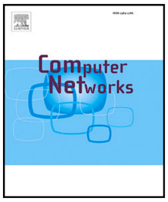
Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



A MEC-IIoT intelligent threat detector based on machine learning boosted tree algorithms

Sergio Ruiz-Villafranca [a,*], José Roldán-Gómez [b,1], Javier Carrillo-Mondéjar [c,1], Juan Manuel Castelo Gómez [a,1], José Miguel Villalón [a,1]

[a] *University of Castilla-La Mancha, Albacete Research Institute of Informatics, Investigación 2, Albacete 02071, Spain*
[b] *Department of Computer Science, University of Oviedo, Gijón, Spain*
[c] *University of Zaragoza, Zaragoza, Spain*

ARTICLE INFO

Dataset link: https://data.mendeley.com/datasets/bk73vtsrpb/draft?a=0146acec-5daa-4968-ace3-34896ae68530

Keywords:
Cybersecurity
Multi-access Edge Computing
Machine Learning
Intrusion detection system
Industrial Internet of Things

ABSTRACT

In recent years, new management methods have appeared that mark the beginning of a new industrial revolution called Industry 4.0 or the Industrial Internet of Things (IIoT). IIoT brings together new emerging technologies, such as the Internet of Things (IoT), Deep Learning (DL) and Machine Learning (ML), that contribute to new applications, industrial processes and efficiency management in factories. This combination of new technologies and contexts is paired with Multi-access Edge Computing (MEC) to reduce costs through the virtualisation of networks and services. As these new paradigms increase in growth, so does the number of threats and vulnerabilities, making IIoT a very desirable target for cybercriminals. In addition, IIoT devices have certain intrinsic limitations, especially due to their limited resources, and this makes it impossible, in many cases, to detect attacks by using solutions designed for other paradigms. So it is necessary to design, implement and evaluate new solutions or adapt existing ones. Therefore, this paper proposes an intelligent threat detector based on boosted tree algorithms. Such detectors have been implemented and evaluated in an environment specifically designed to test IIoT deployments. In this way, we can learn how these algorithms, which have been successful in multiple contexts, behave in a paradigm with known constraints. The results obtained in the study show that our intelligent threat detector achieves a mean efficiency of between 95%–99% in the F1 Score metric, indicating that it is a good option for implementation in these scenarios.

1. Introduction

Since the First Industrial Revolution until today, with Industry 4.0 or the Industrial Internet of Things (IIoT), the industrial environment has evolved with the aim of improving the performance and efficiency of factories. IIoT provides these factories with new emerging technologies focused mainly on data generation and management, such as the Internet of Things (IoT), Big Data, next-generation networks with 5G, and new applications in the field of industry, such as artificial vision, Deep Learning (DL) and Machine Learning (ML) models for the optimisation of industrial processes [1]. All these technologies working together has led to the appearance of new types of factories and an improvement in traditional industries. In addition, these new technologies have to work together with traditional Operational Technology (OT) services until the two have successfully converged. Some authors have proposed to start talking about Industry 5.0 [2], in which the main goal is to solve the problems related to including IIoT technologies in traditional industry. The main way to solve these problems is by carrying out an independent integration for each company focused on the advantages and functionalities of these technologies. However, it is not only IIoT itself that has enabled factories to improve: Multi-access Edge Computing (MEC), which is designed to bring the advantages of cloud computing closer to companies and give support to emerging technologies, also enables and improves the functionality of many of the technologies that incorporate IIoT. Thanks to the amount of computing and networking resources allocated in it, MEC enables the IIoT to provide better services to factories [3]. These services are mainly provided through virtualisation technology, which is one of the bases of MEC, which is an evolved form of edge computing designed to give mobility support to users and applications. MEC allows computational

[*] Corresponding author.
E-mail addresses: sergio.rvillafranca@uclm.es (S. Ruiz-Villafranca), roldangjose@uniovi.es (J. Roldán-Gómez), jcarrillo@unizar.es (J. Carrillo-Mondéjar), juanmanuel.castelo@uclm.es (J.M.C. Gómez), josemiguel.villalon@uclm.es (J.M. Villalón).
[1] Contributing author.

https://doi.org/10.1016/j.comnet.2023.109868
Received 23 January 2023; Received in revised form 31 May 2023; Accepted 6 June 2023
Available online 15 June 2023

and networking resources to be closer to factories, thus providing advantages such as low-latency resource access, scalability and anywhere access to these computational resources. In addition, MEC allows communication between devices regardless of which technology, hardware, or protocol they use for the transportation of information, thanks to the abstraction that virtualisation provides [3].

However, IIoT and MEC also introduce some weaknesses related to cybersecurity [4,5], mainly to the connection of traditional industrial devices, which are not usually updated with the passing of the years, and this is paired with the limitations that they have in terms of capabilities. As a result, these devices are the perfect target for attackers, who can gain access to them and cause huge losses to companies. This is illustrated by the attacks recollected and detailed in [6,7]. As this access is sometimes to critical infrastructure, the damage caused by these attackers can lead to costs for both companies and states. Also, this problem can be greater if the convergence between Information Technologies (IT) and OT is not well integrated, as this leads to attackers having new ways of gaining access. Thus, emerging technologies can introduce new risks into traditional factories if the implementation does not follow a guide of good practices. Moreover, attackers could use the security flaws found in MEC in many ways, since the virtual machines that are run on edge servers may include vulnerabilities of the hypervisor or virtual operating system. The user devices that connect to the edge applications could have some software vulnerabilities that can cause their services to be denied, or even manipulate the application to gain access to the MEC infrastructure [8].

A threat detector is a type of security software designed to identify potential security threats and vulnerabilities in a computer system or network. It works by analysing different types of data and events on the system or network. These include network traffic, system logs, and user behaviour. Through the use of algorithms and heuristics, it identifies patterns and anomalies that may be indicative of a security threat. Once a potential threat has been detected, the threat detector can take a number of actions to mitigate it. For example, it can block network traffic from a suspicious IP address, quarantine a compromised system, or alert security personnel so that they can investigate further [9].

These attacks have already been carried out on some critical infrastructures. The main example is Stuxnext [10], one of the first pieces of malware focused on OT topologies and devices for the control of nuclear plants. This shows the importance of having control and continuously analysing industrial topologies in order to try to reduce or avoid the associated risks and vulnerabilities. In addition, as is mentioned in [11], Cisco predicted that 50% of the traffic on the Internet in 2023 will correspond to Machine-to-Machine communication protocols. This kind of traffic is the most commonly used in IIoT scenarios, suggesting that the attackers will focus their efforts on finding new vulnerabilities in these protocols and industrial devices exposed to external networks.

In this work, we propose an intelligent threat detector that is based on a ML models and which uses boosted trees algorithms, as these perform well when used on classification problems [12]. A comparison between the different implementations of this kind of algorithm is needed, because its performance and the requirements to run can determinate which model can be implemented on the architecture. This new intelligent threat detector implementation allows the automation of the detection of different attacks over the industrial network. Therefore, this approach has been deeply studied in other works such as [13,14], the development, deployment, and implementation in industrial environments, and the specific requirements found in the topologies and the protocols used. Our implementation is possible thanks to the network data generated by the industrial topology designed, this being made up of an IIoT topology with the OT protocols used on the network layer, and a MEC topology that provides some services to the IIoT topology [15]. Also, to solve the problems generated by the attackers on the network, the intelligent threat detector can either be implemented in the scenario on the MEC topology or, depending on the computational resources of the algorithms, on every IIoT device. In accordance with the above, the contributions of this work are:

- **Design a IIoT-MEC architecture.** To integrate the new intelligent threat detector service into the industrial topologies, it is necessary to design a new IIoT-MEC architecture to abstract the integration and facilitate the use and deployment of this service.
- **Definition of an IIoT-MEC scenario and the related attacks.** This objective follows the modelling and deployment of an experimentation scenario to obtain useful network data for the training of the models. The implementation of different kinds of attacks is needed to train our intelligent threat detector and perform the classification of malicious packets.
- **Elaboration of a custom-built industrial dataset.** As in IIoT topologies most traffic is generated by OT devices, it is necessary to define and create a dataset with this characteristic, because the main public datasets used in this kind of study are not focused on IIoT environments, and they are generally made with IoT traffic.
- **Performance analysis of the presented algorithms in threat detection.** Having accomplished the other objectives, it is necessary to analyse the results of the metrics obtained with the validation of the ML models. These results show which algorithm works better with the classification of benign and malicious packets.

The rest of this paper is organised as follows. In Section 2, an analysis is carried out of previous studies on ML-based intelligent threat detectors and datasets used for training and tests. Section 3 introduces the technical background associated with the study in this paper. Section 4 explains the modules considered in the architecture proposed in this paper. Section 6 describes the scenario deployed for study, the applications for each OT protocol, and the attacks implemented on the topology. In Section 7, the experimentation is described from the data extraction up to the model validation for each algorithm. Furthermore, every feature of the dataset is defined together with a description of every step and metric used for the study. At the end of this section, the results are shown. Finally, the main conclusions and future work are presented in Sections 8 and 9, respectively.

2. Related work

In IIoT, adding a cybersecurity system means complex and challenging tasks for the industrial network administrator. The use of an intelligent threat detector based on ML techniques like the proposed in this work or the related work can be run like a service in the industrial topology or even like functionality on each device. In addition, an intelligent threat detector must provide a good performance and avoid errors in the specific industrial scenario. There are some recent related studies with these solutions for IIoT and IoT scenarios.

In [16], a study is performed using five IoT datasets, two of them being well-known ones for the benchmarking of this kind of implementation (NSL-KDD and DS2OS), while the rest are provided from Kaggle competitions (IoTDevNet, IoTID20 and IoTBot20). These datasets are used to benchmark different ML and DL techniques, with the ML ones being Decision Tree (DT), Random Forest (RF), and Support Vector Machine (SVM), and DL techniques being Deep Neural Network (DNN), Deep Belief Network (DBN), Long Short-Term Memory (LSTM), Stacked LSTM and Bi-LSTM. The main goal of this study is to analyse the performance of each technique under the same conditions and obtain the best-proposed model for intelligent threat detector implementation in IoT scenarios. This analysis shows that heavy ML techniques are incompatible with low-storage IoT devices as they are not able to run this kind of implementation, a problem that can be solved using edge computing technologies. On the other hand, the best DL technique, with a performance of around 99% for each dataset, is Bi-LSTM, which is a technique compatible with IoT devices. However, the authors cannot guarantee that the model will work properly with a large amount of data or another adverse situation in the scenario.

In [17], the authors present a testbed for the deployment of an Industrial Control System (ICS) water infrastructure to use for data extraction and research. This test consists in using typical OT hardware interconnected with a simulator that introduces different events to the testbed, such as abnormal water waste or parameter variation. Once the testbed is deployed, the authors extract and process the network data generated and used them to implement an intelligent threat detector applying supervised ML techniques, which are LR, Naive Bayes (NB), K-Nearest Neighbour (KNN), SVM, DT and RF. The attacks implemented in the scenario are ping scanner, Man In The Middle (MITM) using ARP packets, and HTTP web services fuzzing. The results show a good performance using the datasets prepared. However, the authors do not detail how the attacks are distributed over them. The performance of the models are 97.44%, 97.47%, 97.20%, 97.43% and 97,42% for LR, KNN, NB, RF, SVM and ANN, respectively. However, the results returned by the DT model show overfitting with the training data, forcing the authors to test the proposed DT model with different parameters, obtaining a performance of 96.5% for accuracy.

In [18], an intelligent threat detector implementation for a water critical infrastructure is presented. The intelligent threat detector is implemented using ML techniques for packet classification, using six of these kinds of algorithms to establish which has the best performance for this context. The ML algorithms selected for the experimentation are LR, Linear Discriminant Analysis (LDA), KNN, Classification And Regression Tree (CART), NB and SVM, while the dataset used is detailed in [19], and covers fifteen different anomaly scenarios in this context for the training and testing of the models. In the experimentation phase, six different situations are presented, evaluating the behaviour of the models in each one. The results show that CART has the most consistent performance in each situation, with a result of 94% for accuracy in the best case and 81% in the worst. The rest of the models have problems in terms of performance in situations in which more than one attack is happening at the same time, reducing the performance to 31% in the worst case for the NB model.

In [20], the authors define a new intelligent threat detector model using multiple layers to encode the network data and establish a decision engine, depending on the packet inserted as input into the intelligent threat detector network. Each layer corresponds to a different step until the decision is made. The first step in the intelligent threat detector model is the preprocessing stage of the network data, and then the features extracted are provided to an LSTM network that performs the function of feature encoder and reconstruction. After that, in the last step the errors found in the reconstruction are calculated and used as validation data, making it possible to determine that, if the errors are higher than a threshold introduced into the decision engine, the data should be considered abnormal. To validate this implementation, the authors consider two datasets to test the implementation of this intelligent threat detector model in an IoT and IIoT context. These datasets are from a gas pipeline dataset defined in [21] and the well-known dataset UNSW-NB15. The performance achieved in terms of the accuracy rate is 97.95% and 97.62% for the gas pipeline dataset and NSW-NB15, respectively.

In [22], the authors present a custom-built intelligent threat detector that is based on the DL model and which uses a novel Sparse Evolutionary Training (SET) prediction model, and a modified weights evolution behaviour as reference, to make a lightweight version for the IIoT context. To validate the performance of their proposal, the authors make a comparison with another DL technique called Fully Connected MultiLayer Perceptron (FC-MLP), looking to achieve better results for precision, recall, F1 score and accuracy metrics. In addition, the number of connections between layers in the DL models is considered an important metric to obtain a lightweight model. The results show that the proposed SET model needs fewer connections than FC-MLP and obtains better metrics, achieving a rate of 98%, 97%, 95%, and 95% for accuracy, recall, precision and F1 score metrics, respectively, in comparison with the DT, SVM, RF, and ANN models with validation results around of 86% for the same metric for the DS2OS datasets. The same metrics improve when the CICIDS2017 dataset is used, with a performance of 99% for each metric considered.

In [23], the authors present an intelligent threat detector based on Blockchain that is designed to keep the network data that come from the different devices of an IIoT scenario private and secure. Also, the intelligent threat detector is developed using an ML technique, namely KNN, to implement a supervised classification model. The dataset used to train and validate the proposed intelligent threat detector is the UNSWNB15 dataset [24], which contains packets from real IIoT scenarios, including different attacks such as fuzzers, backdoors, Denial of Service (DoS), exploit attacks, worms and shellcodes. In the experimentation stage of the paper, the authors do not make a comparison with other proposals or models, and the results are shown with the accuracy rate obtained for each class of packet and attack. Therefore, it is possible to summarise that the mean accuracy rate of the proposed model is 76.35%. Furthermore, the Blockchain functionality shows that the algorithm designed and implemented by the authors does not result in high latency for the different situations included in the experimentation. This leads to the conclusion that the solution proposed for the Blockchain algorithm brings many advantages for privacy and data security without resulting in a high computational cost and avoiding latency in communications even when the data rate introduced to the intelligent threat detector node is higher than in normal situations.

The authors of [25] present a transfer learning based trajectory anomaly detection (TLTAD) strategy. It is focused on Maritime Transportation Systems (MTS) and IoT environments. The system is implemented in two modules, where the first one performs a pre-processing of the received trajectory data, such as the longitude, the latitude, or the speed. In this preprocessing module, a variational autoencoder is used to discover the correlation between trajectory data, and a graph variational autoencoder is used to discover similarity among normal maritime trajectories. Various pre-processing operations to extract the key features for detecting anomalies in sea trajectories are also performed. These features, and the information extracted from the auto-encoders, are used by the second module, which includes a deep reinforcement learning algorithm, namely TD3, with the aim of implementing and using the model of the trajectories anomalies. The results obtained in comparison with the baseline proposed by other ML techniques show that TLTAD achieves a performance of 96.1%, 95.5%, 94.6%, 95% for the metrics of accuracy, precision, recall, and F1 score. TLTAD improves the performance of DT, iForest, SVM, LSTM, Variational Autoencoder (VAE), and Abnormal AIS Data Screening (AAISS) techniques between 3% and 17% in the metrics previously detailed for the data set of the Whuhan-Shanghai section of the Yangtze River.

Furthermore, the secure data aggregation for the new ML applications deployed in edge environments, which is an interesting topic that is not covered in our proposal, is another key point in this area. This feature could be benefited by using blockchain technologies, as in the work presented in [26], which describes a blockchain-based system for securing IoT data aggregation in the edge computing Layer. This system carries out three important processes: the construction of the block header with a security layer, the partitioning of the end devices receivers, and the partitioning of the most sensitive tasks with a focus on the prevention of privacy disclosure. Under the control of the security layer, the system also considers the implementation of energy-efficient data aggregation during routing generation using the improved self-adaptive double bootstrapped deep deterministic policy gradient (IDDDPG). The results have shown that this proposal achieves a low transaction latency and a high throughput as a strategy to counter collusion attacks. In addition, the proposal achieves a higher aggregation ratio with a lower energy cost when compared with other contemporary data aggregation strategies.

Table 1
Summary of the proposals from the research community.

Reference	Algorithms	Dataset Used	Highlights
[16]	DT, RF, SVM, DNN, DBN, LSTM, Stacked LSTM and Bi-LSTM	NSL-KDD, DS2OS, IoTDevNet, IoTID20,IoTBot20	Test 3 new IoT-oriented datasets, and the best model proposed for the analysis obtains an accuracy of 99% on each dataset.
[17]	LR, NB, KNN, SVM, DT, RF and ANN	Custom-built dataset with Modbus/TCP and S7 protocols	Obtains a general performance for the models of 97%.
[18]	LR, LDA, KNN, CART, NB and SVM	Water Custom Dataset [19]	CART proposal model provides the best performance for most of the situations and attacks implemented on the dataset.
[20]	Author proposed DL model based on LTSM	Gas pipeline [21] and UNSWNB-15	The proposed model returns a performance of 98% in terms of accuracy rate.
[22]	Author proposed SET-based model, FC-MLP	DS2OS, CICIDS2017	The work presents a custom-built DL model for IIoT scenarios, with an average performance of 99%.
[23]	KNN	UNSWNB15	Provides an intelligent threat detector solution with ML and Blockchain technologies working together to direct identifying and mitigating processes and methods against attackers.
[25]	DT, iForest, SVM, LSTM, VAE, AAISS and TLTAD proposed	AIS dataset of Wuhan-Shanghai	The proposed TLTAD system shows the best performance compared with the baseline techniques, obtaining a 95% in the F1-Score metric.
[26]	IDDPG for energy-efficiency design routes to end-devices	None	The present blockchain-based strategy in this work achieves a low latency and a high throughput when compared with the current strategies for the aggregation of data.
Our work	GradientBoosting, AdaBoost, XGBoost, Catboost, LightGBM	Custom-built dataset with Modbus/TCP, OPC UA and S7COMM protocols	XGBoost model achieves a performance of 99% in a situation with multiple simultaneous malicious attacks.

As a summary, Table 1 shows the algorithms, datasets, and highlights of the different pieces of research reviewed in this section. Finally, in our work, which is also shown in Table 1, we present an analysis of the different boosted trees algorithms for the implementation of an intelligent threat detector for MEC-IIoT scenarios, studying the multiple situations and possibilities for them. Moreover, the custom-built dataset generated for this study contains multiple OT protocols with some cyberattacks that can be present in a MEC-IIoT topology.

3. Background

This section introduces the fundamental concepts of MEC, OT protocols and Machine Learning algorithms, which define the context in which the proposal is developed.

3.1. Multi-access Edge Computing

MEC is considered the natural development of the edge computing concept, and it has been standardised by the European Telecommunications Standards Institute (ETSI) [27]. This evolution aims to improve the base function of edge computing by bringing the computational and networking capabilities provided by the cloud computing environment closer to the end users, thus obtaining better results in latency, reducing the bandwidth saturation of the cloud providers and the deployment of new services and applications for companies. In addition, MEC brings new advantages to the IoT and IIoT context [28] by providing support for new network technologies such as 5G, and the management of the heterogeneous network traffic found in this context, as well as enabling the use of mobility for applications and users. To provide all this, MEC uses four virtualisation technologies.

The first one is Network Function Virtualisation (NFV) [29], which provides support for the unified management of the heterogeneous traffic that is present in the IIoT context. Secondly, Software Defined Networking (SDN) [30] facilitates the definition, administration, and control of networks, especially when these grow in complexity and the number of devices connected. Finally, Network Slicing (NS) and Service Function Chaining (SFC) are implemented in MEC to allow the slicing of network resources into multiple virtual ones, depending on different parameters such as traffic type or application destination, among others. SFC facilitates the transition from the original physical network to the new virtual one defined and deployed [27].

3.2. OT protocols

The OT protocols which are considered for implementation in this work are:

Modbus/TCP [31]. This is a specification from the serial Modbus protocol used by industrial devices, in which Modbus packets are embedded in TCP segments and the port assigned is the number 502. Also, the specification maintains the compatibility with serial Modbus with the upper limitation of the payloads being 253 bytes. The schema of communication used by Modbus protocols is the master–slave approach.

S7 communication. This is a proprietary communication protocol developed by Siemens for its industrial devices in 1995 and includes the next updates of the protocol introduced in 2009 and 2012 for the new generation of industrial devices [32]. The S7 protocol permits the transfer of critical configuration and operational information, configuration details, data blocks with data which can be interpreted by a Programmable Logic Controller (PLC) and diagnostic information [33].

Open Platform Communications Unified Architecture (OPC UA) protocol [34]. OPC UA replaces the traditional OPC standards, it is compatible with the traditional client–server communication approach for information management and access. In these OPC UA applications, only devices that use the OPC UA protocol can obtain information from other OPC UA servers, because these servers are configured with a machine control module from the protocol to retrieve information and send control signals.

3.3. Machine Learning algorithms

In this work, we consider the Ensemble Learning algorithms, specifically boosted trees algorithms because , as it shows in [35,36], this type of algorithm is recommended for the classification of tabular data. Another relevant aspect to be considered in this experiment is that the efficiency of these algorithms when used in the IIoT-MEC context is yet-to-be confirmed due to the novelty of the environment. The fundamental principle behind the boosting technique is to train sequential weak models. Each model focuses on classifying categories that previous models misclassified. This allows each new weak model to be trained considering the errors of the previous ones. These weak models are grouped together to obtain a more robust classifier that will perform better than the weak ones would do separately [37]. The boosted trees algorithms considered in our study are:

Gradient tree boosting (GTD) algorithm [38]. GTD algorithms integrate multiple models based on decision trees models. As a core, these algorithms use the negative gradient of the loss function as a possible approximation in the generation of the decision trees algorithm and gradually reduce the loss function. The complexity of the algorithm [39] is given in Eq. (1) and (2), where M is the number of trees generated, d is the depth of the tree, and n is the size of the dataset used.

$$GTDTrainingComplexity = O(M * d * n * log(n)) \tag{1}$$

$$GTDPredictionComplexity = O(M * log(n)) \tag{2}$$

XGBoosting algorithm [40]. The XGBoost algorithm is an extension of the GTD implementation that is designed to avoid the limitations found in the original algorithm, with the principal aim of providing the best performance and computational speed. The implementation of this extension algorithm is developed as an open programming package. The algorithm complexity [41] for the training and the predition steps are shown in Eq. (3) and (4), where t is the number of the trees generated, d is the height of the tree, and x means the number of non-missing data.

$$XgboostTrainingComplexity = O(t * d * x * log(n)) \tag{3}$$

$$XgboostPredictionComplexity = O(t * d) \tag{4}$$

AdaBoost algorithm [42]. This algorithm was one of the first practical boosting algorithms, with many applications in different fields. Over the years, AdaBoost was established as a learning algorithm for classification and regression purposes. The complexity of the Adaboost algorithm [43] is given in Eq. (5) and (6), where n is the size of the dataset, p is the number of features considered, and t is the number of trees generated.

$$AdaboostTrainingComplexity = O(n * p * t) \tag{5}$$

$$AdaboostPredictionComplexity = O(p * t) \tag{6}$$

Catboost algorithm [44]. The Catboost implementation solves the problem of using categorical features that are allocated in many datasets. Catboost handles categorical features and uses them to obtain a better performance during the training stage of the model. Also, it introduces a new schema to calculate the leaf value of the trees as a way to reduce the overfitting of the model trained. The algorithm complexity [45] during the training and prediction is determined by Eq. (7) and (8), where n is the number of features considered during the training stage of the model, T is the number of the trees defined, and s means the number of samples used.

$$CatboostTrainingComplexity = O(T * n * s) \tag{7}$$

$$CatboostPredictionComplexity = O(T * log(n)) \tag{8}$$

LightGBM algorithm [46]. This is an implementation of a GTD algorithm that is designed to have higher training speeds and efficiency. Also, LightGBM is designed to work on low-resource devices because it has lower memory usage than other implementations, better accuracy, and it is capable of handling large-scale data. The estimated complexity of the LightGBM model [47] during the training and prediction phase is presented in Eq. (9), while the prediction complexity is shown in Eq. (10). The parameter it is the number of iterations of the algorithm, T is the number of trees generated, l is the number of leaves of the tree, D is the depth of the tree, and n is the number of features considered.

$$LightGBMTrainingComplexity = O(it * l * max(D)) \tag{9}$$

$$LightGBMPredictionComplexity = O(T * log(l) * n) \tag{10}$$

4. Proposed architecture

Taking into consideration the advantages provided by MEC, mainly the computational resources for the implementation of intelligent threat detectors based on Machine Learning models, we decided to use it as the basis for the proposed IIoT architecture in [48]. This architecture is a three-layer definition of the different parts of an IIoT-MEC scenario, adding a Cloud layer as a way to include the different IoT, Big Data or storage services that could be used as well.

The proposed architecture is shown in Fig. 1, in which a Control Network Node is introduced into the MEC layer for the deployment of the intelligent threat detector. This node receives a mirror of the traffic that flows between the IIoT devices and the different MEC nodes. Thanks to the computational and networking resources that are provided in the MEC topology, it is possible to carry out the processing of the information and the deployment of the ML model for the detection of any intrusion in real-time. In addition, a monitoring and warning service is implemented to use the output of the intelligent threat detector to advise the administrators or to activate defensive measures to avoid possible risks in the topology.

In the IIoT layer, the IIoT devices are connected individually to the Supervisory Control And Data Acquisition (SCADA) node as a way to establish communication with another IIoT device or IIoT server which is in the MEC layer. This SCADA node has the main functionality in our architecture of acquiring network data and storing it in a database. This data is used during the training and testing stages of the ML model implemented in the intelligent threat detector.

5. Threat model

Considering the IDS architecture described in Section 4, our threat detector runs in a machine as an application in the MEC station connected to the IIoT network. The threat detector receives the traffic mirrored by the ICS of the IIoT topology and then analyses and classifies the received packet based on the knowledge extracted after training. If the detector classifies a packet as malicious, it sends an alert to the monitoring application, which is also running in the MEC station and is capable of starting some defensive countermeasures to avoid damages in the IIoT network.

We assume that the node where the IDS is running is not infected with any malware, and that during the collection of normal traffic for training the model, the workload of the devices in the IIoT topology

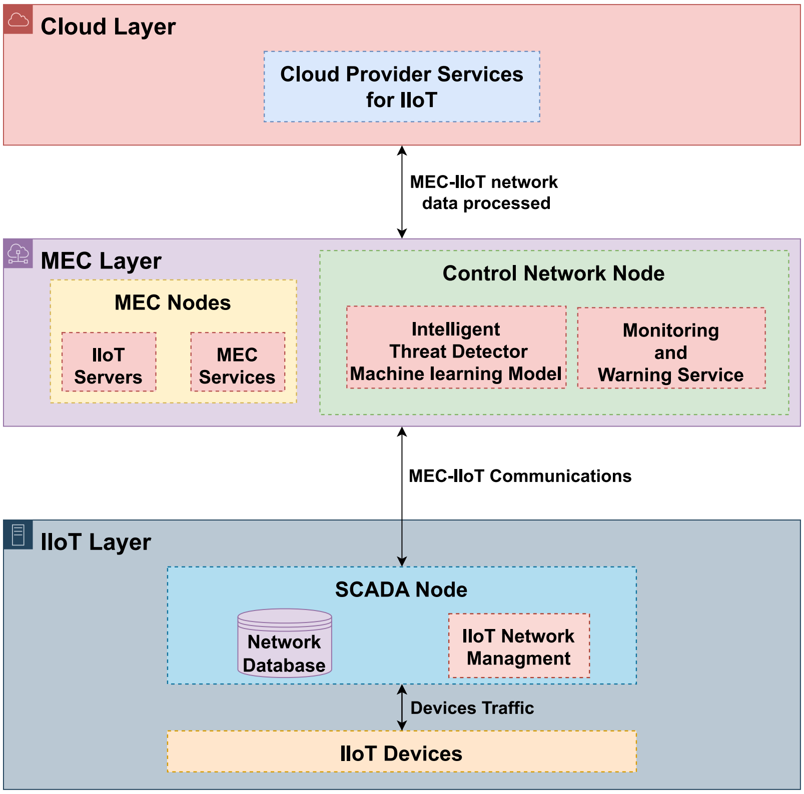


Fig. 1. Proposed IIoT-MEC architecture.

follows a usual behaviour, as well as that these devices are running correctly. The model is trained with attacks that a malicious attacker would use in the IIoT. In addition, during the preprocessing of the data, every packet has been tagged according to the attack from which it comes from.

With regard to the attacker's behaviour, their main goal is to scan the IIoT to get information about the connected devices and alter the normal operation of the industrial devices, manipulate the information exchanged between the devices, stop their service through DoS attacks, and gain access through the web services of the devices. These attacks focus on the network layer and the web application layer. In addition, we assume that the attacker is connected to the IIoT network with a device that has been previously compromised and that the attacks are sent from this device.

The threat detector is limited by only being able to correctly classify those attacks from which it has the knowledge extracted during the training phase, providing incorrect classifications if the attacker carries out an unknown one. Moreover, the IDS is centralised in the MEC layer, so, if there is a disconnection between the two networks, the detector does not work, otherwise the classification could be manipulated by the attacker and make it possible to hide attacks running on the industrial network if they were to gain access to the machine running the detector and infect it.

6. Experimental setup

This section first describes the scenario deployed for the experimentation in this work. Then the implementation of the Modbus/TCP, OPC UA and S7 protocols is explained with regard to the extraction of relevant data. Finally, the implementation of cyberattacks is described.

6.1. Scenario network design

For the design, implementation, and deployment of the IIoT-MEC scenario, container technology has been used. The Docker container engine [49] allows the design of a complete industrial network by considering its network layer, and using the TCP/IP implementation of the typical OT protocols. In addition, openLEON [50] is deployed as well because it is an emulator which deploys a 3-tier data centre, and LTE communication is possible with the appropriate hardware. This emulator allows the deployment of infrastructures similar to the requirements defined for MEC topologies. The use of an emulator allows the reproducibility and rapid prototyping and design of new experiments with different requirements, topologies, and services.

The experimental IIoT scenario deployed for this study is shown in Fig. 2. The IIoT topology is deployed using Docker-Compose, which is a tool developed to facilitate the deployment of multi-container applications. For this scenario, the deployment has two S7 nodes, an OPC UA node, and two Modbus/TCP Nodes. These nodes have been implemented using the programming language Python to generate the traffic for each protocol, and a Docker image for each script to easily deploy the nodes in the IIoT network. The attacker node consists of a custom container image from the Kali Linux container image, which contains the most popular pentesting tools and others made specifically for the experimentation scenario. These custom-built tools made for this study implement the cyberattacks, which are described in Section 6.3. Finally, the SCADA node is the host that runs the Docker-Compose scenario, and it is where the data extraction is performed. In addition, this node sends the corresponding messages to the OT servers allocated in the MEC topology.

The MEC topology is deployed using the openLEON emulator, which employs Containernet [51] and the Ryu [52] controller to provide real functionality to the SDN controller and the edge nodes.

6.2. Scenario devices design

As is shown in Section 6.1, the implementation for each kind of device in accordance with its protocol is as follows:

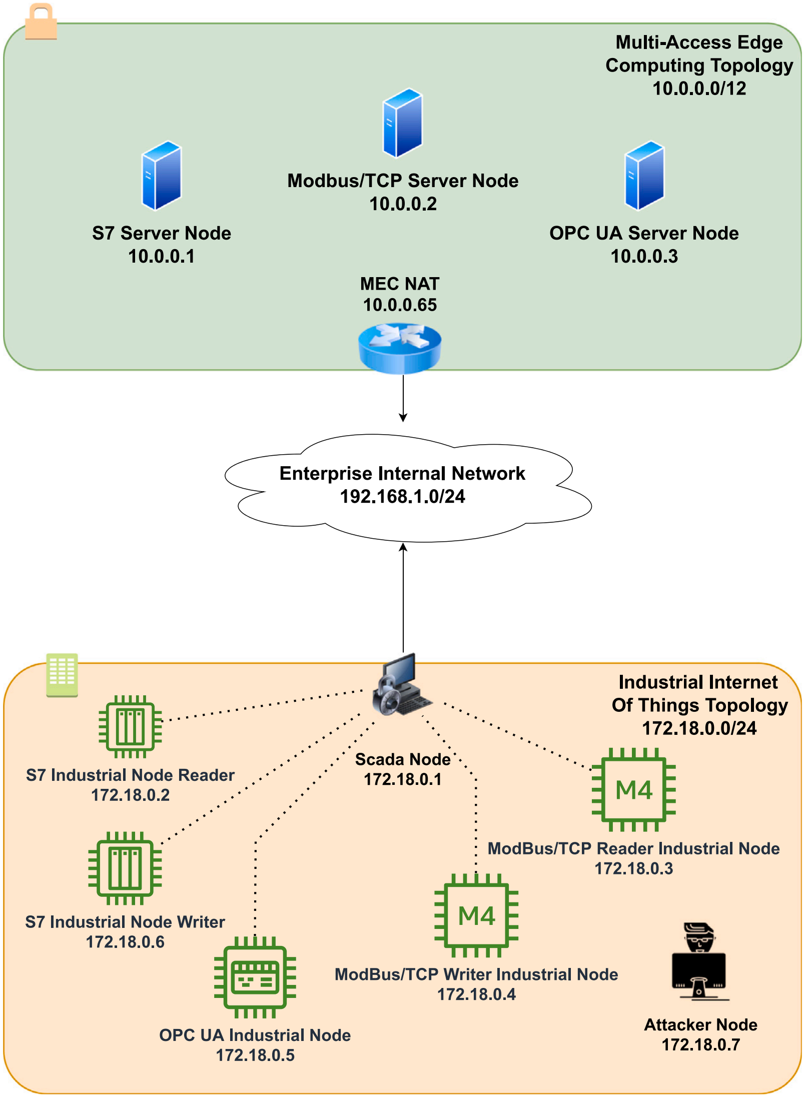

Multi-Access Edge Computing Topology 10.0.0.0/12
Modbus/TCP Server Node 10.0.0.2
S7 Server Node 10.0.0.1
OPC UA Server Node 10.0.0.3
MEC NAT 10.0.0.65
Enterprise Internal Network 192.168.1.0/24
Industrial Internet Of Things Topology 172.18.0.0/24
S7 Industrial Node Reader 172.18.0.2
Scada Node 172.18.0.1
M4
ModBus/TCP Reader Industrial Node 172.18.0.3
S7 Industrial Node Writer 172.18.0.6
M4
ModBus/TCP Writer Industrial Node 172.18.0.4
OPC UA Industrial Node 172.18.0.5
Attacker Node 172.18.0.7

Fig. 2. IIoT experimentation scenario.

- **S7 communication.** The *writer* device sends a random string of twenty non-numerical characters to the S7 Edge server. This string is sent in a period of time between one and seven seconds. Then, the *reader* device tries to obtain the last message stored in the S7 server. To properly emulate an S7 Siemens server, this device has a web server to perform some management task.
- **OPC UA.** OPC UA server contains storage directories to which the OPC UA client can connect in order to retrieve and modify the information that is stored there. This server has only a root route and a subroute, which contains a variable with a string of ten alphabetical characters. The variable changes in periods of between one and nine seconds. The OPC UA node in the industrial topology connects to the endpoint opened by the OPC UA server allocated in the MEC topology and obtains the data of the variable stored.
- **Modbus/TCP.** The Modbus/TCP *reader* node reads four bits allocated on the Modbus/TCP edge server. The values of these bits can only be True, and False. The Modbus/TCP *writer* changes the value of the bits for the opposite. The normal performance of this application is to always obtain all bits with a value of one or with zero, but never mixed in one reading.

As is mentioned in Section 6.1 a SCADA node is present in the IIoT topology. This node has the functionality of monitoring the packets that are sent to the device, or to the MEC topology. The tool used is Wireshark, which allows users to examine the information allocated in the packets. It also enables the extraction and export of the traffic monitor to work with it, and with the data desired.

6.3. Cyberattacks design

Once the scenario has been designed and is ready to be deployed, it is necessary to introduce and implement some cyberattacks to emulate the situation when an attacker is connected to the IIoT topology. This allows the attacker to employ different techniques to access the resources, and information, or to trigger the denial of services in the topology. This implementation is also necessary in order to introduce malicious traffic for the training of intelligent threat detector models. The attacks that are considered in this work are:

- **Packet manipulation attack.** In this kind of attack, the attacker has the option of performing an MITM attack and sniffing the traffic between the devices, modifying the data sent by the devices.

- **Scanning devices.** The normal first step taken by the attackers is scanning the topology in order to find vulnerable devices to attack. The following diverse ways of scanning for this kind of malicious traffic:
  - **TCP connect scan:** This is the default scan carried out by the tools. The attacker sends an SYN request to the target and closes the connection when the target establishes it.
  - **TCP SYN scan:** The attackers send an SYN request and kill the communication as soon as they receive the SYN packet from the target.
  - **TCP NULL/XMAS/FIN scan:** In this kind of scanning, the attacker sends a TCP packet to the target, changing the flags of the packet. Depending on the scan type, the flags of the packet vary. Regardless of what scan technique is used, if there is not any response from the target, the port is considered filtered or open.
  - **UDP scan:** UDP packets are sent to the device ports during this scan. If there is a UDP response from the application, the port is considered to be open, but if it takes more than a certain amount of time to receive a response, or if there is no response, the port is considered to be filtered.
- **DoS.** This is a typical practice for attackers to interrupt the correct operation of the devices and services in a factory. The attacks implemented with the Scapy library with this goal are:
  - **TCP flood:** These attacks consist in sending a huge number of malicious packets to saturate the interfaces of the target.
- **HTTP attacks.** Since S7 devices contain a web application to manage different tasks and monitor parameters. Attacks related to HTTP services are implemented in the scenario, with them being the following are:
  - **Login Brute Force:** This attack uses a dictionary of users and passwords to make multiple login attempts to try to access devices and services.
  - **ShellShock:** This vulnerability has been found in several bash-implemented web applications, some of them in the administration web application of various Siemens industrial products. This vulnerability allows the attackers the injection of commands in the device using malicious payloads using bash commands.

6.4. Hardware configuration

For the deployment of the scenario described in Section 6.1 a laptop is used, which is equipped with an Intel i7-10875H 2.30 GHz CPU and 32 GB of RAM memory, running Windows 10 21H2. To deploy both parts of our MEC-IIoT, two virtual machines are launched. The first contains the openLEON emulator and IIoT servers to deploy the MEC topology, and the second deploys the IIoT topology using Docker Compose. The VirtualBox 6.1.16r hypervisor is used to run the virtual machines. Finally, both virtual machines are deployed in bridged mode, using the IP address that is provided by the router of the private network.

7. Experimentation and results

This section explains the extraction of the network traffic data from the scenario described in Section 6.1. Then, the preprocessing stage corresponding to the cleaning and adequacy of the network data that is used for the training and testing stages of the model construction stage is explained. In addition, the results obtained from the different classification models are presented and explained, paying special attention to the IIoT-MEC context and considering the different options for implementation. Finally, this section explains how the pre-processed data is adapted to be used in a second experiment to test whether the algorithm can detect anomalies in the MEC-IIoT network.

7.1. Experimentation

In this study, a custom dataset is used to train the models. Firstly, with no attacks running on the network, it is necessary to collect the data from our scenario. Once the collection is completed, the attacks described in Section 6.3 are launched and its traffic is acquired in different phases in order to capture the malicious traffic and enable experimentation. After that, a preprocessing stage is carried out to adapt the data correctly so that they can be used in the training stage of the different models. The most representative features are also extracted, using a feature selection process for the detection of attacks. When the dataset is ready, it is necessary to select the ML models to be used and train them with the data. Once every model is trained, the last step is to validate each one using different metrics to determine which one has better performance for each situation. The process followed during the experimentation is shown in Fig. 3.

7.1.1. Network deployment

The scenario described in Section 6.1 is deployed for the study. To do this, it is necessary to deploy the OT services on one virtual machine with the Docker-Compose tool. Also, for the MEC topology deployment, in which the OT servers are allocated, it is necessary to initiate another virtual machine with the implementation of openLEON [50].

7.1.2. Data extraction

When the scenario is deployed correctly, on the SCADA node Wireshark is run and configured to extract and export data in a format compatible with the network data. This data contains the initial set of features that are considered at the beginning of the training of the models. The initial features extracted are shown in Table 2, these being features are the common to the OPC UA, S7 and Modbus/TCP protocols, while in Tables 3 and 4, the features described are specific to each protocol.

7.1.3. Data preprocessing

The data extracted in the previous phases is adapted so that it can be used by the algorithms. Moreover, it is necessary to determine which techniques should be applied to the data and which features are more relevant to use. Also, some features are combined and reconsidered to earn more importance during the training of the models.

This stage is divided into the following steps:

Tagging data. It is necessary first to give the correct tag to each packet depending on whether it is an attack packet or normal traffic. The tags used for the study are detailed in Table 5.

Data cleaning. The extracted data have some features with empty values, usually due to an error during their collection or because a packet does not have this information. For example, the features that are related to a specific protocol. In this study, the value −1 is used for the missing features if these are numerical, and an empty string is used for categorical features. This also serves as a method for indicating the model that it is not a relevant feature.

We establish these criteria because some features are directly related to a specific protocol. For that reason, deleting missing data on a feature is not possible.

Data transformation. ML algorithms are not able to directly process alphanumeric data. Therefore, the features that are categorical need to be transformed. To do so, the binarization technique is used to associate the categories to a specific number value.

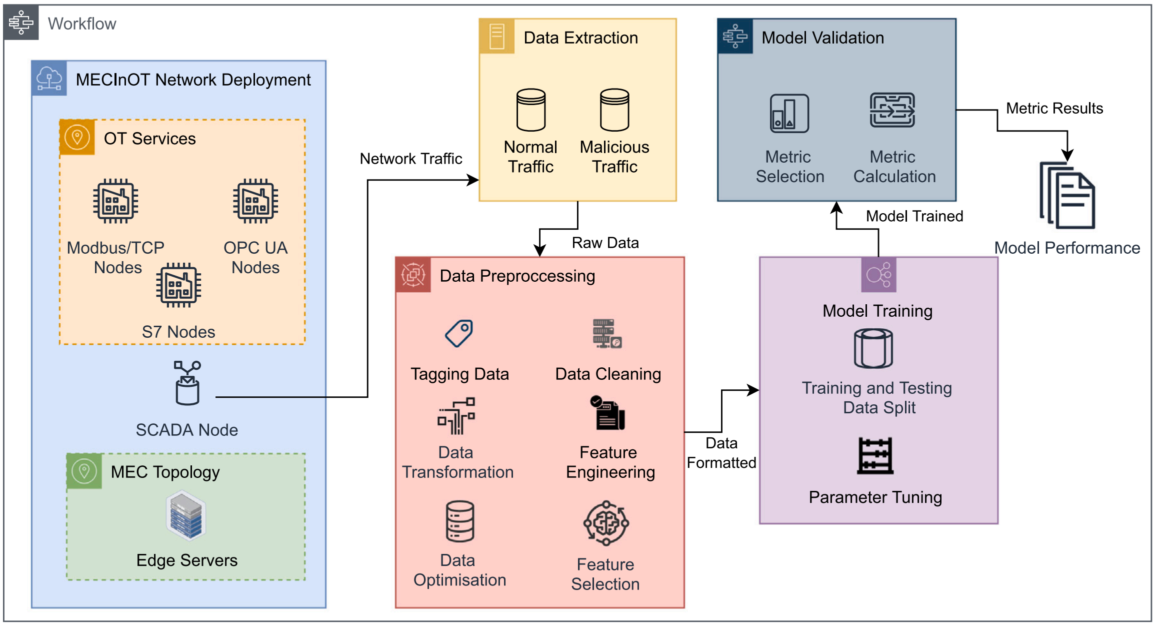


Fig. 3. Workflow of the experimentation.

Table 2
Common features of TCP/IP protocols.

Feature	Description	Type
Packet number	Indicates the number of a packet. It is used as index in the dataset.	Numerical
Time	Represents the specific second in which the packet was captured during the extraction.	Numerical
Source	IP host source of the packet.	Categorical
Destination	IP host destination of the packet.	Categorical
Protocol	Protocol that transports the information of the packet.	Categorical
srcPort	Source port used to send the packet.	Numerical
dstPort	Destination port used to receive the packet.	Numerical
Length	Total length in bytes of the packet.	Numerical
Info	Additional information of the data that transport the packet captured.	Categorical
tcp_Flags	Indicates the states of the flags of the TCP protocol in hexadecimal encode.	Categorical
tcp_WindowSize	Indicates the size of the window of the TCP protocol.	Numerical
Delay	Indicates in seconds, the difference in time between two packets of the same communication.	Numerical

Table 3
Specific features of OPC UA and S7 protocols.

Feature	Description	Type
OPC_Message_Size	Message length of the OPC UA protocol.	Numerical
OPC_Sec.Requ_ID	Indicates the ID of the OPC UA communication that responds.	Numerical
MessageType	Type of message indicated in the OPC UA protocol.	Categorical
Message_String	Content of the message transported in the OPC UA protocol.	Categorical
S7_ROSCTR	Kind of data that the S7 protocol transports.	Categorical
S7_Data_Length	Size of the S7 protocol data.	Numerical
S7_Function	Function performed by the S7 packet.	Categorical
S7_Message	S7 data transported in the packet encoded in base64.	Categorical
S7_Error_Code	Error code from S7 protocol communication.	Numerical

Feature engineering. Once this data is reviewed, outliers and extraneous values are removed to avoid impairing model performance. It is possible to see correlations between features or even typical behaviour in the data. In the latter case, this behaviour can generate new features of greater importance than the base feature. These new features only consider the values of -1 if the base value is null, 0 when the feature indicates that some modification or anomalous values are present, and 1 when the value in the base feature is normal according to the habitual functionality in the scenario. Table 6 shows the features generated during this step.

Data optimisation. After performing the data preprocessing, the dataframes must be reviewed, and any wrong data types should be modified in order to use the memory properly and improve the training stage of the model.

Feature selection. Not every feature considered in the dataset has the same importance during the model training. Some of these features could even lower the performance of the model prediction. In this regard, it is necessary to select the most influential feature during prediction.

Table 4
Specific features of Modbus/TCP and HTTP protocols.

Feature	Description	Type
Mod_Length	Indicates the length of the Modbus frame.	Numerical
Mod_Transa_ID	Shows the Modbus transaction id.	Numerical
Mod_Function	Indicates Modbus action at destination.	Categorical
Mod_Requ_Frame	Number of Modbus packet that corresponds to the request.	Numerical
Mod_Req_Delay	Time between Modbus requests.	Numerical
Mod_Data	Modbus message transported.	Categorical
Http_user	Shows the content of User-Agent from HTTP frame.	Categorical
Http_data	Shows the data transported in the HTTP frame.	Categorical

Table 5
Categories and the tags associated.

Category	Tag(s)
Normal traffic	final_clean
HTTP attacks	brute_http, payload_user_agent
DoS attacks	ping_of_death_dos, tcp_flood_dos
Scanner	scanner_ack, scanner_fin, scanner_tcp, scanner_udp, scanner_xmas, scanner_null
Manipulation attack	manipulation

Table 6
Features created from the feature engineering step.

Feature	Description	Type
type_packet	Tag to identify whether the packet is correct or associated with an attack.	Categorical
Message_String_normal	Indicates whether the message sent is in accordance with the normal functionality of the OPC UA service or not.	Numerical
S7_Message_normal	Indicates whether the message sent is in accordance with the normal functionality of the S7 service or not.	Numerical
Mod_Data_normal	Indicates whether the message sent is in accordance with the normal functionality o thef ModBus/TCP service or not.	Numerical
Strange_user	Indicates whether the User-Agent parameter from the HTTP frame corresponds to a typical value.	Numerical
Normal_ping	Indicates whether the length of ping is higher than 98 bytes.	Numerical

7.1.4. Model training

Once the data in our study is ready to be used with the algorithms, namely XGBoost, AdaBoost, GradientBoost, LightGBM and Catboost, we distribute them into different dataframes. These dataframes are prepared to store benign traffic and one type of attack or multiple attacks.

Also, for model training it is necessary to follow certain steps to improve its performance:

Training and test split dataframes. To carry out the correct training and validation of the model, the dataframe is divided using a ratio of 70% for the training set and 30% for the test set.

Parameter tuning. The parameters of the algorithms can be modified to obtain better performance during the training. Because of the size of the dataset, to obtain the best parameters for each case, RandomSearchGrid is used. This technique search multiple times with a random starting point, allowing to find a global optimum [53]. Once the best parameters are obtained, they are used during the training of the model.

7.1.5. Model validation

When the training stage is finished with the training data set, and the features have been selected, and the best parameters have been found, it is time to validate the predictions of the final model and see whether it correctly predicts the testing dataset. For the validation stage, there are several metrics to consider for the evaluation of the model. All of these use the information in the confusion matrix, which is made up of the number of True Positive (TP), True Negative (TN), False Negative (FN) and False Positive (FP), in accordance with the behaviour of the classification performed by the model for each packet. The metrics are the following:

Accuracy. This metric refers to the number of correct predictions against all those made by the model. It is calculated as shown in Eq. (11). This metric must be studied carefully when the dataset is unbalanced in terms of categories, as a bad model will tend to predict the same category over and over again, and still achieve a good value due to one category being significantly higher than the rest. Knowing that traffic data are normally unbalanced, it is necessary to use other metrics to perform a better evaluation.

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{11}$$

Precision. This metric represents the ratio of the correct positive predictions against the total ones considered correct by the model. Eq. (12) shows how to calculate this metric. A high score in this metric means that the intelligent threat detector does not mismatch benign packets with malicious ones.

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

Recall. This represents the ratio of predictions correctly considered positive against all of those considered positive. This metric is calculated with Eq. (13). A high recall score means that the intelligent threat detector detects a high number of attacks that belong to the kind that it is trying to detect.

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

F1 score. This metric is considered the harmonic mean of the precision and recall metrics, and the accuracy metric for models trained with imbalanced datasets. This metric is normally used to validate models trained with imbalanced datasets because the F1 score gives importance to the ratio of correct predictions and the detection of anomalous classes during classification. The formula for this metric is given by Eq. (14).

$$F1\_Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{14}$$

Cross validation score. This technique makes a partition of the data into subsets of the same size. The model is validated with one of these subsets and trained with the rest, repeating this process for each subset. The technique ends with a summary of the accuracy score obtained in the executions. This procedure is used to estimate the behaviour of the model with testing data and new data that could the model receive and avoid overfitting.

Table 7
Results obtained for the metrics from the algorithms selected.

Attack	Algorithm	Precision	Recall	F1 score	Accuracy	Cross validation score	Training-Test time (s)
Packet manipulation	XGBoost	0.94	0.96	0.95	0.9801	0.97568	0,10867
	LightGBM	0.96	0.96	0.96	0.9835	**0.97696**	**0,03390**
	AdaBoost	0.96	0.96	0.96	**0.9844**	0.97685	3,74168
	CatBoost	0.95	0.96	0.96	0.9827	0.97568	1,26268
	GradientBoost	**0.98**	**0.98**	**0.98**	0.9833	0.97642	0.90929
Scanning	XGBoost	0.92	**0.99**	**0.94**	0.9992	0.99954	53,80689
	LightGBM	0.91	0.96	0.92	0.9990	0.99938	**1,60220**
	AdaBoost	0.92	0.98	**0.94**	0.9991	0.9995	199,25673
	CatBoost	0.91	0.98	**0.94**	0.9991	0.99949	24,04382
	GradientBoost	**1**	0.89	0.91	**0.9994**	**0.99964**	105,50021
DoS	XGBoost	1	0.97	0.98	0.9995	**0.99983**	0,39597
	LightGBM	1	0.95	0.97	0.9991	0.99959	**0,08143**
	AdaBoost	1	0.97	0.98	0.9995	**0.99983**	6,11796
	CatBoost	1	0.97	0.98	0.9995	**0.99983**	1,24134
	GradientBoost	1	0.97	0.98	0.9995	0.99971	2,06225
HTTP application	XGBoost	1	1	1	1	1	0,25039
	LightGBM	1	1	1	1	1	**0,04687**
	AdaBoost	1	1	1	1	1	0,40935
	CatBoost	1	1	1	1	1	0,09632
	GradientBoost	1	1	1	1	1	0,20163
Mixed	XGBoost	**0.99**	**0.99**	**0.99**	**0.9991**	**0.99935**	266,60198
	LightGBM	0.98	0.98	0.98	0.9990	0.99925	**5,34816**
	AdaBoost	0.82	0.82	0.82	0.9954	0.9953	442,41913
	CatBoost	0.98	0.95	0.96	0.9988	0.99904	108,84218
	GradientBoost	**0.99**	0.93	0.95	0.9980	0.99893	1144,95351

Training-test time. This metric shows how much time in seconds is taken by each model to carry out the training phase and to make the prediction during the test phase. The metric shows the complexity of the algorithm of each technique of the analysis and how this affects in the performance with the dataset that is used.

7.1.6. Results

The Table 7 shows the results at the end of the model validation for each metric and algorithm used and for each attack to be detected by the classification model. The analysis is divided into the different attacks implemented and introduced in the dataset to obtain the performance for each situation and the algorithm complexity of each model to find the one with the best complexity/performance ratio.

Packet manipulation attack. It is possible to see that the best algorithm for this kind of attack is the GradientBoost algorithm, which achieves 98% for the Precision, Recall and F1 score metrics. Thus showing a high rate of detection of the manipulated packet in the scenario, with only 2% of missed packets, while the rest of the algorithms obtain a percentage of 94%–96%. This makes it possible to have an efficient intelligent threat detector for monitoring and advising when this attack is occurring in the scenario. LightGBM has the better metric in the cross-validation score, showing good behaviour against overfitting and the reception of new data, although, the difference with GradientBoost for the same metric is not significant. The worst performance is returned by the XGBoost algorithm, with 94% for the Precision and 95% for the F1 score, which represents a difference of 4% and 3%, respectively, with regard to the GradientBoost algorithm. However, this difference between the metrics is not so significant as to prevent the use of the XGBoost model in the implementation of the intelligent threat detector because the computational cost of XGBoost is lower than that of GradientBoost.

Scanner attacks. In general, the results for Precision are between 91% and 92%, except for GradientBoost, which returns a result of 100% for this metric. This could be because it is too complicated for the models to differentiate a TCP scanner packet from a UDP scanner packet. Thus, this makes the model generally predict that a UDP scanner packet is a TCP scanner, returning, as a result, a wrong classification. The Recall metric results, in comparison with the Precision metric, are very different. GradientBoost obtains the lowest rate at 89%, and XGBoost achieves the highest at 99%. This means that GradientBoost normally detects correctly, with a lower rate than the other models if the packet received is normal or a scanner packet, regardless of whether the classification of the scanner type is correct or not.

DoS attacks. For this attack, there are not many significant differences between the algorithms. In fact, XGBoost, AdaBoost, Catboost and GradientBoost obtain the same results for the Precision, Recall, F1 Score, Accuracy and Cross-Validation Score metrics, with 100%, 97%, 98%, 99.95% and 99.98%, respectively. Furthermore, LightGBM, which has a Recall metric result of 95%, does not have a significantly worse performance than the rest of the algorithms, keeping in mind that this algorithm has lower computational needs on the different devices in an IIoT or edge environment.

HTTP application attacks. The results for the HTTP attacks implemented in the scenario show that all the algorithms achieve a 100% for all the metrics. The reason for these results is that the attacks identified, and the parameters considered for the models make it easy to detect the attacks, thanks to the information inside the HTTP frames for payload attacks and the time delay between packets for brute force attacks. Therefore, the selection of the algorithm should consider another metric, such as the resources needed to run the model or the training time of the models, depending on the needs of each scenario in which the intelligent threat detector will be implemented.

Mixed attacks. This category of brings together the attacks analysed above in order to study the behaviour of the models when multiple types of attacks are occurring at the same time. This situation is probably the most common in a real scenario because factories are often being attacked by different malicious hackers at the same time, with these introducing different kinds of malicious packets into the network. Firstly, it should be highlighted that AdaBoost obtains the worst performance in general, with 82% for Precision, Recall and F1 score, even though it achieves 99.54% for Accuracy. This fact shows the importance of carefully choosing which metric is considered during the analysis. This situation shows that AdaBoost does not correctly predict the packets, and directly makes the prediction for the biggest category in the dataframe. Secondly, CatBoost and GradientBoost show a very similar performance, with a difference of 1%–2% in the Precision,

Table 8
Results obtained for the anomaly detection.

Algorithm	Precision	Recall	F1-Score	Accuracy	Cross validation score	Training-Test time (s)
XGBoost	**0.97**	1	**0.98**	0.99832	0,99788	0.26340
LightGBM	**0.97**	1	**0.98**	**0.99833**	0,99792	**0.03446**
AdaBoost	0.96	1	0.97	0.99831	**0,99796**	0.11984
CatBoost	0.96	1	0.97	0.99833	0,99792	0.14545
GradientBoost	0.95	1	0.96	0.99830	0.99788	0.44764

Recall and F1 score metrics. However, XGBoost has the best efficiency, with 99% in all the metrics. LightGBM obtains a performance of 98% in the same metrics, making the difference between the algorithms 1%, and in view of the low resources needed to run it could be considered a good alternative.

Algorithm complexity analysis. Once the classification performance metrics have been analysed depending on the different situations, it is necessary to analyse the complexity of the defined models. It is interesting to check the complexity/performance ratio of the algorithms in order to select the one that gives the best performance with the least complexity. This is even more important in a MEC-IIoT environment because of the type of devices assigned in IIoT topologies and the saving of resources in the MEC layer. The differences between the models are not very significant in terms of classification performance, as shown in Table 7 and analysed previously. However, if we look at the metric Training-Test Time, we can see that LightGBM is the best option. This model shows the best results in every situation of the experiment with a high margin of 1139 s in the Mixed Attack situation compared with GradientBoost. Otherwise, CatBoost shows the second best results in general, without considering the case of packet manipulation, but it also shows a general worse classification performance than LightGBM.

Finally, the analysis of the metrics shows that XGBoost obtains the best performance for scanning attacks and mixed attacks. GradientBoost is better for packet manipulation attacks, while for HTTP application attacks and DoS attacks there are not many differences between the algorithms. However, the difference between the best and the second-best algorithm according to the metrics is usually 1%–2%. This shows that the selection of the algorithm to use in the implementation of the intelligent threat detector should not only take performance metrics into account. The resources of the devices that are to run the intelligent threat detector should be considered because if the intelligent threat detector is to be deployed on a MEC edge server, the best algorithm for each case can be implemented. However, if the intelligent threat detector is to be deployed on an IIoT device with a lightweight algorithm, then LightGBM should be used as it has been showed during the complexity analysis for its complexity/peformance ratio.

7.2. Anomalies experimentation

After the classification performance analysis, it is necessary to carry out another experiment to prove that the models are able to detect anomalies in the network scenario. This is important because the detection of new or zero-day attacks, from which the model has no information, is one of the most important functions that the anomaly detector should support. In order to do so, the models are given new training data in a way that proves whether they are able to detect anomalies in network traffic caused by attacks they have not been trained for.

In this section, the experimentation follows the process explained in Section 7.1 and the steps shown in Fig. 3. However, some modifications have to be made in order to adapt the data obtained from the MEC-IIoT scenario to the new requirements, which is described in detail in this section, together with the results of the metrics of each algorithm.

7.2.1. Dataset elaboration

In order to prepare the dataset, we start with the collected data and preprocessing, which has been described in detail in Section 7.1. However, we decided to change the way in which the data is distributed and tagged when we split the data for the training and testing phases. With respect to the training dataset, 95% of the data is comprised of normal traffic, with the remaining 5% being anomalous. For this experimentation, the training dataset includes manipulation and DoS attacks. Regarding the testing dataset, with the aim of proving that all attacks can be detected, even the ones for which the model has not been trained for, both normal and anomalous traffic for every attack type present in our experimental setup are included . In addition, in terms of data preparation, every packet that comes from an attack is tagged as an anomaly, meaning that the detector is not aware of which type of attack the packet comes from.

Similar to the workflow described in Section 7.1, the rest of the process around model tuning and validation is similar.

7.2.2. Results

Table 8 shows the results obtained after performing a model validation for each metric and algorithm used to detect anomalies by the classification model. The analysis is divided into the detection performance and the complexity of the algorithm for each of the models that are implemented.

Anomaly detection. For this experiment, there are not many significant differences between all the algorithms. The best general results are provided by XGBoost and LightGBM with 97%, 100%, 98% and 99.83% respectively in the Precision, Recall, F1 Score and Accuracy metrics. This shows that these two models can achieve the same performance in detecting anomalies in the network, and the main difference could come from the complexity of the algorithm and how the model could be deployed in the environment. Otherwise, AdaBoost, CatBoost and GradientBoost achieve similar performance, losing only around 1%–2% in Precision and F1-Score compared with the best models. Even for Cross Validation Score, AdaBoost achieves the best result with 99.8%. However, the difference in this metric with the rest of the models is 0.1%, so it can be said that there is not a huge difference.

Algorithm complexity. As far as the complexity of the models is concerned, it is possible to find differences in the execution of the training and testing steps. In this experiment, LightGBM achieves the best results in terms of complexity. It takes it 0.03446 s to complete the whole process. Compared with the rest of the algorithms that have a similar classification performance but need more time to perform the same functionality, this shows the best complexity/performance ratio. However, it is important to highlight that each result is under one second of execution, showing that could be possible to deploy these models in devices with low computational devices.

8. Conclusions

The emergence of the IoT has led to the development of many areas that did not enjoy the advantages that technology brings with it. One of them has been the industrial environment, that has seen the IIoT as the next step in improving the performance and efficiency of factories. In particular, the use of MEC has been a key factor for the adoption of emerging technologies by the IIoT. However, this partnership has raised some concerns with regard to cybersecurity, as there is a clear lack of proper measures to detect and stop the attacks that are performed in this paradigm. Unupdated firmware and software, vulnerable services and the impossibility of executing the computational demanding traditional protection tools are some of the issues that are giving the opportunity to cybercriminals to cause huge damage and losses for companies. Consequently, solutions are needed that can help ensure the data exchanged in the IIoT, as well as the devices and systems in it, is protected.

Under these circumstances, in this work, Docker-Compose and openLEON have been used to deploy an IIoT-MEC scenario with OT services oriented to the network layer, in order to collect network data traffic to generate a custom-built dataset. This dataset has been prepared for use in the training and testing of intelligent threat detectors that are based on ML algorithms, and designed to detect attacks in the network. The algorithms selected for the intelligent threat detector implementation are boosted trees algorithms, specifically XGBoost, LigthGBoost, CatBoost, AdaBoost and GradientBoost. A study of the performance of these algorithms has been carried out, looking for correct classification in the prediction of benign or malicious packets. This study shows how the algorithms work with different sets of network traffic, with the best overall performance being obtained by the XGBoost algorithm, which afford perfectly to be deployed in the MEC topology. However, if the efficiency-cost ratio is an important parameter, LightGBM is the best option because of its implementation characteristics. This is important in IIoT-MEC scenarios because the majority of the devices found there have low computational resources and some models cannot be run if they do not match the device's requirements. The algorithms are also used in a second experiment to show whether they are able to detect anomalies in the network traffic resulting from different attacks for which the models had not been trained on before. In this study, the boosting algorithms perform equally in all metrics, showing that these models can detect anomalous network behaviour, with the only difference being the complexity algorithm. As in the first experiment, LightGBM again showed the best complexity/performance ratio.

9. Future work

In this section, we discuss some additional lines of research that could complement or extend our work:

- **Increase the range of our study.** IIoT not only contains traffic from OT protocols, but also many M2M protocols. For instance, MQ Telemetry Transport (MQTT), Constrained Application Protocol (CoAP), and Advanced Message Queuing Protocol (AMQP) are typical in IoT environments. Therefore, a new dataset can be made by including new protocols used in IIoT, and the algorithms can be analysed again with the new data. Furthermore, new attacks could be added, looking to run typical attack threads for OT and IoT devices. These additions will add variety to the dataset, and therefore to the training of the models, extending the capabilities of the intelligent threat detector for detecting new kinds of attacks and possibly obtaining more differences between the algorithms considered in the new study. Moreover, other ML algorithms could be added to compare with boosted trees algorithms.
- **Analyse the use of deep learning models.** As has been shown in this work, ML algorithms provide good performance in this type of context, although the analysis of DL techniques could introduce new ways of implementing and deploying applications for MEC-IIoT scenarios. For this reason, it is necessary to study the deep learning alternatives for the implementation of an intelligent threat detector, which could apply continuous learning and obtain better performance in dynamic scenarios where new applications are included and a different implementation of attacks can be run. Given the nature of the network data and the functionality of an intelligent threat detector, an interesting proposal in this area is that of using Adversarial Network models.
- **Study the performance of Federated Learning in our IIoT scenario.** Thanks to the characteristics and advantages that MEC brings to IIoT, Federated Learning could be an effective way of implementing an intelligent threat detector on each device in the scenario. For this implementation, the devices only have to store the data generated and shared it with the corresponding edge server, where the intelligent threat detector model will be allocated for that device. Thanks to the study carried out in this work, one of the algorithms can be used as the base model. Then, Federated Learning can be applied to the scenario to implement collaborative learning. Finally, the base model can be modified with the information provided by the rest of the devices, as mentioned, and coordinated on the MEC edge servers.
- **Study the vulnerabilities associated with ML boosted trees models.** There are some vulnerabilities and malicious techniques for manipulating the normal behaviour of a model during its training. A study of the effectiveness of these attacks with the boosted trees models used in this work should be carried out. Also, different defensive proposals should be analysed in an IIoT-MEC environment for this kind of intelligent threat detector, thus extending the study to other tools that use ML models for their implementation.

CRediT authorship contribution statement

Sergio Ruiz-Villafranca: Conceptualisation, Investigation, Data curation, Software, Formal analysis, Writing – original draft. **José Roldán-Gómez:** Supervision, Methodology, Conceptualisation, Validation, Writing – review & editing. **Javier Carrillo-Mondéjar:** Conceptualisation, Writing – review & editing. **Juan Manuel Castelo Gómez:** Resources, Conceptualisation, Writing – review & editing. **José Miguel Villalón:** Project administration, Writing – review & editing.

Declaration of competing 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.

Data availability

The data used in this work is available in the following link: https://data.mendeley.com/datasets/bk73vtsrpb/draft?a=0146acec-5daa-4968-ace3-34896ae68530.

Acknowledgements

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+), by the JCCM under the project SBPLY/21/180501/000195 and, and by the Spanish Education, Culture and Sports Ministry under grants FPU 17/03105. 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".

References

[1] D. Ivanov, C. Tang, A. Dolgui, D. Battini, A. Das, Researchers' perspectives on Industry 4.0: multi-disciplinary analysis and opportunities for operations management, Int. J. Prod. Res. (2020) 1–24, http://dx.doi.org/10.1080/00207543.2020.1798035.

[2] P.K.R. Maddikunta, Q.-V. Pham, P. B, N. Deepa, K. Dev, T.R. Gadekallu, R. Ruby, M. Liyanage, Industry 5.0: A survey on enabling technologies and potential applications, J. Ind. Inf. Integr. 26 (2022) 100257, http://dx.doi.org/10.1016/j.jii.2021.100257.

[3] X. Hou, Z. Ren, K. Yang, C. Chen, H. Zhang, Y. Xiao, IIoT-MEC: A novel mobile edge computing framework for 5G-enabled IIoT, in: 2019 IEEE Wireless Communications and Networking Conference, WCNC, IEEE, 2019, pp. 1–7.

[4] B. Ali, M.A. Gregory, S. Li, Multi-access edge computing architecture, data security and privacy: A review, IEEE Access 9 (2021) 18706–18721, http://dx.doi.org/10.1109/ACCESS.2021.3053233.

[5] C. Xenofontos, I. Zografopoulos, C. Konstantinou, A. Jolfaei, M.K. Khan, K.R. Choo, Consumer, commercial and industrial IoT (In)security: Attack taxonomy and case studies, CoRR abs/2105.06612, 2021.

[6] R. Das, M.Z. Gündüz, Analysis of cyber-attacks in IoT-based critical infrastructures, Int. J. Inf. Secur. Sci. 8 (4) (2019) 122–133.

[7] T. Plėta, M. Tvaronavičienė, S.D. Casa, K. Agafonov, Cyber-Attacks to Critical Energy Infrastructure and Management Issues: Overview of Selected Cases, ENTREPRENEURSHIP and SUSTAINABILITY CENTER, 2020.

[8] C. Alcaraz, J. Lopez, Digital twin: A comprehensive survey of security threats, IEEE Commun. Surv. Tutor. (2022) 1, http://dx.doi.org/10.1109/COMST.2022.3171465.

[9] H.-J. Liao, C.-H. Richard Lin, Y.-C. Lin, K.-Y. Tung, Intrusion detection system: A comprehensive review, J. Netw. Comput. Appl. 36 (1) (2013) 16–24, http://dx.doi.org/10.1016/j.jnca.2012.09.004, URL https://www.sciencedirect.com/science/article/pii/S1084804512001944.

[10] Z. Masood, M.A.Z. Raja, N.I. Chaudhary, K.M. Cheema, A.H. Milyani, Fractional dynamics of stuxnet virus propagation in industrial control systems, Mathematics 9 (17) (2021) http://dx.doi.org/10.3390/math9172160, URL https://www.mdpi.com/2227-7390/9/17/2160.

[11] C. Company, Cisco visual networking index: Global mobile data traffic forecast update, 2018–2023, in: Cisco White Paper, 2020, pp. 1–36.

[12] T. Chen, C. Guestrin, XGBoost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 785–794, http://dx.doi.org/10.1145/2939672.2939785, URL https://doi.org/10.1145/2939672.2939785.

[13] A. Borkar, A. Donode, A. Kumari, A survey on intrusion detection system (IDS) and internal intrusion detection and protection system (IIDPS), in: 2017 International Conference on Inventive Computing and Informatics, ICICI, 2017, pp. 949–953, http://dx.doi.org/10.1109/ICICI.2017.8365277.

[14] R. Panigrahi, S. Borah, A.K. Bhoi, M.F. Ijaz, M. Pramanik, R.H. Jhaveri, C.L. Chowdhary, Performance assessment of supervised classifiers for designing intrusion detection systems: A comprehensive review and recommendations for future research, Mathematics 9 (6) (2021) URL https://www.mdpi.com/2227-7390/9/6/690.

[15] S. Bebortta, D. Senapati, C.R. Panigrahi, B. Pati, Adaptive performance modeling framework for QoS-aware offloading in MEC-based IIoT systems, IEEE Internet Things J. 9 (12) (2022) 10162–10171, http://dx.doi.org/10.1109/JIOT.2021.3123554.

[16] N. Islam, F. Farhin, I. Sultana, M.S. Kaiser, M. Rahman, A.S.M. Hosen, G. Cho, G. Hwan, Towards machine learning based intrusion detection in IoT networks, Cmc -Tech Science Press 69 (2021) 1801–1821, http://dx.doi.org/10.32604/cmc.2021.018466.

[17] S. Mubarak, M.R.I. Mohamed Hadi Habaebi, M.T. Asaad Balla, E.A.A. Elsheikh, F.M. Suliman, Industrial datasets with ICS testbed and attack detection using machine learning techniques, Intell. Autom. Soft Comput. 31 (3) (2022) 1345–1360, http://dx.doi.org/10.32604/iasc.2022.020801, URL http://www.techscience.com/iasc/v31n3/44856.

[18] G.E. Selim, E.E.-D. Hemdan, A. Shehata, N. El-Fishawy, Anomaly events classification and detection system in critical industrial internet of things infrastructure using machine learning algorithms, Multimedia Tools Appl. 80 (2021) 1–22, http://dx.doi.org/10.1007/s11042-020-10354-1.

[19] P.M. Laso, D. Brosset, J. Puentes, Dataset of anomalies and malicious acts in a cyber-physical subsystem, Data Brief 14 (2017) 186–191, http://dx.doi.org/10.1016/j.dib.2017.07.038, URL https://www.sciencedirect.com/science/article/pii/S2352340917303402.

[20] I.A. Khan, M. Keshk, D. Pi, N. Khan, Y. Hussain, H. Soliman, Enhancing IIoT networks protection: A robust security model for attack detection in Internet Industrial Control Systems, Ad Hoc Netw. 134 (2022) 102930, http://dx.doi.org/10.1016/j.adhoc.2022.102930, URL https://www.sciencedirect.com/science/article/pii/S1570870522001159.

[21] T. Morris, A. Srivastava, B. Reaves, W. Gao, K. Pavurapu, R. Reddi, A control system testbed to validate critical infrastructure protection concepts, Int. J. Crit. Infrastruct. Prot. 4 (2) (2011) 88–103, http://dx.doi.org/10.1016/j.ijcip.2011.06.005, URL https://www.sciencedirect.com/science/article/pii/S1874548211000266.

[22] R. Mendonça, J. Silva, R. Rosa, M. Saadi, D.Z. Rodriguez, A. Farouk, A lightweight intelligent intrusion detection system for industrial internet of things using deep learning algorithm, Expert Syst. 39 (2022) http://dx.doi.org/10.1111/exsy.12917.

[23] H. Vargas, C. Lozano-Garzon, G. Montoya, Y. Donoso, Detection of security attacks in industrial IoT networks: A blockchain and machine learning approach, Electronics 10 (2021) 2662, http://dx.doi.org/10.3390/electronics10212662.

[24] N. Moustafa, J. Slay, 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, IEEE, 2015, pp. 1–6.

[25] J. Hu, K. Kaur, H. Lin, X. Wang, M.M. Hassan, I. Razzak, M. Hammoudeh, Intelligent anomaly detection of trajectories for IoT empowered maritime transportation systems, IEEE Trans. Intell. Transp. Syst. 24 (2) (2023) 2382–2391, http://dx.doi.org/10.1109/TITS.2022.3162491.

[26] X. Wang, S. Garg, H. Lin, G. Kaddoum, J. Hu, M.S. Hossain, A secure data aggregation strategy in edge computing and blockchain-empowered internet of things, IEEE Internet Things J. 9 (16) (2022) 14237–14246, http://dx.doi.org/10.1109/JIOT.2020.3023588.

[27] A. Filali, A. Abouaomar, S. Cherkaoui, A. Kobbane, M. Guizani, Multi-access edge computing: A survey, IEEE Access 8 (2020) 197017–197046.

[28] D. Borsatti, G. Davoli, W. Cerroni, C. Raffaelli, Enabling industrial IoT as a service with multi-access edge computing, IEEE Commun. Mag. 59 (8) (2021) 21–27, http://dx.doi.org/10.1109/MCOM.001.2100006.

[29] J. Liu, Q. Li, R. Cao, W. Tang, G. Qiu, MiniNet: An extremely lightweight convolutional neural network for real-time unsupervised monocular depth estimation, ISPRS J. Photogramm. Remote Sens. 166 (2020) 255–267.

[30] D. Kreutz, F.M. Ramos, P.E. Verissimo, C.E. Rothenberg, S. Azodolmolky, S. Uhlig, Software-defined networking: A comprehensive survey, Proc. IEEE 103 (1) (2014) 14–76.

[31] N. Goldenberg, A. Wool, Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems, Int. J. Crit. Infrastruct. Prot. 6 (2) (2013) 63–75, http://dx.doi.org/10.1016/j.ijcip.2013.05.001, URL https://www.sciencedirect.com/science/article/pii/S1874548213000243.

[32] A. Kleinmann, A. Wool, Accurate modeling of the siemens S7 SCADA protocol for intrusion detection and digital forensics, J. Digit. Forensics Secur. Law 9 (2014) 37–50, http://dx.doi.org/10.13140/2.1.1723.8727.

[33] H. Hui, K. McLaughlin, S. Sezer, Vulnerability analysis of S7 PLCs: Manipulating the security mechanism, Int. J. Crit. Infrastruct. Prot. 35 (2021) 100470, http://dx.doi.org/10.1016/j.ijcip.2021.100470, URL https://www.sciencedirect.com/science/article/pii/S1874548221000573.

[34] S.P. Muniraj, X. Xu, An implementation of OPC UA for machine-to-machine communications in a smart factory, Procedia Manuf. 53 (2021) 52–58, http://dx.doi.org/10.1016/j.promfg.2021.06.009, 49th SME North American Manufacturing Research Conference (NAMRC 49, 2021). URL https://www.sciencedirect.com/science/article/pii/S2351978921000093.

[35] R. Shwartz-Ziv, A. Armon, Tabular data: Deep learning is not all you need, Inf. Fusion 81 (2022) 84–90, http://dx.doi.org/10.1016/j.inffus.2021.11.011, URL https://www.sciencedirect.com/science/article/pii/S1566253521002360.

[36] Y. Gorishniy, I. Rubachev, V. Khrulkov, A. Babenko, Revisiting deep learning models for tabular data, in: M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, J.W. Vaughan (Eds.), Advances in Neural Information Processing Systems, Vol. 34, Curran Associates, Inc., 2021, pp. 18932–18943, URL https://proceedings.neurips.cc/paper_files/paper/2021/file/9d86d83f925f2149e9edb0ac3b49229c-Paper.pdf.

[37] B. Mahesh, Machine learning algorithms-a review, Int. J. Sci. Res. (IJSR) 9 (2020) 381–386.

[38] T. Zhang, W. He, H. Zheng, Y. Cui, H. Song, S. Fu, Satellite-based ground PM2.5 estimation using a gradient boosting decision tree, Chemosphere 268 (2021) 128801, http://dx.doi.org/10.1016/j.chemosphere.2020.128801, URL https://www.sciencedirect.com/science/article/pii/S0045653520329994.

[39] L. Li, S. Dai, Z. Cao, J. Hong, S. Jiang, K. Yang, Using improved gradient-boosted decision tree algorithm based on Kalman filter (GBDT-KF) in time series prediction, J. Supercomput. 76 (9) (2020) 6887–6900, http://dx.doi.org/10.1007/s11227-019-03130-y.

[40] A. Ogunleye, Q.-G. Wang, XGBoost model for chronic kidney disease diagnosis, IEEE/ACM Trans. Comput. Biol. Bioinform. 17 (6) (2019) 2131–2140.

[41] B. Ma, G. Yan, B. Chai, X. Hou, XGBLC: an improved survival prediction model based on XGBoost, Bioinformatics 38 (2) (2021) 410–418, http://dx.doi.org/10.1093/bioinformatics/btab675.

[42] R.E. Schapire, Explaining adaboost, in: Empirical Inference, Springer, 2013, pp. 37–52.

[43] W. Hu, W. Hu, S. Maybank, AdaBoost-based algorithm for network intrusion detection, IEEE Trans. Syst. Man Cybern. B 38 (2) (2008) 577–583, http://dx.doi.org/10.1109/TSMCB.2007.914695.

[44] A.V. Dorogush, V. Ershov, A. Gulin, CatBoost: gradient boosting with categorical features support, 2018, arXiv preprint arXiv:1810.11363.

[45] G. Huang, L. Wu, X. Ma, W. Zhang, J. Fan, X. Yu, W. Zeng, H. Zhou, Evaluation of CatBoost method for prediction of reference evapotranspiration in humid regions, J. Hydrol. 574 (2019) 1029–1041, http://dx.doi.org/10.1016/j.jhydrol.2019.04.085, URL https://www.sciencedirect.com/science/article/pii/S0022169419304251.

[46] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, LightGBM: A highly efficient gradient boosting decision tree, in: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 30, Curran Associates, Inc., 2017, pp. 1–9, URL https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[47] D. Rufo, T. Debelee, A. Ibenthal, W. Negera, Diagnosis of diabetes mellitus using gradient boosting machine (LightGBM), Diagnostics 11 (2021) 1714, http://dx.doi.org/10.3390/diagnostics11091714.

[48] J. Kirupakar, S.M. Shalinie, Situation aware intrusion detection system design for industrial IoT gateways, in: 2019 International Conference on Computational Intelligence in Data Science, ICCIDS, 2019, pp. 1–6, http://dx.doi.org/10.1109/ICCIDS.2019.8862038.

[49] Docker Inc., Docker container, 2022, https://www.docker.com/. (Accessed 19 September 2022).

[50] C. Fiandrino, A.B. Pizarro, P.J. Mateo, C.A. Ramiro, N. Ludant, J. Widmer, openLEON: An end-to-end emulation platform from the edge data center to the mobile user, Comput. Commun. 148 (2019) 17–26.

[51] M. Peuster, H. Karl, S. van Rossem, MeDICINE: Rapid prototyping of production-ready network services in multi-PoP environments, in: 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks, NFV-SDN, 2016, pp. 148–153, http://dx.doi.org/10.1109/NFV-SDN.2016.7919490.

[52] S. Asadollahi, B. Goswami, M. Sameer, Ryu controller's scalability experiment on software defined networks, in: 2018 IEEE International Conference on Current Trends in Advanced Computing, ICCTAC, IEEE, 2018, pp. 1–5.

[53] L. Zahedi, F.G. Mohammadi, S. Rezapour, M.W. Ohland, M.H. Amini, Search algorithms for automated hyper-parameter tuning, 2021, http://dx.doi.org/10.48550/ARXIV.2104.14677, URL https://arxiv.org/abs/2104.14677.

Sergio Ruiz Villafranca received a B.Sc. degree in Computer Engineering from the University of Castilla-La Mancha in 2017. In 2021, he received his M.Sc degree in Computer Science from the University of Castilla La Mancha (Spain). Since 2021, he is a researcher in that group and a Ph.D. student in Advanced Information Technologies at the aforementioned university. His research interests are related to cybersecurity, especially Industrial Internet of Things, machine learning for anomaly detection as well as computer forensics.

José Roldán-Gómez, and an M.Sc. in Computer Engineering from the University of Castilla-La Mancha in 2018. Since 2018, he has been working as a Phd student at the Albacete Research Institute of Informatics (I3A). His research interests include cybersecurity, machine learning, pattern generation and the Internet of Things.

Javier Carrillo Mondéjar received a B.Sc. degree in Computer Science and a Master's degree in Advanced Computer Science from the University of Castilla-La Mancha, Spain, in 2016 and 2017, respectively. Currently, he is enrolled full-time on the Ph.D. Program in Advanced Information Technology at this university. In 2016, he joined the Computer Architecture and Technology Group of the Informatics Research Institute of Albacete (I3A) as a researcher. His research interests are related to malware detection and classification techniques, as well as the methods used in malware to spread and remain hidden in computer systems. He has also been a visiting researcher at King's College London.

Juan Manuel Castelo Gómez joined the Computer Architecture and Technology research group at the Albacete Research Institute of Informatics in 2016. In 2017, he received his M.Sc. degree in Computer Science from the University of Castilla La Mancha (Spain). Presently, he is a researcher in that group and a Ph.D. student in Advanced Information Technologies at the aforementioned university. His research interests are related to cybersecurity, especially computer forensics, as well as malware detection.

José Miguel Villalón received an M.Sc. degree in Computer Science and a Ph.D. in Computer Engineering from the University of Castilla-La Mancha, Spain, in 2003 and 2007, respectively. In 2007, he joined the University of CastillaLa Mancha as Assistant Professor. He has been a Visiting Researcher at INRIA, France. His research interests include high-performance networks, wireless networks, QoS and QoE over IEEE 802.11 and WiMAX, multicast transmission, software-defined networking, video distribution, and error-resilient protocol architectures.