REPUBLIC OF TURKEY

YILDIZ TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

ENHANCING SECURITY LEVEL OF INDUSTRIAL INTERNET OF THINGS DEVICES BASED ON BOTNET DETECTION AND FEATURE SELECTION

Weam Husham Abdulwahhab AL-JABBARI

MASTER OF SCIENCE THESIS

Department of Computer Engineering

Computer Engineering Program

Supervisor

Prof. Dr. Hasan Hüseyin BALIK

Co-Supervisor

Assoc. Prof. Dr. Muhammed Ali AYDIN

December, 2022

REPUBLIC OF TURKEY

YILDIZ TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

ENDÜSTRİYEL NESNELERİN İNTERNETİ CİHAZLARININ GÜVENLİK SEVİYESİNİN BOTNET TESPİTİ VE ÖZELLİK SEÇİMİ TABANLI GELİŞTİRİLMESİ

A thesis submitted by Weam Husham Abdulwahhab AL-JABBARI in partial fulfillment of the requirements for the degree of MASTER is approved by the committee on 27.12.2022 in Department of Computer Engineering, Computer Engineering Program.

Prof. Dr. Hasan Hüseyin BALIK
Yıldız Technical University
Supervisor

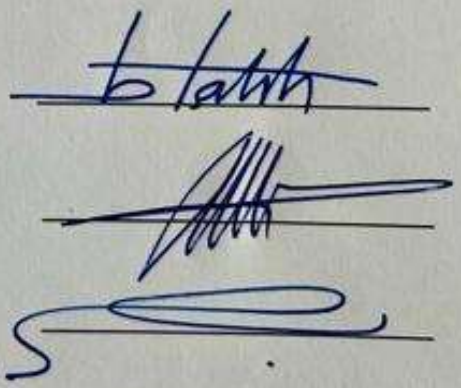
Assoc. Prof. Dr. Mohammed Ali AYDIN
Istanbul University-Cerrahpaşa
Co- supervisor

Approved By the Examining Committee

Prof. Dr. Hasan Hüseyin BALIK, Supervisor
Yildiz Technical University

Prof. Dr. Nizamettin AYDIN, Member
Yildiz Technical University

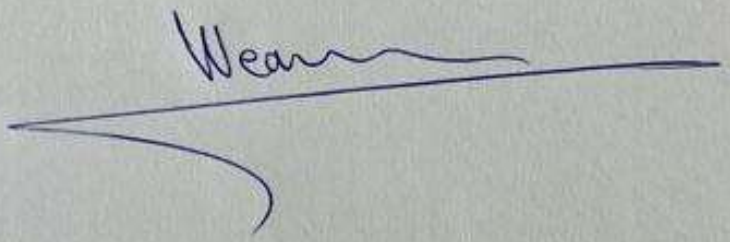
Prof. Dr. Belal Ismael KHALIL, Member
University of ANBAR



I hereby declare that I have obtained the required legal permissions during data collection and exploitation procedures, that I have made the in-text citations and cited the references properly, that I haven't falsified and/or fabricated research data and results of the study and that I have abided by the principles of the scientific research and ethics during my Thesis Study under the title of Enhancing Security Level of Industrial Internet of Things Devices based on BotNet Detection and Feature Selection supervised by my supervisors, Prof. Dr. Hasan Hüseyin BALIK and Assoc. Prof. Dr. Mohammed Ali AYDIN. In the case of a discovery of false statement, I am to acknowledge any legal consequence.

Weam Husham Abdulwahhab AL-JABBARI

Signature



Dedicated to the Souls of

my Father and Sister

ACKNOWLEDGEMENTS

I am grateful to my supervisors, Prof. Dr. Hasan Hüseyin BALIK and Assoc. Prof. Dr. Mohammed Ali AYDIN, for their essential counsel, unwavering support, and tolerance during my M.Sc. research. Their vast wisdom and extensive knowledge has been an inspiration to me throughout my study. I would like to extend my deepest gratitude to everyone in the Computer Engineering Department at Yildiz Technical University, most especially to Department Head Prof. Dr. Nizamettin AYDIN and the secretary (Suna Hanim and Filiz Hanim) for their boundless enthusiasm, help, and support during my study. In addition, I would like to express my gratitude to Lect. Nurdan YILDIZ GÜNDÜZ for all of her guidance, assistance, and encouragement.

I would also like to give special thanks to my dearest friend B. Al-Ani for his continuous support and understanding when undertaking my research and writing my project.

I would like to express my gratitude to my mother and my brothers Ahmed and Ali and all family members for their continuous supporting and prayers.

Finally, words cannot express my gratitude to my soulmate, my husband Wisam for his invaluable patience and feedback. I also could not have undertaken this journey without the tremendous understanding and encouragement of my son Dawood and my Daughter Haya, it would be impossible for me to complete my study without them.

Weam Husham Abdulwahhab AL-JABBARI

TABLE OF CONTENTS

LIST OF SYMBOLS

ang	Angle of a move
C	Caloric value
circ	Circular motion and mimic the circle system
J	Each diminution
E	Energy at a current time
Y	Final Energy
I	Iteration value
Lb	Lower bounds of variables
X'	Normalized feature
n	Number of oyster catcher
Leaderpos	Old position of the leader
L	Optimal Length of oyster
X	Position of (Oyster \ Leader) or significant feature
r2	Radius (random no. Between 0,1)
r	Random value between (0,1)
delta	Random value between (lb,ub)
T	Time required to open oyster
ub	upper bounds of variables

LIST OF ABBREVIATIONS

AF	Agnostic-Features
AFRL	Air Force Research Lab
AI	Artificial Intelligence
AL	Alerts
ANN	Artificial Neural Network
AOA	Arithmetic Optimization Algorithm
ATT&CK	Adversarial Tactics, Techniques and Common Knowledge
BA	Bat Algorithm
BBFO	Binary Bacterial Foraging Optimization
BoT-IoT	Bot-Internet of Things
BotNet	Bot Network
BS	Base Station
BSA	Bird Swarms Algorithm
C&C	Command and Control
CC	Complete Capture
CKC	Cyber Kill Chain
CNSC	Complete Network and System Configuration
Colab	Colaboratory
CPS	cyber-physical systems
CPU	Central Processing Unit
CSHO	Competitive Swarm Henry Optimization
CSV	Comma-Separated Values
DARPA	Defense Advanced Research Projects Agency
DAT	Divers Attacks Scenario and Types
DDD	Diverse Data Duration
DDoS	Distributed Denial Of service

DE	Deferential Evolution
DL	Deep Learning
DoS	Denial Of service
DT	Decision Tree
EO	Eurasian Oystercatcher
EOO	Eurasian Oystercatcher Optimizer
FGGWO	Fractional Gravitational Grey Wolf Optimization
FS	Feature Selection
GA	Genetic Algorithm
GIWRF	Gini Impurity-based Weighted Random Forest
GPS	Global Positioning System
GRU	Gated Recurrent Unit
GTO	Gorilla Troops Optimizer
GWO	Grey Wolf Optimization
HDS	Heterogeneous Data Sources
HIIDS	Health-Intelligent Intrusion Detection System
HR	Host Resources
HTTP	Hypertext Transfer Protocol
IDSs	Intrusion Detection Systems
IIoT	Industrial Internet of Things
IIoT-CP	IIoT Connectivity Protocols
IIoT-T	IIoT Traces
IIRA	Industrial Internet Reference Architecture
IoT	Internet of Things
IP	Internet Protocol
IRC	Internet Relay Chat
IrDA	Infrared Data Association
IT	Information Technology

KDD	Knowledge Discovery in Databases
KNN	K-Nearest Neighbors
L	Logs
LAN	Local Area Network
LD	Labeled Dataset
MALC	Managing Across the Lifecycle Capstone
MB	Mega Byte
MD	Metadata
MIT	Massachusetts Institute of Technology
ML	Machine Learning
MOPSO	Multi-Objective Particle Swarm Optimization
NADAM	Nesterov-Accelerated Adaptive Moment Estimation
NB	Naïve Bayes
N-BaIoT	Network Based on IoT
NFC	Near-Field Communication
NN	Nearest Neighbor
NSL-KDD	Network Security Laboratory-KDD
NT	Network Traffic
OT	Operational Technology
P2P	Peer-to-Peer
PA	Public Availability
PC	Personal Computer
pcap	Packet Capture
PP	Physical Process
PSO	Particle Swarm Optimization
R2L	Remote to Local
RA	Recent Attacks
RAM	Random Access Memory

RaNN	Random Neural Network
RF	Random Forest
RFID	Radio-Frequency Identification
RHSO	Rock Hyraxes Swarm Optimization
RNT	Realistic Network Traffic
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TON-IoT	Telemetry of Network-IoT
U2R	User to Root
UCI	University of California, Irvine
UDP	User Datagram Protocol
UNSW-UNB15	University of New South Wales-Network Bots 15
UWB	Ultra-Wideband
WiFi	Wireless Fidelity
WSN	Wireless Sensor Networks
ZigBee	Zonal Intercommunication Global - Standard

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Enhancing Security Level of Industrıal Internet of Things Devices Based On BotNet Detection And Feature Selection

Weam Husham Abdulwahhab AL-JABBARI

Department of Computer Engineering

Master of Science Thesis

Supervisor: Prof. Dr. Hasan Hüseyin BALIK

Co-Supervisor: Assoc. Prof. Dr. Muhammed Ali AYDIN

The cybersecurity threat landscape in the Industrial Internet of Things (IIoT) context is becoming increasingly sophisticated as a result of the growing attack surface and the accumulative complexity of new system behaviors. The potential for BotNets to take control of devices attached to the Industrial Internet of Things presents enormous security risks. Exploitation of private information, violation of personal rights, and even occasionally the commission of cyberattacks that put people's lives in danger, such as the sabotaging of medical equipment, are all potential consequences of BotNets attack. Techniques based on machine learning (ML) have been found to have good prediction accuracy when it comes to determining whether data retrieved from network traffic is safe or malicious. Net flow anomaly detection using ML is a fascinating topic because it can interpret complex network traffic and detect anomalies. This thesis aims to lower the risk of exposure to BotNet attacks occurring in the environment of the IIoT, which will

ultimately result in an enhancement in the level of security afforded to IIoT devices. Hence, the determination of this thesis is to reveal the design and implementation of a multi-classification BotNets detection system that is based on the selection of features and machine learning algorithms. The X-IIoTID dataset, which is a specialized dataset on IIoT cyber threats, was utilized in the suggested system in order to serve as a benchmark for its performance. Separately from one another, the Eurasian Oystercatcher Optimizer (EOO) and Rock Hyraxes Swarm Optimization (RHSO) algorithms are utilized as features selection approaches in conjunction with a fitness analyzer that makes use of the ML algorithm. The suggested fitness analyzer consisted of two elements: the first of these was the level of accuracy that could be achieved by ML, and the second of these was the total number of features that could be extracted. Since dealing with less data has a favorable effect on both the system's accuracy and its resource consumption, the EOO and RHSO algorithms are employed to reduce the attributes of the dataset. After examining the results obtained by both the EOO and RHSO algorithms, it was determined that the RHSO algorithm produced favorable performance. Both the original X-IIoTID dataset, with all of its original features, as well as the new sub datasets had the machine learning methods Random Forest and KNN applied to it in order to create multiclass classifiers (with two subsets of features using dataset mapping). When compared to the previous research works, the results revealed a substantial amount of development prospects in terms of accuracy and the consumption of available resources. The results of this thesis indicate that the proposed system is capable of producing superior outcomes to those achieved by the comparable approaches. The experimental study revealed that employing the RHSO algorithm in conjunction with the Random Forest algorithm was superior, with a rate of accuracy of (99.88%), time of detection (1.25 minute), and memory consumption of (49.11 MB).

Keywords: Cybersecurity, BotNet, Eurasian Oystercatcher Optimizer (EOO), Rock Hyraxes Swarm Optimization (RHSO), X-IIoTID Dataset.

ÖZET

Endüstriyel Nesnelerin İnterneti Cihazlarının Güvenlik Seviyesinin BotNet Tespiti ve Özellik Seçimi Tabanlı Geliştirilmesi

Weam Husham Abdulwahhab AL-JABBARI

Bilgisayar Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Danışman: Prof. Dr. Hasan Hüseyin BALIK

Eş-Danışman: Doç. Dr. Muhammed Ali AYDIN

Endüstriyel Nesnelerin İnterneti (IIoT) bağlamındaki siber güvenlik tehdidi ortamı, artan saldırı yüzeyi ve yeni sistem davranışlarının biriken karmaşıklığının bir sonucu olarak giderek daha karmaşık hale geliyor. BotNet'lerin Endüstriyel Nesnelerin İnterneti'ne bağlı cihazların kontrolünü ele geçirme potansiyeli, çok büyük güvenlik riskleri sunar. Özel bilgilerin suistimal edilmesi, kişilik haklarının ihlali ve hatta bazen tıbbi ekipmanın sabote edilmesi gibi insanların hayatlarını tehlikeye atan siber saldırıların komisyonu, BotNet saldırısının olası sonuçlarıdır. Ağ trafiğinden alınan verilerin güvenli mi yoksa kötü amaçlı mı olduğunun belirlenmesi söz konusu olduğunda, makine öğrenimine (ML) dayalı tekniklerin iyi bir tahmin doğruluğuna sahip olduğu bulunmuştur. ML kullanarak net akış anormalliği algılama, karmaşık ağ trafiğini yorumlayabildiği ve anormallikleri algılayabildiği için büyüleyici bir konudur. Bu tez, IIoT ortamında meydana gelen ve sonuçta IIoT cihazlarına sağlanan güvenlik düzeyinde bir iyileştirme ile

sonuçlanacak olan BotNet saldırılarına maruz kalma riskini azaltmayı amaçlamaktadır. Bu nedenle, bu tezin amacı, öznitelik seçimine ve makine öğrenimi algoritmalarına dayalı çok sınıflı bir BotNet algılama sisteminin tasarımını ve uygulamasını göstermektir. Önerilen sistemde, IIoT siber tehditler konusunda uzmanlaşmış bir veri seti olan X-IIoTID veri seti, performansı için bir kıyaslama olarak hizmet etmek üzere kullanılmıştır. Eurasian Oystercatcher Optimizer (EOO) ve Rock Hyraxes Swarm Optimization (RHSO) algoritmaları birbirinden ayrı olarak, ML algoritmasını kullanan bir uygunluk analizcisi ile birlikte özellik seçim yaklaşımları olarak kullanılır. Önerilen uygunluk analizörü iki unsurdan oluşuyordu: bunlardan birincisi ML ile elde edilebilecek doğruluk seviyesi ve ikincisi ise çıkarılabilecek toplam öznitelik sayısıydı. Daha az veriyle uğraşmanın hem sistemin doğruluğu hem de kaynak tüketimi üzerinde olumlu bir etkisi olduğundan, veri kümesinin özniteliklerini azaltmak için EOO ve RHSO algoritmaları kullanılır. Hem EOO hem de RHSO algoritmaları ile elde edilen sonuçlar incelendikten sonra RHSO algoritmasının olumlu performans ürettiği belirlendi. Hem orijinal X-IIoTID veri kümesi, hem de tüm orijinal özellikleriyle birlikte yeni alt veri kümeleri, çok sınıflı sınıflandırıcılar (veri kümesi eşlemesini kullanan iki alt özellik alt kümesiyle) oluşturmak için kendisine uygulanan makine öğrenme yöntemleri Random Forest ve KNN'ye sahipti. . Önceki araştırma çalışmalarıyla karşılaştırıldığında, sonuçlar, doğruluk ve mevcut kaynakların tüketimi açısından önemli miktarda gelişme beklentisi ortaya koydu. Bu tezin sonuçları, önerilen sistemin karşılaştırılabilir yaklaşımlarla elde edilenlere göre daha üstün sonuçlar üretme yeteneğine sahip olduğunu göstermektedir. Deneysel çalışma, Rastgele Orman algoritması ile birlikte RHSO algoritmasının kullanılmasının, doğruluk oranı (%99.88), algılama süresi (1.25 dakika) ve bellek tüketimi (49.11 MB) ile üstün olduğunu ortaya koydu.

Anahtar Kelimeler: Siber Güvenlik, BotNet, Avrasya Oystercatcher Optimizer (EOO), Rock Hyraxes Swarm Optimization (RHSO), X-IIoTID Veri Kümesi.

1

INTRODUCTION

1.1 Literature Review

There are many similarities between IoT and IIoT in terms of integrity, privacy, and security because many of the technological solutions for IIoT are built on IoT. While IIoT-enabled devices make people's and businesses' lives easier, their privacy may suffer as a result [1]. IIoT devices have hardware and software that may be used to track user activity, so it is essential to develop regulations and technical solutions to guarantee that users' privacy, security, and autonomy are constantly protected. The IIoT creates a wide platform for sophisticated cyberattacks as more and more devices are linked to the Internet nearly every day. Data leaks, loss of privacy, and the potential for misuse due to unauthorized access that could seize control of the devices are common worries with regard to IIoT devices [2]. Distributed denial of service (DDoS) cyberattacks using various protocols of communication, data breaches by keylogging and data leakage, tracing via biometrics, and network scans for open ports are a few frequent threats that affect IIoT devices and networks. BotNets are often used to carry out many of these cyberattacks on IIoT systems and networks [3].

When it comes to developing a model, which is effective in network attack recognition that is based on machine learning, feature selection is an essential stage. One can anticipate that a more condensed set of features will lead to more effective collection and storage, both of which are essential components of an IIoT network environment with high speeds [4]. To put it another way, the effectiveness of machine learning (ML)-based on intrusion detection systems is contingent on the quality and veracity of the data that is used to train and assess the ML models. Utilizing Features Selection will significantly cut down on the amount of time required to finish the detection, as well as reduce the amount of CPU and main memory (RAM) consumption, and in several cases improve accuracy.

An area of specialization within the field of Artificial Intelligence known as Machine Learning has attracted more attention in modern years. This is largely because ML models and algorithms have grown more potent and are now integrated with more data and computing capacity, boosting their effectiveness. An unindustrialized approach for spotting network movement that indicates harmful behavior on a targeted network is ML-based intrusion detection systems. Researchers have effectively applied ML to the creation of more sophisticated intrusion detection systems technologies, obtaining a reasonable attack detection accuracy [5]. From labelled data samples, supervised ML seeks to understand and learn about sophisticated security events. However, due to a number of logistical and privacy concerns, collecting labeled datasets from real production networks is quite difficult. As a result, researchers have developed simulated intrusion detection system datasets, which are often produced in controlled test conditions in which data tags for both malicious and benign flow can be consistently and quickly inserted.

When working with a large dataset, feature selections plays a crucial role in filtering out unnecessary, duplicate, or redundant features. It is a preprocessing procedure for vast amounts of data that can be used to select a subset of features or a collection of attributes, which aids in the construction of an effective model for characterization of the selected subset. It also has several additional goals, such as reducing dimensionality, decreasing the size of the data needed for the learning process, refining predictive accurateness, and expanding the developed models. This section examines the literature pertaining to feature selection in concentration with attack detection.

By combining the Particle Swarm Optimization (PSO) approach with features selection, Nilesh Kunhare et al. (2020) [6] discussed improving the accuracy and detection rate of IDSs. Researchers only employed 10 features from the NSL-KDD dataset. To obtain the optimum output, the PSO algorithm was employed with a separate number of iterations and fixed particles for the features that were chosen. On the dataset's sets of training and testing, the results were contrasted with ML algorithms. The observed performance metrics, including accuracy, were superior to those produced by other algorithms.

Maria Habib et al. (2020) [7] introduced a method for transforming conventional IDSs into intelligent, adaptive, and multi unbiased for IoT networks. The foundation of the updated approach is the integration of MOPSO-Lévy, which stands for multi-objective particle swarm optimization with a Lévy flight randomization component. The updated MOPSO-Lévy was subjected to testing on actual IoT network data taken from the UCI repository. When compared to cutting-edge evolutionary multi-objective algorithms, MOPSO-Lévy has shown better outcomes.

Maha M. Althobaiti et al. (2021) [8] suggested a model that eliminate noise from the dataset. The described model then adopted a feature selection strategy based on binary bacterial foraging optimization (BBFO) to pick the best set of features. The gated recurrent unit (GRU) model was utilized as a method for the detection of breaches in industrial cyber-physical systems (CPS) context. By fine-tuning the GRU model's hyperparameters, the detection rate was increased with the help of the Nesterov-accelerated Adaptive Moment Estimation (NADAM) optimizer. Tests utilizing industrial CPS data verified the model's accuracy.

Abdullah Alharbi et al. (2021) [9] suggested a Neural Networks with Local Global Best Bat Algorithm to elect feature subgroups for effective BotNet attack recognition. The suggested Bat Algorithm (BA) made use of the local-global best-based inertia weighting in order to apprise the velocity of the bats that were a part of the swarm. Researchers suggested using a Gaussian distribution for population initialization to deal with BA swarm diversity. In order to improve exploration throughout each generation, the local-global best function and Gaussian density function were added after the local search process. The proposed technique was assessed on an N-BaIoT dataset.

By enhancing the functionality of the Gorilla Troops Optimizer (GTO) with the process for bird swarms (BSA), also known as the GTO-BSA method, Saif S. Kareem et al. (2022) [10] attempted to provide a new feature selection strategy. With the aid of BSA, which is effective at locating the regions that yield the optimal solution, GTO's effectiveness was improved. The act of the proposed approach was evaluated on four IoT-IDS datasets: BoT-IoT, CICIDS-2017, UNSW-NB15, and NSL-KDD.

Sohail Saif et al. (2022) [11] presented a ML and meta-heuristic model to construct a health-intelligent intrusion detection system (HIIDS). The framework has been used to

combine ML algorithms like KNN and Decision Tree with metaheuristic algorithms like PSO, GA, and DE. The NSL-KDD training dataset was used to create a decision tree-based ML model.

In order to locate attackers in the IoT setting, Mythili Henry Boopathi (2022) [12] developed an IDS utilizing the suggested Competitive Swarm Henry Optimization (CSHO)-based Deep Maxout network. By incorporating the characteristics from optimization methods into the classifier's training, the implemented approach recognizes attacks by determining the suitable overall result with the best load value for each iteration. The info is routed towards the base station (BS) through the routing process, which is carried out using the Fractional Gravitational Grey Wolf Optimization (FGGWO) algorithm, and is accessible to the network's distributed nodes.

Raisa Abedin Disha et al. (2022) [13] suggested a strategy that utilized multiple classification models for the IDS due to the efficiency of ML techniques. The UNSW-NB15 and TON-IoT datasets were utilized for offline examination to assess the act of the models. A set of attributes was chosen using a Gini Impurity-based Weighted Random Forest (GIWRF) model as the features selection procedure because the effectiveness of IDS suffers with a high-dimensional feature vector. Twenty features from UNSW-NB15 and ten features from the TON-IoT dataset were chosen.

Mohammed Otair et al. (2022) [14] proposed a method for IDS that combines Grey Wolf Optimization (GWO) and Particle Swarm Optimization (PSO) to tackle features selection issues and apply the optimal value to update each grey wolf position's information. The NSL-KDD dataset was utilized to measure the effectiveness of the method. The SVM and k-means algorithms were utilized in order to conduct the evaluations necessary to determine the classification's accuracy, rate of detection, the rate of false alarm, number of attributes. The outcomes have demonstrated that the proposed technique achieved the essential enhancement of the GWO algorithm.

In their research, Aniss Chohra et al. (2022) [15] developed an approach that concentrated on determining the appropriate hyperparameters for ensemble approaches to identify the essential features from a specified dataset. The suggested method was tested on datasets: UNSW-NB15, and NSL-KDD. A swarm intelligence optimization (PSO) was combined with ensemble methods to create the proposed

approach. They chose the proper collection of features on every validation dataset using the best solutions identified by the optimization algorithm. They developed and optimized a self-learning autoenocoder for the identification of anomalies in every one of these datasets using solely those features. According to the results, their anomaly detection algorithms surpassed the most effective techniques used on these datasets.

1.2 Objective of the Thesis

In accordance with [16], developments in the Industrial Internet of Things (IIoT), networking, the continued growth of intelligent ecosystems, as well as the urgent demand for automation of processes have not only created new prospects but also widened the terrain of cyber-threats and attacks. This has mostly been observed even as industries rush to conform to the objectives of the highly anticipated industry 4.0. In particular, it is now clear that the fusion and convergence of industrial operational technology (OT) and information technology (IT) leads to intelligence, complexity in the IIoT ecosystem, changes in society, and general alterations to the designs of digital forensic investigations and security measures. As a result, as many industries try to reach their industry 4.0 goals, the digital forensic side of IIoT isn't really taken into account or integrated. This is because the cyber security threat landscape in this context is also getting more complicated because the attack surface is getting bigger and new system behaviors are becoming more different. According to the previously discussed literature, ML learning techniques have shown good prediction accuracy in identifying data from network traffic as safe or malicious. Devices net flow anomaly detection using a ML technique is a promising subject due to its power of understanding complex traffic on the network and detecting anomalies, even though there are not many solutions used today for larger IIoT contexts, including corporate networks. When deploying anomaly detection models in actual IIoT operational environments, the challenges for ML-based BotNet attack detection include scaling to larger sizes of network traffic; reducing computational power consumption; encompassing different attack types; classifying traffic aggregation as benign or malicious; and gaining high accuracy of ML model results by easing the fitted model to avoid additional investments. In contrast to

other studies, the main target of this thesis is to use feature selection algorithms to obtain high performance of standard ML models with less consumption of resources.

1.3 Hypothesis

The most significant goal of this thesis is to decrease the exposure of BotNet attacks in the IIoT environment, which will ultimately result in enhancing the security of IIoT devices. The following will be the study's primary objectives in brief:

- To explore the IIoT BotNet attack detection as a multiclass classification problem utilizing the IIoT datasets.
- To assess and utilize the new X-IIoTID dataset [17], which is a specialized dataset in the IIoT cyber attacks field. According to the authors' knowledge, three researchers have only used this dataset including the creator of the dataset.
- The Eurasian Oystercatcher Optimizer (EOO) [18], and Rock Hyraxes Swarm Optimization (RHSO) [19] was chosen to be applied as a features selection algorithms. This will be beneficial in getting high system accuracy, decrease training time and overall system resources.
- To utilize a machine learning algorithm as a fitness for the two features selection algorithms. This guarantees that the algorithms' output features will all be trained. Because the feature selection algorithms will actually train on all the chosen features based on their fitness, this strengthens the algorithms.
- To evaluate the new sub datasets having the reduced features using state of the art ML algorithms.
- To compare the results of both features selection algorithms.
- To compare the results with the literatures.

The thesis consisting of four chapters (including this chapter), Chapter Two provides a deep theoretical background about the subject. Chapter Three explains the proposed system and all of its details including system requirements and tools that are used to

implement this system. Chapter Four contains the simulation of the suggested system and the discussion and examination of the outcomes that are gained through out this work. It also includes the most important work conclusions and recommendations aimed at future work.

2

THEORETICAL BACKGROUND

2.1 Introduction

In this chapter, the fundamentals of IoT and IIoT systems are discussed in addition to their differences. A review of BotNet detection systems has been presented, along with a brief description of each type. Additionally, ML and the various types of strategies it employs to detect BotNets have been discussed, with an emphasis on the algorithms employed in this thesis. The strategies for features selection are also demonstrated. The final part of this chapter presents tables that offer a short review of the most current studies conducted in this area. Most of the common datasets in the subject of BotNet detection are explained, as are the environments used for project implementation, with a focus on the dataset and environment utilized in this thesis.

2.2 Internet of Things (IoT)

The Internet of Things (IoT) is a network of computers and other gadgets that can interact and communicate with one another. It has arisen in the modern period and is driving the creation of innovative business process technologies [20]. The networked, IP-enabled objects are referred to as "things". Sensor Panels, self-driving vehicles, infrared data association (IrDA), laser copier, NFC data centers, video observation, ultra-wideband (UWB), tablet devices, cell phones, ZigBee, and mobile and Wi-Fi systems are a few examples of things that could be utilized. The Internet of Things (IoT) is thought of as a net of countless physical things through all of its secondary technologies (In 2026, there will be 76.45 billion digital devices, up from 24.15 billion in 2019) [21]. By 2025, the IoT could have an economic influence on the worldwide budget of $3.9 to $11.1 trillion [22]. These gadgets are equipped with various additional critical tools in addition to Internet Protocol (IP), for instance radio-frequency identification (RFID), GPS services, sensors, near field communication

(NFC), actuators, nanotechnologies, and cloud computing. Figure 2.1 shows some examples of IoT applications.

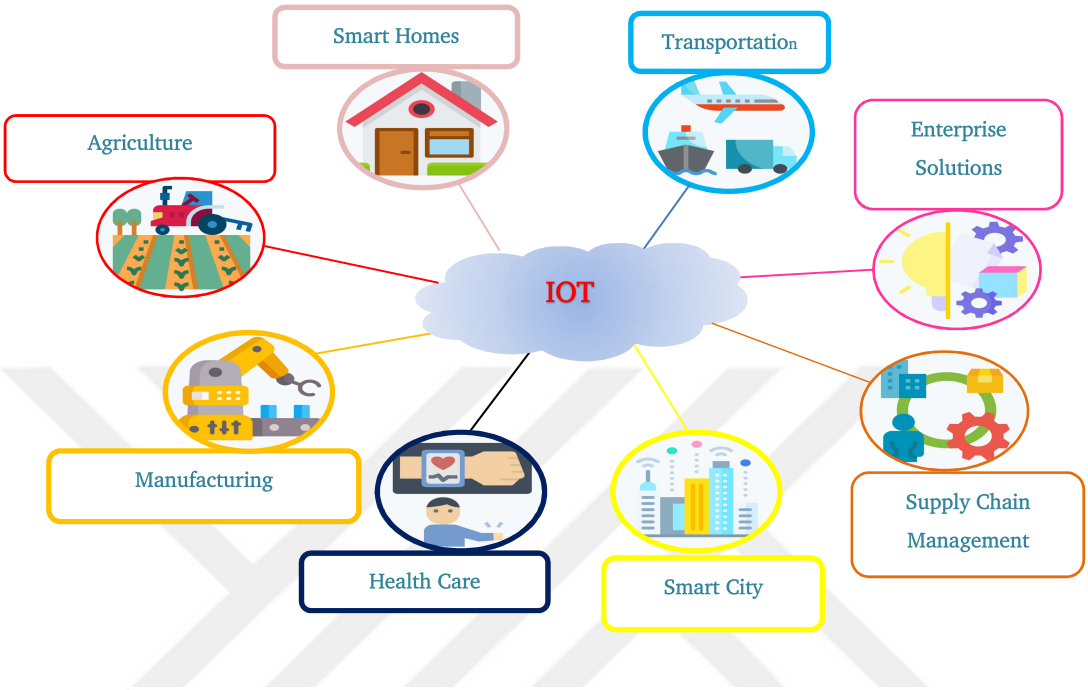


Figure 2.1 Example of IoT applications

2.3 Industrial Internet of Things (IIoT)

A groundbreaking idea that has recently gained widespread attention is the Industrial Internet of Things (IIoT), which represents a completely intelligent, connected, automated, and transparent industrial operation that boosts production processes and efficiency. Industrial IoT (IIoT) or Industry 4.0 is a principle for a cutting-edge, intelligent, completely integrated factory that uses emerging innovation (such as artificial intelligence, cloud computing, IoT, etc.) and innovative technologies (such monitoring, automation, and IIoT) can improve the construction environment through cost savings, quality improvements, and remote operating operations, among other things [23].

Digitization and the spread of smart gadgets or devices have penetrated critical industrial and infrastructure sectors. Healthcare, water, power, as well as other

components of crucial systems can run more effectively, profitably, and dependably due to gadgets having wireless and wired connectivities [24]. Such installations are referred to as the Industrial Internet of Things (IIoT), in which industrial resources are evolving to fully skillful in automatically responding to and modifying their activities depending on info obtained over the vast networked control loop spanning from the Edge and Cloud to the Enterprise layers [25]. Yet, this significant integration between Operational Technology (OT) and Information Technology (IT) systems that is at the core of the technologies of the IIoT, has extended the cyber threats and raised the likelihood that cyber-attacks will be undertaken against such vital systems [26]. Figure 2.2 illustrates a brief evolution of industry.

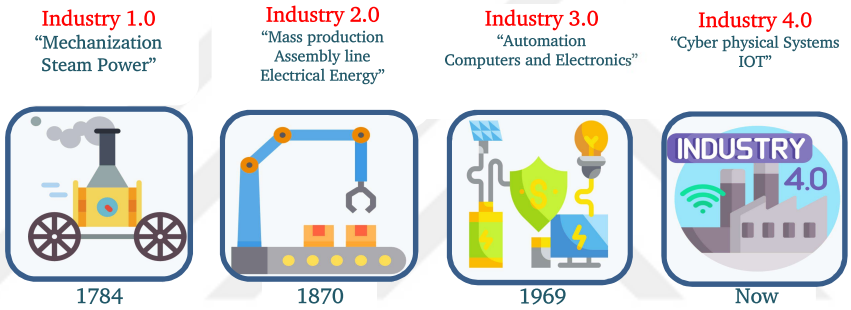


Figure 2.2 Evolution of industry

2.4 Differences between IoT and IIoT

The Industrial Internet of Things (IIoT) denotes to industrial sectors such as energy and production control, as well as interconnected sensors, instruments, and other computer-connected devices [27]. IIoT is for the industrial sector, while IoT is for the commercial sector. The next industrial revolution, formerly written as the Industrial Internet of Things, will begin within the time period called by Industry 4.0 [28]. Industry 4.0 [29] is regarded as being part of the 4th Industrial Revolution. In Industry 4.0, factories feature machines with internet communication and sensors that are connected to a computer that can manage the entire production line and draw significant conclusions on their own [30]. With the support of the Internet of Services,

both inner and intra-organizational facilities are supplied to users in order for them to utilize the production series. Cyber-security, cloud and edge computing, 3D printers, industrial automation, big data, and the Internet of Things are all facilitated by the IIoT.

Briefly, IoT and IIoT rely on hardware like sensors, internet connectivity, and embedded systems. However, IIoT devices tend to be more expensive than IoT ones because of their need for greater precision in comparison to those of IoT devices. IIoT employs more advanced technology for greater precision. Because IIoT operates in key business areas like manufacturing, equipment monitoring, and so on, IIoT services are more sophisticated than IoT systems; as technology breakthroughs improve, so does complexity. Thus, the complexity of IIoT applications exceeds that of IoT applications. IoT end requirements prioritize user comfort, whereas IIoT end requirements prioritize return on investment [31].

Despite the differences between IoT and IIoT, the two techniques have a lot in common, such as connectivity, data acquisition, storage, analytics, and visualization as it is shown in Figure 2.3.

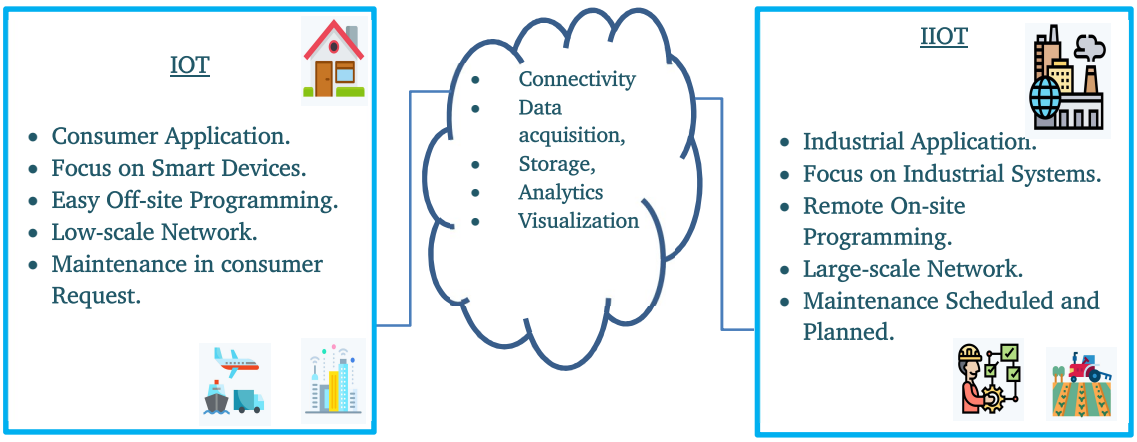


Figure 2.3 Some of the similarities and differences between IoT and IIoT

2.5 BotNets

BotNets are networks made up of host computers that have been made to serve as slaves under the control of one or more hackers known as botmaster, in order to carry out destructive actions. Hackers use a wide array of inventive methods to spread malware that has the potential to transform a system into a zombie or bot. When a hacker instead of the user is in charge of a computer, suspicious actions on the internet are carried out without the user's knowledge. In other words, BotNets are groups of computers that work together to carry out illegal operations using malicious software. Attackers frequently make use of bots in order to infect a substantial number of systems all at once. These machines collectively make up the BotNet. Such zombies can be used to propagate infections, attack servers, send spam emails, perform various forms of fraud, and engage in other online crimes. BotNet size is an unpredictable quantity that can be modest or enormous. It depends on how complex and sophisticated the employed bots are. There are tens of thousands of zombies in a huge BotNet. A smaller BotNet, on the other hand, was made up of just a few thousand zombies.

The owner of the device that has been transformed into a zombie is oblivious of the fact that the impacted device and all of its functions are now being remotely operated, enslaved, and employed for bad intentions by one or more virus operators who are using Internet Relay Chat (IRC) as a crucial instrument for his illicit actions. In addition, the owner of the computer is uninformed of the fact that the affected system is now being used for malicious purposes. Numerous varieties of malware, malicious software, and programs have already and are still entangling the internet. While simpler types of bots lack such capabilities, large bots employ their internal spreaders to propagate infections [32].

Figure 2.4 depicts a brief illustration of an assault carried out by BotNets against a target. Botmaster is in charge of this attack and has full control over it.

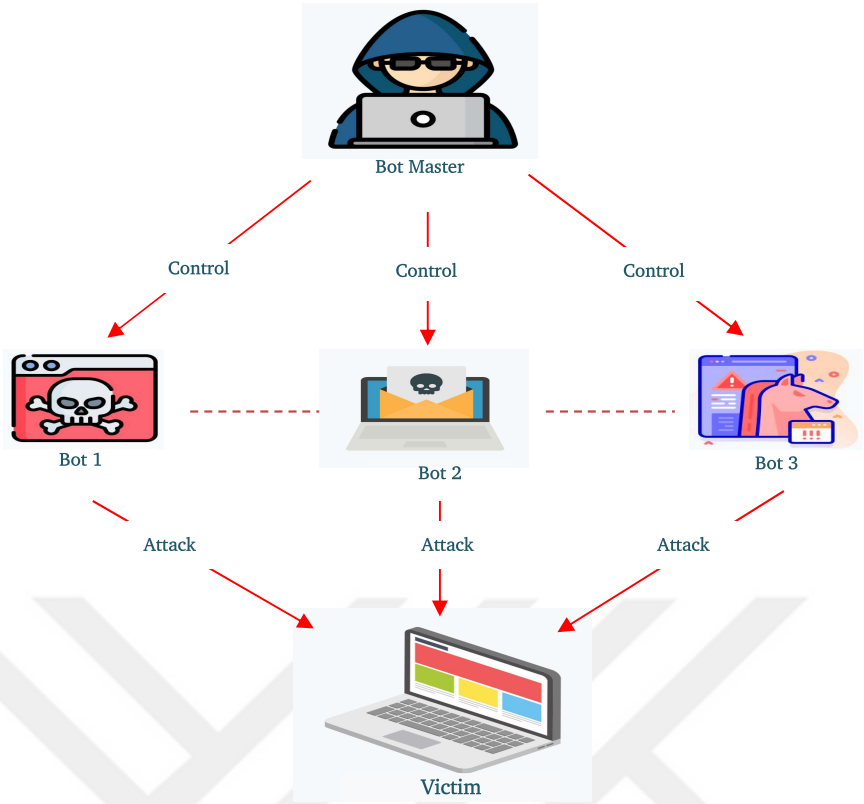


Figure 2.4 BotNets attack

2.6 BotNet Structure and Behavior

Botnets represent one of the most severe hazards that are now presented to the security of systems and the Internet of Things in this modern era of cloud computing and ubiquitous computing. BotNet attacks can leverage a wider variety of infection channels, thanks to recent developments in ubiquitous computing designs, which including connected mobile smartphones and the Internet of Things. Because networked devices and system platforms are changing so quickly, BotNets are always getting new kinds and new ways to attack [33]. As an example, from 2018 to 2019, the number of instances of the IIoT BotNet Mirai grew from about 143,000 to 225,000 [34].

In the 1980s, a known type of program that would later be known as BotNets first arose. In late 1993, a BotNet kit called Eggdrop, which was based on Internet Relay

Chat (IRC), was the first to establish the concept of BotNet development tools. The primary intent of the C Language-based Eggdrop was to facilitate the sharing of data and the coordinated action of several instances. In spite of the fact that the first BotNet was helpful and worked for a good purpose, the newer versions have mostly utilized for bad things since then [35].

A command and control server, or C&C server for short, is typically used to administer BotNets, which are networks of hosts that have been compromised by malware. The architecture of the command and control server makes it possible for damaging distributed assaults to be launched against compromised devices or other connected hosts over the Internet or a LAN. The command and control servers are referred to as botmaster, while infected devices are referred to as bot [36].

Both centralized and peer-to-peer or P2P architectural configurations are often used to categorize BotNets as shown in Figure 2.5. The way commands are delivered through the C&C channel defines these structures. Bots in centralized BotNets receive their instructions from a central C&C server. In a P2P net, on the other hand, the BotNet instructions spread through the P2P overlay network. Centralized botnets are normally further well-organized, but they are less resistant to detecting systems since the centralized C&C server serves as the BotNet's single potential failure point [37].

Botnets can be utilized for a diversity of distributed assaults, including piracy, extortion, the spread of malicious software, DDoS attacks and several more. Internet Relay Chat (IRC) was initially used to distribute BotNets, but today's attack vectors for BotNets are far more complex. Vulnerability attacks, compromised websites, file-sharing networks, and infected email attachments are some of these attack vectors. IIoT widespread devices are becoming increasingly common, giving BotNets a wider attack surface and many more vulnerable targets to infect. The availability of calculating and the connected internet have led to the creation and development of progressively complicated BotNets, as demonstrated by well-known BotNet attacks like Mirai and Zeus, making ongoing research in the area important [38].

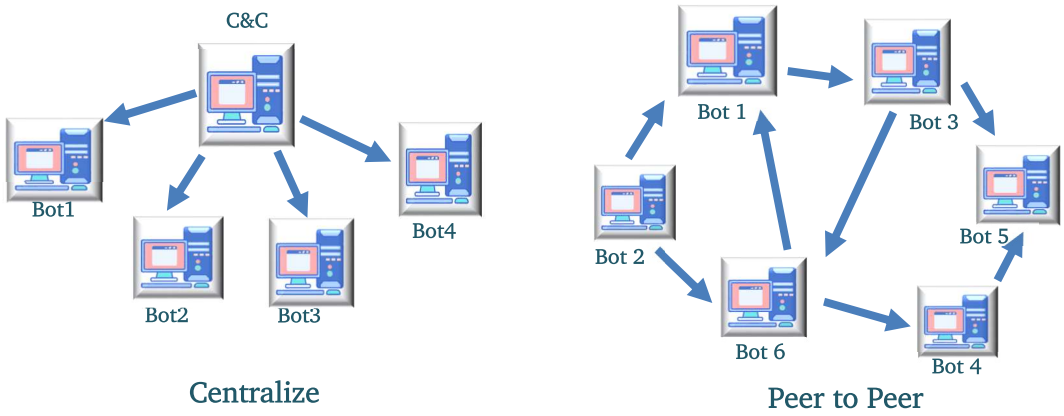


Figure 2.5 Cetralized vs. P2P Botnets

2.7 BotNets and IIoT

The network of compromised machines expands as the BotNet spreads to new IIoT devices, giving the BotNet greater computational power and the ability to launch more intense attacks. Additionally, because of the widespread use of IIoT in important industries and businesses, these entities are now more vulnerable to cybercrimes where foreign adversaries attempt to bypass security measures [39]. The threats associated with the potential takeover of IIoT devices are immense. Hacking risks include stealing private information, violating personal rights, and occasionally even carrying out cyberattacks that put people in danger of dying, such as sabotaging therapeutic gear. For IIoT systems used in Industry 4.0, this can mean manufacture and service disruptions, trade secret theft, and sensitive company data outflow, all of which could result in significant financial losses. IIoT attacks, particularly those launched by BotNets, have significantly increased in frequency in recent years. It becomes more and more challenging to protect the IIoT networks and equipment due to the wide variety of attacks that can be made on different protocols and devices [40].

2.8 Security of IIoT in Accordance to BotNet Detection

The technological age, in which IIoT plays a vital role, has profoundly impacted our lives. Moreover, the IIoT's phenomenal expansion is a major cause of numerous and serious cybersecurity vulnerabilities. Hence, there has recently been a lot of interest in both academic and industry in identifying and mitigating potential cyberattacks on IIoT networks. Establishing an IIoT BotNet, as previously stated, is a major attack; typically, enterprises use a variety of restrictions on unauthorized access, such as threat intelligence and intrusion detection, to identify and stop IIoT BotNets from operating. These strategies could be slightly useful, but they are incapable of detecting the emergence of zero-day IIoT BotNets with no identified patterns. This is why researchers and businesses alike are concentrating on methods for detecting BotNets in the IIoT. Typically, the goal is to identify the source of an attack and minimize the traffic it causes. Analyzing how BotNet structures arise in IIoT systems with the assistance of both business and academics should make it easier to improve security measures for spotting both known and emerging BotNets [41].

Research into the unique characteristics of IIoT BotNets is helping to improve defenses against these threats. In an effort to combat this widespread issue, the field of BotNet attacks detection is seeing progress, thanks to machine learning. It is generally agreed that BotNets present one of the greatest threats to IIoT networks. There are now a wide variety of solutions and services for intrusion detection on the market. These products and services offer varying degrees of security across IIoT devices. Recent developments in machine learning have shown some impressive performance in the recognition and classification of various types of attacks [42].

2.9 BotNet Detection Mechanisms

As the communication and infrastructure of the BotNet evolve, so do the detection methods. Studies have put out a wide range of architectures and strategies for BotNet detection through time. Both passive and active approaches can be used to detect botnets. Nevertheless, the usual taxonomy for detecting BotNets divides them into two

categories: honeynets (also known as honeypots) and intrusion detection systems (IDS) [43].

2.9.1 Honeynets

Since Honeynets only collect samples from BotNets and must be merged with other analysis tools like antivirus and sandbox, they are not extensively used in the particular bot detection system. Additionally, Honeynets are ineffective at spotting P2P as well as other decentralized BotNets. The majority of the time, Honeynets are helpful for analyzing BotNets and their characteristics [44].

2.9.2 Intrusion Detection Systems (IDS)

The operation of IIoT devices and the intrusion detection systems (IDS) are built on the same principles. The IIoT architecture can have an intrusion detection layer added on top of it or it can have it integrated further into the application and connection layer. The key idea underlying how intrusion detection functions is that it gives data nodes that emerge from a certain network a distinct identification. All data nodes with different identifiers that the system cannot figure out are denied, and the client is notified of any security breaches [45].

There are a wide variety of methods available for intrusion detection (as shown in Figure 2.6), which can identify any hostile actions carried out by IIoT systems. Industries and businesses utilize a variety of intrusion detection techniques, including detection system based upon signatures, anomaly, specifications, machine learning, deep learning, and a hybrid of methodologies. An approach that is based on signatures is utilized in order to detect the communication and information sessions that are sent across the connection layer. Those systems identify slightly anomaly in the information sessions sent over the network and send out an alarm centered on the information. It is an actual effective and quick approach to find system intrusion. This strategy makes use of the attack patterns that have been designated for it relying on the information that has been gathered historically. It performs a search in the history for something like the signatures of the data that contributed to the intrusion, and it also performs a

search in the future for the same types of intrusion signals and warns the user if they are found. This technique has the drawback that it cannot identify any further intrusions since the system would not distinguish the signature as one associated with an incursion [10].

Due to the fact that anomaly-based detection relies on irregular or anomalous data packets rather than signatures, it mitigates the disadvantage of the signature-based detection method. An anomaly will be identified in every new data session that attempts to access the system but does not fit the systematic properties. The system will be more secure as a result, but users must maintain consistency in their data to prevent user data from becoming corrupted.

The instructions that are available to the system and that the data packs of the system will obey are the foundation of the approach to intrusion detection that is known as specification-based intrusion detection. This collection of instructions considers every data packet to be an anomaly if it does not fit with the instructions in its entirety [46].

Within the scope of the detection of intrusions, machine learning is used in a diversity of ways. The system is designed to pick up new information from previous attacks. Machine learning applications make this possible. The system quickly stops any similar intrusion and notifies the user if it occurs again on the machine. Deep learning is not the same as traditional machine learning since machine learning employs just one algorithm to teach a computer from its past experiences, whereas deep learning employs multiple algorithms. Contrarily, DL is a branch of ML and is made up of numerous strands of algorithms known as ANNs (artificial neural networks) [47].

Combining the abovementioned intrusion detection techniques results in a more sophisticated intrusion detection system. Each technique has unique advantages, thus a combination of techniques can be employed to shield the system against numerous cyberattacks. Compared to separate approaches, this strategy offers stronger protection [15].

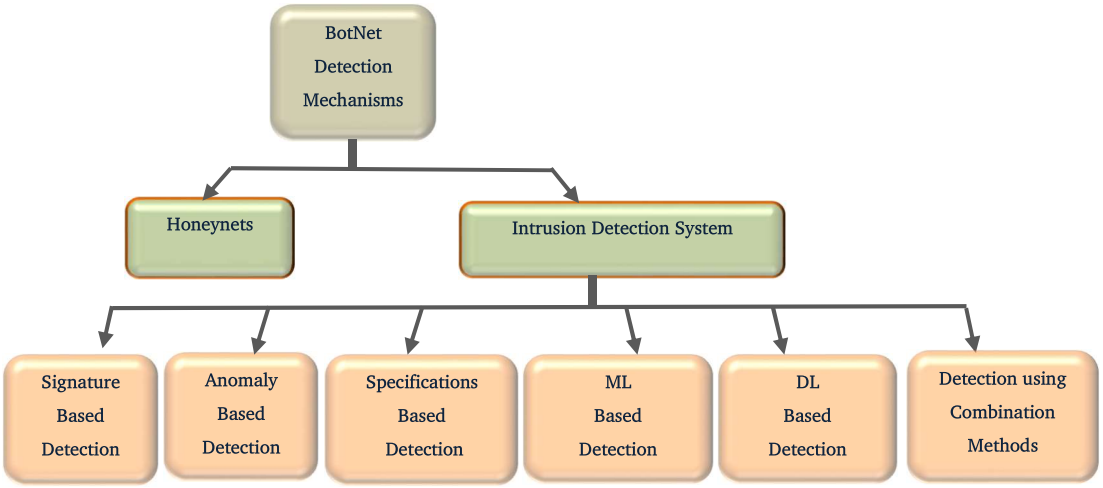


Figure 2.6 Methods of intrusion detection

2.10 Machine Learning (ML)

The comprehensive research of statistics, algorithms, analysis, computations etc., which are employed for systems to complete a certain job without even being programmed is referred to as machine learning (ML). Numerous popular programs exists that employ learning algorithms. One of the reasons a search engine like Google is so effective is because of a learning algorithm that has been taught how to categorize web sites. Such algorithms are put to use for a comprehensive variability of purposes, such as predictive analytics, data mining, image processing, and so forth. The key advantage of utilizing machine learning is that when an algorithm understands what it should do with the data, it can carry out those actions on its own without human intervention [48].

A mathematical model is constructed using the sample data through ML, referred to as "training data," as a means of making inferences or decisions without being explicitly instructed to do so by the programmer. One of the presumed characteristics of machine learning when compared to humans is similar prediction and making decisions capabilities. A distinctive "learning machine" finds rules, which whenever applied to a set of inputs, provide the intended result. These rules will also produce the correct outcome for the majority of additional inputs (in addition to the training data), provided that such data originated from the identical or a related statistical range as the training data [49].

In a conclusion, the machine learning algorithms make huge improvements to ordinary systems in learning and prediction aspects, which leads to improved results as shown in Figure 2.7.

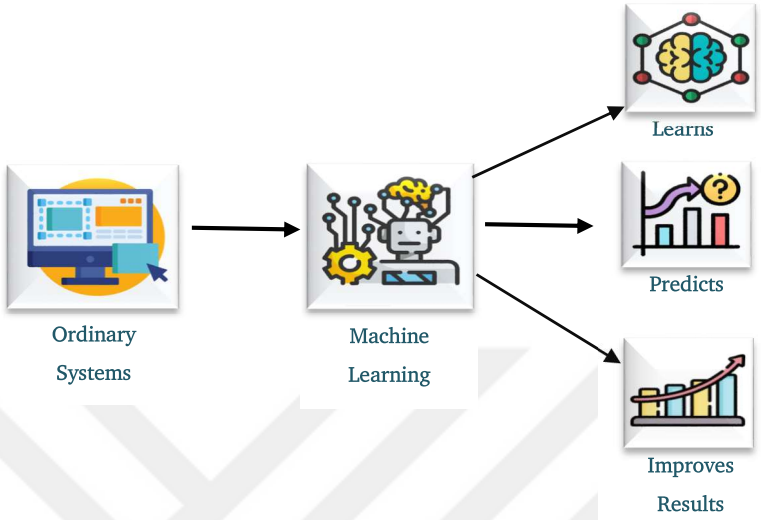


Figure 2.7 Improvement of ordinary systems using ML

2.10.1 Types of Machine Learning

ML covers a broad extensive variety of subject areas and is inspired by artificial intelligence (AI). The field is focused on acquiring skills or knowledge via practical application. Typically, this includes extracting relevant topics from previously gathered data. As a result, machine learning encompasses a vast diversity of learning, ranging from whole academic fields to specific approaches. As indicated in Figure 2.8, machine learning has been divided into four categories due to its complexity: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. All of them take a particular aim and method of operation that results in different forms of data. Supervised learning accounts for more than 70% of learning algorithms, while unsupervised learning accounts for 10% to 20%. The remaining space is used by semi-supervised and reinforcement learning [50].

Supervised learning methods are applied to data sets that contain a collection of inputs and their output data. Through training on both the inputs and the targets, the machine acquires the knowledge necessary to link those inputs to the proper outputs. The model is trained until it reaches the appropriate accuracy level based on that training data. The model's validity is determined by testing it with previously unseen data (test set). Support Vector Machines, Decision Tree and Tree Ensembles (including Gradient Boosting methods, Random Forest etc.), and Artificial Neural Networks (for instance Multi Layer Perceptron) are a few examples of supervised learning algorithms.

Unsupervised learning aims to "make sense" of the data because there isn't a target or label variables to predict. Clustering data populations for certain interventions is a common application of this type. Hierarchical clustering, Density-Based Spatial Clustering of Applications with Noise, and K-means are examples of unsupervised learning techniques.

Semi-supervised learning is a method of machine learning that, during training, combines a large volume of unlabeled data alongside a very small amount of data that has been labeled. Learning in a semi-supervised environment is a middle ground between learning in an unsupervised environment and learning in a supervised environment. The categorization of web material, the classification of text data, and the development of algorithms for speech recognition are all examples of semi-supervised learning.

The machine can be taught to make accurate decisions through the process of reinforcement learning. A situation with an agent is introduced in which it must take activities to increase the accumulative benefit (typically called return in episodic complications) or average benefit (in ongoing problems) that involves a significant number of steps in advance. The agent learns from previous experiences and attempts to gather as much information as possible in order to make good decisions. Reinforcement learning methods include the Deep Reinforcement Learning and Markov Decision Process [48].

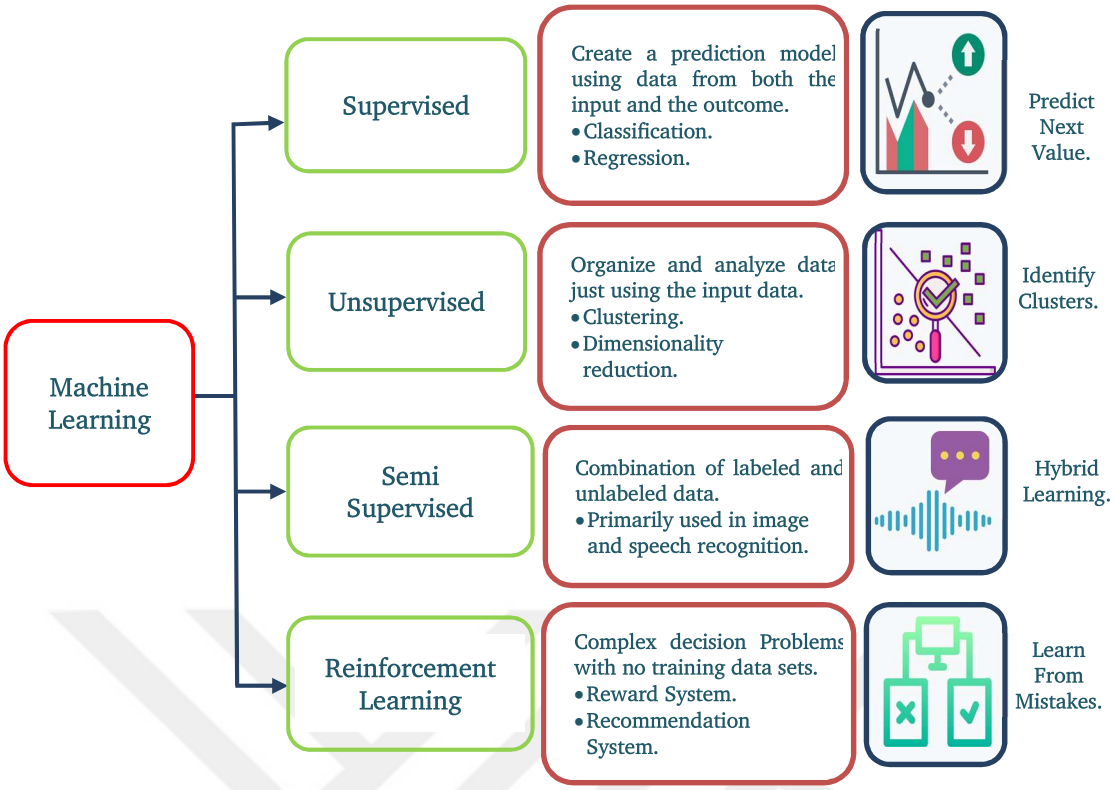


Figure 2.8 Categories of ML

2.10.2 Popular Machine Learning Algorithms

In the next section, two of the machine learning algorithms, which utilize the majority of the time will be investigated. These algorithms are capable of being applied in order to locate a solution to the multi classification issue.

2.10.2.1 Random Forest Algorithm

It was invented by Leo Breiman [51]. The term "Random Forest" refers to a well-known method of supervised machine learning, which is used for problems involving classification and regression. It does this by constructing decision trees based on the many samples that it has, and then using their average in classification and their majority vote in regression. The versatility of the Random Forest Algorithm to cope with data sets that comprise both continuous variables and discrete variables is one of the most essential elements of this algorithm, such as data used in regression, as well

as categorical variables, as used in classification. It offers superior outcomes when it comes to classification concerns that may arise. The following important characteristics may be observed in the Random Forest model:

1. Diversity: Due to the fact that every tree is one of a kind, not all characteristics, factors, or features are taken into consideration while building any given tree.
2. Immune to the curse of dimensionality: The feature representation is being constrained as a result of the fact that every tree doesn't really take into consideration all of the features.
3. Parallelization: Every branch of the tree is constructed independently using a variety of data and attributes. This suggests that it is feasible to construct random forests by making the most of the capabilities of the central processing unit (CPU).
4. Train-Test split: Owing to the fact that there will consistently be thirty percent of the data that is concealed from the decision tree when employing a random forest, it is not necessary to divide the data in half for the purposes of training and testing when using this method.
5. Consistency: The majority vote/average ensures that the outcome is consistent. Random Forests employs a classification strategy known as an ensemble approach in order to provide the desired outcome.

Several decision trees typically trained utilizing the data from the training session. When splitting the nodes, this dataset contains the observations and attributes that will be picked at random. In a system modeled like a rain forest, many decision trees are utilized. Every decision tree has a node that serves as the root, nodes that serve as leaf nodes, and decision nodes. The final outcome that was determined by a certain decision tree is shown by the "leaf node" at the end of each tree. The ultimate decision is made by majority vote. In this particular instance, the ultimate output of the rain forest system is the option that was chosen by the vast majority of the decision trees. [52]. For more details on Random Forest algorithm, please see [51].

2.10.2.2 K-Nearest Neighbor (KNN) Algorithm

As stated by [53], the KNN algorithm was invented by E. Fix and J.L. Hodges in 1951 as Nearest Neighbor algorithm (NN). It was evolved by T. Cover, and P. Hart [54] to the K-Nearest Neighbor algorithm that is known today. The KNN method makes the assumption that there is a resemblance among the new case or data and the existing cases, and then places the new case through into categorization that is the most comparable to the categories that are already there. It saves all of the data that is currently accessible and assigns a category to a new point of information depending on how similarly it is structured. This indicates that the KNN technique may be utilized to quickly and simply place newly discovered data into a category that is a good fit for it whenever this occurs. During the training phase, the KNN algorithm does nothing more than store the dataset. When it receives new data, it then categorizes current data together into group that is significantly similar to the new data [55]. For more details on KNN algorithm, please see [53] and [54].

2.11 Datasets of Widespread Use in ML for Intrusion Detection

Numerous research organizations are currently compiling a vast number of different data sets, not only for the goals of their own research projects, but also with the intention of supplying information to online libraries. Explained below are some of the most common types of intrusion detection datasets that have been utilized in ML data analysis and in this thesis.

2.11.1 DARPA 98 Dataset

DARPA 1998 gathered together and dispersed the first standard data used in the assessment of systems for the detection of intrusions into computer networks by MIT Lincoln Lab according to the budget allocation of "Defense Advanced Research Projects Agency" (DARPA) and "Air Force Research Lab" (AFRL). Since the DARPA dataset consists of fresh files, scientists must mine properties from those files in order to use them in ML methods. The first benchmark dataset for assessing computer intrusion detection systems was developed as part of the 1998 DARPA intrusion detection

assessment. This dataset was created to assess the rates of false alarms and detection rates of systems that detect intrusions utilizing a variety of both well-known and novel attacks concealed in a large volume of regular background traffic. The dataset was gathered using a simulated network that was utilized to automatically produce realistic traffic, which included attempted attacks. The attacks were created to be utilized in the 1998 DARPA study of intrusion detection. The 9 weeks of data that were gathered for the study included information on over 300 attacks in total. These 300 attacks were selected from 32 multiple attack kinds and 7 various attack conditions [56].

2.11.2 KDD CUP 99 Dataset

The KDD Cup 1999 dataset was employed in DARPA's IDS assessment platform, which was carried out by the agency. The data consists of 4 GB of compacted TCP spew packets that was generated by seven weeks' worth of network activity. It entails of around 4,900,000 single connection vectors, every single one has 41 attributes. Each vector is either categorized as an attack or normal (there are 22 defined types of attacks). An online repository dataset called Knowledge Discovery in Databases (KDD) contains data on all different forms of intrusion attempts, including DOS, R2L, U2R, and PROBE. There are 4,898,431 data points in the original KDD dataset. A network session among two hosts is represented by each data point. Each vector includes a label attribute that indicates whether the vector is normal or malicious. There are a total of 2952839 data points, of which 2952781 are attack types and 972781 are just normal data [57].

2.11.3 NSL-KDD Dataset

For the purpose of resolving the essential glitches with the KDD-99 dataset, it was decided to generate the NSL-KDD dataset. The KDD-99 dataset has been revised and modified to provide this new dataset. Three major problems have been resolved in the NSL-KDD dataset. Firstly, replica data in the training and test sets have now been decreased to prevent classification algorithms from being biased toward the most redundant entries. Secondly, in order to get accurate results while simultaneously using classification algorithms, the training and test sets were produced by selecting multiple

entries from a pool of candidates from different areas of the regular KDD-99 dataset. Finally, the imbalance between the quantity of testing and the amount of training has indeed been addressed for a reduction. The structure of the NSL-KDD dataset is nearly identical to that of the KDD-99 dataset (specifically, it features 41 different types of fields and 22 different types of attack or traffic patterns) [58].

2.11.4 UNSW-NB15 Dataset

2015 marked the year when the UNSW-NB15 dataset on the security of computer networks was made available to the public. This dataset comprises 2,540,044 real regular and anomalous (attack) network actions from the contemporary period. The untreated network elements were analyzed, and 49 features were derived from them. These features included flow based as well as packet based aspects. The classifications of attacks consist of Exploits Analysis, Generic, Backdoor, DoS, Reconnaissance, Fuzzers, Worms, and Shellcode [59].

2.11.5 CSE-CIC-IDS2018 Dataset

CSE-CIC-IDS2018 was designed for the purpose of training prediction models for use in network-based detection of intrusion. Instead of serving as a repository for signature-based detection systems, this dataset aims to advance investigation into detection based on anomalies using a variety of ML techniques. CSE-CIC-IDS2018 holds approximately 16,000,000 occurrences gathered during a ten-day period. A network with a PCs, gateway, servers, switch, and routers has been set up to cover all of the network configuration requirements. The attack diversity criteria are being supported by the proposal of a whole spectrum of attacks on various circumstances. Naturally, possessing all created network traffic, system logs, and resource utilization data from tests, such as memory and CPU usage, shows support for the heterogeneity criteria [60].

2.11.6 N-BaIoT Dataset

The N-BaIoT dataset is made up of data samples and has 115 different features. The dataset was gathered using IoT device port mirroring. To guarantee that the data was benign, the benign data was taken immediately after the network was established. The

data was gathered by injecting two kinds of attacks into several IoT devices. Different Bashlite and Mirai malwares were injected to create each dataset. Bashlite, commonly known as gafgyt, infects IoT devices running Linux to carry out DDoS assaults. There were several different flooding assaults, including UDP and TCP attacks. Using IoT devices, Mirai is utilized for massive attacks [61].

2.11.7 Bot-IoT Dataset

In the Cyber Range Lab at the University of New South Wales in Canberra, a heterogeneous network infrastructure was created in order to establish the BoT-IoT dataset. On the network, there was a combination of typical traffic and traffic from botnets. There is a diverse selection of formatting options available for the dataset's source files. In order to facilitate the labeling process and make it more manageable, the files were segmented according to the both the attack group and the attack subgroup. The pcap files that were acquired have a total size of 69.3 gigabytes, and they contain more than 72 million records. The flow of traffic that was extracted is 16.7 gigabytes in size when saved in CSV format. DoS and DDoS attacks, as well as Keylogging, Operating System and Service Scan, and Data Exfiltration assaults, are all accounted for in this dataset. Attacks can be further divided into Dos and DDoS categories based on the protocol that was used [62].

2.11.8 X-IIoTID Dataset

It should be noted that this dataset has been explained in more detail because it is used to select, test, and train features in this thesis. Muna Al-Hawawreh et al. [17] at the New South Wales University, also known as UNSW, located in the capital city of Canberra established the X-IIotID dataset. It is a connection and device-agnostic dataset that can be used to study and train systems for detecting intrusions that are powered by machine learning and deep learning for IoT and IIoT devices. IIRA, which stands for the Industrial Internet Reference Architecture framework serves as the foundation for the laboratory's design. The design employed is separated into three points: the platform, the edge, and the enterprise, where numerous industrial and attack machines, protocols, cloud services, and IoT devices are installed. Three distinct frameworks, notably CKC, MALC, and ATT&CK, were used to create the malicious records. The following attack kinds are included in the simulation of the attack process

[63]: Reconnaissance, Weaponization, Exploitation, Lateral Movement, Command and Control, Exfiltration, Tampering, Crypto-Ransomware, and Ransom DoS.

The final set of extracted network traffic features and the attributes that are associated with the edge gateway's resources is illustrated in Table 2.1, which is taken from [17].

Table 2.1 Features of the X-IIoTID dataset [17]

Feature Name	Type	Description	Feature Name	Type	Description
Date	Discrete	Datestamp	Std user time	Continuous	Standard deviation of user time in the last 10 seconds˜
Ts	Discrete	Timestamp	Avg nice time	Continuous	Average of nice time (the time used for defining the priority of process) in the last 10 seconds
Scr IP	Discrete	Source endpoint's IP address	Std nice time	Continuous	Standard deviation of nice time in the last 10 seconds
Des IP	Discrete	Destination endpoints' IP address	Avg system time	Continuous	Average of system time (time the processor works at operating system functions) in the last 10 seconds
Scr Port	Discrete	Source endpoint's TCP/UDP port, or ICMP code	Std system time	Continuous	Standard deviation of system time in the last 10 seconds
Des Port	Discrete	Destination endpoint's TCP/UDP, or ICMP code	Avg IO wait time	Continuous	Average of I/O wait time (total time of the CPU is idle waiting for I/O operation) in the last 10 seconds
Protocol	Discrete	Protocols (e.g, TCP, UDP, ICMP, or other)	Std IO wait time	Continuous	Standard deviation of I/O wait time in the last 10 seconds
Service	Discrete	Application protocol running in Destination port	Avg idle time	Continuous	Average of idle time (total time of the CPU is not busy and does ˜not have an outstanding disk I/O requests in the last 10 seconds
Duration	Continuous	The time difference between the last packet and the first packet seen	Std idle time	Continuous	Standard deviation of idle time in the last 10 seconds
Scr bytes	Continuous	Number of bytes from source to destination	Avg tps	Continuous	Average of the number of transfer requests per second issued to ˜the device in last 10 seconds
Des bytes	Continuous	Number of bytes from destination to source	Std tps	Continuous	Standard deviation of the number of transfer transaction per second issued to the device in last 10 seconds
Missed byte	Continuous	Number of missing bytes in content gaps	Avg rtps	Continuous	Average of the number of read transaction per second issued to the device in last 10 seconds
Scr pkts	Continuous	Number of sending packet	Std rtps	Continuous	Standard deviation of the number of read transaction per second issued to the˜device in last 10 seconds
Des pkts	Continuous	Number of received packet	Avg wtps	Continuous	Average of the number of write transaction per second issued to the device in last 10 seconds
Scr IP bytes	Continuous	Number of sending bytes in the IP header total length field	Std wtps	Continuous	Standard deviation of the number of write transaction per second issued to the device in last 10 seconds
Des IP bytes	Continuous	Number of received bytes in the IP header total length field	Avg ldavg 1	Continuous	Average of average system load during the last minute in window time size 10 seconds

Table 2.1 Features of the X-IIoTID dataset [17] (Continue)

Conn state	Discrete	Connection status (1 complete, 2 Rest, 3 partial)	Std ldavg 1	Continuous	Standard deviation of average system load during the last minute in window time size 10 seconds
Total bytes	Continuous	Total number of bytes exchanged between source and destination	Avg Kbmemused	Continuous	Average of used memory in kilobytes in last 10 seconds
Byte rate	Continuous	Total number of bytes per second	Std Kbmemused	Continuous	Standard deviation of used memory in kilobytes in last 10 seconds
Total Pkts	Continuous	Total number of packet exchanged between source and destination	Avg num proc/s	Continuous	Average of number of tasks created per second in last 10 seconds
Pkts rate	Continuous	Total number of packet per second	Std num proc/s	Continuous	Standard deviation of number of tasks created per second in last 10 Seconds
orig bytes ratio	Continuous	Percentage of sending bytes to the total bytes	Avg num swch/s	Continuous	Average of number of context switches per second in last 10 seconds
resp bytes ratio	Continuous	Percentage of receiving bytes to the total bytes	Std num swch/s	Continuous	Standard deviation of number of context switches per second in last 10 seconds
orig packts ratio	Continuous	Percentage of sending packets to the total packets	Anomaly Alert	Discrete	1 if the connection has alert from Zeek; 0 otherwise
resp pkts ratio	Continuous	Percentage of receiving packets to source IP packets	OSSEC alert	Discrete	1 if the connection has alert from OSSEC; 0 otherwise
SYN	Discrete	If connection has packet with SYN flag (1 or 0)	Alert level	Discrete	OSSEC alert severity level
SYN-ACK	Discrete	If connection has packet with SYN-ACK flag (1 or 0)	R W physical	Discrete	1 if there is a read or write activity to the physical process; 0 otherwise
Pure ACK	Discrete	If connection has packet with pure ACK flag (1 or 0)	File act	Discrete	1 if there is a file activities (read, write,delete, copy,create and download); 0 otherwise
Packet with payload	Discrete	If connection has packet with payload (1 or 0)	Proc act	Discrete	1 if there is a new process is executed or started; 0 otherwise
FIN or RST	Discrete	If connection has packet with FIN flag or RST (1 or 0)	Is privileged	Discrete	1 if the performed activity (login, process or file activity) is privileged ; 0 otherwise
Bad checksum	Discrete	If connection has packet with bad checksum (1 or 0)	Login attmp	Discrete	1 if there is attempt to login; 0 otherwise
SYN with RST	Discrete	If connection has packet with both SYN and RST flags (yes or No)	Succ login	Discrete	1 if a successful login ; 0 otherwise
Avg user time	Continuous	Average of user time (the time of process runs programs/codes) in the last 10 seconds			

The dataset includes 68 features, which contain 421,417 benign records and 399,417 hostile records. There is a subcategory for attacks as well as a sub-subcategory. The scattering of attack types in the X-IIoTID dataset is presented in Table 2.2, which is taken from [17].

Table 2.2 The X-IIoTID dataset's distribution of attack types [17]

Attack	Total number of instances
Reconnaissance	127590
Weaponization	67260
Exploitation	1133
Lateral Movement	31596
Command &Control	2863
Exfiltration	22134
Tampering	5122
Crypto Ransomware	458
RdoS	141261
Normal	421417

Table 2.3 (also taken from [17]) compares the results of an evaluation of the strengths and weaknesses of seven current datasets with the X-IIoTID dataset. The X-IIoTID represents the first intrusion dataset designed particularly for IIoT systems. It is a clearly defined and deliberately created dataset that mirrors recent alterations and diversity in system events and network traffic configuration created by several and diverse IIoT devices, connection protocols, and patterns of communication. It provides unique insight into the threats and cyberattacks on IIoT networks and systems, as well as innovative and comprehensive features [17]. For these reasons, this dataset is going to be utilized in this thesis for the purpose of features selection, testing, and training.

Table 2.3 Comparison between X-IIoTID with popular existing datasets [17]

Dataset	CNSC	HDS						RNT	DAT	DDD	FS	IIoT-Cp			RA	AF	IIoT-T	LD	MD	PA
		NT	HR	L	PP	AL	CC					MQTT	CoAP	WS						
DARPA 98[56]	YES	YES	NO	YES	NO	NO	YES	NO	YES	N/A	YES	NO	NO	NO	NO	YES	NO	YES	NO	YES
KDD CUP 99 [57]	YES	YES	NO	YES	NO	NO	YES	NO	YES	N/A	YES	NO	NO	NO	NO	YES	NO	YES	NO	YES
NSL-KDD [58]	YES	YES	NO	YES	NO	NO	YES	NO	YES	N/A	YES	NO	NO	NO	NO	YES	NO	YES	NO	YES
UNSW-NB15 [59]	YES	YES	NO	NO	NO	NO	NO	NO	YES	NO	YES	NO	NO	NO	NO	NO	NO	YES	YES	YES
CICIDS [60]	YES	YES	YES	NO	NO	NO	YES	YES	YES	NO	YES	NO	NO	NO	NO	NO	NO	YES	YES	NO
N-BaIoT [61]	NO	YES	NO	NO	NO	NO	YES	YES	NO	YES	YES	NO	NO	NO	YES	YES	NO	YES	YES	YES
BoT-IoT [62]	NO	YES	NO	NO	NO	NO	YES	NO	NO	N/A	YES	YES	NO	NO	YES	YES	NO	YES	YES	YES
X-IIoTID[17]	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES

CNSC: Complete Network and System Configuration
HDS:˜Heterogeneous Data Sources
NT: Network Traffic
HR: Host Resources
L:Logs
PP: Physical Process
AL: Alerts
CC: Complete Capture
RNT: Realistic Network Traffic
DAT:˜Divers Attacks Scenario and Types
DDD: Diverse Data Duration
FS: Feature Set
IIoT-CP: IIoT Connectivity Protocols
RA: Recent Attacks
AF: Agnostic-Features
IIoT-T: IIoT Traces
LD: Labeled Dataset
MD: Metadata˜
PA: Public Availability

Table 2.4 presents a concise summary of the previously listed datasets as well as the recent research that made use of those datasets.

Table 2.4 Datasets and recent literature employing them

Dataset Name	Article	Year	No. Of Features	No. Of Records	Link	Recent Works
DARPA98	[56]	1998	41	4,900,000	"https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset"	• Caleb Belth et al. (2020) [64]. • Yen-Yu Chang (2021) [65]. • Siddharth Bhatia et al. (2021) [66].
KDD CUP99	[57]	1999	41	4,898,431	"http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html"	• Satish Kumar et al. (2022) [67]. • Santosh Kumar Sahu et al. (2022) [68]. • Amir Javadpour et al. (2022) [69].
NSL-KDD	[58]	2009	41	148,517	"https://www.unb.ca/cic/datasets/nsl.html"	• Abhishek Raghuvanshi et al. (2022) [70]. • Ilhan Firat Kilincer et al. (2022) [71]. • Souradip Roy et al. (2022) [72].
UNSW-NB15	[59]	2015	49	2,540,044	"https://research.unsw.edu.au/projects/unsw-nb15-dataset"	• P. G. V. Suresh Kumar et al. (2022) [73]. • Mohammad Humayun Kabir et al. (2022) [74]. • Moutaz Alazab et al. (2022) [75].
CSE-CIC-IDS2018	[60]	2018	80	16,000,000	"https://www.unb.ca/cic/datasets/ids-2018.html"	• Abdulnaser A. Hagar et al. (2022) [76]. • Mário Antunes et al. (2022) [77]. • Mohamed Hammad et al. (2022) [78].
N-BaIoT	[61]	2018	115	7,062,606	"https://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT"	• Sawssen Bacha et al. (2022) [79]. • Jabed Al Faysal et al. (2022) [80]. • Badr Lahasan Le et al. (2022) [81].
BoT-IoT	[62]	2019	46	73,000,000	"https://research.unsw.edu.au/projects/bot-iot-dataset"	• Tanzila Saba et al. (2022) [82]. • Cristiano Antonio de Souza et al. (2022) [83]. • Babita Majhi et al. (2022) [84].
X-IIoTID	[17]	2021	68	820,834	"https://ieee-dataport.org/documents/x-iiotid-connectivity-and-device-agnostic-intrusion-dataset-industrial-internet-things"	• Muna Al-Hawawreh et al. (2021) [85]. • Thi-Thu-Huong Le et al. (2022) [86]. • Rehab Alanazi et al. (2022) [87].

2.12 Features Selection (FS)

A new age of data analysis has begun, thanks to the development of knowledge such as the Internet of Things (IoT) as well as the Industrial Internet of Things (IIoT). In this new era, a gigantic amount of data is generated extremely quickly on a daily basis with great dimensional data in a diversity of fields, including business intelligence, medical services, social networks, transportation, online learning, government, advertising, and the financial system, amongst others. Decision-makers find considerable value and significance in the insights and knowledge that these large-scale datasets contain. In the field of machine learning, getting these kinds of insights and knowledge is one of the hardest things to do [88]. Datasets for modern BotNet identification generally include a lot of redundant and unnecessary information. The effectiveness of data analysis tools might be decreased by redundant and irrelevant features, leading to unintelligible results. Therefore, reducing the dimensionality of such datasets is the very first stage in any BotNet detection scheme, especially if the dataset is large and contains a lot of features. Features Selection (FS) is one of the most popular ways to reduce the extraordinary dimensionality of large datasets. It works by choosing a slight subset of relevant and important properties (features) and removing irrelevant and repetitive features with the intention of build good expectation models [89].

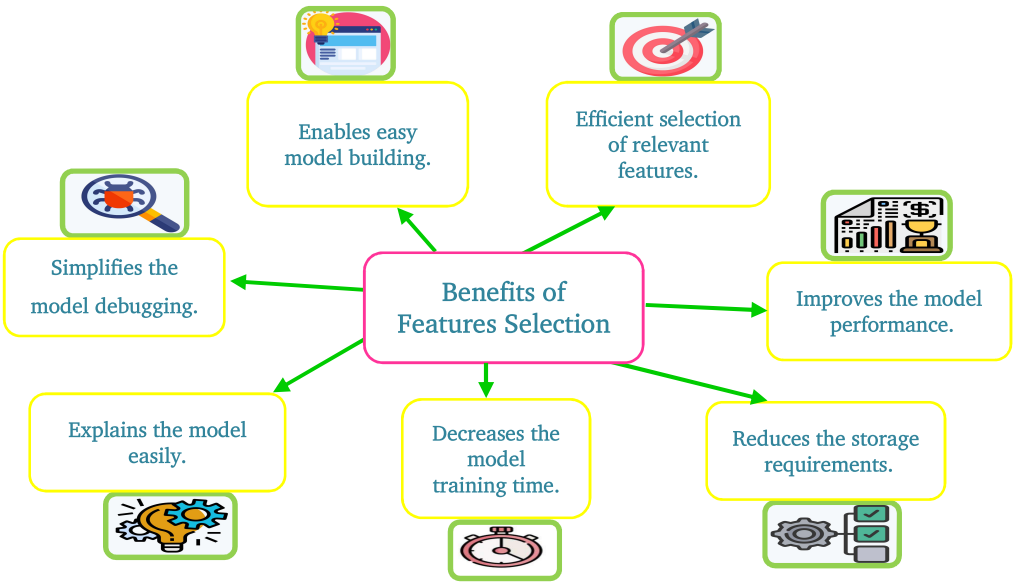


Figure 2.9 Benefits of features selection

In data processing based on machine learning approaches, features selection (FS) is a vital step. It is a significant stage in the knowledge discovery process. The large dimensionality, noisy, and irrelevant aspects of the data that are frequently encountered in this procedure cause an exponential rise in memory and processing time requirements. FS performs an essential function in rising an efficient and effective model. It similarly aids in lowering the number of features evaluated while developing learning models. Many advantages are offered by FS techniques, including fewer storage needs, improved prediction accuracy, effective learning performance, and better comprehension. The main concept of features selection is demonstrated in Figure 2.10. As a result, the process of learning is accelerated while at the same time requiring less space in the learner's memory. In most cases, the FS method is separated into four stages [80]:

- Generating the subset: A certain searching approach is utilized in order to arrive at a conclusion regarding a group of features.
- Evaluating the subset: A set of evaluation criteria is applied to the candidate subset that was chosen for further consideration.
- Stopping criterion: From among all of the candidates that have been reviewed after they have met the stopping condition, the set of features that is chosen to be the best fit for the evaluating criterion is chosen.
- Validating the results: Validation of the selected subset can be done by means of domain knowledge or through the use of a test set.

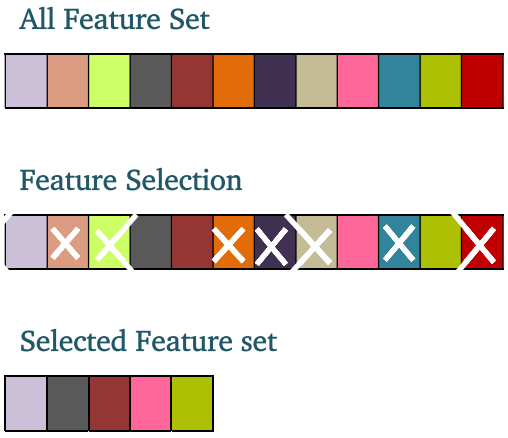


Figure 2.10 Concept of features selection

The selection of features can be accomplished by one of these three standard mechanisms or methodologies:

2.12.1 Filter Features Selection

Filter methods are strategies for feature ranking that assess the usefulness of features by considering the inherently meaningful characteristics of the data, regardless of the classification model. The variables are ranked according to an appropriate ranking criterion, and any variable that scores below the threshold is eliminated. Filter approaches use metrics including distance, information, correlation, and consistency to evaluate how relevant certain features are. The benefits of filter approaches are their speed, scalability, and lack of dependence on a learning algorithm. The evaluation of various classifiers can therefore proceed after only one feature selection procedure is required. The problem with filter approaches is that they don't interact with the classifier. This means that the results are less accurate and cover a wider range [90] [91]. The process of filter features selection is presented in Figure 2.11.

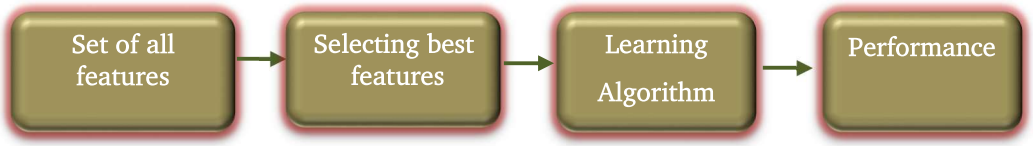


Figure 2.11 The process of filter features selection

2.12.2 Wrapper Features Selection

Wrapper techniques get their name from the fact that they encase a classifier within a feature selection course. Usually, a collection of features is picked, their effectiveness is measured, the original set is changed in some way, and the new set's effectiveness is also measured. In order to assess the variable subset, the predictor is used as a "black box" and its performance serves as the objective function. Different feature subsets are generated and assessed as part of a search strategy that is defined in the space of potential feature subsets. This method is suited to a certain classification algorithm

because it uses a particular classification model that has been trained and tested to evaluate a particular subset of features. This method has the advantages of being able to account for feature dependencies, as well as interaction across feature subset search and model selection [92].

It frequently suffers from the disadvantages of being computationally costly and having a greater risk of overfitting than filter approaches, particularly if the constructing classifier does have a significant computational cost. Overfitting occurs when a classification algorithm learns the data sufficiently well and has limited generalization capabilities. Another issue with this method is that the feature space is huge, and considering every conceivable combination would require a significant amount of time and calculation. To identify the best combinations of features, various heuristic search techniques must be developed [93]. The process of wrapper features selection is presented in Figure 2.12.

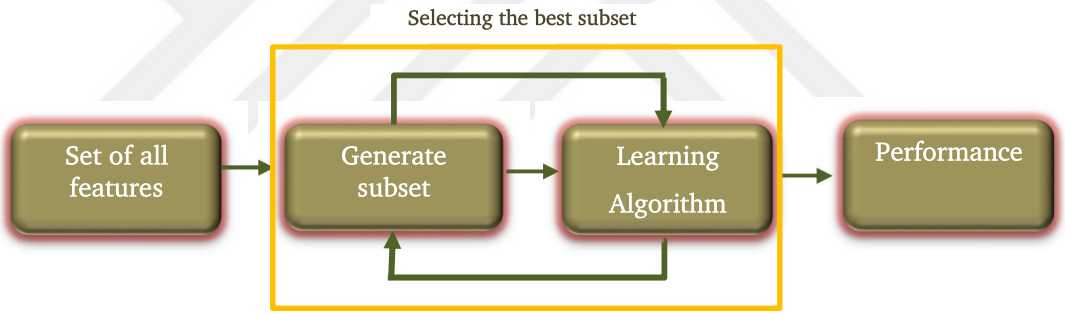


Figure 2.12 The process of wrapper features selection

2.12.3 Embedded Features Selection

Embedded techniques engage alongside algorithms of learning at a reduced processing price when compared to the wrapper approach. It preserves feature dependencies and takes into account not only relationships between input features and output features, but it also looks locally for features that enable for better local classification. It employs independent criteria to determine the best subset for a given cardinality. The learning method is employed to choose the final optimal subset from among the optimal subsets with varying cardinality [94]. In addition to being significantly less computationally intensive than wrapper techniques, this approach

offers the advantage of incorporating the interactions with the classification model [95] [96]. The process of embedded features selection is presented in Figure 2.13.

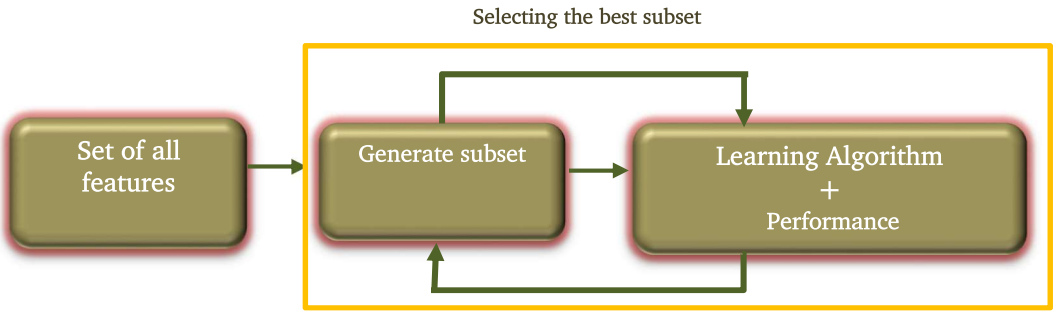


Figure 2.13 The process of embedded features selection

2.13 FS using the Eurasian Oystercatcher Optimizer (EOO)

The Eurasian Oystercatcher Optimizer (EOO), a new optimization technique inspired by nature, was introduced by Ahmad Salim et al. [18]. When looking for mussels, the EOO algorithm imitates the eating habits of Eurasian Oystercatchers (EO) (shown in Figure 2.14 [18]). Every bird (solution) in the community performs the role of a search agent in EOO. To ultimately consume the best mussel, the EO switches the candidate mussel based on the best solution (optimal result). The ratio of mussel size, calories, and energy must be balanced. Meire and Ervynck [97] came up with the caloric maximization hypothesis, which gave rise to EOO swarm algorithms.

Figure 2.14 The Eurasian Oystercatcher [18]

2.13.1 Mathematical Model of the EOO

The primary goal of EOs is to maintain a balance between their own energy as well as the calories they obtain from the mussels. Mussels' size and energy and calories have

been tightly correlated. As the length of the mussels grows, so does the number of calories they contain and the amount of time it takes to open them. Thus, EO waste must have significant energy. The behavior of EO during the search process is illustrated in Equations 2.1 and 2.2 [18]:

$$Y = T + E + L * r * (X_{best} + X_{i-1}) \tag{2.1}$$

$$X_i = X_{i-1} + Y * C \tag{2.2}$$

where *Xi* is a candidate mussel's position, *L* is the mussel's length, and it is a random number between 3 and 5, which represents the range of the ideal mussel length. *Y* stands for the final energy of EO for each solution (iteration). *T* is the amount of time required to crack open the current mussel (solution), and its amount is dependent on *L*, according to Equation 2.3. Using the relevant mussel's size as specified in [18], the values (3 and 5) were utilized in Equation 2.3. The present energy of the EO, denoted by the symbol *E*, is determined by Equation 2.4 [18], and depends on the value of the iteration because the energy of the EO diminishes over time. In order to increase unpredictability and uncover additional locations in the search region, *r* is used, which is a random number between 0 and 1. The caloric value, denoted by the letter *C* in Equation 2.2, is determined by the length of the mussel and can be found in Equation 2.5 [18].

$$T = \left(\frac{(L - 3)}{(5 - 3)} * 10 \right) - 5 \tag{2.3}$$

$$E = \left(\frac{i - 1}{n - 1} \right) - 0.5 \tag{2.4}$$

$$C = \left(\frac{(L - 3)}{(5 - 3)} * 2 \right) + 0.6 \tag{2.5}$$

Equation 2.3 displays the value between (-5) and (5) as well as the value derived from Equation 2.5 in the range of 0.6 to 0.8. E was derived from Equation 2.4 and linearly dropped from 0.5 to -0.5, where i stands for the number of iteration, which begins with the iteration number and finishes with one, and the E value will be fixed in the final two iterations. As a result, T and E that represent the time and energy needed to crack the potential mussel (which must be lower than EO's energy), may have negative values. L, a random number that varies randomly, is used to determine the T value in Equation 2.1 and the C value in Equation 2.2.

2.13.2 EOO Algorithm

Figure 2.15 is an illustration of the pseudo-code that the EOO uses to control how it can handle optimization complications.

```
Initialize the EO population Xi (i=1,2,3,.....,n)
Calculate the Fitness of each search agent
Xbest = the best solution in search agent
while (i>0)
    For each solution in search agent
      L=random (3,5)
      Calculate T, E and C based on equations 2.3 ,2.4 and 2.5
      Update the position of solution based on equations 2.1 and 2.2
    End for
   Calculate the fitness of each search agent
   Xbest = the best solution in search agent
End While
Return Xbest
```

Figure 2.15 Pseudo code for the EOO algorithm [18]

Using the time necessary to break the mussel that is computed with the energy of the bird as well as the size of the mussel as factors to determine the predicted position of the targeted food, thus enhances the worth of choice. The use of arbitrary values during the optimization process allows for the exploration of new territory with each iteration. Consequently, avoid a local minimum problem. The use of random values at each stage of optimization ensures both exploration and exploitation of the space being optimized.

2.14 FS using the Rock Hyraxes Swarm Optimization (RHSO)

Rock Hyraxes Swarm Optimization (RHSO), an entirely novel instance of a metaheuristic algorithm, which Belal Al-Khateeb et al. [19] introduced, was motivated by the natural activities of swarms of rock hyraxes. In order to determine what the Rock Hyraxes (shown in Figure 2.16 [19]) are eating and how they specifically see this food, the RHSO algorithm imitates their social behavior. A small, fuzzy mammal called a rock hyrax (Procavia Capensis) inhabits rocky terrain in sub-Saharan Africa as well as the coast of the Arabian Peninsula. The colonies of these animals, which can contain up to 50 members, are typically led by a lone male who violently defends his territory. They raise their children together, eat together, and even sleep together [98] [99]. Rock hyraxes inhabit regions with a wide range of average temperatures as well as those that have enough food and water. The successful isolation of the rocky protrusions segregated through their dispersal may have been facilitated by decreased metabolic rates and transparent internal temperatures. Every day, the rock hyrax forms a circle around itself to eat, with its head pointed outward to watch for predators. When the group is eating or sunbathing, the male or female of breeding age (leader) will keep an eye on things from a high rock. If danger approaches, they will emit a shrill alarm or call, the group will immediately begin to take shelter at that point [100].

Figure 2.16 The Rock Hyraxes [19]

2.14.1 Mathematical Model of the RHSO

Rock hyraxes begin spending several hours in solar baths and cohabiting together. They look for food in a specific fashion, creating a circle with various angles and diameters. The Leader adopts a higher position when they come upon food so that everyone can eat while being protected from rapacious animals. The population of the Rock hyrax swarm is initially made up of the leader and followers. The leader chooses the highest and finest location to watch over the other members. The Leader uses Equation 2.6 [19] below to update its position based on his previous location:

$$Leader = r1 * x(Leader_{Pos}, j) \tag{2.6}$$

where *r1* denotes a random number ranging from 0 to 1, x denotes the Leader's previous position, *LeaderPos* denotes the Leader's old location, and j denotes each diminution. All members change their positions based on their positions after the leader position has been updated as shown in Equation 2.7 [19]:

$$X(i,j) = (x(i,j) - (circ * x(i,j) + Leader)) \tag{2.7}$$

where *circ* corresponds to a spherical pattern and an attempt is made to replicate the circular system, it is derived as shown in Equations 2.8, 2.9, and 2.10 [19]:

$$n1 = r2 * cos(ang) \tag{2.8}$$

$$n2 = r2 * sin(ang) \tag{2.9}$$

$$circ = sqrt(n1^2 + n2^2) \tag{2.10}$$

where *ang* is an arbitrary value between 0 and 360, and denotes to the angle of a movement, and *r2* is a random integer between 0 and 1, and refers to the radius. The *ang* is updated with each generation, with the update relying on the variables' lower and upper bounds, which are an arbitrary number called *lb* and *ub* as shown in Equations 2.11 and 2.12 [19]:

$$dalta = random\,[lb, ub] \tag{2.11}$$

$$ang = ang + dalta \tag{2.12}$$

2.14.2 RHSO Algorithm

Figure 2.17 displays the RHSO algorithm's pseudo-code. The RHSO first generates random solutions and determines the fitness of those solutions. It classifies the individual with the highest fitness as the Leader, switching from exploratory to localized exploitation mode and concentrating on the favorable areas when the universal optimum may be nearby. The Leader symbolizes the most effective approach to optimization problems that may arise. The search agents restart their exploratory moves and then transition into a new phase of exploitation. The Leader adjusts his position in accordance with Equation 2.6, whereas the other members adjust their

positions in accordance with the Leader's position as depicted in Equation 2.7. This method will proceed through all iterations; when it reaches the stop condition, it returns the Leader as the closest approximation to the optimal solution to the optimization issue.

```
Initialize the population of Rock Hyrax N member
Calculate the Fitness of each search agent
Leader = the best search agent
t=1
while (t < Max number of iterations)
    update Leader position, according to Eq.(2.6).
    update the position of each search agent, according to Eq.(2.7).
    Calculate the fitness of each search agent
    Select the best member of the population as a leader
    Update the angle, according to Eq.(2.11) and Eq.(2.12).
    t= t+1.
End While
Return Leader as the best solution.
```

Figure 2.17 Pseudo code for the RHSO algorithm [19]

3

THE ML-BASED BOTNET DETECTION SYSTEM

3.1 Introduction

Through a comprehensive revision of the literature review, it is decided to utilize the new X-IIoTID dataset [17], which is a specialized dataset in the IIoT cyberattacks field. According to the authors' knowledge, three researchers have only used this dataset including the creator of the dataset. This thesis aims to apply the EOO [18] and RHSO [19] algorithms as a features selection in the BotNet detection problem and evaluate their performance by using well-known machine learning algorithms. This will be helpful in achieving high system accuracy, reducing training time as well as the overall resources required by the system, and increasing the robustness of the system against overfitting.

This chapter is dedicated to present the proposed system and its details. It is distributed into three parts. The first part is implementing a dataset preprocessing on the X- IIoTID dataset. The second part of this chapter is the process of implementing the features selection algorithms ECC and RHSO on the dataset to get a suitable reduced subset of features. The third part is to use the machine learning state of the art algorithms to detect BotNets. This part will be implemented on the original dataset with its original features and on the two subsets features using dataset mapping.

3.2 The Proposed ML-Based BotNet Detection System Architecture

In general, the structure of the proposed system may be broken down into its three primary components, which are as follows: The first is the dataset preprocessing. The second part is the implementation of the EOO and RHSO algorithms separately as features selection methods in which a fitness analyzer that employs the utilization of the machine learning algorithm is utilized. The fitness analyzer that was suggested included two components: the first one was the accuracy that was achieved through ML, and the second component was the quantity of the features that were extracted.

The third part is the implementation of ML algorithms on the original X-IIoTID dataset (with its original features) and on the new datasets (with two subsets features using dataset mapping). This is a BotNet detection system. The “class2” attribute in the X-IIoTID dataset will be consider as the target features, which contains nine types of attack and normal.

The following figure shows the general form of the proposed system, which will be discussed in greater depth in the sections that are to come in this chapter. Throughout the essence of the work, a few processes exist that need to be conducted. These processes include the dataset preprocessing and the dataset mapping, both of which will be detailed later on in this chapter. The proposed system architecture can be abbreviated as shown in Figure 3.1.

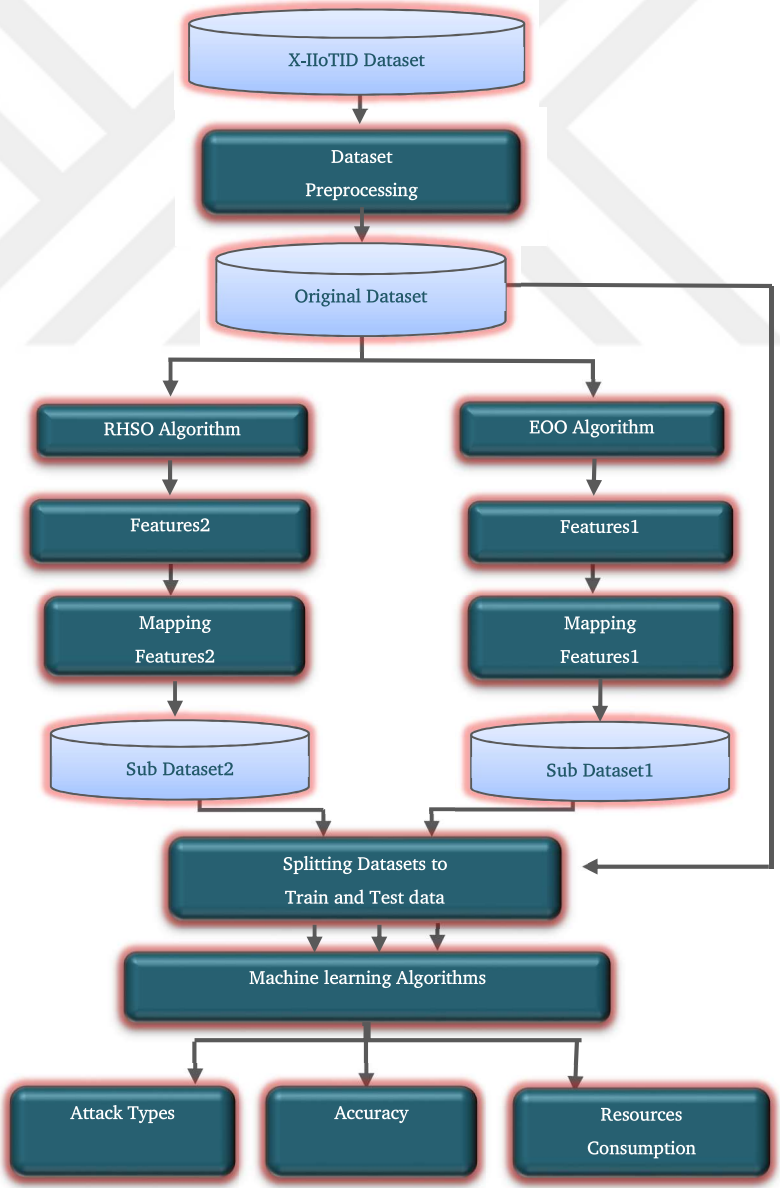


Figure 3.1 The structure of the proposed BotNet detection system

3.3 Dataset Preprocessing Steps

Because machine learning algorithms acquire knowledge from the data, and the assessment of learning for problem solving significantly relies on the appropriate data that is necessary to find an optimal solution (which are called features), preprocessing the data is the required preliminary stage before any machine learning algorithm can be used. Because of the fundamental role that these features play in learning and comprehension, machine learning is frequently interpreted in the context of feature engineering. Briefly, data preprocessing is a part in the direction of data acquisition and data analytics that involves taking fresh data and putting it into a form that can be absorbed by computers and machine learning software. This phase takes place between the data collection phase and the data analysis phase. so that it can then be examined by these technologies [101]. Figure 3.2 summarizes the preprocessing steps of the X-IIoTID dataset that will be discussed in the following subsections.

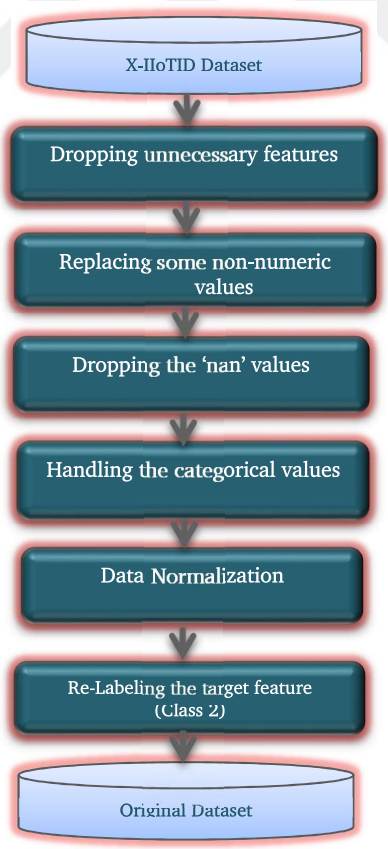


Figure 3.2 X-IIoTID preprocessing steps

3.3.1 Dropping Unnecessary Features

This thesis (as mentioned earlier) will consider the “class2” attribute in the X-IIoTID dataset as the target feature, which contains nine types of attacks and normal behavior. Therefore, “class1” and “class3” attributes (which are used for binary classification or further multi classification) will be dropped. The “Date” and “Timestamp” attributes will also be dropped to increase the process because they do no effect the detection process. So, from the 68 attributes of the X-IIoTID dataset only 64 attributes will be used.

3.3.2 Replacing Some Nonnumeric Values

Some of the nonnumeric values in the dataset that have the same meanings was converted in all its occurrences to numeric with the same values as shown in Table 3.1.

Table 3.1 Replacing equivilant nonnumeric values

Old Nonnumeric Value	New Numeric Value
“FALSE”	0
“False”	0
“false”	0
“TRUE”	1
“True”	1
“true”	1

The dataset also contains some nonsense or unidentified data that was converted to “*nan*” ("*Not a Number*", sometimes known as "*nan*", is a numeric data format that refers to an indeterminate value or a number that cannot be expressed. This is most commonly the case with the outcomes of floating-point calculations [102]) as shown in Table 3.2:

Table 3.2 Converting nonsense values to “nan”

Nonsense Data	New Data
“-”	“nan”
“?”	“nan”
“ ”	“nan”
“ ”	“nan”

3.3.3 Dropping the “nan” Values from the Dataset

The “nan” values are almost inevitable in every real-world dataset and practically impossible to avoid in typical data gathering processes. This can occur for various reasons, such as errors during data entry, issues on the technical front in the data collection process, lost/corrupt files, and many other reasons. In any real-world dataset, there is usually some missing data that have to be dealt with; otherwise, it can potentially lead to several problems in processing the data. Because of this, all the “nan” values were dropped from the dataset.

3.3.4 Handling the Categorical Values (Data Transformation)

The X-IIoTID dataset contains services, protocols, status flags, and other attributes as nominal attributes. Examples of these attributes are: “Scr_IP”, “Scr_port”, “Protocol”, ”Service”, “missed_bytes”, “Avg_ldavg_1” etc. The classification algorithm requires each of the input records to be in the shape of vectors composed of the real numbers. Therefore, the data transformation process changes the categorical values of features into their corresponding numeric values. As a consequence of this, the symbolic or categorical elements of the dataset are converted into numeric data. To accomplish this, these attributes are factorized and then encoded. Utilizing this approach will enable the acquisition of a numerical representation of an attribute, which is really useful.

3.3.5 Data Normalization

During the process of logical evaluation, the normalization of data is a procedure that falls under the category of data preprocessing, and it is mainly important when allocating with data tables. The necessity to minimize the artificial intelligence model's sensitivity to the values of the attributes contained in the dataset in order to improve the analyzed model's level of sufficiency is the primary factor that determines the significance of its implementation [103].

The purpose of changing the values that are contained within the dataset's numerical columns to a similar scale is to achieve the aim of data normalization, which is to accomplish this objective without distorting the variations in the ranges of the values. Each feature in the X-IIoTID dataset has different data limits. As according Equation 3.1, the value of each feature has to be linearly translated to the range 0 to 1, where Max signifies the highest amount for every feature and Min denotes the minimum amount for each feature. The Min-Max Scaler is used to accomplish this goal in order to prevent problems in the machine learning algorithms that might arise from having features with values that are excessively high to dominate the dataset. The significant feature X is normalized to X', so that its minimum value is equal to its highest value [103].

$$X' = \frac{X - Xmin}{Xmax - Xmin} \qquad (3.1)$$

3.3.6 Re-Labelling the Target Feature (“class2”)

Re-labelling the contents (attacks and normal) of the target feature (“class2”) to the numeric values between (0-9) according to their appearance in the “claas2” attribute as shown in Table 3.3.

Table 3.3 Re-labelling the target featute (“class2”)

Label	Re-label
“C&C”	0
“Exfiltration”	1
“Exploitation”	2
“Lateral _movement”	3
“Normal”	4
“RDOS”	5
“Reconnaissance”	6
“Tampering”	7
“Weaponization”	8
“crypto-ransomware”	9

3.4 EOO and RHSO Algorithms Steps for Implementation

In this part, the steps that need to be taken in the process of implementing the EOO and RHSO algorithms are detailed according to this study. These processes include the features selection method, the representation of the problem, the fitness computation procedure for the problem, and the halting condition in order to reach the final results.

3.4.1 Representation of the Problem (Initialize Population)

The aim of this system is to test the EOO and RHSO algorithms (separately) as a features selection and reduction to reduce the features of the X-IIoTID dataset and to choose the best ones to obtain high accuracy of detecting BotNets and reduce resources consumption with the least number of features.

The total number of the X-IIoTID dataset features is 68, but four of them were dropped in the preprocessing phase and one is the target attribute, so the total became 63. Since the number of features is 63, any solution can be represented by 63 binary values of 0 or 1, where 0 represents the absence of the feature and 1 represents its existence. According to both the EOO and RHSO algorithms, the solutions are initialized randomly and then they learn to depend on fitness and the stopping condition to reach the optimal solution. After the solutions generation process, the dataset is processed based on the solution values and a new two datasets are generated to be entered into the next step of the features selection algorithm.

3.4.2 Performing the Calculations on Fitness

In our problem, fitness calculation is a very important part because it determines the best solution that results from the EOO and RHSO algorithms, and which represents the final solution. The fitness is calculated using advanced technologically machine learning algorithms (Random Forest and KNN, which was discussed in Chapter Two). Because the lowest possible number of features and higher accuracy are needed, multi-objective function has been built in order to handle the process of trade-offs between feature count and precision need to be made. It consists of two parts: first is to search for the highest accuracy and second to get the lowest number of features. The result of this function is to represent the final fitness to be used by the EOO and RHSO algorithms.

3.4.3 FS Algorithms Final Results

The results from the step before indicate the final solutions acquired from the EOO and RHSO algorithms, which represent the best possible solutions once all steps have been completed. These best solutions are evaluated one at a time by the machine learning algorithm in order to validate the results that were acquired.

3.5 Mapping of Dataset

In this step, the results (best features) acquired by EOO and RHSO algorithms are projected onto the original dataset so as to select the dataset features to be subsequently trained and reviewed by the machine learning algorithms in order to measure the accuracy and resources usage as a consequence. If the obtained subset of features assumed to be {1, 2, 5, 8, 9, 11, 13… , 62}, Figure 3.3 demonstrates the mechanism of mapping the these subset to the original dataset and obtain a new dataset that contains only the selected features.

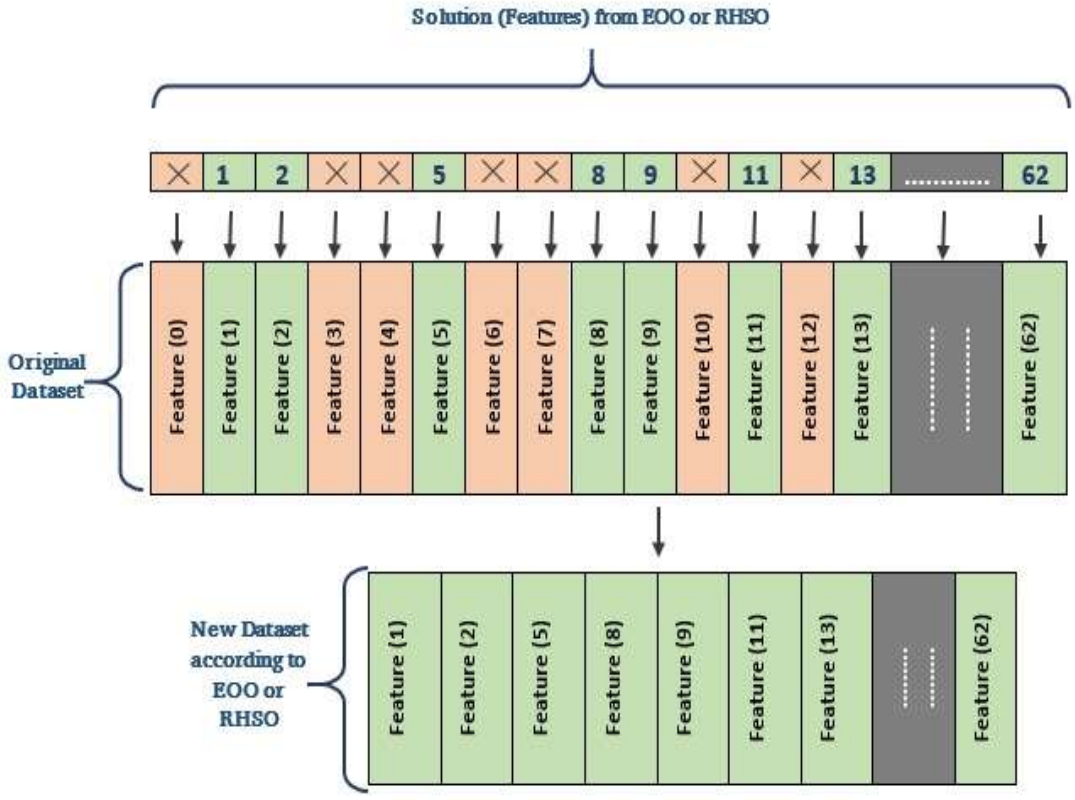


Figure 3.3 The mapping procedure

3.6 Splitting Dataset

In order to evaluate how effectively the machine learning model works, first, the dataset has to be partitioned into the training set and the testing set. The train set is utilized in the process of fitting the model, and the relevant data for the train set is already identified. The other set is referred to as "test data", and its main function is to be used for making predictions. In this thesis, the dataset is divided into a training dataset and a testing dataset, with the ratio being 70% training dataset and 30% testing dataset.

3.7 Machine Learning Algorithms Implementation

Final stage of the work represents interpretation of classification results. The motivation is to evaluate the features subsets obtained from the FS algorithms, and to check which one have the stronger impact on predicting (accuracy) and the resources they consume. Predictions interpretation stage is crucial before making a decision on choosing the most appropriate and trustworthy model for the future deployment.

3.7.1 Random Forest Algorithm Implementation

To produce more accurate results, Random Forest constructs many decision trees and combines them. When decision trees are being formed, randomness is added to the model. Instead of seeking for the most essential feature, it seeks for the best feature from a randomized group of features. It leads to diversity, which improves the model. It is fairly simple to determine the significance of a feature when making predictions [52]. To develop a training model, numerous decision trees are required. Figure 3.4 illustrates the actions taken for every decision tree:

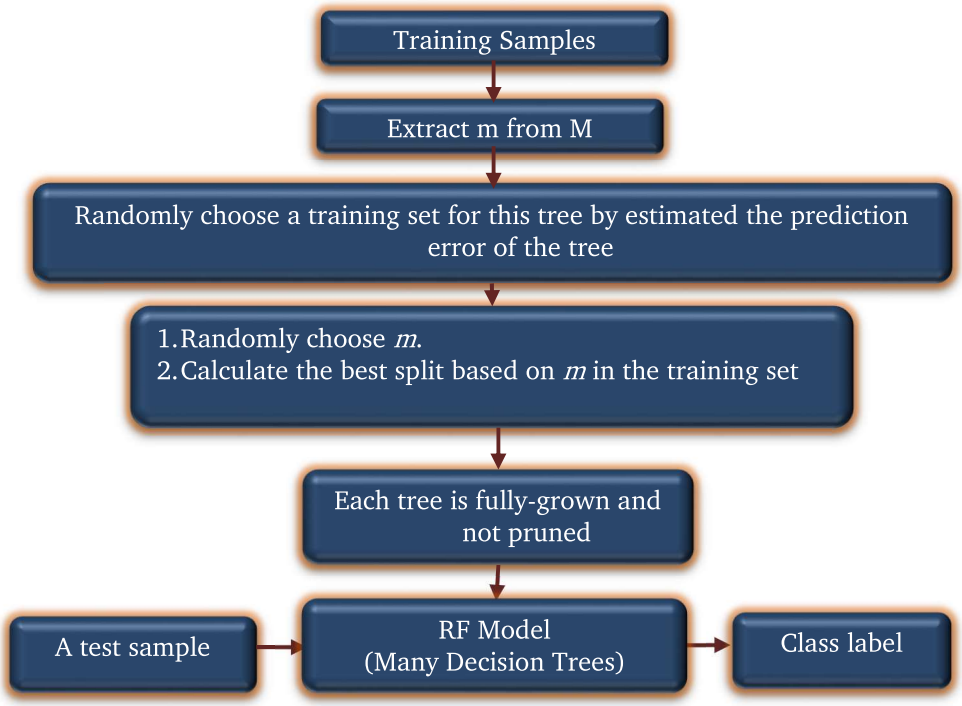


Figure 3.4 Steps for every tree.

where M denotes the feature of the dataset and m denotes a randomly selected features from M.

3.7.2 KNN Algorithm Implementation

The KNN algorithm is one example of a technique that could be applied to find solutions to issues involving classification. Due to the fact that KNN uses entirely the data for training when classifying and does not have a dedicated training stage, it is a lazy learning algorithm. Since it makes no assumptions about the data that it is being applied to, this learning algorithm may also be denoted to as a non-parametric learning algorithm. KNN employs feature likeness to determine the values of new information points, which means the newfangled information point will be allocated a value depending on how closely it resembles existing points inside the training set [55]. Figure 3.5 depicts the steps necessary to build the KNN algorithm.

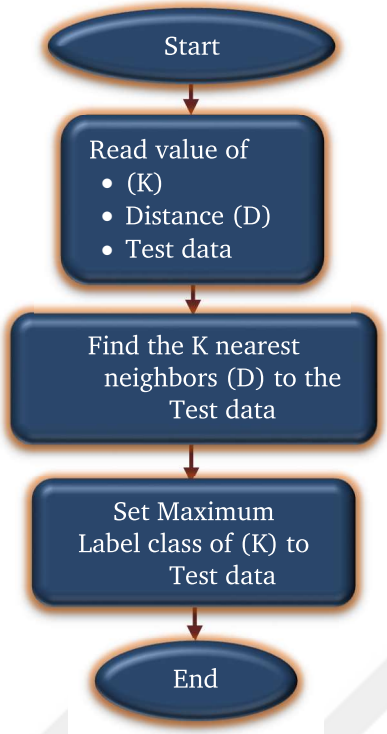


Figure 3.5 Steps to implement the KNN algorithm

where *K* denotes the number of labeled points (neighbors), and *D* denotes the distance between two points.

3.8 The Overall Structure of the Proposed System

The suggested system is demonstrated, and the X- IIoTID dataset preparation was completed. An appropriate reduced subset of features is achieved after the implementation of the feature selection algorithms ECC and RHSO on the original dataset. The resulting solutions are used to apply date mapping on the original dataset. The original dataset and the two sub datasets are split into a training dataset and a testing dataset, with the ratio being 70% training dataset and 30% testing dataset. To detect BotNets and determine accuracy and resource usage, the three datasets were incorporated into cutting-edge machine learning algorithms. Figure 3.6 describes the overall structure of the proposed system.

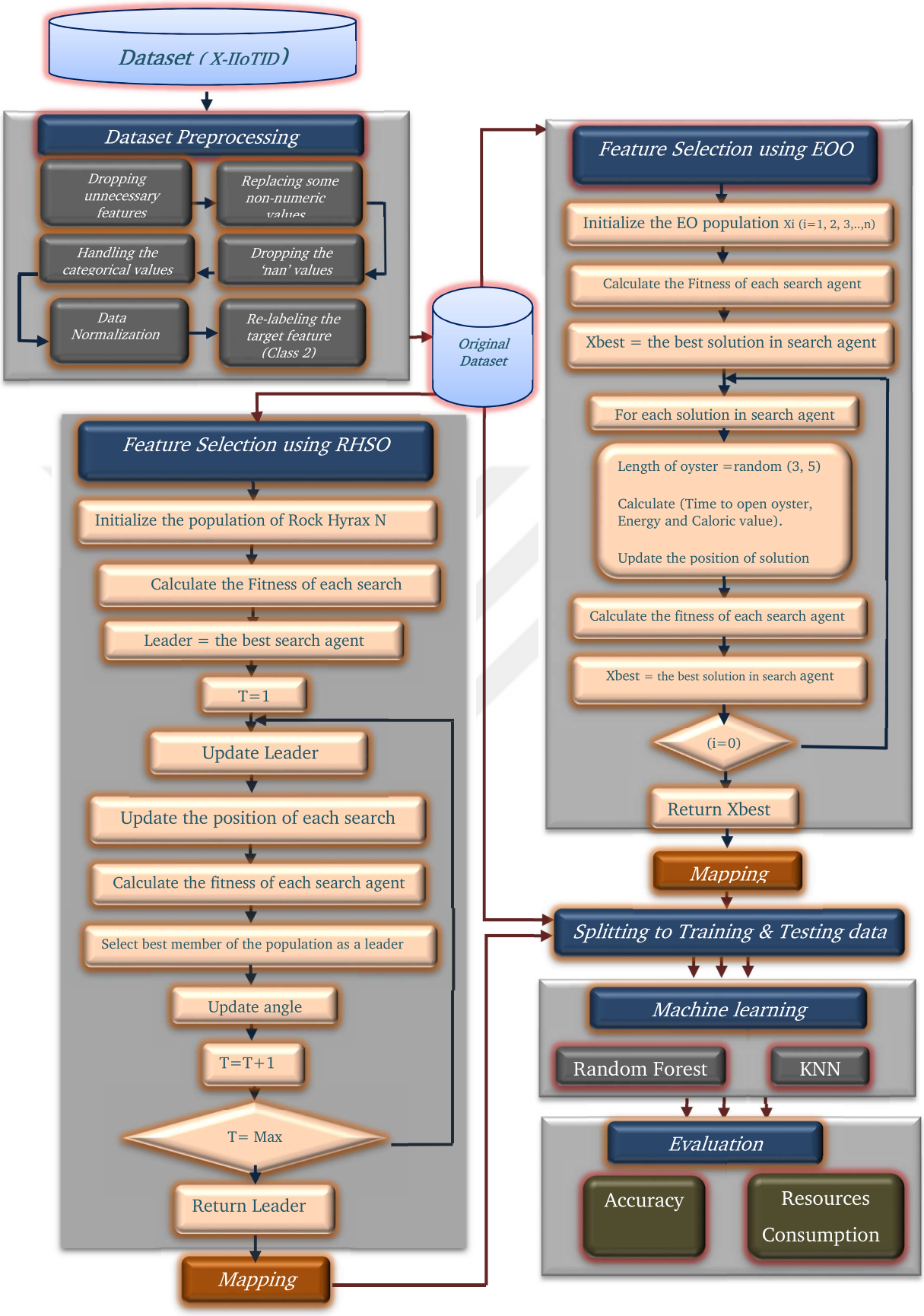


Figure 3.6 The overall structure of the proposed system

4

RESULTS AND DISCUSSION

4.1 Introduction

Reporting of the results and the necessary discussion and conclusions, are included in this chapter. In all the experimental results, the run of the system implementation has been done using Python on Google Colab with a PRO+ subscription. The product that was produced by Google Research and is referred to as the Colaboratory is what is meant to be referred to by the abbreviation "Colab". Colab is ideally suited for use in areas such as machine learning, data analysis, and educational settings since it enables anybody to create and run unrestricted Python script through the Internet. Technically speaking, Colab offers access to computer resources like as GPUs and extra RAM while being a hosted Jupyter notebook service that doesn't need any setup to operate. Background execution is a feature that's available with Colab Pro+, and it allows for uninterrupted executable code for a maximum of 24 hours. Idle timeouts will only take effect if the current code execution comes to an end. By buying a dedicated virtual machine from the GCP Marketplace, there is an ability to completely remove all runtime constraints and idle timeouts [104].

All of the procedures and experiments described in this thesis were carried out using the most well-known dataset in the area of BotNet detection (X-IIoTID) [17]. This dataset was acquired from the IEEEDataPort/DATASETS official website, and it can be accessed at the following URL: “https://ieee-dataport.org/documents/x-iiotid-connectivity-and-device-agnostic-intrusion-dataset-industrial-internet-things”.

The results that were acquired during the research for the thesis are divided up into two primary parts. The first part is devoted to the presentation and discussion of the results of the execution of the EOO and RHSO algorithms as a features selection. These algorithms were used to pick features from the X-IIoTID dataset in order to obtain the fewest possible features while maintaining the greatest possible accuracy. The second part is devoted to the presentation and discussion of the results of detecting BotNets,

as well as the accuracy and resource usage while utilizing 1) the original dataset, 2) the sub dataset resulting from the EOO algorithm, and 3) the sub dataset resulting from the RHSO algorithm. The machine learning algorithms (Random Forest and KNN) were all used in the process of detecting the BotNet.

4.2 Experimental Results and Discussion

The EOO and RHSO algorithms were used separately to pick as few features as possible to be employed in the proposed system. By applying these algorithms to the original dataset (which contains 63 features) using ML algorithms (Random Forest, and KNN), the results could be found in Tables 4.1 and 4.2. Table 4.1 shows the locations of the chosen features when applying EOO algorithm. While Table 4.2 shows the locations of the chosen features when applying RHSO algorithm. The EOO and RHSO algorithms are population based, each algorithm contains ten solutions, the best result (depending on the fitness) is chosen as the best features. These selected features are the best available option to be used in the proposed system.

Table 4.1 Locations of the chosen features when applying EOO algorithm

EOO Algorithm		
Machine Learning Algorithm	Number of Features	Locations of Features
Random Forest	13	[2, 6, 7, 10, 18, 27, 33, 34, 37, 40, 43, 57, 60]
KNN	13	[5, 11, 17, 20, 21, 23, 28, 34, 40, 43, 52, 55, 60]

Table 4.2 Locations of the chosen features when applying RHSO algorithm

RHSO Algorithm		
Machine Learning Algorithm	Number of Features	Locations of Features
Random Forest	10	[7, 10, 20, 23, 34, 40, 44, 58, 60, 62]
KNN	10	[2, 10, 20, 21, 34, 40, 44, 58, 60,62]

The Random Forest and KNN algorithms are used for the BotNet detection using the original dataset and the two sub datasets that obtained from applying the EOO and RHSO algorithms. The result of the detection in accordance to accuracy and resources consumption is illustrated in Table 4.3. When the ML algorithms (Random Forest and KNN) are applied to the three datasets, it is immediately apparent that there is an improvement in the quality of the results. The accuracy maintains its significant value, the time necessary to detect BotNets is reduced, and the memory utilization has been dropped.

Table 4.3 The accuracy and resources consumption in the BotNet detection

Machine Learning Algorithm	Original Dataset			Sub Dataset obtained from EOO			Sub Dataset obtained from RHSO		
	Accuracy	Time	Memory	Accuracy	Time	Memory	Accuracy	Time	Memory
Random Forest	98.71%	2.73 Min.	137.64 MB	99.35%	1.65 Min.	54.05 MB	99.88%	1.25 Min.	49.11 MB
KNN	94.34%	3.35 Min.	30.46 MB	94.44%	2.26 Min.	21.01 MB	97.30%	0.8 Min.	19.69 MB

Table 4.4 and Figure 4.1 show the increasing ratio of accuracy when using the new sub datasets with accordance to the original dataset. It has been demonstrated that the accuracy can indeed be improved by 0.65% when the Random Forest ML algorithm is employed with the subdataset1, and that it can enhance the accuracy by 1.19% when it is used with the subdataset2.

Moreover, it has been demonstrated that the accuracy can in fact be increased by 0.11% when the KNN ML algorithm is utilized with the subdataset1, and that it can improve the accuracy by 3.14% when it is utilized with the subdataset2.

Table 4.4 Increasing ratio of accuracy

Machine Learning Algorithm	Sub Dataset obtained from EOO	Sub Dataset obtained from RHSO
	Accuracy	Accuracy
Random Forest	0.65%	1.19%
KNN	0.11%	3.14%

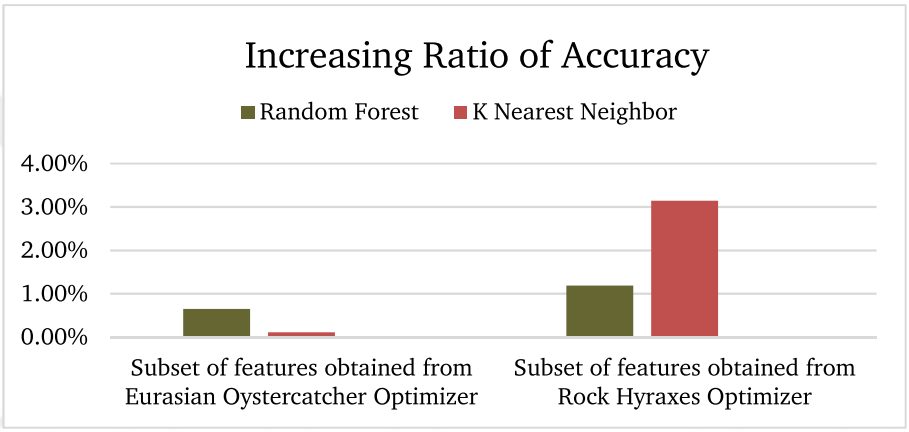


Figure 4.1 Increasing of accuracy ratio chart

Table 4.5 and Figure 4.2 show the decreasing ratio of time consumption when using the new sub datasets with accordance to the original dataset. It has been observed that the time necessary to complete the detection process can be reduced by a ratio of 39.56% when the Random Forest ML algorithm is applied to subdataset1, and that time can be reduced by a ratio of 54.21% when the same ML algorithm is used to subdataset2.

Additionally, it has been found that the duration of time required to finish the detection process can be decreased by a ratio of 32.54% when the KNN ML algorithm is applied to subdataset1, and also that amount of time can be decreased by a ratio of 76.12% when the same ML algorithm is applied to subdataset2.

Table 4.5 Decreasing ratio of time consumption

Machine Learning Algorithm	Sub Dataset obtained from EOO	Sub Dataset obtained from RHSO
	Time	Time
Random Forest	39.56%	54.21%
KNN	32.54%	76.12%

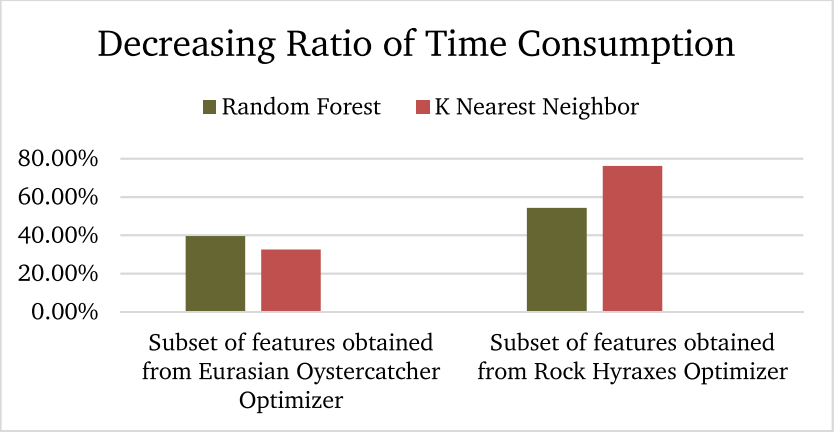


Figure 4.2 Decreasing of time ratio chart

Table 4.6 and Figure 4.3 show the decreasing ratio of memory consumption when using the new sub datasets with accordance to the original dataset. When the Random Forest ML algorithm is applied to the first sub dataset, it is noticed that the amount of memory used in the execution of the detection process is decreased in a ratio of 60.73%. When the same ML algorithm is applied to the second sub dataset, it is noticed that the amount of memory used in the execution of the detection process is decreased in a ratio of 64.32%.

Furthermore, overall memory required for the execution of the detection process is reduced by a ratio of 31.02% when the KNN ML algorithm is applied on the first sub dataset. Memory amount for running the detection process are shown to be reduced

by a factor of 35.36 percent when the same ML technique is applied to the second sub dataset.

.

Table 4.6 Decreasing ratio of memory consumption

Machine Learning Algorithm	Sub Dataset obtained from EOO	Sub Dataset obtained from RHSO
	Memory	Memory
Random Forest	60.73%	64.32%
KNN	31.02%	35.36%

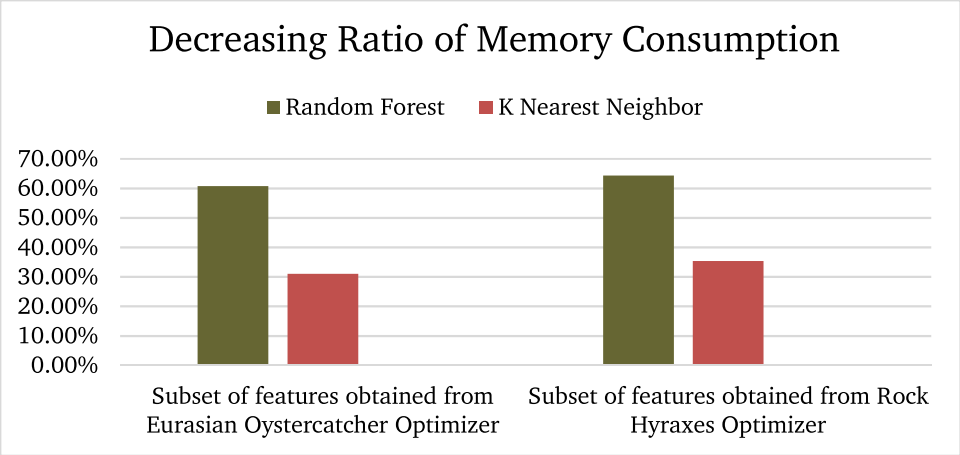


Figure 4.3 Decreasing of memory ratio chart

4.3 Comparing the Results with the Literature

The results that were achieved via the previous studies as well as through the suggested system are shown in Table 4.7. The incorporation of the feature selection algorithms (EOO or RHSO) with the methods for machine learning (Random Forest and KNN) has produced satisfactory results when compared to the results that are reported in the literatures. The features that are acquired by the two algorithms are used in the suggested system, which has high results in terms of the amount of time spent implementing it, the amount of memory consumed, and the accuracy of its predictions.

Table 4.7 Results obtained from the literature and proposed system

No.	Researcher/s	Year	Algorithm	Dataset/s	Number of Selected Features from the Original	Accuracy	Limitations
1	Nilesh Kunhare et al. [6]	2020	PSO	NSL-KDD	10/41	99.32	• Outdated dataset. • Only four attack types. • Perform the feature selection before the dataset preprocessing. • Resource consumption was not mentioned.
2	Maria Habib et al. [7]	2020	MOPSO-Levy	Baby Monitor	46/115	93.5	• Only two attack types. • Dataset preprocessing was not done. • High computational cost. • Resource consumption was not mentioned.
				Danmini Doorbell	43/115	98.6	
				Security Camera PT737	45/115	95.5	
				Security Camera PT838	45/115	96.1	
				Ecobee Thermostat	42/115	99.6	
				Security Camera XC1002	44/115	96.9	
				Security Camera XC1003	45/115	98.4	
3	Maha M. Althobaiti et al. [8]	2021	BBFO	NSL-KDD	18/41	98.1	• Outdated dataset. • Binary classification. • Resource consumption was not mentioned. • The unwanted samples from the dataset did not removed.
				CICIDS-2012	23/80	98.45	
4	Abdullah Alharbi et al. [9]	2021	LGBA-NN	N-BaIoT	*/115	90.00	• Only two attack types. • Dataset preprocessing was not done. • The number of selected features was not reported. • Resource consumption was not mentioned.

Table 4.7 Results obtained from the literature and proposed system (Continue)

5	Saif S. Kareem et al. [10]	2021	GTOBSA	BoT-IoT	3/46	81.5	• Outdated dataset. • Dataset preprocessing was not done. • The proposed system was tested on 5% of the full dataset. • Resource consumption was not mentioned. • Binary classification.
				NSL-KDD	15/41	95.5	
				CICIDS-2012	10/80	98.7	
				UNSW-NB15	17/49	81.5	
6	Sohail Saif, et al. [11]	2022	PSO+GA+DE	NSL-KDD	10/41	93.67	• Outdated dataset. • Binary classification.
7	Mythili Henry Boopathi et al. [12]	2022	CSHO	BoT-IoT	*/115	**	• Accuracy was not mentioned. • Binary classification. • Dataset preprocessing was not done. • The number of selected features was not reported. • Resource consumption was not mentioned.
8	Raisa Abedin Disha et al. [13]	2022	GIWRF	UNSW-NB15	20/49	93.01	• Binary classification. • Resource consumption was not mentioned. • Portion of the dataset was used.
				Network TON-IoT	10/42	99.9	
9	Mohammed Otair et al. [14]	2022	GWO+PSO	NSL-KDD	20/41	74.48	• Outdated dataset. • Binary classification.
10	Aniss Chohra et al. [15]	2022	CHAMELEON	NSL-KDD	9/41	90.71	• Outdated dataset. • Binary classification. • Dataset preprocessing was not done.
				UNSW-NB15	8/49	89.52	
11	Proposed Work	2022	EOO	X-IIoTID	Random Forest 13/68	99.35	• The proposed work was not applied to a real-time environment.
					KNN 13/68	94.44	
			RHSO	X-IIoTID	Random Forest 10/68	99.88	
					KNN 10/68	97.30	

* Not Reported.

** The researcher did not mention the accuracy; however, he mentioned the energy, F-measure, Precision, and Recall as 0.1610, 0.9001, 0.9052, and 0.8993 respectively.

The research results that were included in the previous tables show that the features that were selected by both the EEO and the RHSO algorithms led to the use of less data, which increased the speed of the classification process while also reducing the amount of time it took and the amount of memory it required. These gains were demonstrated by the research results that were included in the previous tables. Correspondingly, it has been realized that utilizing features from the X-IIoTID dataset that have been properly and scientifically selected has resulted in an improved performance of the proposed system. This improvement can be seen in terms of the speed with which the system is implemented, as well as a reduction in the amount of memory that is used while still maintaining a high level of accuracy.

4.4 Conclusions

During the process of developing the framework for the proposed system, a number of observations and inferences were made. The list that follows summarizes the most significant conclusions from this thesis:

1- Due to the strong performance that the ML approaches produce when they are utilized with the X-IIoTID dataset, they are suited to be used in dealing with the multi class classification challenges that arise when attempting to detect BotNet activity.

2- Using a small number of features increases the performance of the ML algorithms in terms of accuracy, and consumed time and memory for training and testing models.

3- As according to what was said in the reviewed literature, there have been earlier efforts that have attempted to decrease the number of features using a variety of methodologies and algorithms. Combining the feature selection algorithms (EOO or RHSO) with the methods for machine learning (Random Forest and KNN) has produced satisfactory results when compared to the

results that are reported in the literatures. This is due to the fact that the EOO or RHSO algorithms are used in conjunction with the Random Forest and KNN methods.

4- The features that are obtained by the two algorithms are employed in the system that has been suggested. This system has high results in terms of the amount of time spent implementing it, the amount of memory that it consumes, and the accuracy of its predictions.

5- To choose from among the available features, the RHSO algorithm is more effective than the EOO algorithm in dealing with the BotNet detection problems, particularly when it comes to working with the X-IIoTID dataset based on the results described in previous tables.

6- The experimental study revealed that employing the RHSO algorithm in conjunction with the Random Forest algorithm was superior, with an accuracy rate of (99.88%), detection time of (1.25 minute), and memory consumption of (49.11 MB).

7- The thesis overcomes most of the limitations of the literature in accordance to:

   a- Utilizing modernistic dataset, and training and testing the proposed system using the entire dataset.

   b- Dataset preprocessing was done before features selection and training the model.

   c- Reducing the number of features while maintaining high accuracy.

   d- Resource consumption was taking into consideration.

   e- Detecting BotNets in a multi classification fashion.

   f- Applying new dataset and two features selection algorithms that had not been used in BotNet detection (as far as the authors know).

4.5 Future Works

The following is a list of potential future works that might be done as extra work to this thesis:

1- Solve the imbalance in the data, which often indicates an uneven distribution of classifications within the dataset. Both oversampling and undersampling are methods that can be used to accomplish this goal.
2- Use the feature selection algorithms in conjunction with one another in a sequential fashion rather than using them separately. This implies that the outcomes of the first algorithm might be passed on to the second algorithm. The features that were produced as a consequence will represent the solution.
3- To demonstrate that the system is not dependent on a single dataset, it is necessary to employ a different BotNet dataset as a benchmark.
4- Engage the "class1" and "class2" attributes from the X-IIoTID dataset to make further classifications of attack types.
5- Setup an IIoT platform or environment and install IIoT devices to observe real-time data processing and transferring between the devices. The BotNet detection system could work within this environment to detect Bots in real life.

REFERENCES

[1] Wang Ren, Xin Tong, Jing Du, Na Wang, Shan Cang Li, Geyong Min, Zhiwei Zhao, Ali Kashif Bashir, “Privacy-preserving using homomorphic encryption in Mobile IoT systems,” Computer Communications, Volume 165, 2021, Pages 105-111, ISSN 0140-3664, https://doi.org/10.1016/j.comcom.2020.10.022.

[2] Mark Mbock Ogonji, George Okeyo, and Joseph Muliaro Wafula, “A survey on privacy and security of Internet of Things,” Computer Science Review, Vol. 38, 2020. 100312, ISSN 1574-0137, https://doi.org/10.1016/j.cosrev.2020.100312.

[3] Gonzalo De La Torre Parra, Paul Rad, Kim-Kwang Raymond Choo, and Nicole Beebe, “Detecting Internet of Things attacks using distributed deep learning,” Journal of Network and Computer Applications, Vol. 163, article 102662, 2020. 102662, ISSN 1084-8045, https://doi.org/10.1016/j.jnca.2020.102662.

[4] I. Ali, A. I. A. Ahmed, A. Almogren, M. A. Raza, S. A. Shah, A. Khan, and A. Gani, “Systematic Literature Review on IoT-Based Botnet Attack,” IEEE Access, Vol. 8, pp. 212220-212232, 2020, doi: 10.1109/ACCESS.2020.3039985.

[5] Mohanad Sarhan, Siamak Layeghy, and Marius Portmann, “Feature Analysis for ML-based IIoT Intrusion Detection,” ArXiv, Vol. abs/2108.12732, 2021. https://doi.org/10.48550/arXiv.2108.12732.

[6] Nilesh Kunhare, Ritu Tiwari, and Joydip Dhar, “Particle swarm optimization and feature selection for intrusion detection system,” Sādhanā, Vol. 45, No. 109, 2020. https://doi.org/10.1007/s12046-020-1308-5.

[7] Maria Habib, Ibrahim Aljarah, and Hossam Faris, “A Modified Multi-objective Particle Swarm Optimizer-Based Lévy Flight: An Approach toward Intrusion Detection in Internet of Things,” Arabian Journal for Science and Engineering, 45:6081–6108, 2020, https://doi.org/10.1007/s13369-020-04476-9.

[8] Maha M. Althobaiti, K. Pradeep Mohan Kumar, Deepak Gupta, Sachin Kumar, and Romany F. Mansour, “An intelligent cognitive computing based intrusion detection for industrial cyber-physical systems,” Measurement, Volume 186, 2021, 110145, ISSN 0263-2241, https://doi.org/10.1016/j.measurement.2021.110145.

[9] Abdullah Alharbi, Wael Alosaimi, Hashem Alyami, Hafiz Tayyab Rauf, and Robertas Damaševicius, “Botnet Attack Detection Using Local Global Best Bat Algorithm for Industrial Internet of Things,” Electronics, Vol. 10, No. 11: 1341, 2021. https://doi.org/10.3390/electronics10111341.

[10] Saif S. Kareem, Reham R. Mostafa, Fatma A. Hashim, and Hazem M. El-Bakry, “An Effective Feature Selection Model Using Hybrid Metaheuristic Algorithms for IoT Intrusion Detection,” Sensors, Vol. 22, No. 4: 1396., 2022. https://doi.org/10.3390/s22041396.

[11] Sohail Saif, Priya Das, Suparna Biswas, Manju Khari, and Vimal Shanmuganathan, “HIIDS: Hybrid intelligent intrusion detection system empowered with machine learning and metaheuristic algorithms for application in IoT based healthcare,” Microprocessors and Microsystems, 2022, 104622, ISSN 0141-9331, https://doi.org/10.1016/j.micpro.2022.104622.

[12] Mythili Henry Boopathi, “Henry MaxNet: tversky index based feature selection and competitive swarm henry gas solubility optimization integrated Deep Maxout network for intrusion detection in IoT,” International Journal of Intelligent Robotics and Applications, Vol. 6, pp. 365–383, 2022. https://doi.org/10.1007/s41315-022-00234-2.

[13] Raisa Abedin Disha, and Sajjad Waheed, “Performance analysis of machine learning models for intrusion detection system using Gini Impurity-based Weighted Random Forest (GIWRF) feature selection technique,” Cybersecurity, Vol. 5, No. 1, 2022. https://doi.org/10.1186/s42400-021-00103-8.

[14] Mohammed Otair, Osama Talab Ibrahim, Laith Abualigah, Maryam Altalhi, and Putra Sumari, “An enhanced Grey Wolf Optimizer based Particle Swarm Optimizer for intrusion detection system in wireless sensor networks,” Wireless Networks, Vol. 28, pp. 721–744, 2022. https://doi.org/10.1007/s11276-021-02866-x.

[15] Aniss Chohra, Paria Shirani, ElMouatez Billah Karbab, and Mourad Debbabi, “Chameleon: Optimized feature selection using particle swarm optimization and ensemble methods for network anomaly detection,” Computers & Security, Vol. 117, 2022, 102684, ISSN 0167-4048, https://doi.org/10.1016/j.cose.2022.102684.

[16] Victor R. Kebande, “Industrial internet of things (IIoT) forensics: The forgotten concept in the race towards industry 4.0,” Forensic Science International: Reports, Volume 5, 2022, 100257, ISSN 2665-9107, https://doi.org/10.1016/j.fsir.2022.100257.

[17] Muna Al-Hawawreh, Elena Sitnikova, Neda Aboutorab, July 30, 2021, "X-IIoTID: A Connectivity- and Device-agnostic Intrusion Dataset for Industrial Internet of Things", IEEE Dataport, 2021, Available at: https://dx.doi.org/10.21227/mpb6-py55.

[18] Ahmad Salim, Wisam K. Jummar, Farah Maath Jasim, and Mohammed Yousif, “Eurasian oystercatcher optimizer: New meta-heuristic algorithm,” Journal of Intelligent Systems, Vol. 31, No. 1, 2022, pp. 332-344. https://doi.org/10.1515/jisys-2022-0017.

[19] Belal Al-Khateeb, Kawther Ahmed, Maha Mahmood, and Dac-Nhuong Le, “Rock Hyraxes Swarm Optimization: A New Nature-Inspired Metaheuristic Optimization Algorithm,” Computers, Materials & Continua, Vol. 68, No. 1, pp. 643-654, 2021, ISSN: 1546-2226, https://doi.org/10.32604/cmc.2021.013648.

[20] Ibrahim M. El-Hasnony, Reham R. Mostafa, Mohamed Elhoseny, and Sherif I. Barakat, “Leveraging Mist and Fog for Big Data Analytics in IoT Environment,” Transactions on Emerging Telecommunications Technologies, Vol. 32, 2021. https://doi.org/10.1002/ett.4057.

[21] In Lee, “Internet of Things (IoT) Cybersecurity: Literature Review and Iot Cyber Risk Management,” Future Internet, Vol. 12, No. 9: 157, 2020. https://doi.org/10.3390/fi12090157.

[22] Gopal Singh Kushwah, and Virender Ranga, “Voting Extreme Learning Machine Based Distributed Denial of Service Attack Detection in Cloud Computing,” Journal of Information Security and Applications, Vol. 53, 2020, 102532, ISSN 2214-2126, https://doi.org/10.1016/j.jisa.2020.102532.

[23] Lubna Luxmi Dhirani and Thomas Newe, “Hybrid Cloud SLAs for Industry 4.0: Bridging the gap,” Annals of Emerging Technologies in Computing (AETiC), Print ISSN: 2516-0281, Online ISSN: 2516-029X, pp. 41-60, Vol. 4, No. 5, 2020, Published by International Association of Educators and Researchers (IAER), http://dx.doi.org/10.33166/AETiC.2020.05.003.

[24] Muna Al-Hawawreh, Elena Sitnikova, and Frank den Hartog, “An efficient intrusion detection model for edge system in brownfield industrial internet of things,” Proceedings of the 3rd International Conference on Big Data and Internet of Things. Melbourne, Australia: ACM, pp. 83–87. 2019. https://doi.org/10.1145/3361758.3361762.

[25] Muna Al-Hawawreh, Frank den Hartog, and Elena Sitnikova, “Targeted ransomware: A new cyber threat to edge system of brownfield industrial internet of things,” IEEE Internet of Things Journal, Vol. 6, No. 4, pp. 7137–7151, 2019. doi: 10.1109/JIOT.2019.2914390.

[26] Koen Tange, Michele De Donno, Xenofon Fafoutis, and Nicola Dragoni, “A systematic survey of industrial internet of things security: Requirements and fog computing opportunities,” IEEE Communications Surveys & Tutorials, Vol. 22, No. 4, pp. 2489-2520, Fourthquarter 2020, doi: 10.1109/COMST.2020.3011208.

[27] Tanishq Varshney, Nikhil Sharma, Ila Kaushik, and Bharat Bhushan, “Authentication & Encryption Based Security Services in Blockchain Technology,” International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), 2019, pp. 63-68, doi: 10.1109/ICCCIS48478.2019.8974500.

[28] Shanshan Zhao, Shancang Li, and Yufeng Yao, “Blockchain enabled industrial internet of things technology,” in IEEE Transactions on Computational Social Systems, vol. 6, no. 6, pp. 1442-1453, Dec. 2019, doi: 10.1109/TCSS.2019.2924054.

[29] Jameela Al-Jaroodi, and Nader Mohamed, “Blockchain in industries: A survey,” in IEEE Access, Vol. 7, pp. 36500-36515, 2019, doi: 10.1109/ACCESS.2019.2903554.

[30] Tiago M. Fernández-Caramés, and Paula Fraga-Lamas, “A review on the application of blockchain to the next generation of cyber secure industry 4.0 smart factories,” in IEEE Access, Vol. 7, pp. 45201-45218, 2019, doi: 10.1109/ACCESS.2019.2908780.

[31] Tiwari, R., Sharma, N., Kaushik, I., Tiwari, A., and Bhushan, B., “Evolution of IoT & Data Analytics using Deep Learning,” International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), 2019. doi: 10.1109/icccis48478.2019.8974481.

[32] Umar Iftikhar, Kashif Asrar, Maria Waqas and Syed Abbas Ali, “BOTNETs: A Network Security Issue,” International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 11, Issue 11, 2020. http://dx.doi.org/10.14569/IJACSA.2020.0111155.

[33] Sam Haria, “The growth of the hide and seek botnet,” Network Security, Vol. 2019, Issue 3, 2019, pp. 14-17, ISSN 1353-4858, https://doi.org/10.1016/S1353-4858(19)30037-6.

[34] European Union Agency for Cybersecurity, ENISA, “From January 2019 to April 2020- Botnet ENISA Threat Landscape”, Available online: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2020-botnet (accessed on 06 October 2022).

[35] Maxwell Scale Osagie, Osatohanmwen Enagbonma, and Amanda Inyang, “The Historical Perspective of Botnet Tools,” Current Journal of Applied Science and Technology 32 (6), 1-8, 2019. https://doi.org/10.9734/cjast/2019/v32i630040.

[36] Shun-Wen Hsiao, Yi-Ning Chen, Yeali S. Sun, and Meng Chang Chen, “A cooperative botnet profiling and detection in virtualized environment,” IEEE Conference on Communications and Network Security (CNS), National Harbor, MD, USA, 14–16, 2020. doi: 10.1109/CNS.2013.6682703.

[37] Constantinos Patsakis, Fran Casino, and Vasilios Katos, “Encrypted and covert DNS queries for botnets: Challenges and countermeasures,” Computers & Security, Vol. 88, 2020, 101614, ISSN 0167-4048, https://doi.org/10.1016/j.cose.2019.101614.

[38] Sonali Kothari, “Real Time Analysis of Android Applications by Calculating Risk Factor to Identify Botnet Attack,” Lecture Notes in Electrical Engineering, Vol 570. Springer, Singapore, 2020. https://doi.org/10.1007/978-981-13-8715-9_7.

[39] R. Kour, “Cybersecurity issues and challenges in Industry 4.0,” in Applications and Challenges of Maintenance and Safety Engineering in Industry 4.0, pp. 84–101, IGI Global, 2020. doi: 10.4018/978-1-7998-3904-0.ch005.

[40] J. Prinsloo, S. Sinha, and B. von Solms, “A review of Industry 4.0 manufacturing process security risks,” Applied Sciences, Vol. 9, No. 23, pp. 5105, 2019. https://doi.org/10.3390/app9235105.

[41] Majda Wazzan, Daniyal Algazzawi, Omaima Bamasaq, Aiiad Albeshri, and Li Cheng, “Internet of Things Botnet Detection Approaches: Analysis and Recommendations for Future Research," Applied Sciences, Vol. 11, No. 12: 5713, 2021. https://doi.org/10.3390/app11125713.

[42] Lav Gupta, Tara Salman, Ali Ghubaish, Devrim Unal, Abdulla Khalid Al-Ali, and Raj Jain, “Cybersecurity of multi-cloud healthcare systems: A hierarchical deep learning approach,” Applied Soft Computing, Vol. 118, 2022, 108439, ISSN 1568-4946, https://doi.org/10.1016/j.asoc.2022.108439.

[43] Zahian Ismail, Aman Jantan, Mohd. NajwadiYusoff, and Muhammad Ubale Kiru, “A Botnet Taxonomy and Detection Approaches,” Test Engineering and Management, Vol. 82, pp. 3386 – 3408, ISSN: 0193-4120, 2020. https://www.testmagzine.biz/index.php/testmagzine/article/view/1405.

[44] J. A. Jupin, T. Sutikno, M. A. Ismail, M. S. Mohamad, and S. Kasim, “Review of the machine learning methods in the classification of phishing attack,” Bulletin of Electrical Engineering and Informatics, Vol. 8, 1545-1555, 2019. ISSN 2302-9285. https://doi.org/10.11591/eei.v8i4.1344.

[45] Iftikhar Ahmad, Qazi Emad Ul Haq, Muhammad Imran, Madini O. Alassafi, and Rayed A. AlGhamdi, “An Efficient Network Intrusion Detection and Classification System,” Mathematics, Vol. 10, No. 3:530, 2022. https://doi.org/10.3390/math10030530.

[46] A. Tabassum, A. Erbad, and M. Guizani, “A survey on recent approaches in intrusion detection system in IoTs,” 15th International Wireless Communications & Mobile Computing Conference (IWCMC), pp. 1190–1197, Tangier, Morocco, June 2019. doi: 10.1109/IWCMC.2019.8766455.

[47] S. U. Jan, S. Ahmed, V. Shakhov and I. Koo, "Toward a Lightweight Intrusion Detection System for the Internet of Things," in IEEE Access, Vol. 7, pp. 42450-42471, 2019, doi: 10.1109/ACCESS.2019.2907965.

[48] Batta Mahesh, “Machine Learning Algorithms - A Review,” International Journal of Science and Research (IJSR) ISSN: 2319-7064, 2019. DOI: 10.21275/ART20203995.

[49] Faraz S. Tehrani, Michele Calvello, Zhongqiang Liu, Limin Zhang, and Suzanne Lacasse, “Machine learning and landslide studies: recent advances and

applications," Natural Hazards, 2022. https://doi.org/10.1007/s11069-022-05423-7.

[50] Khalid Alissa, Tahir Alyas, Kashif Zafar, Qaiser Abbas, Nadia Tabassum, and Shadman Sakib, "Botnet Attack Detection in IoT Using Machine Learning," Computational Intelligence and Neuroscience, Vol. 2022, Article ID 4515642, 14 pages, 2022. https://doi.org/10.1155/2022/4515642.

[51] Leo Breiman, "Random Forests," Machine Learning, Vol. 45, 5–32, 2001. https://doi.org/10.1023/A:1010933404324.

[52] Prajyot Palimkar, Rabindra Nath Shaw, and Ankush Ghosh, "Machine Learning Technique to Prognosis Diabetes Disease: Random Forest Classifier Approach," Advanced Computing and Intelligent Technologies. Lecture Notes in Networks and Systems, Vol. 218, Springer, Singapore, 2022. https://doi.org/10.1007/978-981-16-2164-2_19.

[53] B. W. Silverman, and M. C. Jones "E. Fix and J.L. Hodges (1951): An Important Contribution to Nonparametric Discriminant Analysis and Density Estimation: Commentary on Fix and Hodges (1951)," International Statistical Review / Revue Internationale de Statistique Vol. 57, No. 3 (Dec. 1989), pp. 233-238. https://doi.org/10.2307/1403796.

[54] T. Cover, and P. Hart, "Nearest neighbor pattern classification," in IEEE Transactions on Information Theory, Vol. 13, No. 1, pp. 21-27, January 1967, doi: 10.1109/TIT.1967.1053964.

[55] Shahadat Uddin, Ibtisham Haque, Haohui Lu, Mohammad Ali Moni, and Ergun Gide, "Comparative performance analysis of K-nearest neighbour (KNN) algorithm and its different variants for disease prediction," Scientific Reports, Vol. 12, Article number: 6256, 2022. https://doi.org/10.1038/s41598-022-10358-x.

[56] Kristopher Kendall, "A database of computer attacks for the evaluation of intrusion detection systems," Master Thesis, Department of Electrical Engineering and Computer Science, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, June 1999. http://hdl.handle.net/1721.1/9459.

[57] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, "Cost-based modeling for fraud and intrusion detection: Results from the JAM project," Proceedings

DARPA Information Survivability Conference and Exposition. DISCEX'00, 2000, pp. 130-144 Vol. 2, doi: 10.1109/DISCEX.2000.821515.

[58] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set,” Computational Intelligence for Security and Defense Applications, CISDA 2009. IEEE Symposium on, 2009, pp. 1–6. doi: 10.1109/CISDA.2009.5356528.

[59] N. Moustafa and J. Slay, “UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set),” in 2015 Military Communications and Information Systems Conference (MilCIS), pp. 1–6, Canberra, ACT, Australia, November 2015. doi: 10.1109/MilCIS.2015.7348942.

[60] Iman Sharafaldin, Amirhossein Gharib, Arash Habibi Lashkari and Ali A. Ghorbani, “Towards a reliable intrusion detection benchmark dataset,” Software Networking, Vol. 2017, No. 1, pp. 177–200, 2018, doi: 10.13052/jsn2445-9739.2017.009.

[61] Yair Meidan, Michael Bohadana, Yael Mathov, Yisroel Mirsky, Asaf Shabtai, Dominik Breitenbacher, and Yuval Elovici, “N-BaIoT-Network-based detection of IoT botnet attacks using deep autoencoders,” IEEE Pervasive Computing, Vol. 17, No. 3, 8490192, pp. 12-22, 2018. https://doi.org/10.1109/MPRV.2018.03367731.

[62] Nickolaos Koroniotis, Nour Moustafa, Elena Sitnikova, and Benjamin Turnbull, “Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset,” Future Generation Computer Systems, Vol. 100, 2019, Pages 779-796, ISSN 0167-739X, https://doi.org/10.1016/j.future.2019.05.041.

[63] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, “Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications for Centralized and Federated Learning,” IEEE Access, Vol. 10, pp. 40281-40306, 2022, doi: 10.1109/ACCESS.2022.3165809.

[64] Caleb Belth, Xinyi Zheng, and Danai Koutra, “Mining Persistent Activity in Continually Evolving Networks,” In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20).

Association for Computing Machinery, New York, NY, USA, 934–944, 2020. https://doi.org/10.1145/3394486.3403136.

[65] Yen-Yu Chang, Pan Li, Rok Sosic, M. H. Afifi, Marco Schweighauser, and Jure Leskovec, “F-FADE: Frequency Factorization for Anomaly Detection in Edge Streams,” arXiv:2011.04723, 2021. doi: 10.48550/ARXIV.2011.04723.

[66] Siddharth Bhatia, Mohit Wadhwa, Kenji Kawaguchi, Neil Shah, Philip S. Yu, and Bryan Hooi, “Sketch-Based Anomaly Detection in Streaming Graphs,” arXiv, 2106.04486, 2021. https://doi.org/10.48550/arXiv.2106.04486.

[67] Satish Kumar, Sunanda Gupta, and Sakshi Arora, “A Comparative Simulation of Normalization Methods for Machine Learning-based Intrusion Detection Systems Using KDD Cup’99 Dataset,” Journal of Intelligent & Fuzzy Systems, vol. 42, No. 3, pp. 1749-1766, 2022. https://doi.org/10.3233/JIFS-211191.

[68] Santosh Kumar Sahu, Durga Prasad Mohapatra, Jitendra Kumar Rout, Kshira Sagar Sahoo, Quoc-Viet Pham, and Nhu-Ngoc Dao, “A LSTM-FCNN based multi-class intrusion detection using scalable framework,” Computers and Electrical Engineering, Vol. 99, 2022, 107720, ISSN 0045-7906, https://doi.org/10.1016/j.compeleceng.2022.107720.

[69] Amir Javadpour, Pedro Pinto, Forough Ja’fari, and Weizhe Zhang “A distributed multi-agent intrusion detection and prevention system for cloud IoT environments,” Cluster Computing, 2022. https://doi.org/10.1007/s10586-022-03621-3.

[70] Abhishek Raghuvanshi, Umesh Kumar Singh, Guna Sekhar Sajja, Harikumar Pallathadka, Evans Asenso, Mustafa Kamal, Abha Singh, and Khongdet Phasinam, “Intrusion Detection Using Machine Learning for Risk Mitigation in IoT-Enabled Smart Irrigation in Smart Farming,” Journal of Food Quality, Vol. 2022, Article ID 3955514, 8 pages, 2022. https://doi.org/10.1155/2022/3955514.

[71] Ilhan Firat Kilincer, Fatih Ertam, and Abdulkadir Sengur, “A comprehensive intrusion detection framework using boosting algorithms,” Computers and Electrical Engineering, Vol. 100, 2022, 107869, ISSN 0045-7906, https://doi.org/10.1016/j.compeleceng.2022.107869.

[72] Souradip Roy, Juan Li, Bong-Jin Choi, and Yan Bai, “A lightweight supervised intrusion detection mechanism for IoT networks,” Future Generation Computer

Systems, Vol. 127, 2022, pp. 276-285, ISSN 0167-739X, https://doi.org/10.1016/j.future.2021.09.027.

[73] P. G. V. Suresh Kumar, and Shaheda Akthar, “Execution Improvement of Intrusion Detection System Through Dimensionality Reduction for UNSW-NB15 Information,” Mobile Computing and Sustainable Informatics, Lecture Notes on Data Engineering and Communications Technologies, Vol. 68, Springer, Singapore, 2022. https://doi.org/10.1007/978-981-16-1866-6_28.

[74] Mohammad Humayun Kabir, Md Shahriar Rajib, Abu Saleh Md Towfiqur Rahman, Md. Mahbubur Rahman, and Samrat Kumar, “Network Intrusion Detection Using UNSW-NB15 Dataset: Stacking Machine Learning Based Approach,” International Conference on Advancement in Electrical and Electronic Engineering (ICAEEE), 2022, pp. 1-6, doi: 10.1109/ICAEEE54957.2022.9836404.

[75] Moutaz Alazab, Ruba Abu Khurma, Albara Awajan, and David Camacho, “A new intrusion detection system based on Moth–Flame Optimizer algorithm,” Expert Systems with Applications, Vol. 210, 2022, 118439, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2022.118439.

[76] Abdulnaser A. Hagar, and Bharti W. Gawali, “Apache Spark and Deep Learning Models for High-Performance Network Intrusion Detection Using CSE-CIC-IDS2018,” Computational Intelligence and Neuroscience, Vol. 2022, Article ID 3131153, 11 pages, 2022. https://doi.org/10.1155/2022/3131153.

[77] Mário Antunes, Luís Oliveira, Afonso Seguro, João Veríssimo, Ruben Salgado, and Tiago Murteira, “Benchmarking Deep Learning Methods for Behaviour-Based Network Intrusion Detection,” Informatics, Vol. 9, No. 1: 29, 2022. https://doi.org/10.3390/informatics9010029.

[78] Mohamed Hammad, Nabil Hewahi, and Wael Elmedany, “MMM-RF: A novel high accuracy multinomial mixture model for network intrusion detection systems,” Computers & Security, Vol. 120, 2022, 102777, ISSN 0167-4048, https://doi.org/10.1016/j.cose.2022.102777.

[79] Sawssen Bacha, Ahamed Aljuhani, Khawla Ben Abdellafou, Okba Taouali, Noureddine Liouane, and Mamoun Alazab, “Anomaly-based intrusion detection system in IoT using kernel extreme learning machine,” Journal of Ambient

Intelligence and Humanized Computing, 2022. https://doi.org/10.1007/s12652-022-03887-w.

[80] Jabed Al Faysal, Sk Tahmid Mostafa, Jannatul Sultana Tamanna, Khondoker Mirazul Mumenin, Md. Mashrur Arifin, Md. Abdul Awal, Atanu Shome, and Sheikh Shanawaz Mostafa, "XGB-RF: A Hybrid Machine Learning Approach for IoT Intrusion Detection," Telecom, Vol. 3, No. 1: 52-69, 2022. https://doi.org/10.3390/telecom3010003.

[81] Badr Lahasan, and Hussein Samma, "Optimized Deep Autoencoder Model for Internet of Things Intruder Detection," IEEE Access, Vol. 10, pp. 8434-8448, 2022, doi: 10.1109/ACCESS.2022.3144208.

[82] Tanzila Saba, Amjad Rehman, Tariq Sadad, Hoshang Kolivand, and Saeed Ali Bahaj, "Anomaly-based intrusion detection system for IoT networks through deep learning model," Computers and Electrical Engineering, Vol. 99, 2022, 107810, ISSN 0045-7906, https://doi.org/10.1016/j.compeleceng.2022.107810.

[83] Cristiano Antonio de Souza, Carlos Becker Westphall, and Renato Bobsin Machado, "Two-step ensemble approach for intrusion detection and identification in IoT and fog computing environments," Computers & Electrical Engineering, Vol. 98, 2022, 107694, ISSN 0045-7906, https://doi.org/10.1016/j.compeleceng.2022.107694.

[84] Babita Majhi, and Prastavana, "An Improved Intrusion Detection System using BoT-IoT Dataset," IEEE 11th International Conference on Communication Systems and Network Technologies (CSNT), 2022, pp. 488-492, https://doi.org/10.1109/CSNT54456.2022.9787639.

[85] Muna Al-Hawawreh, Elena Sitnikova, and Neda Aboutorab, "Asynchronous Peer-to-Peer Federated Capability-Based Targeted Ransomware Detection Model for Industrial IoT," IEEE Access, Vol. 9, pp. 148738-148755, 2021. doi: 10.1109/ACCESS.2021.3124634.

[86] Thi-Thu-Huong Le, Yustus Eko Oktian, and Howon Kim, "XGBoost for Imbalanced Multiclass Classification-Based Industrial Internet of Things Intrusion Detection Systems" Sustainability, Vol. 14, No. 14: 8707, 2022. https://doi.org/10.3390/su14148707.

[87] Rehab Alanazi, and Ahamed Aljuhani, “Anomaly detection for industrial internet of things cyberattacks,” Computer Systems Science and Engineering, Vol. 44, No.3, pp. 2361–2378, 2022. https://doi.org/10.32604/csse.2023.026712.

[88] Hudhaifa Mohammed Abdulwahab, S. Ajitha, and Mufeed Ahmed Naji Saif, “Feature selection techniques in the context of big data: taxonomy and analysis,” Applied Intelligence, Vol. 52, pp. 13568–13613, 2022. https://doi.org/10.1007/s10489-021-03118-3.

[89] Pudjihartono Nicholas, Fadason Tayaza, Kempa-Liehr Andreas W., and O'Sullivan Justin M., “A Review of Feature Selection Methods for Machine Learning-Based Disease Risk Prediction,” Frontiers in Bioinformatics, Vol. 2, 2022, ISSN: 2673-7647. doi:10.3389/fbinf.2022.927312.

[90] Pooja Chaudhary, Brij Gupta, and A. K. Singh, “Implementing attack detection system using filter-based feature selection methods for fog-enabled IoT networks,” Telecommunication Systems, Vol. 81, pp. 23–39, 2022. https://doi.org/10.1007/s11235-022-00927-w.

[91] Andrea Bommert, Thomas Welchowski, Matthias Schmid, and Jörg Rahnenführer, “Benchmark of filter methods for feature selection in high-dimensional gene expression survival data,” Briefings in Bioinformatics, Vol. 23, Issue 1, January 2022. bbab354, https://doi.org/10.1093/bib/bbab354.

[92] Jiao Hu, Wenyong Gui, Ali Asghar Heidari, Zhennao Cai, Guoxi Liang, Huiling Chen, and Zhifang Pan, “Dispersed foraging slime mould algorithm: Continuous and binary variants for global optimization and wrapper-based feature selection,” Knowledge-Based Systems, Vol. 237, 2022, 107761, ISSN 0950-7051, https://doi.org/10.1016/j.knosys.2021.107761.

[93] Elnaz Pashaei, and Elham Pashaei, “An efficient binary chimp optimization algorithm for feature selection in biomedical data classification,” Neural Computing and Applications, Vol. 34, pp. 6427–6451, 2022. https://doi.org/10.1007/s00521-021-06775-0.

[94] Saba Bashir, Irfan Ullah Khattak, Aihab Khan, Farhan Hassan Khan, Abdullah Gani, and Muhammad Shiraz, “A Novel Feature Selection Method for Classification of Medical Data Using Filters, Wrappers, and Embedded Approaches,” Complexity,

Vol. 2022, Article ID 8190814, 12 pages, 2022. https://doi.org/10.1155/2022/8190814.

[95] Tingquan Deng, Yang Huang, Ge Yang, and Changzhong Wang, "Pointwise mutual information sparsely embedded feature selection," International Journal of Approximate Reasoning, Vol. 151, 2022, pp. 251-270, ISSN 0888-613X, https://doi.org/10.1016/j.ijar.2022.09.012.

[96] Mohammed CHEMMAKHA, Omar HABIBI, and Mohamed LAZAAR, "Improving Machine Learning Models for Malware Detection Using Embedded Feature Selection Method," IFAC-PapersOnLine, Volume 55, Issue 12, 2022, Pages 771-776, ISSN 2405-8963, https://doi.org/10.1016/j.ifacol.2022.07.406.

[97] Meire P.M., and Ervynck, A., "Are oystercatchers (Haematopus ostralegus) selecting the most profitable mussels (Mytilus edulis)?," Animal Behaviour, Academic Press: London, ISSN 0003-3472; e-ISSN 1095-8282, 34: 1427-1435, 1986. http://dx.doi.org/10.1016/S0003-3472(86)80213-5.

[98] Kelly J. Brown, "Seasonal variation in the thermal biology of the rock hyrax (Procavia capensis)," M.Sc. Thesis, Pietermaritzburg, School of Botany and Zoology, University of KwaZulu-Natal, 2003. http://hdl.handle.net/10413/10124.

[99] Kelly J. Brown, and C. T. Downs, "Basking behavior in the rock hyrax (Procavia capensis) during winter," African Zoology, Vol. 42, No. 1, pp. 70–79, 2007. doi: 10.1080/15627020.2007.11407379.

[100] "Rock Hyrax," San Diego Zoo Wildlife Alliance Animals and Plants. [Online]. Available: https://animals.sandiegozoo.org/animals/rock-hyrax. [Last Accessed: 06-Nov-2022].

[101] Pushpa Singh, Narendra Singh, Krishna Kant Singh, and Akansha Singh, "Chapter 5 - Diagnosing of disease using machine learning," Machine Learning and the Internet of Medical Things in Healthcare, Academic Press, 2021, Pages 89-111, ISBN 9780128212295, https://doi.org/10.1016/B978-0-12-821229-5.00003-3.

[102] David Goldberg, "What every computer scientist should know about floating-point arithmetic," Association for Computing Machinery, New York, NY, USA, Vol. 23, No. 1, ISSN 0360-0300, 1991. https://doi.org/10.1145/103162.103163.

[103] Ivan Izonin, Roman Tkachenko, Nataliya Shakhovska, Bohdan Ilchyshyn, and Krishna Kant Singh, "A Two-Step Data Normalization Approach for Improving

Classification Accuracy in the Medical Diagnosis Domain," Mathematics, Vol. 10, No. 11: 1942, 2022. https://doi.org/10.3390/math10111942.

[104] "Google Colaboratory," Frequently Asked Questions, [Online]. Available: https://research.google.com/colaboratory/faq.html#:~:text=Colaboratory%2C%20or%20%E2%80%9CColab%E2%80%9D%20for,learning%2C%20data%20analysis%20and%20education. [Accessed: 06-Nov-2022].

PUBLICATIONS FROM THESIS

Conference Paper

[1] WEAM HUSHAM ALJABBARI, MUHAMMED ALİ AYDIN, and HASAN HÜSEYİN BALIK, “BotNets detection based on machine learning and features selection,” ORAL SUMMARY PRESENTATION, presented at the 4th International Congress of Engineering Sciences and Multidisciplinary Approaches, pp: 996, 03-04-05 NOVEMBER, 2022. ISTANBUL, Turkey.