



UNIVERSIDADE FEDERAL DE ALAGOAS
INSTITUTO DE COMPUTAÇÃO
COORDENAÇÃO DE PÓS-GRADUAÇÃO EM INFORMÁTICA

EMANUEL ADLER MEDEIROS PEREIRA

**WATER POTABILITY CLASSIFICATION: AN APPROACH
USING MACHINE LEARNING IN AN EMBEDDED SYSTEM**

Maceió

2024

EMANUEL ADLER MEDEIROS PEREIRA

**WATER POTABILITY CLASSIFICATION: AN APPROACH
USING MACHINE LEARNING IN AN EMBEDDED SYSTEM**

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal de Alagoas como requisito parcial para obtenção do título de Mestre em Informática.

Área de Concentração: Engenharia de Sistemas Computacionais.

Orientador: Prof.º Dr.º Erick de Andrade Barboza

Maceió-AL

2024

Catálogo na Fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico

Bibliotecário: Marcelino de Carvalho Freitas Neto – CRB-4 - 1767

P436w Pereira, Emanuel Adler Medeiros.
 Water potability classification : an approach using machine
 learning in an embedded system / Emanuel Adler Medeiros Pereira. –
 2024.
 44 f. : il.

Orientadora: Erick de Andrade Barboza.
Dissertação (mestrado em informática) - Universidade Federal de
Alagoas. Instituto de Computação. Maceió, 2024.

Bibliografia: f. 40-43.
Apêndices: f. 44.

1. TinyML. 2. Água potável. 3. Sistemas embarcados (Computadores). 4.
Aprendizagem de máquina. 5. Inteligência artificial. 6. Random Forest. 7.
Redes neurais. I. Título.

CDU: 004.383.8



MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE ALAGOAS
INSTITUTO DE COMPUTAÇÃO
Av. Lourival Melo Mota, S/N, Tabuleiro do Martins, Maceió - AL, 57.072-970
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO (PROPEP)
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Folha de Aprovação


EMANUEL ADLER MEDEIROS PEREIRA

**CLASSIFICAÇÃO DA POTABILIDADE DA ÁGUA: UMA ABORDAGEM
UTILIZANDO APRENDIZAGEM DE MÁQUINA EM SISTEMA EMBARCADO**


**WATER POTABILITY CLASSIFICATION: AN APPROACH USING MACHINE
LEARNING IN AN EMBEDDED SYSTEM**

Dissertação submetida ao corpo docente do
Programa de Pós-Graduação em Informática
da Universidade Federal de Alagoas e
aprovada em 26 de julho de 2024.


Banca Examinadora:

Documento assinado digitalmente
 **ERICK DE ANDRADE BARBOZA**
Data: 26/07/2024 09:28:03-0300
Verifique em <https://validar.iti.gov.br>

Prof. Dr. ERICK DE ANDRADE BARBOZA
UFAL – Instituto de Computação
Orientador

Documento assinado digitalmente
 **ICARO BEZERRA QUEIROZ DE ARAUJO**
Data: 26/07/2024 09:32:19-0300
Verifique em <https://validar.iti.gov.br>

Prof. Dr. ICARO BEZERRA QUEIROZ DE ARAUJO
UFAL – Instituto de Computação
Examinador Interno

Documento assinado digitalmente
 **ALLAN DE MEDEIROS MARTINS**
Data: 26/07/2024 11:39:55-0300
Verifique em <https://validar.iti.gov.br>

Prof. Dr. ALLAN DE MEDEIROS MARTINS
UFRN - Universidade Federal do Rio Grande do Norte
Examinador Externo

Dedicatória

*Dedico este trabalho a todos que sonham e acreditam
que podem mudar a própria realidade.*

“Dreaming is believing, your wishing well.” - ANGRA.

Agradecimentos

Agradeço primeiramente a Deus, que me guiou por todo o caminho até mais uma conquista, sem deixar que eu desistisse nos momentos mais difíceis.

Agradeço muito à minha família, por todo o suporte ao longo de toda a minha vida. Foram muitos sacrifícios, especialmente das minhas três mães Amazilde, Elúzia e Janicledja, para que eu pudesse ter a oportunidade de ter uma boa educação e ser tudo aquilo que um dia sonhei. Sou muito grato também à minha esposa Camylla, pois sem o apoio diário dela, jamais teria alcançado o fim desta etapa.

Agradeço à Universidade Federal de Alagoas e seus professores, em especial ao meu orientador, Erick Barboza, por toda a sua dedicação e tempo, possibilitando a realização deste trabalho e contribuindo de maneira significativa com meu crescimento pessoal e profissional.

Por fim, agradeço aos amigos e colegas que contribuíram direta e indiretamente com esta minha jornada, em especial ao colega Jeferson Santos, que me auxiliou com a importante tarefa de medir o consumo de energia neste estudo. O caminho se torna muito menos tortuoso quando temos boas companhias para compartilhar a vida.

Resumo

O acesso à água potável é um recurso vital e um direito humano reconhecido. Contudo, ainda hoje, bilhões de pessoas sofrem com a falta de acesso à água adequada para consumo, o que pode levar a diversos problemas de saúde. Um dos principais desafios no monitoramento da qualidade da água é a coleta e análise de grandes volumes de dados. Modelos de Aprendizado de Máquina têm sido amplamente aplicados no monitoramento da qualidade da água para facilitar a tomada de decisão por gestores e prevenir a contaminação. Um sistema embarcado que integre sensores a um modelo de Aprendizado de Máquina poderia oferecer respostas em tempo real e seria viável para ser aplicado em qualquer local, independentemente da conexão com a internet. Esse sistema, no contexto da classificação da potabilidade da água, permitiria respostas mais rápidas diante de potenciais ameaças. Este estudo propõe um modelo de TinyML eficiente em termos energéticos para a classificação da potabilidade da água, utilizando apenas parâmetros que podem ser obtidos por meio de sensoriamento eletrônico. O estudo avaliou o desempenho utilizando métricas como Acurácia, Precisão, Recall, F1-Score, espaço ocupado em memória pelo modelo, tempo de execução e consumo de energia, e comparou modelos desenvolvidos com os algoritmos Random Forest e Redes Neurais. Também foi analisada a melhor combinação entre modelo e biblioteca de adaptação para o sistema embarcado. O modelo de Aprendizado de Máquina inicial, utilizando Random Forest, demonstrou um bom desempenho alcançando uma Precisão de 0.70 e pode funcionar por anos com uma bateria comum como fonte de energia. Comparando todos os modelos e bibliotecas do estudo, o modelo de perceptron multicamadas com a biblioteca EmbML usou a menor memória, com 283.113 bytes, e o modelo Random Forest com Micromlgen teve o menor consumo de energia, usando apenas 104.534 milijoules. Este trabalho pode ajudar pesquisadores e profissionais a implementar sistemas de classificação de potabilidade da água e a usar TinyML em outros problemas de classificação também.

Palavras-chave: TinyML, Água, Potabilidade, Sistemas Embarcados, Aprendizagem de Máquina, Inteligência Artificial, Random Forest, Redes Neurais.

Abstract

Access to clean drinking water is a vital resource and a recognized human right. However, billions of people still suffer from the lack of access to safe drinking water, leading to various health issues. One major challenge in water quality monitoring is the collection and analysis of large amounts of data. Machine learning models have been widely applied in water quality monitoring to aid decision making by managers and prevent contamination. An embedded system that integrates sensors with a Machine Learning model could provide real-time responses and be feasible for deployment anywhere, regardless of internet connectivity requirements. Such a system, in the context of water potability classification, would allow faster responses to potential threats. This study proposes an energy-efficient TinyML model for classifying water potability, using only parameters available through electronic sensing. The study evaluated performance using metrics such as Accuracy, Precision, Recall, F1-Score, memory occupied by the model, execution time, and energy consumption, comparing models developed with Random Forest and Neural Networks algorithms. It also assessed the best combination of model and adaptation library for the embedded system. The initial Machine Learning model, using Random Forest, demonstrated good performance, reaching a Precision of 0.70, and compared to its cloud-based counterpart, it can operate for years on a standard battery power source. When comparing all models and libraries in the study, the multilayer perceptron model with the EmbML library used the least memory, with 283,113 bytes, and the Random Forest model with Micromlgen had the lowest energy consumption, using only 104.534 millijoules. This work can help researchers and professionals implement water potability classification systems and use TinyML in other classification problems as well.

Keywords: TinyML, Water, Potability, Embedded Systems, Machine Learning, Artificial Intelligence, Random Forest, Neural Networks.

List of Figures

1	Types of Machine Learning and some problems that can be solved by them.	6
2	Work flowchart for a classification model (Adapted from (SEN <i>et al.</i> , 2020)).	7
3	Example of a confusion matrix for a binary classification problem.	8
4	Traditional workflows used in TinyML (Adapted from (CAPOGROSSO <i>et al.</i> , 2024)).	10
5	Distribution of the ‘Potability’ Column in the Dataset.	20
6	Description of the components of the ESP32 board manufactured by Espressif (Source: (Espressif Systems (Shanghai) Co., 2023)).	25
7	Tools used to measure the power consumption of the ESP-32 board.	27
8	Print screen of the Nordic Power Profile Software used to measure the power consumption of the ESP-32 board, highlighting the measurement that was discarded and the measurements that were selected to calculate the average amount of charge.	27
9	Architectures used in the two machine learning models developed during the study.	28
10	Confusion matrices from the versions of the RF models built in this work. .	32

List of Tables

1	Metrics of the Random Forest models from the articles that use the same dataset.	18
2	Statistical analysis of the dataset features. The column StdDev means Standard Deviation, while Min and Max mean minimum and maximum, respectively.	19
3	Parameters used for training the RF model using Sci-kit Learn library. . . .	24
4	Parameters used for training the MLP Model.	29
5	Parameters used for training the TF Model.	29
6	Metrics comparing all Random Forest models. The “Adapted Model” is the one reproduced by this work using Ivanov’s model as a basis but considering only electronic sensors, the “Embedded Model” is the adapted model running on the ESP-32 board, and the “Cloud Model” is the adapted model running on Google Cloud and being consumed by the ESP-32 board.	33
7	Metrics comparing all libraries used for the developed machine learning models, highlighting the best results for each metric.	36

Contents

1	Introduction	1
1.1	Motivation of the Work	1
1.2	Objectives	2
1.2.1	General Objective	2
1.2.2	Specific Objectives	2
1.2.3	Research Questions	3
1.3	Structure of the Work	3
2	Theoretical Foundation	4
2.1	Water Quality Monitoring	4
2.2	Machine Learning and Classification	5
2.2.1	Evaluation metrics for classification models	7
2.3	TinyML	9
3	Related Works	14
3.1	Machine Learning and IoT for Water Quality Classification	14
3.2	Relevance of the Proposal	20
4	Materials and methods	22
4.1	Adapting the Random Forest model to work with electronic sensors	22
4.2	Embedding the first Random Forest model	24
4.3	Developing the neural network models	28
4.4	Embedding and comparing multiple machine learning models and TinyML frameworks	30
5	Results and discussion	31
5.1	Adapting the Random Forest model to work with electronic sensors	31
5.2	Results of embedding the first Random Forest model	33
5.3	Comparing results for multiple machine learning models and TinyML frameworks	35
6	Conclusion	38

References	40
Appendices	
Appendix A Publications generated from this work	44

1 Introduction

This chapter outlines the driving factors behind conducting a study to develop a TinyML model for classifying water potability, using data exclusively from electronic sensors.

1.1 Motivation of the Work

Access to clean drinking water and basic sanitation is considered a fundamental human right due to its importance for public health and quality of life. According to (WHO - World Health Organization, 2024) data for 2020, approximately 2 billion people worldwide still lack access to clean water, being forced to use unfit water for consumption. This use of contaminated water is linked to various diseases, such as cholera and diarrhea. The improper handling of urban, industrial, and agricultural wastewater results in dangerous contamination or chemical contamination of the drinking water consumed by hundreds of millions of people. As pointed out by (LI; WU, 2019), over time, various chemicals that are untreated in water sources by different types of industry can be classified as a high risk to human health.

In general, large databases are required to study the parameters and variations in the quality of drinking water. A tool for these studies can be the use of Machine Learning, a field of Artificial Intelligence, which has been widely applied in water treatment and management systems, including the management of drinking water sources, treatment processes, water distribution, and decision making, as observed in the study by (ZHU *et al.*, 2022). Machine learning models can be applied to the identification and evaluation of drinking water quality can be used to prevent contamination.

Some solutions suggest, as the one proposed by (KODITALA; PANDEY, 2018), integrating these intelligence models with other technologies such as Cloud Computing and the Internet of Things (IoT), using embedded devices, sensors, and Internet connectivity. However, these solutions typically perform the inference of the model developed in the cloud, lacking solutions that propose execution on the embedded device. Another common limitation found in existing solutions in the literature relates to the use of data derived from parameters that cannot be acquired through electronic sensors and require chemical and laboratory analysis for their collection. Acquiring this type of data entails significant material

and labor costs compared to the expense of obtaining other data through electronic sensing.

The effort to embed machine learning models in resource constraint devices is called TinyML. (DUTTA; BHARALI, 2021) defines TinyML as an emerging concept focused on running optimized machine learning models on ultra-low power microcontrollers that consume less than 1 mW of power. TinyML stands out as a valuable tool in these settings by enhancing processing capabilities. It achieves this by enabling data processing and Machine Learning services directly within the device, thereby significantly boosting the device's functionality. This effort is made because of two main advantages. First, executing a machine learning model on the embedded device means that the solution does not depend on a communication infrastructure, making its implementation simpler and suitable for a wider variety of locations. Furthermore, because the embedded device does not need to communicate with other devices, it consumes less energy, which makes it capable of operating on battery power for extended periods. Therefore, by running the machine learning model, the solution becomes more versatile, energy efficient, and potentially more cost effective.

1.2 Objectives

1.2.1 General Objective

The aim of this study is to create an energy-efficient machine learning model to classify the potability of water that can be used in an embedded system, considering only data that can be acquired by electronic sensors, and that this model does not require the use of the cloud for its execution.

1.2.2 Specific Objectives

1. Assess the quality of a machine learning model based on literature references, using only sensors that can be embedded.
2. Analyze the feasibility of embedding this model.
3. Compare the performance between different Machine Learning models and TinyML libraries for water potability classification.

1.2.3 Research Questions

- **RQ1:** How different is the performance of the machine learning model when reducing the number of features to consider only sensors that can be embedded?
- **RQ2:** What is the performance outcome when embedding the reference machine learning model in comparison to results found in the literature?
- **RQ3:** How different is the performance of the embedded model compared to the cloud model?
- **RQ4:** Does an embedded model with a neural network algorithm perform better in this problem than the one with a Random Forest algorithm?
- **RQ5:** Which combination of algorithm and TinyML library adaptation performs best under similar conditions for this problem?

1.3 Structure of the Work

This document is organized as follows. Chapter 2 addresses the theoretical foundation of the main topics necessary to understand the methods that will be described throughout this research proposal. Chapter 3 provides a review of related work and outlines the starting point of this research. Chapter 4 describes the research methodology adopted, detailing the set of tasks performed during the different phases of this work, and the tools used. Chapter 5 presents the results achieved, along with a discussion of them and their impact. Finally, Chapter 6 offers the final conclusions of this research and suggests directions for future work.

2 Theoretical Foundation

This chapter delves into the theoretical framework to enhance understanding of the problem under analysis. It covers concepts related to water quality monitoring, machine learning and classification methods, and TinyML as well.

2.1 Water Quality Monitoring

Water quality monitoring is defined by the International Organization for Standardization (ISO) as a planned process of sampling, measuring, and subsequently recording or signaling various water characteristics, aiming to assess their compliance with specified objectives (ISO *et al.*, 2006). These specified objectives are related to how the water will be used. For example, for agricultural use purposes, the sodium content must be low to avoid harming the soil. For human consumption, there should be no microorganisms or harmful chemical substances. This monitoring enables competent authorities to make rational decisions based on all obtained information (BARTRAM; BALLANCE, 1996). Therefore, online and large-scale water quality monitoring data are essential to detect environmental pollution and react in the best possible way to avoid risks to human health (STANDARDIZATION, 1991).

Collecting a large number of samples is one of the main challenges in the field, as it is necessary to ensure an accurate and reliable analysis of water parameters. The conventional processes required to monitor water quality involve manually collecting various samples, possibly from different points in the water body, followed by laboratory tests and analyses. Being a lengthy, laborious process, and sometimes incapable of providing real-time results, it can be considered ineffective for the purpose of promoting more proactive responses against water contamination (PULE *et al.*, 2017).

The use of remote sensing in water quality monitoring is feasible, as there are sensors that can detect physicochemical parameters such as pH, electrical conductivity, turbidity, and chlorine content (PULE *et al.*, 2017). There are also biosensors and optical sensors that can be used to measure other important parameters, such as the amount of bacteria or algae and dissolved oxygen. Each detection and sensing method has its strengths and weaknesses, and the choice of sensor type depends on the application's needs. Some optical sensors, for

example, require the addition of reagents to the water and some level of human manipulation. This type of sensor would not be suitable for an application that wishes to perform automatic and real-time monitoring of water parameters (KRUSE, 2018).

2.2 Machine Learning and Classification

Machine Learning originated in the 1960s as a branch of Artificial Intelligence with the goal of identifying patterns in complex data sets (IZBICKI; SANTOS, 2020). It can be understood as a set of computational techniques that use experience to improve performance or make accurate predictions. In this context, experience refers to the information available to the machine, usually in the form of previously collected data (MOHRI *et al.*, 2018). Machine Learning can also be defined as an interdisciplinary field that encompasses concepts and techniques from various areas, such as mathematics, statistics, information theory, game theory, and optimization (SHALEV-SHWARTZ; BEN-DAVID, 2014).

Machine Learning can be employed to solve a variety of classic problems extensively studied by the scientific community. Among these problems are Classification, Regression, Ranking, Clustering, and Dimensionality Reduction (MOHRI *et al.*, 2018). These techniques allow a machine learning model to be trained to classify items into specific categories, make numerical predictions, rank items according to a certain criterion, group similar datasets together, and reduce the complexity of datasets while preserving their essential characteristics.

Machine Learning types can be classified into Supervised Learning, Unsupervised Learning, and Reinforcement Learning (MOHRI *et al.*, 2018). A flowchart describing these types and some problems that can be solved using them can be seen in Figure 1. Below is a brief description of each:

1. **Supervised Learning:** In this type of learning, the model is trained using a set of labeled data. After that, it makes predictions for new data based on what it learned during training. Classification, Regression, and Ranking problems are commonly addressed by supervised learning algorithms, such as Random Forest, Support Vector Machine (SVM), and Neural Networks.
2. **Unsupervised Learning:** The model is trained only with unlabeled data, that is, without any prior information about expected outcomes. The goal is to find patterns, structures,

or groups in the data. Clustering and Dimensionality Reduction problems are common in unsupervised learning. Algorithms like K-Nearest Neighbors (KNN) and Principal Component Analysis (PCA) are examples of this type of learning.

3. **Reinforcement Learning:** The model actively interacts with the environment, collecting information and receiving rewards or penalties for its actions. The training and testing phases are mixed, and the goal is to learn to make decisions that maximize rewards over time. Algorithms like Q-Learning and Monte Carlo are examples of reinforcement learning.

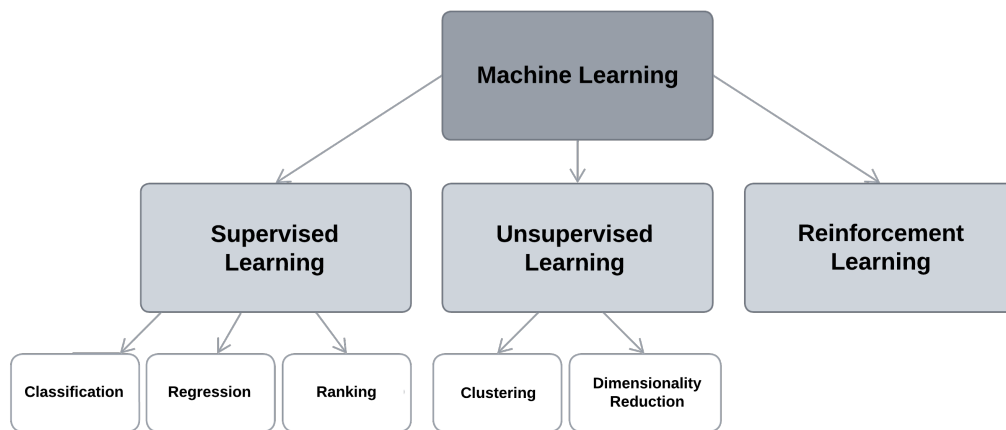


Figure 1: Types of Machine Learning and some problems that can be solved by them.

A flowchart for constructing a Supervised Learning model for classification can be seen in Figure 2. The main steps for this construction, according to (SEN *et al.*, 2020), can be:

1. Collect and clean the database or preprocess the data.
2. Initialize the classifier model.
3. Split the database using cross-validation and feed the model with training data.
4. Predict a label for new data not seen in training.
5. Evaluate the error rate of the classifier model on the test database.

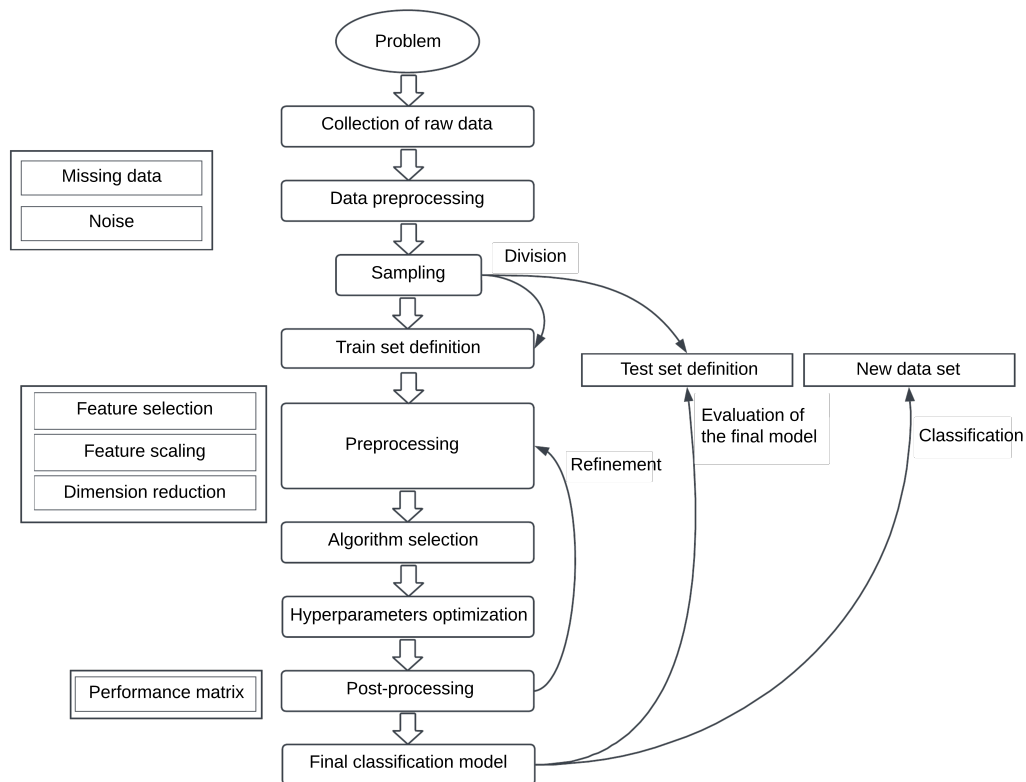


Figure 2: Work flowchart for a classification model (Adapted from (SEN *et al.*, 2020)).

2.2.1 Evaluation metrics for classification models

Evaluation metrics for a classification model serve as tools to quantify the model's effectiveness. The calculation of these metrics is crucial in the development of a classification model and is typically performed at the end of the training process, using a test dataset that the model has not encountered during training.

When analyzing the classification outcome of a model in a binary problem scenario, there are four categories into which the results can fall. A True Positive (TP) occurs when a model correctly classifies an outcome as positive. A True Negative (TN) occurs when a model correctly classifies an outcome as negative. A False Positive (FP) occurs when a model incorrectly classifies an outcome as positive when it is actually negative. A False Negative (FN) occurs when a model incorrectly classifies an outcome as negative when it is actually positive.

A visual representation of the performance of these classifications is the Confusion Matrix, that indicates the number of correct and incorrect predictions for each class. This

matrix aids in identifying which classes the model misclassifies as another class, as described by (BATARSEH; YANG, 2020). Figure 3 presents an example of a confusion matrix for a binary classification issue.

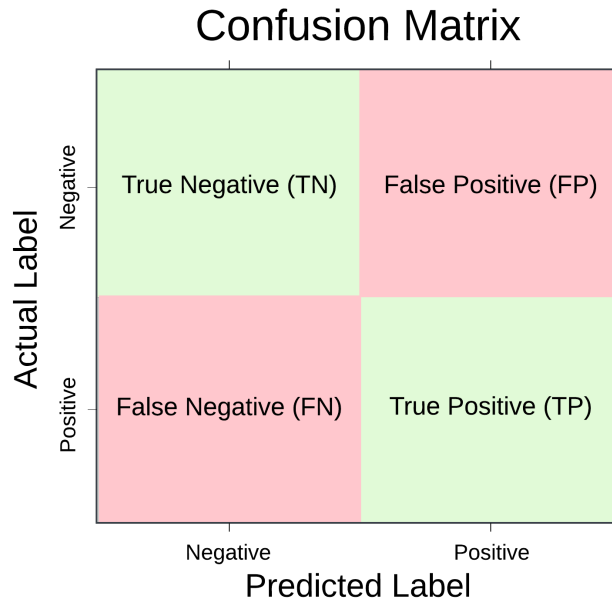


Figure 3: Example of a confusion matrix for a binary classification problem.

(BATARSEH; YANG, 2020) mentions that Accuracy is among the most frequently utilized metrics in classification tasks. The model's Accuracy is determined by the following formula:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy is effective when the class distribution is balanced, with an equal number of samples per class. However, it may provide misleading results with imbalanced datasets. Consequently, other metrics derived from the confusion matrix are recommended for a more reliable evaluation of performance.

Precision evaluates a machine learning model's performance by indicating the proportion of positive predictions that are actually correct. The calculation is as follows:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall quantifies the frequency with which a machine learning model accurately iden-

tifies true positives (TP) among all the actual positive instances present in the dataset. It is calculated as follows:

$$\text{Recall} = \frac{TP}{TP + FN}$$

The F1-Score combines precision and recall into a single measure, calculated as the harmonic mean of these two values. This metric provides insight into how precise (accurately classifying instances) and robust (minimizing missed instances) a classifier is.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

2.3 **TinyML**

TinyML is a significant advancement in artificial intelligence (AI) that focuses on executing AI algorithms on extremely low-power devices, such as microcontrollers. This approach addresses the high energy consumption and carbon dioxide emissions associated with traditional AI by reducing the computational load required for machine learning tasks. Unlike standard CPUs and GPUs, which consume considerable power, TinyML devices operate on milliwatts or microwatts, enabling them to run with batteries for extended periods, from weeks to years. TinyML pushes intelligence to the edge, allowing AI applications to function with low latency, minimal power consumption, and low bandwidth requirements (ABADADE *et al.*, 2023).

By processing data locally on the device where it is generated, TinyML also enhances data privacy and security, as the data does not need to be transmitted to the cloud. This makes TinyML particularly suitable for applications in environments where internet connectivity is unreliable or unavailable (ABADADE *et al.*, 2023). TinyML applications span various domains, including industrial anomaly detection, where it can effectively identify potential failures in extreme environments such as submersible pumps (ANTONINI *et al.*, 2023), and medical applications, where this technology can enable autonomous and safe healthcare by integrating Machine Learning algorithms into wearable devices (TSOUKAS *et al.*, 2021). This innovative field of AI opens up new possibilities for sustainable development and privacy-preserving machine learning applications.

According to (CAPOGROSSO *et al.*, 2024), TinyML systems consist of two key components: the machine learning model and the hardware platform. Developers typically start with the component they are most familiar with, but a more efficient approach is to develop both simultaneously for an integrated solution. The traditional workflows in TinyML are ML-oriented and HW-oriented, whereas the integrated approach is known as co-design. These workflows are described in Figure 4.

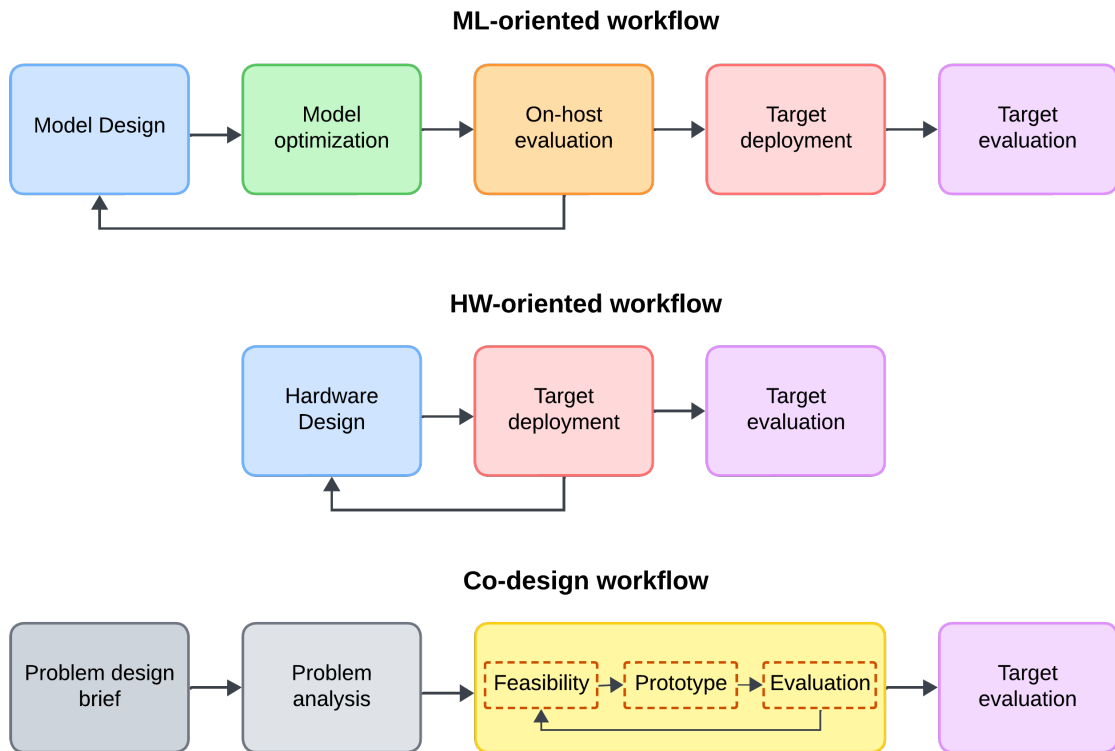


Figure 4: Traditional workflows used in TinyML (Adapted from (CAPOGROSSO *et al.*, 2024)).

In the ML-oriented workflow, the focus is primarily on designing, adapting, training, and evaluating machine learning models, with limited flexibility in hardware platform selection due to specific industrial requirements. An example is adapting modern neural network models for embedded devices, which requires extensive experimentation to ensure efficiency in power consumption, latency, and memory usage, given the limited resources compared to cloud solutions (CAPOGROSSO *et al.*, 2024).

In the HW-oriented approach, developers focus on creating enhanced hardware platforms optimized for embedded applications to run state-of-the-art machine learning algo-

rithms. This often involves addressing bottlenecks in existing architectures, such as improving throughput and consumption in neural network computations by designing hardware accelerator modules. Examples include reducing computational complexity in convolution layers, developing efficient low-power perceptrons, and improving data caches (CAPOGROSSO *et al.*, 2024).

Finally, in the co-design workflow, both model optimization and hardware design are integrated from the outset to achieve significant improvements in performance and resource consumption. Unlike the separate steps in the ML-oriented and HW-oriented workflows, co-design involves intertwined and co-optimized processes. This approach can result in custom architectures for specific machine learning algorithms on FPGAs or enable neural network computations on customized accelerators using analog Compute-in-Memory (CiM) hardware through HW-informed training methodologies (CAPOGROSSO *et al.*, 2024).

In (ABADADE *et al.*, 2023) the authors state that TinyML leverages software to deploy machine learning models on resource-constrained hardware by optimizing model size and computational needs, enabling integration into various devices. Some key techniques include:

- **Pruning:** This technique involves training a neural network and then removing less important connections by identifying weights below a specific threshold. While the initial pruned model may lose some accuracy, retraining the remaining weights can restore it. Pruning also helps eliminate connections and neurons without inputs or outputs.
- **Quantization:** This reduces the precision of weights and activations from 32-bit or 64-bit floating-point numbers to 8-bit or lower fixed-point numbers. This not only decreases the model's memory footprint but also improves processing efficiency. Quantization can be done during or after training, aiming to balance model accuracy with precision.
- **Low-Rank Factorization:** This mathematical technique approximates a high-dimensional matrix with a low-dimensional one, reducing dimensionality while preserving important information. It decomposes a dense weight matrix into two lower-dimensional matrices, making the model more compact and computationally efficient.

- **Huffman Coding:** As a lossless compression method, Huffman coding assigns shorter binary codes to frequently occurring symbols and longer codes to less frequent ones. This reduces the overall space required for data representation without losing any information.
- **Knowledge Distillation:** This technique involves training a smaller, more compact model (student model) to mimic the outputs of a larger, more accurate model (teacher model). The student model learns useful information from the teacher model, making it more efficient without needing to be as complex. The training involves two steps: training the teacher model on the original data and then training the student model using the teacher model's predictions.
- **Hyperparameter Optimization:** Automates the search for optimal hyperparameter values, such as learning rate, batch size, and network size, to enhance model performance. Techniques like Grid Search, Random Search, and Bayesian Optimization explore the hyperparameter space to find the best combination, reducing effort and time while improving performance (CAPOGROSSO *et al.*, 2024).

In (CAPOGROSSO *et al.*, 2024) the authors highlight several challenges that future research in the field of TinyML needs to address:

- **Benchmarking:** The absence of a recognized benchmark, caused by challenges such as low power, limited memory, hardware heterogeneity, and software heterogeneity, poses a significant impediment that could hinder TinyML services.
- **Memory Constraints:** The relentless demand for computation and high accuracy has driven continuous innovation in machine learning algorithms. However, the extremely limited size of SRAM and flash memory makes deploying deep learning on edge devices a significant challenge.
- **Data-driven engineering:** Thoroughly understanding data quality is crucial, relying solely on accuracy can be misleading for predicting model behavior. It is necessary a substantial amount of relevant real-world data to identify instances where the model fails or behaves incorrectly.

-
- **Lack of accepted models:** Many deep learning models are widely accepted for conventional infrastructure. For instance, MobileNet serves as the benchmark for deep neural networks in mobile edge computing devices. However, no similarly popular model exists for adoption within the TinyML ecosystem on MCUs.
 - **Lack of public datasets:** Despite the availability of some datasets specifically designed for TinyML, to date, TinyML is mainly focused on general sensor processing.

3 Related Works

This chapter presents studies from the scientific literature that directly relate to the objectives of this study. The literature review is organized into two subsections. In subsection 3.1, the studies related to proposals for water quality classification systems using Machine learning and IoT are introduced. Finally, in subsection 3.2, the significance of this dissertation proposal is emphasized.

3.1 Machine Learning and IoT for Water Quality Classification

The study (C.ASHWINI *et al.*, 2019) proposes an intelligent water quality monitoring system for domestic use utilizing the Internet of Things (IoT). The system comprises a NodeMCU controller and sensors to collect pH, turbidity, dissolved oxygen, temperature, color, and conductivity data. Sensor readings are sent over the network and stored in a database. The data in this database are fed into a neural network model, which predicts the quality of the water and sends alerts to the user. The neural network was trained with 198 databases.

The study (ALZUBI, 2022) offers a solution for monitoring water quality using IoT and Machine Learning. Unlike (C.ASHWINI *et al.*, 2019), it proposes a fusion of the K-Nearest Neighbor (KNN) and Support Vector Machine (SVM) algorithms for a more accurate classification. The system uses sensors for pH, temperature, turbidity, and conductivity. Sensor values are transferred to a cloud server via the NodeMCU controller with Low Power Wide Area Network (LPWAN) technology. The performance of the proposed classifier is evaluated through cross-validation, achieving higher accuracy than other algorithms compared. In addition to the good performance of the classifier, the study highlights the low cost and applicability of the system in a real-time scenario.

The study (JHA, 2020) aims to classify groundwater quality using a system composed of a microcontroller and sensors in a tank. The architecture involves connecting sensors to the microcontroller via Zigbee/Wi-Fi technology. The system uses a cloud platform to store and analyze sensor data, with water quality classification done using the Decision Tree algorithm. The algorithm achieved high accuracy with a database of 307 records, and the system met its objective.

The study by (ABDULWAHID, 2020) proposes an IoT-based system for real-time monitoring of water quality, leveraging affordable technology and cloud computing. The system integrates several sensors, such as turbidity, temperature, total dissolved solids, and pH, connected to an Arduino unit, which are designed to measure key water quality parameters. Data from these sensors are processed by a microprocessor and transmitted to the ThingSpeak API via a network. The key advantages of the proposed system include efficient data processing, affordability, and the capability to visualize data on cloud platforms and mobile devices using Wi-Fi.

The research by (ALIPIO, 2020) presents the development of an IoT-based system for monitoring water quality and classifying water potability, specifically designed for rural areas in developing countries. The system incorporates portable sensor nodes that gather physicochemical properties of water such as pH, turbidity, total dissolved solids, and temperature from various sources. These nodes wirelessly transmit data to a base station that performs potability classification using ensemble learning techniques. Results are communicated back to households in real-time via 2G/3G networks and are also sent to a cloud server for remote monitoring. The system demonstrated a 93.33% match with conventional industrial water tests and achieved a 97% accuracy in classification. The authors point out that future improvements could include incorporating additional water parameters like dissolved oxygen and advanced machine learning algorithms to enhance the accuracy and viability of the system.

The paper by (MUKTA *et al.*, 2019) introduces an IoT-based smart water quality monitoring system designed to continuously measure water quality through four physical parameters: temperature, pH, electrical conductivity, and turbidity. Utilizing an Arduino Uno connected to four sensors, the system captures data and transmits it to a desktop application developed on the .NET platform. This data is compared against WHO standard values to determine water potability. The system employs a fast forest binary classifier to assess whether water samples are drinkable, demonstrating effective and accurate performance in real-time water quality prediction. The successful implementation of this IoT-enhanced system showcases its potential for expanded real-time monitoring, including the detection of chemical parameters in the future.

The publication by (BRIA *et al.*, 2020) explores a sensor-based system for identi-

fyng six contaminants in tap water, testing nine different classifiers to find an IoT-ready solution. The K-nearest Neighbor (KNN) classifier performed best, achieving a top accuracy of 95.4%. The study found that Multi-Layer Perceptron offers the best trade-off in terms of classification performance, memory usage, and processing time when applied to low-end MCUs. Future work will focus on advancing data processing and classification capabilities directly on MCUs and further improving system performance with high-end MCUs.

The study by (RAWAT *et al.*, 2023) evaluates the effectiveness of eight machine learning algorithms, including Gaussian Naive Bayes, Extreme Gradient Boost, SVM, KNN, Logistic Regression, Random Forest, and Decision Tree, for predicting water quality. The main goal was to identify the most accurate algorithm, with Extreme Gradient Boost and SVM showing the highest accuracies of 69.89% and 65.87% respectively after optimization. The results demonstrate that machine learning can be effectively used to predict water quality, suggesting its potential application in managing and monitoring water systems. The findings encourage further research to optimize these algorithms and test them on larger datasets to improve their predictive performance.

The research by (ALNAQEB *et al.*, 2022) focuses on addressing the critical environmental issue of water quality by designing an intelligent system that employs machine learning models to assess and predict water potability. It compares various machine learning algorithms, such as Decision Tree, K-Nearest Neighbor, Support Vector Machine, Random Forest, and LightGBM, in order to identify the most accurate model for predicting water quality. The study found that the LightGBM model outperformed others, achieving the highest prediction accuracy of 99.74% on experimental data.

The research by (AHMED *et al.*, 2019) addresses the urgent need for cost-effective and real-time water quality monitoring by employing machine learning algorithms to estimate the Water Quality Index (WQI) and classify Water Quality Class (WQC) using just four parameters: temperature, turbidity, pH, and total dissolved solids. Among various algorithms tested, gradient boosting and polynomial regression demonstrated the most efficient predictions of WQI with mean absolute errors of 1.9642 and 2.7273, respectively, while a multi-layer perceptron (MLP) model proved most effective for classifying WQC with 85% accuracy. This streamlined approach contrasts with traditional methods that rely on extensive lab analyses involving numerous parameters, making it suitable for integration into an

inexpensive, real-time detection system. Future enhancements envision implementing these findings in a large-scale IoT-based monitoring system that could provide immediate water quality assessments and alerts, thereby improving public health outcomes.

The research conducted by (KADDOURA, 2022) aimed to assess the efficiency of using machine learning techniques for classifying water potability. This work used an open-access database with water quality parameters. In the data preprocessing phase, rows with null data were removed and normalization was applied. Subsequently, the data were split into training and testing sets in an 80/20 ratio. A performance comparison of various supervised learning algorithms was conducted, using the precision, recall, F1-Score, and ROC AUC metrics.

The study by (PRIYADARSHINI *et al.*, 2022) had the same objective as the previous study and used the same database. The preprocessing step did not explicitly detail the data cleaning strategy, but the split ratio between train and test remained consistent. Regarding the metrics employed, the key difference was the use of Accuracy instead of ROC AUC for model evaluation. Emphasizing Accuracy, the publication identified the Random Forest model as the best performer. The study also conducted a comparative analysis of Machine Learning models using a data set related to the quantity of marine litter.

The research by (IVANOV; TOLEVA, 2023) addresses the critical ecological issue of water potability, proposing a simplified, effective machine learning algorithm designed to predict water quality across various regions. It enhances Decision Tree, Support Vector Machine, and Random Forest by improving accuracy and classification metrics and effectively manages class imbalances. The study demonstrates that this algorithm can quickly evaluate water quality without extensive parameter tuning or the need for balancing techniques, making it particularly useful for initial assessments of imbalanced water data. The metrics used to evaluate the models were the same used by (PRIYADARSHINI *et al.*, 2022).

The article by (ARORA *et al.*, 2022) explores the critical issue of water potability by utilizing various machine learning algorithms to classify whether water is potable based on its chemical and physical properties. With a focus on a binary classification of water potability, the study tests algorithms such as K-Nearest Neighbors, XG-Boost, Decision Tree, SVM, and Random Forest, among others, applying them to a dataset filtered by a correlation matrix. Among these, the Decision Tree model outperformed the rest, achieving the highest

scores with an accuracy of 0.9358, an F1 score of 0.9374, and an ROC AUC score of 0.9220. The primary aim of this paper was to harness machine learning's capability to classify water samples as potable or non-potable based on chemical factors such as pH and turbidity. In comparing various algorithms, Random Forest was found to provide the best accuracy among all models evaluated.

Table 1 presents a comparison of the accuracy, ROC AUC, precision, recall, and F1 score metrics for the Random Forest models used in the four previously mentioned publications. These publications utilized the same dataset, and the Random Forest model yielded noteworthy results in each study.

Table 1: Metrics of the Random Forest models from the articles that use the same dataset.

Metric	(KADDOURA, 2022)	(PRIYADARSHINI <i>et al.</i> , 2022)	(IVANOV; TOLEVA, 2023)	(ARORA <i>et al.</i> , 2022)
Accuracy	-	0.81	0.81	0.82
ROC AUC	0.702	-	-	0.786
Precision	0.46	0.80	0.82	0.86
Recall	0.93	0.91	0.81	0.64
F1-Score	0.61	0.85	0.81	0.73

The database used by the publications in Table 1, entitled "*Water Quality - Drinking water potability*" from (KADIWAL, 2021), is available online and contains water quality metrics for 3276 different entries of water bodies. The presented metrics are: pH value; Hardness; Solids (Total dissolved solids - TDS); Chloramines; Sulfate; Conductivity; Organic carbon (Total organic carbon - TOC); Trihalomethanes (THM); Turbidity; Potability.

According to the documentation of the dataset, the potability column specifies the suitability of water for human consumption, with a value of 1 indicating potability and 0 indicating non-potability. The data for pH are an indicator of the acidic or alkaline condition of the water. Hardness in water primarily results from the presence of calcium and magnesium salts, which dissolve from geological deposits encountered during the water's journey. High levels of total dissolved solids (TDS) signify highly mineralized water, resulting in an undesirable taste and a diluted color in its appearance. Chloramine serves as a disinfectant used in public water systems, typically produced by adding ammonia to chlorine during the treatment of drinking water. Sulfates, naturally occurring substances found in minerals, soil, rocks, ambient air, groundwater, plants, and food, have a primary commercial application in

the chemical industry.

Pure water, being a good insulator, does not conduct electric current effectively. However, the conductivity of water improves with an increase in the concentration of ions. In general, the electrical conductivity of water is influenced by the quantity of dissolved solids present.

The origin of Total Organic Carbon (TOC) in source waters can be traced to the decomposition of natural organic matter (NOM) and synthetic sources. TOC serves as a metric to quantify the total carbon content within the organic compounds present in pure water.

Trihalomethanes (THMs) are substances that may be detected in water treated with chlorine. The presence and concentration of THMs in drinking water are influenced by factors such as the organic content of the water, the amount of chlorine needed for the treatment of the water, and the temperature of the water undergoing treatment.

The water turbidity is dependent on the amount of solid matter present in a suspended state. This parameter gauges the light-emitting characteristics of water and serves as a test to assess the quality of waste discharge with respect to colloidal matter.

Table 2 presents a statistical analysis of the parameters in the database, including the count of items, mean, standard deviation, and minimum and maximum values. Figure 5 displays the distribution of values in the column representing water potability in the dataset, highlighting a higher number of values corresponding to the non-potable water category.

Table 2: Statistical analysis of the dataset features. The column StdDev means Standard Deviation, while Min and Max mean minimum and maximum, respectively.

Parameter	Count	Mean	StdDev	Min	Max
pH	2785	7.08	1.59	0	14
Hardness	3276	196.37	32.88	47.43	323.12
Solids	3276	22014.09	8768.57	320.94	61227.20
Chloramines	3276	7.12	1.58	0.35	13.13
Sulfate	2495	333.78	41.42	129	481.03
Conductivity	3276	426.20	80.82	181.48	753.34
Organic carbon	3276	14.28	3.31	2.20	28.30
Trihalomethanes	3114	66.40	16.18	0.74	124
Turbidity	3276	3.97	0.78	1.45	6.74
Potability	3276	0.39	0.49	0	1

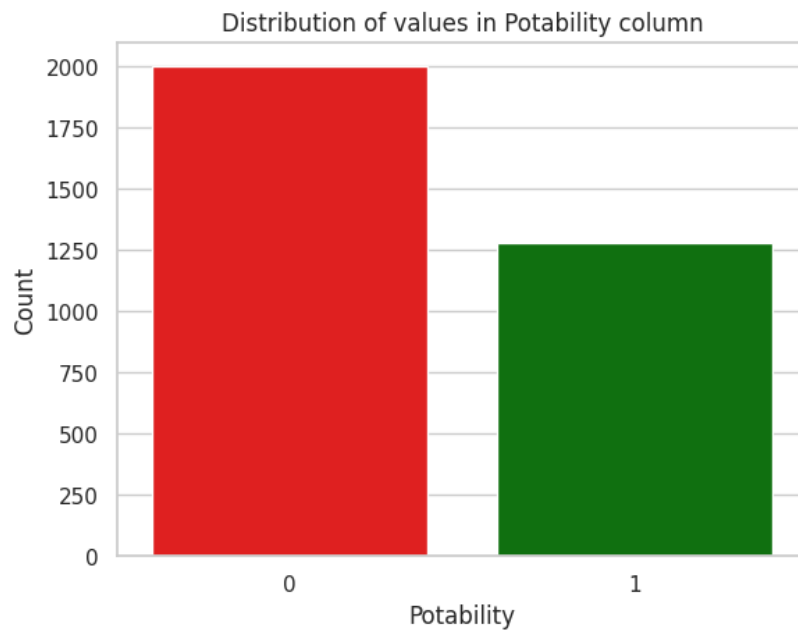


Figure 5: Distribution of the ‘Potability’ Column in the Dataset.

Following (PULE *et al.*, 2017), the use of remote sensing is possible to monitor water quality, as there are sensors capable of detecting physical and chemical parameters such as pH, electrical conductivity, turbidity, and chlorine levels. The study by (LVOVA *et al.*, 2019) demonstrates the effectiveness of chemical sensors in water quality monitoring systems, highlighting their ability to detect various water pollutants such as transition and heavy metals, algal toxins, herbicides, and pesticides. There are also biosensors and optical sensors that can be employed to measure other important parameters, such as the amount of bacteria or algae and dissolved oxygen. The research by (MANJAKKAL *et al.*, 2021) explores innovative sensor deployment and intelligent data analysis for online real-time monitoring of water quality, focusing on the use of multiparametric sensor systems coupled with smart algorithms to standardize data analysis globally, which improves monitoring in various parameters of water quality and addresses critical issues such as food safety and water-related diseases.

3.2 Relevance of the Proposal

The studies (C.ASHWINI *et al.*, 2019) and (ALZUBI, 2022) mention the use of sensors that are more accessible for reproducing the work or even for system production. There

are certain water quality parameters that cannot be automatically obtained through sensing, requiring human intervention through laboratory analysis or the use of reagent substances in the water, making it challenging to implement machine learning models with these parameters in embedded systems. The study developed in this proposal will aim to use only those parameters in the adopted models that can be automatically sensed.

(C.ASHWINI *et al.*, 2019; ALZUBI, 2022; JHA, 2020; ABDULWAHID, 2020; ALIPIO, 2020; MUKTA *et al.*, 2019; BRIA *et al.*, 2020) refer to the use of cloud-based system for the inferences from the intelligence models, not on the devices themselves, implying the need for an internet connection for the proposed systems to function. This requirement for internet connectivity may render the application of these systems infeasible in remote areas or those with low coverage, such as rural zones. Studies (IVANOV; TOLEVA, 2023; RAWAT *et al.*, 2023; ARORA *et al.*, 2022) focus on comparing different machine learning models for water quality prediction but do not explore the possibility of deploying them in embedded devices to evaluate their performance. Another research gap is the lack of attention to energy consumption, which is crucial for IoT and TinyML applications.

Therefore, since no studies in the literature tried to embed the classification model, and real-world solutions should benefit for running on-device, it is relevant to construct a Machine Learning model for classifying water potability that considers sensor accessibility and tests the model inference on an embedded device. Through a performance comparison with the same solution using cloud processing, it is possible to verify the viability of the embedded solution. It is also relevant to find the best combination of algorithm and library to ensure that the model performs optimally.

4 Materials and methods

In this chapter, the methodology used to carry out this research is described. This chapter is divided into four subsections, each representing a stage of the work. The subsection 4.1 outlines the set of tasks undertaken to create and tailor the machine learning model to classify the potability of water using only columns of the database that represent the parameters available through electronic detection. Subsection 4.2 refers to the set of tasks performed to implement the first Random Forest model developed on an embedded device and evaluate its performance. Subsection 4.3 outlines the steps taken to build two neural network models and assess their performance, specifically to draw comparisons with the previous implementation using Random Forest. Subsection 4.4 outlines the procedures followed to evaluate various machine learning models and TinyML libraries used in the development of embedded models.

4.1 Adapting the Random Forest model to work with electronic sensors

The following steps were taken to answer **RQ1** - "How different is the performance of the machine learning model when reducing the number of features to consider only sensors that can be embedded?":

1. The dataset, the same used by the reference machine learning model from the literature, has been modified to remove columns representing parameters that cannot be obtained through electronic sensing;
2. The dataset was preprocessed and split using the same method described in the reference study;
3. A machine learning model was trained using the Random Forest algorithm;
4. The hyperparameters of this model were optimized;
5. The performance of this model was evaluated using the test dataset;
6. The results of the model were compared with the reference machine learning model from the literature.

Each method of detection of water quality parameters has its strengths and weaknesses, with the choice of sensor type depending on the specific needs of the application. For instance, some optical sensors require the addition of reagents to water and a certain level of human manipulation. This type of sensor would not be suitable for an application that aims to automatically and in real time monitor water parameters, as indicated by (KRUSE, 2018). Due to this, in this work, we decided to use only data related to sensors that are suitable for use in a real-time monitoring application.

A study was conducted on the sensors available in the market that could be used to obtain each of the parameters present in the columns of the adopted database. Some parameters, such as organic carbon and sulfate, require detailed chemical analysis in a laboratory for their acquisition, as can be seen in (EVOQUA, 2022) and (TABATABAI, 1974). Additionally, no electronic solution was found to obtain the parameter related to trihalomethanes.

Based on the findings from the examination of available market solutions, the following columns were selected to train the machine learning model: **pH value** (can be obtained using the PH4502C¹ sensor, for example); **Hardness** (can be obtained using the WHB-300² sensor, for example); **Solids or Total Dissolved Solids - TDS** (can be obtained using the CDPB-03³ sensor, for example); **Chloramines** (can be obtained using the PC101⁴ sensor, for example); **Conductivity** (can be obtained using the CDPB-03⁵ sensor, for example); **Turbidity** (can be obtained using the CUS52D⁶ sensor, for example).

This project used Python and the Pandas library to handle the data. During the data preprocessing phase, unwanted columns (non-electronic data) were removed, and missing values were addressed by filling nulls with the average value of their respective columns. Out of the dataset, only three columns contained null values: pH, Sulfate, and Trihalomethanes. With Sulfate and Trihalomethanes columns already removed, the process was applied solely to the pH column. The dataset displayed an imbalance in the number of values classified as potable (labeled "1") and non-potable (labeled "0"), with potable instances occurring less frequently. To address this, a random sampling of non-potable data was conducted to equal-

¹<https://thinkrobotics.com/products/ph4502c-ph-meter>

²<https://pt.aliexpress.com/item/32976651193.html>

³<https://www.lutroninstruments.eu/ph-redox-conductivity-oxygen/conductivity-probe-lutron-cdpb-03/>

⁴<https://pt.aliexpress.com/i/1005004242550901.html>

⁵<https://www.lutroninstruments.eu/ph-redox-conductivity-oxygen/conductivity-probe-lutron-cdpb-03/>

⁶<https://www.endress.com/en/field-instruments-overview/liquid-analysis-product-overview/turbidity-drinking-water-sensor-cus52d>

ize the sample sizes and achieve balance. Finally, the data was split into a training set and a test set at a ratio of 72% for training and 28% for testing, with this division being done randomly. After this, normalization was applied to the data using the StandardScaler from the Sci-kit Learn library, which standardizes the data by removing the mean and scaling to unit variance.

Based on the examples of the references that used the same dataset, (KADDOURA, 2022; PRIYADARSHINI *et al.*, 2022; IVANOV; TOLEVA, 2023; ARORA *et al.*, 2022), the following metrics were chosen to assess the performance of the machine learning model: Accuracy, Precision, Recall and F1 Score.

The machine learning model chosen for this research was Random Forest, which is widely used for data classification and also yielded good results according to the researches from (KADDOURA, 2022; PRIYADARSHINI *et al.*, 2022; IVANOV; TOLEVA, 2023; ARORA *et al.*, 2022). The library used to train the model was Sci-kit Learn. To identify the best hyperparameters for the model, the Optuna tool was used during training. The parameters used in the model can be found in Table 3. Following the training phase, the model was validated using the previously separated test dataset. This Random Forest model will be referred to in this work as the **RF Model**.

Table 3: Parameters used for training the RF model using Sci-kit Learn library.

Parameter	Value
max_features	sqrt
n_estimators	130
max_depth	10
min_samples_split	4
min_samples_leaf	4
criterion	log_loss

4.2 Embedding the first Random Forest model

The following steps were taken to answer **RQ2** - “What is the performance outcome when embedding the reference machine learning model in comparison to results found in the

literature?", and **RQ3** - "How different is the performance of the embedded model compared to the cloud model?":

1. The machine learning model was deployed on the board;
2. The performance of the on-device model was evaluated using the test dataset;
3. The same model is deployed on the cloud for use with the board;
4. The performance of the cloud model was evaluated using the test dataset;
5. The results of both machine learning models were compared.

With the machine learning model ready and validated, it was prepared and deployed on an ESP-32 board, specifically the ESP32-S3-DevKitC-1 model manufactured by Espressif, which can be seen in Figure 6. This board was chosen because its specification is suitable for running various machine learning models and requires a low voltage for power.

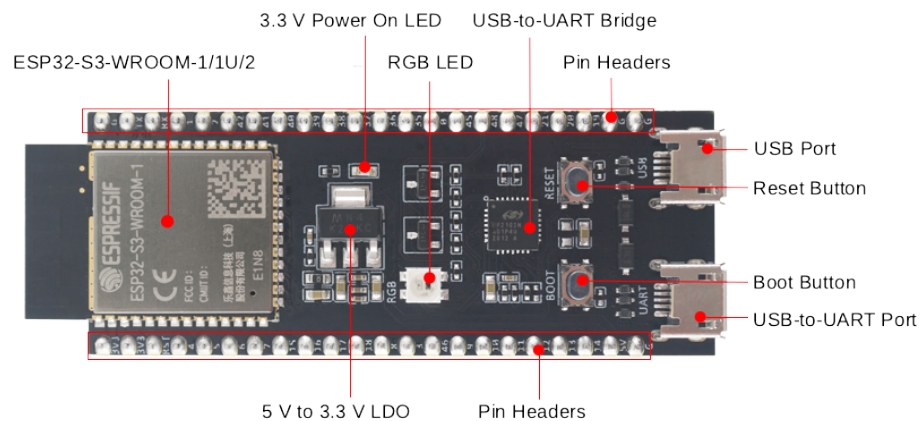


Figure 6: Description of the components of the ESP32 board manufactured by Espressif (Source: (Espressif Systems (Shanghai) Co., 2023)).

To adapt the Random Forest model, the Emlearn library made by (NORDBY *et al.*, 2019) was used, translating the Python model into pure C language code. The Emlearn library serves as an inference engine for machine learning, specifically designed for micro-controllers and embedded devices. It takes a trained machine learning model from Sci-kit Learn as input and outputs a file with the C language code equivalent to the input model.

Subsequently, the C code of the machine learning model was compiled on the board with the assistance of the open source software Integrated Development Environment (IDE) (Arduino, 2024).

After deploying the machine learning model on the board, it was tested considering the same test dataset used to evaluate the model on the computer. The test database, previously segregated from the rest of the database used to train the model in the preceding phase, was converted into six floating-type data arrays, one for each parameter, and integrated into the code compiled on the board. Each inference made by the model used one item from each array, continuing this process until the entire database was processed. Therefore, we did not use data from real sensors, we just simulated the input parameters.

The required time and energy consumption of the machine learning model to run all inferences from the test set was measured, as well as the memory space occupied by the model on the board. The information on the memory space occupied by the model was obtained using the Arduino IDE tool, which reports these data on its console after compiling the code. The execution time was measured with the help of the Arduino's *millis()* function, which starts a millisecond timer during the code execution on the board. The timer was initiated immediately before making the first inference with the model and was stopped immediately after the last inference was executed. The outcome of the timer was displayed on the Arduino IDE console.

Measurement of energy consumption on the ESP-32 board was performed using a Power Profiler Kit II, manufactured by Nordic Semiconductor. The hardware setup used to measure energy consumption can be visualized in Figure 7, and the Power Profiler software is shown in Figure 8. The software provides information on the amount of charge and the average current over a given time window in seconds. Six measurements were taken using the software and the kit, making all inferences from the test dataset for each measurement. The first measurement was discarded because of interference spikes from the process of powering up the board. From the next 5 measurements, an average of the amount of charge used in each execution window of the machine learning model was calculated. By multiplying the amount of charge by the applied voltage on the board, which is 3.3 V, we obtain the energy in Joules.

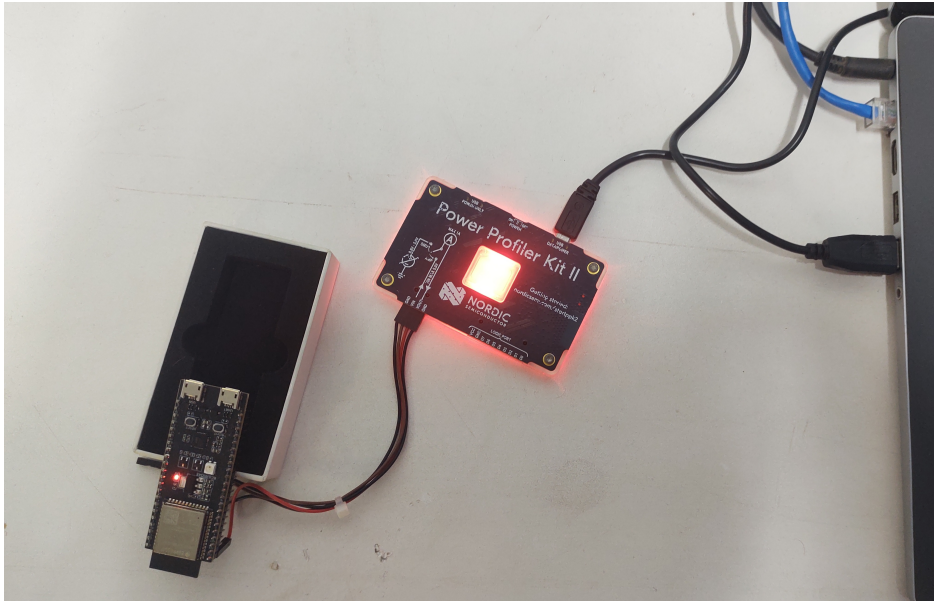


Figure 7: Tools used to measure the power consumption of the ESP-32 board.

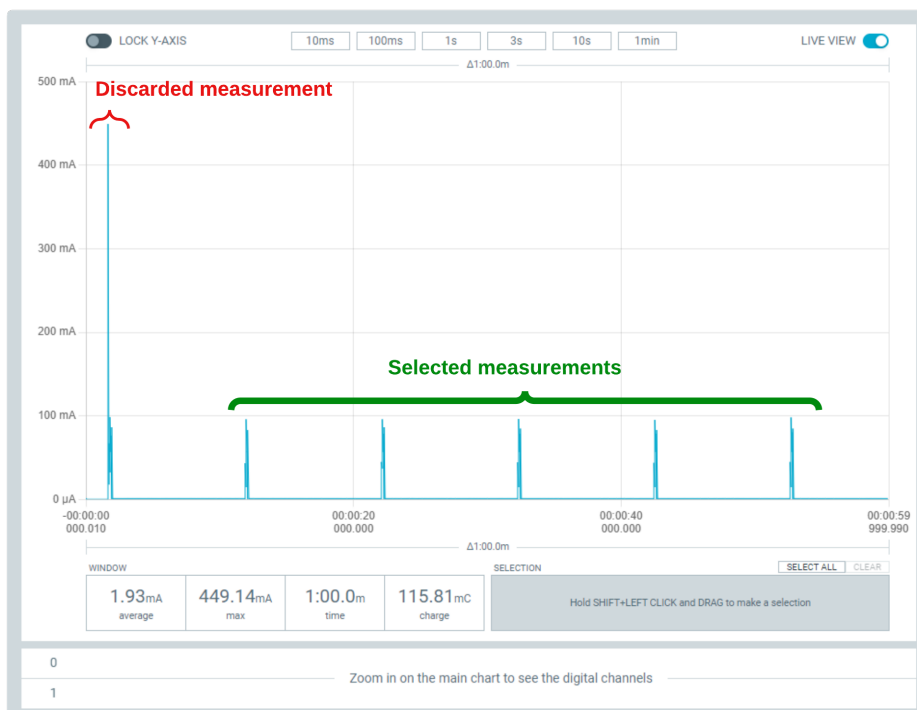


Figure 8: Print screen of the Nordic Power Profile Software used to measure the power consumption of the ESP-32 board, highlighting the measurement that was discarded and the measurements that were selected to calculate the average amount of charge.

The classification model was also deployed in the cloud using the Google Cloud platform. The aim was to compare the model's performance when it is running in the cloud and accessed by the ESP-32 via API with the model's performance when it is running in

the ESP-32. The execution of the inferences was carried out iteratively, repeating the same strategy and using the same test database that was used for the embedded model. Figure 9 presents a diagram that illustrates the architecture used for both the embedded model and the cloud model.

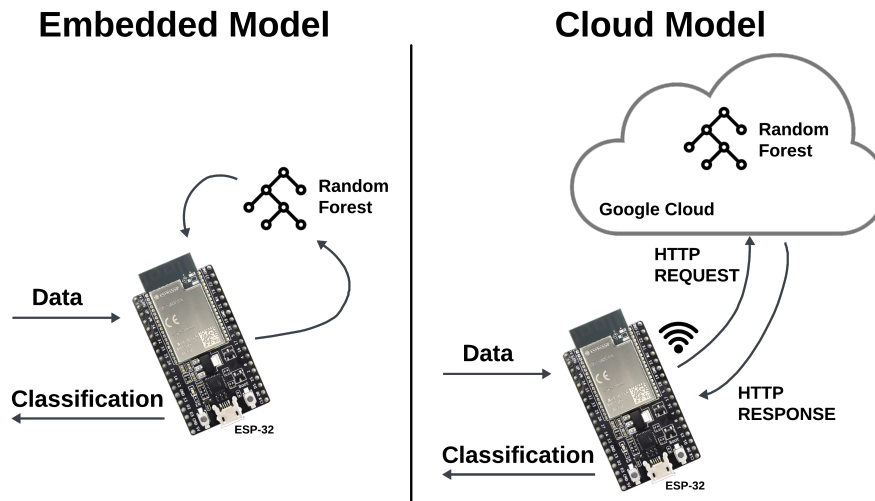


Figure 9: Architectures used in the two machine learning models developed during the study.

4.3 Developing the neural network models

The following steps were taken to answer **RQ4** - “Does an embedded model with a neural network algorithm perform better in this problem than the one with a Random Forest algorithm?”:

1. It was implemented two different machine learning models using neural network algorithms for water potability classification, employing the same training data used with the model that utilizes the Random Forest algorithm;
2. The hyperparameters of these two new models were optimized;
3. The performance of these models was evaluated using the same test dataset that was used with the Random Forest model;
4. The models were deployed on the board used and their performances were compared.

The first neural network model was developed using the Sci-kit Learn library and will be called **MLP Model**. The second neural network model was developed using the

TensorFlow library and its extension for embedded systems (TensorFlow Lite Micro), it will be called **TF Model**. The methodology for selecting and preparing the data from the dataset was thoroughly described for the RF model, in subsection 4.1.

The Optuna framework was employed to determine the optimal hyperparameters for the two machine learning models during their training phase. The parameters used in the MLP Model and TF Model can be found in Table 4 and Table 5, respectively. After training, we validated the models using the test dataset that had been set aside prior to the training phase. We evaluated the performance of the machine learning models considering the following metrics: accuracy, precision, recall, and F1-Score.

Table 4: Parameters used for training the MLP Model.

Parameter	Value
max_iter	200
hidden_layer_sizes	(10, 50)
max_depth	10
activation	relu

Table 5: Parameters used for training the TF Model.

Parameter	Value
n_layers	3
n_units_layer0	80
n_units_layer1	126
n_units_layer2	60
dropout_layer0	0.7004368044240818
dropout_layer1	0.3935802884790261
dropout_layer2	0.40430275636333424
learning_rate	0.00014780473871194358
activation	relu

4.4 Embedding and comparing multiple machine learning models and TinyML frameworks

The following steps were taken to answer **RQ5** - “Which combination of algorithm and TinyML library adaptation performs best under similar conditions for this problem?”:

1. Each combination of machine learning model and selected TinyML library was deployed on the board;
2. The models were run using the test dataset, and the metrics adopted in the study were collected;
3. The results of the obtained metrics were analyzed and compared.

To embed the **RF model**, we considered three libraries: Emlearn (NORDBY *et al.*, 2019), Micromlgen (MICROMLGEN..., 2020), and Everywhereml (EVERYWHEREML..., 2021). To embed the **MLP model**, we considered Emlearn and EmbML (SILVA *et al.*, 2019). To embed the **TF Model**, we considered Emlearn, EloquentTinyML (EloquentTinyML..., 2020), and Everywhereml. Each of these libraries takes a machine learning model that has been pre-trained using Python and its corresponding library (either Sci-kit Learn or TensorFlow, along with TensorFlow Lite Micro), and processes this model by translating it into an equivalent C code file. Then, we compiled the C code on the board using the open-source Arduino Integrated Development Environment (IDE) software (Arduino, 2024).

The selection of libraries was based on their compatibility with the algorithm used by the machine learning model. Some, like EloquentTinyML, are more specific and tailored only for one type of model, in this case neural networks using TensorFlow. Micromlgen, for example, supports some classification and regression models based on Sci-kit Learn but without a type of neural network. Only the Emlearn library supported all the machine learning models adopted in the study.

Following their deployment on the board, each machine learning model underwent testing using the same test dataset that had been prepared earlier. The approach for measuring memory usage, execution time, and energy consumption followed the same methodology described in subsection 4.2.

5 Results and discussion

This chapter details and discusses the findings of the research, organized into three subsections, each corresponding to a different stage of the study. Subsection 5.1 presents the outcome of assessing the machine learning model's ability to classify water potability, utilizing only database columns that correspond to parameters detectable through electronic means. Subsection 5.2 covers the performance results of the Random Forest model when implemented on an embedded device. Finally, Subsection 5.3 provides an analysis of the performance of various machine learning models and libraries on the embedded device to classify water potability.

5.1 Adapting the Random Forest model to work with electronic sensors

Figure 10 depicts the confusion matrix for the Adapted Model and the Embedded Model. The Adapted Model refers to the RF Model that was constructed using only the parameters from the database related to electronic sensors. The Embedded Model, on the other hand, refers to the RF model obtained after adjusting and porting the Adapted Model to operate on the ESP-32 board. A false negative occurs when the model classifies the water as non-potable and it is potable. A false positive occurs when the model classifies the water as potable and it is non-potable. Therefore, in a machine learning model designed to classify water potability, it is desirable to have the least number of false positives possible to minimize the health risks to humans. The Adapted Model returned 135 false positives, while the Embedded Model returned 92. This represents a disadvantage of the Adapted Model from a quality point of view.

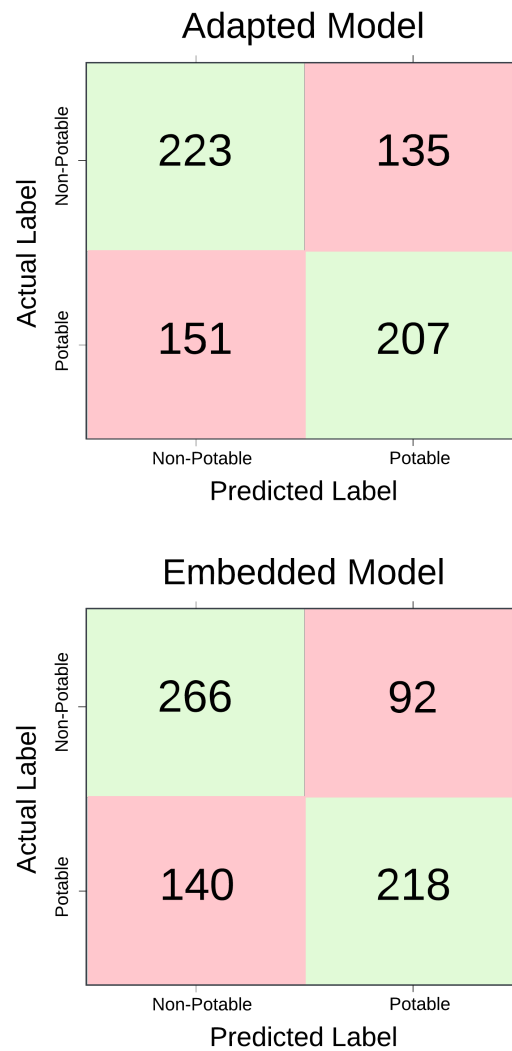


Figure 10: Confusion matrices from the versions of the RF models built in this work.

The metrics obtained from the machine learning model adapted to consider only the electronic detected data, after its validation, can be seen in Table 6. It should be noted that removing columns related to parameters that were not feasible for electronic detection had an impact on the model considering the results of (KADDOURA, 2022; PRIYADARSHINI *et al.*, 2022; IVANOV; TOLEVA, 2023; ARORA *et al.*, 2022) using the same Random Forest algorithm, as can be seen in Table 1. Since the study by (PRIYADARSHINI *et al.*, 2022) does not detail the steps taken for data processing and training of the Random Forest model, we will use only the work of (IVANOV; TOLEVA, 2023) for the comparison of the results, because this study provided a comprehensive description and served as the basis for the replication of the model.

Table 6: Metrics comparing all Random Forest models. The “Adapted Model” is the one reproduced by this work using Ivanov’s model as a basis but considering only electronic sensors, the “Embedded Model” is the adapted model running on the ESP-32 board, and the “Cloud Model” is the adapted model running on Google Cloud and being consumed by the ESP-32 board.

Metric	(IVANOV; TOLEVA, 2023)	Adapted Model	Embedded Model	Cloud Model
Accuracy	0.81	0.60	0.68	0.60
Precision	0.82	0.60	0.70	0.60
Recall	0.81	0.60	0.61	0.60
F1-Score	0.81	0.60	0.65	0.60
Memory occupied by the model (Bytes)	-	-	421,825	864,073
Time to run 716 inferences (ms)	-	-	362	862,549
Energy consumption (J)	-	-	0.112292	237.4578

The Adapted Model achieved a lower value than (IVANOV; TOLEVA, 2023) in all metrics. It was 0.21 lower on Accuracy, Recall and F1-Score, and it was 0.22 lower on Precision. This addresses **RQ1**, which investigated the performance differences of the reference machine learning model from the literature when the number of parameters is reduced.

The Embedded Model achieved a lower value than (IVANOV; TOLEVA, 2023) in all metrics. It was 0.13 lower on Accuracy, 0.12 lower on Precision, 0.20 lower on Recall, and it was 0.16 lower on F1-Score. This addresses **RQ2**, which investigated the performance result of embedding the reference machine learning model from the literature.

5.2 Results of embedding the first Random Forest model

The Table 6 presents the results obtained from the machine learning models running on the chosen board: the embedded and cloud-based models. In this table, we can observe both the evaluation metrics of the model itself and the metrics of the model running on the board. The size of the test dataset is 716 entries, so the time and energy consumption metrics take this number into account.

One can see that in terms of memory space, the Embedded Model used 442,248 Bytes less compared to the Cloud Model. Regarding the time taken to run the 716 inferences on the test dataset, the Embedded Model was faster by 862,187 milliseconds (14 minutes) than the Cloud Model. Furthermore, in terms of energy consumption, the Embedded Model consumed approximately 237.35 Joules less than the Cloud Model.

From Table 6, we can observe a difference in the accuracy, precision, recall, and F1-Score metrics obtained for the embedded model compared to the adapted model and the cloud model. This slight discrepancy in the numbers may be related to differences in floating point operations on the board or in the process of converting the adapted model from Python to C using the Emlearn library. This factor should be taken into account when considering the choice between the two types of models. If this difference is significant, the cloud model has an advantage because it should not exhibit any variance from the results of the adapted model. Another advantage of the cloud model would be the greater ease of change or improvement in the model, as it would only require updating the application once in the cloud and all devices consuming from it would be updated. To achieve the same with the embedded model, it would be necessary to update the firmware of each device to run the new model.

When analyzing the metrics of memory usage, time, and energy consumption, we can observe that the embedded model has a significant advantage over the cloud model. It occupies approximately 51.2% less memory space, runs approximately 99.95% faster for all test inferences, and consumes approximately 99.95% less energy.

Regarding the memory occupied by the machine learning model, despite the fact that the code of the cloud model is simple, dealing only with the board's network connection and handling the sending of requests and receiving of responses from the application, the occupied size was much larger than that of the code with the embedded model. This can be explained by the need to use software libraries that control Wi-Fi hardware and the HTTP protocol.

An explanation for the significantly higher energy consumption of the cloud model is that for it to operate, the board needs to be connected to the network via Wi-Fi, which is extremely costly in terms of energy.

Taking into account the use of the board with a rechargeable Li-Po 482839 battery of 500mAh and 3.7V, in a basic calculation, this battery would provide approximately 6,660 Joules of energy before being completely discharged. With the machine learning models running on a board powered by this battery and processing the validation set once a hour, ignoring the energy consumption of the board in sleep mode, the cloud model would last approximately 1 day and 4 hours. On the other hand, the embedded model would last ap-

proximately 7 years. This calculation only considers the energy consumption of running the machine learning model on the board, but in a real-world scenario, the device would also incur additional costs, such as those associated with the use of sensors and a device to communicate results. Thus, this time difference can be lower, but the magnitude order of the difference will be similar. Moreover, in this context, one possible method to acquire data inside the board could be through Bluetooth Low Energy (BLE), which is a less energy intensive option compared to WiFi (HODDIE; PRADER, 2020).

In the near future, there are plans to build a prototype using real sensors to collect data on parameters and communicate results via Bluetooth Low Energy (BLE) to a central water monitoring system. The results generated by the machine learning model running on the device will then be validated by experts in the field to assess the quality and applicability of the model in a real-world scenario.

These results address **RQ3**, which investigated the performance differences between the Embedded Model and the Cloud Model.

5.3 Comparing results for multiple machine learning models and TinyML frameworks

Table 7 presents the results of the Random Forest models executed on the selected board. This table highlights not only the evaluation metrics of the machine learning models, but also the performance metrics when running on the board. The test dataset consists of 716 entries and both time and energy consumption measurements are based on this quantity. Time is measured as the duration needed to process these 716 entries, and energy consumption is calculated as the average energy required to run the test dataset five times.

Examining the four evaluation metrics of the machine learning model itself, it is observed that the outcomes were identical for Emlearn and Micromlgen. For Everywhereml, there were differences in all metrics, with accuracy 14.7% lower, precision 17.14% lower, recall 4.92% lower, and F1 score 10.77% worse.

Regarding memory usage, the Emlearn model required 20,372 Bytes less than Micromlgen and 47,168 Bytes less than Everywhereml. Taking into account the time, Emlearn and Micromlgen were equally fast, while Everywhereml lagged by 53 milliseconds to exe-

cute the 716 inferences. Regarding energy consumption, Micromlgen used 7.758 millijoules less than Emlearn and 16.63 millijoules less than Everywhereml, indicating a negligible difference in this metric.

Table 7: Metrics comparing all libraries used for the developed machine learning models, highlighting the best results for each metric.

Metric	RF Model			MLP Model		TF Model		
	Emlearn	Micromlgen	Everywhereml	Emlearn	EmbML	Emlearn	EloquentTinyML	Everywhereml
Accuracy	0.68	0.68	0.58	0.59	0.59	0.60	0.60	0.60
Precision	0.70	0.70	0.58	0.59	0.59	0.60	0.60	0.60
Recall	0.61	0.61	0.58	0.59	0.59	0.57	0.57	0.57
F1-Score	0.65	0.65	0.58	0.59	0.59	0.59	0.59	0.59
Memory occupied by the model (Bytes)	421,825	442,197	468,993	285,181	283,133	356,301	472,789	473,177
Time to run 716 inferences (ms)	362	362	415	643	606	4,424	978	3,567
Energy consumption (mJ)	112.292	104.534	121.164	167.199	163.404	1,002.714	246.880	848.063

Table 7 displays the results for each version of the MLP Model. It provides information on both the evaluation metrics of the machine learning models and their operational performance on the board. The test dataset is composed of 716 entries, and the recorded time and energy consumption metrics reflect this total.

Reviewing the four evaluation metrics for the machine learning model, it was found that Emlearn and EmbML produced the same results. In memory consumption, EmbML utilized 2,048 Bytes less than Emlearn. Furthermore, EmbML completed the 716 inferences of the test dataset 37 milliseconds faster and used 3.795 millijoules less energy. This slight variation in energy consumption suggests a minimal and negligible difference in efficiency between the two.

Table 7 presents the performance results for different versions of the TF Model. This table includes information on the machine learning models' evaluation metrics and their operational efficiency on the board. With a test dataset comprising 716 entries, the data on time and energy consumption are calculated based on this figure.

Upon analyzing the four evaluation metrics of the machine learning model, it was observed that the three libraries chosen produced identical results. In terms of memory usage, Emlearn required 116,488 Bytes less than EloquentTinyML and 116,876 Bytes less than Everywhereml. Regarding the execution time for 716 inferences from the test dataset, EloquentTinyML was 2,589 milliseconds faster than Everywhereml and 3,446 milliseconds faster than Emlearn. With respect to energy consumption, EloquentTinyML was more ef-

ficient, consuming 753.834 millijoules less than Emlearn and 599.183 millijoules less than Everywhereml.

In order to answer the **RQ4**, the best outcomes for the evaluation metrics of machine learning models adopted in this work were reviewed. Both the MLP Model and the TF Model were outperformed by the RF Model, which demonstrated superior overall performance. When looking at metrics related to the embedded model's operation on the board, the MLP Model uses slightly less memory, but the RF Model remains faster and more energy-efficient. The decision on which machine learning model to use in future applications should be based on which aspects are more critical for that application, whether they are related to the time taken to perform inferences and energy consumption or to the model's classification performance and memory footprint.

There is no single solution that is best across all metrics. One way to determine the best combination is to see which is fastest and lightest, thereby using less energy to operate, and also which has the best performance in terms of model classification accuracy. The combination of the RF Model and the Emlearn library achieved the best results considering memory footprint and speed for running all inferences. Although the combination of the RF Model and the Micromlgen library takes the same time and consumes roughly the same amount of energy, there is a difference of almost 20 Bytes in memory usage, which should be considered when developing for hardware with limited capacity. The combination of the RF Model with Emlearn and Micromlgen libraries has the best overall results looking at the classification outcomes of the model, with these results being similar among them. This answers **RQ5**, which sought the best combination between the machine learning model and the TinyML library for this problem.

Another notable point is that the Emlearn library is the only library that works with all three types of machine learning models used in this study and has shown good results for each, highlighting its versatility.

6 Conclusion

This study presented a proposal for an energy efficient TinyML model for the water potability classification problem, ensuring that all parameters used for model training could be obtained through electronic sensing. The performance of the obtained model, using the Random Forest machine learning algorithm, was close to a reference in the literature based on metrics such as Accuracy, Precision, Recall, and F1-Score. This is particularly evident compared to other models trained using the same database, but without limitations on the types of water quality parameters.

The findings demonstrated that the embedded RF Model outperforms its cloud-hosted counterpart in various metrics, including memory footprint, inference execution time, and energy consumption. Specifically, it occupies about 51.2% less memory space, executes all test inferences approximately 99.95% faster and consumes approximately 99.95% less energy. This underscores its superior environmental sustainability and energy efficiency. Moreover, it enables a machine learning model to run for 7 years, instead of less than 2 days, in a device very resource-constrained.

As expected, the adaptation of the machine learning model to consider only electronic sensors and the adaptation to run in an embedded system cause a degradation in the quality of the model. In the worst case (RF Model running on cloud), the degradation of the model was 26% (f1-score) compared to the original model proposed in the literature. Thus, the proposed adaptation did not greatly affect the quality of the model.

This study also presented a comparative study of different machine learning models and TinyML adaptation libraries applied to a water potability classification problem. The study evaluated three different algorithms (Random Forest and two types of Neural Networks) and compared their performance across seven different metrics: Accuracy, precision, recall, F1 score, memory occupied by the model, time to run all inferences from the test dataset, and energy consumption to run all inferences from the test dataset. For each model, TinyML adaptation libraries that support the respective algorithm were used.

The study showed that the Random Forest algorithm (RF Model) achieved the best performance metrics (Accuracy, Precision, Recall, and F1-Score). This result was obtained by combining the model with the Emlearn and Micromlgen libraries, both of which also

achieved the best result in terms of the shortest time required to run all inferences on the test dataset, with 362 milliseconds. The MLP Model combined with the EmbML library occupied the least memory space, with 283,113 bytes, which is important for use on memory resource-constrained devices. The RF Model, along with the Micromlgen library, had the lowest energy consumption, at only 104.534 millijoules, demonstrating the highest energy efficiency. The Emlearn library stands out as the only library compatible with all three types of machine learning model used in this study, demonstrating good results for each and highlighting its versatility.

One limitation of the study is that it did not explore other machine learning algorithms, focusing only on neural networks and Random Forest. Another limitation is that it used only one hardware platform for comparison, the ESP-32 board. This study remained focused solely on simulation and did not build a prototype system with real sensors to collect data for feeding and validating the model. In addition, parameters and standards for water potability classification and treatment can vary according to local health authorities, making it challenging to develop a universal water potability classification model.

Future work can explore new combinations of algorithms and TinyML adaptation libraries to compare their performance with those obtained in this study, with the aim of developing even better and more efficient machine learning models. Other hardware platforms can also be considered. Another future objective is to create an application that uses data from real sensors in an experimental environment to assess the impact on the models. Furthermore, future research could involve using a larger dataset containing data from local water treatment stations to build more targeted TinyML models for water potability classification.

References

- ABADADE, Y. *et al.* A comprehensive survey on tinyml. *IEEE Access*, v. 11, p. 96892–96922, 2023.
- ABDULWAHID, A. H. Iot based water quality monitoring system for rural areas. *In: 2020 9th International Conference on Renewable Energy Research and Application (ICRERA)*. [S.l.: s.n.], 2020. p. 279–282.
- AHMED, U. *et al.* Efficient water quality prediction using supervised machine learning. *Water*, v. 11, n. 11, 2019. ISSN 2073-4441. Available at: <https://www.mdpi.com/2073-4441/11/11/2210>.
- ALIPIO, M. I. Data-driven iot-based water quality monitoring and potability classification system in rural areas. *In: 2020 International Conference on Information and Communication Technology Convergence (ICTC)*. [S.l.: s.n.], 2020. p. 634–639.
- ALNAQEB, R. *et al.* Machine learning-based water potability prediction. *In: 2022 IEEE/ACS 19th International Conference on Computer Systems and Applications (AICCSA)*. [S.l.: s.n.], 2022. p. 1–6.
- ALZUBI, A. A. Iot-based automated water pollution treatment using machine learning classifiers. *Environmental Technology*, Taylor & Francis, v. 0, n. 0, p. 1–9, 2022. PMID: 35083949. Available at: <https://doi.org/10.1080/09593330.2022.2034978>.
- ANTONINI, M. *et al.* An adaptable and unsupervised tinyml anomaly detection system for extreme industrial environments. *Sensors (Basel, Switzerland)*, v. 23, 2023.
- Arduino. *Arduino IDE Software Documentation*. 2024. Accessed in February 10, 2024. Available at: <https://docs.arduino.cc/software/ide/>.
- ARORA, A. *et al.* Analyzing the potability of water using machine learning algorithm. *In: 2022 Fifth International Conference on Computational Intelligence and Communication Technologies (CCICT)*. [S.l.: s.n.], 2022. p. 250–256.
- BARTRAM, J.; BALLANCE, R. *Water quality monitoring: a practical guide to the design and implementation of freshwater quality studies and monitoring programmes*. [S.l.: s.n.]: CRC Press, 1996.
- BATARSEH, F. A.; YANG, R. *Data democracy: at the nexus of artificial intelligence, software development, and knowledge engineering*. [S.l.: s.n.]: Academic Press, 2020.
- BRIA, A. *et al.* An iot-ready solution for automated recognition of water contaminants. *Pattern Recognition Letters*, v. 135, p. 188–195, 2020. ISSN 0167-8655. Available at: <https://www.sciencedirect.com/science/article/pii/S0167865520301410>.
- CAPOGROSSO, L. *et al.* A machine learning-oriented survey on tiny machine learning. *IEEE Access*, v. 12, p. 23406–23426, 2024.
- C.ASHWINI *et al.* Water quality monitoring using machine learning and IoT. *International Journal of Scientific & Technology Research*, v. 8, p. 1046–1048, 2019.

DUTTA, D. L.; BHARALI, S. Tinyml meets iot: A comprehensive survey. *Internet of Things*, v. 16, p. 100461, 2021. ISSN 2542-6605. Available at: <https://www.sciencedirect.com/science/article/pii/S2542660521001025>.

EloquentTinyML - Arduino Reference. 2020. Available at: <https://www.arduino.cc/reference/en/libraries/eloquenttinyml/>.

Espressif Systems (Shanghai) Co. *ESP32-S3-DevKitC-1 v1.1 - ESP32-S3 - — ESP-IDF Programming Guide latest documentation*. 2023. Available at: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32s3/hw-reference/esp32s3/user-guide-devkitc-1.html>.

EVERYWHEREML: Train ML in Python, run everywhere. 2021. Available at: <https://github.com/eloquentarduino/everwhereml>.

EVOQUA. *A Guide to Total Organic Carbon Analysis*. 2022. Available at: <https://www.evoqua.com/en-GB/blog/guide-to-total-organic-carbon-analysis/>.

HODDIE, P.; PRADER, L. Bluetooth low energy (ble). *IoT Development for ESP32 and ESP8266 with JavaScript*, 2020.

ISO, B. *et al.* Water quality—sampling—part 1: Guidance on the design of sampling programmes and sampling techniques. *British Standards Institution*, p. 5667–5661, 2006.

IVANOV, I.; TOLEVA, B. Predicting the Water Potability Index Using Machine Learning. *Environment and Ecology Research*, v. 11, n. 4, p. 537–542, ago. 2023. ISSN 2331-625X, 2331-6268. Available at: http://www.hrpub.org/journals/article_info.php?aid=13394.

IZBICKI, R.; SANTOS, T. M. dos. *Aprendizado de máquina: uma abordagem estatística*. [S.l.: s.n.]: Rafael Izbicki, 2020.

JHA, B. Cloud-based smart water quality monitoring system using iot sensors and machine learning. *International Journal of Advanced Trends in Computer Science and Engineering*, v. 9, p. 3403–3409, 06 2020.

KADDOURA, S. Evaluation of machine learning algorithm on drinking water quality for better sustainability. *Sustainability*, v. 14, n. 18, 2022. ISSN 2071-1050. Available at: <https://www.mdpi.com/2071-1050/14/18/11478>.

KADIWAL, A. *Water Quality - Drinking water potability*. 2021. Available at: <https://www.kaggle.com/datasets/adityakadiwal/water-potability>.

KODITALA, N. K.; PANDEY, P. S. Water quality monitoring system using iot and machine learning. In: *2018 International Conference on Research in Intelligent and Computing in Engineering (RICE)*. [S.l.: s.n.], 2018. p. 1–5.

KRUSE, P. Review on water quality sensors. *Journal of Physics D: Applied Physics*, IOP Publishing, v. 51, n. 20, p. 203002, 2018.

LI, P.; WU, J. Drinking water quality and public health. *Exposure and Health*, v. 11, n. 2, p. 73–79, Jun 2019. ISSN 2451-9685. Available at: <https://doi.org/10.1007/s12403-019-00299-8>.

LVOVA, L.; Di Natale, C.; PAOLESSE, R. 7 - chemical sensors for water potability assessment. In: GRUMEZESCU, A. M.; HOLBAN, A. M. (ed.). *Bottled and Packaged Water*. Woodhead Publishing, 2019. p. 177–208. ISBN 978-0-12-815272-0. Available at: <https://www.sciencedirect.com/science/article/pii/B9780128152720000076>.

MANJAKKAL, L. *et al.* Connected sensors, innovative sensor deployment, and intelligent data analysis for online water quality monitoring. *IEEE Internet of Things Journal*, v. 8, n. 18, p. 13805–13824, 2021.

MICROMLGEN: Generate C code for microcontrollers from Python's sklearn classifiers. 2020. Available at: <https://github.com/eloquentarduino/micromlgen>.

MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. *Foundations of Machine Learning*. 2. ed. Cambridge, MA: MIT Press, 2018. (Adaptive Computation and Machine Learning). ISBN 978-0-262-03940-6.

MUKTA, M. *et al.* Iot based smart water quality monitoring system. In: *2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)*. [S.l.: s.n.], 2019. p. 669–673.

NORDBY, J.; COOKE, M.; HORVATH, A. *emlearn: Machine Learning inference engine for Microcontrollers and Embedded Devices*. 2019. Available at: <https://doi.org/10.5281/zenodo.2589394>.

PRIYADARSHINI, I. *et al.* Water pollution reduction for sustainable urban development using machine learning techniques. *Cities*, v. 130, p. 103970, 2022. ISSN 0264-2751. Available at: <https://www.sciencedirect.com/science/article/pii/S0264275122004097>.

PULE, M.; YAHYA, A.; CHUMA, J. Wireless sensor networks: A survey on monitoring water quality. *Journal of Applied Research and Technology*, v. 15, n. 6, p. 562–570, 2017. ISSN 1665-6423. Available at: <https://www.sciencedirect.com/science/article/pii/S1665642317301037>.

RAWAT, P. *et al.* A comprehensive analysis of the effectiveness of machine learning algorithms for predicting water quality. In: *2023 International Conference on Innovative Data Communication Technologies and Application (ICIDCA)*. [S.l.: s.n.], 2023. p. 1108–1114.

SEN, P. C.; HAJRA, M.; GHOSH, M. Supervised classification algorithms in machine learning: A survey and review. In: SPRINGER. *Emerging Technology in Modelling and Graphics: Proceedings of IEM Graph 2018*. [S.l.: s.n.], 2020. p. 99–111.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. *Understanding Machine Learning - From Theory to Algorithms*. [S.l.: s.n.]: Cambridge University Press, 2014. I-XVI, 1-397 p. ISBN 978-1-10-705713-5.

SILVA, L. Tsutsui da; SOUZA, V. M. A.; BATISTA, G. E. A. P. A. Embml tool: Supporting the use of supervised learning algorithms in low-cost embedded systems. In: *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*. [S.l.: s.n.], 2019. p. 1633–1637.

STANDARDIZATION, I. O. for. *Water Quality-Sampling-Part 2: Guidance on Sampling Techniques*. [S.l.: s.n.]: International Organization for Standardization, 1991.

TABATABAI, M. A. A rapid method for determination of sulfate in water samples. *Environmental Letters*, Taylor & Francis, v. 7, n. 3, p. 237–243, 1974.

TSOUKAS, V. *et al.* A review of machine learning and tinyml in healthcare. *Proceedings of the 25th Pan-Hellenic Conference on Informatics*, 2021.

WHO - World Health Organization. *Drinking-water*. 2024. Available in: <https://www.who.int/news-room/fact-sheets/detail/drinking-water>. Accessed in January 17, 2024. Available at: <https://www.who.int/news-room/fact-sheets/detail/drinking-water>.

ZHU, M. *et al.* A review of the application of machine learning in water quality evaluation. *Eco-Environment & Health*, v. 1, n. 2, p. 107–116, 2022. ISSN 2772-9850. Available at: <https://www.sciencedirect.com/science/article/pii/S2772985022000163>.

Appendix A Publications generated from this work

This appendix lists the publications that originated from this work. The results obtained by deploying the machine learning model, derived from the literature reference and using only parameters obtainable through electronic sensors, and comparing it with its cloud counterpart, were published in an article titled “An energy efficient TinyML model for a water potability classification problem”⁷ in the journal *Sustainable Computing: Informatics and Systems*⁸, which holds a Qualis A2 rating according to the data from the Coordination for the Improvement of Higher Education Personnel (CAPES)⁹.

Additionally, the results from comparing different combinations of machine learning algorithms and TinyML libraries, and testing them on the ESP-32 board during this research, were submitted as an article titled “Comparing TinyML models and libraries for on-device water potability classification” for the 2024 edition of the conference *Symposium on Computing Systems Engineering (SBESC)*¹⁰, which holds a Qualis A4 rating according to CAPES data.

⁷<https://doi.org/10.1016/j.suscom.2024.101010>

⁸<https://www.sciencedirect.com/journal/sustainable-computing-informatics-and-systems>

⁹<https://sucupira.capes.gov.br/sucupira/public/consultas/coleta/veiculoPublicacaoQualis/listaConsultaGeralPeriodicos.jsf>

¹⁰<https://sbesc.lisha.ufsc.br/sbesc2024/Home>