National College of Ireland

Intrusion Detection in IoT Systems using Machine Learning

MSc Research Project

MSc Cyber Security

Samuel Avwerosughene Arhore

Student ID: x21134502@student.ncirl.ie

School of Computing

National College of Ireland

Supervisor: Dr. Vanessa Ayala-Rivera

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	…….Samuel Avwerosughene Arhore………………………………………………		
Student ID:	…………x21134502……………………………………………………………………….……		
Programme:	………MSc Cyber Security……………………….…	**Year:**	..2022/2023…….
Module:	……MSc Research Project/Internship…………………………………….………		
Supervisor:	……Dr Vanessa Ayala-Rivera…………………………………………….………		
Submission Due Date:	…………15th December 2022…………………………………………………….………		
Project Title:	…………Intrusion Detection in IoT using Machine Learning…….…..………		
Word Count:	……………8092……………… **Page Count**……………………20……………….…….		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ………Samuel Avwerosughene Arhore……………………………………………

Date: ………………………………14 December 2022…………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Intrusion Detection in IoT Systems using Machine Learning

Samuel Avwerosughene Arhore

x21134502

Abstract

Data and device security has become increasingly important as the Internet of Things (IoT) develops and is used globally. In IoT, there are an enormous number of devices and they are all configured differently, making it difficult to develop trusted interconnections between many of these nodes. Intrusion Detection Systems (IDS) that work collaboratively often deliver better intrusion detection performance, since nodes in an IDS can exchange information between each other to ensure effective intrusion detection. Due to their power consumption, loT devices are normally unable to perform a number of computations. Therefore, encryption and authentication do not provide effective protection from malicious cyber-attacks. Since intrusions pose a threat to security, IDSs have become the forefront of security solutions, detecting abnormal and malicious activities in IoT networks using machine learning algorithms has shown promising results recently. An analysis of several machine learning techniques is presented in the paper in order to determine which one performs best with a chosen set of datasets in detecting intrusion. Each of them is also discussed in terms of its limitations. Using a set of four performance metrics; recall, precision, classification accuracy and F1 score, a confusion matrix is created and the system's performance will be evaluated following an experiment applied with an up-to-date and relevant dataset. It is expected that by the end of this study, a suitable algorithm will be proposed which detects network intrusions in a minimal amount of time with accurate results. The winner technique achieved 99% accuracy to a high degree of efficiency and accuracy in a short amount of time.

1. Introduction

Network security is a passive requirement that cannot be ignored. In an attack or intrusion, sensitive information is revealed, disabled, stolen, or unauthorised access is attempted. It is possible for hackers to damage a single or networked computer by making unauthorised access to it. Network intrusions are made more difficult by the modern detection industry for security breaches [1]. In 2021 and 2022, Identity Theft Resource Centre estimated that data breaches increased by 68% [2]. With this recent surge in security and data breaches, individuals and organisations have had to implement more robust security measures to ensure safety of their data in the rapidly growing information society. Data should be protected against attacks and threats by paying more attention to network security. Breaches that involve mobile and IoT (internet of things) devices have increased in frequency because of a lack of data protection, side effects of global pandemics, and increased exploit sophistication. Furthermore, COVID-19 has increased remote worker numbers, making cyberattacks more likely [3]. In order to prevent malicious behaviours on a system and network, intrusion detection techniques are employed. On the basis of the collection of network status data that are high dimensional and linear inseparable, network intrusion detection determines if the current network behaviour is intrusion or normal [4]. Recently, machine learning algorithms have shown promise in detecting abnormal activity in IoT networks. In parallel with the rapid growth of IoT applications, security challenges have been increasing.

Although many security protocols have been developed to deal with these challenges, intrusion attacks continue to rise. Computer networks are increasingly being penetrated by intrusion attacks [5]. The performance of intrusion detection systems is not satisfactory despite the availability of many detection methods and software solutions today. Real-world IDS can cause some inconvenient events because of certain events that can occur. Therefore, many researchers are still trying to find ways to make intrusion detection systems that are as accurate and as timely as possible. A system for detecting intrusions and reporting them to a network administrator or to authorities is an Intrusion Detection System (IDS) [1].

Security methods for organisations have been developed over the years through experimentation and implementation. There are several methods that have been in use for many years, such as Virtual Private Networks (VPNs), Firewalls, and encryption. Although security methods have been discovered, the search for new methods has not ended. Security can be enhanced by adding IDSs, a recently discovered system. The purpose of intrusion detection is to boost the security of IoT through a variety of security measures. By using various intrusion detection techniques, security can be ensured by detecting intrusions in a system. Preventing attacks from happening in the first place is the best way to defend against them [6].

In order to solve the above research problem, the following research question needs to be answered: **What role can machine learning play in improving the efficiency of intrusion detection in IoT as security concerns are rapidly growing ?**

This study aims to show the different benefits and costs of using machine learning techniques in IoT, as well as determine whether it is effective for intrusion detection; this study's findings are expected to provide researchers and industry professionals with additional insight into how to handle intrusion tolerance in the Internet of Things. The first step in this process is to review the security issues encountered in IoT today. IDSs were analysed to understand how they can be used to solve security problems in the Internet of Things. IDSs were then implemented in IoT scenarios after the reviews have been completed. We evaluated the performance of these techniques in order to better understand their importance in preventing intrusions in IoT. From this analysis, performance metrics were used to determine the merits and drawbacks of using IDSs and suggest improvements whenever necessary. In order to determine which machine learning algorithm worked best for the system, various algorithms were compared. The algorithms experimented with include Random Forest Classifier, XGBoost Classifier and Support Vector Machine (SVM). In the implementation, we chose the random forest algorithm due to its enhanced accuracy compared to other nonlinear classifiers. As this algorithm takes the average of the entire output, it eliminates the problem of overfitting, which arises when bias and variance are not balanced. Lasso and Ridge Classifiers are used by XGBoost in order to penalise highly complex models. As a result of the algorithm's efficient use of hardware resources, even when disk space is limited, the dataset's size is reduced. Additionally, overfitting is prevented by the cross-validation feature. With the help of these models, an IDS can be efficiently and effectively implemented.

2. Literature Review

Researchers are always looking for ways to improve the current intrusion detection techniques or to come up with new ideas due to the growing concerns about privacy and security in IoT. Technology centred on the Internet of Things will greatly contribute to the advancement of informatization in related fields as a result of its emergence and application. Furthermore, it has an important influence on smart city development and management, industrial production upgrades and transformations, and people's daily lives. A number of bottlenecks that restrict social and economic development can also be solved through new ways and technical support.

2.1. Intrusion Detection System

It is imperative to develop security measures for IoT devices in light of the threats that IoT faces. For IoT devices to be safe, data sources must be protected from unauthorised access by malicious users. IDS provides very significant security for IoT networks as it detects intruders and prevents unauthorised access. IDS integration is difficult due to IoT devices' low energy requirements. This difficulty can be overcome with a centralised firewall that monitors the network and identifies intruders. In the event an anomaly is detected, the network administrator is alerted.

Anomaly Based Detection: By utilising machine learning, anomaly-based detection systems are trained to recognize a normalised baseline instead of searching for known threats. Based on this baseline, all network activity is compared to how the system normally behaves. IDSs that use anomaly-based detection simply identify any behaviour that is out-of-the-ordinary as a trigger for alerts, rather than searching for known IOCs [7]. There has been an increase in the use of IoT devices in various aspects of our lives in recent years. In our offices, houses, schools, hospitals, and many other places, these devices have become an integral part of our daily lives in many ways, so we must put in place security measures to protect data and prevent breaches. Twitter, Netflix and PayPal were among the numerous loT networks and websites that were targeted by a series of Distributed Denial-of-Service attacks in 2016 [8]. In [9], multiple machine learning approaches such as XGBoost, KNN, Random Forest Classifier, Logic Regression were compared and applied on an IoT Network Intrusion dataset. The KNN had the highest accuracy with 99% output and a runtime of 2 minutes followed by XGBoost with an accuracy of 97% and a runtime of 10.8 seconds. Through the use of unsupervised deep learning methods, [10] developed an industrial-level deep learning model to detect anomalies in the internet of things, while simultaneously providing continuous training by using deep auto-encoding and deep forward neural network architectures. As well as identifying existing intrusions, this method can identify new types of intrusions as well. Compared to other methods, this method has a low false alarm rate and a high detection rate. In spite of these shortcomings, the model may be unable to detect network intrusions with specific behavioural attributes because of its shortcomings in parameter selection and future value processing. Morfmo et al. [11] developed near-real-time intrusion detection for IoT devices using supervised machine learning in conjunction with Apache Spark. The explicit random forest model is directly implemented on loT devices. A training set with nearly 2 million instances was used by the authors to evaluate the performance on public datasets. According to the results, it achieves greater than 99% accuracy. An IDS called Passban was introduced by Vargas et. al [12]. In a nutshell, Passban is an intelligent IDS that protects any device connected directly to it. There are two main phases of operation for Passban IDS: learning and training. After learning the normal behaviour of the system, a model of the system is created by using machine learning algorithms. Detection of abnormal behaviour is based on the model created. Kejriwal et al. [13] proposed an anomaly based intrusion detection system method using several algorithms such as XGBoost, KNN, Random Forest Classifier, Logic Regression. After the implementation and comparison, it was seen that the Random forest classifier worked best with an accuracy of 99.8% and macro average F1-Score of 0.98.

Signature Based Detection: Signature-based detection is where the IDS monitors the packets being sent and received in the network and compares them to a pre-programmed list of known threats and indicators of compromise, for example, file hashes, malicious IP address or domains, email content or known behaviours that occur before a malicious network attack takes place to flag suspicious behaviours in a network. Sherali et al. [14] employed supervised learning, unsupervised learning, and reinforced learning techniques in their study. As a result of their implementation, the IoT environment was able to provide both host-based and network-based security solutions. Researchers report that network-based machine learning enables almost any approach to be implemented without limitations. A network's ability to encrypt communications is the only limiting factor in detecting security issues. Incorporating machine learning algorithms into IoT devices allows us to detect attacks that are difficult to detect with host-based detection approaches alone. Additionally, they discussed some of the challenges associated with machine learning deployment.
In [15], Lee et al. proposal reflects the features in real-world medical scenarios through a multiclass classification-based intrusion detection scheme. Using this scheme, convolutional neural networks are compared with data collected by real devices. Evaluations were conducted with the use of the actual healthcare IoT environment and real medical data from real devices. Cervantes et al. in [16] proposed an IoT sinkhole attack detection system in this paper. INTI, which is the name of the system, aims to improve IDS performance in IoT with respect to false positives, false negatives, and resource consumption by enhancing the accuracy of IDS to identify threats. In addition to using watchdogs and reputation strategies, trust strategies are also employed in order to detect infractions. In the event of an attack, the

nodes monitor the traffic and broadcast an alert. The network nodes then work together to isolate the adversary node. Based on the results of INTI simulations in static and mobile environments, 92% and 70% of attacks were detected respectively, while 28% of false positives were detected in mobile environments. A similar study was carried out by [17]. Husain et al. developed a feature selection model to evaluate an intrusion detection dataset. To identify which algorithm provided the best feature selection accuracy from the subsets of the data, Naive Bayes, Multilogistic Regression, Neural Network, non-linear Support Vector Machines, Random Forest, and XGBoost were applied. After comparing the results, it was seen that XGBoost and Random Forest had the highest accuracy in detecting different attacks on the network. To minimise loss when adding new models, XGBoost is selected for model development instead of random forest because it optimises the weight function at each iteration.

Hybrid Based Detection: In order to improve the accuracy and efficiency of existing intrusion detection systems, hybrid IDS have been developed to address challenges such as the high number of false positives and low detection rates of novel attacks. By combining signature-based and anomaly-based techniques, hybrid IDS is created [18]. In [19], a hybrid classification algorithm that works as an IDS was proposed. K Nearest Neighbour and decision tree algorithms are hybridised in this algorithm. A decision tree is used to first pass data, and then the output is combined with the original data to create the K Nearest Neighbour model. Finally, it is considered that the output of the KNN model is the final prediction. KNN was found to perform better with the information provided by decision tree models when tested on the NSL-KDD 1999 dataset. Combining KNNs and decision trees to create DT-KNN allows better results than either algorithm alone. A deep learning model was constructed by Ullah et al. [20] for detecting IoT anomalies. They built a multiclass classification model based on CNN to classify 15 different attacks. 3D, 2D, and 1D models were implemented using CNN. For binary and multiclass classification, four blocks of convolutional layers are used, and the method uses the transfer learning principle and CNN multiclass pretrained models. A recursive feature elimination method is used by the authors for feature selection. Their validation was based on four datasets, from which two novel datasets were generated containing 9 and 15 combination attacks. F1-score, recall, precision, and accuracy of the proposed model were better than those of existing deep learning models. Based on the results of the experiment, the proposed model performs 99.96% accurately, 99.9% precisely, 99.95% recall and 99.93% F1-score. According to the same model, the multiclass classification provided results of 99.97% accuracy, 99.95% precision, 99.95% recall, and 99.95% F1- score. However, the proposed model needs to be able to detect zero-day attacks in real time and respond in real time to them.

In this study, we propose an efficient algorithm to detect intrusions in the IoT, adding value to previous research in information security.

3. Research Methodology

In this section, we provide detailed information concerning the research methodology, the approach taken to pre-process the data, the analysis of the data, the architecture, implementation, and limitations of the proposed model. An overview of the steps involved in developing the IoT network attack and threat detection model can be found below. In order to test the IDS system effectively, it is very important to select the right dataset. Once both normal and malicious traffic has been collected during data preparation and preprocessing, packet dumps are passed on to later stages of data processing. To avoid displaying incomplete, inconsistent, or irrelevant data, several steps were taken to prepare the data. Feature selection is the next phase of the training and testing process, where optimal features are identified. Following the training of the dataset, three classifiers were used in the evaluation, followed by the selection of the most accurate classifier based on their performance. Following each stage, the result determines what needs to be done next. Based on the CRISP-DM methodology, this research was conducted. [23]. Using this approach, a data machine learning project can be planned in a systematic and well-tested manner. There is a discussion of the project phases, the roles and responsibilities associated with them, and their connections between the stages. The user's goals, interests, context, and, most importantly, the data are all factors that can make it difficult to identify all connections. In addition, the data may reveal relationships or connections between tasks [24].

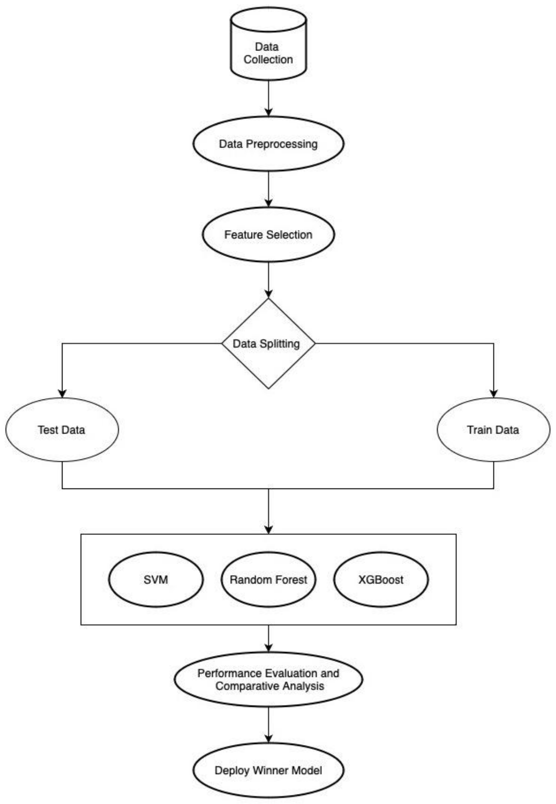
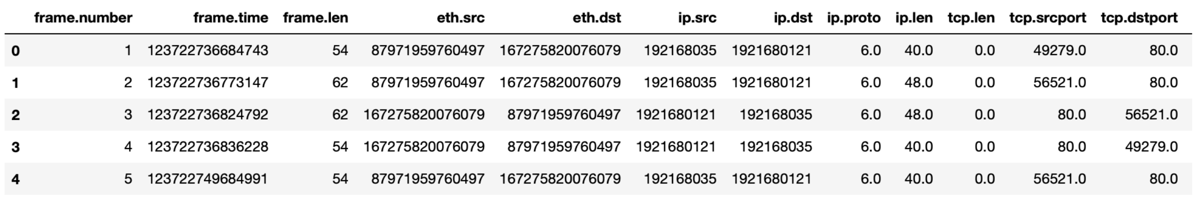


Fig 1: Proposed Methodology for Intrusion Detection System

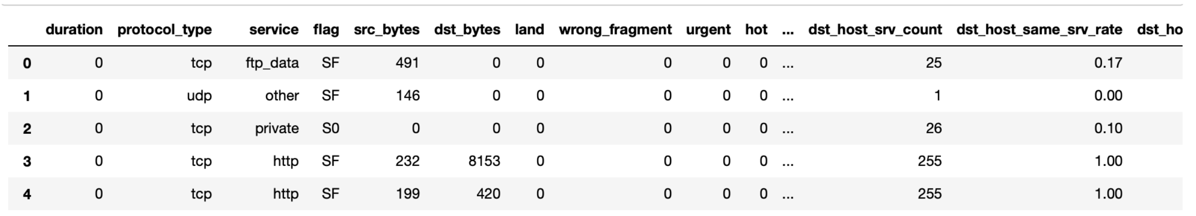
3.1. Dataset

There are many datasets on IDS that are made available to the public. However, a lot of these have flawed and inconsistent performance evaluations, as well as incomplete and outdated information. Metadata and feature sets are lacking, recent attacks are not displayed, and traffic volumes and diversity are inadequate for some of these products. Data from the Network Intrusion dataset [25] includes information on recent and updated network attacks, as well as benign data that can be used to simulate real-world events. Based on ICMP, UDP, and TCP protocols, this dataset was constructed to analyse abstract behavioural patterns. Another dataset used in this study was the IoT Device Network Logs [26]. The dataset consists of 6 main categories. The 6 categories include a category of normal transmissions and 5 cyberattacks. These 5 attack groups cover some of the most common attacks for loT networks. The Wrong Setup accounted for 17.24% of the record, Distributed Denial of Service (DDoS) attacks accounted for 16.55%, Datatype Probing accounted for 16.55%, Scanning attacks accounted for 16.56%, and Man in the Middle (MITM) accounted for 16.55%. The remaining 16.55% were normal packet



	frame.number	frame.time	frame.len	eth.src	eth.dst	ip.src	ip.dst	ip.proto	ip.len	tcp.len	tcp.srcport	tcp.dstport
0	1	123722736684743	54	87971959760497	167275820076079	192168035	1921680121	6.0	40.0	0.0	49279.0	80.0
1	2	123722736773147	62	87971959760497	167275820076079	192168035	1921680121	6.0	48.0	0.0	56521.0	80.0
2	3	123722736824792	62	167275820076079	87971959760497	1921680121	192168035	6.0	48.0	0.0	80.0	56521.0
3	4	123722736836228	54	167275820076079	87971959760497	1921680121	192168035	6.0	40.0	0.0	80.0	49279.0
4	5	123722749684991	54	87971959760497	167275820076079	192168035	1921680121	6.0	40.0	0.0	56521.0	80.0

Fig. 2: Table showing the first 5 rows of the IoT dataset



	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host_same_srv_rate	dst_ho
0	0	tcp	ftp_data	SF	491	0	0	0	0	0	...	25	0.17	
1	0	udp	other	SF	146	0	0	0	0	0	...	1	0.00	
2	0	tcp	private	S0	0	0	0	0	0	0	...	26	0.10	
3	0	tcp	http	SF	232	8153	0	0	0	0	...	255	1.00	
4	0	tcp	http	SF	199	420	0	0	0	0	...	255	1.00	

Fig. 4: Table showing the first 5 rows of the NID dataset

3.2. Data Processing

It is important to pre-process data in machine learning as the quality of the data is an important feature that directly affects the learning capability of the model. The preprocessing of your data is what cleans and prepares it for training. For the purpose of preventing errors that could lead to flawed results, it is imperative to pre-process data before sending it to the model. As part of this process, the datasets were split into train and test so that the model is evaluated before it is deployed. A check was made to see whether the datasets contained null values. In this way, the model would be able to process all the data available to give a more accurate result. The results of the model could be negatively affected if a field was null/missing. Label Encoding was used to handle categorical features. Data typically undergoes encoding or converting before being parsed by a machine in a machine learning process. Some features such as flag, service, protocol type in the first dataset and normality in the second dataset were encoded into numerical labels.

3.3. Feature Selection

Machine learning algorithms are trained by selecting the most relevant features and then inputting them into them and this process is known as Feature Selection. In this process, redundant and irrelevant features are eliminated and the set of features is narrowed down to the ones that are most relevant to the machine learning model [27]. In this study, the correlation coefficient was used to determine the correlation value for all features in comparison with the target variable. There were 43 features in the Network Intrusion Detection dataset to be used to predict the result. A number of features had high negative values, which could affect the prediction of the model. In order to reduce their impact on the algorithm and produce a more accurate result, 37 features with negative values and weights over 0.5 were removed. A total of 6 features were used.The method of feature selection was also applied to the IoT Device Network Logs dataset. There were 14 features to be used in the model to predict the results. A few of the features displayed a high negative value, which may have affected the model's prediction. Feature values that had high negative values were removed from the final prediction to get a more accurate result. This translates to removing features that are not essential as well as leaving those that contribute to the algorithm's performance.

3.4. Modelling

The algorithms used are Random Forest, Support Vector Machine (SVM) and XGBoost. Machine Learning was utilised to execute each classifier to categorise incoming traffic as either normal or anomalous. A 70:30 ratio was used to separate the dataset into a test and a train set. To prevent models from being affected by highly correlated columns, an exploratory analysis of the data is performed. For the first dataset, classifiers were developed for binary classifications. This will allow us to determine whether the model is successful in separating malicious traffic from normal incoming traffic. The class column tells whether the network traffic is "normal" or "anomalous" and this was encoded into a numerical representation using Label Encoder. According to the normality column, the second dataset contains information about the attack type in a multiclass analysis. In every scenario, the test dataset will be assessed using the model with the best performance.

4. Design Specification

A detailed description of our proposed model's design specification is provided in this section. In this study, the Network Intrusion Dataset was used [25]. The dataset consists of a wide variety of instructions simulated in a military network environment. The objective of this project was to provide a simulation of a typical US Air Force LAN in order to collect raw TCP/IP dump data for a network. There were multiple attacks on the LAN as if it were a real environment. Each connection consists of TCP packets sent between two IP addresses under some well-defined protocol that flows data between them starting and stopping at a certain time interval. The connections are also categorised as normal or as attacks with exactly one type of attack for each [25].

The detection of network attacks can be performed using supervised learning algorithms like Random Forest Classifiers, Support Vector Machines (SVM), and XGBoost in IDS. The results from this study are based on these three different classifiers.

4.1. Random Forest Classifier

The Random Forest (RF) algorithm is one that can be used to solve classification or regression problems. It uses supervised learning to build multiple Decision Trees randomly that are combined to enhance performance and accuracy. In Random Forest, rules for class polling are defined using Decision Trees in order to avoid overfitting. In real-time applications and with large training data, this algorithm is unsuitable due to its time-consuming construction of trees [28]. A decision tree voting method is used to predict and rate important characteristics. This training creates 100 decision trees based on 100 estimators. The categorization result of each decision tree is reflected in the fact that they all take into account a new instance. To arrive at the final prediction, these votes are averaged [25]. Due to its high accuracy and the fact that it prevents overfitting, Random Forest provides excellent results.

4.2. Extreme Gradient Boosting (XGBoost)

As a result of gradient-boosted decision trees, XGBoost primarily aims for speed and efficiency. First applied by Tianqi Chen [29], multiple designers have learned to use the technique for machine boosting. It is an instrument that fits well into the Distributed Machine Learning Community (DMLC). A major advantage of XGBoost is that it works on every byte of memory as well as hardware properties in order to benefit tree boosting algorithms. Additionally, it can be incorporated in figuring conditions and offers support for algorithm enhancement and model alteration. As a result of the algorithm, processing time is minimised and memory assets are utilised optimally. As a result, it has the ability to be used in situations where qualities are missing and it will provide analogous structures in the construction of trees, and then will be able to enhance the auxiliary data previously gathered through the competent model. There are tree algorithms built into XGBoost. Currently, the tree ruptures into branches and edges based on the condition at the root node. Generally, piercing is ended to impact a choice at the leaf node, which does not have any extra edges [30].

4.3. Support Vector Machine (SVM)

Various types of data from different subjects are used to train the Support Vector Machine (SVM), which is considered a supervised learning method. As a result of SVM, hyperplanes or multiple hyperplanes are created in a high-dimensional space [31]. In order to determine the best hyperplane, we need to separate the input data into various classes based on their major partition [28]. There are various kernel functions that can be applied to the non-linear classifier in order to calculate the margins between hyperplanes. It is usually necessary to divide data into two types of data in order to perform a classification task, namely, training data and test data. By learning through attack patterns while training the data, SVM is an efficient memory-consuming algorithm capable of detecting real-time attacks, but it is inconvenient to handle large data sets with many instances and is highly sensitive to noise near the hyperplane [32].

4.4. Performance Metrics

Test datasets are used to determine the effectiveness of a machine learning algorithm after it has been implemented. Similarly to performance metrics, machine learning tasks can be categorised into classifications or regressions. Testing different machine learning algorithms can be done with different performance metrics. In order for an algorithm to be efficient, it must be evaluated. A performance metric was used to evaluate the model after it had been trained and then the model was computed based on its performance metrics. To determine which algorithm is most suitable for a particular application, different algorithms are often evaluated. It is imperative to note that machine learning models are designed to interpret newly acquired data well. Metrics used to evaluate the generalisation of the model are important when dealing with newly collected datasets. When evaluating a model, it is sometimes necessary to look at other metrics besides classification accuracy to evaluate its efficiency. The following performance metrics are used to evaluate our model.

Confusion Matrix

In cases where output has two or more kinds of classes, it's the simplest and most concise method of determining classification performance. Model accuracy and reliability can be determined by a confusion matrix, one of the simplest metrics. In binary classification problems, the results are described by two classes, while in multi-class classification problems, the results are described by more than two classes. For the purpose of computing the confusion matrix in this paper, four essential scenarios have been taken into account.

- **False Positive**: A false positive is when a positive attack is forecast and the detection is actually a normal attack.
- **False Negatives**: These are attacks that were considered normal by the model but were malicious in nature. A prediction that turned out to be incorrect.
- **True Positive**: It measures how often attacks are correctly identified as positives.
- **True Negative**: This measures how many times attacks have been incorrectly categorised as negative.

Precision

A classifier's prediction accuracy is represented by the ratio of correctly predicted positives to the overall number of positive predictions. In order to assess the model's ability to correctly categorise positive values, it is used as one of the models' key parameters.

$$\boldsymbol{Precision} = \frac{\boldsymbol{True_{positive}}}{\boldsymbol{True_{positive}} + \boldsymbol{False_{positive}}}$$

Recall

This is the number of positive samples that were correctly classified as positive in comparison to how many positive samples were classified incorrectly. Recall is used to measure how well a model can detect positive samples. Positive samples are detected more frequently when the recall is higher [33].

$$\boldsymbol{Recall = \frac{True_{positive}}{True_{positive} + False_{negative}}}$$

Classification Accuracy

Classification accuracy is a percentage based on the number of predictions that were made on the test set divided by the number of correct predictions. The performance of classification algorithms can be assessed using this metric, which is considered to be one of the simplest. It is more likely the model will be accurate if all data points are classified as majority if the dataset is significantly imbalanced. This means that a model's efficiency should be evaluated by considering more than just accuracy.

$$\boldsymbol{Accuracy = \frac{True_{positive} + True_{negative}}{True_{positive} + True_{negative} + False_{positive} + False_{negative}}}$$

F1 Score

It represents both recall and precision as a weighted average. F1 ranges from 0 to 1, so 1 is the most favourable value and 0 is the most unfavourable. The robustness and precision of a classifier are determined by this factor.

$$F1\ score = 2 * \frac{Precision\ * Recall}{Precision + Recall}$$

5. Implementation

The goals of this section are to describe and explain all the tools and technologies used in implementing the model.

5.1. Python

Object-oriented and high-level programming languages like Python enable artificial intelligence and machine learning as well as data analysis, visualisation, and programming applications. Building machine learning models is very easy with Python codes because humans easily understand them. Code that is readable and precise is produced. Algorithms based on AI and machine learning require a lot of time to implement. Well-structured and thoroughly tested environments are essential to helping developers create the most efficient coding solutions. Python offers a wide variety of machine learning and artificial intelligence libraries due to its rich collection of technologies.

5.2. Jupyter Notebook

Creating and sharing computational documents is easy with the Jupyter Notebook web application. It is an open source application designed to simplify, streamline, and focus on documents. Data cleaning and conversion, mathematical modelling, numerical simulation, machine learning, and mathematical modelling are some of the applications. To prepare data, analyse data, and build machine learning models, this tool was used.

5.3. Libraries

To solve common programming tasks, developers use software libraries, which contain prewritten code. To reduce development time, programmers use Python frameworks and libraries. Seaborn, Matplotlib, Pandas, NumPy and other Python libraries are all included in the language. SciKit-Learn, Matplotlib, Pandas and Seaborn are some of the machine learning libraries used in the implementation of the proposed model.

Scikit-learn

Machine learning models can be implemented in this Python library as well as tools for analysing them with statistical methods. Inbuilt datasets can be used, assembly techniques can be implemented, points can be extracted, features can be selected, and dimensionality can be reduced. In order to build classifier models for SVM, Random Forest, and XGBoost, we used this library.

Seaborn

Statistical graphics can be made in Python using the seaborn library. In addition, it utilises pandas data structures to provide a high-level interface to matplotlib. With a dataset and a plot specification, seaborn translates data values into visual attributes like colour, size, and style, computes statistical transformations, and embellishes the plot with informative axis labels and a legend [34]. Rapid prototyping and exploratory data analysis are made easier with seaborn's simple function call. Furthermore, it allows users to create polished, publication-quality figures due to its extensive customization options and exposure of the underlying matplotlib objects.

Matplotlib

In Python, Matplotlib provides tools for creating static, animated, and interactive visualisations. Machine learning technologies often require Python-based resources such as matplotlib when modelling them. Using these libraries is the idea of integrating the results with other components of a machine learning program, neural network or some other advanced machine from within a Python environment. The main purpose of matplotlib is to provide tools for visual plotting. This implies that it is more analytical than generative in nature. The infrastructure described above, however, enables machine learning programs to generate results that can be used by humans.

Pandas
Data manipulation and analysis is carried out primarily using Pandas by matplotlib. As one of Pandas' many features, it offers a 2D data frame object that can be stored in memory. The pandas library is not a matplotlib dependency, unlike numpy. Using pandas, you can easily operate on “relational” or “labelled” data in a fast, flexible, and expressive manner. As a high-level building block for Python-based data analysis, it aims to be the foundation for all practical data analysis in the real world. The tool also aims to become the most powerful and flexible open source data analysis/manipulation tool available in any language. In data science, data is usually munged and cleaned, analysed / modelled, then arranged into a format suitable for plotting or tabular presentation in several stages. These tasks are all easily accomplished with Pandas.

6. Evaluation

In this section, an evaluation of the proposed model's efficiency is presented. In order to evaluate all algorithms and compare them to our proposed model for detecting IoT network intrusions, we tested all algorithms on the Network Intrusion Detection dataset and IoT Device Network Logs dataset. In the end, the best performing classifier was selected based on the results of the experiments.

6.1. Detecting Network Intrusion using Random Forest Algorithm

The first algorithm is the Random Forest Algorithm used for both datasets. The results are shown in the table below.

Dataset	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Network Intrusion Dataset	0.99	0.99	0.99	0.99
IoT Device Network logs	0.83	0.88	0.83	0.82

Table 1: Results From Both Datasets For Random Forest Algorithm

It is evident in this algorithm that Accuracy, Precision, Recall, F1 Score, and ROC Curve are all identical in the first dataset. Since the dataset had already been preprocessed prior to use, this was a natural outcome. This however, is rare in a real world scenario. The model gave near-perfect results after further preprocessing which made it difficult to know if the prediction was accurate or not when working on an unprocessed dataset. The algorithm performed well on the second dataset as well, which suggests it is not overfit.

6.2. Detecting Network Intrusion using Support Vector Machine (SVM)

The second algorithm is the SVM Algorithm used for both datasets. The results are shown in the table below.

Dataset	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Network Intrusion Dataset	0.53	0.64	0.54	0.38
IoT Device Network logs	0.67	0.72	0.68	0.61

Table 2: Results From Both Datasets For SVM Algorithm

Based on the results in the table above, it is evident that SVM performed poorly in comparison with the other algorithms. Several factors contribute to its reduced reliability, including the high computational efforts required to analyse a dataset. When we have large datasets, the algorithm does not perform well because it requires more training time. Due to the overwhelming computational effort involved in processing the second dataset, we were forced to use 10% of the dataset in order to derive results.

6.3. Detecting Network Intrusion using Extreme Gradient Boosting (XGBoost)

The final algorithm is the XGBoost Algorithm used for both datasets. The results are shown in the table below

Dataset	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Network Intrusion Dataset	0.99	0.99	0.99	0.99
IoT Device Network logs	0.83	0.88	0.83	0.82

Table 3: Results From Both Datasets For XGBoost Algorithm

Based on the table above, it appears that the results are similar to those of the Random Forest. Because the first dataset had been preprocessed before usage, it appears to be overfit. As a result of using the model on the second dataset, it was observed that the algorithm also performed well. In terms of processing time, Random Forest and XGBoost are different from each other. As compared to XGBoost, Random Forest had a faster processing time.

6.4. Discussion

The experiments were carried out for a binary classification and a multi-label classification using two datasets. Table 1, 2 and 3 show the results. The Type I error occurs in the first dataset when we identify a packet as a malicious packet when it actually is a normal packet. This is because normal packets were classified as 0 and anomalous packets as 1. In type II errors, packets are mistakenly treated as normal packets when they are actually anomaly packets. Type II errors should be minimised in the ideal scenario. There were five different types of attacks observed in the second dataset: 0 - normal, 1 - wrong setup, 2 - DDoS, 3 - Data type probing, 4 - scan attack, 5 - man in the middle attack. This shows multi-label classification. As can be noticed from Table 1, the Random forest approach received an accuracy of 99.42%, precision of 99.42%, recall rate of 99% and f1 score of 99% in the first dataset. In contrast, an accuracy of 83%, precision of 88%, recall rate of 83% and f1 score of

82% was recorded from the second dataset. The performance in the first dataset was significantly better than that of the second. When some parameters for the final predictions were modified, it was observed that there was little to no change in the outcomes. This was because most of the null/missing values and incorrect entries were already removed due to preprocessing prior to use. This, however, is not common in real-life situations. After further preprocessing, the model produced near-perfect results, making it difficult to determine whether the prediction was accurate or not. In order to solve this, the correlation scores with highest and lowest scores which would impact the predictions were removed but the model still gave a near flawless result. In the second dataset, the algorithm performed well as well, suggesting it was not overfitted.

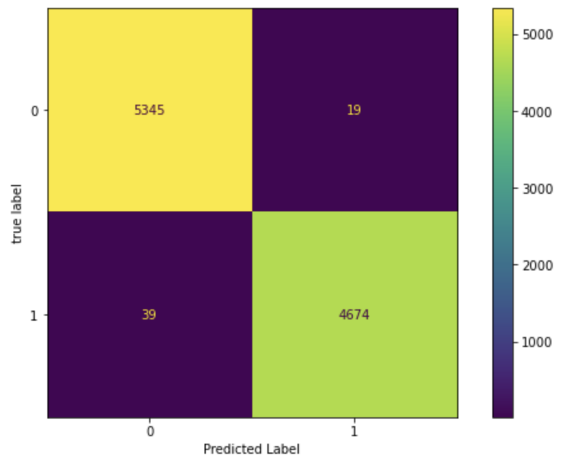


Fig. 6: Confusion Matrix for Random Forest approach using NID dataset

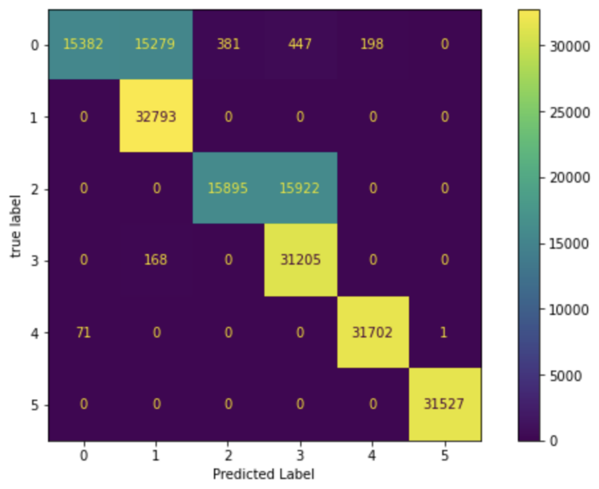


Fig. 7: Confusion Matrix for Random Forest approach using IoT dataset

The experiments using the SVM classifier gave the poorest results, as can be seen in Table 2. With an accuracy rate of 53.60% and 67.63%, precision rate of 64% and 72%, recall rate of 54% and 68% and F1 score of 38% and 61% for the first and second datasets respectively. Although SVM algorithms are highly accurate for classification and have good theoretical foundations, their high training complexity makes them unsuitable for large data sets [35]. In this study, we used a fraction of the second dataset to obtain the results we have. This was because the SVM algorithm requires a large amount of computational effort, making the entire dataset unable to be processed. Therefore, we processed 10% of the dataset to generate these results. This algorithm took too long to run compared to the other algorithms.

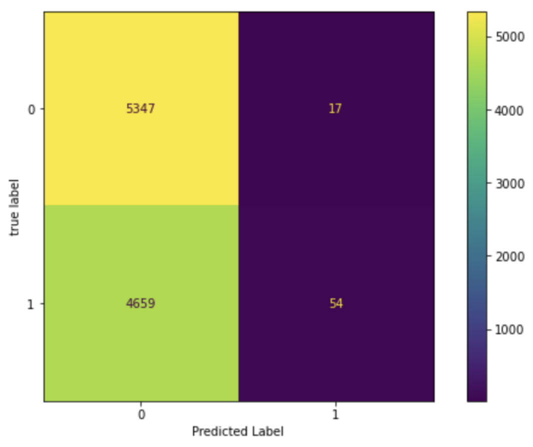


Fig. 8: Confusion Matrix for SVM approach using NID dataset

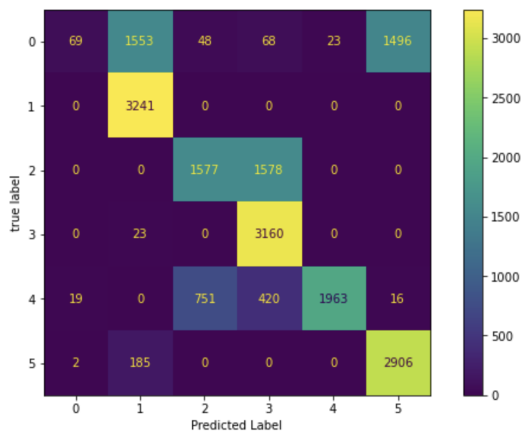


Fig. 9: Confusion Matrix for SVM approach using IoT dataset

The results from the XGBoost approach are found in Table 3. For the test in the first dataset, the accuracy rate reached 99.44%, the precision rate reached 99%, the recall score reached 99% and the f1 score reached 99%, and for the trials in the second dataset, the accuracy rate reached 82.99%, the precision rate reached 88.36%, the recall score reached 83% and the f1 score reached 82%. These tasks were completed with fewer computational efforts.

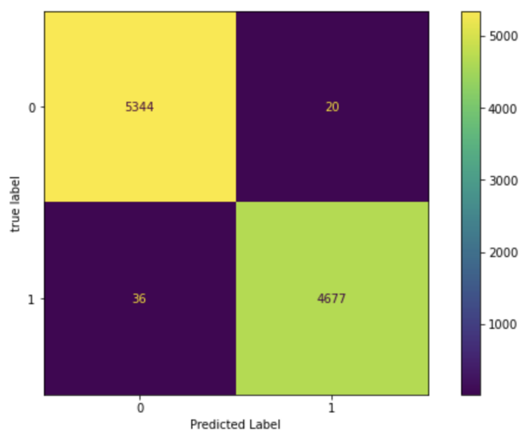


Fig. 10: Confusion Matrix for XGBoost approach using NID dataset

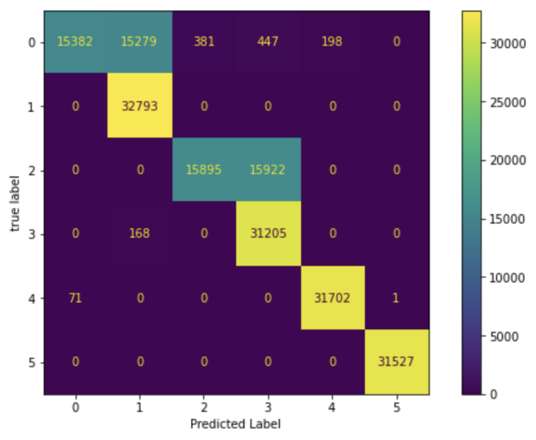


Fig. 11: Confusion Matrix for XGBoost approach using IoT dataset

In general, XGBoost and Random Forest provided the highest metrics scores but XGBoost had a higher computational effort than Random Forest. In contrast, SVM approaches utilised a high amount of computation resources and yielded minimally satisfactory results. Accuracy and other measures of the Random Forest and XGBoost approaches were very favourable. Random Forest is preferred over the other approaches when it comes to real-time detection.

Table 4: Evaluation comparison of Algorithms among different datasets

Network Intrusion Dataset				
Algorithm	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Random Forest	0.99	0.99	0.99	0.99
XGBoost	0.99	0.99	0.99	0.99
SVM	0.53	0.64	0.54	0.38
IoT Dataset				
Algorithm	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Random Forest	0.83	0.88	0.83	0.82
XGBoost	0.83	0.88	0.83	0.82
SVM	0.67	0.72	0.68	0.61

7. Conclusion and Future Work

Using multiple machine learning approaches, we explored intrusion detection in the IoT Device Network Logs dataset and the Network Intrusion Detection dataset to enhance loT security. The two datasets performed very well as far as accuracy and efficiency are concerned. The Random Forest approach had a high degree of accuracy (99.42%), and based on this, helped in answering the research question “What role can machine learning play in solving intrusion detection in IoT as security concerns are rapidly growing?”. The proposed IDS is based on the Random Forest Algorithm and was evaluated on two datasets; the IoT Device Network Logs dataset and the Network Intrusion Detection dataset. This algorithm was compared to the XGBoost and SVM algorithms. Results demonstrate that it had the highest accuracy of 99% in the first dataset and 83% in the second dataset. It also had the fastest run time of 412 milliseconds in the first dataset and 5.5 seconds in the second dataset. Additionally, it reduces computing complexity, which is one of its most attractive features. In the field of network security, intrusion detection systems are very significant. By learning and identifying triggers and behavioural patterns, intrusions can be detected more efficiently. It helps companies meet security regulations and maintain regulatory compliance because it provides increased visibility across the entire network. Research like this can contribute to designing the IDS of the future, which is vital when it comes to security. Random Forest's accuracy and effectiveness when predicting datasets are demonstrated in these findings. By applying these findings to future architectures like IoT, 5G, and others, robust Intrusion Detection Systems can be designed. Using different datasets would also give the opportunity to compare our model's performance with that of other algorithms.

References

[1] Y. Y. Aung and M. M. Min, “An analysis of random forest algorithm based network intrusion detection system,” in *2017 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, Jun. 2017, pp. 127–132. doi: 10.1109/SNPD.2017.8022711.

[2] “Identity Theft Resource Center’s 2021 Annual Data Breach Report Sets New Record for Number of Compromises,” *ITRC*. https://www.idtheftcenter.org/post/identity-theft-resource-center-2021-annual-data-breach-report-sets-new-record-for-number-of-compromises/ (accessed Oct. 19, 2022).

[3] Rob Sobers, “166 Cybersecurity Statistics and Trends.” https://www.varonis.com/blog/cybersecurity-statistics (accessed Oct. 17, 2022).

[4] E. P. Nugroho, T. Djatna, I. S. Sitanggang, A. Buono, and I. Hermadi, “A Review of Intrusion Detection System in IoT with Machine Learning Approach: Current and Future Research,” in *2020 6th International Conference on Science in Information Technology (ICSITech)*, Oct. 2020, pp. 138–143. doi: 10.1109/ICSITech49800.2020.9392075.

[5] M. Choubisa, R. Doshi, N. Khatri, and K. Kant Hiran, “A Simple and Robust Approach of Random Forest for Intrusion Detection System in Cyber Security,” in *2022 International Conference on IoT and Blockchain Technology (ICIBT)*, May 2022, pp. 1–5. doi: 10.1109/ICIBT52874.2022.9807766.

[6] W. Meng, “Intrusion Detection in the Era of IoT: Building Trust via Traffic Filtering and Sampling,” *Computer*, vol. 51, no. 7, pp. 36–43, Jul. 2018, doi: 10.1109/MC.2018.3011034.

[7] N-able, “Intrusion Detection System (IDS): Signature vs. Anomaly-Based,” *N-able*, Mar. 15, 2021. https://www.n-able.com/blog/intrusion-detection-system (accessed Oct. 30, 2022).

[8] P. Prajapati, B. Bhatt, G. Zalavadiya, M. Ajwalia, and P. Shah, “A Review on Recent Intrusion Detection Systems and Intrusion Prevention Systems in IoT,” in *2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Jan. 2021, pp. 588–593. doi: 10.1109/Confluence51648.2021.9377202.

[9] Z. Liu, N. Thapa, A. Shaver, K. Roy, X. Yuan, and S. Khorsandroo, “Anomaly Detection on IoT Network Intrusion Using Machine Learning,” in *2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)*, Aug. 2020, pp. 1–5. doi: 10.1109/icABCD49160.2020.9183842.

[10] M. AL-Hawawreh, N. Moustafa, and E. Sitnikova, “Identification of malicious activities in industrial internet of things based on deep learning models,” *J. Inf. Secur. Appl.*, vol. 41, pp. 1–11, Aug. 2018, doi: 10.1016/j.jisa.2018.05.002.

[11] V. Morfino and S. Rampone, “Towards Near-Real-Time Intrusion Detection for IoT Devices using Supervised Learning and Apache Spark,” *Electronics*, vol. 9, no. 3, Art. no. 3, Mar. 2020, doi: 10.3390/electronics9030444.

[12] H. Vargas, C. Lozano-Garzon, G. Montoya, and Y. Donoso, “Detection of Security Attacks in Industrial IoT Networks: A Blockchain and Machine Learning Approach,” *Electronics*, vol. 10, p. 2662, Oct. 2021, doi: 10.3390/electronics10212662.

[13] S. Kejriwal, D. Patadia, S. Dagli, and P. Tawde, “Machine Learning Based Intrusion Detection,” in *2022 IEEE Fourth International Conference on Advances in Electronics, Computers and Communications (ICAECC)*, Jan. 2022, pp. 1–5. doi: 10.1109/ICAECC54045.2022.9716648.

[14] S. Zeadally and M. Tsikerdekis, “Securing Internet of Things (IoT) with machine learning,” *Int. J. Commun. Syst.*, vol. 33, no. 1, Jan. 2020, doi: 10.1002/dac.4169.

[15] “M-IDM: A Multi-Classification Based Intrusion Detection Model in Healthcare IoT.” https://www.techscience.com/cmc/v67n2/41342/html (accessed Nov. 07, 2022).

[16] C. Cervantes, D. Poplade, M. Nogueira, and A. Santos, “Detection of sinkhole attacks for supporting secure routing on 6LoWPAN for Internet of Things,” in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, May 2015, pp. 606–611. doi: 10.1109/INM.2015.7140344.

[17] A. Husain, A. Salem, C. Jim, and G. Dimitoglou, “Development of an Efficient Network Intrusion Detection Model Using Extreme Gradient Boosting (XGBoost) on the UNSW-NB15 Dataset,” in *2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, Dec. 2019, pp. 1–7. doi: 10.1109/ISSPIT47144.2019.9001867.

[18] E. M. Maseno, Z. Wang, and H. Xing, “A Systematic Review on Hybrid Intrusion Detection System,” *Secur. Commun. Netw.*, vol. 2022, p. e9663052, May 2022, doi: 10.1155/2022/9663052.

[19] K. Rai, M. Devi, D. Professor, and A. Guleria, “Decision Tree Based Algorithm for Intrusion Detection,” *Int. J. Adv. Netw. Appl.*, vol. 07, pp. 2828–2834, Jan. 2016.

[20] I. Ullah and Q. H. Mahmoud, “Design and Development of a Deep Learning-Based Model for Anomaly Detection in IoT Networks,” *IEEE Access*, vol. 9, pp. 103906–103926, 2021, doi: 10.1109/ACCESS.2021.3094024.

[21] P. Eichhammer *et al.*, “Towards a Robust, Self-Organizing IoT Platform for Secure and Dependable Service Execution,” 2019, doi: 10.18420/FBSYS2019-03.

[22] M. S. Ali, M. Vecchio, M. R. Pincheira Caro, K. Dolui, F. Antonelli, and M. H. Rehmani, “Applications of Blockchains in the Internet of Things: A Comprehensive Survey,” *IEEE Commun. Surv. Tutor.*, vol. PP, pp. 1–1, Nov. 2018, doi: 10.1109/COMST.2018.2886932.

[23] N. Hotz, “What is CRISP DM?,” *Data Science Process Alliance*, Sep. 10, 2018. https://www.datascience-pm.com/crisp-dm-2/ (accessed Nov. 05, 2022).

[24] Quantum, “Data Science project management methodologies,” *Medium*, Aug. 20, 2019. https://medium.datadriveninvestor.com/data-science-project-management-methodologies-f6913c6b29eb (accessed Nov. 07, 2022).

[25] “Network Intrusion Detection.” https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection (accessed Nov. 02, 2022).

[26] “Iot Device Network Logs.” https://www.kaggle.com/datasets/437f30ccd4607ae8b5973e241248bffccf0c67f6d07b9f4dcb800a682150bed8 (accessed Dec. 07, 2022).

[27] “What is Feature Selection? Definition and FAQs | HEAVY.AI.” https://www.heavy.ai/technical-glossary/feature-selection (accessed Nov. 16, 2022).

[28] S. Abdelhamid, M. Aref, I. Hegazy, and M. Roushdy, “A Survey on Learning-Based Intrusion Detection Systems for IoT Networks,” in *2021 Tenth International Conference on Intelligent Computing and Information Systems (ICICIS)*, Dec. 2021, pp. 278–288. doi: 10.1109/ICICIS52592.2021.9694226.

[29] R. R. Rai , “XGBoost: The Excalibur for Everyone,” *Medium*, Jun. 24, 2018. https://towardsdatascience.com/xgboost-the-excalibur-for-everyone-8009bd015f1e (accessed Nov. 04, 2022).

[30] H. Azwar, M. Murtaz, M. Siddique, and S. Rehman, “Intrusion Detection in secure network for Cybersecurity systems using Machine Learning and Data Mining,” in *2018*

IEEE 5th International Conference on Engineering Technologies and Applied Sciences (ICETAS), Nov. 2018, pp. 1–9. doi: 10.1109/ICETAS.2018.8629197.

[31] E. Benkhelifa, T. Welsh, and W. Hamouda, “A Critical Review of Practices and Challenges in Intrusion Detection Systems for IoT: Toward Universal and Resilient Systems,” *IEEE Commun. Surv. Tutor.*, vol. 20, no. 4, pp. 3496–3509, 2018, doi: 10.1109/COMST.2018.2844742.

[32] H. Liu and B. Lang, “Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey,” *Appl. Sci.*, vol. 9, p. 4396, Oct. 2019, doi: 10.3390/app9204396.

[33] “Accuracy, Precision, and Recall in Deep Learning,” *Paperspace Blog*, Oct. 12, 2020. https://blog.paperspace.com/deep-learning-metrics-precision-recall-accuracy/ (accessed Nov. 05, 2022).

[34] M. L. Waskom, “seaborn: statistical data visualization,” *J. Open Source Softw.*, vol. 6, no. 60, p. 3021, Apr. 2021, doi: 10.21105/joss.03021.

[35] J. Cervantes, X. Li, and W. Yu, “SVM Classification for Large Data Sets by Considering Models of Classes Distribution,” in *2007 Sixth Mexican International Conference on Artificial Intelligence, Special Session (MICAI)*, Nov. 2007, pp. 51–60. doi: 10.1109/MICAI.2007.27.