the comparison of different approaches to solving the interior location problem. SVM is based on finding the best separation hyperplane; RF takes advantage of the robustness of multiple trees; and boosted RF combines the power of both approaches. To determine which model is most suitable, it is important to perform a thorough evaluation of its performance in terms of metrics such as accuracy and confusion matrix on validation and test sets.

To observe the behavior of the classification model, we can generate an image as shown in Figure 7. Decision regions for a DT classifier are observed using decomposition analysis and dimensionality reduction by principal component analysis (PCA). This two-dimensional projection of the classifier. The resulting graph shows the decision regions generated by the classifier for the different classes in the two-dimensional space. Each class is labeled with a name, and the appropriate symbols and colors are shown to distinguish the classes.

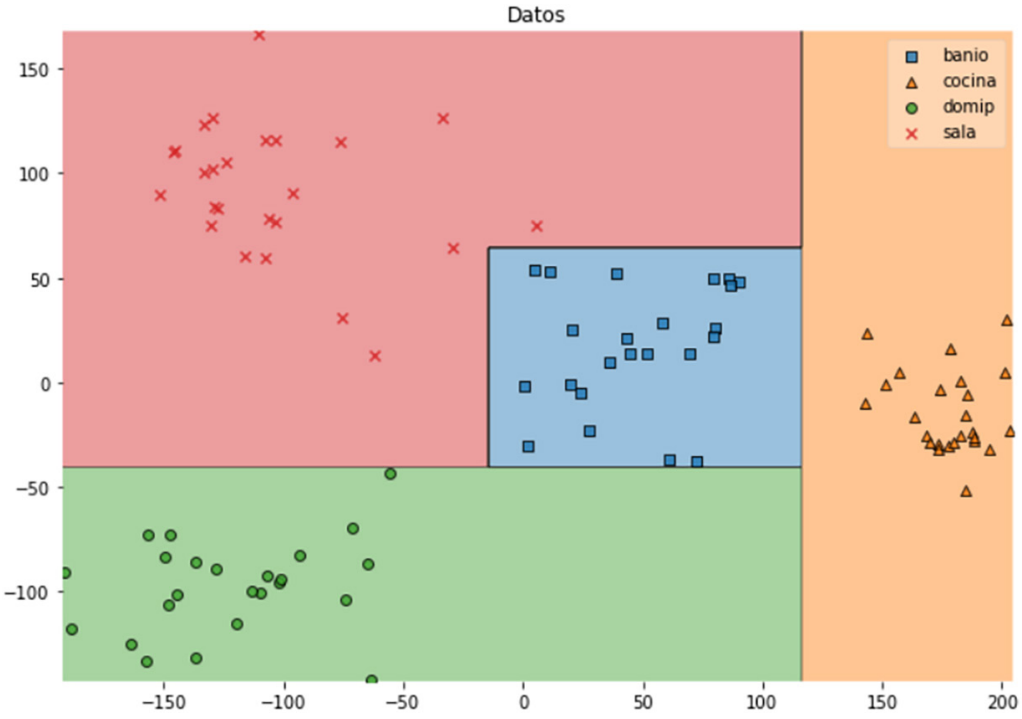

Fig. 7. Visualization of location classification

A function is used to evaluate the performance of a machine learning model trained using a test data set. In our case, being a classification model, we use the cls.score command to return the accuracy of the model in the test set, that is, the proportion of correct predictions (if it were a regression model, the R squared as a proportion of the variance in the target variable explained by the model). The cls.score(X_test, y_test) function compares the model predictions (X_test) with the actual labels (y_test) and calculates the proportion of correct predictions. This

score provides us with an idea of how the model generalizes to unknown data, obtaining the accuracy (as the true positives plus true negatives over the total examples) (refer to Table 5).

Table 5. Accuracy assessment of the models with test data

Algorithm	Accuracy
Decision Trees (DT)	0.714
Random Forests (RF)	0.812
Support Vector Machine (SVM)	0.691

Tables 6, 7, and 8, show the confusion matrices of the three learning models obtained during the testing stage in the model creation software.

Table 6. Decision tree model confusion matrix

	Bathroom	Kitchen	Bedroom	Hall
Bathroom	16	2	1	1
Kitchen	3	14	2	1
Bedroom	1	3	15	1
Hall	2	1	1	16

Table 7. Random forest model confusion matrix

	Bathroom	Kitchen	Bedroom	Hall
Bathroom	17	1	1	1
Kitchen	2	16	1	1
Bedroom	1	3	17	1
Hall	1	1	1	17

Table 8. Support vector machine model confusion matrix

	Bathroom	Kitchen	Bedroom	Hall
Bathroom	16	1	2	1
Kitchen	2	15	1	2
Bedroom	1	1	17	1
Hall	1	1	1	17

Based on the data from the confusion matrices, the following parameters can be calculated:

- Train Score: The accuracy of the training data.
- Test Score: The precision in the test data, that is, the accuracy of the model in new unseen data.
- Precision: The proportion of correct positive predictions in relation to all positive predictions.

- Sensitivity: Also known as the true positive rate (TPR), it is the proportion of true positives in relation to all true positive examples.
- Specificity: Also known as the true negative rate (TNR), it is the proportion of true.

The results in Table 9 show that the RF model performs better in this case due to its nature as an ensemble of DTs. The RF combines multiple DTs to form a more robust and accurate model. Each tree in the ensemble is trained on a random subset of the data and produces a prediction, and then the predictions from all the trees are combined to arrive at a final prediction. In this data set, the RF can take advantage of the strengths of multiple trees and generate accuracy and generalizability compared to a single DT or a linear model (as in the case of SVM). It is also less prone to overfitting due to its training process and the variability introduced by random data sampling. This can result in a better ability to handle unseen test data and generalize to new samples, which is reflected in its higher precision and performance compared to other models.

Table 9. Model evaluation metrics during training and software testing

Alg	Train Score	Test Score (Accuracy)	Precision	Sensitivity	Specificity	F1-Score
RF	0.8375	0.8625	0.8095	0.85	0.967	0.8291
DT	0.83	0.825	0.7894	0.825	0.933	0.8061
SVM	0.89	0.8625	0.8095	0.85	0.967	0.8291

Furthermore, in the case of the size of the models, once the input data conversion file and the classification model file have been obtained, the lightest is the DT, while the largest is 71 KB for the archive with the classification program (Table 10).

Table 10. File size generated with classification models

Alg	Conversion File Size (Kbytes)	File Size Classification (Kbytes)
RF	18	43
DT	18	18
SVM	18	71

4.2 Evaluation of hardware models

This section shows the disparity in the performance of the learning models between the software and its deployment in the ESP3 module, which can be attributed to several reasons. First, embedded systems such as the ESP32 module have limited resources in terms of processing power, memory, and power, which can make more complex learning models inefficient or even incompatible with these devices.

During the deployment of the algorithms in hardware, it was found that the DT have better behavior, which may be since they are simple and of low complexity, even though during the training stage, these can be adjusted very much, well to the training data. This results in high accuracy; however, data in the real world tends to vary and can contain noise (Table 11).

Table 11. Model score (accuracy)

Alg	Train	Test	s. test ESP32
RF	0.90	0.8625	0.78
DT	0.90	0.825	0.85
SVM	0.93	0.8625	0.83

The results in Table 12 show that models can vary in size when deployed on embedded hardware due to the need to accommodate resource constraints, optimize efficiency, and meet specific device requirements. In this case, the C++ algorithm was optimized for optimal performance in these resource-constrained environments. The difference in memory consumption between machine learning models such as RF, DT, and SVM is due to factors such as model complexity. It is inferred that the most complex model is SVM because it requires more memory than DT. Analyzing these factors shown in Table 12 helps to understand and control the memory usage of each model.

Table 12. Program and dynamic memory usage on the ESP32

Alg	% Program Memory	% Dynamic Memory
RF	32	12
DT	31	12
SVM	33	14

5 CONCLUSIONS

The DT model has better behavior in conditions of limited resources during its deployment. Therefore, performing this evaluation helps ensure that learning models work effectively and efficiently in practical real-time applications, such as internal localization in embedded systems or microcontrollers.

Decision trees, especially when they are very deep, can require a large number of computational resources during the training stage. Compared to the training stage, the hardware implementation was subject to more severe resource constraints, which could limit the ability to use large and complex models.

In summary, although DT models can perform well in the training stage, it is important to note that their performance in the testing or hardware implementation stage can be affected by overfitting problems, data variability, and resource requirements. Therefore, it is essential to rigorously evaluate and tune these models to ensure robust performance in real-world settings.

In future research, energy efficiency can be evaluated, as well as in the improvement of precision and adaptation to changing environments. Security and privacy are essential, as are validation in real-world scenarios and integration with specific applications. It is also necessary to evaluate costs and explore interoperability standards, and a continuous evaluation of custom algorithms must be carried out. These efforts will help advance location detection in embedded hardware, enabling a wide range of applications in the fields of edge computing and the Internet of Things.

6 REFERENCES

[1] A. Taneja, S. Rani, J. Breñosa, A. Tolba, and S. Kadry, "An improved WiFi sensing based indoor navigation with reconfigurable intelligent surfaces for 6G enabled IoT network and AI explainable use case," *Future Generation Computer Systems*, vol. 149, pp. 294–303, 2023. https://doi.org/10.1016/j.future.2023.07.016

[2] A. Brunello, A. Montanari, and N. Saccomanno, "Towards interpretability in fingerprint based indoor positioning: May attention be with us," *Expert Syst. Appl.*, vol. 231, 2023. https://doi.org/10.1016/j.eswa.2023.120679

[3] J. Q. Wang, Y. J. Wang, and G. W. Liu, "A model stacking algorithm for indoor positioning system using WiFi fingerprinting," *KSII Transactions on Internet and Information Systems*, vol. 17, no. 4, pp. 1200–1215, 2023. https://doi.org/10.3837/tiis.2023.04.009

[4] B. Gao, F. Yang, N. Cui, K. Xiong, Y. Lu, and Y. Wang, "A federated learning framework for fingerprinting-based indoor localization in multibuilding and multifloor environments," *IEEE Internet Things J*, vol. 10, no. 3, pp. 2615–2629, 2023. https://doi.org/10.1109/JIOT.2022.3214211

[5] S. M. Asaad and H. S. Maghdid, "Novel integrated matching algorithm using a deep learning algorithm for Wi-Fi fingerprint-positioning technique in the indoors-IoT era," *PeerJ. Comput. Sci.*, vol. 9, 2023. https://doi.org/10.7717/peerj-cs.1406

[6] S. Saxena, A. Pandey, and S. Kumar, "RSS based multistage statistical method for attack detection and localization in IoT networks," *Pervasive Mob. Comput.*, vol. 85, 2022. https://doi.org/10.1016/j.pmcj.2022.101648

[7] Z. Liu, L. Chen, X. Zhou, Z. Jiao, G. Guo, and R. Chen, "Machine learning for Time-of-Arrival estimation with 5G signals in indoor positioning," *IEEE Internet Things J.*, vol. 10, no. 11, pp. 9782–9795, 2023. https://doi.org/10.1109/JIOT.2023.3234123

[8] D. Avellaneda, D. Mendez, and G. Fortino, "A tinyML deep learning approach for indoor tracking of assets," *Sensors*, vol. 23, no. 3, 2023. https://doi.org/10.3390/s23031542

[9] R. Yauri and R. Espino, "Edge device for movement pattern classification using neural network algorithms," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 30, no. 1, pp. 229–236, 2023. https://doi.org/10.11591/ijeecs.v30.i1.pp229-236

[10] R. Yauri, R. Acosta, M. Jurado, and M. Rios, "Evaluation of principal component analysis algorithm for Locomotion activities detection in a tiny machine learning device," in *Proceedings of the 2021 IEEE Engineering International Research Conference, EIRCON 2021*, Institute of Electrical and Electronics Engineers Inc., 2021. https://doi.org/10.1109/EIRCON52903.2021.9613450

[11] P. Ramakrishna and P. Rajarajeswari, "Evolutionary optimization algorithm for classification of microarray datasets with Mayfly and Whale survival," *International Journal of Online and Biomedical Engineering*, vol. 19, no. 13, pp. 17–37, 2023. https://doi.org/10.3991/ijoe.v19i13.40145

[12] M. Merenda, L. Catarinucci, R. Colella, D. Iero, F. G. Della Corte, and R. Carotenuto, "RFID-based indoor positioning using edge machine learning," *IEEE Journal of Radio Frequency Identification*, vol. 6, pp. 573–582, 2022. https://doi.org/10.1109/JRFID.2022.3182819

[13] L. Wan, M. Zhang, L. Sun, and X. Wang, "Machine learning empowered IoT for intelligent vehicle location in smart cities," *ACM Trans Internet Technol.*, vol. 21, no. 3, 2021. https://doi.org/10.1145/3448612

[14] P. K. Rai *et al.*, "Localization and activity classification of unmanned aerial vehicle using mmWave FMCW radars," *IEEE Sens. J.*, vol. 21, no. 14, pp. 16043–16053, 2021. https://doi.org/10.1109/JSEN.2021.3075909

[15] C. Li *et al.*, "Blockchain enabled task offloading based on edge cooperation in the digital twin vehicular edge network," *Journal of Cloud Computing*, vol. 12, no. 1, 2023. https://doi.org/10.1186/s13677-023-00496-6

[16] Z. Dai, Q. Zhang, L. Zhao, X. Zhu, and D. Zhou, "Cloud-edge computing technology-based Internet of Things system for smart classroom environment," *International Journal of Emerging Technologies in Learning*, vol. 18, no. 8, pp. 79–96, 2023. https://doi.org/10.3991/ijet.v18i08.28299

[17] J. Chen, Y. Leng, and J. Huang, "An intelligent approach of task offloading for dependent services in mobile edge computing," *Journal of Cloud Computing*, vol. 12, no. 1, 2023. https://doi.org/10.1186/s13677-023-00477-9

[18] S. S. Yadav, R. Agarwal, K. Bharath, S. Rao, and C. S. Thakur, "TinyRadar for fitness: A contactless framework for edge computing," *IEEE Trans Biomed Circuits Syst.*, vol. 17, no. 2, pp. 192–201, 2023. https://doi.org/10.1109/TBCAS.2023.3244240

[19] T. B. Nguyen and T. C. Nguyen, "Design and fabrication of an IoT-based smart electrical meter for residential energy management," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 30, no. 3, pp. 1259–1268, 2023. https://doi.org/10.11591/ijeecs.v30.i3.pp1259-1268

[20] D. Hercog, T. Lerher, M. Truntič, and O. Težak, "Design and implementation of ESP32-based IoT devices," *Sensors*, vol. 23, no. 15, 2023. https://doi.org/10.3390/s23156739

[21] N. A. Jasim and H. T. S. ALRikabi, "Design and implementation of smart city applications based on the Internet of Things," *International Journal of Interactive Mobile Technologies*, vol. 15, no. 13, pp. 4–15, 2021. https://doi.org/10.3991/ijim.v15i13.22331

7 AUTHORS

Ricardo Yauri holds a Master's degree in electronic engineering with a specialization in Biomedical studies. He serves as an Associate Professor at the Universidad Nacional Mayor de San Marcos (UNMSM) and is currently pursuing a Ph.D. in Systems Engineering. He is also a faculty member at Universidad Tecnológica del Perú and Universidad Peruana del Norte. He has actively contributed as an instructor for courses focused on the Internet of Things (IoT) and its applications in home automation (E-mails: C24068@utp.edu.pe and ryaurir@unmsm.edu.pe).

Rafael Espino is a Specialist in the development of electronic systems with solid knowledge in photovoltaic systems and extensive experience in implementing embedded firmware in electronic systems. He is a graduate of the Faculty of Electronics of the National University of Engineering He is a Professor at the University of Engineering and Technology of Peru. With aptitudes for teaching, research and project management (E-mail: rafalloelo@gmail.com).

Antero Castro is a graduate of the National University of Engineering of Peru (UNI) and master's in information and communication Engineering from the Beijing University of Aeronautics and Astronautics, China. With more than 10 years of professional experience and he has collaborated in the CHASQUI-I and Radioskaf-2 satellite projects. He currently works as a research professional at the National Institute of Training and Research in Telecommunications of the National University of Engineering (INICTEL-UNI) (E-mail: c19828@utp.edu.pe).