

Classification And Explanation of Different Internet of Things (IoT) Network Attacks Using Machine Learning, Deep Learning And XAI

by

Anika Tasnim

18301047

Nigah Hossain

18101204

Sabrina Tabassum

18301135

Nazia Parvin

18301140

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
May 2022

© 2022. Brac University
All rights reserved.

Declaration


It is therefore stated that

1. The dissertation we submitted was originally written during our study at Brac University.
2. Unless adequately cited and referenced, the thesis does not include any content that has already been published or created by a third party.
3. The thesis includes no material which has been approved or submitted for a degree or certificate from another university or institution.
4. We have acknowledged all significant sources of support.

Student's Full Name & Signature:



Anika Tasnim
18301047



Nigah Hossain
18101204



Sabrina Tabassum
18301135



Nazia Parvin
18301140

Approval

The thesis entitled “Classification And Explanation of Different Internet of Things (IoT) Network Attacks Using Machine Learning, Deep Learning And XAI” submitted by

1. Anika Tasnim(18301047)
2. Nigah Hossain(18101204)
3. Sabrina Tabassum(18301135)
4. Nazia Parvin(18301140)

Of the Spring of 2022 was recognized as adequate in partial completion of the prerequisite for the degree of B.Sc. in Computer Science and Engineering on August 23, 2015.

Examining Committee:

Supervisor:
(Member)



Dr. Muhammad Iqbal Hossain
Assistant Professor
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)



Rafeed Rahman
Lecturer
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

The internet of things is one of today's most revolutionary technologies. Because of its pervasiveness, increasing network connection capacity, and diversity of linked items, the internet of things (IoT) is adaptable and versatile. The most common problem impeding IoT growth is insufficient security measures. The threat of data breaches is always there since smart gadgets gather and transmit sensitive information that, if disclosed, might have severe consequences. Modern advances in Artificial Intelligence are providing new Machine Learning and Deep Learning approaches to address more complex issues with greater model performance. This predictive capacity, however, comes at the cost of growing complexity, which can make these models hard to understand and interpret. Though these models give highly precise results, an explanation is required in order to comprehend and accept the model's decisions. Here comes XAI which emphasizes a variety of ways for breaking the black-box nature of Machine Learning and Deep Learning models as well as delivering human-level explanations. In this article, to identify and classify IoT network attacks, we have analyzed six machine learning and deep learning approaches: Decision Tree, Random Forest, AdaBoost, XGBoost, ANN, and MLP. Accuracy, Precision, Recall, F1-Score, and Confusion Matrix are some of the metrics we have used to evaluate our models. We have achieved fairly impressive results (above 96%) in binary classification for all the techniques. When all of the classifiers were analyzed, Decision Tree and Random Forest outperformed all others (above 99%) for both binary and multiclass classification. Adaboost and ANN, on the other hand, perform badly for multiclass classification. We have also applied Undersampling, Oversampling, and SMOTE techniques on a dataset to reduce data skewness and to evaluate multiple ML and DL algorithms. We have used LIME, SHAP, and ELI5 approaches to interpret and explain our models. The feasibility of the techniques suggested in this work is demonstrated in the IoT/IIoT dataset of TON_IoT datasets, which incorporate data obtained from telemetry datasets of IoT and IIoT sensors.

Keywords: Machine Learning; Prediction; Decision tree; Random Forest; XG-Boost; Adaboost; XAI; Model.

Acknowledgement

We would like to express our heartfelt gratitude to everyone who helped us in completing this thesis.

Firstly, thanks to the almighty Allah, for allowing us to complete our thesis without any major hurdles.

Secondly, we would like to express our sincere gratitude to our supervisor, Muhammad Iqbal Hossain sir, and co-advisor, Rafeed Rahman sir, for their guidance, constructive feedback, persistent support, and encouragement.

Thirdly, the whole judging panel of The 2022 International Conference on Decision Aid Sciences and Applications (DASA'22) for accepting our paper and publishing it in IEEEExplore by giving all the reviews that helped us develop our later works.

Finally, without our parents' unwavering support, none of this would be possible. We are on the edge of graduating as a result of their kind contributions and prayers.

Table of Contents

Declaration	i
Approval	ii
Abstract	iii
Acknowledgment	iv
Table of Contents	v
List of Figures	vii
List of Tables	viii
Nomenclature	ix
1 Introduction	1
1.1 Introduction	1
1.2 Research Problem	2
1.3 Research Objectives	3
2 Related Works	4
2.1 Literature Review	4
2.1.1 Internet of Things (IoT)	4
2.1.2 Related Works	4
2.2 Background	7
2.2.1 Decision Tree	7
2.2.2 Random Forest (RF)	8
2.2.3 Adaboost	9
2.2.4 XGBoost	9
2.2.5 Multilayer Perceptron	10
2.2.6 ANN	10
2.2.7 LIME	11
2.2.8 SHAP	12
2.2.9 ELI5	12
2.2.10 SKLearn	13

3	Methodology	14
3.1	Methodology	14
3.1.1	Overview of the Dataset	15
3.1.2	Data preprocessing	15
3.1.3	Feature Scaling	16
3.1.4	Feature selection	17
3.1.5	Attacks	17
4	Implementation	22
4.1	Implementation	22
5	Result	25
5.1	Result	25
5.1.1	XAI Result Analysis	26
6	Conclusion and Future Work	32
6.1	Conclusion and Future Work	32
	Bibliography	34

List of Figures

2.1	Decision Tree	8
2.2	ANN	10
3.1	The flow chart of the proposed model	14
3.2	ToN IoT Dataset(IoT_Fridge) sample	16
3.3	Number of Attacks on Different IoT Devices	17
4.1	Confusion Matrix of Decision Tree Binary Classification of Fridge . .	23
4.2	Confusion Matrix of Decision Tree Binary Classification of Garage Door	23
4.3	Confusion Matrix of Decision Tree Multiclass classification of Fridge	24
4.4	Confusion Matrix of XGBoost Multiclass classification of Garage Door	24
5.1	A Summary of Evaluation Metrics of Imbalanced Dataset	25
5.2	A Summary of Evaluation Metrics of Balanced IoT_Fridge Dataset .	26
5.3	LIME explanation of Decision Tree on Modbus dataset	27
5.4	LIME explanation of XGBoost on GPS Tracker dataset	27
5.5	Feature importance calculated by SHAP value of Fridge dataset . . .	28
5.6	SHAP Summary Plot for Fridge dataset using Decision Tree	29
5.7	SHAP Summary Plot for Fridge dataset using MLP	30
5.8	Feature importance of Fridge dataset using ELI5	30
5.9	ELI5 explanation of a prediction using Random Forest in Fridge dataset	31

List of Tables

3.1	Feature Type and Description of IoT Fridge	18
3.2	Feature Type and Description of IoT Garage_Door	18
3.3	Feature Type and Description of IoT GPS_Tracker	19
3.4	Feature Type and Description of IoT Modbus	19
3.5	Feature Type and Description of IoT Motion_Light	19
3.6	Feature Type and Description of IoT Thermostat	20
3.7	Feature Type and Description of IoT Weather	20

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

BLSTM Bidirectional Long Short Term Memory

BTLE Bluetooth Low Energy

CNN Convolution Neural Network

CPPS Cyber-Physical Production System

CPS Cyber-Physical Systems

IoT Internet of Things

LIME LOCAL INTERPRETABLE MODEL-AGNOSTIC EXPLANATIONS

MitM Man-in-the-Middle

ML Machine Learning

MLP Multi Layer Perceptron

MMU Memory Management Unit

OR Runs scored by the opponent team

RF Random Forest

SHAP SHapley Additive exPlanations

VLSTM Variational Long Short-Term Memory

XAI Explainable Artificial Intelligence

Chapter 1

Introduction

1.1 Introduction

The Internet of Things (IoT), a new innovation in communication technology, has significantly outperformed traditional environmental sensors. IoT technologies have the ability to gather, quantify, and grasp data about the environment, enabling modernizations that improve living standards. This enables smart cities to be realized by simplifying new forms of communication between things and humans. [20] The Internet of Things has progressed from a theoretical concept to a key issue for many people's lives and enterprises in recent years. When companies connect IoT devices to an existing network infrastructure, they are looking for new ways to manage and use the collected data. According to [23], Juniper Research's most current prediction, the amount of IoT devices would reach 46 billion by 2021, with 31 billion new IoT devices deployed globally by the end of 2021. There is a diverse range of usage of IoT technology in the modern world. Everything from a watch to a TV to a refrigerator to a fan to a light bulb has become smart lately. The Internet of Things has been broadly used in fields such as healthcare, agriculture, transportation and storage, wearables, smart home applications, electricity [26]. Its goals include the linking of low-value devices via Wireless Sensor Networks. This concept is modified by Cyber-Physical Systems (CPS), which drives automated commercial structures closer to Cyber-Physical Production Systems (CPPS) and the fourth commercial revolution is one of its pillars. Through the usage of CPPS, Industry 4.0 aspires for autonomous navigation, real-time monitoring, participatory production, reduce errors and persistent virtual interconnection. IoT systems are intricate and contain a number of interconnected components. As a result, maintaining the security requirement in an IoT system with a huge area of attack is difficult. Solutions must adopt a holistic approach to satisfy the security need. IoT devices, on the other hand, are typically used in an unattended setting. As a result, an intruder could gain physical access to these IoT devices. Wireless networks are commonly used to link IoT devices, which allow an attacker to eavesdrop on a communication channel and gain access to sensitive information. This Internet of Things (IoT) system is also a cyber-physical system, but it has some resource constraints. Due to limited computing, communication, and power resources, as well as reliable interaction with a physical domain, notably the behavior of a physical environment in unusual and unanticipated modes, sophisticated security structures are necessary. In addition, the IoT environment introduces additional attack vectors. The interdependent and

linked surroundings of the IoT create such attack surfaces. As a result, security in IoT systems is at a larger risk than in conventional computer systems. So regular solutions could be inadequate for such structures [26]. There are some inherent characteristics of IoT systems such as diversity, abundant data production, inter dependency between devices, critical architectures, constraints of resources, power supply, and time delay, etc which make them extremely difficult to protect. IoT systems are diverse in terms of application areas, communication protocols, and hardware platforms. Consequently, building common system protection for heterogeneous devices is difficult, particularly in the industrial domain. IoT-enabled gadgets may perform a wide range of duties since they can connect to a wider network. These devices deal with a large amount of data and handle data in vast quantities and share it with one another. Furthermore, since IoT devices are tiny and diverse, many forms of cyber attacks may occur extremely quickly. The interaction between devices is increasing day by day. As a result, their interdependence is also growing. The target system itself may not be directly hacked, but attackers can simply modify the behavior of other interdependent devices or the adjacent environment to attain the hackers' aims. Again Some lightweight IoT devices lack a memory management unit (MMU). Many sophisticated encryption and authentication methods are used on such devices, occupying too much processing power and causing a significant delay, which impairs the regular functioning of these devices and decreases performance, particularly for real-time IoT devices. As a result, it is simple for attackers to infiltrate these devices by exploiting memory vulnerabilities. The typical high-end security protection solution cannot function correctly in a system with limited resources and diverse components [10][26]. However, this raises a new set of issues: how to keep all of that data safe. It can have severe consequences if an IoT connection is not effectively protected. In order to protect connected IoT devices from sophisticated network attacks, different machine learning (ML) techniques are applied. Deep learning (DL), a cutting-edge ML approach, in particular, offers a one-of-a-kind capability for automatic feature extraction from wide-ranging, high speed network data produced by linked diverse IoT devices. The fact that deep learning is fueled by huge amounts of data is a significant benefit and a vital factor in understanding why it is gaining popularity. The "Big Data Era" of technology will open up a range of new prospects for deep learning innovation. Additionally, to comprehend the results and output created from machine learning algorithms by the average human users, XAI or Explainable Artificial Intelligence, a set of methods and processes is used here. In AI-powered decision making, it's used to assess model correctness, fairness, transparency, and results. In XAI, more explainable models are produced and a high level of learning performance is maintained at the same time. The presence of attacks in the IoT environment can be determined by ML algorithms and those results from ML algorithms can be easily represented by using XAI.

1.2 Research Problem

IoT has already emerged as a serious security issue worldwide. The Internet of Things is gradually becoming a security concern. As we can see, Baby monitors, webcams, thermostats, smart fridges, assault weapons, and even medicine infusion pumps have all been hacked. With so many nodes that are being added and that

are already added to the internet and networks, criminals will get a variety of attack pathways and choices to continue their evil actions, especially because most of those contain security vulnerabilities [1]. To develop an efficient IoT device, a wide range of skills are needed. data acquired by an IoT sensor must pass via a variety of custom-developed systems whereas conventional sensors may store data locally or transport it using a relatively simple standard protocol such as Modbus. The sensor must interact with a mobile app or a custom hub, which involves the purchase of a separate device or the employment of a mobile app developer. A backend developer is required to upload the hub or mobile app to a cloud backend server and administer it there. A corporation must accumulate and evaluate backend data in order to develop a useful IoT solution [2]. Various technologies such as Z-Wave, ZigBee, Bluetooth, Wi-Fi, and Bluetooth Low Energy (BTLE), are competing to be the dominant means of connection between hubs and devices [7]. When a high number of devices must be linked, this causes significant problems; dense connection demands the deployment of extra hardware and software. The following information was revealed in the newest research report given by the State of IoT Security:

An IoT security code of practice is supported by 96% of enterprises and 90% of consumers.

Regardless of the way that 54% of users have a normal of 4 IoT gadgets, only 14% of clients feel they are instructed about IoT gadget security.

65% percent of customers are anxious about a hacker watching their IoT device, and 60% are concerned about their sensitive data being compromised. [9].

1.3 Research Objectives

The research goals that we want to accomplish are:

1. To assess the threats in IoT security.
2. To identify the attacks on the network layer.
3. To classify the attacks based on the use of IoT.
4. To explain and interpret different models

Chapter 2

Related Works

2.1 Literature Review

2.1.1 Internet of Things (IoT)

The Internet of Things (IoT) is one of the most important innovations in everyday life, and it will acquire popularity as more organizations understand the demand of linked gadgets in maintaining global competitiveness. Another aspect where IoT may be useful is the ability to monitor infrastructure activities. Sensors might be used to track activities or modifications in structural buildings, bridges, as well as other infrastructure, for example. This has a number of benefits, including cost reductions, time efficiency, and modifications to workflow, standard of living.

2.1.2 Related Works

Deep learning is a strong approach for detecting botnet attacks. However, for efficient performance, it requires a huge amount of network data. Working with high-dimensional data is also difficult. Implementing those techniques, however, is exceedingly challenging due to the memory limitations of IoT devices. A recent work [15] proposes a hybrid Deep Learning model called LAE-BLSTM which can detect IoT botnet attacks in networks and combines LAE and deep BLSTM algorithms. Feature dimensionality is compressed and reduced using a single hidden-layer long short-term memory autoencoder (LAE). This methodology lessens the dimensionality of huge scope IoT network traffic information and makes a low-dimensional component portrayal at the secret layer without compromising significant network data. Deep BLSTM is used to classify network traffic samples as well. To separate botnet malicious traffic from typical traffic in IoT networks, this methodology assesses the drawn-out interdependent changes in the low-dimensional list of capabilities given by LAE. Along with that, the Bot-IoT data set is being utilized to do extensive tests to see how well LAE-BLSTM performs in binary and multiclass classification. Detecting anomalies from multidimensional data for smart industrial applications is a difficult task in the Industrial Internet of Things.

The research [17] proposes an intelligent anomaly detection technique based on Variational Long Short-Term Memory (VLSTM). A compression network and an estimating network are the key components of the proposed VLSTM model where the compression network converts the multidimensional raw data into a low-dimensional

hidden variable and a lightweight estimate network is subsequently constructed to give classifications for anomaly detection. To learn the low-dimensional feature representation while minimizing substantial loss of critical information, a variational reparameterization approach is applied in collaboration with the LSTM encoder-decoder neural network. Variational Bayes is used to produce the hidden variable, which is then optimized with 3 loss functions. Based on the evaluation outcomes, the suggested VLSTM model could considerably increase the extraction of features, minimize the false detections rate and improve accuracy rate, indicating the applicability of our technique in detecting anomalies for Industrial Big Data. The [17] model considered nine attacks for this research. To evaluate their work, they used an available dataset UNSW-NB15.

The research [26] proposes an intrusion detection system for detecting DDoS botnet attacks on IoT devices using deep learning. The idea of an intrusion detection system (IDS) using Deep Learning was created to identify anomalies by studying traffic patterns on various IoT devices utilizing security checks. For IoT networks, Deep Neural Network (DNN) is capable of detecting IoT botnet attacks which is really adaptable. The proposed approach can be utilized to assist IoT devices in adapting to their dynamic ad hoc environments. Following that, this model is assessed and tested for accuracy and deployment readiness using a number of approaches such as AUC-ROC, kappa statistics, and so on. While establishing a real network environment, a BoT-IoT data set generated by a group from the UNSW Canberra Cyber Range Lab is used. Malware, which exploits the computer system, is used in a number of attacks on the network. In cyber physical networking, artificial intelligence can be used to identify and adjust for such threats. This sort of preventative model also relies on the application, such as smart farming. To safeguard malware architecture from destruction, secured encoding necessitates a strong safety net.

According to [16], they proposed a deep learning model designed for network traffic monitoring of numerous Iot solutions, in which the model analyzes the data and sends an assessment back to that of the network connection, which carries security protocols if necessary. During the research, they tested their method on two independent sets of data. The findings show that the model is quite effective at detecting possible threads, with an accuracy of over 99 percent even when the number of assessed networking features is limited. In this proposed model, the built RNN-LSTM classification model with the NAdam optimization method has been used to increase network security for Android.

The study [13] focuses on network anomalies, and it includes an ML-based anomaly detector as well as efficient techniques for making detection tools more resilient. The server computers, where sensitive data is housed, are a likely target for a rogue user. Finally, the attacker could tamper with critical measures that the operator uses to manage or monitor infrastructure units. By developing an optimization technique aided by stacking generalization techniques to evaluate the outliers, the focus of this paper [13] is to enhance the identification success of intrusion detection systems and networks operating in a drastic environment. Anomaly detection algorithms differ, resulting in various outcomes on the same data. As a result, providing a viable method for a specific domain is a difficult undertaking. To improve model performance, the stacked generalization strategy is used. Because of the requirement for heterogeneity, international quality assurance of feature designations and descriptions, as well as attack varieties, is required. In their research work [18], they show

the significance of dataset heterogeneity for useful IoT intrusion detection as well as how it enhances the improvement rate of detection using machine learning. In a multi-experiment concept, they demonstrate that a wide collection of statistical methods and studied elements are essential for IoT network intrusion datasets to be marketable. They classified this as a gap in current academic research on intelligent IoT network intrusion detection, and they believe that the lack of standardization will stymie industry acceptance of integrated smart IoT intrusion detection. They also emphasize the importance of standardizing feature descriptions and vulnerability classifications for the actual implementation of IoT datasets in operational situations. They provide a descriptive numerical detail of the ToN IoT dataset, a content research, and a classification to other relevant specialized IoT datasets by evaluating the ToN IoT dataset.

This study [24] focuses on a hybrid deep learning model for creating a unique security architecture and attack detection method for the efficient detection of malicious devices. The Convolution Neural Network (CNN) technique is used to extract the high-level feature representation of data before classifying it with the Long Short Term Memory (LSTM) Model. Existing research on utilizing DL to detect assaults only considered a small number of attacks. Also, there is a limitation of properly trained data models. This model was tested using a standard dataset from the Stratosphere lab, released in 2020. The dataset was created by using twenty infected Raspberry Pis along with three harmless IoT devices. The observed study's detection accuracy is 96 percent.

This study [19] presents a deep learning-based architecture for threat detection in an IoT environment that is SDN-enabled. For effective threat identification, cutting-edge Cuda-Deep neural networks, Closed Recurrent Unit (CuDNNGRU), and Cuda-Bidirectional Long Short Term Memory (CuBLSTM) classifiers are employed. To demonstrate the fairness of the results, a 10-time cross-validation is used. Our hybrid model is trained using the most recent publicly accessible CICIDS2018 dataset. The suggested technique achieves 99.87 percent accuracy and 99.96 percent recovery. Furthermore, the proposed hybrid model is compared to CudaGated Recurrent Unit, Long Short Term Memory (CuGRULSTM), CudaDeep Neural Network, Long Short Term Memory (Cu DNNLSTM), and existing reference classifiers. In terms of accuracy, F1 score, accuracy, speed efficiency, and the other assessment criteria, the suggested technique provides outstanding results.

This article [14] focuses on the threats, security needs, issues, and attack vectors that are important to IoT networks. The goal of this article is to offer an overview of SDN and also a comprehensive evaluation of centralized and decentralized SDN-based IoT deployment techniques. In order to provide a full picture of SDSec(Software Defined Security) technologies, we chose to explore Software-Defined Networking(SDN)-based security solutions for IoT. Furthermore, reviewing this literature, significant problems that are major obstacles to integrating all IoT collaborators on a unified system are highlighted, as well as a few outcomes focusing on a security solution which is network-based, for the IoT vision. This study [18] begins logically by outlining a basic IoT architecture, then moves on to describe IoT network protocols and the security problems associated with this at different levels of IoT networks. This study looked at market gaps and concentrated on security solutions based on network, for IoT products. SDN's numerous characteristics, including expandability, programmability, global transparency, and management, can enable it

to overcome traditional network restrictions. IoT deployment models based on SDN are discussed. It also highlighted available commercial tools and offered IoT models which are SDSec (Software Defined Security) based. To combat hyperglycemia, a gap analysis was conducted.

This research [22] uses Explainable Artificial Intelligence to enhance the accuracy and human understanding of the proposed model by describing the detection model’s technique for predicting assaults and expressing the variables or attributes that impact the corresponding prediction by implementing the Xgboost classifier on a dataset called IoT Intrusion Dataset, in which binary and multi-class classification, both are supported to build an IoT attack detection model and utilize XAI tools such as Shap, Lime, and ELI5 to authenticate the performance of the model by analyzing the model’s characteristics by depicting each feature’s contribution and behavior for each prediction in order to have a detailed knowledge of the effectiveness of the model in identifying IoT attacks.

The study [25] assesses the detection accuracy of two feature sets (NetFlow and CICFlowMeter) through three significant datasets named CSE-CIC-IDS2018, BoT-IoT, and ToN-IoT, finding that the NetFlow feature set outperforms other feature sets in improving the accuracy of machine learning models in detecting various network assaults. SHapley Additive exPlanations (SHAP), an explainable AI technique, has also been used to explain and comprehend ML model categorization decisions increasing complexity of learning models. The effect of each feature on the final ML prediction has been determined using the Shapley values of two common feature sets across various datasets. There are various factors that contribute to the large gap between research and production, including the restricted capacity to evaluate ML models comprehensively and a lack of knowledge of internal ML processes. This paper bridges the gap by exploring the generalization ability of a similar feature set to multiple network settings as well as attack scenarios.

2.2 Background

2.2.1 Decision Tree

A tree-like model in which each terminal node reflects a high-level characteristic. The leaves are the label classes, whereas the branches are the outputs. It mostly employs supervised learning for classification and regression, with the goal of mapping characteristics and values to the desired conclusion. It is a tree-like structure with root, internal nodes, leaf nodes, and edges. It primarily consists of two types of nodes: decision nodes and leaf nodes. It continues based on the decision made on the decision node. Any decision has an effect on the leaf nodes. Some fundamental terms for this method include ‘entropy’, ‘Gini Index’, ‘information gain’, and so on. Figure 2.1 represents the structure of decision tree.

In information theory, entropy is a metric that describes the impurity of a set of samples. The entropy S related to this n -wise categorization is defined as if the target characteristic takes on n alternative values [4].

$$E(S) = \sum_{k=1}^n -P_k \log(P_k) \quad (2.1)$$

Here E is the entropy and P_k is the probability of S belonging to a class k . The formula of Gini Index is-

$$G = 1 - \sum_{k=1}^n P_k^2 \quad (2.2)$$

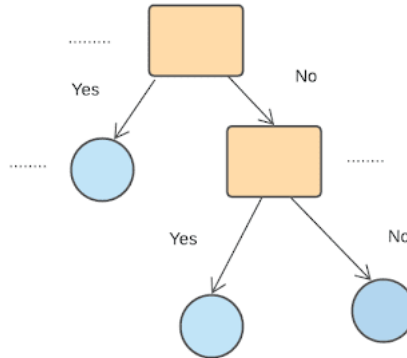


Figure 2.1: Decision Tree

2.2.2 Random Forest (RF)

Random forest is a predictive data mining problem and data augmentation that uses decision trees in a very randomized way. Random forest is also a type of classifier that uses a set of data [8]. Every branch is developed in conjunction with a binomial distribution, and they are all distinct and identically dispersed. Each tree in the ensemble submits a vote about the most common type of vector input. Random forest variety can indeed be produced through sampling an attribute, from a raw data, or simply changing certain decision tree parameters dynamically. The variety of factors to choose in each node, which seems consistent across all branches, and the forest's total number of trees are two random forest characteristics that may be tweaked. The Random Forest classifier is made up of a lot of decision trees that work together as a group. Each decision tree in the Random Forest classifier selects a feature at random. Allowing decision trees to select data with replacement and data characteristics at random reduces correlation amongst decision trees, resulting in higher classification performance. Individual decision trees are used to get the mean of the forecasts or to calculate the majority vote of the classes to arrive at the final classification. Implementing this model for our topic, the procedures are as follows [11]-

1. From the random M feature, pick the m feature. No more than M with the integer m .
2. On the basis of the separation measure, calculate the optimal split point for the k tree and divide the current node into the child node and the amount of M features of this node will be reduced.
3. If the maximum tree depth l is not reached or the separation matrix does not hit its limit, steps 1 and 2 should be repeated.
4. For each tree in the forest, repeat steps 1 through 3.
5. Cast our vote on the forest's production.

2.2.3 Adaboost

One of the most effective boosting algorithms on the market is AdaBoost [3]. It has a lot of practical experiences along with a solid theoretical base. AdaBoost's training and generalization errors are examined in order to determine whether AdaBoost can successfully increase the efficiency of a poor learning system. Multiclass AdaBoost, on the other hand, is presented as a binary-class AdaBoost extension. There's also a discussion of the AdaBoost approach and its uses. There is a discussion of some intriguing possibilities that should be pursued further. Discovering a more precise weak learning condition in a multiclass issue and deducing a tighter generalization error constraint are two of the directions in Boosting theory. Stopping conditions, boosting anti-noise capability, and increasing accuracy by increasing the diversity of the fundamental classifiers are all good challenges for AdaBoost to look at further. AdaBoost is a common approach for building a powerful classifier using a linear combination of member classifiers. During the training process, the member classifiers are chosen to reduce mistakes in each iteration step. AdaBoost provides a straightforward and practical method for generating ensemble classifiers. The ensemble's performance is determined by the diversity of the member classifiers as well as the performance of each individual classifier. The existing AdaBoost algorithms, on the other hand, are designed to solve error minimization problems. To calculate the sample weights:

$$w_i = \frac{1}{N} \quad (2.3)$$

Where $i = 1, 2, \dots, n$ and N is total data points.

Performance of the stump:

$$(\alpha) = \frac{1}{2} \log_e \frac{1 - TotalError}{TotalError} \quad (2.4)$$

Update Weights: New Sample Weight = Old Sample Weight * $e^{\pm\alpha}$

Here α is Performance of the stump Our research contributes to the method of creating diverse ensemble classifiers that are optimized.

2.2.4 XGBoost

[6] eXtreme Gradient Boosting is abbreviated as XGboost. Tree boosting is a popular and successful machine learning technique. Data scientists use XGBoost, a scalable end-to-end tree boosting algorithm, to produce cutting-edge results on a number of machine learning tasks. To minimize a normalized optimum solution, or the regression tree functions, XGBoost integrates a symmetric loss function given the disparity between the anticipated and targeted outcomes with a regularization term for computational cost. It's a convenient and effective open-source version of the gradient boosted trees technique. Gradient boosting is a supervised machine learning system that involves the estimates of a handful of discrete, weaker models to make predictions an input variables accurately. XGBoost scales to billions of examples while using a fraction of the resources used by existing systems. [5] Friedman's (2001) gradient boosting paradigm is applied in an optimized and efficient manner (Friedman et al., 2000). It includes a tree learning approach as well as a fast linear model solver. It provides regression, classification, and ranking, to name some few optimization methods. It's designed to be expandable, allowing users to quickly

specify their own goals. After installing libraries, xgboost, margrittr, Matrix, there are 3 easy steps to implement this model:

Step 1: Using the `xgb.DMatrix()` method, construct a Matrix for the train and test datasets.

Step 2: Configure the parameters for `params` and the `watchlist`.

Step 3: Create a model with the `xgb.train()` method.

2.2.5 Multilayer Perceptron

[4] MLP(multi layer perceptron) is a class of “feedforward artificial neural network”. An input layer takes the information, an output layer gives a judgement or projection about the information, and an unbounded number of hidden layers in between act as the MLP’s connected to the digital engine, with many neurons stacked together. In MLP, there is an input layer that receives signals, for making decision prediction or decision about the input there is an output layer. Moreover, to serve as MLP’s true computational engine, an arbitrary number of hidden layers are there in between the input layer and output layer, with many neurons stacked together. While neurons in a Perceptron must have a threshold-enforcing activation function, such as ReLU or sigmoid, neurons in a Multilayer Perceptron can have any activation function they want. It’s a learning algorithm that’s supervised. It trains a function $f(\cdot) : \mathbb{R}^m \rightarrow \mathbb{R}^o$ by training on a dataset, where ‘m’ is the number of input dimensions and ‘o’ is the number of output dimensions. Back-propagation with feed-forward, the most frequent neural classifier in pattern categorization is the multilayer perceptron (MLP). MLP is the accepted practice for any back propagation supervised pattern identification procedure.

The hyperplane equation:

$$H : w^T(x) + b = 0 \tag{2.5}$$

Here, b is the hyperplane equation’s intercept and bias term.

2.2.6 ANN

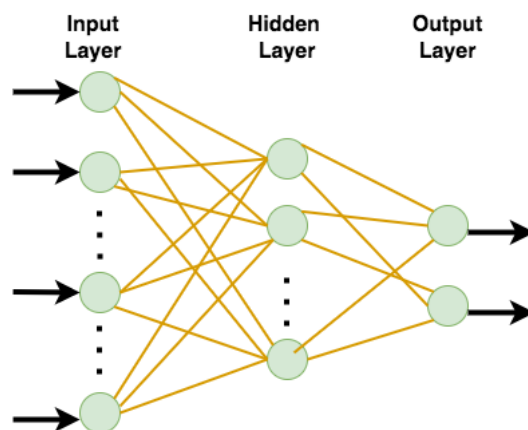


Figure 2.2: ANN

ANN, a machine learning algorithm works on the notion of a human neuron. They are widely used for learning capacitance and adaptability to different settings. ANN

requires a large multidimensional dataset to reduce the chance of extrapolation and can work on nonlinear and non-physical data. Neurons in our brain receive input as impulses. Two measures that reflect neuron activity are the average peak generation rate in repeated runs and the peak rate generated throughout time. Adaptive synaptic weights connect the neurons of the previous layer. Knowledge is frequently stored in the form of a series of weighted connections. From the input nodes to the hidden nodes, and subsequently from those nodes to the output nodes, information flows. The output unit with input connection weights that are almost identical to the input pattern is the winner. The issue inputs and outputs will have the same number of neurons as the input and output layers. The learning process is split into two parts: the first is identifying the hidden layer neuron, and the second is learning how to use it [27].

2.2.7 LIME

Lime (LOCAL INTERPRETABLE MODEL-AGNOSTIC EXPLANATIONS) is being used to try to understand the advanced IoT detection and prevention machine learning model since it can explain all sorts of models that have a prediction probability. Lime, as an ideal version integrator, offers both Model Agnostic and Local Model Interpretability, where Model Agnostic means it can be applied to any machine learning model and Local Model Interpretability means it may interpret the prediction of a single instance or present detection. In our IoT attack detection model, some strategies are used to illustrate the explanation of individual predictions when using the Lime Algorithm. To design a local linear model, bogus data is first created surrounding the observation (known as perturbed data). By generating predictions on perturbed data, the prediction function is then utilized to train the local linear model, indicated as (g) in the equation 2.6. The characteristics are then dispersed by changing continuous variables to discrete variables in order to make the format more human-friendly. The euclidean distance is then calculated by correlating every data point in the perturbed data to the original data point, which estimates the distance between the data point and the original observation. The euclidean distance is expressed in equation 2.6 by $L(f, p, \pi_x)$ to show whether the distance is less or bigger with 0 and 1 because it is converted to the value between 0 and 1, showing either it is closest to or exactly the same as the observation. Lime generates a local linear model utilizing all input data and the weights from the classification model to interpret the local behavior through observation. Lime is used to interpret the data into a human-understandable manner using three label features: Label, Category, and Sub-category. To make the model more comprehensible to the user, Lime explains it as follows, as taken from the paper [26],

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} L(f, g, \pi_x) + \Omega(g) \quad (2.6)$$

We have applied the LIME approach on the examples identified using learning models. We ran various tests using supervised learning approaches such as Random Forest, Decision Tree, and XGBoost with varied sizes of selected features in order to examine the local explanations given for the outputs of these models.

2.2.8 SHAP

The SHAP (short for SHapely Additive explanation) is a framework of XAI. In SHAP, the variability of the prediction is split among the obtainable variables, which allows each explanatory variable's contribution to each point of prediction to be evaluated, whatever the prime model is. SHAP outputs Shapely values, which represent model predictions as linear combinations of binary variables that specify if each covariate is integrated in the model or not. Likewise, the SHAP algorithm makes an approximation of each prediction $f(x)$ with $g(x')$, a linear function of the binary variables $z' \in \{0,1\}^M$ and $\phi_i \in \mathbb{R}$, defined as:

$$g(z') = \phi_0 + \sum_{i=1}^M \phi_i z'_i, \quad (2.7)$$

Here, M determines how many explanatory variables are present.

The desirable properties of this additive feature method are- accuracy, missingness and consistency. In this paper [21], these properties are obtained in SHAP attributing each variable x_i with an effect, ϕ_i (Shaply value), defined as:

$$\phi_i(f, x) = \sum_{z' \subseteq x'} \frac{|z'|!(M - |z'| - 1)!}{M!} [f_x(z') - f_x(z'|i)] \quad (2.8)$$

Here, f is the models, x are the variable and x' are the selected variables. The term

$$[f_x(z') - f_x(z'|i)]$$

indicates the divergence of Shaply value from their mean for each single prediction: the contribution of their i th variable.

2.2.9 ELI5

ELI5 stands for the phrase, 'Explain Like I'm 5'. It is used to verify and explain machine learning classifier predictions.. The main use of ELI5 is to debug algorithms. It supports the machine learning frameworks and packages are listed below:

scikit-learn: Users may now print decision trees as SVG or text, see feature importances, and explain decision tree and tree-based ensemble forecasts using ELI5.

keras: describe divination of image classifiers through Grad-CAM visualizations.

XGBoost: Explains divination of XGBClassifier, XGBRegressor and xgboost.Booster and display feature importances and

LightGBM: Explains predictions of LGBMClassifier and LGBMRegressor and display feature importance.

CatBoost: CatBoostClassifier and CatBoostRegressor feature importances are displayed.

Lighting: Weights and predictions of lightning classifiers and regressors are explained.

sklearn-crfsuite: We can examine the weights of sklearn crfsuite.CRF models with ELI5. ELI5 also executes many algorithms to inspect black-box models. There are primarily two approaches to examine a classification or regression model:

(1) Examine model parameters to see how the model works on a larger scale.

(2) Examine a model’s individual prognosis and try to understand why it makes the decisions it does.

ELI5 provides the `eli5.show_weights()` method for (1) and the `eli5.show_prediction()` function for (2).

To start with, we need a dataset and a problem. In our paper, we will be using the ToN IoT dataset.

2.2.10 SKLearn

Scikit-learn is a powerful and adaptable Python machine learning toolbox. It has a lot of advanced machine learning and statistical analysis features. The Sklearn library is more concerned with data modeling. Sklearn provides some popular groups of Supervised Learning algorithms and classifiers, which we have implemented in our work. For Decision Trees supervised learning method we have imported `DecisionTreeClassifier` from the SKlearn library. For using another machine learning supervised learning technique, Random Forest, we have imported `RandomForestClassifier`. AdaBoost classifier is also imported from the Sklearn library. To convert the data into machine readable form, we have encoded the dataset by importing `LabelEncoder` from Sklearn. Sklearn’s ‘train test split’ was used to divide the dataset into subsets that limit the possibility of bias in the evaluation and validation processes. Sklearn supports importing the Standard Scaler, which standardizes a feature by eliminating the mean and scaling it to unit variance. Also, we have used Robust Scaler and `MinMaxScaler` from the Sklearn library for scaling the dataset. Sklearn library has helped for all these preprocessing in different classifiers in the dataset. For summarizing the performance of the classifiers, we have used the Sklearn Confusion Matrix technique. The Classification Report from the Sklearn library is used to measure the quality of prediction from the classification algorithms. The Scikit-Learn Metrics module has been imported to calculate accuracy of the model.

Chapter 3

Methodology

3.1 Methodology

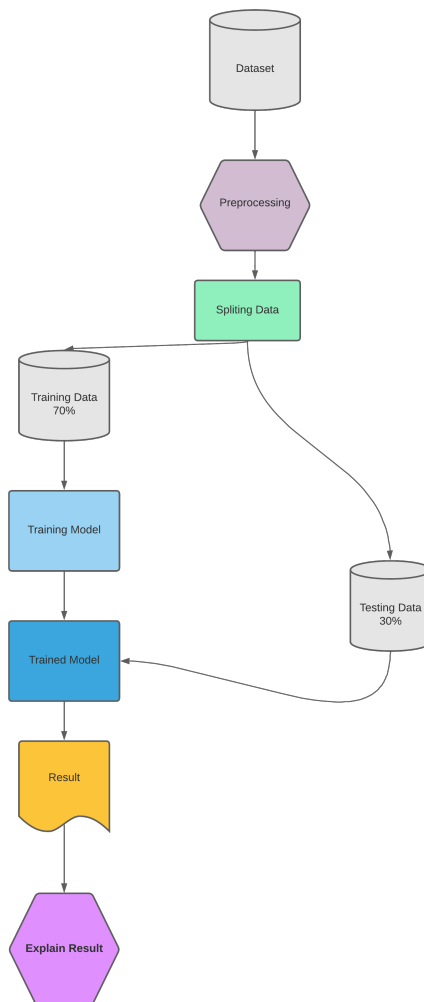


Figure 3.1: The flow chart of the proposed model

Figure 3.1 is the workflow of our proposed model.

Three essential phases required for attack detection and classification are:

1. Data Preprocessing : This step focuses on structuring the input data such that network attack detection can be handled quickly.
2. Training Data: This stage is concerned with the training of input using algorithms of ML.
3. Testing Data: This step is focused on testing data utilizing the previously constructed DT.
4. Result: This step shows the prediction result of identification and classification from the trained model.
5. Explain Result: This stage explains the result using XAI.

The input data is forwarded via the preprocessing step, where ML techniques are applied to distinguish between normal and attack data.

After that, the preprocessed data is separated into two categories, one for training and creating classifiers, and another for evaluating the classifiers' and models' efficiency in differentiating between normal and attack data.

3.1.1 Overview of the Dataset

Over a period of time, multiple datasets have been generated to detect IOT attacks. The TON-IoT dataset comprises 161043 threat and 300,000 normal observations and includes threats such as DoS, DDoS, backdoors, injection, scanning, password, Man-in-the-Middle(MitM), ransomware, assaults, and XSS. Because of its different attack scenarios, telemetry data, label and heterogeneity of IoT data sources, and IoT scenarios on a sample test case, the ToN IOT Dataset was used to test our method.

Telemetry data from IIoT/IoT devices, Operating System logs and network traffic from an IoT network, were all acquired out of a genuine medium-scale network approximation and it was built at UNSW Canberra's Cyber Range and IoT Labs. The proposed Telemetry data dataset for IIoT/IoT services, as well as its properties, are the subject of this study. The ToN-IoT repository contains ToN-IoT datasets. Furthermore, a label feature was used to label the proposed datasets (which shows whether or not a dataset has been labeled).

There are seven (7) IoT and IIoT sensors in total (e.g. weather, temperature, and Modbus sensors), as well as two cellphones and a smart TV, were used to collect telemetry data, which was collected in log and CSV files. Datasets that have been processed are referred to as 'processed datasets' and 'Train Test datasets.' The 'Processed datasets' folder provides a CSV version of the processed and filtered datasets, together with their regular features and labels. To analyze the accurateness of network security apps and machine learning technologies, sampling of datasets are utilized as train-test datasets in CSV format in the 'Train Test datasets' folder. Every IoT device has its own "Train Test IoT device name.csv" CSV file. Each IIoT device has its own Train-Test dataset: refrigerator, GPS tracker, motion light, garage door, Modbus, thermostat, and weather. Figure 3.2 is the ToN IoT Dataset for IoT Fridge sample.

3.1.2 Data preprocessing

The very first approach to improving the training process for models is to analyze the data. It is a method of processing raw data for usage by machine learning mod-

04-Apr-19	07:50:01	2	low	0	normal
04-Apr-19	07:50:01	4.8	low	0	normal
04-Apr-19	07:50:01	6.7	high	0	normal
04-Apr-19	07:50:01	8.55	high	0	normal
04-Apr-19	07:50:01	13.4	high	0	normal
28-Apr-19	06:59:38	3.2	low	1	backdoor
28-Apr-19	06:59:43	4	low	1	backdoor
28-Apr-19	06:59:48	4.65	low	1	backdoor
28-Apr-19	06:59:53	2	low	1	backdoor
28-Apr-19	06:59:58	4.95	low	1	backdoor
28-Apr-19	07:00:03	12.65	high	1	backdoor
28-Apr-19	07:00:08	1.9	low	1	backdoor
28-Apr-19	07:00:13	5.65	low	0	normal

Figure 3.2: ToN IoT Dataset(IoT_Fridge) sample

els. There are several techniques and steps for data preprocessing. To lower the data storage and eliminate redundancy, duplicate samples, missing can be deleted. Pre-processing was applied to handle missing values. In ToN IoT dataset, most of the data are preprocessed. But there were some null values in the Thermostat and the Garage Door dataset. The null values of the “Date” and “Time” features in the IoT Thermostat Dataset and “date”, “time”, “door-state” and “sphone_signal” features of the Garage Door dataset have been dropped. To split the dataset, 70% of it has been utilized for training and 30% for testing. For binary classification, the “label” feature is considered the target or dependent variable, whereas all other characteristics except ‘type’ are considered independent variables. But in multiclass classification, the “type” feature is considered as target or dependent and other features except “label” are considered as independent variables.

3.1.3 Feature Scaling

Feature scaling is an important step in ML and DL models. Scaling changes all of the values of the features to the same scale. Some models are quite sensitive to diverse types of data. It also allows models to do computations more quickly. Scaling allows us to provide the equal weight to each feature value, ensuring that no feature value is prioritized because it has a higher value. For feature scaling in our models, we used Min Max Normalization and Robust Scaling. Min Max Normalization is a method that converts numbers in the [0.0,1.0] range.

$$X_{new} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (3.1)$$

Robust Scaling is a very efficient technique if we have outliers in our data. It scales the data according to the interquartile range(IQR) .

$$X_{new} = \frac{X - X_{median}}{IQR} \quad (3.2)$$

3.1.4 Feature selection

The characteristics for these scenarios were retrieved depending on the modeled IoT/IIoT services' embedded sensors. The following is a breakdown of the extracted IoT/IIoT scenarios and features.

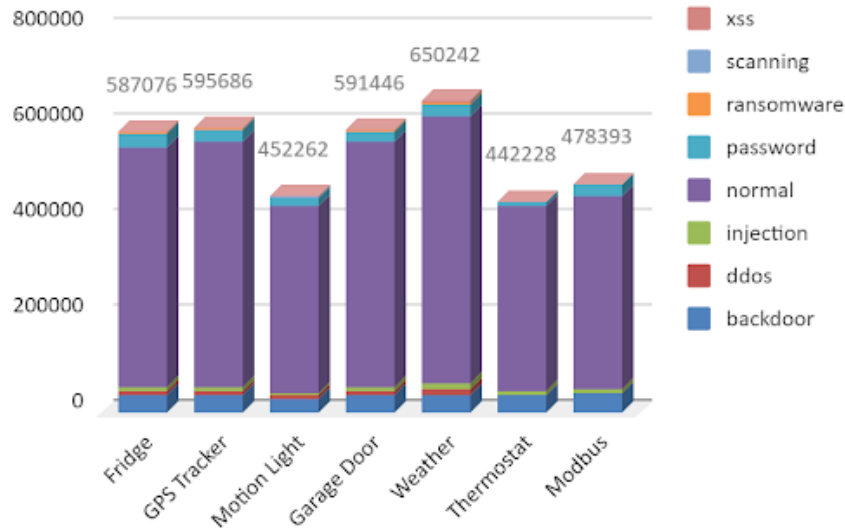


Figure 3.3: Number of Attacks on Different IoT Devices

The IoT fridge monitors the thermal condition and changes it as needed to keep it within a specified threshold. Table 3.1 displays and explains the characteristics of the IoT fridge dataset. A probabilistic input allows a remotely actuated garage door to open or close. Table 3.2 shows the characteristics of garage door sensors. The GPS characteristics in Table 3.3 explain how GPS records the position coordinates of an item remotely, such as latitude and longitude. The Modbus service replicates the capabilities of Modbus devices used in several industrial uses by connecting them over serial lines. A Modbus device that sends a request is referred to as a Master Modbus device, whereas a Modbus device that receives a request is referred to as a Slave Modbus device. Table 3.4 provides further information about it. The clever motion sensor activates or deactivates the light depending on a pseudo-random signal, as described in Table 3.5. A smart thermostat may regulate a heating/cooling system to control the temperature of a physical system. As shown in Table 3.6, the current temperature and thermostat status are the most important elements of a smart thermostat sensor. Humidity, air pressure, and temperature are measured by a weather monitoring system. In Table 3.7, further specifics are given.

3.1.5 Attacks

Denial of Service (DoS)

It's a very well-flooding attack in which an attacker makes a series of malicious attempts to prevent a genuine user from accessing resources. The IIoT dataset includes distributed denial-of-service (DDoS) and denial-of-service (DoS) assaults. DDoS attacks are typically carried out by hots or botnets, which are massive groups

IoT Fridge activity			
ID	Feature	Type	Description
1	date	Date	IoT telemetry data recording date
2	time	Time	IoT telemetry data recording time
3	fridge_temperature	Number	Thermal reading from a network-connected fridge sensor
4	temp_condition	String	Temperature circumstances of a network-connected fridge sensor, whether the temperature is excessive or lower based on a specified threshold
5	label	Number	0 to represent normal and a 1 to represent attack
6	type	String	Categories different types of attacks and normal data

Table 3.1: Feature Type and Description of IoT Fridge

IoT Garage_Door activity			
ID	Feature	Type	Description
1	date	Date	IoT telemetry data recording date
2	time	Time	IoT telemetry data recording time
3	door_state	Boolean	When a door sensor is connected to the network, it determines whether the door is open or not
4	sphone_signal	Boolean	When a phone receives a door signal, the signal is either true or false.
5	label	Number	0 to represent normal and a 1 to represent attack
6	type	String	Categories different types of attacks and normal data

Table 3.2: Feature Type and Description of IoT Garage_Door

of infected machines. This attack includes flooding IoT devices with a large amount of nodes in order to deplete their resources (e.g., CPU and memory).

Ransomware

This is a sophisticated sort of malware that encrypts a system or service to deny a legitimate user access and then attempts to sell the encrypted message, which allows the user to re-enter the system. An IoT ransomware is identical to a traditional ransomware, except it prevents IoT devices from being accessed. IIoT devices and apps are vulnerable to IoT ransomware because they frequently perform mission-critical functions to which denying access or locking down these applications might result in catastrophic effects, such as financial losses to enterprises [12].

IoT GPS_Tracker activity			
ID	Feature	Type	Description
1	date	Date	IoT telemetry data recording date
2	time	Time	IoT telemetry data recording time
3	latitude	Number	The network-connected GPS tracker sensor's latitude value.
4	longitude	Number	longitude value of network-connected GPS tracker sensor.
5	label	Number	0 to represent normal and a 1 to represent attack
6	type	String	Categories different types of attacks and normal data

Table 3.3: Feature Type and Description of IoT GPS_Tracker

IoT Modbus activity			
ID	Feature	Type	Description
1	date	Date	IoT telemetry data recording date
2	time	Time	IoT telemetry data recording time
3	FC1_Read_Input_Register	Number	An input register function which helps in identification
4	FC2_Read_Discrete_Value	Number	A function identifier for accessing an input discrete value.
5	FC3_Read_Holding_Register	Number	Reading a holding register with Modbus function code
6	FC4_Read_Coil	Number	Reading a coil using Modbus function code.
7	label	Number	0 to represent normal and a 1 to represent attack.
8	type	String	categories different types of attacks and normal data.

Table 3.4: Feature Type and Description of IoT Modbus

IoT Motion_Light activity			
ID	Feature	Type	Description
1	date	Date	IoT telemetry data recording date
2	time	Time	IoT telemetry data recording time
3	motion_status	Number	This can be on or off.
4	light_status	Boolean	This can be on or off.
5	label	Number	0 to represent normal and a 1 to represent attack.
6	type	String	categories different types of attacks and normal data.

Table 3.5: Feature Type and Description of IoT Motion_Light

IoT Thermostat activity			
ID	Feature	Type	Description
1	date	Date	IoT telemetry data recording date
2	time	Time	IoT telemetry data recording time
3	current_temperature	Number	Thermal reading from a network-connected thermostat sensor.
4	thermostat_status	Boolean	This can be on or off.
5	label	Number	0 to represent normal and a 1 to represent attack
6	type	String	Categories different types of attacks and normal data

Table 3.6: Feature Type and Description of IoT Thermostat

IoT Weather activity			
ID	Feature	Type	Description
1	date	Date	IoT telemetry data recording date
2	time	Time	IoT telemetry data recording time
3	temperature	Number	A network-connected weather sensor's thermal readout
4	pressure	Number	Pressure measurements from a network-connected weather sensor
5	humidity	Number	Humidity readings from a network-connected weather sensor.
6	label	Number	0 to represent normal and a 1 to represent attack
7	type	String	Categories different types of attacks and normal data

Table 3.7: Feature Type and Description of IoT Weather

Backdoor

It's a passive assault in which a backdoor malware allows an attacker to get unauthorized remote access to infected IIoT devices. The attacker employs the backdoor to take control of affected IIoT devices and integrates them into botnets in order to launch DDoS attacks [30].

Injection Attack

This attack attempts to run malicious code or introduce harmful data into IIoT apps on a regular basis. Furthermore, the injection attack may disrupt regular operations by altering the telemetry data and controlling the instructions of the IIoT system. Two shell scripts were created to inject input data into web programs such as (DVWA and the vulnerable public PHP), as well as Security Shepherd VMs and IoT websites [12].

Cross Site Scripting (XSS)

In IIoT applications, it frequently tries to execute malicious commands on a Web server. In IIoT applications, a web server is attacked by the frequent attempt to execute the malicious command of the attacker. An attacker can employ XSS to inject arbitrary Web scripts from a remote location, as in malicious HTTP or JavaScript instructions. This attack may endanger data and verification mechanisms between a remote Web server and IIoT devices. [12].

Password Cracking Attack

When an attacker attempts to access the account of IIoT devices using password cracking strategies like brute force and dictionary assaults. As a result, the attacker may be able to bypass identity management and access IIoT devices [12].

Chapter 4

Implementation

4.1 Implementation

Performance Measures

Evaluation metrics are used to analyze and interpret a model's performance. This also indicates how well or poorly the models are performing. Furthermore, Evaluation metrics are essential for comparing different models. There are several metrics for measuring performance. Some of the measures we use to evaluate our models include accuracy, precision, recall, and F1-Score, as well as the Confusion Matrix.

Accuracy

Accuracy is the ratio of correct predictions out of the total number of predictions. It represents a model's overall efficacy as the proportion of all normal and attack data that are accurately categorized.

$$Accuracy = \frac{(TP + TN)}{TP + TN + FP + FN} \quad (4.1)$$

Precision

It is defined as the proportion of accurately predicted positive data to total anticipated positive data, including True Positive and False Positive values. It is a proportional quantity of accurately anticipated to total anticipated positive data, including True Positive and False Positive values. It represents a model's overall efficacy as out of all attack scenarios how many are accurate.

$$Precision = \frac{TP}{TP + FP} \quad (4.2)$$

Recall

Recall is defined as the ratio of correctly predicted positive data to total anticipated True Positive and False Negative values. The ratio of accurately predicted positive data to total expected True Positive and False Negative values is known as recall. If any models detects False Negative values, it affects the recall metrics

$$Recall = \frac{TP}{TP + FN} \quad (4.3)$$

F1-Score

It establishes a balance between precision and recall. It determines the harmonic mean between those two's

$$F - Score = 2 * \frac{Recall * Precision}{Recall + Precision} \quad (4.4)$$

Confusion Matrix

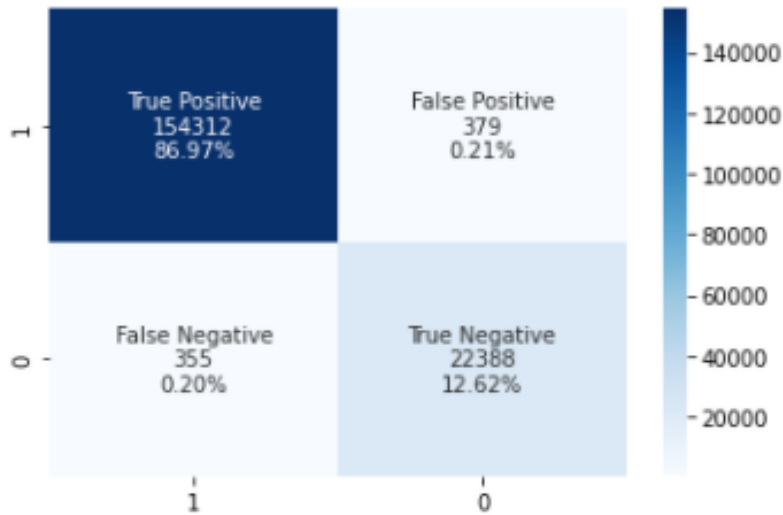


Figure 4.1: Confusion Matrix of Decision Tree Binary Classification of Fridge

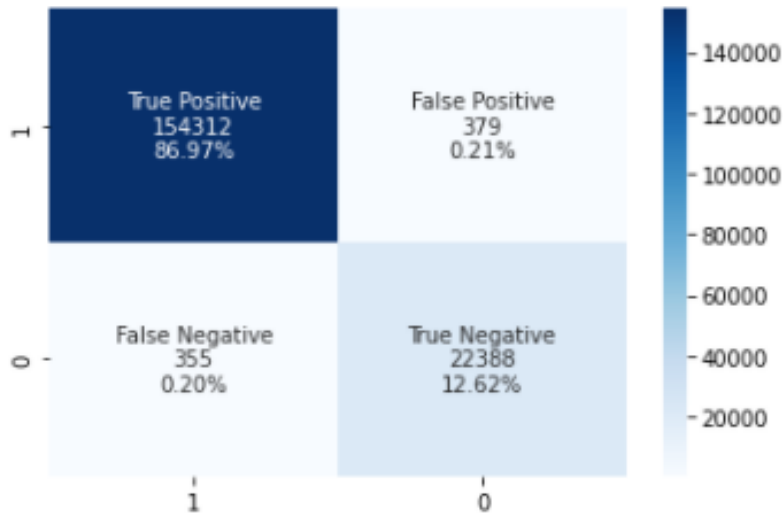


Figure 4.2: Confusion Matrix of Decision Tree Binary Classification of Garage Door

The Confusion Matrix represents the relationship between expected and actual value. It is a summary from which correct and incorrect predictions may be clearly understood. It is a N X N matrix where N is the number of classes in the target variable. For Binary classification it is a 2 X 2 matrix and for multiclass classification it depends on the “Type” label for our dataset. Here Figure 4.1, 4.2, 4.3, 4.4 are the confusion matrices after applying the Decision Tree and XGBoost algorithm on the IoT_Fridge dataset and IoT_Garage Door dataset.

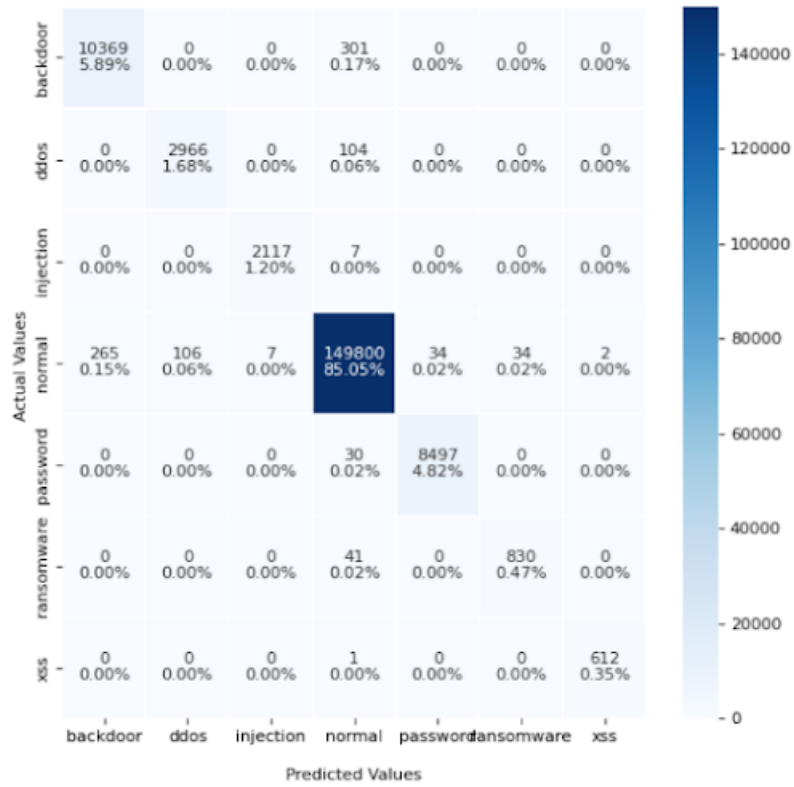


Figure 4.3: Confusion Matrix of Decision Tree Multiclass classification of Fridge

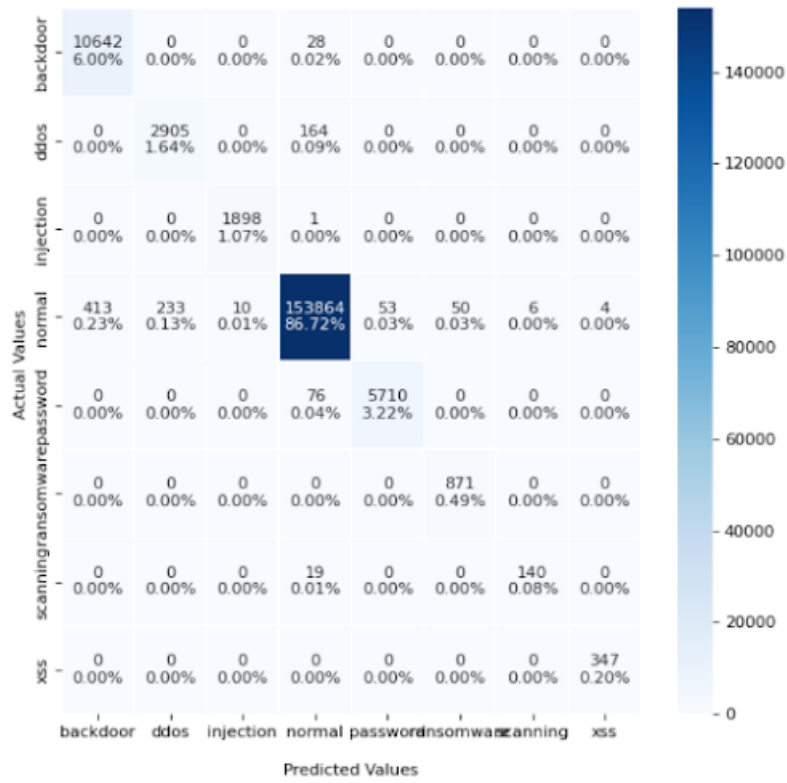


Figure 4.4: Confusion Matrix of XGBoost Multiclass classification of Garage Door

Chapter 5

Result

5.1 Result

Datasets	Binary Classification							Multiclass Classification					
	Models	Decision Tree	Random Forest	AdaBoost	XGBoost	ANN	MLP	Decision Tree	Random Forest	AdaBoost	XGBoost	ANN	MLP
Fridge	precisio	0.99	1.00	0.98	0.99	0.98	0.98	0.99	1.00	0.79	0.99	0.85	0.98
	Recall	0.99	1.00	0.98	0.99	0.98	0.98	0.99	1.00	0.86	0.99	0.71	0.98
	F1 Score	0.99	1.00	0.98	0.99	0.98	0.98	0.99	1.00	0.82	0.99	0.76	0.98
	Accuracy	0.9947	0.9957	0.9826	0.9912	0.9757	0.97899	0.9947	0.9956	0.854	0.994	0.7146	0.9809
GPS Tracker	precisio	1.00	1.00	0.98	0.99	0.99	0.98	0.99	1.00	0.74	0.99	0.88	0.96
	Recall	1.00	1.00	0.98	0.99	0.99	0.98	0.99	1.00	0.86	0.99	0.79	0.96
	F1 Score	1.00	1.00	0.98	0.99	0.99	0.98	0.99	1.00	0.8	0.99	0.81	0.96
	Accuracy	0.9951	0.9954	0.9795	0.9911	0.9895	0.97597	0.9944	0.9955	0.8628	0.9946	0.7907	0.9637
Garage Door	precisio	1.00	1.00	0.99	0.99	0.98	0.98	1.00	1.00	0.76	0.99	0.87	0.98
	Recall	1.00	1.00	0.99	0.99	0.98	0.98	1.00	1.00	0.87	0.99	0.78	0.98
	F1 Score	1.00	1.00	0.99	0.99	0.98	0.98	1.00	1.00	0.81	0.99	0.8	0.98
	Accuracy	0.9959	0.9957	0.9853	0.9941	0.9787	0.9837	0.996	0.9957	0.8699	0.994	0.7781	0.9822
ModBus	precisio	0.99	0.99	0.97	0.99	0.96	0.97	0.99	0.99	0.6	0.99	0.77	0.98
	Recall	0.99	0.99	0.97	0.99	0.96	0.97	0.99	0.99	0.78	0.99	0.17	0.98
	F1 Score	0.99	0.99	0.97	0.99	0.96	0.97	0.99	0.99	0.68	0.99	0.27	0.98
	Accuracy	0.9926	0.9926	0.9683	0.98695	0.9619	0.9703	0.9925	0.9926	0.777	0.9927	0.1678	0.9799
Motion Light	precisio	1.00	1.00	0.98	0.99	0.98	0.98	1.00	1.00	0.77	0.99	0.87	0.98
	Recall	1.00	1.00	0.98	0.99	0.98	0.98	1.00	1.00	0.87	0.99	0.75	0.98
	F1 Score	1.00	1.00	0.98	0.99	0.98	0.98	1.00	1.00	0.81	0.99	0.79	0.98
	Accuracy	0.9955	0.995	0.9775	0.9931	0.9796	0.9764	0.9954	0.995	0.866	0.9942	0.7531	0.9846
Thermostat	precisio	0.99	1.00	1.00	1.00	1.00	0.99	0.99	1.00	0.76	1.00	0.9	0.99
	Recall	0.99	1.00	1.00	1.00	1.00	0.99	0.99	1.00	0.87	1.00	0.87	0.99
	F1 Score	0.99	1.00	1.00	1.00	1.00	0.99	0.99	1.00	0.81	1.00	0.87	0.99
	Accuracy	0.9948	0.9967	0.9952	0.9954	0.9955	0.99091	0.9947	0.9967	0.8715	0.9961	0.8732	0.9926
Weather	precisio	0.99	1.00	0.98	0.99	0.97	0.97	1.00	1.00	0.74	0.99	0.86	0.98
	Recall	0.99	1.00	0.98	0.99	0.97	0.97	1.00	1.00	0.86	0.99	0.72	0.98
	F1 Score	0.99	1.00	0.98	0.99	0.97	0.97	1.00	1.00	0.8	0.99	0.77	0.98
	Accuracy	0.9947	0.9955	0.9771	0.9887	0.9655	0.9671	0.995	0.9955	0.8616	0.9938	0.7249	0.982

Table 5.1: A Summary of Evaluation Metrics of Imbalanced Dataset

Evaluation metrics are used to analyze and interpret a model's performance. This also indicates how well or poorly the models are performing. Furthermore, Evaluation metrics are essential for comparing different models. There are several metrics for measuring performance. Some of the measures we use to evaluate our models include accuracy, precision, recall, and F1-Score, as well as the Confusion Matrix. The Confusion Matrix represents the relationship between expected and actual value. It is a summary from which correct and incorrect predictions can be clearly understood. The performance of the imbalanced dataset is described in Table 5.1 and the balanced IoT Fridge dataset's performance is presented in Table 5.2. For the performance measurements, the weighted average is calculated for each model's result. In case of Imbalanced Dataset, Decision Tree and Random Forest have outperformed each other in terms of accuracy, precision, recall, and f1 score for binary and multi-class classification. The XGBoost classifier has achieved a score of more than 98%

Data Balancing Method	Binary Classification							Multiclass Classification					
	Models	Decision Tree	Random Forest	AdaBoost	XGBoost	ANN	MLP	Decision Tree	Random Forest	AdaBoost	XGBoost	ANN	MLP
Undersampling	precision	1.00	1.00	0.95	0.99	0.97	0.97	1.00	1.00	0.65	1.00	0.55	0.98
	Recall	1.00	1.00	0.94	0.99	0.97	0.97	1.00	1.00	0.74	1.00	0.40	0.98
	F1 Score	1.00	1.00	0.94	0.99	0.97	0.97	1.00	1.00	0.69	1.00	0.37	0.98
	Accuracy	0.9950	0.9965	0.9425	0.9945	0.9693	0.9724	0.9954	0.9965	0.7426	0.9951	0.4029	0.9794
Oversampling	precision	1.00	1.00	0.95	1.00	0.98	0.98	1.00	1.00	0.66	1.00	0.50	0.99
	Recall	1.00	1.00	0.94	1.00	0.98	0.98	1.00	1.00	0.74	1.00	0.36	0.98
	F1 Score	1.00	1.00	0.94	1.00	0.98	0.98	1.00	1.00	0.69	1.00	0.41	0.98
	Accuracy	0.9982	0.9979	0.9426	0.9950	0.9840	0.9803	0.9982	0.9979	0.7436	0.9950	0.3570	0.9846
SMOTE	precision	0.99	0.99	0.95	0.99	0.97	0.97	1.00	1.00	0.65	0.99	0.56	0.98
	Recall	0.99	0.99	0.94	0.99	0.97	0.97	1.00	1.00	0.74	0.99	0.42	0.98
	F1 Score	0.99	0.99	0.94	0.99	0.97	0.97	1.00	1.00	0.69	0.99	0.39	0.98
	Accuracy	0.9908	0.9911	0.9429	0.9856	0.9658	0.9735	0.9950	0.9962	0.7407	0.9947	0.4195	0.9780

Table 5.2: A Summary of Evaluation Metrics of Balanced IoT_Fridge Dataset

for binary classification and more than 99% for multiclass classification. MLP has performed well in both classifications, and its performance has improved once the data is scaled. Though Adaboost and ANN have achieved a score of more than 96% for binary classification across all datasets, their score for multiclass classification has decreased. Adaboost has obtained the lowest score of 77% in multiclass classification for the Modbus dataset, while it is above 85% for the other datasets. Furthermore, ANN has the lowest accuracy of 16.78% and the lowest f1 score of 27%. Overall, Decision Tree, Random Forest, XGBoost and MLP have performed really well for most of the datasets. On the contrary, the accuracy of Adaboost and ANN have decreased for multiclass classification.

Figure 3.3 shows that the dataset is imbalanced and the majority of the data is normal. As a result, approaches like undersampling, oversampling, and SMOTE (Synthetic Minority Oversampling Technique) are applied to balance the dataset and analyze the performance for both binary and multiclass classification. In case of the balanced dataset, Decision Tree and Random Forest have achieved greater than 99% accuracy for all data balancing approaches, with a higher f1 score than the imbalanced dataset. However, Adaboost’s performance in binary classification has dropped to nearly 94% when compared to the binary classification of the balanced dataset. XGBoost and MLP have performed well for all the methods. Adaboost has failed to classify four of the classes using undersampling, oversampling, and the SMOTE technique, with a weighted average f1 score of 0.69. ANN could not classify five of the classes and scored for accuracy the lowest of technique, 40%, 35% and 41% for undersampling, oversampling and SMOTE respectively.

5.1.1 XAI Result Analysis

LIME

Figure 5.3 and Figure 5.4 show the LIME explanations of the predictions for two input instances using Decision Tree and XGBoost classifier on the Modbus dataset and GPS Tracker dataset respectively. We can see from these two figures that each of these explanations is divided into three parts: prediction probabilities for all possible outputs, the bar charts for all outputs which represent the weights and

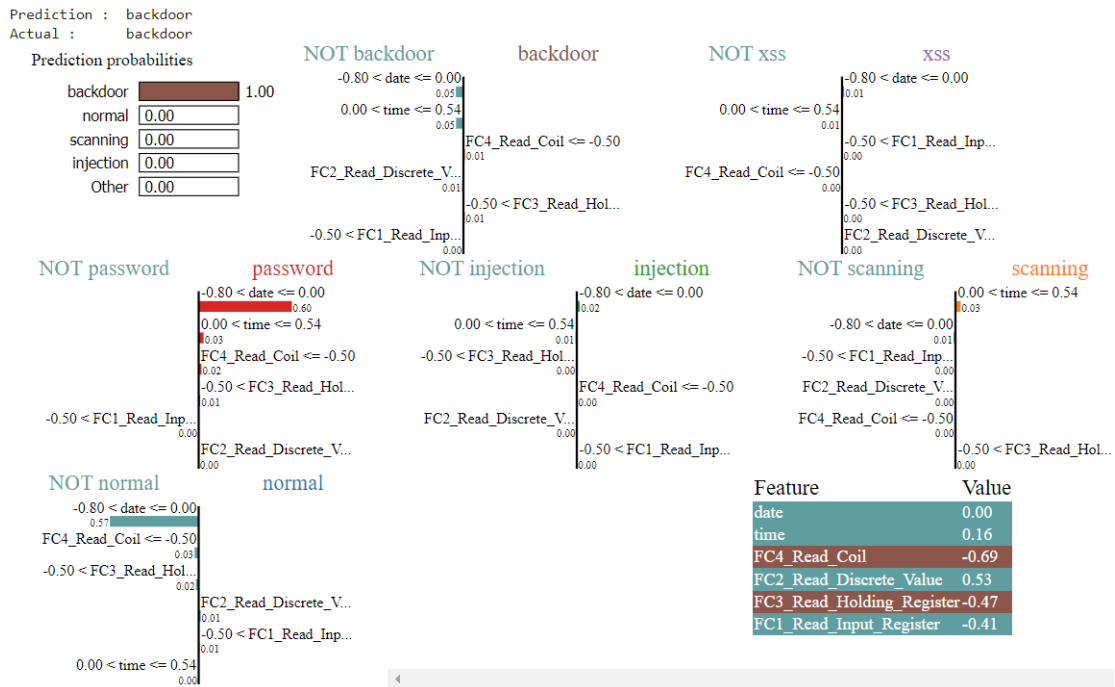


Figure 5.3: LIME explanation of Decision Tree on Modbus dataset

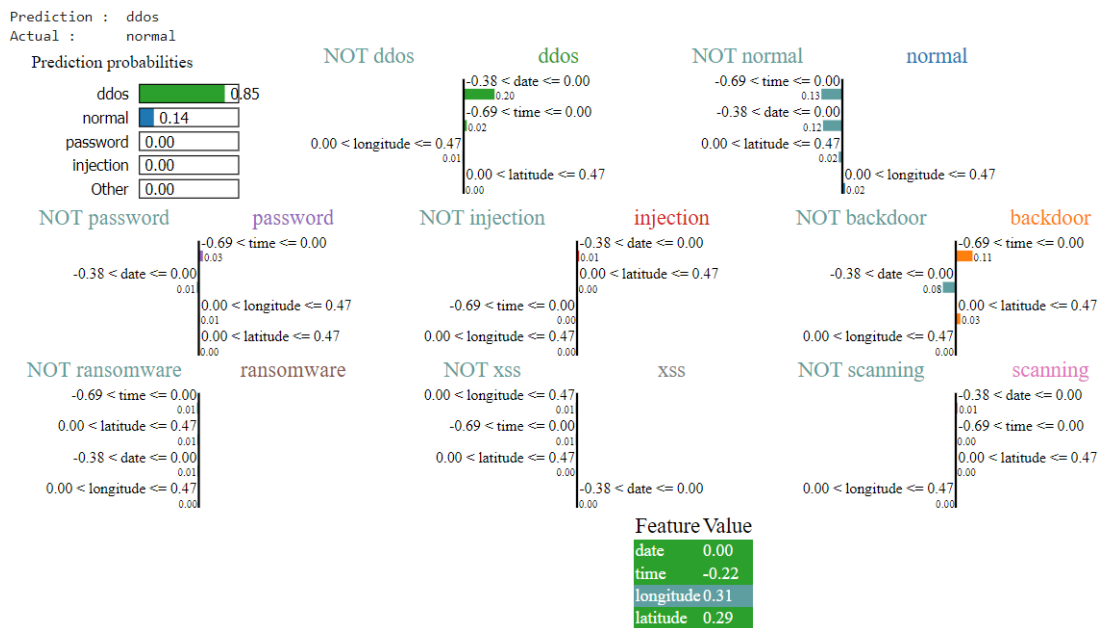


Figure 5.4: LIME explanation of XGBoost on GPS Tracker dataset

contribution of each feature in the prediction and a feature value table. According to Figure 5.3, the Decision Tree classifier has accurately predicted the output for a particular input data where the prediction probability of the “backdoor” is 1.00 and 0.00 for others. From the feature value table, we can observe that features with Maroon color support ‘backdoor’ as a prediction outcome whereas features with Blue support for the outcome of ‘not backdoor’. Therefore, the backdoor is predicted as an output for the features ‘FC3_Read_Holding_Register_n’ and ‘FC4_Read_Coiln’. These two features have a positive impact on the prediction since the feature value of ‘FC4_Read_Coiln’ is less than or equal to -0.50 and the feature value of ‘FC3_Read_Holding_Register_n’ is larger than -0.50. The ‘date’, ‘time’, ‘FC_Read_Input_Register_n’ and ‘FC2_Read_Discrete_Value_n’ have a negative impact on predicting the output as a ‘backdoor’.

From Figure 5.4, we can see that for an input instance the XGBoost classifier could not predict the output correctly. It has predicted the output as “ddos” but the actual output would be “normal”. The prediction probability is 0.85 for ‘ddos’, 0.14 for ‘normal’ and 0.00 for other possible outputs. From the feature value table, we can observe that ‘time’, ‘date’ and ‘latitude’ are the features that have a positive impact on the predicted output and for these features’ contribution this prediction is made. The feature ‘longitude’ has a negative impact on the prediction, whereas it has a positive impact for ‘normal’ output since it has a feature value of 0.31 which is in the range of $0 < \text{longitude} \leq 0.47$.

SHAP

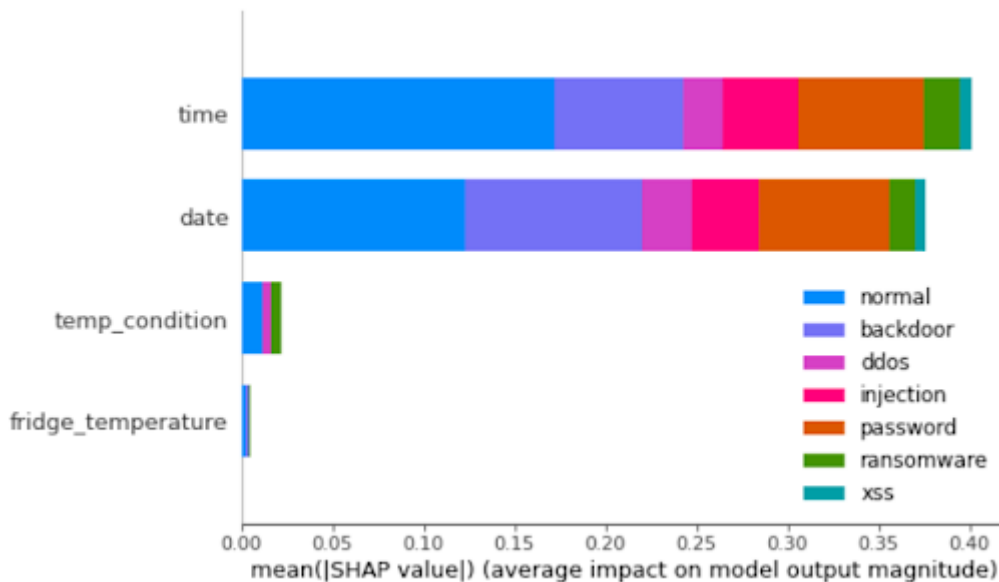


Figure 5.5: Feature importance calculated by SHAP value of Fridge dataset

Figure 5.5 shows the average impact of each feature on the prediction of the IoT Fridge dataset using the Decision Tree model. This summary plot gives a global explanation of this model. The names of the features are displayed on the Y-axis in descending order of contribution, from highest (time) to lowest (fridge_temperature). The X-axis represents the absolute means of SHAP values. In this figure,

various colors symbolize various classes. There are seven different classes in this figure. We can see from the figure that the ‘time’ feature contributes the most to all classes. Moreover, the ‘time’ and ‘date’ features have a significant impact on all classes. Whereas other classes have ignored this feature, ‘temp_condition’ has a minor influence on ‘normal’, ‘ddos’, and ‘ransomware’. The ‘fridge_temperature’ has the least influence and almost all the classes have completely ignored this feature.

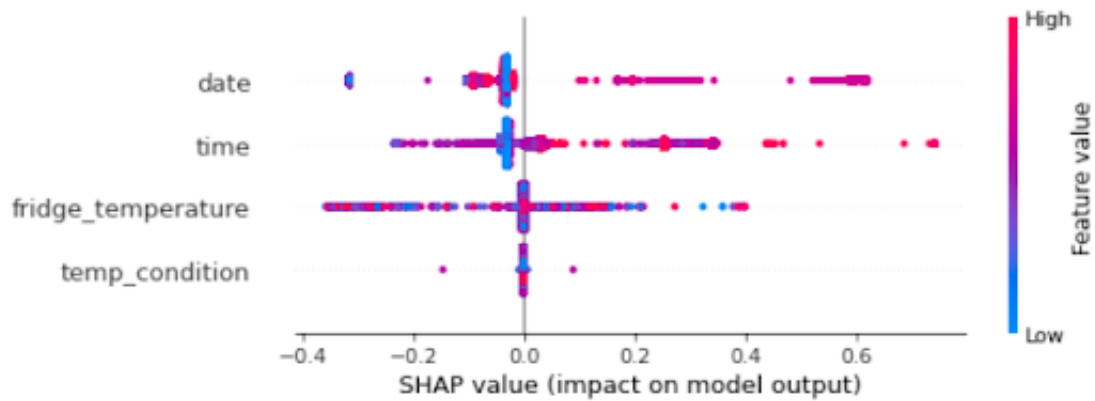


Figure 5.6: SHAP Summary Plot for Fridge dataset using Decision Tree

Figure 5.6 shows summary plot of SHAP for the Fridge dataset using Decision Tree. The dots in the plot are the data points of the IoT Fridge dataset. The color represents feature value, with red being high and blue representing low. On the Y-axis, the names of the features are given in descending order of contribution, from highest (date) to lowest (temp_condition). The X-axis shows SHAP values which shows whether the feature contribution has a positive impact or negative. As there are seven different classes in this dataset, there can be a total of seven possible outcomes. For each possible outcome, there is an array of SHAP values. So for this plot, we have used shap values for a possible outcome.

From this figure, we can see that the ‘date’ is the most important feature, whereas ‘temp_condition’ is the least important. The feature ‘date’ has high feature value which has a positive impact on the outcome. We can also observe that, as the feature value of ‘time’ is increasing, the positive impact on the prediction is also increasing. Though the feature ‘temp_condition’ has no effect on model prediction, there are two extreme cases where a high value has a negative impact and another high value has a positive impact on the model output.

The summary plot of SHAP for the Fridge dataset using MLP is shown in Figure 5.7. The data points from the IoT Fridge dataset are represented by the dots in the figure. The names of the characteristics are shown on the Y-axis in descending order of contribution, from highest (sphone_signal) to lowest (door_state). The X-axis displays SHAP values, which indicate whether the feature contribution has a positive or negative influence.

According to this figure, the ‘sphone_signal’ property is the most significant, while the ‘door state’ feature is the least important. The feature ‘sphone_signal’ has a low feature value, which has a favorable influence on the outcome. We can also see that when the feature value of ‘door state’ increases, so does its positive influence on prediction.

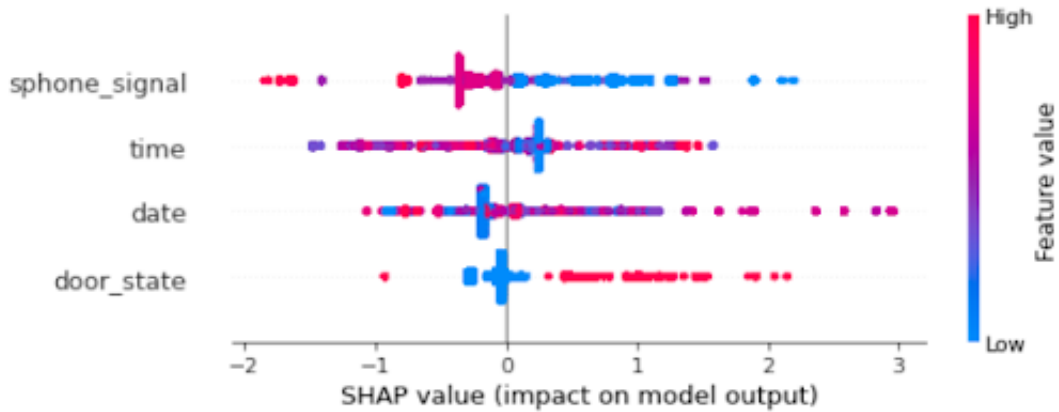


Figure 5.7: SHAP Summary Plot for Fridge dataset using MLP

ELI5

Weight	Feature
0.4929 ± 0.1220	date
0.4412 ± 0.0795	time
0.0570 ± 0.1074	temp_condition
0.0090 ± 0.0176	fridge_temperature

Figure 5.8: Feature importance of Fridge dataset using ELI5

In Figure 5.8, we have used the `show_weights()` method of ELI5 to show the feature importance in descending order for the prediction of the Random Forest model on the IoT Fridge dataset. We can use this to find out how this model works on a global scale. According to the figure, the feature 'date' has the highest approximate weight of 0.4929, while the feature 'fridge-temperature' has the lowest importance with an approximate weight of 0.0090. We can also observe that the 'date' and 'time' features are in Green color but the 'temp-condition' and 'fridge_temperature' features are not. The Green color represents the most significant feature with a positive contribution whereas Red represents the least significant feature with a negative impact. Because there are no Red features in this table and the feature weights are all positive, we can conclude that all of the features have a contribution to the model prediction.

In Figure 5.9, we have used the `show_prediction()` method of ELI5 for a single instance. This allows us to see how each feature contributes to the prediction of a certain input sample. The Random Forest model has accurately predicted the output for an input sample of the IoT fridge dataset. We have observed the top four targets for this prediction. The green color represents the most significant features of that prediction, while the red color represents the least important. According to this figure, the prediction probability of 'ddos' is 0.810, whereas the likelihood of 'normal' is 0.190. Other targets have a 0% chance of being predicted. The 'date' and 'time' are the two most important features with the contribution of (+0.557) and (+0.200) respectively for this particular prediction. The actual values of 'date' and 'time' are (-0.125) and (0.928) respectively. However, with the target class 'normal',

Prediction : ddos
Actual : ddos
y=ddos (probability **0.810**) top features

Contribution ²	Feature	Value
+0.557	date	-0.125
+0.200	time	0.928
+0.034	fridge_temperature	-0.646
+0.017	<BIAS>	1.000
+0.001	temp_condition	0.333

y=normal (probability **0.190**) top features

Contribution ²	Feature	Value
+0.853	<BIAS>	1.000
-0.006	temp_condition	0.333
-0.042	fridge_temperature	-0.646
-0.254	time	0.928
-0.360	date	-0.125

y=backdoor (probability **0.000**) top features

Contribution ²	Feature	Value
+0.202	time	0.928
+0.061	<BIAS>	1.000
+0.007	fridge_temperature	-0.646
+0.004	temp_condition	0.333
-0.274	date	-0.125

y=injection (probability **0.000**) top features

Contribution ²	Feature	Value
+0.054	date	-0.125
+0.012	<BIAS>	1.000
+0.001	temp_condition	0.333
+0.001	fridge_temperature	-0.646
-0.068	time	0.928

Figure 5.9: ELI5 explanation of a prediction using Random Forest in Fridge dataset

a completely opposite scenario can be observed. These ‘time’ and ‘date’ features contribute the least to the target ‘normal’. Similarly, the feature ‘date’ contributes the least to the ‘backdoor’ attack, while ‘time’ contributes the least to the ‘injection’ attack.

Chapter 6

Conclusion and Future Work

6.1 Conclusion and Future Work

The expansion of the IoT sector increases the number of potential threats that may have an influence on the effectiveness, gadget safety, and especially our confidentiality. As a result, keeping IoT systems secure has become a big challenge nowadays. Because of the unique structure of the number of co networks and design, the Operating system presents new security issues that go far beyond traditional information protection. Consequently, to put up with this factor, we created a brief summary of the identification and classification of different IoT network threats using machine learning and deep learning algorithms. All algorithms performed well (above 96 percent) in binary classification for the unbalanced dataset. When all of the classifiers were compared, Decision Tree and Random Forest outperformed the others (by more than 99 percent) in both binary and multiclass classification. We have applied undersampling, oversampling, and SMOTE techniques on the Fridge dataset and evaluated the result for different classifiers. Moreover, approaches like LIME, SHAP, and ELI5 are used to interpret and understand the models. However, For the device constraints, we could not use real-time data for our experiment. In the future, we will try to use real-time data for testing the classifiers.

Bibliography

- [1] A. Banafa, “Major challenges iot is facing,” *Retrieved from*, 3.
- [2] D. Roe, “Big problems with the internet of things,” *CMS Wire. Disponível em: <https://www.cmswire.com/cms/internet-of-things/7-big-problems-with-the-internet-of-things-024571.php>*, 7.
- [3] T.-K. An and M.-H. Kim, “A new diverse adaboost classifier,” in *2010 International Conference on Artificial Intelligence and Computational Intelligence*, vol. 1, 2010, pp. 359–363. DOI: 10.1109/AICI.2010.82.
- [4] C. Zhao, Y. Gao, J. He, and J. Lian, “Recognition of driving postures by multi-wavelet transform and multilayer perceptron classifier,” *Engineering Applications of Artificial Intelligence*, vol. 25, no. 8, pp. 1677–1686, 2012, ISSN: 0952-1976. DOI: <https://doi.org/10.1016/j.engappai.2012.09.018>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0952197612002564>.
- [5] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, *et al.*, “Xgboost: Extreme gradient boosting,” *R package version 0.4-2*, vol. 1, no. 4, pp. 1–4, 2015.
- [6] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.
- [7] N. Joshi, “4 challenges that are faced by iot developers,” 2017.
- [8] R. Primartha and B. A. Tama, “Anomaly detection using random forest: A performance revisited,” in *2017 International conference on data and software engineering (ICoDSE)*, IEEE, 2017, pp. 1–6.
- [9] C. Albert, “Problems with the internet of things you need to know,” 2018.
- [10] W. Zhou, Y. Jia, A. Peng, Y. Zhang, and P. Liu, “The effect of iot new features on security and privacy: New threats, existing solutions, and challenges yet to be solved,” *IEEE Internet of things Journal*, vol. 6, no. 2, pp. 1606–1616, 2018.
- [11] I. Wildani, I. Yulita, *et al.*, “Classifying botnet attack on internet of things device using random forest,” in *IOP Conference Series: Earth and Environmental Science*, IOP Publishing, vol. 248, 2019, p. 012 002.
- [12] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, “Ton_iot telemetry dataset: A new generation dataset of iot and iiot for data-driven intrusion detection systems,” *IEEE Access*, vol. 8, pp. 165 130–165 150, 2020.
- [13] V. Dutta, M. Choraś, M. Pawlicki, and R. Kozik, “A deep learning ensemble for network anomaly and cyber-attack detection,” *Sensors*, vol. 20, no. 16, p. 4583, 2020.

- [14] W. Iqbal, H. Abbas, M. Daneshmand, B. Rauf, and Y. A. Bangash, “An in-depth analysis of iot security requirements, challenges, and their countermeasures via software-defined security,” *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10 250–10 276, 2020.
- [15] S. I. Popoola, B. Adebisi, M. Hammoudeh, G. Gui, and H. Gacanin, “Hybrid deep learning for botnet attack detection in the internet-of-things networks,” *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4944–4956, 2020.
- [16] M. Woźniak, J. Siłka, M. Wiczorek, and M. Alrashoud, “Recurrent neural network model for iot and networking malware threat detection,” *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5583–5594, 2020.
- [17] X. Zhou, Y. Hu, W. Liang, J. Ma, and Q. Jin, “Variational lstm enhanced anomaly detection for industrial big data,” *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3469–3477, 2020.
- [18] T. M. Booij, I. Chiscop, E. Meeuwissen, N. Moustafa, and F. T. den Hartog, “Ton_iiot: The role of heterogeneity and the need for standardization of features and attack types in iot network intrusion data sets,” *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 485–496, 2021.
- [19] D. Javeed, T. Gao, M. T. Khan, and I. Ahmad, “A hybrid deep learning-driven sdn enabled mechanism for secure communication in internet of things (iiot),” *Sensors*, vol. 21, no. 14, p. 4884, 2021.
- [20] B. Jovanović, “45 fascinating iot statistics for 2021 | the state of the industry,” 2021.
- [21] A. H. K. Mohammed, H. Jebamikyous, D. Nawara, and R. Kashef, “Iiot cyber-attack detection: A comparative analysis,” in *International Conference on Data Science, E-learning and Information Systems 2021*, 2021, pp. 117–123.
- [22] R. K. Muna, H. T. Maliha, and M. Hasan, “Demystifying machine learning models for iot attack detection with explainable ai,” Ph.D. dissertation, Brac University, 2021.
- [23] G. Nick, “How many iot devices are there in 2021?[all you need to know],” *Tech Jury*, 2021.
- [24] A. K. Sahu, S. Sharma, M. Tanveer, and R. Raja, “Internet of things attack detection using hybrid deep learning model,” *Computer Communications*, vol. 176, pp. 146–154, 2021.
- [25] M. Sarhan, S. Layeghy, and M. Portmann, “Evaluating standard feature sets towards increased generalisability and explainability of ml-based network intrusion detection,” *arXiv preprint arXiv:2104.07183*, 2021.
- [26] J. Shareena, A. Ramdas, H. AP, *et al.*, “Intrusion detection system for iot botnet attacks using deep learning,” *SN Computer Science*, vol. 2, no. 3, pp. 1–8, 2021.
- [27] S. Zarazua, “Artificial neural networks,” 2021.