Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2023.040287
Article

AID4I: An Intrusion Detection Framework for Industrial Internet of Things Using Automated Machine Learning

Anıl Sezgin[1,2,*] and Aytuğ Boyacı[3]

[1]ATASAREN, National Defence University, Istanbul, 34334, Turkey
[2]Software in Motion Department, Siemens Advanta, Siemens Corporate Technology, Istanbul, 34870, Turkey
[3]Department of Computer Engineering National Defence University, Air Force Academy, Istanbul, 34149, Turkey
*Corresponding Author: Anıl Sezgin. Email: anil.sezgin@siemens.com
Received: 12 March 2023; Accepted: 09 June 2023; Published: 30 August 2023

Abstract: By identifying and responding to any malicious behavior that could endanger the system, the Intrusion Detection System (IDS) is crucial for preserving the security of the Industrial Internet of Things (IIoT) network. The benefit of anomaly-based IDS is that they are able to recognize zero-day attacks due to the fact that they do not rely on a signature database to identify abnormal activity. In order to improve control over datasets and the process, this study proposes using an automated machine learning (AutoML) technique to automate the machine learning processes for IDS. Our groundbreaking architecture, known as AID4I, makes use of automatic machine learning methods for intrusion detection. Through automation of preprocessing, feature selection, model selection, and hyperparameter tuning, the objective is to identify an appropriate machine learning model for intrusion detection. Experimental studies demonstrate that the AID4I framework successfully proposes a suitable model. The integrity, security, and confidentiality of data transmitted across the IIoT network can be ensured by automating machine learning processes in the IDS to enhance its capacity to identify and stop threatening activities. With a comprehensive solution that takes advantage of the latest advances in automated machine learning methods to improve network security, AID4I is a powerful and effective instrument for intrusion detection. In preprocessing module, three distinct imputation methods are utilized to handle missing data, ensuring the robustness of the intrusion detection system in the presence of incomplete information. Feature selection module adopts a hybrid approach that combines Shapley values and genetic algorithm. The Parameter Optimization module encompasses a diverse set of 14 classification methods, allowing for thorough exploration and optimization of the parameters associated with each algorithm. By carefully tuning these parameters, the framework enhances its adaptability and accuracy in identifying potential intrusions. Experimental results demonstrate that the AID4I framework can achieve high levels of accuracy in detecting network intrusions up to 14.39% on public datasets, outperforming traditional

intrusion detection methods while concurrently reducing the elapsed time for training and testing.

Keywords: Automated machine learning; intrusion detection system; industrial internet of things; parameter optimization

1 Introduction

The concept of Internet of Things (IoT) is used in many areas, including health, defense, transportation, agriculture, smart cities, smart homes, wearable technologies, etc. The IoT involves the integration of intelligent devices and management systems to achieve practical goals. The adoption of IoT technology has significantly changed our lives.

The implementation of IoT in industrial settings has given rise to the concept of IIoT. The interconnection of devices is not a novel idea in the industry, as they require coordination and collaboration to function effectively. The exchange of data and communication between devices is common but usually restricted within a manufacturing facility or a specific region of the industry. Supervisory Control and Data Acquisition (SCADA) systems collect data from remote sensors and industrial equipment and transmit it to a central location for monitoring or control [1].

Industrial IoT refers to the use of sensors, data analytics, and smart machines to enhance scalability, efficiency, and interoperability in critical infrastructure, leading to improved automation and increased corporate productivity [2]. To attain the objective of increased productivity, certain obstacles must be overcome. The highest priority is the security of industrial infrastructure and its components. Cyberattacks on crucial infrastructure can lead to substantial financial losses for industries.

There are different ways IIoT nodes can connect to the internet. Examples of these ways are Tranmission Control Protocol/Internet Protocol (TCP/IP), Message Queuing Telemetry Transport (MQTT), Modbus TCP, a cellular connection, LoRaWaN. IIoT nodes, capable of collecting, processing, and transmitting data, can also be vulnerable to privacy and security threats that may compromise the system [3]. A key feature of IIoT nodes is that they are always active during data collection, processing and transmission. IIoT systems basically consist of the following layers: sensor/perception layer, application layer, network layer and cloud. At each layer, there are various attack and penetration methods that can compromise systems with the IIoT. The growing reliance on IIoT systems has made them a target for cyber-attacks. Some of the most common hacking and infiltration methods used against IIoT systems include access control attacks, where unauthorized individuals attempt to gain access to sensitive data or systems; data corruption breaches, which involve tampering with data in transit or storage; spoofing attacks, in which an attacker poses as a trusted source to trick a system into providing access or divulging sensitive information; and denial of service (DoS) and distributed denial of service (DDoS) attacks, which involve overwhelming a system with traffic to make it unavailable to users. It is important for IIoT systems to have robust security measures in place to prevent such threats and ensure the privacy and integrity of sensitive data. To counter these attacks and to ensure the security of IIoT nodes, many organizations use intrusion detection systems.

Intrusion detection is an illegal attempt that affects the integrity, privacy, availability of the network [4]. Traditional intrusion detection system are not effective systems because of the wide variety of intrusions. Therefore, it is very important to develop efficient and robust systems according to modern requirements.

Intrusion Detection Systems use various techniques to identify and analyze potential cyber-attacks. Signature-based IDS work by comparing network data with pre-existing attack patterns stored in a database. When a match is found, the system alerts administrators by triggering an alarm. Nevertheless, a major drawback of this approach is its inability to recognize novel and unidentified attacks if they are not included in the database. On the other hand, rule-based or anomaly-based IDS store normal network behavior in a database and trigger an alarm when there is a deviation from established rules. This method can detect previously unknown attacks. Hybrid IDS systems blend the advantages of both signature-based and anomaly-based IDS to detect attacks. Despite their versatility, traditional IDS systems have limitations such as a high rate of false positives and low accuracy in detecting attacks.

Attackers develop new attack methods and software they use every day. The significance of intrusion detection systems is growing daily, and advancements are being made to enhance their ability to defend against malicious software and improve the effectiveness of network systems. Research in this field is abundant and new studies are constantly being conducted to enhance the performance of intrusion detection systems.

Intrusion Detection Systems play a crucial role in monitoring real-time internet traffic and identifying any abnormal behavior [5]. The task of determining traffic behavior and the type of network attacks can be approached as both a binary classification problem and a multi-class problem [6]. With the increasing demand for accurate and efficient security solutions, there has been a growing interest in improving the attack prediction accuracy of existing IDS classifiers. The application of machine learning techniques in IDS has become an active area of research in recent years. Researchers aim to develop new and improved methods to improve the performance of IDS in detecting and preventing network attacks [7]. The objective is to enhance security against cyber-attacks and secure sensitive data by utilizing machine learning technology [8].

Developing an AutoML framework for intrusion detection in Industrial Internet of Things (IIoT) can bring several benefits. These are primary causes:

- Increased Efficiency: AutoML framework automates the building of intrusion detection models, reducing time and effort. It allows rapid model development, deployment, and monitoring, improving the effectiveness of identifying potential attacks.
- Enhanced Accuracy: AutoML framework incorporates machine learning algorithms that can evaluate massive amounts of data, identify complicated patterns, and learn from the past. By lowering false positives and false negatives, this can improve intrusion detection accuracy in comparison to traditional rule-based techniques.
- Scalability: With numerous devices producing enormous volumes of data, IIoT networks can have a very large scale. By automating the model development and training process, enabling effective use of computational resources, and ensuring scalability as the network grows, autoML frameworks can manage this amount of data.
- Reduced Expertise Requirements: The design and maintenance of rules for traditional intrusion detection systems frequently requires extensive domain knowledge. AutoML framework automates a lot of the technical complexity, which reduces the necessity of specific knowledge and makes it possible for non-experts to create efficient intrusion detection models.

The scope of this study is to construct an AutoML framework that focuses on intrusion detection for IIoT systems. Section 2 outlines the research methodology employed in this study. In Section 3, the focus will be on the field of machine learning and AutoML. This section will provide a comprehensive overview of these technologies and their applications in various industries. It will cover

the various algorithms and techniques used in machine learning, including supervised, unsupervised, and reinforcement learning. In Section 4, AID4I framework is introduced for automating the machine learning pipeline. This framework aims to automate all the steps involved in the machine learning process, from preprocessing to model selection, to achieve the best performance on a given dataset.

On Section 5 the results of the tests are shared. We conclude the study in Section 6 and in the 7th Section what can be added to this work in the future and how it can be improved are explained.

2 Methodology

Traditional intrusion detection methods, such as signature-based detection and rule-based systems, often struggle to keep pace with the rapidly evolving landscape of cyber threats. These methods rely on predefined patterns and rules, making them vulnerable to novel and sophisticated attack techniques. As a result, they can generate a significant number of false positives or miss new and emerging threats altogether.

Several benefits may arise from addressing this issue and enhancing the precision and effectiveness of intrusion detection systems. It can improve an organization's overall security posture, lowering the danger of successful assaults and potential harm. Additionally, it may allow for more prompt and proactive reactions to identified breaches, enabling firms to lessen the effect and restrict the scope of a breach. Organizations can optimize training and testing times to increase efficiency, utilize fewer resources, and reduce cost.

Increased efficiency in the development and deployment of intrusion detection models is one advantage of using AutoML. These frameworks automate the selection of models, hyperparameter tuning, and feature engineering. Because of the automation, building models and deployment can happen more quickly because human experts are not required to put in as much manual effort. As a result, organizations can react to new threats more quickly, ensuring the quick identification and containment of network intrusions.

Improved accuracy in intrusion detection is frequently achieved using autoML approaches. AutoML may analyze a variety of models and configurations in order to identify the most suitable for the task at hand by utilizing sophisticated algorithms and search techniques. Traditional approaches might miss the best solutions, but AutoML is capable of identifying them, leading to more precise intrusion detection models. This increase in accuracy contributes to a decrease in false positives and false negatives, improving the intrusion detection system's overall efficiency.

AutoML can be explained as a large and versatile optimization problem. It can be expressed as a solution space that can produce predictions for a dataset according to a certain computational process without human intervention. AutoML enables the automatic construction of the machine learning pipeline to achieve optimum performances on a dataset. It can automate difficult task and time-consuming tasks such as data preprocessing, model selection, and hyperparameter optimization and create pipelines. The steps of the AutoML process in the most general form are shown in Fig. 1.

Hyperparameter optimization is also an import problem of the machine learning process [9]. Because hyperparameter optimization is used to explore hyperparameter configurations that can work best in different datasets [10], to design pipelines that can be used in certain areas, and to contribute to the development of what is offered by default in common machine learning libraries [11,12].

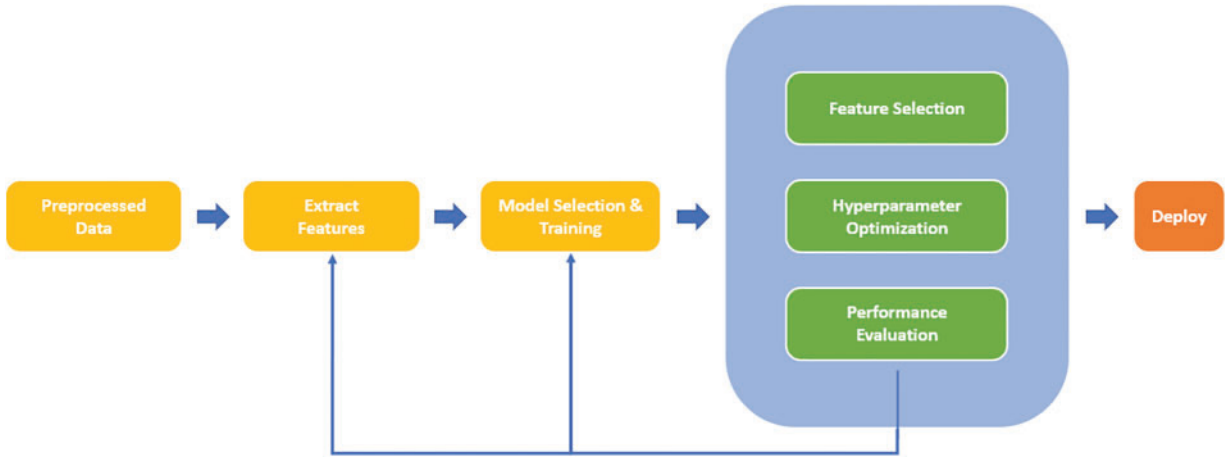

Figure 1: The steps of AutoML

Given the critical nature of network intrusion detection and the limitations of existing approaches, this research aims to propose and evaluate the effectiveness of the AID4I framework in achieving higher accuracy levels and reducing training and testing time. The following sections will outline the methodology employed to conduct this study and provide detailed insights into the experimental setup, data collection, and evaluation metrics used to assess the performance of the AID4I framework.

3 Automated Machine Learning

Machine learning enables systems to improve and strengthen their capability to learn from experiences and make decisions without human intervention [13]. Top level machine learning approaches are examined in two categories as supervised and unsupervised. However, at the granular level, machine learning is divided into 4 categories: supervised, semi-supervised, unsupervised, and reinforcement. Supervised machine learning methods learn from a labeled dataset to make predictions for the future. If the data is unlabeled, unsupervised machine learning techniques are used. Semi-supervised machine learning techniques utilize a mixture of data with and without annotations in the learning phase. Reinforcement learning algorithms, on the other hand, determine rewards or penalties based on interactions within a defined environment [14].

AutoML aims to use the machine learning process in a smooth workflow by eliminating the cost of many trial and error processes that the hyperparameter problem will do in line with the experience of the individual [15,16]. They listed the automated steps within the scope of AutoML as follows:

1. Automated data preparation
   - Detection of column-based attribute types (categorical, numeric, etc.)
   - Detection of column-based attribute role
   - Task detection (binary classification, clustering, etc.)
2. Automated feature engineering
   - Selection and extraction of features
   - Meta learning and transfer learning
3. Automated model selection
4. Hyperparameter optimization
5. Automated pipeline selection

6. Automated selection of performance evaluation metrics and validation methods
7. Automated issue detection
   - Leak detection
   - Incorrect configuration detection
8. Automatic evaluation of obtained outcomes.
9. User interfaces and visualization for the machine learning process

It is seen that AutoML produces impressive results [15]. In AutoML studies, it's very important to be able to explain how the system works. Because this automated process should be able to meet both the questionability of the success of the constructed model and the concept of explainability in artificial intelligence. For this purpose, pipelines that show how the process progresses in AutoML studies must be defined. Fig. 2 show the pipeline of a classic AutoML process.

Figure 2: Pipeline of a classic AutoML process

So far, different AutoML studies have been done and various frameworks have been developed and in literature there are studies on automatic machine learning model selection. However, most of these studies have focused on some parts of AutoML pipeline. In study [16] compared in 150 supervised classification tasks. As a result of the benchmarking, it was stated that TPOT (Tree-based Pipeline Optimization Tool) outperformed a basic machine learning analysis on 21 classification tasks. In study [17], they created classification pipeline by building on scikit-learn but only traditional machine learning methods such as SVM (Support Vector Machine) and kNN (k-Nearest Neighbors) are being used. Auto-Keras [18] is an open-source library focused on searching for deep learning models developed based on Keras framework. Study [19] includes a comprehensive review of the Oracle AutoML platform which has been proposed as a fast and predictive AutoML pipeline. In this automated machine learning model, algorithm selection, adaptive sampling, feature selection and then hyperparameter tuning are performed. It has been noted that a feed-forward approach generates superior results in a shorter amount of time compared to state-of-the-art open source AutoML tools like H2O and Auto-Sklearn. Many AutoML algorithms work on fixed datasets to solve certains tasks. However, an AutoML system should possess the ability to continually learn and have the capability to advance the entire process autonomously.

Our research on the development of an AutoML framework with advanced features such as imputation for missing values, hybrid feature selection using Shapley values and genetic algorithms, and hyperparameter optimization with grid search holds significant importance in the field of machine learning. Firstly, the inclusion of imputation for missing values addresses a common challenge faced in real-world datasets, where missing data can hinder the performance of machine learning models. By incorporating an imputation method within the AutoML framework, the quality and integrity of the data can be enhanced while leading to improved model performance and more accurate predictions. Secondly, the hybrid feature selection approach combining Shapley values and genetic algorithms offers a novel and powerful method for identifying the most informative features within a dataset. Shapley values provide a rigorous and theoretically sound approach to quantify the contribution of each feature to the model's performance, enabling the elimination of irrelevant or redundant features. Integrating genetic algorithms further enhances the feature selection process by exploring

different combinations of features and optimizing the selection based on specific evaluation criteria. This combination of techniques enables the AutoML framework to automatically select the most relevant features, reducing dimensionality, improving interpretability, and potentially enhancing the generalization capability of the machine learning models.

4 Experimental Evaluation

In this study, it is aimed to develop an AutoML system that can take part in intrusion detection for IIoT systems. During the development of the AutoML system, priority was given to the X-IIoTID dataset. This dataset, which is suitable for the classification task, has 59 features and 820,834 instances. With this dataset, a clear and precise attack classification can be made and it includes new generation attacks such as WebSocket fuzzing, Constrained Application Protocol (CoAP) resource discovery, MQTT malicious subscription, crypto-ransomware. It also covers protocols such as Modbus, WebSocket, MQTT, TCP, Address Resolution Protocol (ARP), Hyper-Text Transfer Protocol (HTTP), Secure Shell (SSH), Domain Name System (DNS), Simple Mail Transfer Protocol (SMTP), User Datagram Protocol (UDP). It is a more comprehensive dataset than other dataset created for IIoT and IDS [20].

The AutoML pipeline developed within the scope of this study consists of the following stages.

1. Preprocessing data
2. Model selection
   - Feature selection
   - Parameter tuning
1. Performance/model comparison
2. Deployment

4.1 Preprocessing

Data preparation is the first step in the machine learning process. Data preparation basically consists of 3 parts: data collection, cleaning, and augmentation. Data collection is one of the necessary steps to obtain a dataset or to expand an existing dataset. Although it is natural to have noise in datasets and this may negatively affect model training. In the data cleaning process, noisy data that may cause a decrease in accuracy rates in model training, are cleaned. Therefore, data cleaning should be done [9]. The data augmentation process is an important step to increase the performance and robustness of the model. Data augmentation can also be considered as a data gathering tool, as it generates new data from existing data. It can also be used to prevent the model from being overfitted. As the size of the data increases, the accuracy and speed of models based on machine learning are significantly affected. Data preprocessing is a phase that consists of transforming raw data, so that problems arising from incompleteness and inconsistencies in the data set are resolved. At this stage, it is aimed to obtain a more understandable data set.

Preprocessing is a crucial stage in Automated Machine Learning as it involves transforming raw data into a format that is suitable for building machine learning models. The following are three important preprocessing steps after data preparation that should be performed in AutoML: handling missing values, scaling, and encoding. Data collected in real-world scenarios often contain missing values, which can negatively impact the performance of machine learning models. To overcome this challenge, different techniques such as imputation (filling in the missing values with a value such as the mean of median of the column) or deletion (removing the entire row or column with missing values) can be used. The imputation method selected is dependent on the data type, the degree of

missing values, and the distribution of the data. In some machine learning algorithms, the magnitude of the features can impact their contribution to the model. To ensure that all features are given equal consideration, scaling is performed to bring all features to the same maginute. This is typically achieved by normalizing the features to have a mean of zero and a standard deviation of one. Machine learning algorithms work with numerical data and therefore categorical variables need to be transformed into numerical form. Common encoding techniques include one-hot encoding, label encoding and the ordinal encoding. The choice of encoding method depends on the type of categorical data and the specific requirements of the machine learning algorithm used.

4.2 Feature Selection

Feature engineering is a crucial step in the machine learning pipeline that involves transforming raw data into a format that can be easily understood and analyzed by algorithms and models. This step is crucial for the AutoML pipeline, as the quality of the attributes has a significant impact on the performance of a model.

Feature selection involves creating a subset of the original features by eliminating redundant ones to expedite the training of the model. This process improves model performance by avoiding overfitting. The selected attributes are often different and hight.

Feature selection provides benefits in creating simpler and more understandable models, preparing more understandable data and increasing result performance. Feature selection has been used for many years in different tasks such as eliminating noisy irrelevant, and redundant features for image recognition, addressing language difficulties such as abbreviations, misspellings, and synonyms for natural language processing, reducing processing cost, minimizing storage, and providing better understanding of test data for intrusion detection.

Feature selection reduces irrelevant or redundant features, creating a subset based on the original feature set. In this way, the model can be simplified, preventing overfitting and improving model performance. According to the study [21], the feature selection process consists of 4 basic steps as shown in Fig. 3 below.

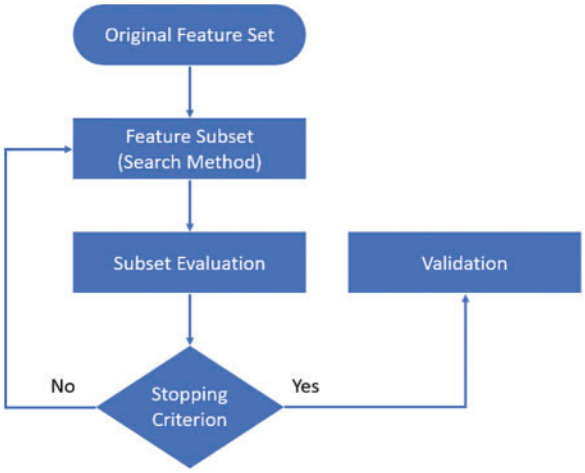

Figure 3: Basic steps of feature selection process

The feature selection process involves a systematic evaluation of a subset of features selected using a search method. The selected subset is then validated through a validation process to determine its validity. This process is repeated until a stopping criterion is met.

In feature selection, three types of search methods are utilized: complete search, heuristic search, and random search. The complete search can be divided into exhaustive and non-exhaustive searches. The non-exhaustive search has various techniques including breadth-first, beam search, branch and bound, and best-first search methods.

Heuristic search involves the use of specific methods. One such method is sequential forward selection (SFS). Another is sequential backward selection (SBS), and yet another is bidirectional search (BS). The SFS and SBS methods involve adding properties to an empty set or removing them from a full set. In the BS method, these two algorithms are used in combination to search until the same subset is obtained. Commonly used random search methods are genetic algorithms and particle swarm optimization.

Classifying feature selection methods can be done into three broad categories: filtering, wrapper, and embedded. Filtering methods use only statistical information for selection, while wrapper methods employ searches that are based on the features themselves. On the other hand, embedded methods focus on finding the best criterion for division [22].

In this study, 2 different methods were used for feature selection. One of them is Shapley Additive Explanations (SHAP) and the other is Genetic Algorithm.

Shapley Values, introduced by Lloyd Shapley [23] in the field of cooperative game theory, provide a means of measuring each player's contribution to a given game. Game theory is a strategic analysis of how two or more players interact in a situation where their outcomes are interdependent. Shapley Values are particularly relevant in situations where players' contributions are unequal, but they still work together to produce collective results or profits.

The Shapley Values method has since been adapted for use in interpreting the predictions made by machine learning models, through the creation of the SHAP method. The SHAP method, as presented in [24], computes the Shapley values for each feature of a sample under analysis, giving insights into the contribution of each feature to the final prediction. This enables greater transparency in the decision-making processes of machine learning models, allowing for a better understanding of how they arrive at their predictions. Figs. 4 and 5 show the most important 20 features by using SHAP and different classification models.

Genetic algorithm (GA) which is one of the optimization algorithms, is likened to the intergenerational transition process of human genes. Genetic Algorithm (GA) is a probabilistic optimization technique that mimics the process of natural selection and genetics, for finding the best solution to a particular problem.

The GA technique is a well-used optimization strategy in the realm of machine learning models for selecting features. The process of identifying crucial or relevant features for a specific machine learning model through the use of a genetic algorithm (GA) is known as feature selection with GA. The main objective is to reduce the dimensionality of the data while preserving the performance of the model. In the GA-based feature selection, the genetic algorithm is used to optimize the hyperparameters, such as the population size, the crossover and mutation rates, and the selection criteria, to determine the optimal combination of features. The fitness function used in GA-based feature selection evaluates the performance of the machine learning model using a specific set of features, and the best-performing features are selected and combined to form a new generation. GA feature selection is an iterative

process that is performed until a performance threshold that is suitable has been reached or the termination criterion is met. This process aims to reduce the risk of overfitting and improves the model's ability to generalize by finding the crucial features that have a significant impact on how well the machine learning model performs. Population information, fitness value, parent selection, crossover, and mutation are some of the stages that make up the algorithm. Based on the population statistics, a suitable population for the data is produced at random in the first stage. The fitness value of each solution (chromosome) is determined in the second stage. In the third step, the best chromosomes are chosen as parents. As a result, new people are born and genes are spread. In the fourth stage, the parents are combined to form a new set of chromosomes through crossover. This newly formed population is then added to the population set. In the final stage, mutations are made to the chromosomes in the population set.

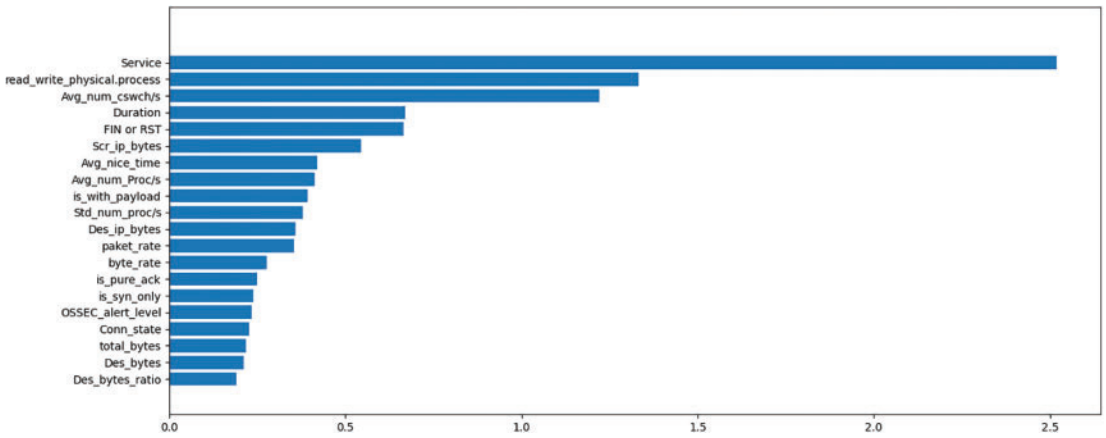

Figure 4: Feature selection with SHAP and XGBoost

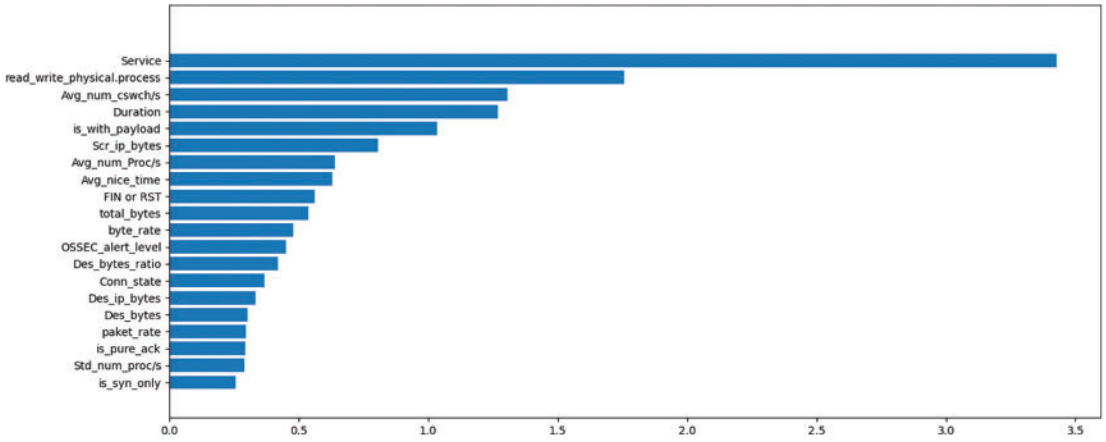

Figure 5: Feature selection with SHAP and LightGBM

The main advantage of using GA for feature selection is its ability to search for the optimal feature subset by considering the interactions between features, rather than just evaluating the individual features. Moreover, GA can handle a large number of features and can provide a robust solution to the feature selection problem. However, GA requires a large number of function evaluations, which can be computationally expensive, especially when dealing with high-dimensional datasets. Therefore,

it is important to carefully choose the evaluation metric and set appropriate termination criteria to balance the trade-off between computational efficiency and solution quality.

The steps of the genetic algorithm are shown in Table 1, while Table 2 demonstrates the application of the genetic algorithm in feature selection by using the CatBoost classification method. The algorithm selects the features used in each iteration and calculates the accuracy values obtained when using those features. In conclusion, the genetic algorithm is an efficient method for feature selection in machine learning models.

Table 1: Steps of genetic algorithm for feature selection

Steps of Genetic Algorithm for Feature Selection
1. Initialization: random population.
2. Adaptation: fitness value is calculated for each chromosome, fitness values indicate the solution quality of the sequences, chromosome is a set of features.
3. Selection: Selection is based on the survival of the individuals whose fitness value is calculated by passing them through a certain selection operator.
4. Crossover: Crossover operators are used to create new individuals from individuals determined with the help of selection methods.
5. Mutation: The aim here is to create diversity by preventing the formation of similar individuals in future generations, and to expand the search space and ensure that the best and strongest individuals are revealed with a more detailed search.

Table 2: Sample of selected features and accuracy with CatBoost

Feature no	Accuracy
1, 3, 4, 5, 7, 8, 11, 12, 13, 14, 16, 18, 19, 20, 23, 24, 26, 27, 28, 29, 31, 34, 35, 36, 37, 39, 41, 43, 45, 47, 50, 51, 52, 53, 54, 57, 58, 59	0.9963
1, 3, 4, 5, 6, 7, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21, 24, 25, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 41, 43, 45, 46, 47, 48, 49, 50, 51, 52, 54, 55, 58, 59	0.9964
1, 2, 3, 4, 6, 12, 13, 15, 16, 17, 20, 23, 24, 26, 28, 29, 30, 31, 32, 34, 35, 36, 38, 45, 47, 49, 51, 52, 54, 55, 57, 58, 59	0.9965
2, 3, 4, 5, 6, 7, 11, 13, 14, 16, 17, 18, 21, 23, 24, 25, 27, 29, 30, 31, 32, 33, 34, 36, 39, 41, 43, 44, 45, 48, 49, 50, 51, 52, 53, 55, 57, 58, 59	0.9966
1, 2, 3, 4, 6, 7, 11, 12, 15, 16, 17, 18, 19, 21, 22, 24, 25, 26, 28, 29, 30, 33, 34, 38, 41, 43, 44, 48, 49, 50, 51, 52, 53, 55, 57, 58, 59	0.9967

4.3 Hyperparameter Tuning

Machine learning focuses on model building to make predictions and decisions by taking training data as input. It is a common problem in machine learning that parameter values are needed to be set before the training process for both supervised and unsupervised learning algorithms. These parameters are called hyperparameters. Hyperparameters are used to configure many results of a given learning algorithm, such as learning rate, kernel parameters, network architecture. Hyperparameters should be given in an optimized way in order to get results with a high success rate while creating the

model. Hyperparameter tuning is an essential step in the machine learning process that requires careful consideration and selection of these parameters. The selection of hyperparameters has a significant impact on the performance of the model, and it can affect the accuracy, generalization capability, and the speed of the learning process. The main goal of hyperparameter tuning is to find the optimal combination of hyperparameters that result in the best performance of the model. The steps of the hyperparameter tuning algorithm are shown in Table 3.

Table 3: Hyperparameter tuning algorithm

Hyperparameter Tuning Algorithm
initialize dataset
handle missing values (mean/median/most frequent)
apply label encoder
choose scaler (standard/minmax/robust)
Save scaler pkl file
Select features SHAP/Genetic Algorithm (model based)
for all classification models:
//use grid searchs
foreach parameter in grid:
apply parameters and calculate with selected features
compare calculated accuracy with previous accuracies
save the highest accuracy with parameter information to the database
get all hyperparameters information from the database with ordered accuracies
use scaler pkl file
create models with selected hyperparameters
save model pkl file

Hyperparameter optimized models used in AID4I are as follows:

- AdaBoost
- CatBoost
- Decision Tree
- Gaussian Process
- Gradient Boosting
- K-Nearest Neighbor
- LightGBM
- Neural Network
- Perceptron
- Random Forest
- Ridge
- Stochastic Gradient Descent
- SVM
- XGBoost

In proposed framework, Grid Search method was preferred for hyperparameter tuning. The Grid Search method aims to find the most suitable parameter values by trying all the values in order in the range determined for the classification algorithms. This method is considered as one of the most accurate analysis methods that can be used in the extraction and optimization of optimal hyperparameters. In the most recent applications, it is seen that the Grid Search and Cross Validation methods is frequently preferred. It is one of the most preferred optimization algorithms because it is viable method for many different optimization problems. Although it works according to a standard and specific point scanning logic, the hyperparameter values it offers as output generally have a positive effect on learning performance.

Cross-validation is a crucial step in the AutoML pipeline, as it helps to prevent overfitting. Overfitting occurs when a model is too complex and tries to fit the training data too closely, resulting in poor performance on new, unseen data. By dividing the dataset into K subsets, each of which is utilized as the test data only once while the remaining data is used for model training, cross-validation helps to mitigate this. Each subset is used as test data once, and this is repeated K times. The best model is then selected based on the model's overall performance.

A new model is built from the training data in each iteration, and its performance is evaluated using the test data. By evaluating the performance on several subsets of the data, this approach helps identify the most suitable model while avoiding overfitting.

Cross-validation also reduces the possibility of selecting an inferior model as a result of a random data split. Cross-validation contributes to the building of a more robust and trustworthy model by utilizing all the data for both model training and evaluation. The only drawback is that the cost of cross-validation is high because K iterations are required, but the low error rate makes it a valuable step in the AutoML pipeline.

5 Results

The aim of this study is to develop a machine learning pipeline that can automate the entire process of intrusion detection in industrial IoT, making it accessible to individuals with little to no experience in data mining. By utilizing multiple techniques to identify the suitable parameter values, the focus is on identifying the classification algorithms for intrusion detection that are the most efficient. To achieve this, the models are trained and tested using the input dataset in order to determine the approach that has the highest success rate. In order to detect intrusions in industrial IoT systems, a straightforward and user-friendly solution must be offered, one that does not require substantial knowledge or technical proficiency in data mining. This will increase the overall effectiveness and security of industrial IoT systems.

The development of an intrusion detection system can be automated using an AutoML pipeline. The pipeline can complete all significant operations, including feature selection, hyperparameter tuning, and model training, without the requirement for manual observation. The pipeline can then provide a model that has been built specifically for the given dataset, leading to high accuracy in recognizing and classifying attacks.

In this study, there are three different imputation methods: mean imputation, median imputation, and most frequent imputation to handle data that is missing. To select the most effective strategy, the datasets were tested to each of these methods, and the results were compared.

The mean imputation method replaces the average of the non-missing values in a column for any missing data points. Although this method is straightforward to use, it can cause bias if the missing

values are not distributed randomly. The median of the non-missing values in the same column is used as the replacement for missing values when using the median imputation method. Compared to mean imputation, which is vulnerable to outliers, median imputation provides a more reliable evaluation of the data's central tendancy. However, if the missing values are not missing at random, this strategy may continue to induce bias. The most common imputation method involved substituting the most frequent value in the same column for any missing data. This method is useful when dealing with categorical data, but it can lead to loss of information if the missing values represent a less frequent category.

The framework compares results after using each imputation technique to determine the most effective method of approach. Framework evaluates each imputation method's performance using a range of criteria, including as accuracy, precision, recall, and F1-score. The approach used for the feature selection step is the one that outperformed all classification models in terms of performance.

By computing the Shapley values for each feature of the sample, the SHAP technique can be used to interpret the predictions of the machine learning models. On the contrary, the GA can be used for optimization to select the most important features from a large set of features.

The combination of SHAP and GA improves the optimization process. The features with the highest importance are chosen using GA after determining the SHAP values for each feature. Because GA is capable of handling the computation of large dimensional data and effectively discovering the most important characteristics, the feature selection process is improved in this approach.

The proposed methodology uses Shapley values to evaluate each characteristic's importance while setting a threshold for it. In the study, two distinct methods were used to determine the threshold. In the first approach, a fixed threshold value was selected. As a result, a general threshold can be obtained for any classification algorithm and dataset. The second approach required calculating the mean absolute Shapley value and using it as the threshold.

A Shapley-based measure for feature importance is the mean absolute Shapley value. Shapley values, which can be either positive or negative, represent each feature's contribution to the model's prediction. The absolute value of each feature's Shapley value is calculated, as well as the mean value over all instances in the dataset, to provide the mean absolute Shapley value. By considering the magnitude and direction of the Shapley values, it provides an overall measure of the importance of each characteristic. Any feature whose importance is below the threshold is not used in the genetic algorithm.

By identifying and removing the irrelevant, redundant or noisy features, feature selection can reduce the dimensionality of the dataset and make the model more robust and interpretable. This results in better generalization performance, reduced overfitting, and faster training times. The effectiveness of feature selection can be seen in Table 4, where it is shown that the elapsed time for training and testing is significantly reduced after feature selection, and the accuracy of the model is increased. In order to calculate the elapsed time for training, modelling stages were repeated 10 times and the average was taken. Table 5 shows the comparison of the model accuracy before and after hyperparameter tuning in binary classification. By carefully choosing the hyperparameters, it is possible to significantly improve the accuracy of the model, resulting in better results on the test data.

The results of the models were analyzed using the confusion matrices and classification reports to assess their accuracy and performance. The framework calculated these results for all the models that were used in the AutoML system. Figs. 6–9 show the confusion matrices of four of the methods used,

and Tables 6–9 include the corresponding classification reports. These results were obtained before any hyperparameter tuning was performed.

Table 4: Model accuracies and elapsed time with selected features

Model name	Accuracy (Before FS)	Accuracy (After FS)	Elapsed time for training (Before FS) (second)	Elapsed time for training (After FS) (second)	Number of removed features
AdaBoost	0.9691	0.9792	801.82	695.48	18
CatBoost	0.9963	0.9969	367.04	332.25	17
Decision tree	0.9962	0.9966	118.2	82.88	16
Gradient boosting	0.9966	0.9968	15751.3	11728.44	16
kNN	0.9901	0.9920	5811.32	5114.28	17
LightGBM	0.9960	0.9968	42.48	38.22	15
Naïve Bayes	0.5783	0.8625	5.59	4.97	17
Neural network	0.9878	0.9927	7197.74	5293.49	18
Perceptron	0.9296	0.9385	8.9	6.4	16
Random forest	0.9970	0.9975	1088.63	722.49	19
Ridge	0.9342	0.9519	5.04	4.01	17
Stochastic gradient descent	0.9403	0.9488	11.23	8.74	17
SVM	0.9828	0.9904	37612.4	32274.54	15
XGBoost	0.9970	0.9972	194.67	143.29	16

Table 5: Model accuracies w/o hyperparameters

Model name	Accuracy (Before tuning)	Accuracy (After tuning)
AdaBoost	0.9691	0.9849
CatBoost	0.9963	0.9981
Decision tree	0.9962	0.9967
Gradient boosting	0.9966	0.9971
kNN	0.9901	0.9930
LightGBM	0.9960	0.9972
Naïve Bayes	0.5783	0.8710
Neural network	0.9878	0.9954
Perceptron	0.9296	0.9753
Random forest	0.9970	0.9986
Ridge	0.9342	0.9588
Stochastic gradient descent	0.9403	0.9537
SVM	0.9828	0.9970
XGBoost	0.9970	0.9982

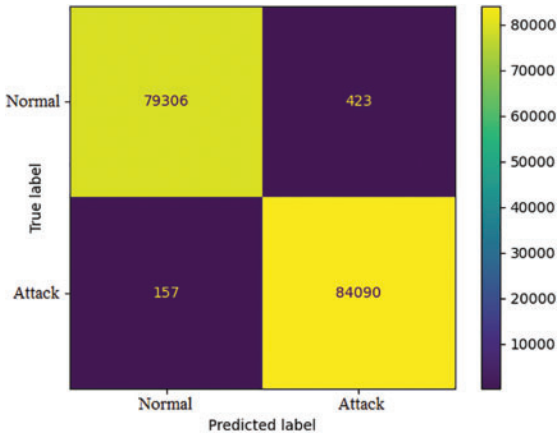

Figure 6: CatBoost confusion matrix

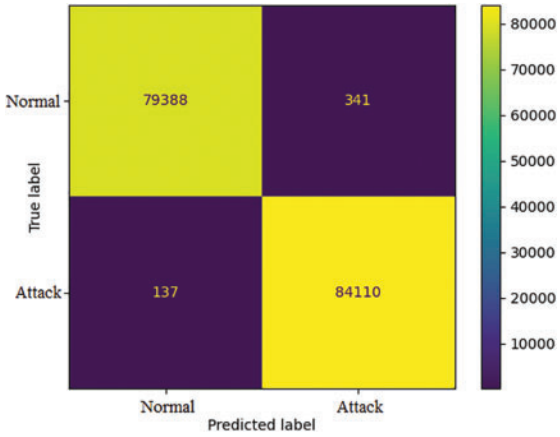

Figure 7: XGBoost confusion matrix

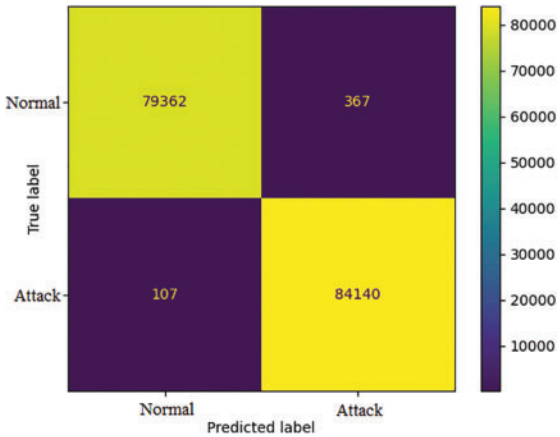

Figure 8: Random forest confusion matrix

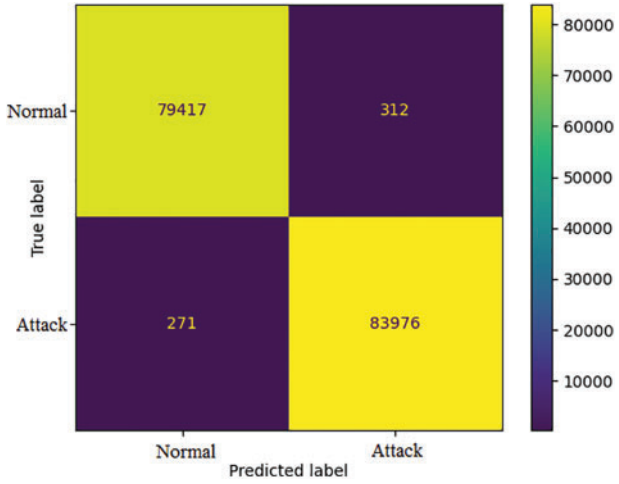

Figure 9: Decision tree confusion matrix

Table 6: Classification report of CatBoost

	Precision	Recall	F1-score	Support
Normal	0.9980	0.9947	0.9964	79729
Attack	0.9950	0.9981	0.9966	84247
Accuracy				0.9965

Table 7: Classification report of XGBoost

	Precision	Recall	F1-score	Support
Normal	0.9983	0.9957	0.9970	79729
Attack	0.9960	0.9984	0.9972	84247
Accuracy				0.9971

Table 8: Classification report of random forest

	Precision	Recall	F1-score	Support
Normal	0.9987	0.9954	0.9970	79729
Attack	0.9955	0.9987	0.9972	84247
Accuracy				0.9971

Table 9: Classification report of decision tree

	Precision	Recall	F1-score	Support
Normal	0.9966	0.9961	0.9963	79729
Attack	0.9963	0.9968	0.9965	84247
Accuracy				0.9964

Confusion matrices offer a visual illustration of the model's aptitude to correctly predict the target categories, while classification reports offer a more in-depth evaluation of the model's performance, encompassing metrics such as precision, recall, F1-score, and support. Confusion matrix is a commonly used evaluation metric in machine learning. It facilitates the evaluation of the effectiveness of a classification model. It shows the distribution of the actual outcomes *vs.* the predicted outcomes. The confusion matrix consists of 4 elements: True Positives (TP), False Positives (FP), False Negatives (FN), and True Negatives (TN). TP stands for the number of instances correctly classified as positive, FP represents the number of instances wrongly classified as positive, FN symbolizes the number of instances wrongly classified as negative, and TN represents the number of instances correctly classified as negative. The accuracy of the model can be calculated by dividing the total number of correct predictions with the total number of instances. recision is calculated by dividing TP with (TP + FN), which shows the proportion of positive predictions that are actually correct. Recall is calculated by dividing TP with (TP + FN), which shows the proportion of positive instances that are correctly predicted. F1-score is the harmonic mean of precision and recall, which provides a single score that balances both precision and recall. A high F1-score means that the model has a good balance of precision and recall. The results from the confusion matrices and classification reports are crucial in determining the effectiveness of the AutoML system and guiding future improvements. Eqs. (1)–(3) present the metrics for recall (detection rate), accuracy, and precision.

$$Recall = \frac{TP}{TP + FN} \tag{1}$$

$$Accuracy = \frac{TP + FN}{TN + TP + FN + FP} \tag{2}$$

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

Grid search was preferred as parameter search method for optimization. It is a simple and straightforward method for hyperparameter optimization. It is widely used and widely recognized as an effective method. The main advantage of grid search is that it is easy to implement, and it can be automated. Grid search generates a list of all possible combinations of hyperparameters and runs the algorithm for each combination. It then selects the combination that results in the best performance. This allows for an exhaustive search of all possible hyperparameters, which makes it an effective method for hyperparameter optimization. However, the disadvantage of grid search is that it is computationally expensive, especially for large datasets or models with many hyperparameters. Despite this, grid search is still widely used due to its ease of use and its ability to produce reliable results.

The performance of the proposed system was compared with the studies in which the X_IIoTID dataset was created and used. In these studies, classification was made with different methods, and the highest success rate in [25] study was 98.23, while the highest success rate in [20] study was 99.54 with

decision tree. As seen in Table 9, the accuracy of decision tree method without hyperparameter tuning is higher than the existing studies. The main reason for achieving a higher success rate with the same method is the automation of preprocessing steps and feature selection. When Table 5 is examined, it was seen that the highest success rate (0.9986) in the X_IIoTID dataset, was obtained with random forest method by performing hyperparameter tuning.

All these processes have performed for binary classification (attack/normal), and the same processes can also be run for multi-class classification by proposed model. The results of the multi-class classification show the accuracy rates of AutoML model according to the attack types.

Table 10 shows the detection rates of 3 of the classification methods optimized by proposed AutoML model in 4 of the 18 different attack types in the X-IIoTID dataset. If the results of the decision tree method are examined, it has much more accurate in detecting Bruteforce attacks than Command&Control attacks.

Table 10: Detection rate for 4 attacks

Method	Bruteforce	C&C	Ransomware	Dictionary
Decision tree	0.9999	0.9961	0.9999	0.9984
Random forest	0.9999	0.9955	0.9999	0.9997
XgBoost	0.9999	0.9974	0.9999	0.9999

Apart from model performance metrics, the time required for training and prediction is also important. When it comes to an IDS ve IIoT, the latency here should be very low. Thanks to the developed framework, time spent in each step of AutoML processes can be calculated. The graph of the time spent in the model training process examined in Table 4 is shown in Fig. 6. In addition, Fig. 10 shows howlong it takes to predict a request to each model.

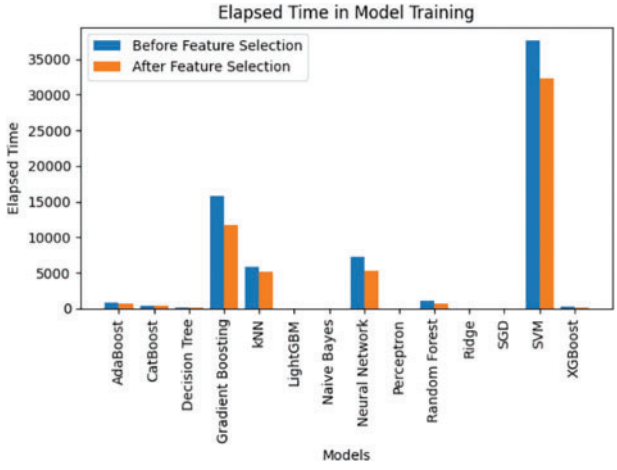

Figure 10: Elapsed time in model training

As seen in Fig. 11, although the prediction success of the request is high with XGBoost, the prediction process over the model file takes a very long time.

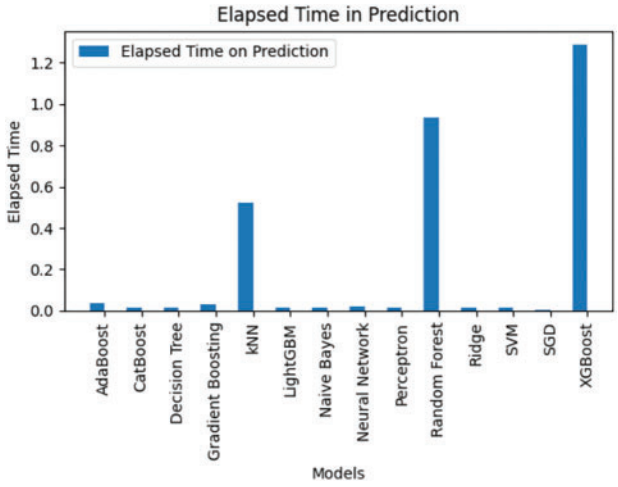

Figure 11: Elapsed time in prediction

There are also old but popular datasets used in intrusion detection system development studies with machine learning methods. Examples of these datasets are KDD99, NSL-KDD, UNSW-NB15 datasets. Since the model proposed in this study is an AutoML, it is expected to give successful results in training and prediction processes in different datasets. When the proposed AutoML model is applied to the above-mentioned datasets, it has been observed that the results are successful. The result of these comparisons are shown in Table 11.

Table 11: Comparison between proposed model with previous approaches on KDD99, NSL-KDD, UNSW-NB15 datasets

Study	Approach	Dataset	Accuracy
[26]	Naïve Bayes	KDD99	0.950
[27]	K-Means and ANN	KDD99	0.975
[28]	Convolutional neural network	KDD99	0.9984
[29]	Meanshift clustering algorithm	KDD99	0.8120
[30]	Multi-agent system	KDD99	0.9582
AID4I	**Auto-Selected Model: CatBoost**	**KDD99**	**0.9998**
[31]	C4.5, Naïve Bayes, Random Forest	NSL-KDD	0.9965
[32]	k-NN, K-Means	NSL-KDD	0.9943
[33]	Random forest	NSL-KDD	0.9870
AID4I	**Auto-Selected Model: XGBoost**	**NSL-KDD**	**0.9993**
[34]	Decision Tree, Naïve Bayes, ANN, Logistic Regression, EM Clustering	UNSW-NB15	0.8556
[35]	MSCNN-LSTM	UNSW-NB15	0.9560
[8]	Fuzzy C-means	UNSW-NB15	0.9890
AID4I	**Auto-selected model: Decision tree**	**UNSW-NB15**	**0.9995**

6 Conclusion

In a World where the demand in the field of machine learning is increasing, it's an important problem that the desired success in this field depends on the number of qualified people in the field. AutoML concept has been developed to find a solution to this problem. Because no matter what the real-world problem is, it's an important advantage to approach the machine learning process of the problem that contains the data with an iterative or linear pipeline setup by approaching it's an optimization problem.

Feature selection, machine learning method selection and parameter tuning become more important as the data size increases. In this study, model-dependent feature selection operations, scale operations and optimization of hyperparameters were automated. Along with summarizing the relevant algorithms, the methods used for the automation process, including their pros, cons are mentioned.

The goal of this study is to automate essential cyber security-related processes, including feature selection, hyperparameter optimization, and machine learning technique selection. The methods employed in the automation process are thoroughly described, along with their benefits and drawbacks. The viability of the algorithms utilized for more complex models and their usage of computational resources are also compared and evaluated in this work. The aim of this research is to provide a thorough overview of AutoML as a tool for researchers and users.

By automating the process of building intrusion detection models, AutoML may help in reducing the workload for security professionals and IT employees. With the use of AutoML, the focus can be shifted from manually fine-tuning models to evaluating the results and enhancing the IIoT system's overall security posture. Additionally, AutoML can increase the reliability and accuracy of IDS systems in recognizing cyberattacks on IIoT systems. AutoML can assist in identifying previously unidentified threats and lowering the amount of false positive alarms by analyzing vast amounts of data. The way we approach cyber security in these systems could be completely changed by the introduction of AutoML in intrusion detection systems for IIoT. By automating the model building process, the workload for security experts can be reduced and the overall security posture of IIoT systems can be improved.

7 Future Research

Grid search is a commonly used hyperparameter optimization algorithm that is easy to implement. However, it is subject to a dimensionality problem where the number of hyperparameters and their corresponding values increase, leading to a huge search space that can slow down the optimization process.

As an alternative to the grid search method, random search, Bayesian search, or Tree-structured parzen estimator can be used. However, each of these methods has its own limitations. Random search method may not guarantee convergence to the optimal value and its efficiency can be low. Bayesian Search method is conceptually more complex and challenging to implement in a parallel computing environment. Additionally, the method requires a good understanding of Bayesian optimization, which can be challenging for practitioners. The Tree-structured parzen estimator is also a complex method and requires a lot of computational resources.

Therefore, choosing the best hyperparameter optimization algorithm requires a trade-off between the simplicity of implementation, efficiency, and accuracy.

Acknowledgement: This study is a part of the Ph.D. Thesis at Atatürk Strategic Studies and Graduate Institute, National Defence University, Istanbul, Turkey. We would like to thank Siemens Turkey for their support in the completion of the study.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] A. Bagri, R. Netto and D. Jhaveri, "Supervisory control and data acquisition," *International Journal of Computer Applications*, vol. 102, no. 10, pp. 1–5, 2014.

[2] A. Hassanzadeh, S. Modi and S. Mulchandani, "Towards effective security control assignment in the industrial internet of things," in *IEEE World Forum on Internet of Things (WF-IoT)*. Milan, Italy: IEEE, pp. 795–800, 2015.

[3] A. Meddeb, "Internet of things standards: Who stands out from the crowd?" *IEEE Communication Magazine*, vol. 54, no. 7, pp. 40–47, 2016.

[4] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *Communication Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[5] R. Samrin and D. Vasumathi, "Review on anomaly based network intrusion detection system," in *Int. Conf. on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)*, Mysuru, India, pp. 141–147, 2017.

[6] S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han *et al.,* "Enhanced network anomaly detection based on deep neural networks," *IEEE Access*, vol. 6, pp. 48321–48246, 2018.

[7] J. Yin, Y. Shi, W. Deng, C. Yin and T. Wang, "Internet of things intrusion detection system based on convolutional neural network," *Computers, Materials & Continua*, vol. 75, no. 1, pp. 2119–2135, 2023.

[8] V. Hajisalem and S. Babaie, "A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection," *Computer Networks*, vol. 136, pp. 37–50, 2018.

[9] X. Zhang, "Machine learning," in *Matrix Algebra Approach to Artificial Intelligence*, 1st ed., Singapore: Springer, 2020.

[10] M. Mohammed, M. B. Khan and E. B. M. Bashier, "Supervised learning," in *Machine Learning: Algorithms and Applications*, 1st ed., Boca Raton, FL, USA: CRC Press, 2016.

[11] Y. Bengio and J. Bergstra, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

[12] A. Sezgin and A. Boyacı, "Enhancing intrusion detection in industrial internet of things through automated preprocessing," *Advances in Science and Technology Research Journal*, vol. 17, no. 2, pp. 120–135, 2023.

[13] S. Sanders and C. Giraud-Carrier, "Informing the use of hyperparameter optimization through metalearning," in *Int. Conf. on Data Mining (ICDM)*, New Orleans, LA, USA, pp. 1051–1056, 2017.

[14] R. S. Olson, W. La Cava, Z. Mustahsan, A. Varik and J. H. Moore, "Data-driven advice for applying machine learning to bioinformatics problems," in *Pacific Symp. on Biocomputing*, vol. 23, Hawaii, USA, pp. 192–203, 2018.

[15] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter and K. Leyton-Brown, "Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA," *Journal of Machine Learning Research*, vol. 18, no. 25, pp. 1–5, 2017.

[16] R. S. Olson and J. H. Moore, "TPOT: A tree-based pipeline optimization tool for automating machine learning," in *Automated Machine Learning*. NYC, USA: Springer, Chapter 8, pp. 151–160, 2019.

[17] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum *et al.,* "Efficient and robust automated machine learning," in *Int. Conf. on Neural Information Processing Systems*, vol. 2, Montreal, Canada, pp. 2755–2763, 2015.

[18] H. Jin, Q. Song and X. Hu, "Auto-keras: An efficient neural architecture search system," in *Int. Conf. on Knowledge Discovery & Data Mining*, *ACM SIGKDD*, Anchorage, AK, USA, pp. 1946–1956, 2019.
[19] A. Yakovlev, H. F. Moghadam, A. Moharrer, J. Cai, N. Chavoshi *et al.* "Oracle AutoML: A fast and predictive AutoML pipeline," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 3166–3180, 2020.
[20] M. Al-Hawawreh, E. Sitnikova and N. Aboutorab, "X-IIoTID: A connectivity-agnostic and device-agnostic intrusion data set for industrial internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3962–3977, 2022.
[21] N. Talpur, S. J. Abdulkadir, M. H. Hasan, H. Alhussian and A. Alwadain, "A novel wrapper-based optimization algorithm for the feature selection and classification," *Computers, Materials & Continua*, vol. 74, no. 3, pp. 5799–5820, 2022.
[22] Y. Saeys, I. Inza and P. Larranaga, "A review of feature selection techniques in bioinformatics," *Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.
[23] L. S. Shapley, "A value for n-person games," in *Contributions to the Theory of Games (AM-28)*, vol. 2, Princeton, Princeton University Press, pp. 307–318, 2016.
[24] S. M. Lundberg and S. Lee, "A unified approach to interpreting model predictions," in *Int. Conf. on Neural Information Processing Systems*, Long Beach, California, USA, pp. 4768–4777, 2017.
[25] M. Al-Hawawreh, E. Sitnikova and N. Aboutorab, "Asynchronous peer-to-peer federated capability-based targeted ransomware detection model for industrial IoT," *IEEE Access*, vol. 9, pp. 148738–148755, 2021.
[26] M. Panda and M. R. Patra, "Network intrusion detection using Naive Bayes," *International Journal of Computer Science and Network Security*, vol. 7, no. 12, pp. 258–263, 2007.
[27] B. Ren, M. Hu, H. Yan and P. Yu, "Classification and prediction of network abnormal data based on machine learning," in *Int. Conf. on Robots & Intelligent System (ICRIS)*, Haikou, China, pp. 273–276, 2019.
[28] N. Ding, Y. Liu, Y. Fan and D. Jie, "Network attack detection methods based on convolutional neural network," in *Chinese Intelligent Systems Conf.*, Haikou, China, pp. 610–620, 2019.
[29] A. Kumar, W. Glisson and R. Benton, "Network attack detection using an unsupervised machine learning algorithm," in *Hawaii Int. Conf. on System Sciences*, Hawaii, USA, pp. 6496–6505, 2020.
[30] M. A. Riyad, I. Ahmed and R. Khan, "An adaptive distributed intrusion detection system architecture using multi agents," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 6, pp. 4951–4960, 2019.
[31] C. Khammassi and S. Krichen, "A NSGA2-LR wrapper approach for feature selection in network intrusion detection," *Computer Networks*, vol. 172, pp. 1–18, 2020.
[32] H. Wang, B. Han, J. Su and X. Wang, "A high-performance intrusion detection method based on combining supervised and unsupervised learning," in *IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. Guangzhou, China: IEEE, pp. 1803–1810, 2018.
[33] Z. E. Mrabet, H. E. Ghazi and N. Kaabouch, "A performance comparison of data mining algorithms based intrusion detection system for smart grid," in *Int. Conf. on Electro Information Technology (EIT)*, Brookings, SD, USA, pp. 298–303, 2019.
[34] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Information Security Journal: A Global Perspective*, vol. 25, no. 1–3, pp. 18–31, 2016.
[35] J. Zhang, Y. Ling, X. Fu, X. Yang, G. Xiong *et al.,* "Model of the intrusion detection system based on the integration of spatial-temporal features," *Computers & Security*, vol. 89, 101681, 2020. https://www.sciencedirect.com/science/article/abs/pii/S0167404819302214