

Virender Ranga

bhawanaThesis26May2026

 quick chk

Document Details

Submission ID

trn:oid:::27535:140504005

Submission Date

May 26, 2026, 2:08 PM GMT+5:30

Download Date

May 26, 2026, 2:12 PM GMT+5:30

File Name

bhawanaThesis26May2026.pdf

File Size

3.5 MB

88 Pages**21,276 Words****125,343 Characters**

2% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text
- Small Matches (less than 14 words)

Match Groups

-  **14 Not Cited or Quoted 2%**
Matches with neither in-text citation nor quotation marks
-  **0 Missing Quotations 0%**
Matches that are still very similar to source material
-  **0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 1%  Internet sources
- 1%  Publications
- 1%  Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- 14 Not Cited or Quoted 2%**
Matches with neither in-text citation nor quotation marks
- 0 Missing Quotations 0%**
Matches that are still very similar to source material
- 0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 1% Internet sources
- 1% Publications
- 1% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Publication	"Evolution in Signal Processing and Telecommunication Networks", Springer Scie...	<1%
2	Student papers	Central Institute of Technology Kokrajhar on 2026-05-18	<1%
3	Internet	eprints.utm.my	<1%
4	Internet	annals-csis.org	<1%
5	Internet	etheses.whiterose.ac.uk	<1%
6	Student papers	Delhi Technological University on 2026-05-24	<1%
7	Student papers	Tilburg University on 2026-05-24	<1%
8	Student papers	Kean University on 2025-04-21	<1%
9	Student papers	King Faisal University on 2025-05-13	<1%
10	Publication	Tran, Long Dinh. "Enhancing Telecom Churn Prediction: Adaboost with Oversamp...	<1%

11

Student papers

University of Portsmouth on 2024-05-20

<1%

12

Internet

theses.gla.ac.uk

<1%

ABSTRACT

The deployment of numerous connected devices in various areas, including as health care, home automation, transportation systems, and industry automation systems, has advanced significantly due to the of IoT technology advancement. The growing prevalence of IoT devices is linked to significant security risks that have surfaced as a result of different communication channels and cyberattacks. Traditional methods of intrusion detection mainly depend on the flow-based characteristics of network traffic and may not be able to detect some aspects of the device communication behaviour..

This thesis presents a framework using behaviour-based machine learning for identifying IoT Device also multi-class Anomaly Detection (AD) with joint evaluation capability. The work is carried out in two phases using the CICIoT-DIAD 2024 dataset. In Phase 1, a flow-based baseline model is developed for anomaly detection and attack classification. In Phase 2, behaviour-based traffic representations are utilized to perform both recognise device and anomaly detection tasks. Multiple machine learning algorithms are trained and evaluated. Experimental results show that behaviour-based features provide more discriminative information than traditional flow-based features. Random Forest outperformed all other models in overall performance for both DI and AD tasks.

For Device Identification, the Random Forest model achieved a classification accuracy of 99.24% on the test dataset, while the 5-fold cross-validation analysis produced an average accuracy of 99.15%, demonstrating stable and consistent performance across different folds. Similarly, for Anomaly Detection, the model attained an accuracy of accuracy of 99.72% on test, whereas the average 5-fold cross-validation accuracy was 99.68% with a very high AUC score, indicating strong generalization capability and robustness.

To improve processing efficiency and eliminate unnecessary information from the dataset, a combined feature refinement strategy using Mutual Information and Recursive Feature Elimination (RFE) is adopted. Rather than relying on all available attributes, this approach prioritizes the most informative features by filtering out less relevant ones, leading to a more compact feature space while maintaining comparable detection performance. In addition, a joint correctness evaluation is performed to analyze the capability of the proposed framework in simultaneously identifying IoT devices and classifying attack categories.

The overall findings demonstrate that behaviour-based traffic analysis can signif-

icantly improve IoT security performance while also enabling accurate device identification. The developed framework offers a practical and scalable mechanism for intelligent IoT security systems operating in realistic network environments.

Keywords: IoT Security, Intrusion detection , Device Identification, Anomaly Detection, Machine Learning, Behaviour-Based Features

LIST OF PUBLICATIONS

1. Bhawana Shakeet and Virender Ranga, “Machine Learning Based Anomaly Detection on the CICIOT-DIAD 2024 Dataset in IoT Environments,” in *10th International Conference on Information and Communication Technology for Competitive Strategies (ICTCS-2025)*, Jaipur, India, December 2025.
2. Bhawana Shakeet and Virender Ranga, “Behaviour-Based Device Identification and Anomaly Detection in IoT Networks Using Machine Learning,” in *International Conference on Signal, Systems, and Computing for Next-Gen Automation (ICSSCNA-2026)*, Eluru District, Andhra Pradesh, India, May 2026.

CONTENTS

Title	Page No.
Acknowledgement	ii
Candidate's Declaration	iii
Certificate	iv
Abstract	v
List of Publications	vii
List of Tables	x
List of Figures	xi
List of Algorithms	xii
List of Abbreviations	xiii
1 Introduction	1
1.1 Background and Motivation	1
1.2 Problem Statement	5
1.3 Objectives of the Study	6
1.4 Scope of the Work	6
1.5 Organisation of the Thesis	7
2 Literature Review	8
2.1 Overview	8
2.2 Related Work	8
2.3 Research Gaps	11
2.4 Summary	11
3 Methodology	12
3.1 Overview of Proposed Approach	12
3.2 Phase 1: Flow-Based Baseline	13
3.3 Phase 2: Behaviour-Based Framework	15
3.3.1 Device Identification process	16
3.3.2 Anomaly Detection process	18

3.3.3	Hybrid Feature Selection	20
3.3.4	Joint Evaluation Framework	21
3.4	Dataset and Experimental Setup	22
3.5	Evaluation	26
3.5.1	Performance Evaluation Metrics	27
3.5.2	Joint Correctness Evaluation	27
3.6	Summary	28
4	Results and Discussion	29
4.1	Experimental Results	29
4.1.1	Phase 1 Results: Flow-Based Baseline	29
4.1.2	Phase 2 Results: Device Identification (DI)	31
4.1.3	Phase 2 Results: Multi-Class Anomaly Detection (AD)	33
4.1.4	Phase 2 Results: Joint Correctness Evaluation	35
4.2	Discussion	37
4.3	Summary	38
5	Conclusion and Future Work	39
5.1	Conclusion	39
5.2	Future Work	40
	Bibliography	41
	Supplementary Data	46
	Source code	49
	Device Identification Source Code	49
	Multi-Class Anomaly Detection Source Code	57
	Hybrid Feature Selection for AD Source Code	64
	Joint Evaluation Source Code	69
	Proof of Publications	77
	Curriculum Vitae	80

LIST OF TABLES

3.1	Traffic Records Summary for Flow-Based Dataset	24
3.2	Traffic Records Summary for Behaviour-Based Dataset	25
4.1	ML models for binary classification (Attack vs Benign) using flow based feature for baseline	29
4.2	ML models on multi-class classification (7 attack types vs benign) using flow based feature for baseline	30
4.3	Phase 2: ML model evaluation for Device Identification (DI)	32
4.4	Five-Fold Cross-Validation Performance of the Random Forest Model for DI	33
4.5	Performance Comparison Before and After Feature Selection for Device Identification	33
4.6	ML model evaluation for Anomaly Detection (AD)	34
4.7	Five-Fold Cross-Validation Performance of the Random Forest Model for AD	34
4.8	Performance Comparison Before and After Feature Selection for Anomaly Detection	35
4.9	Summary Comparison of Flow-Based and Behaviour-Based Experimental Results	38
I.1	Flow-Based Traffic Features Used in the Phase 1 Baseline Model . . .	46
I.2	IoT Device Categories Used in Device Identification (DI)	47
I.3	Behaviour-Based Features Used in the Phase 2 Framework	48

LIST OF FIGURES

1.1	IoT Network	2
1.2	Common Security Vulnerabilities in IoT Environments	3
3.1	Overview	13
3.2	Workflow of the Flow-Based Baseline	14
3.3	Overall Workflow for phase 2	16
3.4	Model Selection and Evaluation Process for phase 2	20
3.5	Major Attack Categories in CICIoT-DIAD 2024 Dataset	23
4.1	Results for Binary class classification (Attack vs Benign) using flow-based features for baseline	30
4.2	Results for Multiclass classification (7 types of Attack vs Benign) using flow based feature for baseline	31
4.3	Phase 2: ML Classifiers for Device Identification with Behaviour based features	32
4.4	Phase 2: ML Classifiers for Anomaly Detection with Behaviour based features	34
4.5	Joint Correctness Evaluation of Device Identification and Anomaly Detection	36

LIST OF ALGORITHMS

1	Flow-Based Baseline Model Training and Evaluation	15
2	Behaviour-Based Device Identification Process	18
3	Behaviour-Based Multi-Class Anomaly Detection Process	19
4	Hybrid Feature Selection using MI and RFE	21
5	Joint Evaluation of Device Identification and Anomaly Detection . . .	22

LIST OF ABBREVIATIONS

IoT	Internet of Things
IDS	Intrusion Detection System
DI	Device Identification
AD	Anomaly Detection
ML	Machine Learning
RF	Random Forest
DT	Decision Tree
LR	Logistic Regression
NB	Naive Bayes
MLP	Multi-Layer Perceptron
XGB	Extreme Gradient Boosting
AdaBoost	Adaptive Boosting
RFE	Recursive Feature Elimination
MI	Mutual Information
AUC	Area Under Curve
ROC	Receiver Operating Characteristic
DDoS	Distributed Denial of Service
DoS	Denial of Service
CIC	Canadian Institute for Cybersecurity
RAM	Random Access Memory



CHAPTER 1

Introduction

1.1 Background and Motivation

The Internet of Things (IoT) has emerged as one of the most rapidly growing technologies in modern communication networks. IoT enables billions of interconnected devices to communicate and exchange information over the Internet using wired and wireless communication technologies [1,2,8,29,40]. These devices include smart sensors, wearable devices, healthcare monitoring systems, surveillance cameras, industrial controllers, smart appliances, and various embedded systems. The rapid growth of IoT technology has transformed traditional communication infrastructures into intelligent and automated environments capable of performing real-time monitoring, remote management, and data-driven decision making [2,4,8].

The continuous expansion of IoT environments has introduced significant improvements in automation, operational efficiency, and remote accessibility across multiple application domains. However, the large-scale deployment of heterogeneous IoT devices has also created serious security and privacy challenges [4,5,9,16,41,42]. Unlike traditional computer systems, IoT devices are generally lightweight and operate in highly distributed environments where a large number of devices continuously communicate and exchange sensitive information. Since many IoT devices are directly connected to public networks, they become highly exposed to cyber threats and malicious activities. As a result, securing IoT communication environments has become a major research challenge in recent years [5,9,23].

IoT systems are widely used in several real-world applications including smart homes, smart healthcare systems, industrial automation, transportation systems, environmental monitoring, agriculture, and smart cities [1–3,7,8,30]. In smart home environments, IoT devices such as smart televisions, smart bulbs, security cameras, smart switches and voice assistants enable intelligent home automation and remote device control. Smart healthcare systems utilize IoT-enabled wearable sensors and medical devices to continuously monitor patient health conditions and provide real-time medical assistance [5,30]. Similarly, industrial IoT environments support automated manufacturing systems, predictive maintenance, and intelligent monitoring applications. In transportation and smart city infrastructures, IoT technologies are used for traffic management, energy optimization, pollution monitoring, public safety, and intelligent

surveillance systems.

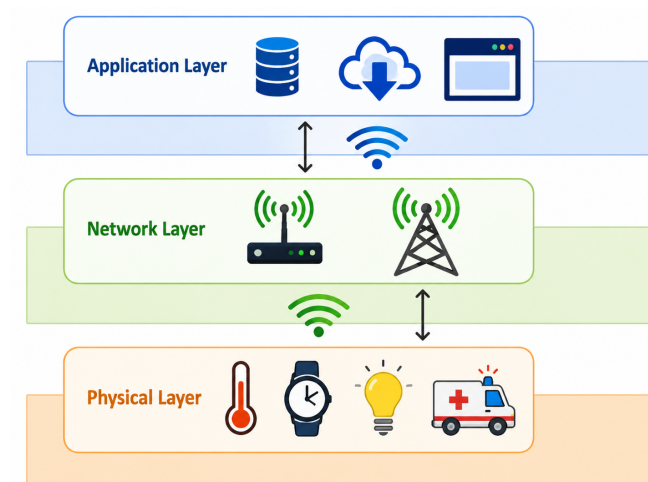


Figure 1.1: IoT Network

A typical IoT architecture consists of multiple interconnected layers responsible for sensing, communication, processing, and application management as shown in Figure 1.1. The sensing layer contains physical devices have sensors that collect environmental and operational information. The communication layer is responsible for transmitting collected data using communication protocols such as Wi-Fi, Bluetooth, ZigBee, MQTT, and cellular networks [2, 8]. The processing layer performs data storage, computation, and analytics using edge computing or –based infrastructures [5, 7]. Finally, the application layer delivers intelligent services and user-oriented applications based on the processed information [2, 3, 5, 7, 8]. Although this layered architecture improves automation and connectivity, it also introduces several security vulnerabilities owing to the involvement of multiple communication channels, distributed devices, and heterogeneous network protocols.

One of the principal security challenges is the resource-constrained nature of IoT devices. Most IoT devices are designed with limited computational power, restricted memory capacity, low battery availability, and lightweight operating systems [5, 6, 10, 11]. These hardware limitations make it difficult to implement complex security mechanisms such as sophisticated encryption schemes, multi-factor authentication systems, and computationally intensive intrusion detection techniques. As a result, many IoT devices are deployed with weak security configurations and insufficient protection mechanisms.

In many practical IoT deployments, devices are configured using default usernames and passwords, weak authentication mechanisms, insecure communication protocols,

and outdated firmware versions [12, 16, 39, 41, 42, 42]. Several devices also lack secure software update mechanisms and remain vulnerable to known security exploits for long periods of time. Furthermore, lightweight communication protocols commonly used in IoT environments often prioritize efficiency and low power consumption over strong security protections [5, 10, 12]. These limitations significantly increase the attack surface of IoT networks and create opportunities for attackers to exploit vulnerable devices and communication infrastructures.

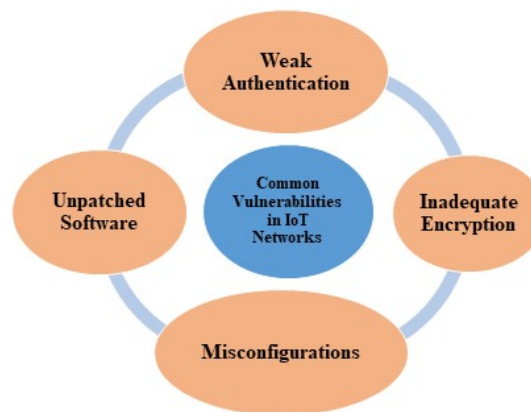


Figure 1.2: Common Security Vulnerabilities in IoT Environments

The growth in number of vulnerable IoT devices connected to public networks has resulted in a rapid rise in cyber-attacks targeting IoT infrastructures. IoT environments are highly susceptible to attacks such as Denial of Service (DoS), Distributed Denial of Service (DDoS), spoofing attacks, brute-force attacks, reconnaissance attacks, malware injection, web-based attacks, and botnet activities [9, 13–15, 28, 41, 42]. Attackers often exploit insecure devices to gain unauthorized access, steal sensitive information, interrupt communication services, or compromise entire IoT infrastructures.

Among these threats, Distributed Denial of Service (DDoS) attacks have become one of the most serious security concerns in IoT environments. In DDoS attacks, a large number of compromised devices simultaneously generate malicious traffic to overwhelm targeted systems and disrupt their services. Large-scale botnets such as Mirai demonstrated how insecure IoT devices can be hijacked and utilized to launch massive DDoS attacks against online platforms and communication infrastructures [15]. The Mirai botnets compromised thousands of IoT devices by exploiting weak authentication credentials and generated extremely high volumes of malicious traffic capable of disabling critical Internet services [2, 32]. Such attacks highlighted the urgent need for robust IoT security mechanisms capable of detecting malicious activities at an early stage.

To address these growing security challenges, Intrusion Detection Systems (IDS) have become an essential component of modern IoT security frameworks [6, 22, 23, 31, 34]. IDS techniques are designed to continuously monitor network activities and identify abnormal or malicious behaviour within communication environments [22, 23, 31]. Traditional IDS approaches mainly rely on flow-based traffic analysis using statistical traffic information such as flow duration, packet counts, byte counts, communication ports, and protocol-level characteristics [31, 34]. These approaches can effectively identify several network-level anomalies and attack patterns.

However, conventional traffic analysis approaches often fail to capture the detailed behavioural communication characteristics of IoT devices. Unlike traditional computing systems, IoT devices exhibit unique communication behaviours depending on their functionality and operational characteristics. For example, smart cameras continuously generate multimedia traffic streams, while smart plugs and environmental sensors produce lightweight periodic communication patterns. These behavioural characteristics can vary significantly across different device categories and application environments.

Behaviour-based traffic analysis focuses on extracting communication patterns, interaction behaviours, and device-specific traffic characteristics from network activities [17–20]. Such behavioural information can act as communication fingerprints for IoT devices and provide more discriminative insights compared to traditional flow-based traffic analysis. Behaviour-based analysis enables intrusion detection systems to better understand how devices interact with networks during normal operation and how these patterns change during malicious activities.

In recent years, behavioural fingerprinting and device identification techniques have gained significant attention in IoT security research. Device fingerprinting methods analyze communication behaviours to identify specific IoT device categories based on their traffic characteristics [17–20]. Accurate device identification can improve network visibility, device management, and security monitoring in large-scale IoT environments. In addition, behavioural traffic analysis can also improve anomaly detection performance by identifying deviations from normal device communication patterns.

Another important limitation observed in existing IoT security research is that device identification and attack detection are generally treated as separate problems. Several studies primarily focus on intrusion detection, while others mainly concentrate on IoT device profiling and fingerprinting [17–19]. However, in practical IoT environments, it is equally important to determine both the identity of the communicating device and whether its network behaviour is malicious. Simultaneous analysis of device identity and attack activity can provide more reliable and intelligent security monitoring for modern IoT networks.

Motivated by these challenges, this thesis proposes a behaviour-based machine learning framework for Device Identification (DI) and multi-class Anomaly Detection (AD) using the CICIoT-DIAD 2024 dataset [25, 26]. The proposed research is conducted in two phases. In the first phase, a flow-based baseline framework is developed for anomaly detection and multi-class attack classification using conventional traffic features. In the second phase, behaviour-based traffic representations are utilized to perform both identification of devices (DI) and anomaly detection tasks simultaneously.

The efficacy of several machine learning algorithms in Internet of Things security applications is assessed and contrasted, including Logistic Regression, Multi-Layer Perceptron (MLP), Naive Bayes, Decision Tree, XGBoost, Random Forest, and Adaboost. Additionally, a hybrid feature selection strategy based on Recursive Feature Elimination (RFE) and Mutual Information (MI) is used to preserve crucial traffic characteristics needed for precise classification while reducing feature dimensionality and increasing computing efficiency.

The proposed work aims to demonstrate that behaviour-based traffic analysis can provide more discriminative information than conventional traffic analysis approaches and can significantly improve the effectiveness of IoT security systems in realistic communication environments. The proposed framework also aims to provide a unified solution capable of simultaneously performing device identification and anomaly detection, thereby improving the reliability and intelligence of modern IoT security architectures.

1.2 Problem Statement

Most existing IoT intrusion detection systems primarily focus on flow-based traffic analysis for attack detection. While flow-based features provide network-level statistical information, they do not effectively capture the behavioural characteristics of IoT devices. As a result, such systems often fail to provide detailed understanding of device-specific communication behaviour.

In addition, many existing approaches are limited to binary anomaly detection, where traffic is normal or malicious. These approaches do not provide detailed multi-class attack classification. Furthermore, device identification and anomaly detection are generally treated as separate problems, which limits the effectiveness of current IoT security solutions.

Another important limitation is the absence of a unified evaluation strategy that can measure whether both device identification and attack classification are simultaneously correct. In real-world IoT networks, accurate prediction of both the device type and

attack type is necessary for reliable security monitoring and threat analysis. Therefore, there is a need for a unified behaviour-based machine learning framework that can:

- Perform Device Identification and multi-class Anomaly Detection together using behavioural traffic features for improved IoT security analysis.
- Evaluate the practical reliability of the framework through simultaneous assessment of Device Identification and Anomaly Detection predictions.

1.3 Objectives of the Study

As a result of rising complexities within IoT environments and increasing cyberattacks, there is a need for sophisticated intrusion detection systems to be put in place. Traditional flow-based approaches mainly focus on traffic-level statistics and often fail to capture the behavioural characteristics of IoT devices. In addition, many existing studies treat device identification and attack detection as separate tasks, which limits their effectiveness in practical IoT environments.

In order to overcome these difficulties, this study explores the creation of machine learning based on behavioral analysis for Device Identification(DI) and multi-class Anomaly Detection(AD). The suggested study also explores how various machine learning algorithms and feature selection methods can enhance detection capabilities, lower computational demands, and improve the overall dependability of IoT intrusion detection systems.

The major objectives of research work are as follows:

1. To develop a flow-based machine learning model as a baseline approach for anomaly detection and attack classification using the flow-based CICIOT-DIAD 2024 dataset.
2. To design a behaviour-based machine learning framework for Device Identification (DI) and multi-class attack classification in IoT networks using behavioural traffic features.

1.4 Scope of the Work

This study concentrates on employing machine learning for intrusion detection in Internet of Things (IoT) environments utilizing the CICIOT-DIAD 2024 dataset. The study includes both flow-based and behaviour-based feature datasets for security analysis.

In the first phase, a flow-based baseline model is developed for binary anomaly detection and multi-class attack classification. In the second phase, a behaviour-based framework is designed for Device Identification(DI) along with multi-class Anomaly Detection(AD).

The work includes comparative analysis of seven traditional machine learning algorithms and investigates the effectiveness of behaviour-based features for improving detection performance. A hybrid feature selection technique is also applied to reduce computational complexity and training time.

The scope of this work is confined to offline analysis of experimental work using supervised machine learning techniques and publicly available benchmark datasets. Real-time deployment and deep learning-based approaches are not considered within the scope of this research.

1.5 Organisation of the Thesis

The remainder of this thesis is organized as follows:

- **Chapter 2** presents a detailed review of existing literature related to IoT intrusion detection systems, behaviour-based security analysis, device identification methods, and machine learning techniques used for IoT security.
- **Chapter 3** describes the methodology, followed by dataset description, preprocessing steps, machine learning models, feature selection techniques, robustness evaluation, and the overall framework architecture.
- **Chapter 4** presents the experimental results and discussion for both flow-based and behaviour-based approaches, including comparative analysis, feature selection analysis, robustness evaluation, and joint correctness evaluation.
- **Chapter 5** devoted to the conclusion of this thesis by summarizing the significant findings of the research work and proposing probable future study possibilities.

The next chapter provides an overview of previous studies pertaining to IoT intrusion detection systems, attack detection using machine learning and device identification techniques. The chapter also highlights the research gaps and talks about the shortcomings of current methods.



CHAPTER 2

Literature Review

2.1 Overview

The rapid increasing deployment of Internet of Things(IoT) technology in applications such as smart homes, health, industrial automation, transportation, and smartcities has significantly increased the importance of IoT security. Due to the large number of interconnected devices and their limited computational capabilities, IoT networks are highly vulnerable to various cyber-attacks [16,26,39,41,42]. To address these security challenges, Intrusion Detection Systems (IDS) have become an important research area in IoT environments [1, 15].

Traditional IoT intrusion detection approaches mainly rely on flow-based traffic features for detecting malicious activities. These approaches use network-level statistical information such as packet count, flow duration, protocol type, and byte count to identify abnormal traffic patterns. Although flow-based methods are computationally efficient and widely used, they often fail to capture the behavioural communication characteristics of IoT devices [19].

In recent years, behaviour-based traffic analysis has become an important area of research in IoT security because different IoT devices communicate in different ways depending on their functionality and usage behaviour. These behavioural communication patterns provide valuable information about device activities, which can help in identifying IoT devices as well as detecting malicious behaviour in the network. At the same time, machine learning techniques have been widely adopted in IoT intrusion detection systems because they can effectively learn complex traffic patterns and improve detection performance in dynamic IoT environments [22].

This chapter provides an overview of previous studies pertaining to IoT intrusion detection systems(IDS) with machine learning-based attack detection, behaviour-based analysis, and device identification techniques.The chapter also highlights the research gaps this thesis fills and talks about the shortcomings of current methods.

2.2 Related Work

The growth in Internet of Things (IoT) devices has significantly increased security concerns related to unauthorized device access, malicious traffic generation, and large-scale cyber-attacks. As a result, Device Identification (DI) and Anomaly Detection

(AD) have emerged as two important research directions for securing IoT environments. Numerous studies have examined deep learning and machine learning for recognising IoT devices and detecting anomalous network behaviour using traffic-based features.

Early studies on IoT device identification demonstrated that statistical traffic features can effectively distinguish IoT devices even in encrypted communication environments. Basha *et al.* [21] proposed a deep-learning-based device fingerprinting framework that utilizes multiple traffic data streams for robust IoT device identification. The study demonstrated that behavioural communication patterns can improve fingerprinting performance and provide resilience against varying network channel conditions.

Meidan *et al.* [19] proposed Profiliot, a machine learning-based framework that identifies IoT devices using network traffic behaviour and flow statistics. Similarly, Marchal *et al.* [17] presented an off-path device fingerprinting technique for identifying IoT devices through encrypted traffic analysis. These studies established the effectiveness of behaviour-oriented traffic analysis for IoT device identification without relying on payload inspection.

In recent years, anomaly detection techniques for IoT environments have been widely investigated using statistical approaches, clustering techniques, and supervised machine learning models. Ahmad *et al.* [31] investigated various deep learning and machine learning techniques for intrusion detection in Internet of Things settings. Their research made clear how important it is for intrusion detection systems to have good detection performance while lowering false alarm rates. To detect malicious network activity in IoT traffic, the scientists assessed a number of methods, such as Support Vector, Random Forest, recurrent neural networks, and autoencoders. Basha *et al.* [21] proposed a deep-learning-based device fingerprinting framework that utilizes multiple traffic data streams for robust IoT device identification. The study demonstrated that behavioural communication patterns can improve fingerprinting performance and provide resilience against varying network channel conditions. Sheng *et al.* [27] presented a comprehensive survey on network traffic fingerprinting techniques for Industrial Internet of Things (IIoT) device identification. The study analyzed different deep learning approaches used for traffic-based device fingerprinting and discussed the challenges associated with heterogeneous IIoT environments.

Inayat *et al.* [28] provided a comprehensive survey on intrusion detection based machine learning and deep learning techniques for IoT, focusing on attacks such as spoofing, webbased, DDOS, botnet, and Man-in-the-Middle (MITM) attacks. The authors highlighted major challenges including lack of large-scale benchmark datasets, feature

selection complexity, and the overhead of computation associated with advanced deep learning models.

To address the limitation of insufficient IoT security datasets, Neto et al. [29] introduced the CICIoT2023 dataset, which contains realistic IoT network traffic generated from 105 IoT devices under multiple attack scenarios. The dataset includes seven major attack categories and has become an important benchmark dataset for evaluating IoT intrusion detection systems for IoT environment using machine learning approaches.

Several researchers have also explored hybrid and behaviour-based approaches for improving IoT security analysis. Rabbani et al. [26] proposed a hybrid framework that combines behaviour-based and flow-based features for simultaneous device recognition and anomaly detection using the CICIoT2023 dataset. Their work demonstrated that integrating behavioural communication characteristics with flow statistics can improve IoT traffic analysis performance.

The security requirements of large-scale IoT applications have also been extensively studied in domains such as healthcare and smart cities. Chataut et al. [30] analyzed the security challenges in IoT deployments and emphasized the need for scalable and reliable anomaly detection mechanisms capable of handling heterogeneous IoT environments.

Recently, explainable artificial intelligence (XAI) techniques have drawn most attention in detecting IoT intrusion research. Hosain et al. [32] proposed an explainable XAI-XGBoost-based intrusion detection framework for Internet of Medical Things (IoMT) environments, improving both detection accuracy and interpretability. Similarly, Akif et al. [34] investigated standalone and hybrid machine learning and deep learning models for both binary and multi-class anomaly detection tasks and demonstrated that hybrid models can provide an effective balance between computational efficiency and detection performance.

Although existing studies have demonstrated promising results for either device identification or anomaly detection individually, only limited research has focused on jointly evaluating both tasks within a unified IoT security framework. In addition, many existing approaches lack a system-level evaluation strategy that simultaneously measures the correctness of both device identification and attack classification. Furthermore, several existing studies primarily rely on older datasets or focus only on binary anomaly detection tasks.

Motivated by these limitations, the present work performs a behaviour-based joint evaluation of Device Identification and multi-class Anomaly Detection using the CICIoT-DIAD 2024 dataset. Unlike previous studies, this work evaluates the simultaneous

correctness of both tasks using a common test set and analyzes the effectiveness of behavioural communication features for device-aware intrusion detection in IoT environments.

2.3 Research Gaps

Although IoT intrusion detection has advanced recently, several challenges still exist. Most existing approaches mainly rely on flow-based traffic features, which capture network-level statistics but often fail to represent the behavioural characteristics of IoT devices effectively [19]. In addition, many studies focus only on binary anomaly detection and do not provide detailed multi-class attack classification.

Another important limitation is that Device Identification (DI) and Anomaly Detection (AD) are usually treated as separate tasks rather than being performed within a unified framework. Existing methods also often use high-dimensional feature sets, which increase computational complexity and training time [39]. Furthermore, very few studies evaluate whether both DI and AD predictions are simultaneously correct for the same traffic sample.

To solve these restrictions, this research presents a work on behaviour-based machine learning for Device Identification and multi-class Anomaly Detection along with joint correctness evaluation.

2.4 Summary

This chapter reviewed existing literature related to IoT intrusion detection systems, machine learning (ML) for attack detection, behaviour-based traffic analysis, and device identification techniques. The discussion highlighted the increasing importance of behaviour-based analysis for improving IoT security and increase utilization of machine learning for intrusion detection tasks. The chapter also identified several research gaps in existing studies, including the limited use of behavioural features, the separate treatment of device identification and attack detection, the lack of unified evaluation strategies, and the high computational complexity associated with large feature sets.

Based on these observations, the proposed research aims to develop a behaviour-based framework for Device Identification and multi-class Anomaly Detection with joint evaluation and improved detection performance.

The next chapter describes the proposed methodology, including dataset description, preprocessing processes, machine learning models, and feature selection strategies employed in this work.

CHAPTER 3

Methodology

This chapter presents the complete methodology chosen for developing the proposed framework for IoT intrusion detection based on machine learning using the CICIoT-DIAD 2024 dataset. The suggested framework is structured into two major phases in order to systematically analyze flow-based and behaviour-based traffic representations for different IoT security tasks. The first phase focuses on building a flow-based baseline for anomaly detection and attack classification, while the second phase presents the proposed behaviour-based framework for Device Identification (DI) and multi-class Anomaly Detection (AD).

The chapter also describes the preprocessing techniques, machine learning models, mathematical formulation, dataset characteristics, experimental setup, and evaluation strategy used throughout the research work.

3.1 Overview of Proposed Approach

The proposed framework follows a two-phase methodology for IoT intrusion detection using both flow-based and behaviour-based traffic representations. In the first phase, a flow-based baseline model is developed using the flow-based CICIoT-DIAD 2024 dataset. This phase focuses on anomaly detection tasks using traditional network flow features. Two classification tasks are performed in this phase: binary anomaly detection, where traffic is either as benign or attack traffic, and multi-class attack classification, where different attack categories are identified. The main intention is to establish a baseline model for performance comparison.

In the second phase, a behaviour-based machine learning framework is developed using the behaviour-based feature dataset of CICIoT-DIAD 2024. The behaviour-based dataset supports both Device Identification (DI) and multi-class Anomaly Detection (AD) tasks. In this machine learning models are trained and evaluated for identifying IoT devices as well as classifying different attack categories based on behavioural communication features. The behaviour-based framework is further evaluated and compared with the flow-based baseline approach to analyze the effectiveness of behavioural traffic features for IoT intrusion detection.

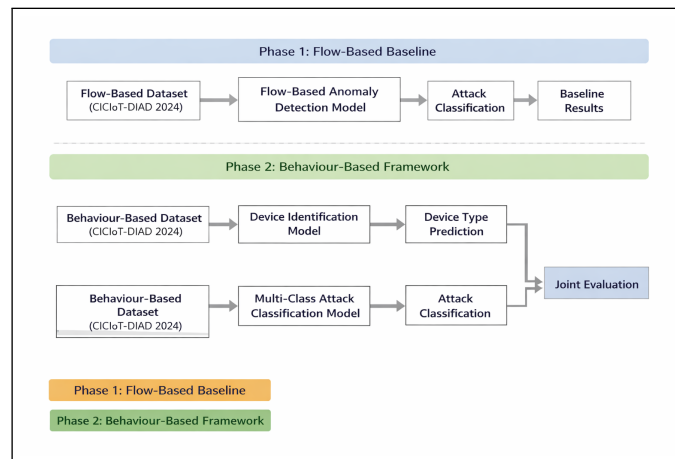


Figure 3.1: Overview

3.2 Phase 1: Flow-Based Baseline

The first phase of the proposed methodology focuses on developing a flow-based baseline for IoT intrusion detection using the flow-based feature dataset of CICIoT-DIAD 2024. The main purpose of this phase is to establish a reference model for comparing the performance of the proposed behaviour-based framework presented in Phase 2.

The entire process of the flow-based baseline is illustrated in Figure 3.2. This includes preprocessing, machine learning model training, performance evaluation, and baseline model selection steps. The flow-based dataset contains 84 columns, including 83 network flow-based traffic features and a single target label. These features mainly represent statistical properties of network traffic such as packet length, flow duration, packet rate, inter-arrival time, flag information, and active-idle traffic behaviour.

The workflow adopted for flow based baseline is presented in Algorithm 1. Initially, the dataset is preprocessed using data cleaning, encoding labels, feature scale, and stratified train-test separation techniques. To maintain the initial class distribution a stratified 80:20 split is employed. Standard Scaling is applied before model training to normalize feature values and improve learning performance.

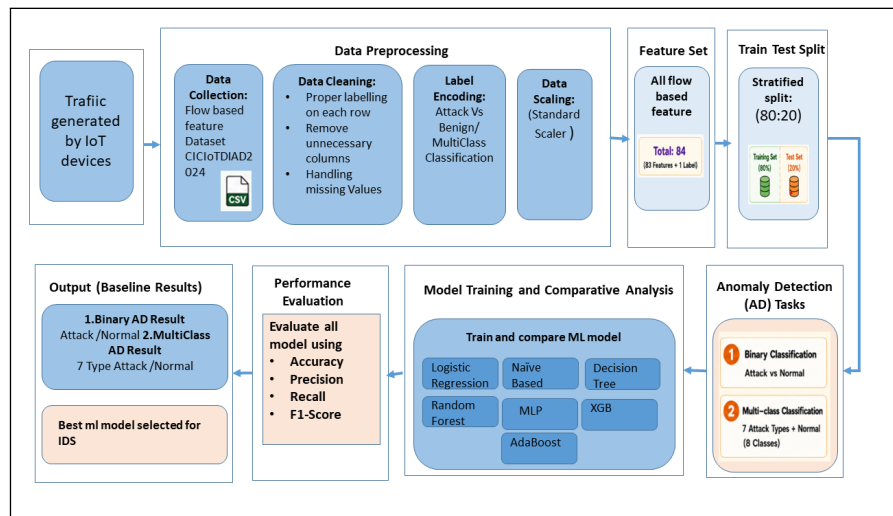


Figure 3.2: Workflow of the Flow-Based Baseline

After preprocessing, multiple machine learning algorithms including Naive Bayes, Logistic Regression, Decision Tree, Random Forest, MLP, AdaBoost and XGBoost are trained and evaluated using the complete flow-based feature set [35–37]. No feature selection technique is applied in this phase because the primary objective is analyse the overall performance of different machine learning models using the original feature space. The first task performs binary anomaly detection, where traffic samples are classified as either normal traffic or attack traffic. The second task performs multi-class attack classification, where multiple attack categories are identified individually.

The trained models are further evaluated using standard classification metrics such as accuracy with precision, F1-score and recall.

Based on comparative performance analysis the best-performing machine learning model is taken as the baseline model for further comparison with the proposed behaviour-based framework developed in Phase 2.

Algorithm 1 Flow-Based Baseline Model Training and Evaluation

Require: Flow-based dataset with 83 features and one target label

Ensure: Best-performing baseline machine learning model

- 1: Load CICIoT-DIAD 2024 flow-based dataset
 - 2: Perform preprocessing: Data cleaning, Label encoding, Standard Scaling, and stratified 80:20 train-test split
 - 3: Define classification tasks: Binary anomaly detection and multi-class attack classification
 - 4: Initialize machine learning models chosen of comparison
 - 5: **for** each machine learning model **do**
 - 6: Train model using training dataset
 - 7: Generate predictions on testing dataset
 - 8: Compute: Accuracy, Precision, Recall, and F1-score
 - 9: Store classification results
 - 10: **end for**
 - 11: Compare performance of all trained models
 - 12: Select best-performing model as baseline framework
 - 13: Generate comparative analysis for (i)anomaly detection (ii)multi-class attack classification
-

3.3 Phase 2: Behaviour-Based Framework

The second phase of the proposed methodology focuses on developing a machine learning framework using behaviour-based feature dataset of the CICIoT-DIAD 2024 dataset. The behaviour-based dataset supports both Device Identification (DI) and multi-class Anomaly Detection (AD) tasks. This phase mainly focuses on analysing behavioural communication patterns of IoT with attack detection so that enhance the intrusion detection performance.

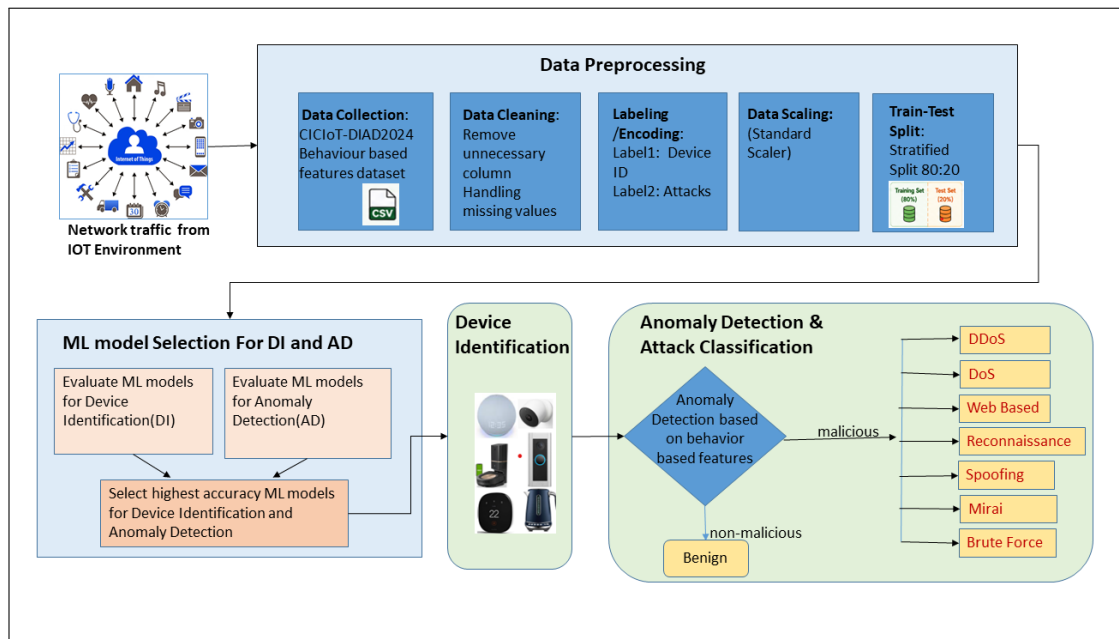


Figure 3.3: Overall Workflow for phase 2

Initially, separate feature sets are prepared for Device Identification(DI) and Anomaly Detection(AD) processes. The Device Identification task uses behavioural communication features associated with different IoT device categories, while the Anomaly Detection task uses behavioural traffic features associated with multiple attack categories and normal traffic.

The dataset is preprocessed and evaluated using multiple machine learning models for both Device Identification (DI) and Anomaly Detection (AD). Model performance is assessed using standard evaluation measures, while robustness is further examined through cross-validation. To improve feature relevance and reduce redundancy, a hybrid feature selection approach based on Mutual Information (MI) and Recursive Feature Elimination (RFE) is employed [39,43].

Finally, the proposed framework is evaluated using confusion matrices, classification reports, cross-validation analysis, and joint correctness evaluation. The performance of the behaviour-based framework is also compared with the flow-based baseline model for analyze the overall reliability of the proposed framework.

3.3.1 Device Identification process

The Device Identification process is developed to identify IoT devices based on their behavioural communication patterns. In this task, predefined behavioural features are extracted from the behaviour-based dataset and used train model and evaluate model.

The Device Identification (DI) process is carried out using the behaviour-based dataset of the CICIoT-DIAD 2024 framework.

The workflow adopted for the Device Identification process is presented in Algorithm 2.

At the beginning, the dataset is loaded, and the behavioural features required for identifying IoT devices are selected. Before training the models, preprocessing is done to prepare the data for the learning process. This involves using label encoding in order to turn device labels into numbers and feature scaling to bring down all the input features to a comparable scale. Afterwards, an 80/20 train test split is done to ensure that both the training and testing dataset have balanced categories of devices. In order to assess the effectiveness of various learning algorithms, we use different machine learning models. They are trained using the training dataset and evaluated using the testing data. Evaluation is done based on accuracy, F1 score, precision, and recall metrics.

After selecting the best-performing model, five-fold cross-validation is performed to examine its stability and consistency across different data partitions. To further improve computational efficiency and remove less informative attributes, a hybrid feature selection approach based on Mutual Information (MI) followed by RFE is applied. The optimized feature subset obtained through this process is then used to retrain the selected model. At last, the performance of the optimized Device Identification is evaluated to analyze its capability in identifying IoT device categories more effectively.

Algorithm 2 Behaviour-Based Device Identification Process

Require: Behaviour-based dataset with DI features

Ensure: Optimized Device Identification model and evaluation results

- 1: Load behaviour-based features dataset of CICDIAD2024
 - 2: Select predefined behavioural features for Device Identification
 - 3: Perform preprocessing:
 - Label encoding, Standard Scaling, Stratified 80:20 train-test split
 - 4: Initialize machine learning models:
 - (i) LR,(ii) NB, (iii)RF , (vi)DT, (v)AdaBoost, (vi) XGBoost,(vii)MLP
 - 5: **for** each machine learning model **do**
 - 6: Train model using training dataset
 - 7: Generate predictions on testing dataset
 - 8: Compute evaluation metrics
 - 9: Store classification results
 - 10: **end for**
 - 11: Compare performance of all trained models
 - 12: Select best-performing model for further optimization
 - 13: Perform 5-fold cross-validation on selected model
 - 14: Apply hybrid feature selection on selected model
 - 15: Obtain optimized DI feature subset
 - 16: Retrain selected model using optimized features
 - 17: Generate final Device Identification evaluation results
-

3.3.2 Anomaly Detection process

The Anomaly Detection process focuses on identifying malicious traffic and classifying different attack categories using behavioural traffic features. The behaviour-based dataset comprises multiple attack classes along with normal traffic samples for the AD task.

The multi-class Anomaly Detection (AD) process follows a methodology similar to the Device Identification framework. The workflow adopted for the multi-class Anomaly Detection framework is presented in Algorithm 3. The behaviour-based training and testing datasets are first loaded, and the predefined behavioural features related to anomaly detection are selected. Data preprocessing is then performed, including label encoding and feature scaling, followed by a stratified 80:20 train-test split to preserve the distribution of attack categories and benign traffic samples.

Several machine learning models are trained and evaluated for multi-class attack

classification. Based on the obtained results, the top performer is selected for further analysis.

Algorithm 3 Behaviour-Based Multi-Class Anomaly Detection Process

Require: Behaviour-based dataset with predefined AD features

Ensure: Optimized multi-class Anomaly Detection model and evaluation results

- 1: Load behaviour-based features datasets CICDIAD2024
 - 2: Select behavioural features for multi-class Anomaly Detection
 - 3: Perform preprocessing:
 - Label encoding, Standard Scaling, Stratified 80:20 train-test split
 - 4: Initialize machine learning models:
 - (i) LR, (ii) NB, (iii) RF, (vi) DT, (v) AdaBoost, (vi) XGBoost, (vii) MLP
 - 5: **for** each machine learning model **do**
 - 6: Train model using training dataset
 - 7: Generate predictions on testing dataset
 - 8: Compute all evaluation metrics
 - 9: Store classification results
 - 10: **end for**
 - 11: Compare performance of all trained models
 - 12: Select best-performing model for further optimization
 - 13: Perform 5-fold cross-validation on selected model
 - 14: Apply hybrid feature selection on selected model
 - 15: Obtain optimized AD feature subset
 - 16: Retrain selected model using optimized features
 - 17: Generate final multi-class Anomaly Detection evaluation results
-

To examine model reliability, five-fold cross-validation is performed on the selected classifier. A hybrid feature selection method based on Mutual Information with RFE is then applied to identify the most informative behavioural features for anomaly detection. The selected model is retrained using the optimized feature subset, and the final multi-class anomaly detection performance is evaluated to analyze its effectiveness in distinguishing different attack categories and benign traffic. The overall process of machine learning model selection, hybrid feature selection, and final joint evaluation used in Phase 2 is shown in Figure 3.4.

The overall process of machine learning model selection, hybrid feature selection, and final joint evaluation used in Phase 2 is depicted in Figure 3.4.

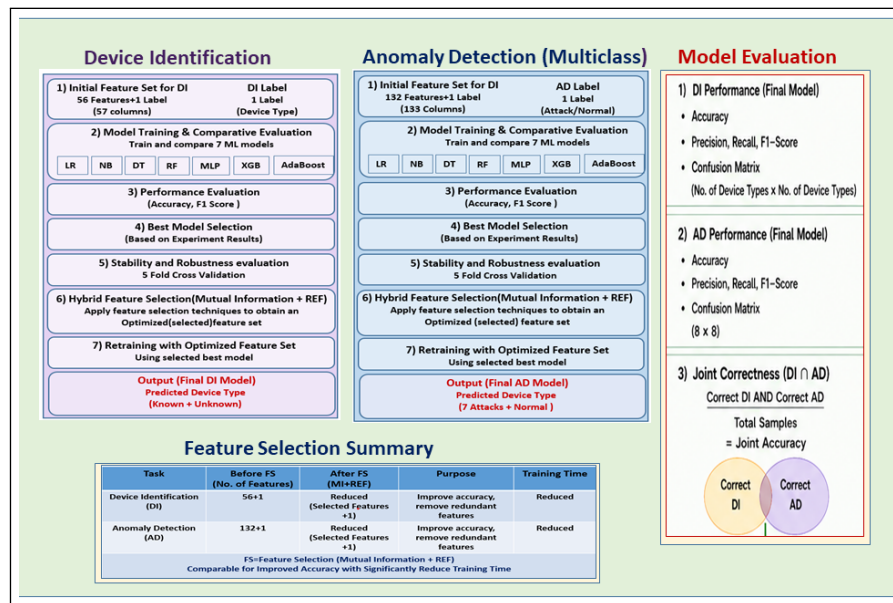


Figure 3.4: Model Selection and Evaluation Process for phase 2

Based on results of comparative analysis, the best ML algorithm is chosen as the final model for both DI and AD tasks.

3.3.3 Hybrid Feature Selection

A hybrid feature selection approach based on Mutual Information (MI) and Recursive Feature Elimination (RFE) is employed in this research to reduce feature redundancy and improve computational efficiency [38, 43]. Feature selection plays an important role in handling high-dimensional IoT traffic data and identifying the most informative features for machine learning-based security analysis.

Initially, Mutual Information is applied to gauge the dependency between individual input features and the target labels. Features with higher relevance scores contain more discriminative information related to Device its anomalous behaviour. Based on the computed scores, less informative features are filtered out before further optimization.

After the initial filtering stage, Recursive Feature Elimination is utilized to further refine the feature subset. RFE iteratively removes less significant features using the importance scores generated by the learning model. This iterative process finds most relevant traffic and behavioural features required for accurate classification.

The hybrid feature selection process is performed using the complete training dataset in order to capture the full traffic distribution and behavioural characteristics present in the CICIOT-DIAD 2024 dataset. The test dataset is not utilized during feature selection

to avoid data leakage and ensure fair model evaluation.

Algorithm 4 Hybrid Feature Selection using MI and RFE

Require: Behaviour-based DI and AD feature sets

Ensure: Optimized feature subsets

- 1: Apply Mutual Information (MI)
 - 2: `MI_Scores = mutual_info_classif(X, y)`
 - 3: Rank features based on importance
 - 4: Select highly relevant features
 - 5: Apply Recursive Feature Elimination (RFE)
 - 6: `rfe = RFE(estimator=RandomForestClassifier())`
 - 7: `rfe.fit(X, y)`
 - 8: Generate optimized feature subsets
 - 9: Obtain final optimized features:
 - 10: DI features reduced
 - 11: AD features reduced
-

The proposed hybrid approach combines the advantages of filter-based and wrapper-based techniques. Mutual Information provides fast relevance-based filtering, while Recursive Feature Elimination performs iterative optimization of the selected feature subset. This combined strategy improves computational efficiency while preserving important traffic characteristics required for IoT security analysis.

3.3.4 Joint Evaluation Framework

In addition to individual evaluation of Device Identification and Anomaly Detection models, the proposed framework also performs joint evaluation to analyse overall framework reliability.

In this process, a prediction is considered correct only when both the Device Identification prediction and the corresponding Anomaly Detection prediction are simultaneously correct for the same traffic sample.

Algorithm 5 Joint Evaluation of Device Identification and Anomaly Detection

Require: DI predictions, AD predictions, and true labels

Ensure: Joint Correctness Score

- 1: Generate final predictions
 - 2: `di_pred = rf_di.predict(X_test_di)`
 - 3: `ad_pred = rf_ad.predict(X_test_ad)`
 - 4: Compare predictions with true labels
 - 5: `di_correct = (di_pred == y_test_di)`
 - 6: `ad_correct = (ad_pred == y_test_ad)`
 - 7: Identify jointly correct samples
 - 8: `joint_correct = (di_correct & ad_correct)`
 - 9: Count jointly correct predictions
 - 10: `joint_count = np.sum(joint_correct)`
 - 11: Compute Joint Correctness Score
 - 12:
$$\text{Joint Correctness} = \frac{\text{Number of jointly correct predictions}}{\text{Total test samples}}$$
 - 13: Report overall framework reliability
-

The final joint correctness score is computed using the total number of jointly correct predictions and the total number of testing samples. This evaluation provides a more realistic analysis of the practical performance of suggested framework in IoT environments.

3.4 Dataset and Experimental Setup

This study's experiments are based on the CICIoT-DIAD 2024. The Canadian Institute for Cybersecurity (CIC) created this dataset. The dataset is specifically designed for IoT intrusion detection research and have both flow-based and behaviour-based traffic representations collected 105 devices of different 45 types of devices from realistic IoT network environments. The dataset includes multiple IoT device categories, normal traffic, and different cyber-attack classes, making it suitable for evaluating machine learning-based IoT security frameworks.

The dataset contains a total of 33 different attack types generated from realistic IoT network environments as shown in Figure 3.5. For effective analysis and multi-class classification, these attacks are broadly categorized into seven major attack types, namely Denial of Service, Distributed Denial of Service, Web-Based attacks, Flooding

attacks, Brute Force attacks, Mirai attacks, and Reconnaissance attacks. These attack categories represent diverse malicious behaviours commonly observed in IoT environments and are included in both the flow-based and behaviour-based datasets used in this research.

All experiments conducted in this research were performed on a high-performance computing system equipped with 32 CPU cores and 512 GB RAM. The large computational resources enabled efficient model training, and evaluation on the large-scale CICIoT-DIAD 2024 dataset.

Python and machine learning libraries like Pandas, NumPy, XGBoost and Scikit-Learn were utilised to create the suggested framework. Several algorithms for machine learning were assessed using common performance criteria, including AUC, F1-Score, Accuracy, Precision, and Recall.

DDoS	DoS	Recon	Mirai	Spoofing	Web based	BruteForce
• ACK fragmentation • UDP flood • SlowLoris • ICMP flood • RSTFIN flood • PSHACK flood • HTTP flood • UDP fragmentation • TCP flood • SYN flood • Synonymous IP flood	• TCP flood • HTTP flood • SYN flood • UDP flood	• Ping sweep • OS scan • Vulnerability scan • Port scan • Host discovery	• GREIP flood • GREETH flood • UDPPPlain	• ARP spoofing • DNS spoofing	• SQL injection • Command injection • Backdoor malware • Uploading attack • Cross-Site Scripting (XSS) • Browser hijacking	• Dictionary

Figure 3.5: Major Attack Categories in CICIoT-DIAD 2024 Dataset

In Phase 1, the flow-based feature dataset of CICIoT-DIAD 2024 is used to develop the baseline intrusion detection. The flow-based dataset contains 84 columns, including 83 network flow features and target label column as shown in Table I.1. These features mainly represent statistical traffic-level information such as byte counts, protocol information, packet counts, flow duration, and inter-arrival times.

The flow-based dataset is used for two different classification tasks. The first task performs binary anomaly detection, where traffic samples are classified as either benign traffic or attack traffic. The second task performs multi-class attack classification, where multiple attack categories are identified separately.

Table 3.1 presents the traffic records in the training and testing sets used in this research.

Table 3.1: Traffic Records Summary for Flow-Based Dataset

Traffic Type	Records in Train Set	Records in Test Set	Total Records
Benign	318553	79645	398198
DDoS	2781524	695369	3476893
DoS	18446189	4611364	23057553
Reconnaissance	352834	88197	441031
Mirai	139618	34900	174518
Spoofing	125706	31428	157134
Web Attack	9058	2267	11325
BruteForce	2895	724	3619
Total	22176377	5543894	27720271

It can be observed from Table 3.1 that the dataset is highly imbalanced, where DoS and DDoS attacks constitute the majority of traffic records, while Web Attack and Brute Force categories contain comparatively fewer samples. Such imbalance reflects realistic IoT attack scenarios and increases the complexity of intrusion detection and multi-class attack classification tasks.

Before model training, preprocessing operations such as removal of invalid values, label encoding, feature normalization using Standard Scaling, and stratified 80:20 train-test splitting are applied. Since the dataset contains imbalanced class distributions, stratified splitting is used to preserve the proportion of traffic type in both training and testing datasets.

The best-performing model from this phase is selected for comparison with the behaviour-based approach used in Phase 2.

In Phase 2, the behaviour-based feature dataset of CICIoT-DIAD 2024 is used for Device Identification (DI) and multi-class Anomaly Detection (AD).

The behavior-based dataset comprises features extracted by means of packet-per-packet analysis of IoT traffic . These features capture detailed communication behaviour and interaction characteristics of IoT devices during network communication as shown in Table I.3. Unlike flow-based representations, behaviour-based features provide finer-grained information related to device activity patterns, making them more suitable for simultaneous Device Identification (DI) and Anomaly Detection (AD) tasks. While flow-based features represent aggregated communication flow characteristics, behaviour-based features capture packet-level communication behaviour and device interaction patterns more effectively.

The behaviour-based dataset includes traffic generated from multiple IoT devices

under both benign and attack scenarios. These features represent behavioural communication patterns such as packet timing characteristics, payload-related statistics, jitter, inter-arrival time, and device interaction behaviour, making them suitable for identifying device-specific communication characteristics and malicious activities.

Table 3.2: Traffic Records Summary for Behaviour-Based Dataset

Traffic Type	Records in Train Set	Records in Test Set	Total Records
BenignTraffic	946237	236561	1182798
DDoS	17976093	4494071	22470164
DoS	2802927	700753	3503680
Reconnaissance	918766	229692	1148458
Mirai_greip	4877767	1219452	6097219
Spoofing	707057	176766	883823
Webbased	203119	50783	253902
BruteForce	105181	26296	131477
Total	28537147	7134374	35671521

Table 3.2 presents the traffic distribution of the behaviour-based CICIoT-DIAD 2024 dataset used in Phase 2 of this research work. The dataset contains both malicious and benign traffic collected from realistic IoT network environments. A total of 35,671,521 traffic records are available, which are divided into training and testing sets containing 28,537,147 and 7,134,374 records, respectively.

Among all attack classes, DDoS traffic contributes the largest number of records. In addition to attack labels, the behaviour-based dataset also contains device-type labels corresponding to different IoT devices as shown in Table I.2. These device labels are utilized for the Device Identification (DI) task, while the attack labels are used for multi-class Anomaly Detection (AD). The availability of both device and attack information makes the dataset suitable for performing simultaneous DI and AD analysis in realistic IoT environments.

For the Device Identification task, the dataset already provides a predefined set of 56 behavioural features specifically designed for DI. For the Anomaly Detection task, the complete set of 132 behavioural features is initially utilized before applying hybrid feature selection.

Evaluation of the trained models is done with the help of accuracy, precision, recall, f-score, confusion matrix, and classification report metrics. Moreover, 5-fold cross-validation is done to evaluate model stability, generalization, and robustness.

For the Anomaly Detection task, the dataset initially contains 132 behavioural features along with one AD label representing different attack categories and benign traffic. The AD process focuses on identifying malicious activities and classifying different attack types present in the IoT network traffic.

Similar preprocessing operations are applied before model training. The same set of machine learning algorithms used in Phase 1 are trained and evaluated separately for Device Identification and Anomaly Detection tasks.

The trained models are evaluated using Accuracy, Precision, Recall, F1-score, confusion matrices, and classification reports. In addition, 5-fold cross-validation is performed to analyze model robustness, stability, and generalization capability on best selected model.

After selecting the best-performing machine learning model, a hybrid feature selection approach based on Mutual Information (MI) followed by Recursive Feature Elimination (RFE) is applied separately for Device Identification and Anomaly Detection tasks. Using this approach, the Device Identification feature set is reduced from 56 features to 30 features, while the Anomaly Detection feature set is reduced from 132 features to 76 features without significant degradation in performance.

Finally, joint correctness evaluation is performed to analyze the effectiveness of simultaneous Device Identification and multi-class attack classification in IoT environments. This evaluation measures whether both the device category and attack category are predicted correctly for the same traffic instance, thereby providing a more realistic assessment of IoT network security performance.

3.5 Evaluation

This section describes the mathematical representation and evaluation metrics used in the proposed IoT security framework for Device Identification (DI) and Anomaly Detection (AD). The dataset is represented as a collection of traffic samples, where each sample contains multiple network traffic features extracted from IoT communication behaviour. The corresponding target labels represent the output classes associated with each traffic sample, such as device categories, attack types, or normal traffic classes. These feature sets and labels are used for training and evaluating the machine learning models in both Device Identification and Anomaly Detection tasks.

Standard classification measures such as Accuracy, Precision, Recall, and F1-score are used to assess the trained machine learning models. In addition, 5-fold cross-validation is performed to analyze the robustness and generalization capability of the models.

For the proposed behaviour-based framework, joint correctness evaluation is also performed to measure whether both Device Identification (DI) and Anomaly Detection (AD) predictions are simultaneously correct for the same traffic sample.

3.5.1 Performance Evaluation Metrics

The trained models are evaluated using metrics including Accuracy, Precision, Recall, and F1-score.

The classification accuracy is calculated as:

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

It represents the fraction of all instances that are correctly identified.

Precision is calculated as:

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

Recall is calculated as:

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

The F1-score is calculated as:

$$F1-Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

The F1-score is used to deal with imbalanced classes in datasets, as it balances both undetected instances and false alarms.

where TP , TN , FP , and FN represent True Positive, True Negative, False Positive, and False Negative values respectively.

Due to the extreme imbalance of the CICIOT-DIAD 2024 dataset, we reported all above metrics with weighted averages.

The top-performing models are also subjected to five-fold cross-validation in order to assess their robustness and capacity for generalization.

3.5.2 Joint Correctness Evaluation

In addition to individual model evaluation, joint correctness evaluation is performed for Device Identification and Anomaly Detection tasks. A prediction is considered jointly correct only when both the DI prediction and the corresponding AD prediction are simultaneously correct for the same traffic sample.

The joint correctness score is calculated as:

$$JC = \frac{N_{DI \cap AD}}{N_{Total}}$$

where $N_{DI \cap AD}$ represents the number of traffic samples for which both Device Identification and Anomaly Detection predictions are correct simultaneously, and N_{Total} represents the total number of evaluated traffic samples.

This evaluation provides a more practical assessment of the effectiveness of the proposed behaviour-based IoT intrusion detection framework.

3.6 Summary

This chapter outlined the development process used for the proposed IoT intrusion detection framework using the CICIOT-DIAD 2024 dataset. The chapter discussed the overall two-phase framework, flow-based baseline model, behaviour-based framework, preprocessing pipeline, mathematical formulation, and experimental setup used in this research work.

The next chapter presents the experimental results, comparative analysis, and performance evaluation of the proposed framework.

CHAPTER 4

Results and Discussion

4.1 Experimental Results

This chapter discusses the outcomes obtained from the experiments conducted using both the baseline framework and the proposed behaviour-based approach. The performance of different machine learning models is examined for Device Identification (DI) and Anomaly Detection (AD) tasks using commonly adopted evaluation measures. A comparative analysis of the models is presented to understand their effectiveness in handling IoT traffic classification. Further, additional experiments are included to examine model consistency, feature optimization, and the overall effectiveness of the proposed framework.

4.1.1 Phase 1 Results: Flow-Based Baseline

In Phase 1, the flow-based version of the CICIoT-DIAD 2024 dataset is utilized to establish a baseline framework for anomaly detection and attack classification in IoT environments. This phase focuses on examining how traffic flow characteristics can help distinguish normal and malicious network behaviour. Different classification approaches are evaluated using flow-level traffic information to analyze their capability in identifying abnormal activities and separating different attack categories from benign communication.

The performance results for binary anomaly detection are presented in Table 4.1, while the comparative graphical analysis is shown in Figure 4.1.

Table 4.1: ML models for binary classification (Attack vs Benign) using flow based feature for baseline

Classifier	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.98561	0.99725	0.98561	0.99108
Naive Bayes	0.90768	0.89758	0.90768	0.87516
Decision Tree	0.98991	0.98985	0.98991	0.98988
Random Forest	0.99083	0.99146	0.99083	0.99110
AdaBoost	0.98521	0.99668	0.98521	0.99063
XGBoost	0.99012	0.99327	0.99012	0.99135
MLP	0.98686	0.99516	0.98686	0.99046

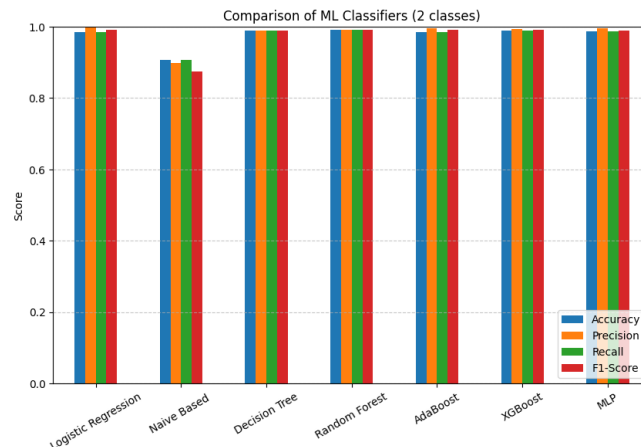


Figure 4.1: Results for Binary class classification (Attack vs Benign) using flowbased features for baseline

The results show that ensemble-based machine learning models achieved better performance compared to traditional classifiers. Among all evaluated models, Random Forest achieved the best accuracy of 99.08% along with strong Precision, Recall, and F1-Score values. XGBoost and Decision Tree also produced highly competitive results with accuracies above 99%. In contrast, Naive Bayes achieved lower performance due to its limited capability in handling complex traffic distributions present in large-scale IoT flow data.

The performance results for multi-class attack classification using seven attack categories and benign traffic are presented in Table 4.2. Similar performance trends are observed for the multi-class classification task as shown Figure 4.2.

Table 4.2: ML models on multi-class classification (7 attack types vs benign)using flow based feature for baseline

Classifier	Accuracy	Precision	Recall	F1 Score
LogisticRegression	0.92108	0.95433	0.92108	0.93577
NaiveBayes	0.81404	0.82477	0.81404	0.76261
DecisionTree	0.96685	0.96677	0.96685	0.96681
RandomForest	0.96867	0.96988	0.96867	0.96916
AdaBoost	0.91821	0.95987	0.91820	0.93820
XGBoost	0.96435	0.96841	0.96435	0.96575
MLP	0.94785	0.95766	0.94785	0.95182

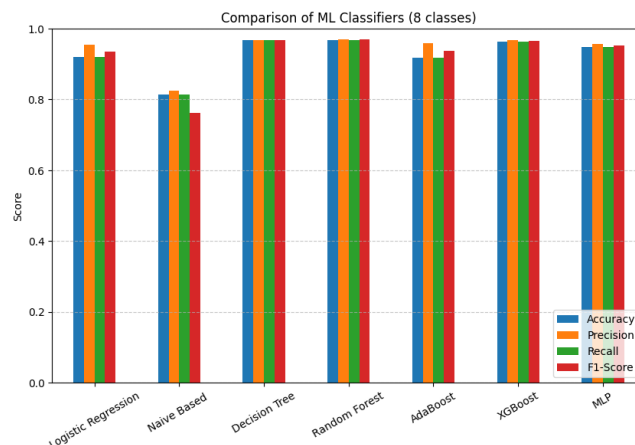


Figure 4.2: Results for Multiclass classification (7 types of Attack vs Benign) using flow based feature for baseline

Random Forest again achieved the best with an accuracy of 96.86% and high Precision,F1-score and Recall values. Decision Tree and XGBoost also demonstrated strong classification capability with accuracies above 96%. MLP achieved moderate performance, while Naive Bayes produced the lowest classification accuracy among all evaluated models.

The results of the experiment demonstrate that ensemble-based methods are somewhat better at identifying malicious activity in flow-based IoT network traffic. The flow-based features capture communication flow characteristics such as traffic duration, packet flow behaviour, protocol-level information, and connection statistics between communicating entities. These features are effective for identifying malicious network activities; however, they may not fully represent detailed device-specific behavioural patterns in IoT environments. This restriction encourages to use of behaviour-based traffic analysis in Phase 2 of the proposed research work.

4.1.2 Phase 2 Results: Device Identification (DI)

In Phase 2, the behavior-based data set is used for Device Identification using 56 behavioral features and a DI label representing multiple categories of IoT devices as shown in Table 4.3 with comparative graphical analysis shown in Figure 4.3 Experimental analysis demonstrates that behavior-based traffic representations provide highly discriminative information to identify IoT devices.Out of all the models that were assessed, Random Forest performed the best overall and showed high classification accuracy across different device classes.

By reducing the complexity involved, the Hybrid Feature Selection technique—which incorporates both Mutual Information and RFE—makes the model efficient. It aids in

reducing the number of behavior features from 56 to 30 without significantly impairing its capacity for categorization.

Table 4.3: Phase 2: ML model evaluation for Device Identification (DI)

Model	Accuracy	Precision	Recall	F1 Score	Training Time (s)
LogisticRegression	0.82554	0.74472	0.82554	0.77142	4556.13
NaiveBayes	0.08999	0.81873	0.08999	0.14491	154.23
DecisionTree	0.99014	0.99015	0.99014	0.99014	1284.02
RandomForest	0.99191	0.99186	0.99191	0.99186	1130.25
AdaBoost	0.80291	0.64466	0.80291	0.71514	11710.02
XGBoost	0.95653	0.95151	0.95653	0.95620	3175.55
MLP	0.93855	0.93745	0.93855	0.93596	43310.80

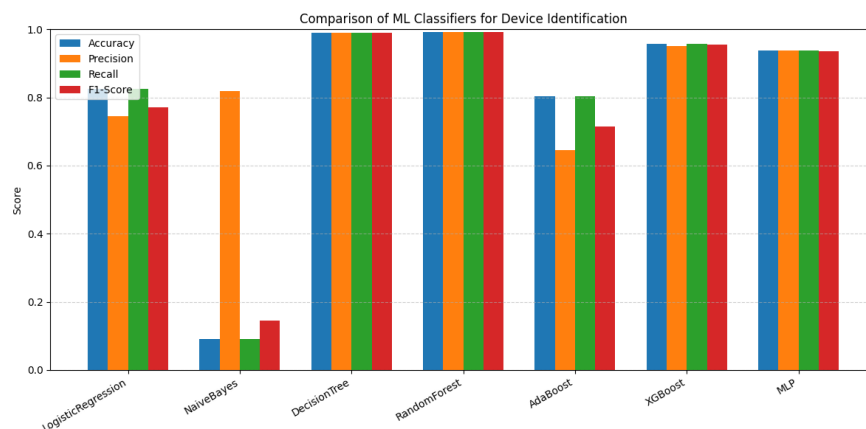


Figure 4.3: Phase 2: ML Classifiers for Device Identification with Behaviour based features

To confirm the Random Forest model's efficacy for Device Identification, 5-fold cross-validation is performed using the behaviour-based feature dataset. The cross-validation results presented in Table 4.4. The results obtained from cross-validation show stable performance across all folds, with only minor differences observed in the evaluation measures. The Random Forest model records an average accuracy of 99.15%, precision of 99.14%, recall of 99.15%, and F1-score of 99.14%. Moreover, an average AUC score of 99.16% is achieved, indicating that the model is highly effective in distinguishing among different IoT device categories.

Table 4.4: Five-Fold Cross-Validation Performance of the Random Forest Model for DI

Fold	Accuracy	Precision	Recall	F1 Score	AUC
F1	0.991526	0.991471	0.991526	0.991469	0.991628
F2	0.991501	0.991452	0.991501	0.991446	0.991579
F3	0.991614	0.991560	0.991614	0.991559	0.991677
F4	0.991501	0.991447	0.991501	0.991445	0.991557
F5	0.991450	0.991400	0.991450	0.991394	0.991605
Mean $\pm$ Std	0.99152 $\pm$ 0.00006	0.99147 $\pm$ 0.00006	0.99152 $\pm$ 0.00006	0.99146 $\pm$ 0.00006	0.99161 $\pm$ 0.00005

The performance values obtained across different folds remain highly consistent, as reflected by the extremely low standard deviation values for all evaluation metrics. This consistency confirms the robustness and stability of the Random Forest model and demonstrates that the selected behaviour-based features effectively capture device-specific communication patterns. The results further indicate that the model generalizes well across different data partitions and does not suffer from significant overfitting during the Device Identification task.

To further improve computational efficiency and reduce feature dimensionality, a hybrid feature selection approach based on Mutual Information (MI) and Recursive Feature Elimination (RFE) is applied to the Device Identification task. Initially, the behaviour-based Device Identification dataset contains 56 behavioural features. After applying the hybrid feature selection approach, the feature set is reduced to 30 important features while maintaining comparable classification performance.

Table 4.5: Performance Comparison Before and After Feature Selection for Device Identification

Feature Set	No. of Features	Accuracy	Precision	Recall	F1 Score	Training Time (s)
Original Features	56	0.99191	0.99186	0.99191	0.99186	1130.25
Selected Features	30	0.99154	0.99154	0.99154	0.99149	864.71

The performance comparison before and after feature selection is presented in Table 4.5. The outcomes indicate that the selected feature subset effectively preserves the important behavioural characteristics required for reliable IoT device identification while reducing computational complexity and training overhead.

4.1.3 Phase 2 Results: Multi-Class Anomaly Detection (AD)

For the Anomaly Detection task, the behaviour-based dataset initially contains 132 behavioural features and one AD label representing multiple attack categories and normal traffic. The chosen models are trained and evaluated separately for multi-class attack classification.

The performance results for AD task in phase 2 are presented in Table 4.6, while the comparative graphical analysis is shown in Figure 4.4.

Table 4.6: ML model evaluation for Anomaly Detection (AD)

Model	Accuracy	Precision	Recall	F1 Score	Training Time (s)
LogisticRegression	0.89508	0.88943	0.89508	0.88894	2713.00
NaiveBayes	0.45576	0.85405	0.45576	0.47759	289.49
DecisionTree	0.99562	0.99562	0.99562	0.99562	8978.91
RandomForest	0.99721	0.99720	0.99720	0.99721	2606.98
AdaBoost	0.79079	0.81307	0.79073	0.79522	44690.63
XGBoost	0.99095	0.99100	0.99095	0.99095	1008.69
MLP	0.97478	0.97530	0.97478	0.97475	20530.39

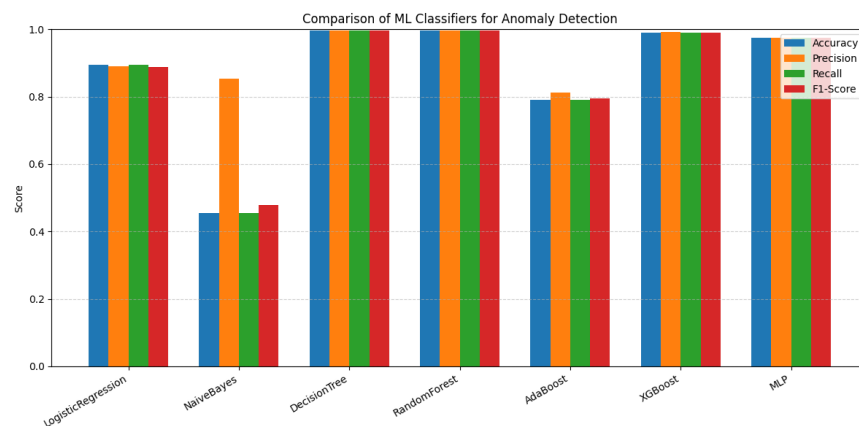


Figure 4.4: Phase 2: ML Classifiers for Anomaly Detection with Behaviour based features

The results indicate that behaviour-based features improve attack classification performance by capturing device communication behaviour more effectively than traditional flow-level traffic features.

Table 4.7: Five-Fold Cross-Validation Performance of the Random Forest Model for AD

Fold	Accuracy	Precision	Recall	F1 Score	AUC
F1	0.996783	0.996782	0.996783	0.996777	0.99991
F2	0.996831	0.99683	0.996831	0.996825	0.999933
F3	0.996814	0.996812	0.996814	0.996807	0.999914
F4	0.996796	0.996794	0.996796	0.99679	0.999944
F5	0.996919	0.996918	0.996919	0.996913	0.999918
Mean $\pm$ Std	0.99683 $\pm$ 0.00005	0.99683 $\pm$ 0.00005	0.99683 $\pm$ 0.00005	0.99682 $\pm$ 0.00005	0.99992 $\pm$ 0.00001

The robustness of the Random Forest model for multi-class attack Detection is further examined using 5-fold cross-validation on the behaviour-based dataset. The

results presented in Table 4.7 show stable performance with very small variation in the evaluation measures. On average, the model achieves 99.68% accuracy, precision, recall, and F1-score. An average AUC score of 99.99% is also obtained, showing the strong ability of the model to distinguish between benign traffic and different attack categories.

The low variation observed across the validation folds indicates that the model performs consistently and generalizes well to unseen data. These findings suggest that behaviour-based traffic features effectively represent malicious communication behaviour in IoT environments and support reliable anomaly detection.

After validating the performance of the model, a hybrid feature selection process based on Mutual Information (MI) with Recursive Feature Elimination (RFE) is applied to reduce feature complexity and improve computational efficiency for multi-class anomaly detection. The aim of this process is to identify the most informative behavioural attributes for attack classification while removing features with limited contribution.

Initially, the behaviour-based Anomaly Detection dataset contains 132 features representing different communication and behavioural characteristics of IoT traffic. Using the hybrid MI-RFE feature selection approach, the feature set is reduced to 76 important features without causing significant degradation in classification performance. The reduced feature subset helps decrease computational complexity, memory usage, and model training overhead while maintaining high detection capability.

Table 4.8: Performance Comparison Before and After Feature Selection for Anomaly Detection

Feature Set	No. of Features	Accuracy	Precision	Recall	F1 Score	Training Time (s)
Original Features	132	0.99721	0.99720	0.99720	0.99721	2606.98
Selected Features	76	0.99764	0.99764	0.99764	0.99764	1744.28

The performance comparison of before and after feature selection with Random Forest is presented in Table 4.8. The experimental results tells that the selected features preserves nearly the same classification effectiveness as the original feature set, indicating that the selected behavioural features effectively capture the important malicious communication patterns required for reliable IoT anomaly detection.

4.1.4 Phase 2 Results: Joint Correctness Evaluation

After selecting Random Forest as the best-performing machine learning model for both Device Identification (DI) and multi-class Anomaly Detection (AD), an integrated

evaluation is done to analyze the overall effectiveness, reliability, and practical applicability of the behaviour-based IoT security approach. This evaluation determine whether the framework can simultaneously identify IoT devices and accurately detect malicious activities within the same network traffic instance. A common test dataset containing 7,134,374 traffic samples is employed to evaluate the combined performance of the DI and AD models under realistic IoT network conditions.

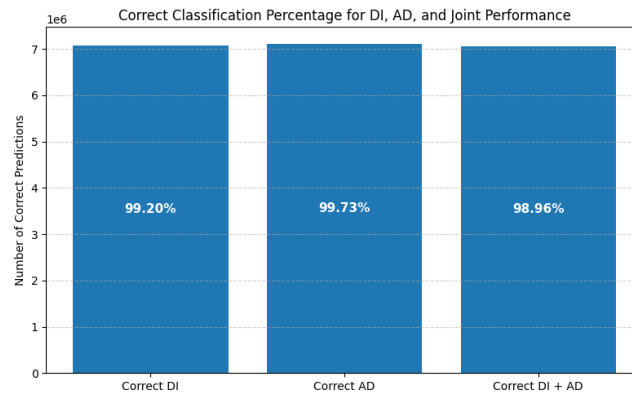


Figure 4.5: Joint Correctness Evaluation of Device Identification and Anomaly Detection

The proposed behaviour-based approach follows a sequential decision-making mechanism in which the Device Identification model first determines the category of the IoT device based on its behavioural communication characteristics. After device identification, the corresponding traffic instance is further analyzed by the Anomaly Detection model to determine whether the traffic represents benign behaviour or malicious activity and, if malicious, to classify the corresponding attack type. For the Device Identification task, the Random Forest model achieves an accuracy of 99.20%, successfully classifying 7,077,542 traffic samples into their corresponding IoT device categories. Similarly, for the multi-class Anomaly Detection task, the Random Forest model achieves an accuracy of 99.72%, correctly identifying 7,114,999 traffic samples across different attack categories and benign traffic classes. This sequential analysis enables the framework to perform device-aware intrusion detection by combining device behaviour understanding with attack classification.

To evaluate the integrated effectiveness of the framework, a joint correctness evaluation strategy is adopted. In this evaluation, a traffic sample is considered correctly classified only when both the Device Identification model and the Anomaly Detection model produce correct predictions simultaneously for the same traffic instance. In other words, the device category prediction and the attack category prediction must both be accurate at the same time for the prediction to be counted as jointly correct. This evaluation criterion provides a significantly more realistic assessment of practi-

cal IoT security systems because real-world IoT environments require simultaneous awareness of device identity and network behaviour in order to effectively detect and respond to cyber threats.

The joint evaluation results demonstrate strong integrated performance of the behaviour-based approach. Out of the total 7,134,374 test samples, 7,060,466 traffic flows are correctly classified simultaneously by both the Device Identification and Anomaly Detection models. As a result, the overall joint correctness accuracy of the integrated DI + AD system reaches 98.96% as shown in Figure 4.5. These results indicate that the selected behavioural features are highly effective in capturing both device-specific communication patterns and malicious behavioural characteristics within IoT traffic.

Furthermore, the high joint correctness accuracy confirms that the Random Forest-based models possess strong generalization capability and robustness for simultaneous device identification and attack classification tasks. The experimental findings also demonstrate that integrating Device Identification with multi-class Anomaly Detection provides a more comprehensive and reliable IoT security solution compared to performing both tasks independently. Therefore, the proposed behaviour-based approach shows strong potential for practical deployment in intelligent IoT intrusion detection systems and device-aware cybersecurity applications.

4.2 Discussion

The experimental results demonstrate that behaviour-based traffic representations provide more comprehensive and discriminative information for IoT security analysis compared to traditional flow-based traffic features. In the flow-based dataset, the experiments are limited to anomaly detection and attack classification tasks because flow-level features mainly capture statistical network traffic characteristics. Although the flow-based approach achieves strong detection performance, it lacks sufficient behavioural information for reliable device identification.

In contrast, the behaviour-based dataset enables both Device Identification (DI) and multi-class Anomaly Detection (AD) by capturing detailed communication behaviour and interaction patterns of IoT devices. The behaviour-based features effectively represent device-specific activities, payload characteristics, communication timing behaviour, and network interaction patterns, making them more suitable for simultaneous device-aware intrusion detection.

Among all evaluated machine learning algorithms, Random Forest consistently achieves the best overall performance for both Device Identification and Anomaly Detection tasks. The experimental findings also demonstrate that the behaviour-based

approach achieves higher robustness, stronger generalization capability, and improved overall system effectiveness during cross-validation and joint correctness evaluation. Table 4.9 presents the overall comparison between flow-based and behaviour-based experimental results.

Furthermore, the hybrid feature selection approach based on Mutual Information (MI) and Recursive Feature Elimination (RFE) successfully reduces the feature dimensionality for both Device Identification and Anomaly Detection tasks without causing significant degradation in classification performance. The reduced feature subsets decrease computational complexity and training overhead while preserving the important behavioural characteristics required for reliable IoT security analysis.

Table 4.9: Summary Comparison of Flow-Based and Behaviour-Based Experimental Results

Dataset Type	Task	Best Model	Accuracy	Remarks
Flow-Based	Binary Anomaly Detection	Random Forest	99.08%	Baseline intrusion detection
Flow-Based	Multi-Class Attack Classification	Random Forest	96.86%	Limited to traffic-level analysis
Behaviour-Based	Device Identification (DI)	Random Forest	99.20%	Device identification using behavioural Features
Behaviour-Based	Multi-Class Anomaly Detection (AD)	Random Forest	99.72%	Improved behavioural attack detection
Behaviour-Based	Joint Correctness (DI + AD)	Random Forest	98.96%	Simultaneous device and attack prediction

4.3 Summary

The experimental results from both the suggested behavior-based framework and the flow-based baseline approach were covered in this chapter. The Random Forest model performed the best on the Device Identification and Anomaly Detection tasks, according to the comparative analysis. Behaviour-based traffic features significantly improved classification performance compared to traditional flow-based approaches. In addition, the proposed hybrid feature selection strategy successfully reduced feature dimensionality and computational complexity while maintaining classification effectiveness. The results of the studies show how well the suggested framework works to meet practical IoT security needs. The study's main conclusions are outlined in the next chapter, which also explores possible avenues for further research based on the findings from the suggested framework.

The next chapter summarizes the major outcomes of this study and discusses potential directions for future work based on the observations obtained from the proposed framework.

CHAPTER 5

Conclusion and Future Work

5.1 Conclusion

The fast proliferation of IoT devices and the advancement in cyber-attacks have presented security issues in current-day IoT systems. Conventional IDS techniques depend primarily on the examination of network flow and fail to consider the behavioral nature of IoT devices. In order to cater for this shortcoming, an investigation using a behavior-based machine learning model for DI and multi-class AD was carried out utilizing the CICIoTDIAD2024 dataset.

The research presented in this thesis was completed in two stages. The first stage focused on building a baseline framework using flow-based traffic information for binary anomaly detection and multi-class attack classification in IoT environments. The second stage introduced a behaviour-based framework that utilized communication patterns of IoT devices to perform Device Identification and multi-class Anomaly Detection.

For evaluation of the efficacy of the proposed framework, various machine learning techniques were tested at both stages. It turned out that Random Forest yielded the best performance among all tested machine learning techniques. Its ability to maintain stable results and effectively learn from complex IoT traffic patterns made it the most suitable model for the proposed framework.

The research conducted for the purposes of this thesis was performed in two stages. In the first stage, the development of the framework was based on flow-based data regarding IoT traffic for binary anomaly detection and multi-class attacks classification. In the second stage, the development of the proposed behaviour-based approach for the identification and multi-class anomaly detection was conducted through the analysis of the communication patterns of IoT devices. In order to evaluate the efficiency of the suggested framework, various machine learning approaches have been considered at both stages of research. According to comparative analysis results, Random Forest demonstrated the best stability and efficiency in relation to the analysed problems, making it the most appropriate algorithm for the proposed framework.

The proposed framework also applied a hybrid feature selection approach based on Mutual Information followed by Recursive Feature Elimination to reduce feature dimensionality and computational complexity. The optimized feature subsets signifi-

cantly reduced training time while maintaining comparable classification performance.

In addition, a joint correctness evaluation strategy was introduced to analyze the simultaneous correctness of Device Identification and Anomaly Detection predictions. The joint evaluation results demonstrated the practical effectiveness of the proposed framework for realistic IoT intrusion detection environments.

Overall, the experimental findings indicate that behaviour-based traffic representations provide improved intrusion detection performance compared to traditional flow-based approaches and can effectively support device-aware IoT security analysis.

5.2 Future Work

While promising results have been attained through the suggested framework, there are some areas where enhancements can be made to facilitate future research. Some future work may involve making extensions to the suggested framework through adopting better learning mechanisms for improving IoT security analysis. Different deep learning approaches can be investigated to better learn complex traffic behaviour and strengthen attack detection capability in dynamic IoT environments. Further studies may also examine efficient and distributed security mechanisms suitable for networks with limited computational resources and privacy concerns.

One of the other directions that could be taken in the future is increasing the interpretability of prediction results. Explainable AI methods may be used for providing better explainability of predictions made by the model and finding the factors responsible for identifying devices and detecting anomalies.

Finally, the proposed joint evaluation framework can be further extended to support additional IoT security tasks such as attack attribution, device authentication, and unknown threat and attack detection in extensive, diverse IoT networks.

Bibliography

- [1] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 14, no. 4, pp. 2787–2805, 2010.
- [2] B. Ibrahim and H. Hameed, “An Overview of Internet of Things (IoT): Definitions, Architecture, Security, Applications and Future Directions,” *International Journal of Engineering in Computer Science*, vol. 1, no. 2, pp. 16–26, 2019.
- [3] I. Andrea, C. Chrysostomou, and G. Hadjichristofi, “Internet of Things: Security Vulnerabilities and Challenges,” in *Proc. IEEE Symposium on Computers and Communication (ISCC)*, 2015, pp. 180–187.
- [4] W. Fei, H. Ohno, and S. Sampalli, “A Systematic Review of IoT Security: Research Potential, Challenges, and Future Directions,” *ACM Computing Surveys*, vol. 56, no. 5, Article no. 111, pp. 1–40, Nov. 2023. doi: 10.1145/3625094.
- [5] S. F. Ahmed, M. S. B. Alam, S. Afrin, S. J. Rafa, S. B. Taher, and M. Kabir, “Toward a Secure 5G-Enabled Internet of Things: A Survey on Requirements, Privacy, Security, Challenges, and Opportunities,” *IEEE Access*, vol. 10, pp. 127952–127987, 2022.
- [6] U. Farooq, N. Tariq, M. Asim, T. Baker, and A. Al-Shamma’a, “Machine Learning and the Internet of Things Security: Solutions and Open Challenges,” *Future Generation Computer Systems*, vol. 82, pp. 395–411, 2018.
- [7] M. Burhan, H. Alam, A. Arsalan, R. A. Rehman, M. Anwar, and M. Faheem, “A Comprehensive Survey on the Cooperation of Fog Computing Paradigm-Based IoT Applications: Layered Architecture, Real-Time Security Issues, and Solutions,” *IEEE Access*, vol. 10, pp. 122512–122552, 2022.
- [8] D. Kanellopoulos, V. K. Sharma, T. Panagiotakopoulos, and A. Kameas, “Networking Architectures and Protocols for IoT Applications in Smart Cities: Recent Developments and Perspectives,” *Future Internet*, vol. 15, no. 2, pp. 1–29, 2023.
- [9] M. Adam, M. Hammoudeh, R. Alrawashdeh, and B. Alsulaimy, “A Survey on Security, Privacy, Trust, and Architectural Challenges in IoT Systems,” *IEEE Access*, vol. 12, pp. 1–32, 2024.

- [10] G. Kornaros, “Hardware-Assisted Machine Learning in Resource-Constrained IoT Environments for Security: Review and Future Prospective,” *IEEE Access*, vol. 10, pp. 58603–58622, 2022.
- [11] Z. Alwaisi, T. Kumar, E. Harjula, and S. Soderi, “Securing Constrained IoT Systems: A Lightweight Machine Learning Approach for Anomaly Detection and Prevention,” *Internet of Things*, vol. 28, Art. no. 101398, Dec. 2024, doi: 10.1016/j.iot.2024.101398.
- [12] K. Shahzad, T. Zia, and E. H. Qazi, “A Review of Functional Encryption in IoT Applications,” *Sensors*, vol. 22, no. 12, Art. no. 4552, 2022, doi: 10.3390/s22124552.
- [13] Z. Alwaisi, “Memory-Efficient and Robust Detection of Mirai Botnet for Future 6G-Enabled IoT Networks,” *Computer Networks*, vol. 245, Art. no. 110401, 2024, doi: 10.1016/j.comnet.2024.110401.
- [14] A. A. Alahmadi, M. Aljabri, F. Alhaidari, D. J. Alharthi, G. E. Rayani, L. A. Marghalani, O. B. Alotaibi, and S. A. Bajandouh, “DDoS Attack Detection in IoT-Based Networks Using Machine Learning Models: A Survey and Research Directions,” *Sensors*, vol. 24, no. 3, Art. no. 947, 2024, doi: 10.3390/s24030947.
- [15] C. Koliass, G. Kambourakis, A. Stavrou, and J. Voas, “DDoS in the IoT: Mirai and other botnets,” *Computer*, vol. 50, no. 7, pp. 80–84, 2017.
- [16] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, “Security, privacy and trust in Internet of Things: The road ahead,” *Computer Networks*, vol. 76, pp. 146–164, 2015.
- [17] S. Marchal, M. Miettinen, T. D. Nguyen, A.-R. Sadeghi, and N. Asokan, “AuDI: Toward autonomous IoT device-type identification using periodic communication,” *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1402–1412, 2019.
- [18] S. Ali, O. Abusabha, F. Ali, M. Imran, and T. Abuhmed, “Effective Multitask Deep Learning for IoT Malware Detection and Identification Using Behavioral Traffic Analysis,” *IEEE Internet of Things Journal*, vol. 10, no. 21, pp. 18801–18815, Nov. 2023, doi: 10.1109/JIOT.2023.3296368.
- [19] Y. Meidan, M. Bohadana, A. Mathov, S. Mirsky, D. Breitenbacher, A. Shabtai, and Y. Elovici, “ProfilIoT: A machine learning approach for IoT device identifi-

- cation based on network traffic analysis,” in *Proc. ACM Symposium on Applied Computing (SAC)*, 2017, pp. 506–509.
- [20] M. Safi, S. Dadkhah, F. Shoeleh, H. Mahdikhani, H. Molyneaux, and A. A. Ghorbani, “A Survey on IoT Profiling, Fingerprinting, and Identification,” *ACM Transactions on Internet of Things*, vol. 3, no. 4, Article no. 26, pp. 1–39, 2022, doi: 10.1145/3539736.
 - [21] N. Basha, B. Hamdaoui, K. Sivanesan, and M. Guizani, “Channel-Resilient Deep-Learning-Driven Device Fingerprinting Through Multiple Data Streams,” *IEEE Internet of Things Journal*, vol. 10, no. 18, pp. 16064–16077, 2023.
 - [22] K. Albulayhi, A. Abu Al-Haija, S. Zein-Sabatto, and M. Al-Saqqar, “IoT intrusion detection taxonomy, reference architecture, and analyses,” *IEEE Access*, vol. 9, pp. 46864–46920, 2021.
 - [23] P. L. S. Jayalaxmi, R. Saha, G. Kumar, M. Conti, and T.-H. Kim, “Machine and Deep Learning Solutions for Intrusion Detection and Prevention in IoTs: A Survey,” *IEEE Access*, vol. 11, pp. 128462–128495, 2023, doi: 10.1109/ACCESS.2023.3335286.
 - [24] R. Doshi, N. Aphthorpe, and N. Feamster, “Machine learning DDoS detection for consumer Internet of Things devices,” in *Proc. IEEE Security and Privacy Workshops*, 2018, pp. 29–35.
 - [25] M. Rabbani *et al.*, “CICIOT-DIAD 2024: A dual-label dataset for IoT device identification and anomaly detection,” Canadian Institute for Cybersecurity, University of New Brunswick, 2024.
 - [26] M. Rabbani, J. Gui, F. Nejati, Z. Zhou, A. Kaniyamattam, M. Mirani, G. Piya, I. Opushnyev, R. Lu, and A. A. Ghorbani, “Device identification and anomaly detection in IoT environments,” *IEEE Internet of Things Journal*, vol. 12, no. 10, pp. 13625–13637, 2025.
 - [27] C. Sheng, W. Zhou, Q.-L. Han, W. Ma, X. Zhu, and S. Wen, “Network Traffic Fingerprinting for IIoT Device Identification: A Survey,” *IEEE Communications Surveys & Tutorials*, vol. 26, no. 1, pp. 620–643, 2024.
 - [28] U. Inayat, M. F. Zia, S. Mahmood, H. M. Khalid, and M. Benbouzid, “Learning-based methods for cyber attacks detection in IoT systems: A survey on methods, analysis, and future prospects,” *Electronics*, vol. 11, no. 1502, 2022.

- [29] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment," *Sensors*, vol. 23, no. 5941, 2023.
- [30] R. Chataut, A. Phoummalayvane, and R. Akl, "Unleashing the power of IoT: A comprehensive review of IoT applications and future prospects," *Sensors*, vol. 23, no. 7194, 2023.
- [31] Z. Ahmad, A. S. Khan, C. W. Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, e4150, 2020.
- [32] Y. Hosain, M. H. Kabir, M. F. Mridha, and M. A. Rahman, "XAI-XGBoost: An innovative explainable intrusion detection framework for IoMT systems," *Scientific Reports*, vol. 15, no. 1, 2025.
- [33] J. R. Vergara and P. A. Estévez, "A review of feature selection methods based on mutual information," *Neural Computing and Applications*, vol. 24, no. 1, pp. 175–186, 2014.
- [34] M. A. Akif, "Binary and multi-class intrusion detection in IoT using standalone and hybrid machine and deep learning models," *arXiv preprint arXiv:2503.22684*, 2025.
- [35] Z. Sun, G. Wang, P. Li, H. Wang, M. Zhang, and X. Liang, "An Improved Random Forest Based on the Classification Accuracy and Correlation Measurement of Decision Trees," *Expert Systems with Applications*, vol. 237, Art. no. 121549, 2024.
- [36] K. A. Awan, I. U. Din, A. Almogren, B.-S. Kim, and M. Guizani, "Enhancing IoT Security With Trust Management Using Ensemble XGBoost and AdaBoost Techniques," *IEEE Access*, vol. 12, pp. 1–15, 2024.
- [37] T.-T.-H. Le, Y. E. Oktian, and H. Kim, "XGBoost for Imbalanced Multiclass Classification-Based Industrial Internet of Things Intrusion Detection Systems," *Sustainability*, vol. 14, no. 14, Art. no. 8707, 2022.
- [38] M. Awad and S. Fraihat, "Recursive Feature Elimination with Cross-Validation with Decision Tree: Feature Selection Method for Machine Learning-Based Intrusion Detection Systems," *Sensors*, vol. 24, no. 5, Art. no. 1542, 2024, doi: 10.3390/s24051542.

- [39] A. Alrawais, A. Alhothaily, C. Hu, and X. Cheng, “Fog computing for the Internet of Things: Security and privacy issues,” *IEEE Internet Computing*, vol. 21, no. 2, pp. 34–42, 2017.
- [40] M. Ammar, G. Russello, and B. Crispo, “Internet of Things: A survey on the security of IoT frameworks,” *Journal of Information Security and Applications*, vol. 38, pp. 8–27, 2018.
- [41] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, “Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on Internet-scale IoT exploitations,” *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2702–2733, 2019.
- [42] I. Butun, P. Österberg, and H. Song, “Security of the Internet of Things: Vulnerabilities, attacks, and countermeasures,” *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 616–644, 2020.
- [43] J. Li, M. S. Othman, H. Chen, and L. M. Yusuf, “Optimizing IoT Intrusion Detection System: Feature Selection Versus Feature Extraction in Machine Learning,” *Journal of Big Data*, vol. 11, Art. no. 1.

Supplementary Data

This appendix presents the supplementary data used in the proposed research work, consisting of three tables that summarize the feature sets and class definitions employed across both phases of the framework.

The Phase 1 baseline model utilizes flow-based traffic features shown in Table I.1 extracted from network flow statistics for anomaly detection and attack classification.

Table I.1: Flow-Based Traffic Features Used in the Phase 1 Baseline Model

No.	Feature Name	No.	Feature Name	No.	Feature Name
1	FlowID	29	FwdIAT Std	57	ECEFlagCount
2	SrcIP	30	FwdIATMax	58	Down/UpRatio
3	SrcPort	31	FwdIATMin	59	AveragePacketSize
4	DstIP	32	BwdIATTotal	60	FwdSegmentSizeAvg
5	DstPort	33	BwdIATMean	61	BwdSegmentSize Avg
6	Protocol	34	BwdIATStd	62	FwdBytes/BulkAvg
7	Timestamp	35	BwdIATMax	63	FwdPacket/BulkAvg
8	FlowDuration	36	BwdIATMin	64	FwdBulkRateAvg
9	TotalFwdPacket	37	FwdPSHFlags	65	BwdBytes/BulkAvg
10	TotalBwdPackets	38	BwdPSHFlags	66	BwdPacket/BulkAvg
11	TotalLengthofFwdPacket	39	FwdURGFlags	67	BwdBulkRateAvg
12	TotalLengthofBwdPacket	40	BwdURGFlags	68	SubflowFwdPackets
13	FwdPacketLengthMax	41	FwdHeaderLength	69	SubflowFwdBytes
14	FwdPacketLengthMin	42	BwdHeaderLength	70	SubflowBwdPackets
15	FwdPacketLengthMean	43	FwdPackets/s	71	SubflowBwdBytes
16	FwdPacketLengthStd	44	BwdPackets/s	72	FWDInitWinBytes
17	BwdPacketLengthMax	45	PacketLengthMin	73	BwdInitWinBytes
18	BwdPacketLengthMin	46	PacketLengthMax	74	FwdActDataPkts
19	BwdPacketLengthMean	47	PacketLengthMean	75	FwdSegSizeMin
20	BwdPacketLengthStd	48	PacketLengthStd	76	ActiveMean
21	FlowBytes/s	49	PacketLengthVariance	77	ActiveStd
22	FlowPackets/s	50	FINFlagCount	78	ActiveMax
23	FlowIATMean	51	SYNFlagCount	79	ActiveMin
24	FlowIATStd	52	RSTFlagCount	80	IdleMean
25	FlowIATMax	53	PSHFlagCount	81	IdleStd
26	FlowIATMin	54	ACKFlagCount	82	IdleMax
27	FwdIATTotal	55	URGFlagCount	83	IdleMin
28	FwdIATMean	56	CWRFlagCount	84	Label

The Phase 2 framework leverages behaviour-based features derived from packet-level traffic analysis to perform Device Identification (DI) and multi-class Anomaly Detection (AD) shown in Table I.3. Additionally, the IoT device categories used for Device Identification are also provided to define the classification space. Together, these tables offer a comprehensive view of the datasets used in the study, capturing both high-level flow characteristics and fine-grained communication behaviour essential for model training and evaluation.

This table I.2 presents the list of IoT device categories used for the Device Identification (DI) task in the behaviour-based dataset.

Table I.2: IoT Device Categories Used in Device Identification (DI)

No.	Device Type	No.	Device Type	No.	Device Type
1	UNKNOWN_DEVICE	17	YI INDOOR CAMERA	33	TP-LINK TAPO CAMERA
2	AMAZON ECHO	18	LG SMART TV	34	RASPBERRY PI 4-2 GB
3	ARLO Q INDOOR CAMERA	19	GOSUND BULB	35	TECKIN LIGHT STRIP
4	GOSUND SMART PLUG	20	SMARTTHINGS HUB	36	AMAZON PLUG
5	GOOGLE NEST MINI SPEAKER	21	BORUN/SICHUAN-AI CAMERA	37	LUMIMAN BULB
6	NEST INDOOR CAMERA	22	SONOS ONE SPEAKER	38	SMART BOARD
7	WEMO SMART PLUG	23	AMCREST WIFI CAMERA	39	ATOMI COFFEE MAKER
8	RBCIOR CAMERA	24	WYZE CAMERA	40	NETATMO WEATHER STATION
9	GOSUND POWER STRIP	25	D-LINK MINI CAMERA	41	LEVOIT AIR PURIFIER
10	AMAZON ECHO SHOW	26	NETATMO CAMERA	42	COCOON SMART HVAC FAN
11	YUTRON PLUG	27	HOME EYE CAMERA	43	IROBOT ROOMBA
12	YI OUTDOOR CAMERA	28	EUFY HOMEBASE	44	GOVEE SMART HUMIDIFIER
13	HEIMVISION CAMERA	29	AEOTEC SMART HOME HUB	45	LIFX LIGHTBULB
14	TECKIN PLUG	30	PHILIPS HUE BRIDGE	46	LUOHE CAM DOG
15	SIMCAM 1S (AMPAKTEC)	31	HARMAN KARDON (AMPAK TECHNOLOGY)		
16	LAMPUX RGB	32	ARLO BASE STATION		

Table I.3: Behaviour-Based Features Used in the Phase 2 Framework

No.	Feature Name	No.	Feature Name	No.	Feature Name
1	Stream	46	VarE	91	Stream10Var
2	(DeviceMac) Label 1 for DI	47	Q1E	92	SrcIp10Count
3	SrcIp	48	SumP	93	SrcIp10Mean
4	DstIp	49	MinP	94	SrcIp10Var
5	SrcPort	50	MaxP	95	SrcIpMac10Count
6	DstPort	51	MedP	96	SrcIpMac10Mean
7	InterArrivalTime	52	AverageP	97	SrcIpMac10Var
8	TimeSincePreviouslyDisplayedFrame	53	VarP	98	Channel10Count
9	PortClassDst	54	Q3P	99	Channel10Mean
10	L4Tcp	55	Q1P	100	Channel10Var
11	L4Udp	56	IqrP	101	StreamJitter10Sum
12	Ttl	57	L3IpDstCount	102	StreamJitter10Mean
13	EthSize	58	Jitter	103	StreamJitter10Var
14	TcpWindowSize	59	Stream1Count	104	Stream30Count
15	PayloadEntropy	60	Stream1Mean	105	Stream30Mean
16	HandshakeVersion	61	Stream1Var	106	Stream30Var
17	HandshakeCipherSuitesLength	62	SrcIp1Count	107	SrcIp30Count
18	HandshakeCipherSuites	63	SrcIp1Mean	108	SrcIp30Mean
19	HandshakeExtensionsLength	64	SrcIp1Var	109	SrcIp30Var
20	TlsServer	65	SrcIpMac1Count	110	SrcIpMac30Count
21	HandshakeSigHashAlgLen	66	SrcIpMac1Mean	111	SrcIpMac30Mean
22	HttpRequestMethod	67	SrcIpMac1Var	112	SrcIpMac30Var
23	HttpHost	68	Channel1Count	113	Channel30Count
24	HttpResponseCode	69	Channel1Mean	114	Channel30Mean
25	UserAgent	70	Channel1Var	115	Channel30Var
26	DnsServer	71	StreamJitter1Sum	116	StreamJitter30Sum
27	DnsQueryType	72	StreamJitter1Mean	117	StreamJitter30Mean
28	DnsLenQry	73	StreamJitter1Var	118	StreamJitter30Var
29	DnsInterval	74	Stream5Count	119	Stream60Count
30	DnsLenAns	75	Stream5Mean	120	Stream60Mean
31	EthSrcOui	76	Stream5Var	121	Stream60Var
32	EthDstOui	77	SrcIp5Count	122	SrcIp60Count
33	PayloadLength	78	SrcIp5Mean	123	SrcIp60Mean
34	HighestLayer	79	SrcIp5Var	124	SrcIp60Var
35	HttpUri	80	SrcIpMac5Count	125	SrcIpMac60Count
36	HttpContentLen	81	SrcIpMac5Mean	126	SrcIpMac60Mean
37	HttpContentType	82	SrcIpMac5Var	127	SrcIpMac60Var
38	IcmpType	83	Channel5Count	128	Channel60Count
39	IcmpChecksumStatus	84	Channel5Mean	129	Channel60Mean
40	IcmpDataSize	85	Channel5Var	130	Channel60Var
41	NtpInterval	86	StreamJitter5Sum	131	StreamJitter60Sum
42	MostFreqSpot	87	StreamJitter5Mean	132	StreamJitter60Mean
43	MinEt	88	StreamJitter5Var	133	StreamJitter60Var
44	Q1	89 ⁴⁸	Stream10Count	134	Label 2 for AD
45	MinE	90	Stream10Mean		

Source Code

This chapter presents the implementation details of the proposed behaviour-based machine learning approach developed for Device Identification (DI), multi-class Anomaly Detection (AD), and joint evaluation. The source code includes the major steps involved in data preprocessing, model training, prediction generation, and performance assessment of the proposed framework.

Device Identification Source Code

The implementation of the Device Identification process is presented below.

Source Code for Device Identification (DI)

```

1
2 # code for DI
3 import pandas as pd
4 import numpy as np
5 import os
6 from tqdm import tqdm
7 import warnings
8 warnings.filterwarnings('ignore')
9 from sklearn.linear_model import LogisticRegression
10 print("\nWELCOME:")
11 #DATASET_DIRECTORY_training="D:/24ISY26/
    NewDatasetAnomalyDetection/splitlabeleddata/trainingdata"
12 DATASET_DIRECTORY_training="C:/Users/hp/Desktop/
    behaviourbasedfeatures/AddnewlabelforDeviceID/spiltdataset
    /10jan-updatelabel/train"
13 #D:\24ISY26\NewDatasetAnomalyDetection\splitlabeleddata\
    trainingdata
14 #DATASET_DIRECTORY_testset="D:/24ISY26/
    NewDatasetAnomalyDetection/splitlabeleddata/testdata"
15 DATASET_DIRECTORY_testset="C:/Users/hp/Desktop/
    behaviourbasedfeatures/AddnewlabelforDeviceID/spiltdataset
    /10jan-updatelabel/test"
16 import os
17 import pandas as pd
18 from sklearn.model_selection import train_test_split

```

```

19
20 # keepcolumns for Device ID process
21
22 keep_cols = ["inter_arrival_time", "
    time_since_previously_displayed_frame", "port_class_dst",
23     "l4_tcp", "l4_udp", "ttl", "eth_size", "
        tcp_window_size", "payload_entropy",
24     "handshake_cipher_suites_length", "
        handshake_ciphersuites", "
        handshake_extensions_length",
25     "handshake_sig_hash_alg_len",
26     "http_response_code",
27     "dns_query_type", "dns_len_qry",
28     "dns_interval", "dns_len_ans", "payload_length", # "
        highest_layer", # "http_uri",
29     "http_content_len", # "http_content_type",
30     "icmp_type", "icmp_checksum_status",
31     "icmp_data_size", "ntp_interval", "most_freq_spot",
32     "min_et", "q1", "min_e", "var_e", "q1_e", "sum_p", "
        min_p", "max_p",
33     "med_p", "average_p", "var_p", "q3_p", "q1_p", "iqr_p
        ",
34     "l3_ip_dst_count",
35     "labelID"]
36
37
38 # Get sorted list of CSV file paths
39
40 df_sets_train = [os.path.join(DATASET_DIRECTORY_training, k)
    for k in os.listdir(DATASET_DIRECTORY_training) if k.
    endswith('.csv')]
41 df_sets_train.sort()
42
43 df_sets_test = [os.path.join(DATASET_DIRECTORY_testset, k) for
    k in os.listdir(DATASET_DIRECTORY_testset) if k.endswith('
    .csv')]
44 df_sets_test.sort()
45 # Containers for combined train/test data
46 train_list = []

```

```

47 test_list = []
48 i=0
49 for file_path in df_sets_train:
50     i += 1
51     try:
52         train_df = pd.read_csv(file_path)
53         train_df = train_df[[col for col in keep_cols if col
54                               in train_df.columns]]
55         train_list.append(train_df)
56     except Exception as e:
57         print(f"      Failed on 1 {file_path}: {e}")
58 i=0
59 for file_path in df_sets_test:
60     i += 1
61     try:
62         test_df = pd.read_csv(file_path)
63         test_df = test_df[[col for col in keep_cols if col in
64                             test_df.columns]]
65         test_list.append(test_df)
66     except Exception as e:
67         print(f"      Failed on 2 {file_path}: {e}")
68
69 # Combine all train/test sets
70 training_set = pd.concat(train_list, ignore_index=True)
71 test_set = pd.concat(test_list, ignore_index=True)
72 # train test already split in 80:20 by another code store in
73   two saperate folder
74 print("Labels in training set:")
75 print(training_set['labelID'].value_counts())
76
77 print("\nLabels in test set:")
78 print(test_set['labelID'].value_counts())
79 print(f"\ n   Done! Total rows      Train: {training_set.shape
80         }, Test: {test_set.shape}")
81 missing = [c for c in keep_cols if c not in train_df.columns]
82 print("Missing columns:", missing)
83 print("Count missing:", len(missing))

```

```

82
83 # suffle all trainset and test set because my data is in
    sequence like row of benign , DDos,DOS and so on dat should
    be mix
84 from sklearn.utils import shuffle
85 # Shuffle full training and test set
86 training_set = shuffle(training_set, random_state=42).
    reset_index(drop=True)
87 test_set = shuffle(test_set, random_state=42).reset_index(drop
    =True)
88 print(f"\n Done! Total rows after shuffle      Train: {
    training_set.shape}, Test: {test_set.shape}")
89 # bhawana 11 jan 2026 redefine this take oonly those which is
    required for device identification
90 X_columns = [c for c in keep_cols if c != "labelID"]
91 y_column = "labelID"
92
93 # Code Cell: Safe Cleaning + Scaling (No Massive Row Loss)
94 from sklearn.preprocessing import StandardScaler
95 import numpy as np
96
97 # 1. Replace inf values with NaN
98 for df in [training_set, test_set]:
99     df.replace([np.inf, -np.inf], np.nan, inplace=True)
100
101 # 2. Convert X columns to numeric safely
102 for col in X_columns:
103     if col in training_set.columns:
104         training_set[col] = pd.to_numeric(training_set[col],
            errors='coerce')
105         test_set[col] = pd.to_numeric(test_set[col], errors='
            coerce')
106
107 # 3. Drop rows only if ALL feature columns are missing
108 training_set = training_set.dropna(subset=X_columns, how='all'
    )
109 test_set = test_set.dropna(subset=X_columns, how='all')
110
111 # 4. Fill remaining NaN with column mean (instead of dropping

```

```

        rows)
112 training_set[X_columns] = training_set[X_columns].fillna(
        training_set[X_columns].mean())
113 test_set[X_columns] = test_set[X_columns].fillna(training_set[
        X_columns].mean())
114
115 # 5. Scaling
116 scaler = StandardScaler()
117 training_set[X_columns] = scaler.fit_transform(training_set[
        X_columns])
118 test_set[X_columns] = scaler.transform(test_set[X_columns])
119
120 print(f"      Final rows      Train: {training_set.shape}, Test:
        {test_set.shape}")
121
122 print("Number of unique labelID in training set:",
        training_set['labelID'].nunique())
123 print("Number of unique labelID in test set:", test_set['
        labelID'].nunique())
124 training_set['labelID'] = training_set['labelID'].replace({
125     'NOT SURE': 'UNKNOWN_DEVICE',
126     'Unknown_device': 'UNKNOWN_DEVICE',
127     'Unknown_Device': 'UNKNOWN_DEVICE',
128     #'Unknown': 'Unknown_Device'
129 })
130
131 test_set['labelID'] = test_set['labelID'].replace({
132     'NOT SURE': 'UNKNOWN_DEVICE',
133     'Unknown_device': 'UNKNOWN_DEVICE',
134     'Unknown_Device': 'UNKNOWN_DEVICE',
135     #'UNKNOWN': 'Unknown_Device',
136     #'Unknown': 'Unknown_Device'
137 })
138
139 print("Unique labels in training set  after mapping:",
        training_set[y_column].unique())
140
141
142 from sklearn.ensemble import RandomForestClassifier

```

```

143 from sklearn.tree import DecisionTreeClassifier
144 #from sklearn.ensemble import AdaBoostClassifier
145 #from sklearn.naive_bayes import GaussianNB
146 #from sklearn.neural_network import MLPClassifier
147 from sklearn.linear_model import LogisticRegression
148 import time
149 from xgboost import XGBClassifier
150
151
152 xgb_model = XGBClassifier(
153     objective='multi:softmax',    # or 'multi:softprob' for
154     probabilities
155     num_class=2,                # replace with number of classes
156     n_jobs=-1,
157     use_label_encoder=False,
158     eval_metric='logloss'        # Multi-class log loss
159 )
160 ML_models = [
161     LogisticRegression(n_jobs=-1),
162     GaussianNB(),
163     DecisionTreeClassifier(random_state=42),
164     RandomForestClassifier(n_jobs=-1, random_state=42),
165     #adaboost is ok but do it saperately as it is time
166     consuming
167     #AdaBoostClassifier(n_estimators=50, random_state=42),
168     MLPClassifier(random_state=42),
169     xgb_model
170 ]
171 ML_neams = [
172     "LogisticRegressionClassifier",
173     "NaiveBased(GaussianNB)Classifier",
174     "DecisionTreeClassifier",
175     "RandomForestClassifier",
176     # "AdaBoostClassifier",
177     "MLPClassifier"
178     "XGBoost Classifier"
179 ]

```

```

180 ]
181 #check how many clesse is there in training data set
182 #for binary classification it should be attack and benign
183 other wise more
184 print("Unique labels before mapping:", training_set[y_column].
    unique())
185 print("Class distribution in training set:")
186 print(training_set[y_column].value_counts())
187
188 for model in (ML_models):
189     try:
190         name = model.__class__.__name__
191
192         print(f"      training started for {name}")
193         start_time = time.time()    # start timer
194         #with tqdm(total=1,desc="Training Model"+ name) as bar:
195         model.fit(training_set[X_columns], training_set[y_column])
196         end_time = time.time()      # end timer
197         total_time = end_time - start_time
198         print(f"      {name} trained successfully")
199         print(f"      Training time for {name}: {total_time:.2f}
    seconds\n")
200         print(f"      trained")
201     except Exception as e:
202         #print(f"      Failed : {e}")
203         print(f"      {model.__class__.__name__} Failed : {e}")
204
205 print("      hiiiiiiiiiii training complete complete.")
206 # after model has been trained , check its on test data to see
207 its accuracy so predict on test data
208 y_test = []
209 preds = {i:[] for i in range(len(ML_models))}
210 y_test += list(test_set[y_column].values)
211 print("Class distribution in test set:")
212 print(test_set[y_column].value_counts())
213 for i in range(len(ML_models)):
214     model = ML_models[i]
215     name = model.__class__.__name__

```

```

215 y_pred = list(model.predict(test_set[X_columns]))
216 print("Prediction done using :", {name})
217 preds[i] = preds[i] + y_pred
218 print("Prediction done for all model:")
219 from sklearn.metrics import accuracy_score, recall_score,
    precision_score, f1_score
220 for k,v in preds.items():
221     y_pred = v
222     print(f"##### {ML_neams[k]} (all classes) #####")
223     #print('accuracy_score = ', accuracy_score(y_pred,
    y_test))
224     print('recall_score_weighted = ', recall_score(y_test,
    y_pred, average='weighted'))
225     print('precision_score_weighted = ', precision_score(
    y_test, y_pred, average='weighted'))
226     print('f1_score_weighted= ', f1_score(y_test, y_pred,
    average='weighted'))
227     print()
228 from sklearn.metrics import confusion_matrix,
    ConfusionMatrixDisplay
229 import matplotlib.pyplot as plt
230 import numpy as np
231 # If you trained multiple models and stored predictions, loop
    over them:
232 for i in range(len(ML_models)):
233     y_pred = preds[i]
234     # Create confusion matrix
235     cm = confusion_matrix(y_test, y_pred, labels=np.unique(
    y_test))
236     # Display confusion matrix
237     disp = ConfusionMatrixDisplay(confusion_matrix=cm,
    display_labels=np.unique(y_test))
238     fig,ax=plt.subplots(figsize=(14,10))
239     disp.plot(ax=ax, xticks_rotation=90, cmap='Blues',
    values_format='d')
240     plt.title(f"Confusion Matrix for Model {ML_neams[i]}")
241     plt.tight_layout()
242     #plt.show()
243 from sklearn.metrics import accuracy_score, recall_score,

```

```

precision_score, f1_score
244 for k,v in preds.items():
245     y_pred = v
246     print(f"##### {ML_neams[k]} (8 classes) #####")
247     print('accuracy_score = ', accuracy_score(y_test, y_pred))
248     print('recall_score_macro = ', recall_score(y_test, y_pred
        , average='macro'))
249     print('precision_score_macro = ', precision_score(y_test,
        y_pred, average='macro'))
250     print('f1_score_macro= ', f1_score(y_test, y_pred, average
        ='macro'))
251     print()

```

Multi-Class Attack Classification (AD) Source Code

The implementation of the multi-class Anomaly Detection process is presented below.

Source Code for Multi-Class Anomaly Detection (AD)

```

1
2 # code for AD on CICdataset
3 import pandas as pd
4 import numpy as np
5 import os
6 from tqdm import tqdm
7 import warnings
8 warnings.filterwarnings('ignore')
9 from sklearn.linear_model import LogisticRegression
10 print("\nWELCOME:")
11
12 DATASET_DIRECTORY_training="C:/Users/hp/Desktop/
    behaviourbasedfeatures/AddnewlabelforDeviceID/spiltdataset
    /10jan-updatelabel/train"
13 #DATASET already split in 80:20
14 DATASET_DIRECTORY_testset="C:/Users/hp/Desktop/
    behaviourbasedfeatures/AddnewlabelforDeviceID/spiltdataset
    /10jan-updatelabel/test"
15 #dict_8classesto2class = {}
16 #dict_8classesto2class['DDoS'] = 'Attack'
17 #dict_8classesto2class['DoS']='Attack'

```

```

18 #dict_8classesto2class['Mirai'] = 'Attack'
19 #dict_8classesto2class['Recon']='Attack'
20 #dict_8classesto2class['Spoofing']='Attack'
21 #dict_8classesto2class['Benign']='Benign'
22 #dict_8classesto2class['Web']='Attack'
23 #dict_8classesto2class['BruteForce']='Attack'
24 import os
25 import pandas as pd
26 from sklearn.model_selection import train_test_split
27
28 drop_cols = [
29     'stream', 'src_mac', 'dst_mac','src_ip','dst_ip',o'
30     src_port', 'dst_port',
31     'eth_src_oui', 'eth_dst_oui', # Non-numeric columns
32     'handshake_version','tls_server','http_request_method',
33     'http_host', 'user_agent','dns_server','highest_layer',
34     'http_uri','http_content_type','device_mac',
35     'labelID'
36 ]
37 import os
38 import pandas as pd
39 from sklearn.model_selection import train_test_split
40 # Get sorted list of CSV file paths
41 df_sets_train = [os.path.join(DATASET_DIRECTORY_training, k)
42     for k in os.listdir(DATASET_DIRECTORY_training) if k.
43     endswith('.csv')]
44 df_sets_train.sort()
45 df_sets_test = [os.path.join(DATASET_DIRECTORY_testset, k) for
46     k in os.listdir(DATASET_DIRECTORY_testset) if k.endswith('
47     .csv')]
48 df_sets_test.sort()
49 # Containers for combined train/test data
50 train_list = []
51 test_list = []
52 i=0
53 for file_path in df_sets_train:
54     i += 1
55     try:
56         train_df = pd.read_csv(file_path)

```

```

52         #train_df = train_df[[col for col in keep_cols if col
           in train_df.columns]]
53     train_df = train_df.drop(columns=[col for col in
           drop_cols if col in train_df.columns], errors='
           ignore')
54     train_list.append(train_df)
55     except Exception as e:
56         print(f"        Failed on 1 {file_path}: {e}")
57 i=0
58 for file_path in df_sets_test:
59     i += 1
60     try:
61         test_df = pd.read_csv(file_path)
62         #test_df = test_df[[col for col in keep_cols if col in
           test_df.columns]]
63         test_df = test_df.drop(columns=[col for col in
           drop_cols if col in test_df.columns], errors='
           ignore')
64         test_list.append(test_df)
65     except Exception as e:
66         print(f"        Failed on 2 {file_path}: {e}")
67 # Combine all train/test sets
68 training_set = pd.concat(train_list, ignore_index=True)
69 test_set = pd.concat(test_list, ignore_index=True)
70 print("Labels in training set:")
71 print(training_set['label2'].value_counts())
72 print("\nLabels in test set:")
73 print(test_set['label2'].value_counts())
74 print(f"\n    Done! Total rows      Train: {training_set.shape
           }, Test: {test_set.shape}")
75 y_column = "label2"
76 # Take all columns except drop_cols and except label
77 X_columns = [col for col in train_df.columns if col not in
           drop_cols + [y_column]]
78 # check fo non numeric column
79 non_numeric_cols = training_set[X_columns].select_dtypes(
           exclude=['int64','float64']).columns.tolist()
80 numeric_cols = training_set[X_columns].select_dtypes(include=[
           'int64','float64']).columns.tolist()

```

```

81
82 # suffle all trainset and test set because my data is in
    sequence like row of benign , DDos,DOS and so on dat should
    be mix
83 from sklearn.utils import shuffle
84 # Shuffle full training and test set
85 training_set = shuffle(training_set, random_state=42).
    reset_index(drop=True)
86 test_set = shuffle(test_set, random_state=42).reset_index(drop
    =True)
87 print(f"\n Done! Total rows after shuffle      Train: {
    training_set.shape}, Test: {test_set.shape}")
88 # Code Cell: Safe Cleaning + Scaling (No Massive Row Loss)
89 from sklearn.preprocessing import StandardScaler
90 import numpy as np
91 # 1. Replace inf values with NaN
92 for df in [training_set, test_set]:
93     df.replace([np.inf, -np.inf], np.nan, inplace=True)
94 # 2. Convert X columns to numeric safely
95 for col in X_columns:
96     if col in training_set.columns:
97         training_set[col] = pd.to_numeric(training_set[col],
            errors='coerce')
98         test_set[col] = pd.to_numeric(test_set[col], errors='
            coerce')
99 # 3. Drop rows only if ALL feature columns are missing
100 training_set = training_set.dropna(subset=X_columns, how='all'
    )
101 test_set = test_set.dropna(subset=X_columns, how='all')
102 # 4. Fill remaining NaN with column mean (instead of dropping
    rows)
103 training_set[X_columns] = training_set[X_columns].fillna(
    training_set[X_columns].mean())
104 test_set[X_columns] = test_set[X_columns].fillna(training_set[
    X_columns].mean())
105 # 5. Scaling
106 scaler = StandardScaler()
107 training_set[X_columns] = scaler.fit_transform(training_set[
    X_columns])

```

```

108 test_set[X_columns] = scaler.transform(test_set[X_columns])
109 print(f"      Final rows      Train: {training_set.shape}, Test:
      {test_set.shape}")
110
111 print("Number of unique label2 in training set:", training_set
      ['label2'].nunique())
112 print("Number of unique label2 in test set:", test_set['label2
      '].nunique())
113 # code if you want to map attack label 2 dict or 8
114 from sklearn.ensemble import RandomForestClassifier
115 from sklearn.tree import DecisionTreeClassifier
116 from sklearn.ensemble import AdaBoostClassifier
117 from sklearn.naive_bayes import GaussianNB
118 from sklearn.neural_network import MLPClassifier
119 from sklearn.linear_model import LogisticRegression
120 from xgboost import XGBClassifier
121 import time
122 xgb_model = XGBClassifier(
123     objective='multi:softmax',    # or 'multi:softprob' for
        probabilities
124     num_class=8,                # replace with number of classes 8
        attack
125     n_jobs=-1,
126     use_label_encoder=False,
127     eval_metric='logloss'        # Multi-class log loss
128 )
129 ML_models = [
130     LogisticRegression(n_jobs=-1),
131     GaussianNB(),
132     DecisionTreeClassifier(random_state=42),
133     RandomForestClassifier(n_jobs=-1, random_state=42),
134     #adaboost is ok but do it separately as it is time
        consuming
135     AdaBoostClassifier(n_estimators=50, random_state=42),
136     MLPClassifier(random_state=42)
137     xgb_model
138 ]
139
140 ML_neams = [

```

```

141         "LogisticRegressionClassifier",
142         "NaiveBased(GaussianNB)Classifier",
143         "DecisionTreeClassifier",
144         "RandomForestClassifier",
145         "AdaBoostClassifier",
146         "MLPClassifier"
147         "XGBoost Classifier"
148     ]
149     #check how many classes is there in training data set
150     #for binary classification it should be attack and benign
151     other wise more
152     # also check how many row in each class
153     print("Unique labels before mapping:", training_set[y_column].
154           unique())
155     print("Class distribution in training set:")
156     print(training_set[y_column].value_counts())
157     for model in (ML_models):
158         try:
159             name = model.__class__.__name__
160             print(f"      training started for {name}")
161             start_time = time.time()    # start timer
162             #with tqdm(total=1,desc="Training Model"+ name) as bar:
163             model.fit(training_set[X_columns], training_set[y_column])
164             #pbar.update(1)
165             end_time = time.time()      # end timer
166             total_time = end_time - start_time
167             print(f"      {name} trained successfully")
168             print(f"      Training time for {name}: {total_time:.2f}
169                   seconds\n")
170             print(f"      trained")
171         except Exception as e:
172             #print(f"      Failed : {e}")
173             print(f"      {model.__class__.__name__} Failed : {e}")
174     print("      hiiiiiiiiiii training complete complete.")
175     # after model has been trained , check its on test data to see
176     its accuracy so predict on test data
177     y_test = []
178     preds = {i:[] for i in range(len(ML_models))}
179     y_test += list(test_set[y_column].values)

```

```

176 print("Class distribution in test set:")
177 print(test_set[y_column].value_counts())
178 for i in range(len(ML_models)):
179     model = ML_models[i]
180     name = model.__class__.__name__
181     y_pred = list(model.predict(test_set[X_columns]))
182     print("Prediction done using :", {name})
183     preds[i] = preds[i] + y_pred
184 print("Prediction done for all model:")
185 from sklearn.metrics import accuracy_score, recall_score,
    precision_score, f1_score
186 for k,v in preds.items():
187     y_pred = v
188     print(f"##### {ML_neams[k]} (all classes) #####")
189     print('accuracy_score = ', accuracy_score(y_test, y_pred))
190     print()
191     #print('accuracy_score = ', accuracy_score(y_pred, y_test)
    )
192     print('recall_score_weighted = ', recall_score(y_test,
    y_pred, average='weighted'))
193     print('precision_score_weighted = ', precision_score(
    y_test, y_pred, average='weighted'))
194     print('f1_score_weighted= ', f1_score(y_test, y_pred,
    average='weighted'))
195 from sklearn.metrics import confusion_matrix,
    ConfusionMatrixDisplay
196 import matplotlib.pyplot as plt
197 import numpy as np
198 # If you trained multiple models and stored predictions, loop
    over them:
199 for i in range(len(ML_models)):
200     y_pred = preds[i]
201     # Create confusion matrix
202     cm = confusion_matrix(y_test, y_pred, labels=np.unique(
    y_test))
203     # Display confusion matrix
204     disp = ConfusionMatrixDisplay(confusion_matrix=cm,
    display_labels=np.unique(y_test))
205     fig,ax=plt.subplots(figsize=(14,10))

```

```

206     disp.plot(ax=ax, xticks_rotation=90, cmap='Blues',
                values_format='d')
207     plt.title(f"Confusion Matrix for Model {ML_neams[i]}")
208     plt.tight_layout()
209     #plt.show()
210     from sklearn.metrics import accuracy_score, recall_score,
        precision_score, f1_score
211     for k,v in preds.items():
212         y_pred = v
213         print(f"##### {ML_neams[k]} (8 classes) #####")
214         print('accuracy_score = ', accuracy_score(y_test, y_pred))
215         print('recall_score_macro = ', recall_score(y_test, y_pred
                , average='macro'))
216         print('precision_score_macro = ', precision_score(y_test,
                y_pred, average='macro'))
217         print('f1_score_macro= ', f1_score(y_test, y_pred, average
                ='macro'))
218         print()

```

Hybrid Feature Selection for AD Source Code

The implementation of hybrid Features selection for Attack classification is presented below.

Source Code for Joint Evaluation

```

1  import pandas as pd
2  import numpy as np
3  import time
4  import pandas as pd
5  import glob
6  import os
7  from sklearn.feature_selection import SelectKBest,
        mutual_info_classif, RFE
8  from sklearn.ensemble import RandomForestClassifier
9  from sklearn.model_selection import StratifiedShuffleSplit
10 from sklearn.metrics import accuracy_score, f1_score,
        classification_report
11 # ----- TRAIN DATA -----
12 #train_folder = "folder1"    # change path

```

```

13 train_folder="C:/Users/hp/Desktop/behaviourbasedfeatures/
    AddnewlabelforDeviceID/spiltdataset/10jan-updatelabel/train
    "
14 train_files = glob.glob(os.path.join(train_folder, "*.csv"))
15 train_df = pd.concat(
16     (pd.read_csv(file) for file in train_files),
17     ignore_index=True
18 )
19
20 print("Train Shape:", train_df.shape)
21 # ----- TEST DATA -----
22 #test_folder = "folder2"    # change path
23 test_folder="C:/Users/hp/Desktop/behaviourbasedfeatures/
    AddnewlabelforDeviceID/spiltdataset/10jan-updatelabel/test"
24 test_files = glob.glob(os.path.join(test_folder, "*.csv"))
25 test_df = pd.concat(
26     (pd.read_csv(file) for file in test_files),
27     ignore_index=True
28 )
29 print("Test Shape:", test_df.shape)
30 # load train and test data
31 target_column = "label2"
32 dropcolumn=["label2","device_mac","labelID","stream"]
33 # Train
34 X_train = train_df.drop(columns=dropcolumn)
35 y_train = train_df[target_column]
36 # Test
37 X_test = test_df.drop(columns=dropcolumn)
38 y_test = test_df[target_column]
39 # Keep only numeric features
40 X_train = X_train.select_dtypes(include=['int64', 'float64', '
    int32', 'float32'])
41 X_test = X_test[X_train.columns]    # align columns
42 # Handle missing values
43 X_train = X_train.fillna(0)
44 X_test = X_test.fillna(0)
45 print("Total Features:", X_train.shape[1])
46 print("Number of numeric features selected for feature
    selection:", X_train.shape[1])

```

```

47 print("Feature Names:\n", list(X_train.columns))
48 #Use full training data (MI is fast compared to RFE).
49 from sklearn.feature_selection import SelectKBest,
    mutual_info_classif
50 mi_selector = SelectKBest(score_func=mutual_info_classif, k
    =80)
51 mi_selector.fit(X_train, y_train)
52 selected_80 = X_train.columns[mi_selector.get_support()]
53 print("After MI:", len(selected_80))
54 # dropped feature calculation
55 print("Feature Names:\n", list(selected_80))
56 # All initial features
57 all_features = X_train.columns
58 # Dropped features = those not in selected_80
59 dropped_features = all_features.difference(selected_80)
60 print("Total Initial Features:", len(all_features))
61 print("Selected Features:", len(selected_80))
62 print("Dropped Features:", len(dropped_features))
63 print("Dropped Feature Names:\n")
64 print(list(dropped_features))
65 # apply REF on of training data
66 # create stratified subset data from training set for feature
    selection
67 sss = StratifiedShuffleSplit(n_splits=1, test_size=0.95,
    random_state=42)
68 for subset_idx, _ in sss.split(X_train[selected_80], y_train):
69     X_subset = X_train[selected_80].iloc[subset_idx]
70     y_subset = y_train.iloc[subset_idx]
71 print("Subset size:", X_subset.shape)
72 # i want check hoe much efficiency on these 80 feature on RF
    model
73 rf_final = RandomForestClassifier(
74     n_estimators=100,
75     random_state=42,
76     n_jobs=-1
77 )
78 start = time.perf_counter()
79 #rf_final.fit(X_train[final_features], y_train)
80 rf_final.fit(X_train[selected_80],y_train)# i want to check

```

```

    accuracy on 80 feature
81 train_time = time.perf_counter() - start
82 print("model trained using 80 features")
83 print("training time", train_time)
84 #measure prediction time
85 start = time.perf_counter()
86 y_pred = rf_final.predict(X_test[selected_80]) # i want to
    check on 80 feature
87 test_time = time.perf_counter() - start
88 print("prediction time", test_time)
89 #calculate performance metrics
90 from sklearn.metrics import accuracy_score, f1_score,
    classification_report, confusion_matrix
91 accuracy = accuracy_score(y_test, y_pred)
92 macro_f1 = f1_score(y_test, y_pred, average='macro')
93 weighted_f1 = f1_score(y_test, y_pred, average='weighted')
94 print("Accuracy:", accuracy)
95 print("Macro F1:", macro_f1)
96 print("Weighted F1:", weighted_f1)
97 # REF
98 from sklearn.feature_selection import RFE
99 from sklearn.ensemble import RandomForestClassifier
100 from sklearn.metrics import accuracy_score, f1_score
101 import time
102 results = []
103 for k in [77,76,75]: # give range from 80 to 75 like 80,79, 78
    than 77,76,75
104     rf = RandomForestClassifier(
105         n_estimators=100,
106         random_state=42,
107         n_jobs=-1
108     )
109
110     rfe = RFE(estimator=rf, n_features_to_select=k)
111     start = time.time()
112     rfe.fit(X_train[selected_80], y_train)
113     final_features = selected_80[rfe.support_]
114     # Train final model
115     rf.fit(X_train[final_features], y_train)

```

```

116     y_pred = rf.predict(X_test[final_features])
117     acc = accuracy_score(y_test, y_pred)
118     macro_f1 = f1_score(y_test, y_pred, average='macro')
119     weighted_f1 = f1_score(y_test, y_pred, average='weighted')
120     elapsed = time.time() - start
121     results.append([k, acc, macro_f1, weighted_f1, elapsed])
122 results_df = pd.DataFrame(
123     results,
124     columns=["Features", "Accuracy", "Macro_F1", "Weighted_F1",
125             , "Time"]
126 )
127 print(results_df)
128 print("time for REF", elapsed)
129 # based on above result select the best k and assign to best_k
130
131 # dont run until you decide how many feature
132 best_k = 76 # change based on results
133 rf = RandomForestClassifier(n_estimators=100, random_state=42,
134                             n_jobs=-1)
135 rfe = RFE(estimator=rf, n_features_to_select=best_k)
136 rfe.fit(X_train[selected_80], y_train)
137 final_features = selected_80[rfe.support_]
138 pd.Series(final_features).to_csv("selected_features_"+str(
139     best_k)+"multiclass1.csv", index=False)
140 print("Final Selected Features:", len(final_features))
141 pd.Series(final_features).to_csv("selected_features_multiclass
142     "+str(best_k)+"79feature.csv", index=False)
143 print("Final Selected Features:", len(final_features))
144 # train model on best 75 features for AD
145 rf_final = RandomForestClassifier(
146     n_estimators=100,
147     random_state=42,
148     n_jobs=-1
149 )
150 start = time.perf_counter()
151 rf_final.fit(X_train[final_features], y_train)
152 #rf_final.fit(X_train[selected_80],y_train)# i want to check
153 accuracy on 80 feature

```

```

150 train_time = time.perf_counter() - start
151 print("model trained using "+str(best_k)+"features")
152 print("model trained using "+str(best_k)+"features")
153 print("training time", train_time)
154 #measure prediction time
155 start = time.perf_counter()
156 #y_pred = rf_final.predict(X_test[final_features])
157 y_pred = rf_final.predict(X_test[selected_80]) # i want to
    check on 80 feature
158 test_time = time.perf_counter() - start
159 print("prediction time", test_time)
160 #calculate performance metrics
161 from sklearn.metrics import accuracy_score, f1_score,
    classification_report, confusion_matrix
162 accuracy = accuracy_score(y_test, y_pred)
163 macro_f1 = f1_score(y_test, y_pred, average='macro')
164 weighted_f1 = f1_score(y_test, y_pred, average='weighted')
165 print("Accuracy:", accuracy)
166 print("Macro F1:", macro_f1)
167 print("Weighted F1:", weighted_f1)
168 print(classification_report(y_test, y_pred))

```

Joint Evaluation Source Code

The implementation of joint correctness evaluation is presented below.

Source Code for Joint Evaluation

```

1
2 # joint evaluation with RF for DI and RF for AD
3 import warnings
4 warnings.filterwarnings('ignore')
5 DATASET_DIRECTORY_training="C:/Users/hp/Desktop/
    behaviourbasedfeatures/AddnewlabelforDeviceID/spiltdataset
    /10jan-updatelabel/train"
6 DATASET_DIRECTORY_testset="C:/Users/hp/Desktop/
    behaviourbasedfeatures/AddnewlabelforDeviceID/spiltdataset
    /10jan-updatelabel/test"
7 import os
8 import pandas as pd

```

```

9  from sklearn.model_selection import train_test_split
10 # keepcolumns for Device ID process
11 drop_cols = [
12
13     'stream','src_mac', 'dst_mac','src_ip',
14     'dst_ip','src_port','dst_port',
15     'eth_src_oui','eth_dst_oui',
16     # Non-numeric columns
17     'handshake_version','tls_server', 'http_request_method','
        http_host','user_agent','dns_server','highest_layer','
        http_uri','http_content_type',
18
19 ]
20 # read all csv
21 import os
22 import pandas as pd
23 folder_path = DATASET_DIRECTORY_training    # change to your
        path
24 # Get sorted list of CSV file paths
25 df_sets_train = [os.path.join(DATASET_DIRECTORY_training, k)
        for k in os.listdir(DATASET_DIRECTORY_training) if k.
        endswith('.csv')]
26 df_sets_train.sort()
27 df_sets_test = [os.path.join(DATASET_DIRECTORY_testset, k) for
        k in os.listdir(DATASET_DIRECTORY_testset) if k.endswith('
        .csv')]
28 df_sets_test.sort()
29 # Containers for combined train/test data
30 train_list = []
31 test_list = []
32 i=0
33 for file_path in df_sets_train:
34     i += 1
35     try:
36         train_df_all = pd.read_csv(file_path)
37         #train_df_for_DI = train_df[[col for col in keep_cols
            if col in train_df.columns]]
38         train_df_all= train_df_all.drop(columns=[col for col
            in drop_cols if col in train_df_all.columns],

```

```

        errors='ignore')
39         train_list.append(train_df_all)
40     except Exception as e:
41         print(f"        Failed on 1 {file_path}: {e}")
42 i=0
43 for file_path in df_sets_test:
44     i += 1
45     try:
46         test_df_all = pd.read_csv(file_path)
47         # test_df_for_DI = test_df[[col for col in keep_cols if
48             col in test_df.columns]]
49         test_df_all = test_df_all.drop(columns=[col for col in
50             drop_cols if col in test_df_all.columns], errors='
51             ignore')
52         test_list.append(test_df_all)
53     except Exception as e:
54         print(f"        Failed on 2 {file_path}: {e}")
55 # Combine all train/test sets
56 training_set_all = pd.concat(train_list, ignore_index=True)
57 test_set_all = pd.concat(test_list, ignore_index=True)
58 print(f"\n    Done! Total rows    Train: {training_set_all.
59     shape}, Test: {test_set_all.shape}")
60 print("Number of unique labelID in training set before mapping
61     :", training_set_all['labelID'].nunique())
62 print("Number of unique labelID in test set before mapping:",
63     test_set_all['labelID'].nunique())
64 training_set_all['labelID'] = training_set_all['labelID'].
65     replace({
66         'NOT SURE': 'UNKNOWN_DEVICE',
67         'Unknown_device': 'UNKNOWN_DEVICE',
68         'Unknown_Device': 'UNKNOWN_DEVICE',
69     })
70 test_set_all['labelID'] = test_set_all['labelID'].replace({
71     'NOT SURE': 'UNKNOWN_DEVICE',
72     'Unknown_device': 'UNKNOWN_DEVICE',
73     'Unknown_Device': 'UNKNOWN_DEVICE',
74 })
75 print("Number of unique labelID in training set:",
76     training_set_all['labelID'].nunique())

```

```

69 print("Number of unique labelID in test set:", test_set_all['
    labelID'].nunique())
70 X_columns_DI= ["inter_arrival_time",
71                "time_since_previously_displayed_frame", "
                    port_class_dst",
72                "l4_tcp", "l4_udp", "ttl",
73                "eth_size","tcp_window_size",
74                "payload_entropy", #"handshake_version",
75                "handshake_cipher_suites_length", "
                    handshake_ciphersuites",
76                "handshake_extensions_length", # "tls_server",
77                "handshake_sig_hash_alg_len", # "
                    http_request_method",
78                #"http_host",
79                "http_response_code",          # "user_agent", # "
                    dns_server",
80                "dns_query_type", "dns_len_qry", "dns_interval",
81                "dns_len_ans", "payload_length",
82                # "highest_layer", # "http_uri",
83                "http_content_len", # "http_content_type",
84                "icmp_type", "icmp_checksum_status",
85                "icmp_data_size", "ntp_interval",
86                "most_freq_spot", "min_et",
87                "q1", "min_e", "var_e",
88                "q1_e", "sum_p",
89                "min_p", "max_p", med_p",
90                "average_p", "var_p",
91                "q3_p", "q1_p",
92                "iqr_p", "l3_ip_dst_count",
93                #"labelID"
94                ]
95 Y_column_DI="labelID"
96 X_columns_AD = [col for col in train_df_all.columns if col not
                    in ["labelID", "label2", "device_mac"]]
97 Y_column_AD = "label2"
98 print("Number of AD feature columns:", len(X_columns_AD))# it
    should be 116
99 from sklearn.utils import shuffle
100 # Shuffle full training and test set

```

```

101 training_set_all = shuffle(training_set_all, random_state=42).
    reset_index(drop=True)
102 test_set_all = shuffle(test_set_all, random_state=42).
    reset_index(drop=True)
103 print(f"\n Done! Total rows after shuffle      Train: {
    training_set_all.shape}, Test: {test_set_all.shape}")
104 from sklearn.tree import DecisionTreeClassifier
105 from sklearn.ensemble import RandomForestClassifier
106 X_DI = training_set_all[X_columns_DI]      # features for
    device identification
107 y_DI = training_set_all[Y_column_DI]      # device ID or device
    type
108 print("X_DI shape =", X_DI.shape)
109 print("Unique values in y_DI:")
110 print(len(y_DI.unique()))
111 # for device identification
112 try:
113     device_model_DI=RandomForestClassifier(n_jobs=-1,
        random_state=42)
114     device_model_DI.fit(X_DI, y_DI)
115     print("\n Training done for Device Identification (DI)"
        )
116 except Exception as e:
117     print(f"\n Training failed for DI: {e}")
118 from sklearn.tree import DecisionTreeClassifier
119 from sklearn.ensemble import RandomForestClassifier
120 X_AD = training_set_all[X_columns_AD]      # features for
    anomaly detection
121 Y_AD = training_set_all[Y_column_AD]      # anomaly name at
    label2
122 print("X_AD shape =", X_AD.shape)
123 print("Unique values in Y_AD:")
124 print(len(Y_AD.unique()))
125 # for device Anomaly detection
126 try:
127     device_model_AD=RandomForestClassifier(n_jobs=-1,
        random_state=42)
128     device_model_AD.fit(X_AD, Y_AD)
129     print("\n Training done for AD")

```

```

130 except Exception as e:
131     print(f"\n Training failed for AD: {e}")
132 #y_pred = list(model.predict(test_set[X_columns]))
133 from sklearn.metrics import accuracy_score,
    classification_report, accuracy_score, recall_score,
    precision_score, f1_score
134 # Use same feature columns as training
135 X_test_DI = test_set_all[X_columns_DI]
136 y_test_DI = test_set_all[Y_column_DI]
137 missing = [c for c in X_columns_DI if c not in X_test_DI.
    columns]
138 extra = [c for c in X_test_DI.columns if c not in X_columns_DI
    ]
139 print("Missing:", missing)
140 print("Extra:", extra)
141 print("bhaX_test_DI shape =", X_test_DI.shape)
142 print("bhaUnique values in y_test_DI:")
143 print(len(y_test_DI.unique()))
144 # Predict
145 y_pred_DI = device_model_DI.predict(X_test_DI)
146 # Accuracy
147 accuracy_DI = accuracy_score(y_test_DI, y_pred_DI)
148 print(" Device Identification Accuracy:", accuracy_DI)
149 recall_DI=recall_score(y_test_DI, y_pred_DI, average='weighted
    ')
150 print('DI recall_score_weighted = ', recall_DI)
151 precision_DI=precision_score(y_test_DI, y_pred_DI, average='
    weighted')
152 print('DI precision_score_weighted = ', precision_DI )
153 f1_score_DI=f1_score(y_test_DI, y_pred_DI, average='weighted')
154 print('DI f1_score_weighted= ', f1_score_DI )
155 print()
156 print('Bhawana_DI Done')
157 from sklearn.metrics import accuracy_score, recall_score,
    precision_score, f1_score
158 #print('accuracy_score_weighted 1= ', accuracy_score(
    y_pred_DI, y_test_DI))
159 print('accuracy_score_weighted 1= ', accuracy_score(y_test_DI,
    y_pred_DI))

```

```

160 print('recall_score_weighted = ', recall_score( y_test_DI,
    y_pred_DI, average='weighted'))    #first agru is y_true
    here it is y_test
161 print('precision_score_weighted = ', precision_score(y_test_DI
    , y_pred_DI,average='weighted'))
162 print('f1_score_weighted= ', f1_score(y_test_DI, y_pred_DI,
    average='weighted'))
163 #y_pred = list(model.predict(test_set[X_columns]))
164 from sklearn.metrics import accuracy_score,
    classification_report
165 # Use same feature columns as training
166 X_test_AD = test_set_all[X_columns_AD]
167 y_test_AD = test_set_all[Y_column_AD]
168 # Predict
169 y_pred_AD = device_model_AD.predict(X_test_AD)
170 # Accuracy
171 accuracy_AD = accuracy_score(y_test_AD, y_pred_AD)
172 print("    AD Accuracy:", accuracy_AD)
173 from sklearn.metrics import accuracy_score, recall_score,
    precision_score, f1_score
174 print('Result for AD')
175 print('accuracy_score_weighted 1AD= ', accuracy_score(
    y_test_AD, y_pred_AD))
176 print('recall_score_weighted AD= ', recall_score( y_test_AD,
    y_pred_AD, average='weighted'))    #first agru is y_true
    here it is y_test
177 print('precision_score_weighted AD= ', precision_score(
    y_test_AD, y_pred_AD,average='weighted'))
178 print('f1_score_weighted= AD', f1_score(y_test_AD, y_pred_AD,
    average='weighted'))
179 print()
180 print('Bhawana AD done')
181 # test code for both
182 import numpy as np
183 # Boolean arrays for correctness
184 di_correct = (y_pred_DI == y_test_DI)
185 ad_correct = (y_pred_AD == y_test_AD)
186 print(len(di_correct))
187 # Both correct together

```

```
188 both_correct = di_correct & ad_correct
189 # Counts
190 total_samples = len(y_test_DI)
191 both_correct_count = both_correct.sum()
192 # Combined accuracy
193 combined_accuracy = both_correct_count / total_samples
194 print("Total test samples =", total_samples)
195 print("Correct in both DI and AD =", both_correct_count)
196 print("Combined (DI + AD) Accuracy =", combined_accuracy)
197 print ("no. of correct prection for DI=", di_correct.sum() )
198 print ("no. of correct prection for AD=", ad_correct.sum() )
```

Proof of Publications

International Conferences:

1. **Bhawana Shakeet** and Virender Ranga, “Machine Learning Based Anomaly Detection on the CICIoT-DIAD 2024 Dataset in IoT Environments,” in *10th International Conference on Information and Communication Technology for Competitive Strategies (ICTCS-2025)*, Jaipur, India, December 2025.





2. **Bhawana Shakeet** and Virender Ranga, “Behaviour-Based Device Identification and Anomaly Detection in IoT Networks Using Machine Learning,” in *International Conference on Signal, Systems, and Computing for Next-Gen Automation (ICSSCNA-2026)*, Eluru District, Andhra Pradesh, India, May 2026.

