The Journal of Supercomputing (2024) 80:20080–20117
https://doi.org/10.1007/s11227-024-06166-x

A TabPFN-based intrusion detection system for the industrial internet of things

Sergio Ruiz-Villafranca[1] · José Roldán-Gómez[2] · Juan Manuel Castelo Gómez[3] · Javier Carrillo-Mondéjar[4] · José Luis Martinez[1]

Accepted: 26 April 2024 / Published online: 30 May 2024

Abstract

The industrial internet of things (IIoT) has undergone rapid growth in recent years, which has resulted in an increase in the number of threats targeting both IIoT devices and their connecting technologies. However, deploying tools to counter these threats involves tackling inherent limitations, such as limited processing power, memory, and network bandwidth. As a result, traditional solutions, such as the ones used for desktop computers or servers, cannot be applied directly in the IIoT, and the development of new technologies is essential to overcome this issue. One approach that has shown potential for this new paradigm is the implementation of intrusion detection system (IDS) that rely on machine learning (ML) techniques. These IDSs can be deployed in the industrial control system or even at the edge layer of the IIoT topology. However, one of their drawbacks is that, depending on the factory's specifications, it can be quite challenging to locate sufficient traffic data to train these models. In order to address this problem, this study introduces a novel IDS based on the TabPFN model, which can operate on small datasets of IIoT traffic and protocols, as not in general much traffic is generated in this environment. To assess its efficacy, it is compared against other ML algorithms, such as random forest, XGBoost, and LightGBM, by evaluating each method with different training set sizes and varying numbers of classes to classify. Overall, TabPFN produced the most promising outcomes, with a 10–20% differentiation in each metric. The best performance was observed when working with 1000 training set samples, obtaining an F1 score of 81% for 6-class classification and 72% for 10-class classification.

Keywords Industrial internet of things · Machine learning · TabPFN · Cybersecurity

Extended author information available on the last page of the article

1 Introduction

The industrial environment has undergone significant changes since the first industrial revolution, with the latest development being called Industry 4.0 or the industrial internet of things (IIoT). New technologies and implementations such as the internet of things (IoT), artificial intelligence (AI), and 5G networks are converging with traditional operational technology (OT) protocols and their devices, with the aim of improving the performance and efficiency of companies [1]. Some authors [2–4] are even proposing the start of a discussion of Industry 5.0.

The convergence of OT and information technology (IT) protocols, as well as the IoT and web applications, can lead to security risks if good implementation practices are not followed, with the risks being greater for critical infrastructure, because, if compromised, this can cause significant losses for businesses and governments. In addition, many of the devices found in such infrastructures are unlikely to be updated on a regular basis [5]. As a result, they become prime targets for attackers and their strategies, as there is ample room for attack techniques to change and evolve rapidly. Cybersecurity researchers can utilize ML algorithms to train multiple tools by analysing data left by attackers on their network or applications, which is a method of monitoring and preventing (or mitigating) the impacts of these attacks [6]. However, depending on the plant's or industry's characteristics, the researchers may gather insufficient data from these situations, and the challenges they face in generating datasets to train their models include issues of confidentiality and the difficulty of replicating the attacks in a given scenario [7]. New approaches using ML techniques, such as prior-data fitted network (PFN), have been proposed to address these challenges [8]. These models can be trained with a small amount of data and achieve good performance, making them suitable for use in industrial scenarios with specific data security requirements and device vulnerabilities.

This paper presents an intrusion detection system based on the TabPFN model, in order to address these issues. The aim of this system is to serve as an architecture for the monitoring and detection of anomalies in IIoT environments using a small number of samples for its training phase. Our proposal is compared with other state-of-the-art ML algorithms that return good results with tabular data. To conduct this comparison, several small datasets of 2500 samples were processed and used for training different ML models. The purpose was to evaluate whether our TabPFN proposal achieves better results than the other models. To address the cyber-attack problems that arise in IIoT environments, the proposed architecture can be deployed in an edge layer connected to the industrial network or be integrated into the intrusion detection approach within the IIoT layer, using the industrial control system to implement its functionalities. Given these aspects, this work makes the following contributions:

- *We present an IDS architecture for the detection of intrusions in the IIoT* The architecture is designed with specific requirements to allow the IDS to operate efficiently in various IIoT environments, particularly in new industrial topologies.

- *We design small IIoT datasets that can be used for the training and testing of ML models* In real industry settings, it is difficult to acquire large numbers of real-world samples needed for training ML models [9]. To address this challenge, the scientific community offers researchers public datasets generated using IIoT topologies that contain a substantial number of samples. To evaluate the efficacy of our proposal using fewer samples during the training phase, it is crucial to use these public datasets and create smaller datasets from them.
- *We carry out an evaluation of the performance of our proposal compared with other state-of-the-art ML algorithms* We compare our TabPFN proposal with other ML techniques to confirm that it offers better performance than other techniques. By using multiple metrics for model validation, we can determine which ML implementation performs best in classifying malicious or benign packets.

The rest of this paper is structured as follows. The technical background of our study is presented in Sect. 2. Section 3 outlines the research community's proposals for IDS designed for IIoT environments. The IDS architecture proposed in this experiment, together with the relevant modules and deployment considerations, is presented in Sect. 4. In Sect. 5, the process of creating small datasets to train and test the models considered in the study is explained. Furthermore, this paper provides a detailed methodology for implementing the IDS using ML techniques for the aforementioned datasets. Section 6 evaluates the performance of all the models in each of the studied scenarios. Finally, Sect. 7 presents the conclusions that can be drawn from this experiment.

2 Technical background

Machine learning has grown rapidly in recent years, especially in computing and data analysis, encompassing technologies such as the IoT, and making it possible to develop intelligent applications with innovative features by leveraging insights obtained from data analysis [10]. This approach empowers systems to extract valuable knowledge from incoming data autonomously, eliminating the need for explicit programming [11].

As mentioned above, we adopt a machine learning approach as it is considered the best in environments where there are unknown threats, especially in highly changeable and heterogeneous ones where manually establishing rules for each type of attack may be difficult. Several studies show that machine learning-based approaches perform better in these contexts [12, 13]. In addition to obtain better performance, a machine learning-based IDS also offers other advantages, such as the following:

- *Accuracy* Machine learning algorithms are capable of analysing vast amounts of data and detecting patterns and anomalies that might be difficult for humans to identify or articulate explicitly.
- *Flexibility* Machine learning-based IDS can continuously learn and adapt to new types of attacks and changing network environments.

- *Anomaly detection* Modelling standard system behaviour enables the identification of anomalies that rule-based systems cannot detect.

In machine learning-related research [14], cutting-edge algorithms such as neural networks and their various architectures have received considerable attention. However, especially in the context of IIoT environments, where resource constraints are prevalent, they have specific limitations when applied to classification problems involving tabular data. The following are some of the drawbacks associated with the use of neural networks in this specific domain [15]:

- *Overfitting* DL algorithms, especially neural networks, can be prone to overfitting, particularly when trained on smaller datasets. Although they may perform well on the training data, their capacity to generalize to new, unseen data can be significantly reduced.
- *Computational complexity* DL algorithms are complex models that require significant computational resources for training. This can pose challenges in their implementation and execution, especially when dealing with large datasets.
- *Time-consuming training phase* The training phase of DL algorithms can be time-consuming, particularly when dealing with large datasets. This can pose practical challenges in real-time applications where prompt predictions are crucial.
- *Data requirements* DL algorithms often require a large amount of data to achieve optimal performance. This can be a disadvantage in scenarios where data are scarce or difficult to collect.
- *Data domain* DL algorithms often try to take advantage of the spatial and temporal relationships within the data. However, in the domain of tabular data, which often lacks such relationships, this is typically not relevant. Instead, there are often more direct, causal relationships in tabular data [16, 17].

When it comes to classifying tabular data, neural networks can be powerful, but they can also introduce complexities and time-consuming aspects. Although neural networks are excellent in some scenarios, they may not be universally optimal. For specific use cases, alternative machine learning algorithms, such as random forest or XGBoost, may prove more suitable [15, 18–20]. Therefore, we have decided against evaluating our proposal using neural network models, such as convolutional neural network (CNN) or long short-term memory (LSTM). This section describes some technical concepts used in this work.

2.1 IDS in IIoT environments

IDS is essential tools in the IIoT cybersecurity landscape. Their purpose is to detect and identify intrusion attacks by monitoring network traffic within IIoT environments. IDS plays a vital role in maintaining the integrity of IIoT networks. The design and implementation of IDS in IIoT often utilize machine learning techniques for enhanced cybersecurity [21].

In the realm of IIoT, IDS can be broadly classified into signature-based intrusion detection systems (SIDS) and anomaly-based intrusion detection systems (AIDS). The selection and configuration of an IDS depend on various factors, such as the IDS's quality profile, the cost structure of the firm employing the IDS, and the strategic behaviour of hackers [22]. These systems are typically lightweight and intended for use in ad hoc IIoT networks. The open and scattered structure of IIoT services makes them an attractive target for potential cyber-attacks [23].

IDS in IIoT environments operates under specific technical assumptions and within particular environments. Their purpose is to detect computer security violations, such as illegal entry by outsiders and abuse of privileges by insiders. In the context of IIoT, critical considerations for an IDS include effectiveness, efficiency, ease of use, security, interoperability, transparency, and collaboration [24].

IIoT platforms address various use cases and revolve around similar business objectives, resulting in considerable differences in their architectural set-up. This is partly due to the technical complexity of business-to-business (B2B) environments and the lack of established standards in the IIoT. The gateway within the secure execution domain handles sensitive procedures of the IIoT, such as device admission, bootstrapping, key management, authentication, and data exchange between OT and IT [25].

Intruder detection across IIoT networks has been shown to be efficient using ML techniques. These techniques are adopted in IDSs to improve their accuracy in detecting intruders, reduce false positives, and identify new threats. The use of ML methods in the complex architecture of an IDS in the IIoT involves relying on various ML algorithms and methods, both for binary and multi-classification approaches [26]. It is crucial to carefully select datasets to ensure the suitability of the model construction for IDS usage in IIoT. An ML-enabled IDS based on IIoT is a crucial security method that can help defend computer networks and the IIoT environment against various attacks and malicious activities [27].

2.2 Tree-based algorithms

To classify and predict the type of packet received from the IIoT network, previous studies [19, 20] recommend the application of machine learning techniques such as boosted trees algorithms for classification of tabular data. Our study aims to analyse the efficiency of selected models when there are few training samples in the IIoT environment.

Ensemble learning methods use multiple machine learning models that generate weakly predictive results based on data features extracted from various projections. The results are subsequently combined with various voting mechanisms to achieve higher performance than any individual algorithm alone. The algorithms considered in this study include:

- *Random forest* [28]: The algorithm generates a parallel ensemble by concurrently fitting multiple decision trees. The trees are trained using either distinct

datasets or subsamples of the same dataset. The algorithm generates either the best solution from the trees or the average result as an output.

- *XGBoost algorithm* [29]: This technique is a boosting algorithm that has been proved to enhance weak learners. It is effective in solving both classification and regression problems. In XGBoost, trees can create a new one by considering the previous prediction value of the given input data for the tree and then maximizing the gain in prediction. In order to overcome the limitations of the gradient tree boosting implementation, and to achieve superior performance and computational speed, the XGBoost algorithm has been extended. The extended algorithm is currently being developed as an open source software package.
- *LightGBM algorithm* [30]: This algorithm is a gradient tree boosting algorithm implementation that is optimized for faster training and efficiency. Moreover, LightGBM is suitable for being running on low-resource devices as it consumes less memory compared with other implementations. It also provides better accuracy and can handle considerable data volumes. To achieve this performance, LightGBM uses two main techniques to improve its training speed: (1) gradient-based one-side sampling (GOSS), which selects data samples with higher gradients, thus increasing the focus on the computational gain of information; and (2) exclusive feature bundling (EFB), which is the bundling of some features of the data, thus reducing the dimensionality of the data [31].

2.3 TabPFN

TabPFN is a PFN [32] model developed by Hollman et al. [33]. It is used for performing supervised classification tasks on tabular data. This model uses a transformer that has been trained to perform Bayesian inference. To obtain such a transformer, structural causal models are designed, which generates data depicting causal relationships. This data is then used for pretraining the PFN. Additionally, data containing simple causal relationships is preferred. This choice is based on the principle of Ockham's razor [34], as it can be observed on multiple occasions that causal relationships with fewer parameters tend to have higher a posteriori probabilities. Pretraining the transformer with synthetic data allows it to approximate a large number of new causal relationships based on the simple relationships it was previously trained on. As a result, the transformer can approximate Bayesian inference for a wide range of causal relationships. This approach is unique when compared with other algorithms that currently comprise the state of the art.

Despite the usually high cost of computing integrals involved in Bayesian inference, this approach enables the model to be trained with actual data in less than a second if a GPU is available. Moreover, it does not require hyperparameter optimization.

However, TabPFN does have several limitations. In particular, this model is confined to 1000 training samples, 10 categories, and 100 features. These limitations result from the training time of TabPFN scaling quadratically in relation to the volume of training data.

Nevertheless, despite its limitations, TabPFN can achieve very good classification results, comparable to other state-of-the-art algorithms, when the requirements are met. The results of the TabPFN model are particularly promising in situations in which all features are numerical, there are no missing values, and obtaining labelled datasets with enough samples to train other state-of-the-art algorithms is challenging.

2.4 Evaluation model metrics

A variety of metrics are employed to thoroughly evaluate the effectiveness of the model. These metrics use the information found in the confusion matrix, which is generated from the model's classification results for each packet. This matrix includes essential factors, such as true positive (TP), true negative (TN), false negative (FN), and false positive (FP), which indicate the model's classification performance. The meaning of each factor of the confusion matrix in this kind of classification problem is:

- *True positives (TP)* In this scenario, TP signifies the instances where the model accurately detects packets as belonging to a particular attack type or benign classification.
- *True negatives (TN)* TN represents the number of instances correctly classified as not belonging to a specific attack or category. For each attack or benign category, TN shows the accurate identification of non-membership in that class.
- *False positives (FP)* When dealing with specific attacks or benign categories, false positives (FP) refer to cases where the model mistakenly categorizes benign traffic as a particular attack type or vice versa. It is essential to minimize FP to avoid false alarms for specific attacks or benign classifications.
- *False negatives (FN)* In this scenario, FN refers to situations where packets of a specific attack type are wrongly labelled as non-attack or benign traffic. Efficient reduction of FN is vital to enhance the detection of individual attacks or categories.

The selection of suitable metrics is crucial for assessing various aspects of the model's performance, particularly when the dataset utilized is imbalanced. For that reason, we selected the following metrics [35]:

Accuracy It is a crucial metric that calculates the proportion of accurate predictions to the total predictions created by the model. This measure is computed using the formula displayed in Eq. 1. Nevertheless, in cases where the dataset demonstrates imbalances amongst various categories, it is necessary to interpret accuracy cautiously. In such cases, a below-average model could predict the majority class repeatedly and still appear to have achieved a favourable accuracy owing to the considerable prevalence of that category.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FN} + \text{FP}} \tag{1}$$

Considering the typical imbalanced nature of traffic data, relying only on accuracy for evaluation might be insufficient. It is essential to employ supplementary metrics that provide a more comprehensive evaluation of the model's performance.

Precision This metric indicates the ratio of correct positive predictions to the total number of positive predictions identified by the model. Equation 2 describes its calculation. A noteworthy observation is that a higher value of this metric indicates that the IDS is able to avoid TP classifications. This result highlights the efficiency of the detector in reducing the number of legitimate packets being mistakenly categorized as malicious.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

Recall This metric represents the ratio of accurately detected positive instances to the total number detected as positive by the model. Its calculation is defined by Eq. 3. A higher recall score indicates the proficiency of the intelligent threat detector in identifying a significant number of attacks that accurately match the category it aims to detect. In essence, a high recall score emphasizes the detector's capability to efficiently identify a substantial number of attacks relevant to the specific threat category it aims to detect.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

F1 Score The F1 score metric, frequently identified as the harmonic mean of precision and recall, is crucial in evaluating models trained on imbalanced datasets. It particularly highlights models wrestling with skewed class distributions, as it emphasizes both the accurate prediction ratio and the successful detection of anomalous classes during classification. The formula for this metric is given by Eq. 4, which encapsulates the comprehensive assessment it offers for these models. The F1 score provides a dependable measure which presents a fair view of a model's capacity to uphold accuracy while efficiently identifying significant occurrences, particularly in situations where imbalanced data dominates.

$$\text{F1\_Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

Training–test time This metric measures the time, in seconds, required by each model to complete both the training and testing phases. It indicates the algorithmic

complexity of each analytical technique and provides insight regarding the impact of this complexity on performance when applied to the dataset.

3 Related work

In this section, we present a review of the major studies on intrusion detection systems using machine learning and deep learning techniques in IIoT environments. The scope of this review is limited to pieces of research that are similar to ours, which have the most relevance in this area of study.

The authors in [36] introduce an intrusion detection system (IDS) that uses a convolutional neural network (CNN) to identify distributed denial of service (DDoS) attacks in heterogeneous internet of things (HeIoT) networks. The model is deployed at the gateways of these networks. The CNN model's architecture is multi-layered, comprising of two 1D-convolution layers, two 1D-max-pooling layers, and a fully connected dense layer that serves as the output layer. The CiCDDoS2019 dataset [37] was used for the model's training and testing. The proposed IDS's performance was evaluated under three classification scenarios: binary, 8-class, and 13-class. The evaluation metrics used were precision, recall, F1 score, and accuracy. The model achieved high accuracies of 99.7%, 99.95%, and 99.99% for binary, 8-class, and 13-class classification, respectively.

This work highlights the challenges addressed by their solution in HeIoT environments, which are characterized by the complexity and heterogeneity of protocols and devices. It is important to note, however, that the proposed system is primarily designed to counter DDoS attacks. Although DDoS attacks are a significant threat that requires attention, it is important to note that there are other types that can disrupt the normal functionality of devices and their associated applications in HeIoT scenarios. Additionally, due to the limited computational resources inherent in HeIoT environments, the use of a CNN model may not be the most cost-effective solution. Future research in this area should consider this aspect.

In the innovative study made in [38], a novel method for IDS in IIoT networks is presented. The proposed model uses machine learning and optimization techniques to classify malicious activities. The authors analyse the model using various classifiers and optimization algorithms, including random forest (RF), K-nearest neighbour (KNN), and multi-layer perceptron (MLP) models. Particle swarm optimization (PSO) and the bat algorithm (BA) are considered for optimizing these classifiers. The proposed IDS architecture is a two-step system. First, to train the classifiers, the data is processed using a feature selection technique to extract the most relevant features from the datasets. In the second step, the reduced data obtained from the optimization process is used for the implementation.

Although the authors' proposal presents an interesting solution to the challenges of classifying IIoT traffic, it does not offer any specific techniques to reduce the dimensionality of the received packets. Additionally, the paper lacks detailed information on implementing the proposed solution in real-world IIoT scenarios. Therefore, further research and development in this area are necessary.

The results of the study, using the WUSTL-IIoT-2021 [39] dataset and different implementations of the IDS architecture, show that the use of RF as a classifier with BA for feature selection achieves a performance of 99.6% in the F1 score metric.

In [40], the authors present an innovative study that combines various ensemble machine learning techniques with a Chi-square statistical method for feature selection. This combination is used to detect intrusions and attacks in industrial internet of things (IIoT) networks. The study considers several models, including XGBoost, bagging, RF, extremely randomized trees (ET), and adaptive boosting (AdaBoost). The authors evaluate the performance of each model using seven datasets from ToN-IoT [41]. These datasets are derived from the telemetry data of various IIoT and IoT networks. The results reveal that XGBoost outperforms the other models across all datasets, achieving an impressive F1 score of 98.30%.

The proposed method addresses some of the limitations found in other IIoT intrusion detection techniques. It applies an ensemble-based algorithm for traffic classification in IIoT scenarios and reduces feature dimensionality using a statistical method for feature selection. However, this approach has its own limitations. It is a two-layer IDS, and the authors do not provide details on how it could be deployed in a real-world scenario. Furthermore, due to resource constraints inherent to the IIoT environment, it may not be feasible to deploy in every scenario.

The authors should compare the Chi-square method used for feature selection with other proposals and methods to ensure its optimality in selecting the most relevant features for the algorithms. Lastly, considering the characteristics of the ToN-IoT dataset, it would be beneficial to include another public dataset in the experimentation.

The research presented in [42] explores the creation of an IDS using an RF model. To manage feature selection and reduce the dimensionality of the datasets used for model training and evaluation, the approach incorporates Pearson's correlation coefficient (PCC) and isolation forest (IF). These techniques enable the model to overcome the challenges posed by unbalanced datasets. This is a common problem when deploying an IDS based on ML techniques in an IIoT environment.

The study evaluates the performance of each model individually and in combination, using the Bot-IoT [43] and WUSTL-IIOT-2021 [39] datasets. The results indicate that the proposed approach, which combines the three models under consideration, outperforms other baseline models and combinations, achieving an overall F1 score of 93.57%.

Although the proposal to reduce dataset dimensionality is innovative and can enhance performance while minimizing computational requirements for predictions, the study lacks a clear architectural framework for deploying this IDS. It only detects whether a network packet is an attack or normal traffic. Implementing this solution in a real-world system may pose significant challenges due to the difficulty in determining the appropriate response to mitigate potential attack risks. Additionally, the model may not have enough information to identify new applications and devices deployed as potential attackers attempting to inject malicious applications. In the constantly evolving cybersecurity threat landscape, this could be a critical shortcoming.

The authors of [44] focus their investigation on evaluating the effectiveness of the XGBoost algorithm as an IDS in IIoT/IoT networks. Their aim is to identify and classify specific malicious activities in these inherently imbalanced environments. To assess the performance of the XGBoost model, they use the ToN-IoT [41] and X-IIoTDS [45] datasets and metrics such as F1 score, precision, recall, and accuracy. The datasets contain malicious traffic from various cyber-attacks that can occur in these networks. The proposed IDS design aligns with the traditional architecture of IIoT networks, and the design of the datasets takes into account the imbalanced nature of this paradigm. The evaluation of the XGBoost algorithm using these two datasets shows a high level of performance, with F1 scores of 99.87% for the ToN-IoT dataset and 99.90% for the X-IIoTDS dataset.

Although the authors' proposed IDS is tailored to the IIoT network scheme, applying XGBoost in this context is not straightforward due to the computational requirements needed to run this model correctly. The authors do not provide specific details about where the IDS will be deployed. During the experimental phase, the preprocessing of the data only follows the minimum steps required for the model to work, and there are no additional processes to extract more information or to filter the features of the dataset in order to optimize it for the training phase. This approach could potentially limit the model's ability to adapt to new threats and evolving attack vectors in the dynamic landscape of IIoT security.

The authors in [46] describe the implementation of an IDS for IIoT using convolutional neural network (CNN) deep learning approaches. The study includes a CNN model, long short-term memory (LSTM), and a novel hybrid approach that combines CNN and LSTM to create an enhanced classifier model. The proposed architecture uses a CNN model with two convolutional layers and two max-pooling ones. The model's output is processed by the LSTM model to classify the packet as normal or malicious. This hybrid approach captures spatial features before identifying temporal dependencies in the data.

The performance of the proposed architecture is evaluated using the UNSW-NB15 [47] and X-IIoTID [45] datasets. Considering both binary and multi-class classifications, the study compares the performance of the proposed model with that of the baseline CNN and LSTM models. The results show that the latter outperform the other models considered. They achieve an overall accuracy of 93.06% and 93.57% for UNSW-NB15 and X-IIoTID, respectively.

Although this approach provides an interesting analysis and understanding of data features, it demands a significant amount of computational resources for deployment, which may not be available in traditional IIoT scenarios. Furthermore, the proposal does not present any architecture for deploying the IDS. Additionally, due to the computational requirements, the scalability of this solution for larger IIoT networks cannot be guaranteed. This limitation could potentially affect the applicability of the architecture in real-world scenarios where network size and resource allocation are critical factors.

The paper in [48] presents a novel architecture for an IDS based on multi-access edge computing (MEC) and IIoT environments. The objective is to take advantage of MEC's benefits, particularly in network management and computational resources. Additionally, it includes a comparative study that evaluates the performance and

computational cost of various boosting tree algorithms. The authors assess five models, with XGBoost, LightGBM, AdaBoost, CatBoost, and GradientBoost being evaluated using a custom dataset and a testbed that incorporates traditional industrial protocols such as Modbus/TCP, OPC Unified Architecture (OPC UA), and S7 communications (S7COMM).

The models were tested for their ability to detect various attacks, including scanning techniques, DDoS attacks, packet manipulation, and web application attacks. The study's findings show that the XGBoost classifier achieves the best performance when multiple attacks occur simultaneously, with an F1 score of 99%. On the other hand, the LightGBM classifier demonstrates superior performance in terms of the classification ratio and computational cost.

The study focuses on deploying the model in devices or in the MEC layer, specifically in an IIoT-MEC environment. The generalizability of the findings may be limited by this focus. This would provide insights into how the proposed models perform in different environments and scenarios. To enhance the robustness of the IDS and its applicability across a broader range of IIoT contexts, it would be beneficial to extend the analysis to include other datasets.

In conclusion, our proposal presents a new intelligent IDS based on TabPFN. The system is designed to train models with a smaller sample size and is tailored to address challenges in IIoT environments where access to traffic data is limited due to privacy or confidentiality concerns. This scope has not been extensively explored in the related work reviewed in this section. Additionally, our IDS is designed for deployment in any traditional IIoT environment, providing both flexibility and adaptability.

Moreover, we conducted a thorough analysis and comparison of our approach with other state-of-the-art ML algorithms known for their effective performance with tabular data. It is important to note that the models developed in other studies often require a large number of samples to achieve satisfactory performance results. In contrast, our approach aims to achieve high performance with a smaller sample size, making it a promising alternative for scenarios where data availability is limited. This aspect sets our proposal apart from existing solutions and enhances its potential for practical implementation in real-world IIoT environments.

Table 1 summarizes the algorithms, datasets, and number the samples used in training and testing the models, together with the key highlights and limitations of each study discussed in this section.

4 Proposed IDS architecture

To tackle the issue of having limited data for training ML models and detecting attacks in environments with imbalanced traffic between normal and anomalous packets, this section presents a IIoT architecture. This architecture includes an ML algorithm that was trained with a limited dataset.

The architecture proposed, as shown in Fig. 1, consists of three layers that encapsulate the main components found in IIoT scenarios, these being the IIoT physical layer, the network layer, and the application layer [49].

Table 1 Related work corresponding to proposals from other researchers

References	Models	Dataset	No. samples	Performance	Key highlight	Limitation
[36]	CNN	CICDDos2019	+ 12.5 million	Accuracy 99.99% for 13 classes	The proposal takes into account the complexity of the HeIoT network in its implementation	It supports only the detection of DDoS attacks
[38]	RF, KNN, MLP with PSO or BA for feature selection	WUSTL-IIOT-2021	+ 1.1 million	F1 score for RF-BA: 99.6%	An optimization technique is included to enhance the performance of the proposal	Dimensionality reduction techniques are not included proposal
[40]	Bagging, XGBoost, RF, ET, and AdaBoost	ToN-IoT	+ 400 thousand	F1 score for XGBoost: 98.3%	Implement a new Chi-square technique for feature selection	Due to computational requirements, implementing the proposal in an IIoT network is not trivial
[42]	RF with IF and PCC	Bot-IoT, WUSTL-IIOT-2021	Bot-IoT: + 72 million WUSTL-IIOT: + 1.1 million	F1 score: 93.57%	The proposal includes an automated solution to enhance the preprocessing stage of the data	The proposal is limited to making a binary classification of whether the packet is normal or an attack
[44]	XGBoost	X-IIoTDS and ToN-IoT	ToN-IoT: + 400 thousand X-IIoTDS: + 820 thousand	F1 score X-IIoTDS dataset: 99.87% F1 score ToN-IoT datastet: 99.90%	The design of the IDS is aimed at seamless integration into traditional IIoT environments	The proposal includes a simple preprocessing stage, but it lacks details regarding the model's deployment
[46]	CNN, LSTM, and CNN-LSTM	UNSW-NB15 and X-IIoTDS	UNSW-NB15: + 1 million X-IIoTDS: + 820 thousand	Accuracy UNSW-NB15: 93.06% Accuracy X-IIoTID: 99.82%	A novel hybrid proposal aims to leverage the spatial and temporal features and dependencies inherent in the data	The computational requirements of the proposal could not be accommodated in IIoT scenarios

Table 1 (continued)

References	Models	Dataset	No. samples	Performance	Key highlight	Limitation
[48]	XGBoost, LightGBM, AdaBoost, CatBoost, and GradientBoost	Custom-built dataset with Modbus/TCP, OPC UA and S7COMM protocols	+ 310 thousand	F1 score for XGBoost: 99%	The study delves into which model can be more cost-effective for deployment in the edge layer or for operation by the devices	The study concentrates solely on a single, self-created dataset
Our work	RF, XGBoost, LightGBM, and TabPFN	Three datasets of 2500 samples from Edge-IIoTSet	2500	Accuracy TabPFN: 97% for 10 classes F1 score TabpFN: 72%	The proposal has identified a solution that enables an ML-based IDS to function with limited data availability	The proposal can classify a maximum of 10 classes

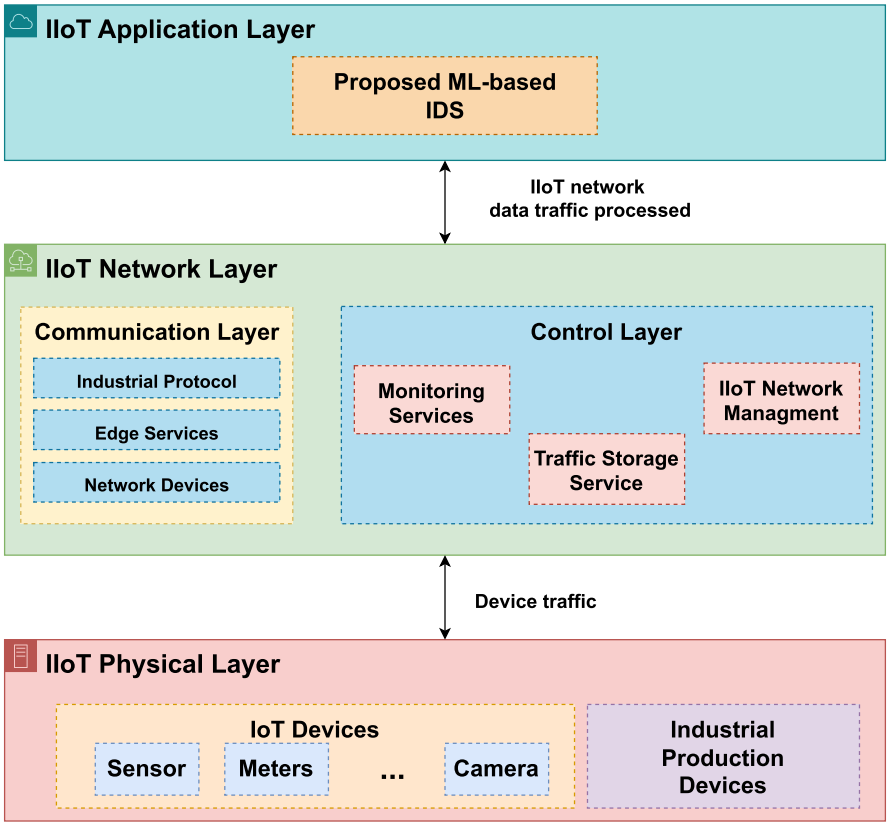

Fig. 1 Proposed IIoT ML-based IDS architecture

The IIoT physical layer includes both IoT devices and industrial machinery. These devices perform functions such as environmental monitoring, raw material transportation, and various supply chain operations, and play a crucial role in the IIoT ecosystem. Each device operates based on a set of industrial design functionalities that guarantee optimal performance and security, each with specific responsibilities. The devices are equipped with inherent security features to prevent unauthorized access and tampering. However, this does not offer complete protection against malicious activities initiated by attackers.

The network layer comprises of two sublayers, each with different components based on their functionalities and implications within the IIoT architecture.

The communication layer initially manages and controls the industrial protocols used by IIoT devices in their communications, including Wi-Fi, Bluetooth, Modbus/TCP, OPC UA, S7COMM, and MQ telemetry transport (MQTT). These protocols enable the transfer of information between different network devices in the IIoT topology, including routers, IoT gateways, supervisory control and data acquisition (SCADA) systems, databases, and web servers. This layer establishes multiple

relationships between IIoT devices and the industrial protocols they use in order for it to be designed. Additionally, this layer takes into account the location of edge services in IIoT environments. These services can improve network configurations, resources, and characteristics due to the benefits of edge computing in IIoT environments [46]. The edge services can be managed in an external network owned by the company of the IIoT network, enabling the unification and centralization of control and management of different IIoT networks that are not physically co-located. Alternatively, they can be located in the Edge Router of the IIoT networks, which establishes a connection with the Internet, and these services will only control the specific IIoT network. Edge services allow the management of diverse protocols and device communications to improve industrial processes and ensure the proper functioning of IIoT applications.

In addition, edge services improve the performance of the control layer, which manages communications between IIoT devices. This layer establishes local monitoring services that periodically interact with our proposed ML-based IDS to distinguish normal from anomalous communications. If anomalous behaviour is detected by the monitoring services, the administrator will be alerted about the potential attack on the IIoT network. The monitoring services will also communicate with the IIoT network management to implement initial countermeasures to mitigate the most significant risks in the network. The element also includes additional functionalities, such as ensuring the correct operation of every IIoT device in the network and the capability to establish and register new devices that may be added to the network in the future. The Traffic Storage Service retains the traffic available for storage to maintain a periodic snapshot of the IIoT network. The service will store information in accordance with the company's privacy and confidentiality policy. As such, the traffic database size, packets, and their information will be automatically filtered.

The IIoT application layer provides the functionality of our IIoT application to the end user, including smart industrial devices and smart factory capabilities. Our ML-based IDS is deployed within this layer. The IDS uses ML algorithms trained with small datasets to detect anomalies in network traffic. It communicates with the control layer, which receives the traffic, classifies it, and returns the prediction result. Additionally, this application could be installed on a network device with sufficient computational and network resources to run the IDS, or it could be integrated into the edge services.

5 Experimentation

The proposed IDS architecture was tested using a public dataset modified to establish three different situations in real IIoT environments in order to detect possible cybersecurity attacks with a small amount of data. This experimentation aimed to analyse the performance of our architecture using the TabPFN technique, and compare it with the ML techniques described in Sect. 2.

In this section, the hardware and dataset set-up used to establish our study with the different models are presented, together with the data science process followed to measure the performance of each model and obtain the results for the situations

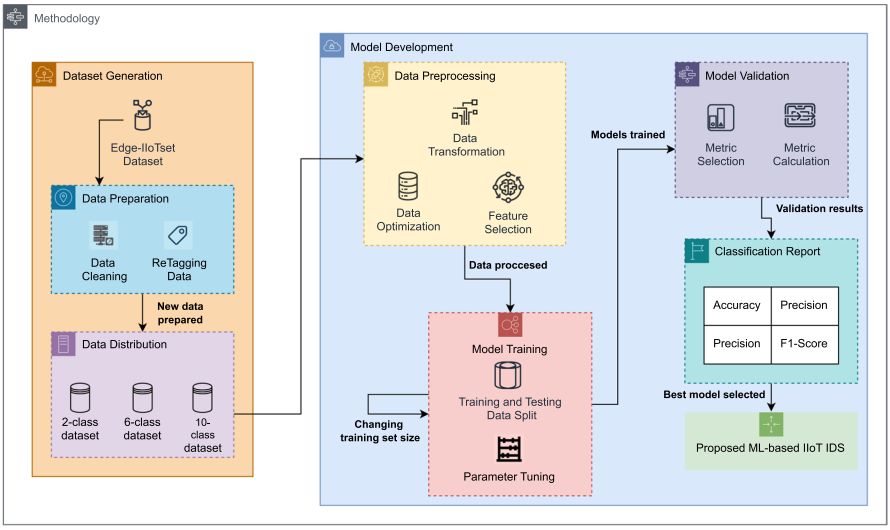

Fig. 2 Proposed IIoT ML-based IDS methodology for classifying attacks with small datasets

considered. The methodology followed during this process is shown in Fig. 2. Moreover, the proposed ML-based IIoT IDS obtained from the experimentation is going to be tested using an external dataset, to prove the functionality of the proposal when is deployed in a IIoT network.

5.1 Hardware set-up

For the experimentation phase of our study, a workstation computer with an Intel i7-13700KF CPU with 32 GB RAM memory running Ubuntu 22.03 LTS was used. Also, the computer included an NVIDIA RTX 3060 Ti graphics card, which was used to speed up the execution of the models. We used Python 3.10 as the programming language, together with the libraries needed for the execution of the different models using the graphics card.

5.2 Dataset generation

Our study is focused on comparing the performance of ML algorithms with small datasets, although in the research community these datasets are not easily available, and the ones that predominate are those with a huge number of samples. For that reason, we decided to use the Edge-IIoT set [50] as a base dataset to generate three other small datasets, whose main differences are the number of classes considered in each one, and the randomly selected packets that they contain in order to avoid possible similarities in the datasets.

5.2.1 Original dataset

The Edge-IIoTset [50] data collection was accomplished by orchestrating an IoT/IIoT testbed design. This specifically tailored testbed incorporates a variety of devices, sensors, protocols, and configurations to provide a comprehensive and representative dataset.

Over ten different types of IoT devices contribute to the data composition of the dataset domains. These devices are designed for specific tasks, such as digital sensors for monitoring temperature and humidity, ultrasonic sensors, water level sensors, pH metres, soil moisture sensors, heart rate sensors, and flame sensors, amongst others.

To enhance the dataset's specificity, diverse features are employed from different sources including alarms, system resources, logs, and network traffic. Especially relevant are the 61 novel characteristics that have been introduced after performing a diligent analysis. These inclusions, chosen from a pool of 1176 features, enhance the intricacy of the dataset, transforming it into a valuable resource for our analytical and subsequent modelling process. These features are shown in Table 2.

At its core, the Edge-IIoTset embodies the connectivity challenges inherent to IoT and IIoT protocols, analysed through a thorough examination of 14 interrelated attacks. These attacks can be broadly grouped into the following five categories:

- *DoS/DDoS attacks* Within these categorized attack types, malicious attackers have a tendency to hinder services available to authorized users, either in isolation or through a distributed approach. In this dataset specifically, four principal techniques often utilized in such situations are discussed: TCP SYN flood, UDP flood, HTTP flood, and ICMP flood.
- *Information gathering attacks* Collecting information about the intended target is typically the first step in any effective attack. This dataset explores three crucial actions that malicious entities often carry out during the information gathering phase: port scanning, OS fingerprinting, and vulnerability scanning.
- *Man in the middle (MitM) attacks* The aim of this type of attack is to relay and control the communication between two entities who consider themselves to be involved in direct interaction. The dataset's focus is on utilizing this attack plan, targeting widely used protocols in almost all modern systems: domain name system (DNS) and address resolution protocol (ARP).
- *Injection attacks* The aim of these attacks is to compromise the security and confidentiality of the system being investigated. Three different methods have been employed for this purpose, namely cross-site scripting (XSS), SQL injection, and upload attacks.
- *Malware attacks* These types of attacks are usually performed by different pieces of malware found in the last years due to the significant damage they have caused and the considerable losses reported. The dataset examination is focused on three variations of such attacks: backdoor, password cracking, and ransomware attacks.

In Table 3, we show how the various types of traffic are distributed amongst the different attacks and techniques in the dataset.

Table 2 Features of Edge-IIoTset and its type

Name	Type	Name	Type
frame.time	Categorical	tcp.payload	Categorical
ip.src_host	Categorical	tcp.seq	Numeric
ip.dst_host	Categorical	tcp.srcport	Numeric
arp.dst.proto_ipv4	Categorical	udp.port	Numeric
arp.opcode	Numeric	udp.stream	Numeric
arp.hw.size	Numeric	udp.time_delta	Categorical
arp.src.proto_ipv4	Categorical	dns.qry.name	Categorical
icmp.checksum	Numeric	dns.qry.name.len	Numeric
icmp.seq_le	Numeric	dns.qry.qu	Numeric
icmp.transmit_timestamp	Numeric	dns.qry.type	Numeric
icmp.unused	Categorical	dns.retransmission	Numeric
http.file_data	Categorical	dns.retransmit_request	Categorical
http.content_length	Numeric	dns.retransmit_request_in	Numeric
http.request.uri.query	Categorical	mqtt.conack.flags	Numeric
http.request.method	Categorical	mqtt.conflag.cleansess	Numeric
http.referer	Categorical	mqtt.conflags	Numeric
http.request.full_uri	Categorical	mqtt.hdrflags	Numeric
http.request.version	Categorical	mqtt.len	Numeric
http.response	Numeric	mqtt.msg_decoded_as	Categorical
http.tls_port	Categorical	mqtt.msg	Categorical
tcp.ack	Numeric	mqtt.msgtype	Numeric
tcp.ack_raw	Numeric	mqtt.proto_len	Numeric
tcp.checksum	Numeric	mqtt.protoname	Categorical
tcp.connection.fin	Numeric	mqtt.topic	Categorical
tcp.connection.rst	Numeric	mqtt.topic_len	Numeric
tcp.connection.syn	Numeric	mqtt.ver	Numeric
tcp.connection.synack	Numeric	mbtcp.len	Numeric
tcp.dstport	Numeric	mbtcp.trans_id	Numeric
tcp.flags	Numeric	mbtcp.unit_id	Numeric
tcp.flags.ack	Numeric	Attack_label	Numeric
tcp.len	Numeric	Attack_type	Categorical
tcp.options	Categorical		

5.2.2 Derived dataset preparation

Once the Edge-IIotset had been selected, it was necessary to establish a process to generate the small datasets needed for our experimentation. Firstly, we explain how the datasets were formed and the proportion between normal and anomalous ones samples, and how we distributed the different anomalous into five and ten

Table 3 Distribution of class values in the Edge-IIoTset

Traffic	Classes	Records	Total
Normal	Normal	11,223,940	11,223,940
Attack	Backdoor	24,862	9,728,708
	DDoS_HTTP	229,022	
	DDoS_ICMP	2,914,354	
	DDoS_TCP	2,020,120	
	DDoS_UDP	3,291,626	
	Fingerprinting	1001	
	Man in the middle	1229	
	Password	1,053,385	
	Port_Scanning	22,564	
	Ransomware	10,925	
	SQL_injection	51,203	
	Uploading	37,634	
	Vulnerability_scanner	145,869	
	XSS	15,915	

anomalous classes. In order to perform this, the following three steps were carried out:

Data cleaning The data from the dataset may include features with missing values, often due to collection errors or the omission of information in certain packets relating to specific protocols. To address this situation, we adapt a standardized approach in our research. Numeric features were assigned a value of -1 to denote the lack of data, while empty strings represent categorical features. This option not only aids data management but also indicates to the model that these characteristics can be relevant to the analysis.

ReTagging data The tags assigned to the samples in the dataset are presented in Table 4. Our study concentrates on the performance classification of models selected with two, six, and ten classes. Consequently, we redefined the tags for every small dataset containing a different number of classes to the original. For the two-class dataset, we designated every attack as an anomalous packet, resulting in two classes: normal and anomalous. For the six-class dataset, we identified the five primary attack categories, which allowed us to develop an IDS that can classify anomalous attacks on the basis of their type. This dataset consists of normal packets, DoS packets, information gathering packets, injection packets, MitM packets, and malware packets. Finally, for the dataset, we chosen to focus on the specific techniques that exhibit the most notable disparities between packets within the general attack group. Where applicable, technical term abbreviations are explained upon first use. This ten-class dataset is comprised of each DoS technique individually, as well as general information gathering attacks, specific SQL injection attacks including their injection class, and packets from the MitM and malware groups.

Table 4 Distribution of classes values in the different small datasets generated

Dataset	Classes	Count	Per cent (%)
2-class	Normal	2000	80
	Anomalous	500	20
6-class	Normal	2000	80
	DoS	203	8.12
	Inf.Gathering	61	2.44
	Injection	101	4.04
	Malware	121	4.84
	Man in the middle	14	0.56
10-Class	Normal	2000	80
	DoS_HTTP	29	1.16
	DoS_ICMP	61	2.44
	DoS_TCP	44	1.76
	DoS_UDP	67	2.68
	Inf.Gathering	83	3.32
	Injection	72	2.88
	SQL_Injection	36	1.44
	Man in the middle	6	0.24
	Malware	102	4.08

Data distribution Once the new data were ready for distribution, we established the ratios of the small datasets, with 2000 samples representing normal traffic and 300 containing anomalous packets. This decision was deemed representative of a realistic scenario in an IIoT network during the traffic extraction process [51]. This property is very interesting and can be detrimental to models that do not perform well in unbalanced environments.

5.3 Model development

During this second phase of our experimentation, we processed the data and trained various models with different iterations based on the size of the training set. We evaluated these models with selected metrics to validate their performance.

5.3.1 Data preprocessing

The datasets require adaptation to make them compatible with the algorithms. It is essential to identify techniques that are suitable for data manipulation and determine the most relevant features. The process involves preparing the data for analysis and also involves the strategic selection of techniques to improve the quality of data processing. Furthermore, the careful selection of features greatly enhances the performance of the algorithms, maximizing their ability to detect patterns and provide accurate results. This stage is divided into the following tasks:

Data transformation ML algorithms are incapable of directly processing alphanumeric data. Consequently, it is necessary to convert all categorical features into a binary format. This transformation can be accomplished through binarization techniques, which generate a new column in the dataset for each categorical value of a feature. This column indicates whether the categorical value is present in the data instance, represented as a Boolean value. The naming convention for these new columns is "feature_name-categorical_value".

In addition, we used the frame.time feature to create a new feature called "delay". This feature adds more temporal information to our dataset, helping us to classify packets more accurately. To calculate the delay feature, we set the first packet or sample as the zero instance and then calculate in seconds the time difference between packets by subtracting the time of the current packet from the previous one. This method enables us to capture the temporal relationships between packets, thereby improving the resilience of our classification model.

Data optimization After successfully completing the data preprocessing phase, it is crucial to thoroughly examine the data storage. This examination is not just a formality, but a critical step that involves correcting any misclassified or incorrectly assigned data types from the preprocessing phase.

This optimization process serves a dual purpose. Firstly, aligning data types with their corresponding storage requirements ensures efficient memory usage. Misclassified data types can lead to unnecessary memory consumption, thus reducing the overall efficiency of the data storage system.

Secondly, this optimization process plays a significant role in facilitating the model training process. Ensuring correct data types is crucial for the interaction between ML algorithms and data.

Feature selection In the context of model training, not all features contribute equally to the predictive power of the model. In fact, some features may even degrade the model's performance. Therefore, it is of paramount importance to identify and select the features that are most influential for prediction. In our study, we employed extremely randomized trees (ERT) [52] during the experimental phase to select the features that provide the most valuable information to the models. The features deemed most relevant by the algorithm are listed in Table 5. These are the features utilized in the subsequent training and validation stages.

5.3.2 Model training and validation

When the data are ready for the proccessing stage, the final dataset is used to train the ML models considered in our study. This training stage is divided into the following tasks:

Training and testing data split The task at hand is of significant importance in our study as our primary objective is to evaluate and compare the performance of various ML algorithms under conditions where the availability of training data is limited. Our dataset, although small with only 2500 samples per dataset, was meticulously

Table 5 Features selected from the Edge-IIoTset dataset during feature selection step

Name	
http.request.version-0.0	icmp.seq_le
http.request.method-0.0	tcp.ack
http.request.version-0	tcp.connection.rst
http.request.method-0	tcp.seq
http.referer-0.0	tcp.len
http.referer-0	http.request.version-HTTP/1.1
tcp.flags	http.request.method-GET
tcp.flags.ack	http.response
tcp.checksum	http.content_length
tcp.ack_raw	tcp.connection.fin
icmp.checksum	dns.qry.name.len
udp.stream	http.request.version-HTTP/1.0
tcp.connection.syn	delay

utilized to conduct a series of experiments. For each iteration, we tested various training data sizes, including 250, 500, 750, and 1000 samples.

The remaining dataset is used to validate the models. The validation process is essential as it offers insight into the generalization of trained models to unseen data, indicating their real-world applicability. We used the stratified hold-out method to split the dataset for training and validation, ensuring consistent proportions of different classes in both splits. This is especially crucial in situations where the dataset is imbalanced, as it prevents the model from being biased towards the majority class.

The experiment aimed not only to compare the performance of the ML algorithms but also to comprehend their behaviour and performance evolution when trained with different data sizes. By doing this, the goal is to study the scalability of these algorithms and their sensitivity to the amount of training data. These are important factors to consider when deploying ML models practically.

Parameter tuning The performance of machine learning algorithms is significantly influenced by their hyperparameters, which can be fine-tuned to enhance training performance. Hyperparameter tuning is a critical aspect of machine learning model development, as it can greatly improve predictive accuracy and generalization capabilities.

In our study, we employed grid search [53] that is a widely recognized and extensively used technique for hyperparameter tuning. It operates by exhaustively exploring a predefined set of hyperparameters to determine the combination that yields the best performance for machine learning models. This is achieved through a heuristic approach. The performance of the model is evaluated for each combination of hyperparameters on a validation set.

The use of grid search ensures that our models are not only optimized for performance but also exhibit robustness. However, it is worth noting that for our TabPFN

Table 6 List of best parameters found using grid search

Algorithm	Best hyperparameters
RF	max_depth: 12 n_estimators: 174 criterion: gini max_features: auto random_state: 42
XGBoost	learning_rate: 0.1 eta: 0.1 max_depth: 6 subsample: 0.8 seed: 42
LightGBM	learning_rate: 0.0952 max_bin: 20 max_depth: 15 num_leaves: 80 subsample:0.75

approach, hyperparameter tuning using grid search is considered unnecessary. The best parameters found for the tree-based algorithms are indicated in Table 6.

Once the training phase is completed, it is necessary to verify the predictions made by the finalized model against the testing dataset. This validation phase is critical for evaluating the model's predictive behaviour during the classification of IIoT traffic. The evaluation of each model is done using the metrics defined in Sect. 2.4, which allow us to compare the performance of each model in the experiment and to consider which one is better for each situation.

5.4 Model deployment

Once each model has been validated and the best one has been selected for the most common scenarios, it is necessary to evaluate it by deploying it into a IIoT network and checking the real performance of the IDS when actual, common, and unknown attacks are running in the scenario. For that reason, the dataset generated by the authors [54] using their own emulator, generates a dataset with multiple different attacks in a IIoT network, which contains multiple OT protocols such as Modbus/TCP, OPC UA, and S7COMM. For this specific case, we deploy an IDS that classifies up to 6 classes. The dataset follows the methodology presented in Fig. 2 with an additional step to retagging the label that identifies the individual packets. This retagging step and the proportion of each class are indicated in Table 7. It can be seen that the manipulation attack is considered as a MiTM attack because manipulation attacks use an MiTM technique to capture the packets which are modified. Moreover, this attack allows us to check the behaviour of the model with the modified packets because it has received no knowledge regarding this attack during the training phase. Otherwise, brute_http is included as a typical brute-force attacks that uses the malware into different login services to infect the device. Finally, payload_user_agent represents the packets derivated to the inclusion into the user agent argument from the HTTP frames of a payload

Table 7 Conversion of the original tags to the tags used by the model deployed

Original tag	Adapted tag	No packets
final_clean	Normal	14,634
brute_http	Malware	2160
payload_user_agent	Injection	2160
ping_of_death_dos	DoS	5533
tcp_flood_dos		
scanner_ack	Inf.Gathering	5709
scanner_tcp		
scanner_udp		
scanner_fin		
scanner_xmas		
scanner_null		
manipulation	MiTM	3003

that tries to use Shellshock vulnerability. The rest of the tags are adapted directly to the category of its purpose.

6 Results

The results obtained during model validation are presented in Tables 8 and 9. We considered each metric, the algorithm used, and the size of the training set in our tests. The analysis has been categorized by the classification type of each dataset, and we obtained the performance of binary, 6-class, and 10-class classification for each situation and training dataset size. Furthermore, a general time complexity analysis is made between the ML algorithms. The metrics highlighted in the tables are the best results obtained in each case studied.

6.1 Binary classification

The results of binary classification using the dataset created for this particular scenario show that all the algorithms obtained full marks for all the metrics. This is even the case when the training dataset size is only 250 samples. The anomalous activities identified in the attacks, and the traits considered in the models during their predictions, are the reasons behind these results. They help in detecting malicious attacks with ease since the information that resides within the traffic packets and the differences between benign and malicious packets are taken into account. The selection of algorithms must therefore take into account additional metrics such as resource consumption and computational cost, in accordance with the requirements of the IIoT scenarios in which the IDS would be deployed.

Table 8 Results of training–testing time used by the ML techniques during the experimentation

Training set size	Algorithm	2-class	6-Class	10-Class
Training–test time (seconds)				
250	RF	1.97	2.40	3.08
	XGBoost	2.84	3.72	4.61
	LightGBM	2.59	3.24	3.79
	TabPFN	**1.86**	**1.95**	**2.50**
500	RF	2.88	3.18	4.17
	XGBoost	4.42	5.42	6.04
	LightGBM	3.61	4.25	5.37
	TabPFN	**1.99**	**2.36**	**2.64**
750	RF	4.17	3.99	5.31
	XGBoost	4.86	6.61	8.43
	LightGBM	4.38	5.77	6.86
	TabPFN	**2.71**	**2.95**	**2.96**
1000	RF	4.04	5.32	7.78
	XGBoost	6.31	7.14	10.42
	LightGBM	4.77	6.39	9.07
	TabPFN	**2.53**	**3.18**	**4.01**

6.2 6-Class classification

When analysing the performance of the models in identifying six traffic packets classes, differences in the behaviour of the models are observed depending on the size of the dataset used for training.

The performance of the models ranges from 56 to 74%, as determined by the F1 score, with 250 training samples being the minimum. The TabPFN proposal performs best for all F1 score, precision, and recall metrics, averaging 74%, as well as 96% in accuracy. Both RF and XGBoost behave similarly, with a marginal difference of 1% in the accuracy and F1 score metrics, although XGBoost may have a slight advantage over RF as F1 score is considered a better metric for imbalanced classification situations.

Training the models using 500 samples results in distinct performances for each one. RF achieves the best performance with 97%, 79%, 78%, and 78% in accuracy, precision, recall, and F1 score, respectively. This is a significant difference with respect to the other proposals. The TabPFN and XGBoost models show a similar performance improvement of between 1 and 7% when trained on this dataset size compared with using a training dataset of 250 samples. In this scenario, LightGBM exhibits the worst performance again. However, its performance improves by around 10% in precision, recall, and F1 score when trained on 500 samples during the training stage instead of 250 samples.

When using 750 training samples, the results indicate that the TabPFN proposal achieves the best performance, being comparable to RF using 500 samples. In addition, the results demonstrate that using this model and XGBoost yields similar results with 97% for the accuracy metric and 74% in the other ones, whereas RF's

Table 9 Results of each technique during experimentation

Training set size	Algorithm	2-Class classification				6-Class classification				10-Class classification			
		Accuracy	Precision	Recall	F1 score	Accuracy	Precision	Recall	F1 score	Accuracy	Precision	Recall	F1 score
250 (10%)	RF	1	1	1	1	**0.96**	0.69	0.69	0.68	0.94	0.52	0.53	0.51
	XGBoost	1	1	1	1	0.95	0.69	0.69	0.69	**0.95**	**0.60**	**0.60**	**0.58**
	LightGBM	1	1	1	1	0.94	0.57	0.56	0.56	0.91	0.43	0.35	0.31
	TabPFN	1	1	1	1	**0.96**	**0.74**	**0.74**	**0.74**	**0.95**	0.55	**0.60**	0.55
500 (20%)	RF	1	1	1	1	**0.97**	**0.79**	**0.78**	**0.78**	**0.96**	0.64	0.62	0.62
	XGBoost	1	1	1	1	**0.97**	0.74	0.76	0.75	**0.96**	0.63	0.63	0.63
	LightGBM	1	1	1	1	0.96	0.68	0.69	0.68	0.94	0.53	0.49	0.44
	TabPFN	1	1	1	1	**0.97**	0.75	0.75	0.75	**0.96**	**0.67**	**0.72**	**0.67**
750 (30%)	RF	1	1	1	1	**0.97**	0.74	0.74	0.74	0.96	0.62	0.64	0.63
	XGBoost	1	1	1	1	**0.97**	0.74	0.74	0.74	0.96	0.62	0.61	0.61
	LightGBM	1	1	1	1	0.95	0.67	0.67	0.66	0.94	0.56	0.48	0.45
	TabPFN	1	1	1	1	**0.97**	**0.79**	**0.78**	**0.78**	**0.97**	**0.72**	**0.73**	**0.71**
1000 (40%)	RF	1	1	1	1	**0.97**	0.74	0.74	0.74	0.96	0.61	0.61	0.60
	XGBoost	1	1	1	1	**0.97**	0.79	0.78	0.78	0.96	0.63	0.60	0.61
	LightGBM	1	1	1	1	0.96	0.72	0.72	0.72	0.94	0.51	0.46	0.41
	TabPFN	1	1	1	1	**0.97**	**0.82**	**0.80**	**0.81**	**0.97**	**0.73**	**0.73**	**0.72**

6-Class TabPFN Confusion Matrix						
TARGET / OUTPUT	Normal	DoS	Inf.Gathering	Injection	Malware	MiTM
Normal	1200 79.89%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%
DoS	0 0.00%	97 6.46%	0 0.00%	0 0.00%	0 0.00%	0 0.00%
Inf.Gathering	0 0.00%	0 0.00%	10 0.67%	0 0.00%	0 0.00%	0 0.00%
Injection	0 0.00%	6 0.40%	8 0.53%	61 4.06%	9 0.60%	0 0.00%
Malware	0 0.00%	19 1.26%	19 1.26%	0 0.00%	64 4.26%	0 0.00%
MiTM	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	9 0.60%

Fig. 3 Confusion matrix of TabPFN model with 1000 training samples for 6 classes

performance is comparatively worse than when using 500 samples in the training dataset. A similar behaviour is found in the evaluation of LightGBM's results, where it can be observed that the performance improvement is around 1–2% compared with using 500 training samples.

Finally, the use of 1000 training samples shows a significant improvement in the metrics of the models. XGBoost and LightGBM show an improvement of 4–6% in the imbalanced metrics. In the case of the XGBoost model, it achieves 97% in accuracy, 79% in precision, 78% in recall, and 78% for the F1 score, which is better than the other two models. Furthermore, the RF algorithm does not show an improvement in performance when trained with 750 samples and has similar results to LightGBM with only a 2% difference in precision, recall, and F1 score, and a 1% difference in accuracy. In this scenario, our TabPFN proposal achieves the highest overall performance, with 97% in accuracy and around 80% for the other metrics, making this option the most viable to be deployed as a solution in IIoT networks. Figure 3 shows the distribution of predictions amongst the various classes. The confusion matrix indicates that the ML model performs well overall, but struggles to differentiate between certain types of attacks. In particular, DoS attacks are sometimes misclassified as information gathering and injection attacks, likely due to similarities in network traffic patterns. Similarly, information gathering attacks, which often involve a lot of seemingly normal traffic, may be misclassified as DoS or injection attacks.

Injection attacks, which involve inserting malicious data packets into the network, may also be misclassified as DoS or information gathering attacks. The model accurately classifies most malware attacks, but it misclassifies a few instances as man-in-the-middle (MiTM) ones. This is likely due to both types of attacks involving intercepting and possibly altering network traffic. Similarly, some MiTM attacks are misclassified as malware ones, again likely due to the interception of traffic common to both attack types. These misclassifications indicate potential areas for model refinement and improvement.

In summary, the TabPFN model demonstrates superior performance across the board with training set sizes of 250, 750, and 1000 samples, with the best results being achieved in the final experiment with 97% in accuracy, 82% in precision, 80% in recall, and 81% for the F1 score, showing improvements in the results of 3–10% compared with the other techniques. RF delivers the best results during the training phase with a sample size of 500. Most situations show similar performances for RF and XGBoost. This is indicated by the performance metric scores, with a difference ranging from 2 to 18% when compared with the other algorithms.

6.3 10-Class classification

These experiments demonstrate the performance of the models in identifying specific attack techniques, including variations in behaviour as the number of classes to classify increases.

With a training set size of 250, the overall model performance is inadequate to be considered as a viable solution. In this scenario, the XGBoost algorithm achieves the best performance, with 60%, 60%, and 58% in precision, recall, and F1 score, respectively. The second-best method is TabPFN, which has the same recall metric as XGBoost, but behaves worse in the precision and F1 score metrics, scoring only 55% in each of them. The worst performance by far is observed when using the LightGBM technique, with 43%, 35%, and 31% for precision, recall, and F1 score, respectively.

Using 500 samples during the training stage leads to improved performance in all models, as evidenced by the validation results. TabPFN shows the following metrics: 67% precision, 72% recall, 67% F1 score, and 96% accuracy, all of which are highlighted below. These performance enhancements suggest that increasing the number of samples used in the training stage could result in acceptable performance for IIoT situations.

When the training set size is increased to 750, the performance of the TabPFN solution improves, with the following metrics being achieved during classification: 72% in precision, 73% in recall, and 71% in F1 score. The remaining solutions demonstrate a similar, or inferior, performance compared with a training set size of 500. This situation may occur depending on the samples included during the training stage, and the new samples introduced may overfit the models.

When using 1000 training samples, the same situation as that mentioned above arises, and TabPFN again emerges as the best solution with a performance of 73% in the precision and recall metrics and 72% for the F1 score, proving that it is the ideal

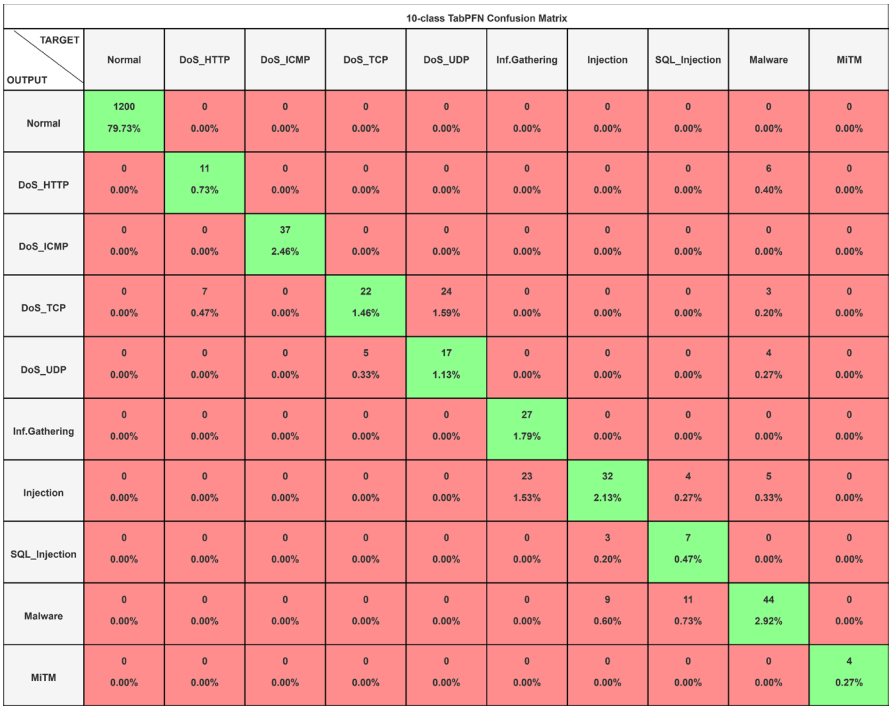

Fig. 4 Confusion matrix of TabPFN model with 1000 training samples for 10 classes

candidate to classify attack techniques in IIoT networks. In general, the other models produce worse results than those when using 500 or 750 training samples. When considering the three scenarios, XGBoost maintains an overall performance of 62% in precision, recall, and F1 score, while RF's performance decreases by 1–3% in precision, recall, and F1 score when compared with its best performance when using 500 samples during the training stage. LightGBM displays performance that is comparable to RF's, with a 2–3% reduction in its effectiveness. Figure 4 shows the distribution of predictions across various classes. Additionally, the confusion matrix illustrates the performance of the TabPFN model. Although the model accurately identifies normal traffic and certain attacks such as DoS_ICMP and MiTM, there are some misclassifications that require further investigation. One particular area for improvement is distinguishing between DoS_TCP attacks, DoS_UDP, and DoS_HTTP. The model struggles to differentiate between DoS attacks that use different transport protocols and to determine if the attack is using the HTTP application protocol. Another challenging attack to classify is Inf.Gathering related techniques, where approximately 50% of the packets have been classified as injection attacks. This may be due to the different flags used in the TCP protocol for scanning techniques. Furthermore, SQL injection has proved difficult to be accurately classified even within its general attack category. This is due to 11 samples of this attack being identified as malware behaviour, which may be related to the various attacks

performed by different malware samples. The limitations of the current model and the need for further refinement are highlighted by these misclassifications.

To conclude, this experiment indicates that, in general, TabPFN outperforms the other machine learning models included in the study, particularly in 10-class classification scenarios that involve 1000 training set samples. The difference in performance is considerable in such cases. RF and XGBoost, however, achieve superior results in two specific scenarios, so these outcomes might be considered depending on the environment in which the IDS is to be deployed. LightGBM, in contrast, exhibits the poorest performance in all scenarios, highlighting the limited utility of this technique when only a few samples are available to train the model.

6.4 Analysis of algorithmic complexity

Upon the examination of the performance metrics of the classifiers under a variety of conditions, the complexity of the models warrants a thorough investigation. The evaluation of the complexity-performance trade-off of the algorithms is of paramount importance to pinpoint the algorithm that yields the best performance with the least complexity. This factor becomes increasingly significant in an IIoT setting due to the characteristics of the devices deployed in IIoT architectures.

Table 8 delineates notable differences in time complexity amongst the models. TabPFN emerges as the most time-efficient option, as evidenced by its superior training–test time metric results across all experimental scenarios compared to other algorithms. For a 10-class problem with a training set size of 1000, TabPFN requires only 4.01 s. The results indicate that RF is the second most favourable option in terms of complexity, with times ranging from 1.97 s for a training set size of 250–4.04 s for a training set size of 1000, suggesting that RF possesses a reasonable time complexity. Conversely, XGBoost exhibits the highest time complexity amongst the four algorithms, with time complexity escalating significantly with the size of the training set and the number of classes. For a 10-class problem with a training set size of 1000, XGBoost necessitates the longest time amongst all the scenarios at 10.42 s. LightGBM has a lower time complexity than XGBoost but higher than RF and TabPFN, requiring 9.07 s for the same problem.

The metric analysis indicates that TabPFN outperforms in most of the situations presented during the experimentation. RF and XGBoost exhibit superior performance for the 6-class and 10-class classifications with 500 and 250 training samples, respectively. However, the difference between the best and second-best algorithms, according to the metrics, is typically only 3–5%. This suggests that TabPFN should be the algorithm of choice for most situations in the IIoT environment.

The computational resources of the devices that are tasked to run and deploy the IDS based on ML algorithms are a critical factor in algorithm selection. If the algorithm possesses a higher computational complexity than the device can support, the performance of the IDS may be compromised. Therefore, the analysis of time complexity demonstrates that TabPFN yields the best results in this regard. RF is the second most favourable option based on its classification performance and is a suitable ML technique to deploy.

6-Class TabPFN Confusion Matrix						
TARGET / OUTPUT	Normal	DoS	Inf.Gathering	Injection	Malware	MiTM
Normal	14634 44.08%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	1316 3.96%
DoS	0 0.00%	5101 15.36%	1191 3.59%	0 0.00%	0 0.00%	0 0.00%
Inf.Gathering	0 0.00%	0 0.00%	3873 11.67%	0 0.00%	264 0.80%	0 0.00%
Injection	0 0.00%	110 0.33%	227 0.68%	1931 5.82%	0 0.00%	0 0.00%
Malware	0 0.00%	322 0.97%	418 1.26%	0 0.00%	1896 5.71%	0 0.00%
MiTM	0 0.00%	0 0.00%	0 0.00%	229 0.69%	0 0.00%	1687 5.08%

Fig. 5 Confusion matrix of TabPFN model during deployment experimentation

6.5 Analysis of TabPFN model deployment

The IDS deployed, as depicted the confusion matrix in Fig. 5, was evaluated within an IIoT context using the dataset delineated in the Sect. 5.4. The IDS demonstrated robust performance in accurately classifying normal activities, with no instances of false positives. However, the model exhibited certain limitations, particularly in differentiating between various types of attacks.

For example, while the model accurately identified DoS attacks a majority of the time, it erroneously classified information gathering as DoS. This suggests that there may be overlapping features between these two classes. Similarly, the information gathering class was frequently misclassified as DoS, injection, and malware, indicating a need for additional training to enhance the model's ability to distinguish between these attack types.

The model exhibited commendable performance in identifying injection attacks, but it also misclassified DoS and information gathering as injection ones, implying shared features amongst these classes. The model accurately identified malware attacks, but it also misclassified information gathering and MiTM attacks as malware. This suggests a potential requirement for supplementary training data for these classes or a review of feature importance to comprehend this overlap, which

could be attributed to the common behaviour of malware binaries with these attack categories.

Lastly, the model demonstrated satisfactory performance in identifying MiTM attacks, but it also misclassified Normal activities as MiTM attacks. This could potentially be due to the fact that manipulated packets are considered as normal traffic because the model does not account for this possibility, and the correct MiTM detection is made by the address resolution protocol (ARP) spoofing packets that are used by attackers to intercept legitimate packets.

Upon analysis of the confusion matrix results, it was determined that the model achieved performance metrics of 87%, 85%, 82%, and 83% for accuracy, precision, recall, and F1 score, respectively. This indicates that, although the deployed model yielded a lower result in the accuracy metric, it demonstrated superior results in the precision, recall, and F1 score metrics. This is particularly valuable for imbalanced datasets, with an improvement ranging between 2 and 5%.

7 Conclusions and future work

Various IDS techniques have been proposed to safeguard the IIoT environment from external attackers and intruders and the threats that they create. The utilization of big data, in conjunction with ML-based classifiers, has proved to be a powerful tool in the study of large datasets for the protection of IIoT devices in the current cybersecurity landscape.

However, in some industrial environments, accessing the large volumes of data required to train machine learning models may not be feasible. Therefore, alternative models must be sought to compensate for this lack of data.

The objective of this work was to address this issue by developing a TabPFN IDS. Our aim was to compare our proposal with other state-of-the-art ML models that have demonstrated good results in tabular data classification with larger datasets.

The study results indicate that TabPFN outperforms other techniques in most of the analysed situations. It achieved an 81% F1 score when classifying packets into 6 classes and a 72% F1 score when classifying into 10 classes.

These results validate the effectiveness of our TabPFN proposal and demonstrate its readiness to be deployed as a cybersecurity solution in IIoT environments, and prove that it has the potential to enhance the security measures of IIoT systems, contributing to the broader goal of creating safer and more secure digital environments.

In conclusion, our study has shown that the TabPFN IDS is effective for detecting intrusions in IIoT systems, highlighting its potential as a cybersecurity solution for real-world deployment. However, further exploration and improvement are necessary. Thus, future research should focus on addressing key challenges and advancing the capabilities of the TabPFN IDS as well as aim to extend the scope and impact of our research, providing valuable directions for further exploration and innovation in the field of IIoT cybersecurity. Some suggestions could be the following:

- *Dynamic model adaptation* Investigate methods for dynamically adapting the TabPFN model to evolving IIoT environments. Develop mechanisms for con-

tinuously updating the model based on incoming data streams and changing network conditions, ensuring ongoing effectiveness against emerging threats.

- *Scalability and resource optimization* Examine strategies to optimize the scalability and resource utilization of the TabPFN IDS, particularly for deployment in resource-constrained IIoT devices or edge computing environments. Explore techniques such as model compression, quantization, or distributed inference to reduce memory and computational requirements.
- *Anomaly detection and zero-day threats* Expand the capabilities of the TabPFN IDS to incorporate advanced anomaly detection techniques for identifying zero-day threats and previously unseen attack patterns. Investigate unsupervised learning approaches or anomaly detection algorithms to complement the existing classification-based detection mechanisms.
- *Integration with threat intelligence feeds* Augment the TabPFN IDS by integrating with external threat intelligence feeds and cybersecurity information sources. Develop mechanisms to leverage real-time threat intelligence data for proactive threat detection and response, enabling the system to adapt to the latest threat landscape.
- *Real-world industrial deployment and evaluation* Conduct real-world deployment experiments to evaluate the performance and effectiveness of the TabPFN IDS in operational IIoT environments. Collaborate with industry partners or deploy the system in pilot deployments to assess its practical utility, usability, and impact on overall cybersecurity posture.
- *Development and deployment of a network IDS (NIDS) in real-world scenarios* The TabPFN IDS model should be adapted and deployed in real-world IIoT scenarios for the purpose of enhancing the detection of anomalies and identification of attacks within traffic flows. This process involves leveraging the model's pattern recognition capabilities in a variety of environments, such as those comprising corporate networks or critical infrastructure. In addition, it is important to ensure that the model undergoes continuous evolution in order to adapt to the emergence of new threats and the complexities of evolving network environments.

Author contributions Sergio Ruiz-Villafranca conceived and designed the architecture and experiments, performed the experiments, analysed the data, performed the computation work, prepared figures and tables, authored and reviewed drafts of the paper, and approved the final draft. Jose Roldan-Gomez conceived and designed the experiments, performed the experiments, prepared figures and tables, authored and reviewed drafts of the paper, and approved the final draft. Juan Manuel Castelo Gómez conceived and designed the architecture, analysed the data, performed the computation work, prepared figures and tables, authored and reviewed drafts of the paper, and approved the final draft. Javier Carrillo-Mondéjar analysed the data, performed the computation work, authored and reviewed drafts of the paper, and approved the final draft. José Luis Martinez prepared and reviewed figures and tables, authored and reviewed drafts of the paper, and approved the final draft.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. This work was supported by the Universidad de Castilla-La Mancha under the predoctoral contract 2022-PRED-20677 and the project 2023-GRIN-34056, both financed by the European Social Fund Plus

(FSE+), and by the JCCM under the project SBPLY/21/180501/000195. Also, this work is part of the R&D project PID2021-123627OB-C52, funded by the MCIN and the European Regional Development Fund: "a way of making Europe", and it is also funded by MCIN/AEI/10.13039/501100011033 (FEDER, EU) under grant PID2022-142332OA-I00. Javier Carrillo-Mondéjar is also supported by MCIN/AEI/10.13039/501100011033 and European Union NextGenerationEU/PRTR under grant TED2021-131115A-I00, by the Spanish National Cybersecurity Institute (INCIBE) under *Proyectos Estratégicos de Ciberseguridad—CIBERSEGURIDAD EINA UNIZAR*, and by the University, Industry and Innovation Department of the Aragonese Government under *Programa de Proyectos Estratégicos de Grupos de Investigación* (DisCo research group, ref. T21-23R).

Data availability The data used in this work is available in the following link: https://data.mendeley.com/datasets/kyz3nw793w/draft?a=c67afe5e-13fb-4ead-96b5-c7d00c4998fd.

Declarations

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

Ethical approval Not applicable.

References

1. Ivanov D, Tang C, Dolgui A, Battini D, Das A (2020) Researchers' perspectives on industry 4.0: multi-disciplinary analysis and opportunities for operations management. Int J Prod Res 59:1–24. https://doi.org/10.1080/00207543.2020.1798035
2. Maddikunta PKR, Pham Q-V, Prabadevi B, Deepa N, Dev K, Gadekallu TR, Ruby R, Liyanage M (2022) Industry 5.0: a survey on enabling technologies and potential applications. J Ind Inf Integr 26:100257. https://doi.org/10.1016/j.jii.2021.100257
3. Golovianko M, Terziyan V, Branytskyi V, Malyk D (2023) Industry 4.0 vs. industry 5.0: co-existence, transition, or a hybrid. Procedia Comput Sci 217:102–113. https://doi.org/10.1016/j.procs.2022.12.206. (**4th International Conference on Industry 4.0 and Smart Manufacturing**)
4. Möller DPF, Vakilzadian H, Haas RE (2022) From industry 4.0 towards industry 5.0. In: 2022 IEEE International Conference on Electro Information Technology (eIT), pp 61–68. https://doi.org/10.1109/eIT53891.2022.9813831
5. Dhirani LL, Armstrong E, Newe T (2021) Industrial IoT, cyber threats, and standards landscape: evaluation and roadmap. Sensors 21(11):3901
6. Halbouni A, Gunawan TS, Habaebi MH, Halbouni M, Kartiwi M, Ahmad R (2022) Machine learning and deep learning approaches for cybersecurity: a review. IEEE Access 10:19572–19585. https://doi.org/10.1109/ACCESS.2022.3151248
7. Cao X, Wang Y, Chen B, Zeng N (2021) Domain-adaptive intelligence for fault diagnosis based on deep transfer learning from scientific test rigs to industrial applications. Neural Comput Appl 33:4483–4499
8. Hollmann N, Müller S, Eggensperger K, Hutter F (2022) Meta-learning a real-time tabular autoML method for small data. arXiv e-prints, arXiv:2207.01848

9. Li L, Kumar Damarla S, Wang Y, Huang B (2021) A gaussian mixture model based virtual sample generation approach for small datasets in industrial processes. Inf Sci 581:262–277. https://doi.org/10.1016/j.ins.2021.09.014
10. Milić SD, Durovic Z, Stojanović MD (2023) Data science and machine learning in the IIoT concepts of power plants. Int J Electr Power Energy Syst 145:108711. https://doi.org/10.1016/j.ijepes.2022.108711
11. Mahesh B (2020) Machine learning algorithms-a review. Int J Sci Res IJSR 9:381–386
12. Roldán J, Boubeta-Puig J, Luis Martínez J, Ortiz G (2020) Integrating complex event processing and machine learning: an intelligent architecture for detecting IoT security attacks. Expert Syst Appl 149:113251. https://doi.org/10.1016/j.eswa.2020.113251
13. Suthishni DNP, Kumar KSS (2022) A review on machine learning based security approaches in intrusion detection system. In: 2022 9th International Conference on Computing for Sustainable Global Development (INDIACom), pp 341–348. https://doi.org/10.23919/INDIACom54597.2022.9763261
14. Smys S, Chen JIZ, Shakya S (2020) Survey on neural network architectures with deep learning. J Soft Comput Paradig JSCP 2(03):186–194
15. Grinsztajn L, Oyallon E, Varoquaux G (2022) Why do tree-based models still outperform deep learning on typical tabular data? In: Koyejo S, Mohamed S, Agarwal A, Belgrave D, Cho K, Oh A (eds) Advances in neural information processing systems, vol 35. Curran Associates, Inc., pp 507–520 .https://proceedings.neurips.cc/paper_files/paper/2022/file/0378c7692da36807bdec87ab043cdadc-Paper-Datasets_and_Benchmarks.pdf
16. Mohammadpour L, Ling TC, Liew CS, Aryanfar A (2022) A survey of CNN-based network intrusion detection. Appl Sci 12(16):8162. https://doi.org/10.3390/app12168162
17. Sun P, Liu P, Li Q, Liu C, Lu X, Hao R, Chen J (2020) DL-IDS: extracting features using CNN-LSTM hybrid network for intrusion detection system. Secur Commun Netw 2020:1–11
18. Attia A, Faezipour M, Abuzneid A (2020) Network intrusion detection with XGBoost and deep learning algorithms: an evaluation study. In: 2020 International Conference on Computational Science and Computational Intelligence (CSCI), pp 138–143. https://doi.org/10.1109/CSCI51800.2020.00031
19. Gorishniy Y, Rubachev I, Khrulkov V, Babenko A (2021) Revisiting deep learning models for tabular data. Adv Neural Inf Process Syst 34:18932–18943
20. Shwartz-Ziv R, Armon A (2022) Tabular data: deep learning is not all you need. Inf Fusion 81:84–90. https://doi.org/10.1016/j.inffus.2021.11.011. (**Accessed 2022-12-29**)
21. Heidari A, Jabraeil Jamali MA (2023) Internet of things intrusion detection systems: a comprehensive review and future directions. Clust Comput 26(6):3753–3780
22. Rosay A, Cheval E, Ghanmi M, Carlier F, Leroux P (2023) Study of network ids in IoT devices. SN Comput Sci 4(4):407
23. Qurashi MA, Angelopoulos CM, Katos V (2020) An architecture for resilient intrusion detection in IoT networks. In: ICC 2020—2020 IEEE International Conference on Communications (ICC), pp 1–7. https://doi.org/10.1109/ICC40277.2020.9148868
24. Alabadi M, Habbal A, Wei X (2022) Industrial internet of things: requirements, architecture, challenges, and future research directions. IEEE Access 10:66374–66400. https://doi.org/10.1109/ACCESS.2022.3185049
25. Fröhlich AA, Horstmann LP, Hoffmann JLC (2023) A secure IIoT gateway architecture based on trusted execution environments. J Netw Syst Manag 31(2):32
26. Choudhry MD, Jeevanandham S, Rose B, Sruthi MP (2022) Machine learning frameworks for industrial internet of things (IIoT): a comprehensive analysis. In: 2022 First International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT), pp 1–6. https://doi.org/10.1109/ICEEICT53079.2022.9768630
27. Nuaimi M, Fourati LC, Hamed BB (2023) Intelligent approaches toward intrusion detection systems for industrial internet of things: a systematic comprehensive review. J Netw Comput Appl 215:103637. https://doi.org/10.1016/j.jnca.2023.103637
28. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V et al (2011) Scikit-learn: machine learning in python. J Mach Learn Res 12:2825–2830
29. Ogunleye A, Wang Q-G (2019) XGBoost model for chronic kidney disease diagnosis. IEEE/ACM Trans Comput Biol Bioinform 17(6):2131–2140
30. Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, Liu T-Y (2017) LightGBM: a highly efficient gradient boosting decision tree. Adv Neural Inf Process Syst 30:1–9

31. ZHAO G, WANG Y, WANG J (2023) Intrusion detection model of internet of things based on Light-GBM. IEICE Trans Commun E106.B(8):622–634. https://doi.org/10.1587/transcom.2022EBP3169
32. Müller S, Hollmann N, Arango SP, Grabocka J, Hutter F. Transformers can do Bayesian inference. arXiv. https://doi.org/10.48550/arXiv.2112.10510. http://arxiv.org/abs/2112.10510 Accessed 17 May 2023
33. Hollmann N, Müller S, Eggensperger K, Hutter F. TabPFN: a transformer that solves small tabular classification problems in a second. arXiv. https://doi.org/10.48550/arXiv.2207.01848. http://arxiv.org/abs/2207.01848 Accessed 17 May 2023
34. Jefferys WH, Berger JO (1992) Ockham's razor and Bayesian analysis. Am Sci 80(1):64–72 (**Accessed 2023-08-14**)
35. Adler P, Falk C, Friedler SA, Nix T, Rybeck G, Scheidegger C, Smith B, Venkatasubramanian S (2018) Auditing black-box models for indirect influence. Knowl Inf Syst 54:95–122
36. Mahadik S, Pawar PM, Muthalagu R (2023) Efficient intelligent intrusion detection system for heterogeneous Internet of Things (HetIoT). J Netw Syst Manag 31(1):2
37. Sharafaldin I, Lashkari AH, Hakak S, Ghorbani AA (2019) Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. In: 2019 International Carnahan Conference on Security Technology (ICCST), pp 1–8. https://doi.org/10.1109/CCST.2019.8888419
38. Sung T-W, Lee C-Y, Gaber T, Nassar H et al (2023) Innovative artificial intelligence-based Internet of Things for smart cities and smart homes. Hindawi, London
39. Zolanvari M (2021) Wustl-IIoT-2021 dataset. https://doi.org/10.21227/yftq-n229
40. Awotunde JB, Folorunso SO, Imoize AL, Odunuga JO, Lee C-C, Li C-T, Do D-T (2023) An ensemble tree-based model for intrusion detection in industrial internet of things networks. Appl Sci 13(4):2479. https://doi.org/10.3390/app13042479
41. Moustafa N (2020) ToN-IoT dataset. https://cloudstor.aarnet.edu.au/plus/s/ds5zW91vdgjEj9i
42. Mohy-eddine M, Guezzaz A, Benkirane S, Azrour M (2022) An effective intrusion detection approach based on ensemble learning for IIoT edge computing. J Comput Virol Hack Tech 19:1–13
43. Peterson JM, Leevy JL, Khoshgoftaar TM (2021) A review and analysis of the Bot-IoT dataset. In: 2021 IEEE International Conference on Service-Oriented System Engineering (SOSE), pp 20–27. https://doi.org/10.1109/SOSE52839.2021.00007
44. Le T-T-H, Oktian YE, Kim H (2022) XGBoost for imbalanced multiclass classification-based industrial internet of things intrusion detection systems. Sustainability 14(14):8707. https://doi.org/10.3390/su14148707
45. Al-Hawawreh M, Sitnikova E, Aboutorab N (2022) X-IIoTID: a connectivity-agnostic and device-agnostic intrusion data set for industrial internet of things. IEEE Internet Things J 9(5):3962–3977. https://doi.org/10.1109/JIOT.2021.3102056
46. Altunay HC, Albayrak Z (2023) A hybrid CNN+LSTM-based intrusion detection system for industrial IoT networks. Eng Sci Technol Int J 38:101322. https://doi.org/10.1016/j.jestch.2022.101322
47. Moustafa N, Slay J (2015) UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: 2015 Military Communications and Information Systems Conference (MilCIS), pp 1–6. https://doi.org/10.1109/MilCIS.2015.7348942
48. Ruiz-Villafranca S, Roldán-Gómez J, Carrillo-Mondéjar J, Gómez JMC, Villalón JM (2023) A MEC-IIoT intelligent threat detector based on machine learning boosted tree algorithms. Comput Netw 233:109868. https://doi.org/10.1016/j.comnet.2023.109868
49. Zhou J, Cao Z, Dong X, Vasilakos AV (2017) Security and privacy for cloud-based IoT: challenges. IEEE Commun Mag 55(1):26–33. https://doi.org/10.1109/MCOM.2017.1600363CM
50. Ferrag MA, Friha O, Hamouda D, Maglaras L, Janicke H (2022) Edge-IIoTset: a new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning. IEEE Access 10:40281–40306. https://doi.org/10.1109/ACCESS.2022.3165809
51. Yang Z, Liu X, Li T, Wu D, Wang J, Zhao Y, Han H (2022) A systematic literature review of methods and datasets for anomaly-based network intrusion detection. Comput Secur 116:102675. https://doi.org/10.1016/j.cose.2022.102675
52. Liang Y, Zhang S, Qiao H, Yao Y (2021) iPromoter-ET: identifying promoters and their strength by extremely randomized trees-based feature selection. Anal Biochem 630:114335. https://doi.org/10.1016/j.ab.2021.114335
53. Liashchynskyi P, Liashchynskyi P (2019) Grid search, random search, genetic algorithm: a big comparison for NAS

54. Ruiz-Villafranca S, Carrillo-Mondéjar J, Gómez J, Roldán-Gómez J (2023) MECInOT: a multi-access edge computing and industrial internet of things emulator for the modelling and study of cybersecurity threats. J Supercomput 79:1–39. https://doi.org/10.1007/s11227-023-05098-2

Authors and Affiliations

Sergio Ruiz-Villafranca[1] · José Roldán-Gómez[2] · Juan Manuel Castelo Gómez[3] · Javier Carrillo-Mondéjar[4] · José Luis Martinez[1]

✉ Juan Manuel Castelo Gómez
juanmanuel.castelo@upm.es

Sergio Ruiz-Villafranca
sergio.rvillafranca@uclm.es

José Roldán-Gómez
roldangjose@uniovi.es

Javier Carrillo-Mondéjar
jcarrillo@unizar.es

José Luis Martinez
joseluis.martinez@uclm.es

1 University of Castilla-La Mancha, Campus Universitario s/n, 02006 Albacete, Albacete, Spain

2 Department of Computer Science, University of Oviedo, Gijón, Spain

3 Department of Information Systems, Polytechnic University of Madrid, Madrid, Spain

4 Department of Computer Science and Systems Engineering, Universidad de Zaragoza, Zaragoza, Spain