Advances in Science and Technology Research Journal

Advances in Science and Technology Research Journal, 2026, 20(1), 203–221
https://doi.org/10.12913/22998624/210686
ISSN 2299-8624, License CC-BY 4.0

Received: 2025.07.14
Accepted: 2025.09.13
Published: 2025.11.21

Optimization of machine learning models for effective anomaly detection in industrial IoT systems

Paweł Kuraś[1*], Marek Bolanowski[2], Mateusz Łoza[3]

[1] AGH University of Krakow, The Faculty of Computer Science, Electronics and Telecommunications, Institute of Telecommunications, al. A. Mickiewicza 30, 30-059 Krakow, Poland
[2] Department of Complex Systems, The Faculty of Electrical and Computer Engineering, Rzeszow University of Technology, Al. Powstańców Warszawy 12, 35-959 Rzeszów, Poland
[3] Independent researcher
* Corresponding author's e-mail: pawel.kuras@agh.edu.pl

ABSTRACT

In the era of increasing industrial internet of things (IIoT) devices, effective and efficient anomaly detection in network traffic is crucial for ensuring the security and reliability of industrial systems. This paper introduces a systematic methodology for optimizing machine learning models by focusing on the critical trade-off between detection accuracy and computational efficiency for resource-constrained IIoT environments. The methodology was evaluated using decision tree-based algorithms (RandomForest, ExtraTrees, AdaBoost, XGBoost, CatBoost) on a realistic dataset with simulated network attacks. The analysis involved a comprehensive evaluation of data preparation strategies, including class balancing, data aggregation, sampling, feature selection, and hyperparameter tuning, with a specific focus on the XGBoost model. The results demonstrate that this holistic optimization enables high detection accuracy (over 92% for binary classification and 87% for multi-class classification) while simultaneously maintaining high computational efficiency (short training time, small model size). This approach provides a practical pathway for developing resilient and resource-aware cybersecurity systems for industry, smart city, and IIoT environments.

Keywords: cybersecurity, random forest, XGBoost, network intrusion detection, anomaly detection, Industrial IoT (IIoT), machine learning optimization, decision tree algorithms

INTRODUCTION

The proliferation of internet of things (IoT) and industrial internet of things (IIoT) devices has led to a significant increase in the volume and complexity of data generated within interconnected systems. While these technologies drive innovation and efficiency across various sectors, they also introduce new vulnerabilities and expand the attack surface for cyber threats[27]. The number of security incidents in IT systems is growing annually. Threats such as phishing and malware pose a significant and well-documented risk [28], while distributed denial-of-service (DDoS) attacks continue to increase in scale and frequency [29]. Ensuring the security and reliability of these critical infrastructures necessitates robust and intelligent anomaly detection systems capable of identifying subtle deviations from normal behavior that may indicate malicious activity or system faults.

Multi-parameter anomaly detection methods, which analyze multiple variables simultaneously, offer the potential for more precise and effective threat identification compared to single-parameter approaches. These methods can uncover complex dependencies between variables, leading to a better understanding of system behavior and more accurate detection of unusual patterns. However, they also present challenges, including increased complexity, potential for model overfitting, and sensitivity to outliers, which require careful data preprocessing and model tuning.

This paper focuses on the application and optimization of machine learning (ML) models, specifically those based on decision tree algorithms, for multi-parameter anomaly detection in IIoT network traffic. The research investigates the efficacy of models such as RandomForest, ExtraTrees, AdaBoost, XGBoost, and CatBoost in identifying various network attacks. A key aspect of this work is the systematic evaluation of data preparation techniques, including class balancing strategies (OverSampling and UnderSampling), data aggregation within time windows, and data sampling methods. Furthermore, the study emphasizes the optimization of the XGBoost model through meticulous feature selection and hyperparameter tuning to achieve high detection accuracy while maintaining computational efficiency.

The primary contribution of this work lies in presenting a systematic optimization methodology that explicitly addresses the trade-off between detection accuracy and computational cost, a critical challenge in resource-constrained IIoT environments. Unlike previous studies that often focus singularly on maximizing accuracy, this paper demonstrates a practical pathway involving data reduction strategies (aggregation and sampling), feature selection, and hyperparameter tuning. The research proves that this holistic approach allows for the development of lightweight yet robust anomaly detection systems, significantly lowering the barrier for real-world deployment.

The general system model is presented in Figure 1. It consists of three main areas: IIoT, Network, and Processing. The IIoT Area is responsible for data aggregation, control operations, and the initial preprocessing of collected data. In the next step, data is transmitted through the Network Area using various communication protocols such as HTTP, MQTT, gRPC, and others. Finally, the data reaches the Processing Area, where it is stored and used by machine learning models for anomaly detection. In the traditional approach, the anomaly detection process for a large number of IIoT elements is highly complex due to the volume of transmitted data and the significant demand for computational power. This paper proposes a set of optimization methods that reduce both bandwidth usage and computational requirements, thereby enabling support for a greater number of IoT devices without the need to scale the communication or processing infrastructure. Particular attention should be paid to the boundary between the IIoT and Network areas. Network gateways are most commonly deployed at this point, and they can also serve as aggregation points for IoT data. If equipped with data preprocessing capabilities, these gateways can perform operations such as data aggregation, sampling, and feature selection. This approach aligns with the Edge Computing paradigm, which enables the distribution of processing across the entire system, thereby increasing its reliability and reducing operational costs. To verify the feasibility of the proposed approach, a series of experiments were conducted using the CIC-IoT 2023 dataset [1], which provides realistic network traffic data,

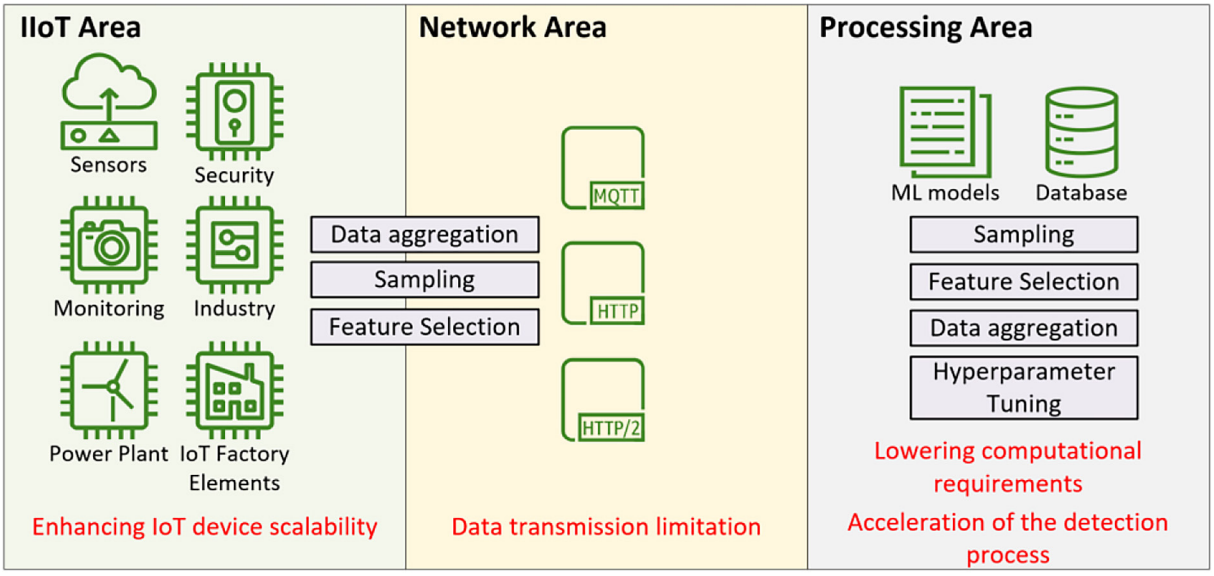

Figure 1. General model of an anomaly detection system in industrial IoT systems

including various simulated IoT attacks such as DDoS, DoS, Recon, and Spoofing. The findings indicate that through appropriate data preparation and rigorous model optimization, particularly for XGBoost, it is possible to achieve high anomaly detection rates: over 92% accuracy for binary classification and 87% for multi-class classification after optimization. Importantly, these results are achievable with reduced data requirements and lower computational overhead, making the proposed approach suitable for real-world deployment in resource-constrained IIoT environments. The insights gained from this research can contribute to the development of more resilient, efficient, and scalable cybersecurity solutions for industrial control systems, smart cities, and broader IIoT ecosystems.

This study makes several contributions to the field of anomaly detection in Industrial IoT. First, we propose a unified optimisation pipeline that integrates feature selection, class rebalancing, and hyperparameter tuning, all tailored for IIoT deployment scenarios. While each of these components has been explored separately in the literature, our novelty lies in systematically combining them to address resource constraints typical of edge devices, including limited memory, computation power, and latency tolerance. Second, we provide a detailed analysis of trade-offs between model accuracy, training time, and model size, which are rarely considered jointly in prior studies. Finally, we demonstrate that optimised tree-based models can achieve competitive accuracy with a significantly smaller computational footprint, highlighting their practical suitability for real-time IIoT anomaly detection.

This paper is structured as follows: Section 2 will briefly review related work in anomaly detection. Section 3 will describe the methodology, including the dataset, machine learning models, and optimization techniques. Section 4 will present and discuss the experimental results. Finally, Section 5 will conclude the paper and suggest future research directions.

RELATED WORK

The detection of anomalies in network traffic, particularly within IoT and IIoT environments, has become a critical area of research due to the increasing sophistication and volume of cyber threats. Effective anomaly detection systems are paramount for maintaining the security and operational integrity of these interconnected systems. This section reviews relevant literature concerning datasets, detection methodologies, and specific research efforts in this domain.

Datasets for anomaly detection

In this subsection, we provide an overview of publicly available datasets that have been used in the literature for anomaly detection in networked and IoT/IIoT environments. The aim is to position our choice of dataset in the broader research landscape by briefly summarising how earlier works relied on benchmarks such as KDD'99, NSL-KDD, or IoT-23, and why the CICIoT2023 dataset offers a more realistic and up-to-date basis for evaluating anomaly detection methods in IIoT systems.

The availability of representative datasets is crucial for the development and evaluation of anomaly detection models. Several publicly available datasets have been utilized by the research community. Early datasets, such as KDD'99 and its successor NSL-KDD, served as benchmarks for network intrusion detection [2]. However, with the rise of IoT, more specific datasets have emerged. TON_IoT, for instance, includes data from various IoT devices and simulated attacks such as DoS, DDoS, and ransomware [24]. The IoT-23 dataset provides traffic from real IoT devices, capturing activity from 2018–2019 [25]. Other notable datasets include CIDDS, focusing on anomaly-based intrusion detection in virtual environments, and CIC-IDS2017, which aimed to generate realistic network traffic data [26].

For this study, the CIC-IoT Dataset 2023 [1] was selected. This dataset is particularly relevant as it was generated by simulating attacks on a large topology of 105 IoT devices and includes seven categories of modern attacks such as DDoS, DoS, Reconnaissance, Brute Force, and Spoofing, with malicious activities originating from IoT devices themselves. The recency and specificity of this dataset to IoT environments make it well-suited for evaluating the proposed multi-parameter anomaly detection methods.

Anomaly detection methodologies

This subsection reviews the main methodological approaches to anomaly detection as presented in the literature, including statistical, time-series,

and AI-based techniques. By highlighting the range of methods adopted in previous studies, we place our own focus on supervised, tree-based machine learning models into context and underline why they are particularly relevant for tabular network traffic data in IIoT environments.

Various approaches have been proposed for anomaly detection, broadly categorized into statistical methods, time-series analysis, and artificial intelligence (AI) techniques. Statistical methods, both parametric and non-parametric, analyze data distributions to identify outliers. Time-series analysis focuses on identifying deviations from established patterns, seasonality, or trends, often employing models like ARMA or ARIMA.

AI-based methods, which are the focus of this paper, encompass machine learning (ML) and Neural Networks (NN). Machine learning algorithms learn patterns from data to classify normal and anomalous behavior. Popular ML algorithms include decision trees, support vector machines (SVM), and ensemble methods like random forests. Neural networks, with their flexible architectures, have also shown promise, with deep neural networks (DNNs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs) being applied to anomaly detection. These AI methods can be trained using supervised, unsupervised, semi-supervised, or reinforcement learning paradigms, depending on the availability of labeled data and the specific problem requirements. This work specifically explores supervised ML algorithms based on decision trees due to their interpretability and effectiveness with tabular data.

Recent works in IoT/IIoT anomaly detection have shown promising results with neural models. For example, Chen et al. [37] propose a hybrid XGBoost-LSTM architecture, named MIX_LSTM, that combines feature selection and sequence modeling for balanced, accurate detection on UNSW-NB15 and NSL-KDD datasets. Ayad et al. [38] achieve high detection rates with a lightweight autoencoder-DNN pipeline designed for real-time IoT traffic. Xin et al. [39] compare convolutional neural networks and variational autoencoders, finding CNN excels in accuracy (~95.9 %) while VAE effectively captures anomalies via reconstruction error. Kusumastuti et al. [40] benchmark transformer, 1D-CNN, and GrowNet architectures against classical models, including assessing inference latency and training time – important considerations for deployment. Finally, Al-Qudah and AlMahamid [41] provide a multi-step evaluation framework comparing RNN-LSTM, autoencoders, and Gradient Boosting under varied preprocessing, demonstrating that tree-based methods often outperform deep models under limited resources.

Relevant studies

Numerous studies have explored anomaly detection in network and IoT environments. For instance, Poornima and Paramasivan [3] proposed a model based on linear weighted projection regression (OLWPR) combined with PCA for dimensionality reduction, achieving 86% accuracy in detecting anomalies in wireless sensor networks. Suthaharan et al. [4] discussed the challenges of obtaining labeled data for anomaly detection in wireless sensor networks and proposed an ellipse-based detection system, highlighting issues with adapting such models to diverse environmental data.

Zhou and Li [5] developed a network traffic anomaly detection model using multilevel autoregression based on information entropy, reporting 95% effectiveness inDDoS attack detection by analyzing data in time windows. Fouad and Abdel-Hamid [6] utilized hidden Markov models (HMM) to model power consumption patterns of IoT sensor nodes to detect attacks and hardware trojans, deploying their system in a cloud environment. More recently, Kisanga et al. [7] proposed a graph neural network (GNN) for anomaly detection, achieving accuracies of 76% on a DDoS dataset and 88% on a TOR-nonTOR dataset, demonstrating the application of deep learning on graph-structured data.

While these studies showcase diverse approaches, there remains a need for optimized ML models that are both highly accurate and computationally efficient, especially for the resource-constrained nature of many IIoT systems. This paper addresses this gap by focusing on the optimization of decision tree-based ensemble models through data preparation strategies, feature selection, and hyperparameter tuning, with a particular emphasis on the XGBoost algorithm, aiming to provide a practical and effective solution for anomaly detection in IIoT environments. Furthermore, recent research by Sezgin and Boyacı emphasizes the critical role of automated preprocessing and feature selection in enhancing IDS performance in IIoT environments [30]. Their work on hybrid feature selection aligns with our

study's focus on data preparation as a key step towards building more accurate and computationally efficient detection systems.

METHODOLOGY

This section details the research methodology, including the dataset characteristics, data preparation steps, machine learning models employed, experimental setup, and the metrics used for evaluating model performance.

Dataset

The primary dataset used in this study is the CIC-IoT Dataset 2023 [1], chosen for its relevance to IoT environments and inclusion of contemporary network attacks. This dataset was generated from a topology of 105 IoT devices and includes traffic related to 33 types of attacks, categorized into seven main groups such as DDoS, DoS, Reconnaissance, Brute Force, Spoofing, and Mirai. For this research, a 10% stratified sample of the original dataset was utilized, resulting in 3,132,327 records with 44 features, to ensure manageable processing times while maintaining class proportions. The attack types specifically included in our analysis were Benign, DDoS, Spoofing, Reconnaissance, and a composite 'other' category comprising SQL Injection, cross-site scripting (XSS), and Dictionary attacks.

Instead of using the pre-processed versions of the dataset, features were extracted directly from the raw.pcap files. This was accomplished using custom Python scripts leveraging the Scapy library to parse network packet headers from various layers of the TCP/IP model, capturing all available fields. This approach allowed for a more granular selection of features relevant to multi-parameter anomaly detection. The extracted features were then compiled into CSV files. Standard preprocessing steps included handling missing values (if any), converting data to appropriate numerical types, and encoding categorical features (though tree-based models can often handle categorical features natively or with specific encoding strategies). Numerical features were scaled using standardization (Z-score normalization) to ensure that features with larger value ranges did not dominate those with smaller ranges, which can be beneficial for some algorithms and for consistent interpretation of feature importance.

The CICIoT2023 dataset used in our experiments did not contain missing values in the selected features; therefore, no imputation procedure was required. All attributes were numeric, and categorical encoding was unnecessary. Feature standardisation was applied to ensure comparability across different models. While tree-based classifiers are generally insensitive to feature scaling, algorithms such as logistic regression, SVM, and kNN require normalised input for stable training. By applying the same preprocessing pipeline across all models, we ensured consistent evaluation under comparable conditions.

The dataset was divided into training and testing sets using an 80/20 split, respectively, a common practice to evaluate model generalization on unseen data. Stratified sampling was employed during the split to maintain the original proportion of classes in both training and testing sets. Network traffic datasets are often imbalanced, with benign traffic significantly outnumbering malicious traffic, or certain attack types being rare. To address this, various techniques from the imbalanced-learn library [12] were explored. These include:

- Oversampling techniques:
  - SMOTE (synthetic minority over-sampling technique) [13]: Generates synthetic samples for the minority class.
  - Borderline-SMOTE [14]: A SMOTE variant that focuses on samples near the class border.
  - ADASYN (adaptive synthetic sampling) [15]: Adaptively generates more synthetic data for minority class samples that are harder to learn.
- Undersampling techniques:
  - NearMiss [16]: Selects majority class samples based on their distance to minority class samples.
  - One-Sided Selection (OSS) [17]: Removes noisy and borderline majority class samples. The impact of these balancing strategies was evaluated as part of the model optimization process.

Machine learning models

This research focused on supervised machine learning algorithms based on decision trees, known for their effectiveness on tabular data and interpretability. The selected models were:

- Random forest: An ensemble method that constructs multiple decision trees on different sub-samples of the dataset and features, and aggregates their predictions (e.g., by voting) to improve accuracy and control overfitting [18].
- ExtraTrees (extremely randomized trees): Similar to random forest, but introduces more randomness by selecting random thresholds for splitting nodes and using the whole training sample to grow trees [19]. This can reduce variance and training time.
- AdaBoost (adaptive boosting): An iterative ensemble method that combines multiple weak learners (typically decision trees). It adaptively reweights samples in each iteration, giving more weight to misclassified instances, thus focusing on harder-to-classify samples [20].
- XGBoost (extreme gradient boosting): A highly efficient and scalable implementation of gradient boosting, which sequentially adds predictors, each correcting its predecessor's errors. It employs regularization to prevent overfitting and handles missing values [21].
- CatBoost (categorical boosting): Another gradient boosting algorithm that effectively handles categorical features and is designed to combat overfitting. It uses a novel approach for processing categorical data and ordered boosting [22].

In the initial stage of the experiments, all models were trained with a set of standard hyperparameter values that served as a baseline for further optimisation. For Decision Trees and Random Forests, we adopted the classical *gini* criterion, no restriction on tree depth, and the default minimum number of samples required to split a node. In the case of Random Forest, one hundred trees were used, which represents a common compromise between accuracy and computational cost. For boosting methods, namely Gradient Boosting and XGBoost, the initial configuration consisted of 100 estimators, a learning rate of 0.1, and maximum tree depths of 3 and 6 respectively. XGBoost additionally used full subsampling of both observations and features (subsample = 1.0, colsample_bytree = 1.0). Logistic Regression was configured with L2 regularisation, the *lbfgs* solver, and a penalty parameter C = 1.0. Support Vector Machines employed the radial basis function kernel (*rbf*), with C = 1.0 and gamma set to "scale". The k-Nearest Neighbours algorithm was initially run with five neighbours, using the Minkowski distance metric with p = 2. These baseline settings reflect commonly used defaults in libraries and the literature and provided a transparent and reproducible starting point for the subsequent hyperparameter optimisation process.

Experimental environment

Experiments were conducted using Python version 3.11 within the JupyterLab environment. Key libraries included Scikit-learn [23] for core ML algorithms and preprocessing, Pandas for data manipulation, NumPy for numerical operations, Matplotlib for visualization, Imbalanced-learn [12] for handling imbalanced datasets, and the specific libraries for XGBoost and CatBoost. Computations were performed on a standard desktop computer equipped with an Intel i3 10th generation processor and 32GB of RAM. Model performance was evaluated using a comprehensive set of metrics suitable for classification tasks, particularly in the context of imbalanced data: accuracy, precision, recall, F1-score, confusion matrix, ROC AUC (area under the receiver operating characteristic curve)

Metrics

To evaluate model performance, several standard classification metrics were employed [31]:

- Accuracy – the ratio of correctly classified instances to the total number of instances. While widely used, it can be misleading in imbalanced datasets, as a model predicting only the majority class may still achieve high accuracy [32].
- Precision – the proportion of correctly predicted positive instances among all predicted positives. High precision indicates few false alarms [33].
- Recall (sensitivity) – the proportion of correctly predicted positive instances among all actual positives. Recall is particularly critical in anomaly detection, as false negatives (missed attacks) may have severe consequences in IIoT environments [34].
- F1-score – the harmonic mean of precision and recall, balancing the two and providing a single measure that is less sensitive to class imbalance than accuracy alone [35].
- ROC AUC (area under the receiver operating characteristic curve) – measures the trade-off between true positive and false positive rates across thresholds. While useful, ROC AUC

may overestimate performance in imbalanced settings [36].
- PR AUC (area under the precision-recall curve) – especially informative for imbalanced data, as it focuses on the performance with respect to the minority class [36].

Given the strongly imbalanced nature of IIoT traffic data, we emphasised metrics beyond accuracy, particularly recall, F1-score, and PR AUC, to better capture the model's ability to detect minority-class attacks.

Optimization strategies

The environment of IoT and IIoT devices very often combines hardware limitations and a large amount of data transmitted over the network coming from a very large number of sources. In order to effectively detect anomalous situations - including threats – in such an environment, it is necessary to apply a two-pronged approach to the design of this class of systems. On the one hand, we need to limit the amount of data sent to the detection system and, on the other hand, optimize the use of hardware resources. To achieve optimal performance, four main optimization strategies were employed:
- Data aggregation – data aggregation: is a strategy for reducing the number of records used in the process of learning and exploiting the ML model. Its use allows it to determine a single representative record for a group of rows from the analyzed dataset. This approach reduces the time required for learning and anomaly detection while maintaining good quality parameters of the ML model.
- Sampling – is a technique involving random or periodic selection of rows from a data set. It reduces the learning time of the ML model, and reduces the demand of the anomaly detection system for hardware system resources. However, too "rare" sampling of data can lead to a drastic degradation of the quality parameters of the anomaly detection model.
- Feature selection – identifying and selecting the most relevant features to reduce dimensionality, improve model training time, and potentially enhance generalization by removing irrelevant or redundant features. (Specific methods used will be detailed in the results section).
- Hyperparameter tuning – systematically searching for the best combination of model hyperparameters using techniques such as Grid Search or Randomized Search with cross-validation. This was a critical step, especially for complex models like XGBoost, to maximize their predictive power.

The models were evaluated for both binary classification (distinguishing between 'Benign' and any 'Attack') and multi-class classification (identifying specific attack categories).

PROPOSED APPROACH AND DISCUSSION

This section presents the experimental results obtained from applying the methodologies described in Section 3. It includes an initial analysis of the dataset, baseline performance of the selected machine learning models, and a detailed account of the impact of various optimization strategies, including class balancing, feature selection, and hyperparameter tuning. The discussion will focus on the effectiveness of these strategies in improving anomaly detection in IIoT network traffic.

Data analysis and initial observations

The CIC-IoT Dataset 2023 [1], after preprocessing and sampling (10% of the original), consisted of 3,132,327 records and 44 features. A critical initial step was to analyze the distribution of classes within the dataset. The dataset exhibited a significant class imbalance, which is common in network traffic data where benign traffic usually dominates. For instance, in the multi-class scenario (Figure 2c), the 'Benign' (labeled 1) class was overwhelmingly prevalent, while some attack classes like 'Spoofing' or 'Recon' (detailed in Figure 2a and grouped in Figure 2b) were significantly less represented. This imbalance can bias machine learning models towards the majority class, leading to poor detection of minority attack classes. A visual representation of the class distribution for binary and multi-class labels is shown in Figure 2.

Feature importance analysis was also conducted. Tree-based models, such as XGBoost, provide inherent mechanisms to rank features. An initial assessment of feature importance revealed that features related to packet timing, flow duration, and packet sizes were among the most discriminative.

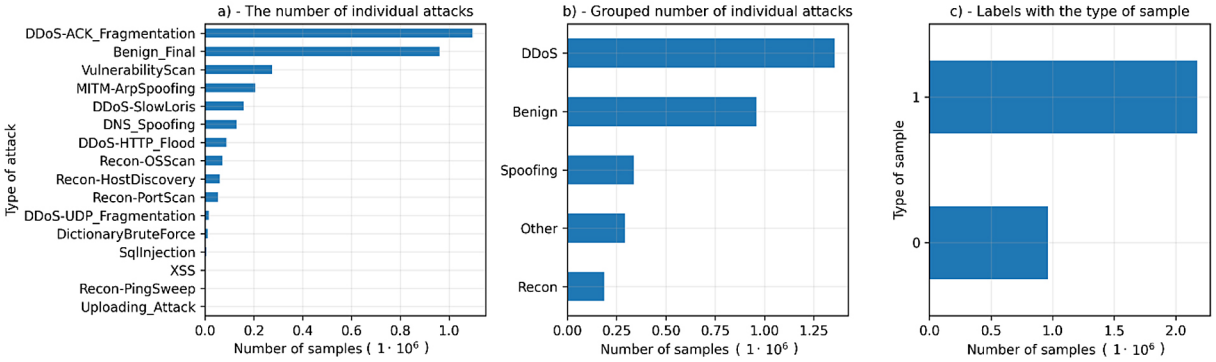

Figure 2. Class distribution in the sampled CIC-IoT Dataset 2023

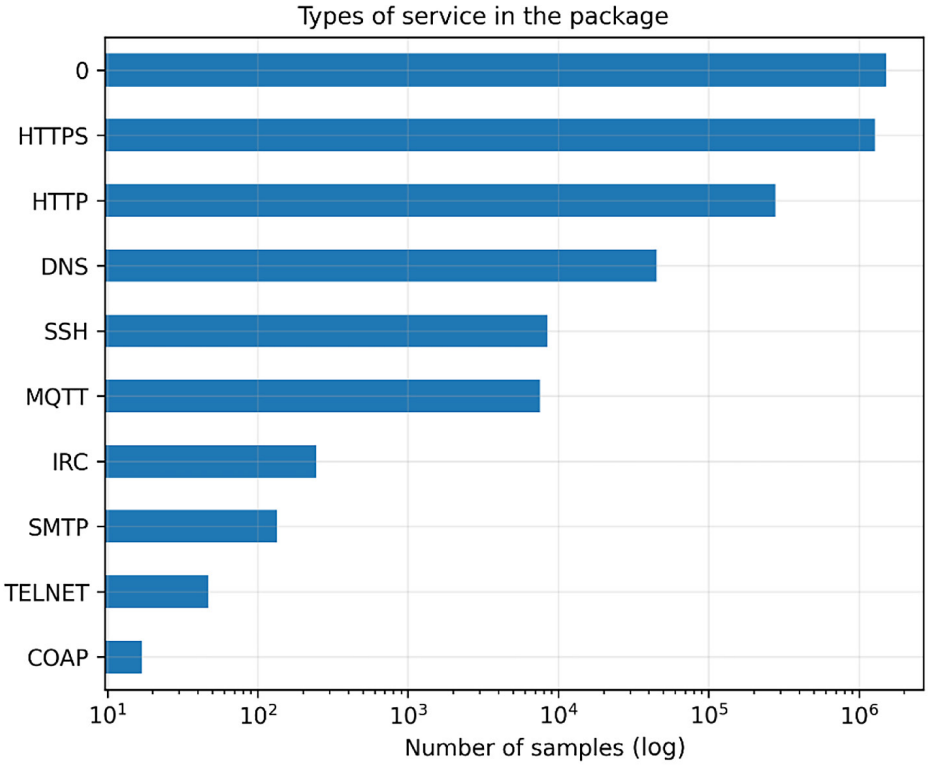

Figure 3. Distribution of application layer protocols (Service Name)

The dataset also contained categorical features, such as application layer protocols (Figure 3), which were numerically encoded. Certain columns like timestamps and IP/MAC addresses were removed as they were not deemed essential for the research focus or could lead to overfitting in a general context. A correlation matrix (Figure 4) was also examined to understand relationships between features and reveals several important patterns. A subset of features, particularly those related to UDP header statistics, show strong correlations, indicating potential redundancy in the dataset. At the same time, most other features are weakly correlated, suggesting that they contribute distinct information useful for classification. Such redundancy can increase training time without necessarily improving accuracy; however, tree-based models are relatively robust to correlated inputs. These observations motivated the later use of feature selection to reduce dimensionality and highlight the most informative variables.

The correlation analysis confirmed that although the dataset contains groups of redundant features, the majority of variables are complementary. This justified the application of feature selection methods, which contributed to improving both the efficiency and interpretability of the final models.

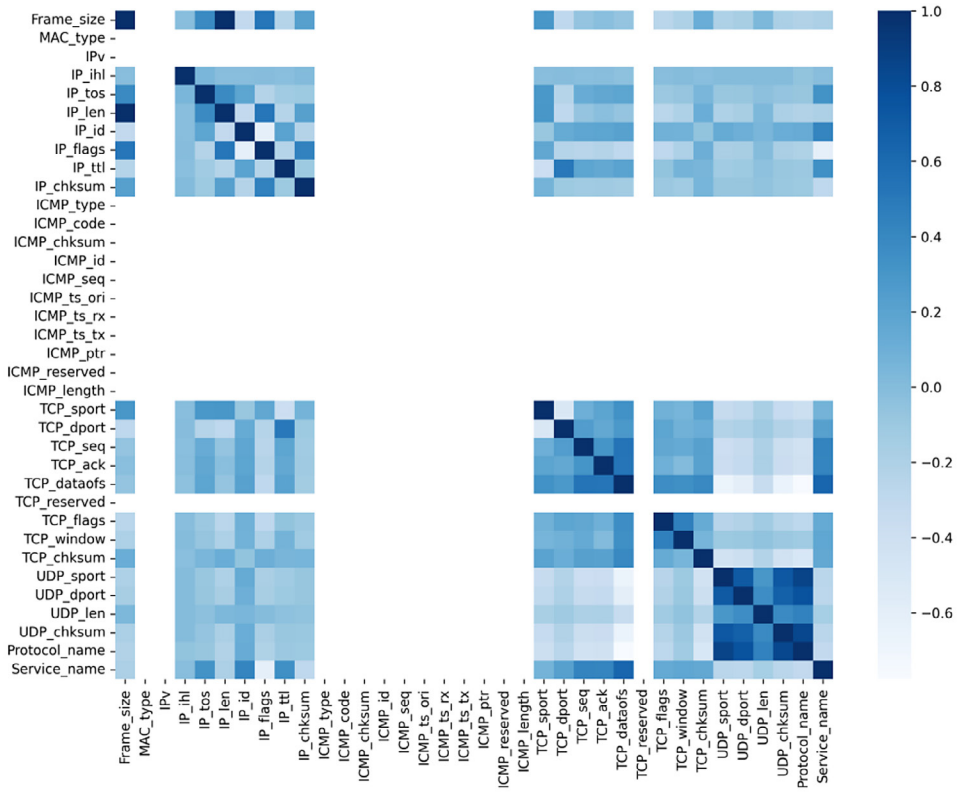

Figure 4. Feature correlation matrix

Baseline model performance

Before applying any optimization techniques, the selected machine learning models (RandomForest, ExtraTrees, AdaBoost, XGBoost, CatBoost) were trained and evaluated using their default hyperparameters on the full, initially imbalanced 10% dataset sample. Cross-validation (3-fold) was used to assess baseline accuracy. The results showed that models based on decision trees are generally insensitive to data normalization, yielding identical accuracy scores with and without it (Figure 5 and Figure 6). For binary classification (Attack vs. Benign), RandomForest and ExtraTrees achieved the highest baseline cross-validated accuracy at 96%, while AdaBoost performed the worst at 86% (Figure 5). For multi-class classification, a similar trend was observed, with RandomForest and ExtraTrees again leading, though overall accuracies were lower, and AdaBoost showed a significant drop of over 10% (Figure 6).

Impact of class balancing techniques

To address class imbalance, various oversampling (RandomOverSampler, SMOTE [13], Borderline-SMOTE [14], ADASYN [15]) and undersampling (RandomUnderSampler, NearMiss [16], One-Sided Selection [17]) techniques were applied and their processing times recorded (Table 1, Table 2).

RandomOverSampler and RandomUnderSampler were the fastest due to their simpler mechanisms. ADASYN and BorderlineSMOTE were the most time-consuming oversamplers, while OneSidedSelection took the longest among undersamplers. The effect of these techniques on class distribution is shown in Table 3.

Most methods aimed to equalize class counts, except ADASYN and OneSidedSelection, which resulted in differing class proportions based on their algorithms.

Model selection

For subsequent experiments, RandomUnderSampler was chosen due to its speed and simplicity, creating balanced datasets for binary (approx. 864k samples per class) and multi-class (approx. 168k samples per class) scenarios. After applying the undersampling procedure, the dataset ach13ieved a balanced distribution between classes, reducing the dominance of the benign

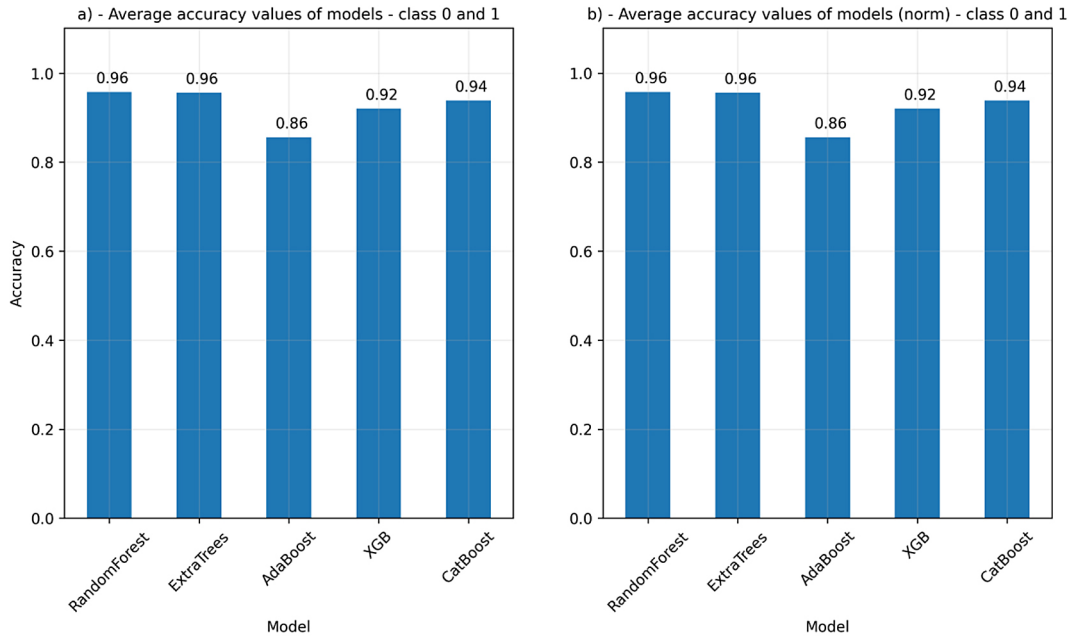

Figure 5. Baseline mean accuracy of models for binary (0/1) classification with 3-fold cross-validation, with and without data normalization.

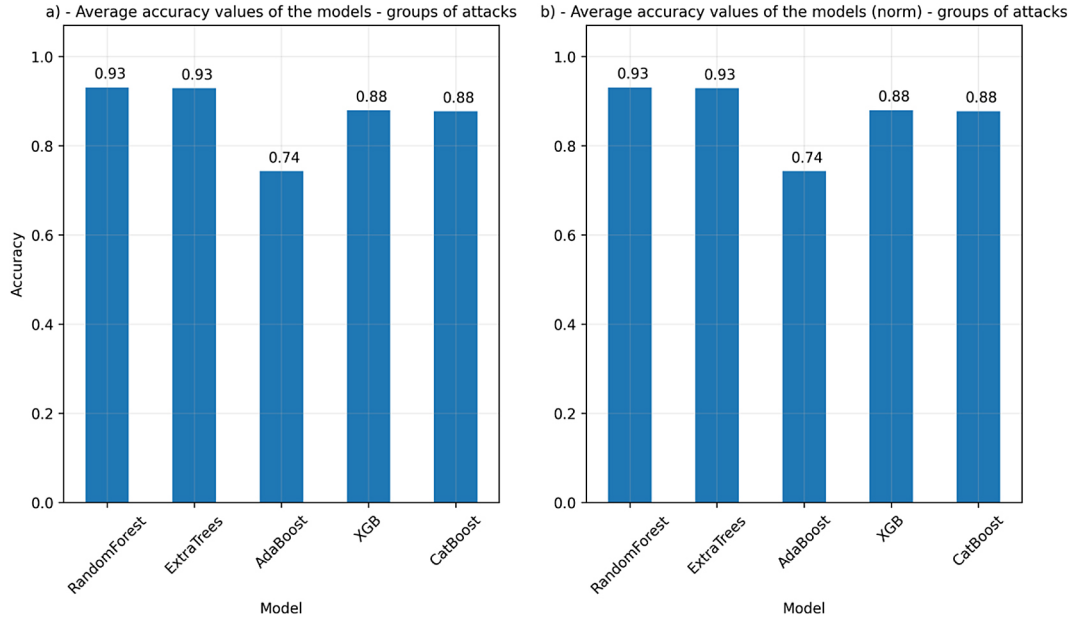

Figure 6. Baseline mean accuracy of models for multi-class (grouped attacks) classification with 3-fold cross-validation, with and without data normalization.

traffic and making the minority attack categories more comparable in size.

On these balanced datasets (without cross-validation for this phase), RandomForest generally showed the highest F1-scores (0.950 for binary, 0.890 for multi-class) but had long training times and large model sizes (Table 4 and Table 5).

XGBoost demonstrated the fastest training times (8.27 s for binary, 23.56 s for multi-class) with good F1-scores (0.912 for binary, 0.832 for multi-class) and smaller model sizes compared to RandomForest/ExtraTrees. AdaBoost had the smallest model size but lower performance metrics. RandomForest and ExtraTrees showed signs of overfitting, with training accuracy at 0.999 while test accuracy was lower. Due to its efficiency and strong performance, XGBoost was selected for further optimization.

Table 1. Oversampling methods

Oversampling	
Model	Time [s]
RandomOverSampler	1.94
SMOTE	1045.63
BorderlineSMOTE	5240.32
ADASYN	6082.01

Table 2. Undersampling methods

Undersampling	
Model	Time [s]
OneSidedSelection	9395.79
RandomUnderSampler	0.87
NearMiss	491.00

Impact of data aggregation and sampling strategies for XGBoost

Data aggregation

A data aggregation strategy was tested where the original dataset (31,323,200 rows) was divided into 100-element windows, from which 10 records were randomly sampled. Various aggregation functions (sum, mean, std, variance, median) were applied per column, and the label was determined by a threshold on the sum of attack labels in the window. This resulted in a dataset of 313,232 rows. A schematic of this process is shown in Figure 7.

For better understanding and visualization of this process, the diagram above has been prepared. It should be read from left to right, and the individual symbols represent:

- n – the number of selected rows for aggregation,
- m – the number of randomly selected rows from the n rows,
- $k_1, k_2,..., k_i$ – columns in the dataset,
- a_{ni} – values in the selected n rows; where a_n = 1, 2,..., n and a_i = 1, 2,..., i,
- a_{mi} – values in the m randomly selected rows from the n rows; where a_m = 1, 2,..., m and a_i = 1, 2,..., i,
- AGR_i – the chosen aggregation operation applied to the m rows for each column k_i; where k = 1, 2,..., i and AGR = 1, 2,..., i,
- $\sum$ – the summation symbol, which can be expressed with the formula $\sum_{i=1}^{m} {}^{i}$, and it represents the operation of summing m rows of a column with a label to determine the threshold indicating the occurrence of a threat.

The aggregated dataset was compared against a randomly sampled 10% subset of the original 3 million row dataset (313,232 rows) using 5-fold cross-validation. Aggregation positively impacted model accuracy, especially for AdaBoost, with average accuracies around 95% for (Figure 8a) compared to generally lower scores for 10% sampled dataset (Figure 8b).

Application of the approach to reduce the size of data sets in the process of anomaly detection contributes to lower demand for computing power of anomaly detection systems. It also allows the use of local data aggregation systems, which

Table 3. Results of over-sampling and under-sampling methods applied to CICIoT2023 dataset

Method	Model	Class distribution (number of samples)
Oversampling	RandomOverSampler	1.0: 1219754 2.0: 1219754 3.0: 1219754 4.0: 1219754
	SMOTE	
	BorderlineSMOTE	
	ADASYN	0.0: 1225001 1.0: 1219754 2.0: 1222312 3.0: 1236447 4.0: 1224470
Undersampling	OneSidedSelection	0.0: 821529 1.0: 1172414 2.0: 218128 3.0: 168329 4.0: 286282
	RandomUnderSampler	0.0: 168329 1.0: 168329 2.0: 168329 3.0: 168329 4.0: 168329
	NearMiss	

Table 4. Results of tests for a label with two classes 0 and 1

Model	Learning time [s]	Accuracy (train)	Accuracy (test)	F1	Recall	Precision	Size [B]
RandomForest	585.10	0.999	0.950	0.950	0.943	0.957	1141M
ExtraTrees	304.10	0.999	0.948	0.948	0.942	0.954	3323M
AdaBoost	174.00	0.826	0.826	0.817	0.774	0.865	28056
XGB	8.27	0.915	0.914	0.912	0.885	0.940	376107
CatBoost	142.24	0.933	0.930	0.928	0.908	0.950	1096591

Table 5. Results of tests for the label with grouped threat

Model	Learning time [s]	Accuracy (train)	Accuracy (test)	F1	Recall	Precision	Size [B]
RandomForest	278.74	0.999	0.888	0.890	0.888	0.895	1694M
ExtraTrees	136.50	0.999	0.887	0.889	0.887	0.894	4048M
AdaBoost	81.12	0.616	0.617	0.625	0.617	0.648	31890
XGB	23.56	0.832	0.829	0.832	0.829	0.847	1932667
CatBoost	225.65	0.828	0.826	0.828	0.826	0.844	3144255

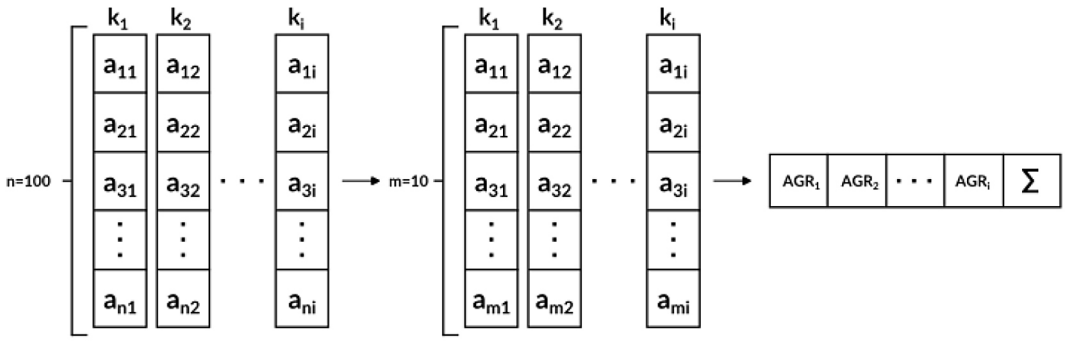

Figure 7. Schematic of the data aggregation process

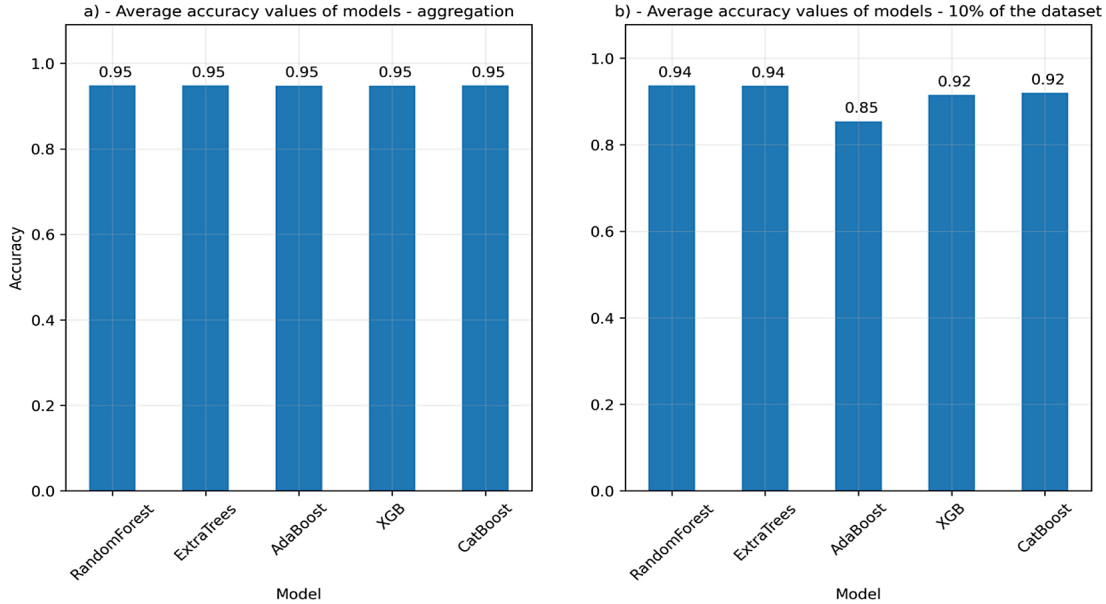

Figure 8. Mean accuracy comparison after 5-fold cross-validation: (a) aggregated dataset, (b) 10% sampled dataset

consequently limits the amount of data sent over the network to central monitoring systems.

Data sampling

Sampling is the process during which a new set of data is obtained, a smaller but still representative subset. The impact of reducing the training and testing set size for XGBoost was investigated by iteratively training and testing on decreasing fractions of the balanced dataset (from 1/5th down to 1/200th). Only in the first step there is a value of 5 and the next steps, starting from 10, are 10 more than the previous one, up to the value of 200 (i.e. 5, 10, 20, 30,..., 200). Sampling has a big advantage over data aggregation - it can be realizedon the fly by a dedicated probe, which selects part of the packets and sends it to the detection system. The rest of the packets are skipped. In the case of aggregation, all packets must be processed and a record representative of a certain group of packets is selected on the basis of their analysis. This process is much more computationally complex than sampling data "on the fly".

Results (Figure 9 for binary, Figure 10 for multi-class) showed that as the number of samples decreased (moving from left to right on the x-axis, which represents the divisor of the entire dataset of size N), test accuracy generally declined, and the gap between training and test accuracy widened, indicating increased overfitting. Model size predictably decreased with fewer samples. It should be noted that in some applications, anomaly detection accuracy at a reduced level of, for example, 75% may be sufficient – since AI models are one of the indicators of threat occurrence. In both cases considered (Figure 11, Figure 12), these types of detection accuracy levels are achieved for up to 100 times reduction of analyzed data – which significantly increases the potential for implementation of such an approach in industrial systems.

Feature selection for XGBoost

Another area that is important from the point of view of optimizing the process of anomaly detection in IoT systems is related to the selection of the most relevant features (columns) from the perspective of the model. Limiting the set of analyzed features reduces the size of the model, and limits the amount of data that must be sent from monitored elements to the anomaly detection system. Feature selection was performed based on XGBoost's inherent feature importance scores. For binary classification, selecting a subset of the most important features to achieve >

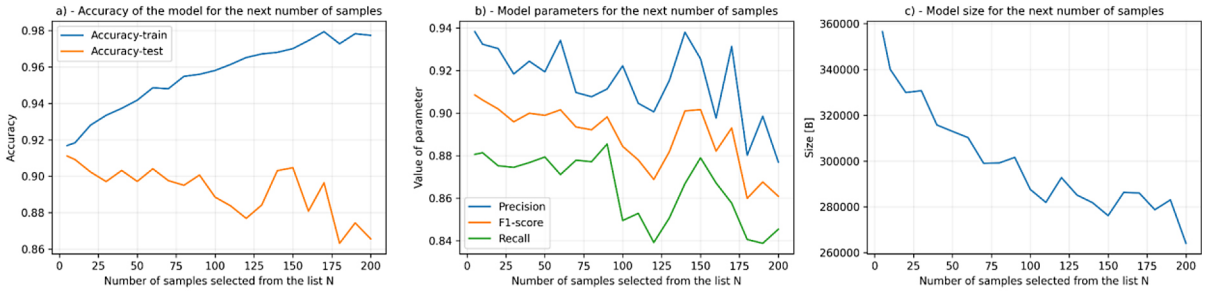

Figure 9. XGBoost performance for binary (0/1) classification with varying sub-sample sizes: (a) training vs. test accuracy, (b) test metrics (F1, Precision, Recall), (c) model size

Figure 10. XGBoost performance for multi-class (grouped attacks) classification with varying sub-sample sizes: (a) training vs. test accuracy, (b) test metrics (F1, Precision, Recall), (c) model size

	Learning time[s]	Accuracy-train	Accuracy-test	F1-score	Recall	Precision	Size [B]
['IP_flags', 'Service_name', 'IP_id', 'TCP_flags', 'IP_tos', 'TCP_window', 'UDP_len', 'IP_ttl', 'TCP_sport', 'Frame_size', 'UDP_sport', 'TCP_dport']	0.86	0.910189	0.904702	0.901379	0.86883	0.936462	343789.0

Figure 11. Metrics for XGBoost (binary classification) after feature selection

90% test accuracy resulted in slightly lower accuracy but further reduced training time to 0.86 s and model size to 344 KB (Figure 11). For multi-class classification, aiming for > 82% test accuracy, feature selection maintained accuracy, reduced training time to 6.67 s, but slightly increased model size (Figure 12).

Hyperparameter optimization of XGBoost

The final optimization step involved hyperparameter tuning for XGBoost using GridSearchCV on the datasets with the selected number of rows and features. Parameters tuned included eta (learning_rate), gamma (min_split_loss), max_depth, lambda (reg_lambda), and alpha (reg_alpha). The parameter grid searched is exemplified in Figure 13.

The selected grid was designed to cover both conservative and aggressive parameter values, ensuring that the optimisation captured a wide spectrum of possible configurations while remaining computationally feasible. The learning rate (η) was varied between 0.1 and 0.99 to examine the17 trade-off between gradual convergence and rapid training, with lower values expected to improve generalisation and higher values providing faster training at the risk of overshooting. The minimum split loss (γ) was tested in the range 0–3 to assess the effect of regularisation on controlling unnecessary splits. The maximum depth parameter was set between 6 and 18, which allows comparison between relatively shallow trees that reduce overfitting and deeper trees that may capture more complex patterns but increase model size and training time. Finally, the regularisation terms λ (L2) and α (L1) were varied from 0 to 3, following recommendations from related studies, to evaluate their role in penalising model complexity. These ranges ensured that the tuning process considered both model performance and practical constraints, such as training time and memory consumption, which are crucial in IIoT deployment scenarios.

The results are summarized in Table 6. For binary classification, test accuracy improved from 0.904 (optimized rows/features, default params) to 0.924 after tuning, with training accuracy increasing to 0.973. For multi-class classification, test accuracy rose from 0.825 to 0.871, with training accuracy reaching 0.957. While this indicates some increase in overfitting (higher training accuracy compared to test), the improvement in test accuracy was significant. The tuning process itself was time-consuming, taking from approximately 1 to 5 hours. These final accuracies (92.4% for binary, 87.1% for multi-class) represent the best performance achieved for XGBoost, aligning with the figures cited in the abstract.

DISCUSSION

The final hyperparameter optimization of the XGBoost model yielded a significant improvement in performance, achieving over 92% accuracy for binary classification and 87% for multi-class classification. These results are compelling within the context of existing research on anomaly detection in IoT and IIoT. At the same time, it is important to acknowledge that direct cross-paper comparisons are not appropriate, as

	Learning time[s]	Accuracy-train	Accuracy-test	F1-score	Recall	Precision	Size [B]
['IP_flags', 'TCP_flags', 'IP_tos', 'UDP_len', 'Frame_size', 'Service_name', 'TCP_window', 'IP_ttl', 'IP_len', 'TCP_sport', 'TCP_dport', 'TCP_dataofs', 'TCP_ack', 'TCP_seq', 'IP_id', 'UDP_sport']	6.67	0.83134	0.822183	0.825511	0.822183	0.841987	1851556

Figure 12. Metrics for XGBoost (multi-class classification) after feature selection

```
params = {
    "eta" : np.linspace(0.1, 0.99, 5),
    "gamma" : np.linspace(0, 1, 3),
    "max_depth" : [6, 8, 10, 12, 14, 16, 18],
    "lambda" : np.linspace(0, 1, 3),
    "alpha" : np.linspace(0, 1, 3),
}
```

Figure 13. Example hyperparameter grid for XGBoost tuning

prior studies have relied on different datasets and experimental setups. For example, Poornima and Paramasivan [3] reported 86% accuracy in wireless sensor networks, Zhou and Li [5] achieved a 95% detection rate on a DDoS-specific dataset using time-window analysis, and Kisanga et al. [7] obtained 76% accuracy with a Graph Neural Network applied only to DDoS scenarios. Because these works employed different traffic traces, protocols, and attack categories, their results cannot be numerically contrasted with ours. Our intention is therefore not to claim superiority over [3], [5], or [7], but to situate our findings within the broader IIoT anomaly-detection literature. What can be observed consistently is that ensemble tree methods, including RandomForest, ExtraTrees, and XGBoost, show robustness under class imbalance and deliver competitive accuracy without excessive computational cost. This trend is confirmed in our experiments on the CICIoT2023 dataset. In future research, we plan to extend our evaluation to multiple benchmark datasets such as NSL-KDD, CIC-IDS2017, TON_IoT, and IoT-23, using a unified experimental protocol that will allow for direct and statistically meaningful cross-study comparisons.

The journey to this optimized model also underscores the critical trade-offs in designing IIoT security systems. While models like RandomForest initially showed high F1-scores, their tendency to overfit and larger resource requirements made them less practical than the leaner, regularized XGBoost model. The success of data reduction strategies provides a viable path to creating lightweight detection systems that reduce data loads and align with the Edge Computing paradigm. While the results are promising, the study's focus on a single dataset and tree-based algorithms highlights the need for future validation on diverse datasets and comparative analysis with neural network architectures.

Deployment implications and limitations

Beyond the numerical results, several practical aspects are critical for deploying anomaly detection models in Industrial IoT systems. First, resource-awareness is essential: models must operate under strict constraints of memory, processing power, and latency at the network edge. The optimised XGBoost configuration demonstrated competitive accuracy while maintaining relatively small model size and fast training, which supports its applicability in such constrained environments.

At the same time, some limitations must be acknowledged. The evaluation relied solely on the CICIoT2023 dataset, which, while comprehensive, may not fully capture the heterogeneity of real-world IIoT traffic. Further validation on additional datasets and diverse deployment scenarios would strengthen the generalisability of the findings.

One limitation of this study is the use of a 10% dataset extraction for evaluation. This decision was made to balance experimental feasibility with the large size of CICIoT2023 and the computational resources available during the study. While this strategy allowed us to complete a comprehensive set of experiments within the project scope, it restricts direct comparability with some works that report results on the full dataset. In future research, we plan to extend the analysis to the entire dataset to provide even more robust and generalisable results.

Another limitation relates to preprocessing choices. The uniform pipeline of standardisation, while ensuring comparability, may not be equally optimal for all algorithms. In future

Table 6. Comparison of model performance

Model		Model		Tuned model	
Label	Accuracy (train)	Accuracy (test)	Accuracy (train)	Accuracy (test)	Tuning time [s]
Class 0 and 1	0.910	0.904	0.973	0.924	2441.61
Grouped threats	0.831	0.825	0.957	0.871	18737,34

research, algorithm-specific preprocessing strategies could be explored to further improve model performance and efficiency. It should be noted that normalisation had no impact on tree-based models, which are scale-invariant by design. Its inclusion in the preprocessing pipeline was primarily motivated by the need for consistent input across all algorithms.

Moreover, potential risks need to be considered. False negatives remain critical, as missed attacks could have severe operational and security consequences. Conversely, excessive false positives may overwhelm monitoring systems and reduce trust in the detector. Finally, model drift is a realistic challenge in IIoT, where traffic patterns evolve over time. Without periodic retraining or adaptive mechanisms, model performance may degrade. These challenges represent natural directions for future work and highlight the importance of continuous monitoring and adaptive learning strategies in practical deployments.

Practical translation of efficiency gains

The improvements observed in training time and model size have direct implications for deployment in IIoT environments. Reduced training time allows for more frequent model updates, which is important for mitigating model drift and adapting to evolving traffic patterns. A smaller model size lowers the memory footprint, enabling deployment on resource-constrained edge devices such as gateways or embedded controllers. Furthermore, optimised tree depth and learning rate configurations yielded reduced inference latency, which is crucial for near real-time anomaly detection in industrial settings where delays cannot be tolerated. At the same time, it is important to acknowledge the trade-off between model complexity and efficiency: deeper trees may provide marginally higher accuracy but at the cost of increased latency and resource consumption. By balancing these aspects, the proposed optimisation pipeline supports the practical requirements of IIoT deployments.

CONCLUSIONS

This paper presented a systematic methodology for optimizing machine learning models for anomaly detection in Industrial IoT (IIoT) traffic, demonstrating that a practical balance between high detection accuracy and computational efficiency is achievable. The research focused on decision tree-based algorithms and proved that through a holistic optimization pipeline - encompassing data preparation, feature selection, and hyperparameter tuning - it is possible to develop resilient and resource-aware cybersecurity solutions for the growing IIoT landscape.

The research utilized the CIC-IoT Dataset 2023, with features extracted directly from.pcap files to provide a granular view of network behavior. Key findings demonstrate that meticulous data preparation is paramount. Techniques such as class balancing (specifically, RandomUnderSampler was chosen for its efficiency in later stages), appropriate data sampling, feature selection, and data aggregation significantly impact model performance. For instance, data aggregation showed a positive influence on the accuracy of all tested models, particularly AdaBoost. Experiments on reducing the dataset size by selecting an optimal number of rows and features for the XGBoost model proved that it's possible to maintain high performance while considerably decreasing training times and model footprint.

The study placed a particular emphasis on the XGBoost model. Through systematic optimization, including the selection of approximately 270,000–315,000 rows for training (depending on the classification task) and a reduced feature set, followed by rigorous hyperparameter tuning, the XGBoost model achieved notable results. Post-optimization, it yielded detection accuracies exceeding 92% for binary classification (distinguishing attack vs. benign) and 87% for multiclass classification (identifying specific attack types). These results were achieved with significantly reduced data requirements and improved computational efficiency, such as shorter training times (e.g., binary XGBoost training time reduced from over 8 s to under 1 s after row/feature selection) and smaller model sizes.

A significant contribution of this work lies in highlighting that refining existing, well-understood machine learning algorithms through careful data handling and parameter tuning can lead to highly effective and efficient anomaly detection systems, often outperforming or matching more complex approaches without the need for designing entirely new architectures. The findings underscore the critical role of dataset selection and tailored feature engineering in the success of ML-based security solutions.

While the results are promising, this study primarily focused on decision tree-based algorithms. Neural networks, with their distinct architectures and learning capabilities, were not explored but represent a potential avenue for future comparison. The current findings are based on a specific, albeit comprehensive, dataset; further validation on other diverse IIoT datasets or real-world deployments would be beneficial.

Future research could extend this work by:

- Investigating the application and optimization of various neural network architectures (e.g., CNNs, RNNs, Autoencoders) for this task and comparing their performance against the optimized tree-based models.
- Exploring more advanced feature engineering and automated feature selection techniques.
- Developing adaptive or online learning models capable of evolving with changing network patterns and new attack vectors in dynamic IIoT environments.
- Conducting real-world deployment and performance analysis of the optimized models in operational IIoT systems.

In conclusion, this research successfully demonstrated that optimized machine learning models, particularly XGBoost, can achieve a strong balance of high detection accuracy and computational efficiency for anomaly detection in IIoT systems. The presented methodologies for data preparation and model tuning offer practical pathways to developing more resilient and resource-aware cybersecurity solutions for the growing IIoT landscape.

To our knowledge, this is one of the first works to systematically evaluate the optimisation of tree-based models for anomaly detection in IIoT environments, explicitly balancing detection performance with resource-awareness, which is crucial for deployment at the network edge.

REFERENCES

1. Neto EC, Dadkhah S, Ferreira R, Zohourian A, Lu R, Ghorbani AA. CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment. *Sensors*. 2023;23(13):5941. https://doi.org/10.3390/s23135941
2. Tavallaee M, Bagheri E, Lu W, Ghorbani AA. A detailed analysis of the KDD CUP 99 data set. In: Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications. 2009.
3. Poornima IG, Paramasivan B. Anomaly detection in wireless sensor network using machine learning algorithm. *Computer Communications*. 2020;151:33–337. https://doi.org/10.1016/j.comcom.2020.01.005
4. Suthaharan S, Alzahrani M, Rajasegarar S, Leckie C, Palaniswami M. Labelled data collection for anomaly detection in wireless sensor networks. In: *2010 Sixth International Conference on Intelligent Sensors, Sensor Networks and Information Processing*; 2010; 269–274. https://doi.org/10.1109/ISSNIP.2010.5706782
5. Zhou Y, Li J. Research of Network Traffic Anomaly Detection Model Based on Multilevel Autoregression. In: *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*; 2019; 380–384. https://doi.org/10.1109/ICCSNT47585.2019.8962517
6. Fouad MA, Abdel-Hamid AT. On Detecting IoT Power Signature Anomalies using Hidden Markov Model (HMM). In: *2019 31st International Conference on Microelectronics (ICM)*; 2019; 108–112. https://doi.org/10.1109/ICM48031.2019.9021483
7. Kisanga P, Woungang I, Traore I, Carvalho GHS. Network Anomaly Detection Using a Graph Neural Network. In: *2023 International Conference on Computing, Networking and Communications (ICNC)*; 2023; 61–65. https://doi.org/10.1109/ICNC57223.2023.10074111
8. Erhan L, Ndubuaku M, Di Mauro M, Song W, Chen M, Fortino G, Bagdasar O, Liotta A. Smart anomaly detection in sensor systems: A multi-perspective review. *Information Fusion*. 2021;67:64–79. https://doi.org/10.1016/j.inffus.2020.10.001
9. Bhuyan MH, Bhattacharyya DK, Kalita JK. Network Anomaly Detection: Methods, Systems and Tools. *IEEE Communications Surveys And Tutorials*. 2014;16(1):303–336. https://doi.org/10.1109/SURV.2013.052213.00046
10. Géron A. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Incorporated; 2019.
11. Grinsztajn L, Oyallon E, Varoquaux G. Why do tree-based models still outperform deep learning on tabular data?. *arXiv preprint arXiv:2207.08815*. 2022.
12. Lemaître G, Nogueira F, Aridas CK. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*. 2017;18(17):1−5.
13. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*. 2002;16:321–357. https://doi.org/10.1613/jair.953

14. Han H, Wang WY, Mao BH. Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In: Huang DS, Zhang XP, Huang GB, editors. *Advances in Intelligent Computing. ICIC 2005*. Lecture Notes in Computer Science, vol 3644. Springer, Berlin, Heidelberg; 2005. https://doi.org/10.1007/11538059_91
15. He H, Bai Y, Garcia EA, Li S. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. 2008; 1322–1328. https://doi.org/10.1109/IJCNN.2008.4633969
16. Brownlee J. Undersampling Algorithms for Imbalanced Classification. MachineLearningMastery.com. Jan 27, 2020.
17. Geapa B, Prati RC, Monard MC. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor Newsl*. 2004;6(1):20–29. https://doi.org/10.1145/1007730.1007735
18. Ho TK. Random decision forests. In: *Proceedings of 3rd International Conference on Document Analysis and Recognition*. 1995;1:278–282. https://doi.org/10.1109/ICDAR.1995.598994
19. Geurts P, Ernst D, Wehenke L. Extremely randomized trees. *Machine Learning*. 2006 Apr;63(1):3–42. https://doi.org/10.1007/s10994-006-6226-1
20. Freund Y, Schapire RE. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*. 1997;55(1):119–139. https://doi.org/10.1006/jcss.1997.1504
21. Chen T, Guestrin C. XGBoost: A Scalable Tree Boosting System. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016; 785–794. https://doi.org/10.1145/2939672.2939785
22. Prokhorenkova L, Gusev G, Vorobev A, Dorogush AV, Gulin A. CatBoost: unbiased boosting with categorical features. In: *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*. 2018; 6639–6650.
23. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*. 2011;12:2825–2830.
24. Moustafa N, Slay J. 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; 1–6.
25. Garcia S, Erquiaga MJ, Parmisano A. IoT-23: A labeled dataset with malicious and benign IoT network traffic. Stratosphere Laboratory, AIC Group, Czech Technical University in Prague. 2020.
26. Ring M, Wunderlich S, Grüdl D, Landes D, Hotho A. A survey of network-based intrusion detection data sets. Computers & Security. 2019;86:147–167.
27. Kuraś P, Organiściak P, Kowal B et al. IoT communication security: challenges and solutions. ZN SGSP 2024;2(89):69–90. https://doi.org/10.5604/01.3001.0054.3829
28. Bolanowski, M., Paszkiewicz, A., Rumak, B. (2021). Coarse Traffic Classification for High-Bandwidth Connections in a Computer Network Using Deep Learning Techniques. In: Barolli, L., Yim, K., Enokido, T. (eds) Complex, Intelligent and Software Intensive Systems. CISIS 2021. Lecture Notes in Networks and Systems, vol 278. Springer, Cham. https://doi.org/10.1007/978-3-030-79725-6_13
29. Bădică, A. et al. (2022). Cascaded Anomaly Detection with Coarse Sampling in Distributed Systems. In: Sachdeva, S., Watanobe, Y., Bhalla, S. (eds) Big-Data-Analytics in Astronomy, Science, and Engineering. BDA 2021. Lecture Notes in Computer Science(), vol 13167. Springer, Cham. https://doi.org/10.1007/978-3-030-96600-3_13
30. Sezgin A, Boyacı A. Enhancing intrusion detection in industrial internet of things through automated preprocessing. Advances in Science and Technology Research Journal. 2023;17(2):167–176. https://doi.org/10.12913/22998624/162004
31. Sokolova M, Lapalme G. A systematic analysis of performance measures for classification tasks. Information Processing & Management. 2009;45(4):427–437. https://doi.org/10.1016/j.ipm.2009.03.002
32. Powers D.M.W. Evaluation: From precision, recall and f-measure to ROC, informedness, markedness & correlation. Journal of Machine Learning Technologies. 2011;2(1):37–63.
33. Davis J, Goadrich M. The relationship between Precision-Recall and ROC curves. Proceedings of the 23rd International Conference on Machine Learning (ICML). ACM; 2006;233–240. https://doi.org/10.1145/1143844.1143874
34. Chawla N.V., Bowyer K.W., Hall L.O., Kegelmeyer W.P. SMOTE: Synthetic minority oversampling technique. Journal of Artificial Intelligence Research. 2002;16:321–357. https://doi.org/10.1613/jair.953
35. He H, Garcia E.A. Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering. 2009;21(9):1263–1284. https://doi.org/10.1109/TKDE.2008.239

36. Fawcett T. An introduction to ROC analysis. Pattern Recognition Letters. 2006;27(8):861–874. https://doi.org/10.1016/j.patrec.2005.10.010
37. Chen, Z., Li, Z. W., Huang, J., Liu, S., Long, H. X. (2024). An effective method for anomaly detection in Industrial Internet of Things using XGBoost and LSTM. Scientific Reports, https://doi.org/10.1038/s41598-023-44448-1
38. Ayad, A. G., El-Gayar, M. M., Hikal, N. A., Sakr, N. A. (2025). Efficient real-time anomaly detection in IoT networks using one-class autoencoder and deep neural network. Electronics, 14(1), Article 104. https://doi.org/10.3390/electronics14010104
39. Kusumastuti, A. F., Rangelov, D., Lammel, P., Boerger, M., Aleksandrov, A., Tcholtchev, N. (2025). Anomaly Detection in IoT Networks: Performance comparison of transformer, 1D-CNN, and GrowNet models on the Bot-IoT dataset. In: 14th International Conference on Data Science, Technology and Applications. https://doi.org/10.5220/0013637600003967
40. Al-Qudah, M., AlMahamid, F. (2025, May 22). A Multi-Step Comparative Framework for Anomaly Detection in IoT Data Streams. arXiv. https://arxiv.org/abs/2505.16872