Multi-class Detection and Classification of Security Attacks in IoT Networks Using Machine Learning Based on BoT-IoT Dataset

by

Kasra PAHLEVANINEJAD

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT FOR A MASTER'S DEGREE
WITH THESIS IN ELECTRICAL ENGINEERING M.A.SC

MONTRÉAL, AUGUST 24, 2026

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS (THESIS M.SC.A.)

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Michel Kadoch, Thesis Supervisor
Department of Electrical Engineering, École de technologie supérieure

Professor Pierre Bourque, President of the Board of Examiners
Department of Software Engineering and IT, École de technologie supérieure

Professor Carlos Vázquez Member of the jury
Department of Software Engineering and IT, École de technologie supérieure

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND PUBLIC

ON AUGUST 6, 2026

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGMENT

I would like to express my sincere gratitude to my research supervisor, Professor Michel Kadoch, for his guidance, support, and valuable feedback throughout my master's studies and the development of this thesis.

I would also like to thank the members of the jury for their time, thoughtful comments, and valuable suggestions, which contributed to improving the quality of this thesis.

Finally, I would like to express my deepest gratitude to my wife for her constant support, patience, and encouragement throughout this journey, and to my family for their continuous support and encouragement throughout my studies.

Détection et classification multiclasses des attaques de sécurité dans les réseaux IoT à l'aide de l'apprentissage automatique et du jeu de données BoT-IoT

Kasra PAHLEVANINEJAD

RÉSUMÉ

La prolifération rapide des dispositifs de l'Internet des objets (IoT) et des réseaux de capteurs sans fil (Wireless Sensor Networks – WSN) a considérablement élargi la surface d'attaque des réseaux modernes, rendant les mécanismes de détection d'intrusion plus importants que jamais. Cependant, de nombreuses études dans ce domaine s'appuient encore sur des jeux de données de référence anciens, tels que KDD Cup 99, NSL-KDD et DARPA, qui ne reflètent plus adéquatement les caractéristiques du trafic IoT actuel. De plus, les travaux existants se concentrent souvent sur la classification binaire des attaques et accordent peu d'attention au problème du déséquilibre des classes, pourtant très présent dans les ensembles de données réels, ainsi qu'aux contraintes de calcul propres aux environnements IoT.

Cette thèse évalue expérimentalement plusieurs modèles d'apprentissage automatique pour la détection et la classification multiclasses d'attaques dans les réseaux IoT à partir d'un sous-ensemble stratifié du jeu de données BoT-IoT. L'étude examine de manière systématique l'influence de différentes techniques de traitement du déséquilibre des classes, notamment le suréchantillonnage aléatoire, le sous-échantillonnage aléatoire, la technique de suréchantillonnage synthétique des classes minoritaires (SMOTE) ainsi que l'apprentissage sensible aux coûts. Plusieurs algorithmes de classification sont évalués dans ces différentes conditions, soit les forêts aléatoires (Random Forest), le gradient boosting, la régression logistique et les réseaux de neurones artificiels (ANN), afin d'analyser leur capacité à détecter diverses catégories d'attaques. Les performances sont évaluées à l'aide de plusieurs indicateurs, notamment l'exactitude, le rappel moyen macro (macro-recall) et le score F1 moyen macro (macro-F1), ainsi qu'au moyen de mesures liées aux coûts computationnels, telles que les temps d'entraînement et de test.

Les résultats montrent que le choix du modèle de classification et de la stratégie de gestion du déséquilibre des classes influence fortement les performances de détection. Plus précisément, la comparaison entre les modèles de référence et leurs variantes utilisant le suréchantillonnage, le sous-échantillonnage, SMOTE ou la pondération des classes met en évidence différents compromis entre la qualité de la prédiction et les coûts de calcul. Les expérimentations démontrent également que les approches tenant compte du déséquilibre des classes améliorent la détection multiclasses des attaques par rapport aux modèles non ajustés. Par ailleurs, l'apprentissage sensible aux coûts apparaît comme une solution intéressante aux méthodes de rééchantillonnage, puisqu'il permet de réduire les traitements supplémentaires liés aux données tout en maintenant de bonnes performances de classification. Dans l'ensemble, cette recherche met en évidence des combinaisons modèle–stratégie offrant un meilleur équilibre entre efficacité de détection et efficience computationnelle, contribuant ainsi au

développement de systèmes de détection d'intrusion adaptés aux environnements IoT à ressources limitées.

Mots-clés: Internet des objets (IoT), détection de cyberattaques, système de détection d'intrusion (IDS), gestion du déséquilibre des classes, apprentissage automatique, réseaux de capteurs sans fil (WSN), efficacité computationnelle.

Multi-class Detection and Classification of Security Attacks in IoT Networks Using Machine Learning Based on BoT-IoT Dataset

Kasra PAHLEVANINEJAD

ABSTRACT

The rapid proliferation of Internet of Things (IoT) and Wireless Sensor Network (WSN) devices has significantly expanded the attack surface of modern networks, making effective intrusion detection mechanisms essential. However, many existing intrusion detection studies rely on outdated benchmark datasets, such as KDD Cup 99, NSL-KDD, and DARPA, which fail to capture the characteristics of contemporary IoT traffic. Furthermore, prior research frequently focuses on binary classification and overlooks the severe class imbalance inherent in real-world IoT datasets, as well as the computational constraints of IoT environments.

This thesis experimentally evaluates several machine learning models for the detection and multi-class classification of attacks in IoT networks using a stratified subset of the BoT-IoT dataset. The study systematically investigates the impact of different class imbalance handling techniques, including random oversampling, random undersampling, Synthetic Minority Oversampling Technique (SMOTE), and cost-sensitive learning. Several classifiers are assessed under these conditions, namely Random Forest, Gradient Boosting, Logistic Regression, and Artificial Neural Networks (ANNs), to determine their effectiveness in detecting diverse attack categories. Performance is evaluated using several metrics, including accuracy, macro-averaged recall, and macro-averaged F1-score, together with computational measures such as training time and testing time.

The results demonstrate that the choice of classifier and imbalance-handling strategy has a substantial impact on detection performance. In particular, the study compares standard models with oversampling, undersampling, SMOTE, and class-weighted/cost-sensitive variants, highlighting trade-offs between predictive performance and computational overhead. The experimental findings show that imbalance-aware training improves multi-class attack detection compared with unadjusted baseline models, while cost-sensitive learning offers a promising alternative to resampling-based approaches by reducing data processing overhead. Overall, the thesis identifies model–strategy combinations that better balance detection effectiveness and efficiency, supporting the development of practical intrusion detection systems for resource-constrained IoT environments.

Keywords: IoT, Cyber Attack Detection, IDS, Imbalance Handling, Machine Learning, WSN, Resource-Efficient.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AFP	Attack Footprint
ANN	Artificial Neural Network
AUC	Area Under the Curve
CIA	Confidentiality, Integrity, Availability
CNN	Convolutional Neural Network
CPS	Cyber-Physical System
DBN	Deep Belief Network
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DDoS	Distributed Denial of Service
DL	Deep Learning
DNN	Deep Neural Network
DoS	Denial of Service
DT	Decision Tree
FN	False Negative
FP	False Positive
GB	Gradient Boosting
IDS	Intrusion Detection System
IG	Interpretability and Generalization
IoT	Internet of Things
KNN	K-Nearest Neighbors
LR	Logistic Regression
LSTM	Long Short-Term Memory
ML	Machine Learning
MLP	Multi-Layer Perceptron
MQTT	Message Queuing Telemetry Transport
NB	Naive Bayes

OCSVM	One-Class Support Vector Machine
PCA	Principal Component Analysis
PNN	Probabilistic Neural Network
QoS	Quality of Service
RBN	Restricted Boltzmann Network
RF	Random Forest
RL	Reinforcement Learning
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
RTDS	Real-Time Digital Simulator
SDN	Software Defined Networking
SMOTE	Synthetic Minority Over-sampling Technique
SSC	Sparse Subspace Clustering
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TN	True Negative
TP	True Positive
UDP	User Datagram Protocol
WSN	Wireless Sensor Network
XGBoost	Extreme Gradient Boosting

INTRODUCTION

Concepts

The Internet of Things (IoT) and Wireless Sensor Networks (WSNs) have grown rapidly in recent years and have become important components of modern cyber-physical systems. These technologies are widely used in areas such as healthcare, smart cities, industrial monitoring, and military applications. By connecting physical devices to digital systems, IoT enables large-scale data collection, automation, and improved decision-making. However, the rapid expansion of these technologies has also created several security challenges that require serious attention from researchers and cybersecurity experts (Hameed, Khan, & Hameed, 2023).

WSNs are a key part of IoT infrastructures. They consist of distributed sensor nodes that monitor environmental or physical conditions and transmit data through wireless communication. Although these networks provide advantages such as scalability, flexibility, and relatively low deployment cost, the sensor nodes have limited processing power, memory, and energy resources. Because of these limitations, implementing traditional security mechanisms is often difficult. As a result, maintaining the main security requirements, confidentiality, integrity, and availability, remains a major challenge in these systems (Bello & Zeadally, 2022).

IoT and WSN environments are also exposed to different types of cyber-attacks. For example, DoS/DDoS attacks attempt to exhaust network resources, while routing attacks such as Sinkhole, Black Hole, Wormhole, and Sybil can disrupt communication and reduce network reliability. In addition, wireless communication makes these networks vulnerable to threats like jamming and eavesdropping. These issues are especially critical in applications where reliable data transmission is essential, such as healthcare monitoring, industrial systems, and defense environments (Qureshi, Ahmad, Iqbal, & Aloqaily, 2022), (Stellios, Kotzanikolaou, & Grigoriadis, 2022).

To improve security in such environments, Intrusion Detection Systems (IDS) have been proposed as an additional protection layer. Traditional approaches based on predefined rules or signatures are often not suitable for WSN environments because they cannot easily detect

new attacks and may require high computational resources (Faker & Dogdu, 2019). Machine learning methods have therefore attracted attention, as they can learn normal network behavior from data and detect abnormal activities. In particular, supervised learning algorithms provide good detection accuracy while maintaining relatively efficient performance. Recent developments in deep learning and federated learning have also improved IDS capabilities in distributed IoT systems (Thamilarasu, Odesile, & Hoang, 2020), (Mothukuri, et al., 2021).
Based on these considerations, this study aims to design and evaluate a machine learning-based intrusion detection framework specifically developed for IoT and WSN environments.

Problem Statement

Although machine learning and deep learning techniques have been widely studied for intrusion detection, especially in IoT environments, several research gaps still exist. One major issue is that many studies rely on outdated datasets such as KDD Cup 99, NSL-KDD, and DARPA. These datasets were created many years ago and do not accurately represent modern IoT networks, which include heterogeneous devices, dynamic traffic patterns, and new types of cyber-attacks. As a result, models trained on these datasets may not perform well in real IoT environments (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020).
To address these limitations, the BoT-IoT dataset was developed specifically for IoT security research. This dataset includes various attack categories such as DDoS, DoS, reconnaissance, and theft. It contains millions of records generated using real IoT devices and realistic attack tools, making the traffic closer to real network conditions. In addition, BoT-IoT provides both normal and malicious traffic as well as raw packet captures and extracted features, which makes it suitable for different types of analysis. However, despite these advantages, comprehensive comparative studies using the BoT-IoT dataset are still limited. In particular, systematic evaluations of traditional machine learning methods and modern deep learning models are not sufficiently explored.
Therefore, the main problem addressed in this thesis is the design and evaluation of an effective intrusion detection system for IoT networks. The proposed system should be able to classify multiple types of IoT attacks rather than only distinguishing between normal and malicious traffic. It should also handle large volumes of network traffic efficiently, maintain low false

positive and false negative rates, and use computational resources efficiently so that it can be deployed in resource-constrained environments. In addition, the system should be capable of adapting to new and evolving attack behaviors.

Research Gap

Although many studies have explored intrusion detection in IoT networks, several important gaps still remain in the existing literature. In many cases, researchers focus on either data-level techniques or algorithm-level methods to address class imbalance, but rarely evaluate both approaches together within a single and consistent experimental framework. Because of this, it is difficult to clearly compare their effectiveness across different machine learning models under the same conditions.

Another limitation is that most previous works mainly report detection performance metrics such as precision, recall, and F1-score, while paying little attention to computational requirements. In practical IoT environments, factors such as training time, memory usage, and overall computational cost are important because many IoT devices operate with limited hardware resources.

In addition, a large portion of the literature focuses on binary classification problems that simply distinguish between normal and malicious traffic. However, real IoT datasets such as BoT-IoT contain multiple attack categories and often suffer from severe class imbalance. Handling this multi-class imbalance problem within IoT-specific traffic scenarios has not been sufficiently investigated.

Furthermore, detection accuracy and computational efficiency are usually studied separately in previous research. Few studies attempt to analyze the balance between these two objectives at the same time, even though achieving an effective trade-off between performance and efficiency is important for practical deployment.

Finally, many existing studies evaluate only a limited number of machine learning models, which makes it difficult to draw reliable comparisons between different algorithms. Comprehensive evaluations involving multiple classifiers, such as Random Forest, Gradient Boosting, and Artificial Neural Networks, under the same imbalance handling strategies are still relatively limited.

Research Objectives

The main aim of this research is to design, implement, and evaluate an effective intrusion detection system for IoT network environments using contemporary machine learning techniques. The study focuses on the BoT-IoT dataset and examines how different strategies for handling class imbalance affect the ability of machine learning models to detect multiple types of cyber-attacks. In particular, it considers several widely used imbalance-handling methods, including random oversampling, random undersampling, SMOTE, and cost-sensitive learning, and evaluates their influence on multi-class intrusion detection performance.

Another key objective is to determine which combination of machine learning model and imbalance-handling technique offers the best trade-off between detection accuracy and computational efficiency. This issue is especially important in IoT environments, where devices often have limited processing power, memory, and energy resources. For this reason, the research does not rely only on detection measures such as macro F1-score and recall. It also considers computational aspects, including training time and resource usage. Within this context, the study investigates whether cost-sensitive learning can provide performance close to data-level resampling techniques such as SMOTE, while requiring less computational effort.

The research also compares the performance of several supervised learning models, including Logistic Regression, Random Forest, Gradient Boosting, and Artificial Neural Network. The comparison is mainly concerned with how well these models deal with highly imbalanced IoT intrusion detection data. Special attention is given to their ability to identify minority attack classes, since these classes are often the most difficult to detect but may represent serious security threats. The study further explores whether ensemble-based methods, particularly Random Forest and Gradient Boosting, are more effective than other models under severe class imbalance conditions.

Finally, this research aims to establish a systematic evaluation framework that enables a fair comparison of different models and imbalance-handling strategies under the same experimental conditions. By carrying out this analysis, the study seeks to offer practical guidance for selecting suitable machine learning techniques for real-world IoT intrusion detection systems. Overall, the research considers both detection performance and deployment limitations, making the findings more relevant for resource-constrained IoT environments.

Research Contributions and Innovations

This research contributes to the field of IoT security by providing a comprehensive evaluation of several machine learning approaches using the realistic and large-scale BoT-IoT dataset. The study performs systematic performance comparisons of different models under consistent experimental conditions in order to better understand their effectiveness in detecting IoT-related cyber-attacks.

Unlike many previous works that focus mainly on binary detection, this research examines the ability of machine learning models to perform multi-class classification of different attack types. It also investigates the impact of class imbalance, which is a common issue in intrusion detection datasets, and evaluates both data-level and algorithm-level techniques to improve the detection of minority attack classes.

In addition to detection performance, the study considers practical factors such as training time and testing time as indicators of computational cost which are important for deployment in resource-constrained IoT environments. Through this analysis, the research highlights the strengths and limitations of different machine learning models and provides practical insights for selecting appropriate intrusion detection approaches for IoT networks. Overall, the findings aim to support researchers and practitioners in developing more effective and deployable machine learning-based security solutions for IoT systems.

Significance of the Study

This study is important from several academic, practical, and methodological perspectives. From an academic point of view, it contributes to the theoretical understanding of intrusion detection in IoT environments by formulating the problem of class imbalance as a multi-objective optimization task. In this context, the study examines how weighted empirical risk minimization interacts with different classifier architectures when the dataset is highly imbalanced. This relationship can be expressed as:

$$\min_{\theta} \sum_{m=1}^{M} \mathbf{W}_{ym}\, \ell\left(y_m, \pi_{\boldsymbol{\theta}}(\mathbf{x}_m)\right) \tag{I.1}$$

Using this formulation, the research performs a consistent and reproducible comparison across multiple experimental configurations, combining several machine learning models with

different imbalance handling strategies. Such a structured evaluation helps clarify how different approaches behave under the same experimental conditions and contributes to a more systematic understanding of model performance in IoT intrusion detection.

From a practical perspective, the study addresses real security challenges in IoT networks by using the BoT-IoT dataset, which reflects realistic network traffic and attack scenarios. The results provide useful guidance for security practitioners who need to select detection models that achieve good accuracy while remaining computationally feasible for deployment on devices with limited processing power and memory. In addition, the use of a Pareto-based optimization perspective allows the balance between detection performance and computational cost to be adjusted depending on available hardware resources.

Methodologically, the study also expands the traditional evaluation approach used in many intrusion detection studies. In addition to common classification metrics, it treats computational factors such as training time as an important evaluation criterion. The overall optimization problem can therefore be expressed as:

$$min_{\theta\, S}[\mathcal{L}_{\text{imbalance}}(\boldsymbol{\theta}, S) + \lambda\, C(\boldsymbol{\theta})] \quad \text{(I.2)}$$

Where $\boldsymbol{\theta}$ represents the model parameters, S represents the imbalance handling strategy, $\mathcal{L}_{\text{imbalance}}$ represents the classification loss under class imbalance, $C(\boldsymbol{\theta})$ denotes computational cost, and λ controls the trade-off between performance and complexity.

This formulation integrates both detection performance and computational cost into a single framework. As a result, it can serve as a general methodological template for future research on imbalanced learning problems, not only in IoT security but also in other machine learning domains where class imbalance and resource constraints are important considerations.

Scope and Limitations

This study focuses on evaluating machine learning methods for intrusion detection in IoT networks using the BoT-IoT dataset. The problem is addressed as a supervised multi-class classification task based on features extracted from network traffic.

Several limitations should be acknowledged. First, the results are based solely on the BoT-IoT dataset and may therefore not fully generalize to IoT environments with different devices, traffic patterns, or attack characteristics. Second, model evaluation was performed using a

single training–test split, and cross-validation was not applied. Consequently, the reported results may be influenced by the selected data split and may not fully represent the variability of model performance across different subsets of the dataset. In addition, the study assumes a fixed set of attack classes and does not address zero-day attacks or concept drift. Computational measures, particularly training time, are dependent on the hardware and experimental environment. Finally, the hyperparameter search was limited to a predefined search space, and advanced deep-learning architectures, such as CNNs, LSTMs, and transformers, were excluded because of their greater computational requirements.

CHAPTER 1
BACKGROUND

This chapter presents the background knowledge required to understand the security challenges and detection mechanisms in Internet of Things (IoT) environments and Wireless Sensor Networks (WSNs). First, the fundamental characteristics of IoT devices and WSN architectures are discussed, with emphasis on their resource constraints and inherent security vulnerabilities. Subsequently, different categories of cyberattacks targeting WSN-based IoT systems are introduced. Major attack types that threaten network availability, integrity, and confidentiality are also explained. Then, the chapter reviews existing intrusion detection approaches and machine learning techniques that have been applied to cybersecurity problems in resource-constrained environments. Finally, common evaluation metrics used to assess detection performance are presented. This background provides the conceptual and technical foundation for the research presented in the following chapters.

1.1. Introduction to IoT Devices and Wireless Sensor Networks

The rapid increase in the use of Internet of Things (IoT) devices has revolutionized modern cyber-physical systems. This enabled data acquisition, automation, and intelligent decision-making at a large-scale in various areas. However, this exponential growth has introduced critical security vulnerabilities. This is because most IoT devices have constraints in resources and a lack of computational capacity to execute traditional security mechanisms. This weakness results in being targeted by cyber threats, including botnets, Distributed Denial of Service (DDoS) attacks, data exfiltration, and reconnaissance operations (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020).

A Wireless Sensor Network (WSN) is a set of wirelessly connected sensor nodes that are distributed in a geographical area. The purpose of such a network is cooperating to monitor physical or environmental parameters. WSNs have significant advantages, such as low-cost deployment, flexibility, scalability, and ease of integration. On the other hand, sensor nodes have some inherent design limitation including battery power, processing capacity, memory, and bandwidth. These limitations make them different from traditional wired or wireless networks. Due to such constraints, WSNs are not able to use heavyweight security mechanisms

such as public-key cryptography and complex authentication protocols (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020), (Elsadig, 2023).

From a security point of view, every environment, including IoT and WSN, should satisfy the classical CIA triad: Confidentiality, Integrity, and Availability. Among these, availability is particularly critical in WSN-based IoT systems, because service disruption can lead to catastrophic consequences in critical applications. WSNs are highly vulnerable to cyberattacks because of limited resources, unsupervised nodes and wireless medium. The Denial-of-Service (DoS) attacks are the most destructive among all. This is because they can exhaust the network resources and make them inoperable by just using low attacker count and effort (Elsadig, 2023).

1.2. Taxonomy of Cyber Attacks in WSN-Based IoT Systems

A comprehensive taxonomy of cyberattacks is essential for understanding their characteristics, impact, and appropriate defense against them. Attacks on WSNs and IoT systems can be classified in multiple aspects. To mention a few, they can be categorized by attacker's position relative to the network and by the OSI communication layer being targeted. In this section, these classification schemes are explained, and their most important examples are detailed (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020), (Elsadig, 2023).

1.2.1. Attacker-Based Classification

The Cyberattacks can be categorized into three groups based on the position of the attacker and its relationship to the network.

External Attacks

External attacks are from adversaries who have no legal access to the network. These attackers use the inherent open wireless medium to inject malicious traffic, intercept data, or disrupt services without having valid credentials. Familiar examples include, but are not limited to, radio-frequency jamming, eavesdropping, and basic flooding attacks. Since the attacker operates from outside the network perimeter, their activities are generally easier to detect but can still cause significant interference (Elsadig, 2023).

Internal Attacks

Internal attacks are operated by malicious or compromised sensor nodes that have valid network credentials. These represent a particularly severe threat, since the attacker has critical knowledge of routing protocols, data formats, and node behavior. Internal attackers can silently manipulate data, disrupt routing, or drain neighbor nodes' energy without triggering ordinary perimeter-based detection mechanisms. For example, include Sybil attacks, black hole attacks, selective forwarding, and data integrity manipulation (Elsadig, 2023).

Hybrid (External/Internal) Attacks

Hybrid attacks combine characteristics of both internal and external threats. These attacks often involve collusion between an external attacker and one or more compromised internal nodes. The most difficult to detect and mitigate attack is of this class. because it can exploit both network boundaries and internal trust relationships simultaneously. Man-in-the-middle (MitM) attacks, denial of service, and hardware hacking are some examples of this category.
Experimental evidence reported in (Bello & Zeadally, 2022) and (Qureshi, Ahmad, Iqbal, & Aloqaily, 2022), indicates that internal attacks account for a higher proportion of successful WSN security cracks. These attacks target the fundamental inadequacy of perimeter-based defenses in WSN environments (Elsadig, 2023).

1.2.2. Layer-Based (OSI) Classification

A second widely used taxonomy classifies attacks based on the layers of the OSI communication model, identifying which protocol layer is being exploited. This perspective is particularly useful for designing appropriate countermeasures and defenses at target network components (Elsadig, 2023).

1.2.3. Detailed Explanation of Major Attack Types

Understanding different types of attack aiming WSNs is essential for designing effective defense mechanisms. Each attack type targets specific part of the network architecture, communication, or node behavior. The following sections explain some of the most significant attack types that threaten WSN environments.

Denial-of-Service (DoS) and Distributed DoS (DDoS)

DoS attacks are among the most damaging and widespread threats in WSN environments. Their first goal is to disrupt the network's normal operation. Through this end, they exhaust critical resources, including bandwidth, processing capacity, memory, and battery energy. So, the sensor nodes become unresponsive after a while. Symptoms of a DoS attack include, but are not limited to, network performance degradation, packet loss, increased latency, node unresponsiveness, and usually abnormal increases in traffic volume. DoS attacks can be executed over all OSI layers. As WSNs sensor nodes have limited resources, they are extremely vulnerable to these attacks. DDoS attacks increase this threat by using multiple compromised nodes to amplify the scale and intensity of the attack (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020), (Elsadig, 2023), (Ashfaq, Malik, Fatima, & Shahzad, 2022).

Sinkhole Attack

In a sinkhole attack, a malicious node advertises fake high-quality routes to draw traffic of the network toward itself. Once packets are received, the attacker can do anything with them, such as selectively drop, change, or analyze them. Sinkhole attacks are effective against routing protocols of WSNs and can be combined with other attacks, such as selective forwarding, to further decrease network integrity (Elsadig, 2023), (Nadim, Eugenio, & Chennamaneni, 2026).

Black Hole Attack

A black hole attack occurs when a compromised or malicious node falsely advertises itself as having the shortest or best path to a destination. Then, when the packet arrived, drop it away. All packets directed to the black hole node are silently discarded rather than forwarded. This effectively creates a void in the network's routing path. Such attacks are difficult to detect because the malicious node continues to cooperate normally in the routing discovery process (Elsadig, 2023), (Aissaoui, Deneuville, Guerber, & Pirovano, 2023).

Wormhole Attack

The wormhole attack involves two malicious nodes connected by a separate and private high-speed channel (the 'wormhole tunnel'). One node traps packet in one region of the network and transmits them through the tunnel to the second node. Then, the second node replays them in a different region. This creates a fake direct link between distant network regions, which causes disrupted routing. As a result, this enables the attackers to control a significant amount of network traffic without being caught (Elsadig, 2023), (Sharma & Kaul, 2018).

Sybil Attack

In a Sybil attack, a single malicious attacker creates multiple fake identities (Sybil nodes) within the network. These falsified identities can be used to disrupt voting mechanisms, manipulate data aggregation to influence and overwhelm legitimate routing.
Sybil attacks are particularly targeting trust-based and reputation-based security mechanisms (Elsadig, 2023), (Zhukabayeva , Mardenov E. M, & Abdildaeva A. A, 2020).

Selective Forwarding

In selective forwarding attacks, a compromised node acts as a partially functional router. It forwards most packets normally and selectively drops specific packets, often those carrying critical data. This subtle behavior makes the attack difficult to distinguish from natural packet loss due to link quality issues. This feature allows the attacker to operate secretly for a long period of time without being suspected and degrades the reliability and integrity of data delivery (Elsadig, 2023), (Alajmi & Elleithy, 2015).

Hello Flood Attack

In many WSN routing protocols, nodes broadcast 'hello' messages to discover neighbors. In a hello flood attack, an adversary broadcasts high-power hello messages, convincing distant nodes that it is a single-hop neighbor. When these nodes attempt to connect, join, or route through the attacker, packets are lost. The attack exploits the asymmetry between a node's ability to hear a powerful transmission and its ability to communicate back to the source (Elsadig, 2023), (Magotra & Kumar, 2014).

Replay Attack

A replay attacker receives valid network messages and retransmits them at a later time. In this way, it causes unauthorized or disruptive effects. In WSNs, replayed routing messages can corrupt routing tables, cause sensor nodes to accept outdated or duplicate data, and drain node energy through unnecessary processing and communication. These attacks are used against WSNs without proper timestamp validation or sequence numbering (Elsadig, 2023), (Zhao, Yang, Li, & Zhang, 2025).

Jamming

Jamming attacks operate at the physical layer, where an attacker continuously or periodically transmits radio signals on the same frequency band used by the WSN, to overwhelm the network's normal transmissions. Jamming prevents nodes from communicating and partitions the network. An advanced variant of this class is Reactive Jamming. It activates the parasite only when legitimate traffic is detected. This makes it more difficult to identify and more energy-efficient for the attacker (Elsadig, 2023), (Feng & Hua, 2018).

Eavesdropping

Eavesdropping is a passive attack in which an adversary monitors and records wireless transmissions without modifying or disrupting them. As the WSNs use open wireless channels, any node within radio range can capture transmitted data. In the case of not using encryption, eavesdropping can access sensitive measurements, user data, routing topology, and network credentials. This enables more complex following active attacks (Elsadig, 2023), (Li & Dou, 2023).

1.3. Intrusion Detection Techniques for IoT and WSN Security

Traditional security mechanisms, such as firewalls, access controls, and static rule-based systems, are not applicable to WSNs. This is because of their computational complexity, inability to adapt to novel threats, and dependency on static rule databases. Intrusion Detection Systems (IDSs) represent a critical second line of defense in WSN security and have become essential (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020).

1.3.1. Conventional IDS Approaches

Signature-based detection relies on a database of known attack patterns (signatures) and indicates any traffic that matches these patterns as malicious. Although these are highly accurate for known attacks, they fail against zero-day, polymorphic, and previously unseen threats.

Specification-based detection uses manually defined rules that describe the correct system behavior. Any deviation would be flagged as malicious. These are more flexible than signature-based methods, but it is difficult to scale them and requires significant expert knowledge to maintain.

Anomaly-based method models normal network behavior and detects statistically significant deviations as potential intrusions. This approach is theoretically capable of detecting unknown attacks but often has higher false alarm rates (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020).

1.3.2. Machine Learning Methods for Cyberattack Detection

Machine learning is a family of algorithmic approaches that enable systems to learn patterns from data rather than being specifically programmed. In the subject of cyberattack detection, ML methods are applied to classify network traffic as normal or malicious and detect anomalous behavior. Furthermore, they can classify the intrusion threats. Here, a comprehensive taxonomy of ML methods applied to cybersecurity is explained. The three primary instances are Supervised Learning, Unsupervised Learning, and Reinforcement Learning. Deep Learning is a specialized subfield applicable to all three mentioned methods (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020).

Supervised Learning

Supervised learning basically is training a model on a labeled dataset. Each data instance belongs to a known output class or target value. The model learns a mapping function from input features to output labels, which it can then apply to classify unseen instances. In intrusion detection, labeled datasets typically contain records of both normal network behavior and various attack types. This allows the model to learn discriminative patterns for each class.

Supervised learning is the commonly used paradigm in WSN intrusion detection due to the availability of labeled benchmark datasets such as WSN-DS, NSL-KDD, and CICIDS 2017 (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020), (Elsadig, 2023).

Decision Tree (DT)

A Decision Tree is a supervised learning algorithm that does not rely on parametric assumptions. It divides the feature space into hierarchy of binary segments using a tree-like structure of if-then rules. Each internal node evaluates a condition related to one of the conditions, each branch indicates the outcome of that feature, and each leaf node indicates a class label. The tree is grown by selecting the feature repeatedly and split point that maximize impurity reduction, using criteria such as Gini impurity or Information Gain. DTs are computationally efficient, interpretable, and capable of handling both categorical and numerical features without normalization (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020).

Random Forest (RF)

Random Forest is an ensemble learning technique that builds a collection of decision trees during the training process. Instead of relying on one tree, the model combines the predictions of all trees by using majority voting for classification tasks or averaging their outputs for regression. Each tree is trained on a random bootstrap sample of the data, and at each split, only a random subset of features is examined. This dual randomization reduces overfitting compared to a single DT. RF consistently achieves high detection accuracy. However, RF burdens higher computational and memory overhead than a single DT. This means they are less suitable for severely resource-constrained WSN nodes (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020).

Support Vector Machine (SVM)

SVM is a maximum-margin classification method that aims to separate data points from different classes by finding a decision boundary with the largest possible margin. For non-linearly separable data, SVM uses the kernel trick to map input features into a higher-dimensional space where a linear separator is applicable. Common kernels include Radial Basis Function (RBF), polynomial, and linear kernels. SVM is particularly effective in high-dimensional feature spaces and is relatively robust against overfitting in such settings. However, its computational complexity scales poorly with large datasets, and as there are millions of records in network traffic datasets, training can be slow for these datasets (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020).

k-Nearest Neighbors (KNN)

KNN is a non-parametric, instance-based learning algorithm that classifies a new data point by majority vote among its k nearest neighbors in the feature space. It is measured using a distance metric such as Euclidean distance. Unlike many other methods, KNN requires no traditional training phase, which means the entire training set is stored and used at inference time. Although, KNN is simple and effective for small datasets, it is computationally expensive during prediction because it must calculate distances between the new instance and all stored training samples. As a result, KNN is not suited for real-time WSN intrusion detection where low-latency decisions are required and memory is limited (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020).

Naive Bayes (NB)

Naive Bayes is a probabilistic classifier based on Bayes' theorem that assumes conditional independence between features given the class label. Although this assumption is 'naive' in practice, it often performs well empirically. For each class, the model learns the prior probability and the conditional likelihood of each feature given that class. This means the classification is performed by selecting the class with the highest posterior probability. NB is extremely lightweight in both training and inference. It is well suited for resource-constrained environments such as Wireless Sensor Networks. However, its accuracy is limited because the

independence assumption is frequently unrealistic for network traffic data, where many features are naturally correlated (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020).

Gradient Boosting and XGBoost

Gradient Boosting works by building an ensemble of weak learner (usually shallow decision trees) one after another in a sequential process. Each new tree is trained to correct the errors left behind by the previous ones. To minimize the loss function, it uses the gradient descent directly in the function space. XGBoost (Extreme Gradient Boosting) is an optimized and regularized implementation. It uses second-order gradient statistics, tree pruning, and parallel computation to improve speed and reduce overfitting. While researchers have found that these methods deliver excellent detection accuracy, they do come with a higher computational cost compared to DT-based methods (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020).

Unsupervised Learning

Unsupervised learning works with unlabeled data. It discovers inherent structure, patterns, or groupings without predefined class labels. Since it does not rely on predefined class labels, it is useful in cybersecurity situations where labeled attack data is limited or unavailable. This makes it suitable for detecting previously unknown attack patterns without requiring prior knowledge about them.

In the case of network intrusion detection, unsupervised methods model the distribution of normal network behavior. Then, they flag data points that deviate significantly from this distribution as anomalies. Here are some key algorithms: K-Means Clustering partitions data into k groups based on feature similarity; DBSCAN (Density-Based Spatial Clustering of Applications with Noise) identifies clusters of arbitrary shape and treats isolated points as anomalies; Autoencoders which are neural network architectures trained to reconstruct their input, where high reconstruction error shows anomalous behavior. Although unsupervised methods do not require labeled data and can detect unseen attacks, they produce higher false alarm rates than supervised methods, so they require careful threshold calibration (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020). (Elsadig, 2023) note that unsupervised approaches

are well-suited for preliminary anomaly detection but when labeled training data is available, they are generally inferior to supervised methods in precision and recall.

Reinforcement Learning

Reinforcement Learning (RL) involves an agent that learns optimal behavior through trial-and-error interaction with an environment. At each step, the agent takes an action, receives feedback in the form of a reward or penalty, and adjusts its policy to maximize the total reward over time. Unlike supervised and unsupervised learning, RL does not require a pre-existing dataset, instead, it learns through direct experience (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020).

In cybersecurity, RL has been explored for adaptive intrusion response systems. The agent learns to select the most effective countermeasure in response to detected threats. It has also been applied to adversarial attack generation for stress-testing IDS systems. However, as acknowledged by (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020), RL remains in relatively early stages of development for WSN security because of its slow convergence, the difficulty of defining appropriate reward functions, and the high computational cost of online learning on resource-constrained sensor hardware.

Deep Learning

Deep Learning (DL) is a subfield of machine learning that uses artificial neural networks with multiple hidden layers to learn hierarchical representations of data. DL models can automatically extract complex and abstract features from raw input. This reduces the need for manual feature engineering. Their application to cyberattack detection has grown substantially in recent years (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020), (Elsadig, 2023).

Multilayer Perceptron (MLP) is a type of feedforward neural network using fully connected layers, and it is trained using backpropagation and gradient descent. Convolutional Neural Networks (CNNs) work by applying learnable filter kernels across input sequences to extract local spatial patterns and making them useful for structured traffic data. Long Short-Term Memory (LSTM) networks are a class of Recurrent Neural Networks (RNNs) which are designed to model sequential dependencies and long-range temporal patterns. This makes them

suitable for time-series network traffic analysis. Deep Belief Networks (DBNs), on the other hand, are generative probabilistic models composed of stacked Restricted Boltzmann Machines (RBMs). They are capable of learning hierarchical representations in an unsupervised pre-training phase followed by supervised fine-tuning (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020).

Despite their strong detection performance in benchmark evaluations, (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020), (Elsadig, 2023) show that may present deployment challenges on resource-constrained WSN sensor nodes because their inference requirements, including memory usage, processing latency, and energy consumption, can exceed the available hardware resources. Although model training is considerably more computationally demanding and is normally performed offline, trained DL models can provide efficient inference when appropriately optimized. Therefore, lightweight supervised methods, such as Decision Trees, may be more suitable for highly constrained sensor nodes, whereas optimized DL models may be deployed on capable sensor nodes, gateway devices, or centralized processing.

1.4. Evaluation Metrics for Intrusion Detection Systems

Precise evaluation is essential for comparing IDS approaches in term of their real-world applicability. The foundation of IDS evaluation is the confusion matrix, which categorizes classification outcomes into four cells: True Positives (TP) which are attack instances correctly identified as attacks; False Positives (FP) which are normal instances incorrectly classified as attacks; True Negatives (TN) which are normal instances correctly classified as normal; and False Negatives (FN) which are attack instances incorrectly classified as normal. From these four quantities, a comprehensive set of performance metrics can be derived (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020).

Accuracy measures the overall proportion of correct classifications (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.1}$$

Precision reflects the proportion of flagged instances that are genuine attacks (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020):

$$Precision = \frac{TP}{TP + FP} \tag{1.2}$$

Recall (Detection Rate) measures the proportion of actual attacks that are detected (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020):

$$Recall = \frac{TP}{TP + FN} \tag{1.3}$$

The F1-Score is the harmonic mean of precision and recall (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020):

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{1.4}$$

This provides a balanced metric when class distribution is uneven.

Specificity measures the proportion of normal instances correctly classified as normal (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020):

$$Specifity = \frac{TN}{TN + FP} \tag{1.5}$$

The False Alarm Rate quantifies the rate of legitimate traffic incorrectly flagged as attacks (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020):

$$False\ Alarm\ Rate = \frac{FP}{FP + TN} \tag{1.6}$$

The ROC curve plots detection rate against false alarm rate across different classification thresholds, and the Area Under the Curve (AUC) provides a single scalar measure of overall discriminative capability (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020).

For WSN environments specifically, processing time and computational overhead are equally important metrics, since they directly impact sensor node battery life and network QoS (Elsadig, 2023)

CHAPTER 2
LITERATURE REVIEW

This chapter presents a structured review of existing research related to cyberattack detection in IoT, WSN, smart grid, and cyber-physical system environments. The reviewed studies are analyzed with emphasis on detection methodologies, handling of class imbalance, computational efficiency, and experimental validation on benchmark datasets.

2.1. Review of Previous Studies

The following section provides a comprehensive review of recent studies in cyber-attack detection field, focusing on methodologies, challenges, and experimental results.

The author's main focus in (Elsadig, 2023) is to detect DoS attack in WSN using lightweight machine learning algorithms. The lightweight traditional models are more suitable for this application considering the limitation of processing source in WSNs. To aim this goal, the WSN-DS dataset is fed to RF, KNN, DT and XGBoost classifiers. Furthermore, in order to reduce the processing time and to improve the model accuracy, the Gini feature selection method is leveraged. The evaluation metrics show the accuracy of DT is %99.5 with minimum overhead compared to other classifiers while it only takes 9.7%, 13%, and 2% of the processing time required by FR, XGBoost, and KNN respectively.

As it has been shown that the detection rate can be severely affected by the imbalanced distribution of attack classes which is a widespread issue with most intrusion datasets, the authors in (Mirsadeghi, Bahsi, Vaarandi, & Inoubli, 2023) detect cyber intrusions while they address the class imbalance problem in InSDN dataset. Four data-level balancing methods, RUS, ROS, SMOTE, and GANs, to three learning models: MLP with 6 layers, MLP with 10 layers, and Random Forest are applied in this study. Furthermore, two classifier-level methods are selected: weighted RF and one-shot learning via Siamese Neural Networks. The experimental results show weighted RF yielded the best performance among the other models. However, the original RF provided almost the same performance. It is indicated that weighted RF's only slightly superior Macro and Micro-F1 values of 99.85% and 94.67% as compared to RF's Macro and Micro-F1 values of 99.84% and 94.32%, respectively. The results show that

RF without any balancing effort can provide the highest detection performance on imbalanced SDN intrusion data, especially in minority classes.

In (Agarwal, 2022), the essence of accurate load forecasting for modern power grid operation is discussed. Load forecasting plays a critical role in distribution of electricity. With the rapid growth of smart grids and their increasing dependence on communication and computing infrastructure, load forecasting systems have been vulnerable to cyberattacks. Despite research on increasing forecast accuracy, little attention has been paid to the security and accuracy of forecast data. One of the main challenges is the lack of historical labeled data. To address this problem, recent research leverages a two-step approach. In the first step, a reference of historical real load data is constructed using unsupervised methods such as k-means clustering. The optimal number of clusters is determined using the elbow method and it was 24. Then the deviations between the new prediction and the nearest cluster are used as input features for detection. In the second phase, the investigated algorithms such as DT, RF, and KNN are used to detect the tampered cases. To make the model able to learn all of the features the ensemble method is leveraged by stacking the mentioned classifiers. It shows that using forests models as estimators can increase the detection to 97.25%, which indicates the high efficiency of hybrid models in increasing the security of forecasting systems.

The authors in (Zarandi & Sharifi, 2020) address the critical challenge of detecting and mitigating cyber-attacks in cyber-physical systems (CPS). CPS integrate physical processes with computational and communication resources, making them vulnerable to cyber threats such as deception attacks, which inject false data into sensors or controllers to reduce system performance. The core contribution of this study is a hybrid approach combining machine learning for attack detection with resilient control strategies for mitigation. The system is modeled as a leader-follower multi-agent network with four non-holonomic agents (e.g., robots), where one agent acts as the leader issuing commands. Deception attacks are simulated following a Bernoulli distribution with a probability of 0.8, assuming only one follower agent is compromised at a time. Detection employs a DNN, trained on MATLAB-generated datasets capturing agent movements and velocities before and after attacks. Simulation results show that DNN configurations with 7 hidden layers have better performance in mean squared error and accuracy curves.

Authors in (Tsogbaatar, et al., 2021) introduce DeL-IoT, a deep ensemble learning framework integrated with Software Defined Networking (SDN) to detect anomalies in IoT networks under imbalance distribution of data and manages the dynamic traffic flow based on the detection. Also, the proposed system predicts the future status of IoT devices using a LSTM network. In this architecture, network traffic generated by IoT devices is collected through SDN switches and sent to the SDN controller for real time analysis and detection. The system starts with a feature extraction stage, where raw network flow information is transformed into structured features for next step. These features are then processed using deep autoencoders or stacked autoencoders to generate predictive and informative representations. This step is used to reduce data dimensionality as computational complexity and chance of overfitting while enhancing the discrimination between normal and anomalous traffic patterns. Then these improved representations are fed to an ensemble of Probabilistic Neural Networks (PNNs). Multiple PNN models act as weak classifiers, and the final prediction label is obtained from majority voting of the individual prediction. The experimental results demonstrate that DeL-IoT achieves F1-scores between 99.5% and 99.9% and Matthews Correlation Coefficient values between 91.04% and 99.95%, validating this system effectiveness in detecting dynamic IoT attacks even under highly imbalanced data conditions.

The goal of the authors in (Pu, Wang, Shen, & Dong, 2021) is to design an unsupervised hybrid anomaly detection system for IDS to identify known, unknown, and zero-day attacks without any prior knowledge. To achieve this, the authors proposed a novel architecture named SSC-OCSVM. This method is an integration of Sub-Space Clustering (SSC) with One-Class Support Vector Machine (OCSVM). The method splits the feature space into multiple two-dimensional sub-spaces. Then it applies OCSVM to each sub-space. Finally, it accumulates evidence of abnormality across all sub-spaces into a dissimilarity vector, which is finally compared with a threshold to identify anomalies. The preprocessing steps include one-hot encoding of categorical features, F-test-based feature selection, and standardization. The NSL-KDD dataset is used for evaluation. K-means, DBSCAN, and SSC-EA are the models that compared against SSC-OCSVM. ROC curve analysis demonstrates that SSC-OCSVM achieves the largest area under the curve, showing its superior detection performance over the mentioned baseline methods.

The authors in (Kandhro & et al., 2023) evaluate and compare seven approaches to perform real-time cyber threats detection in IoT-empowered cybersecurity infrastructures, including discriminative deep learning models (RNN, CNN, DNN) and generative models (RBM, DBN, DBM, and DA), with a Generative Adversarial Network (GAN) being considered as the core proposed method. Three benchmark datasets are used for evaluation: NSL-KDD, KDDCup99, and UNSW-NB15. Results show that the system achieves 95–97% accuracy. DBM achieves the highest detection rates across five attack types, while the KDDCup99 dataset gains a peak accuracy of 99.8% and NSL-KDD reaches 98.3%, as it is more complex dataset. The UNSW-NB15 results demonstrate that DBN is the best model comparing others due to its multi-layer capacity for processing unlabeled data.

The study (Salehpour & Al-Anbagi, 2024) proposes RTAP which is a real-time model to detect and to predict the cyber-attacks in smart grid systems. To simulate both power network and communication network simultaneously in the smart grid, authors develop a co-simulation testbed integrating Real-Time Digital Simulator (RTDS) and NS3 through a custom interface module. To validate this testbed IEEE 14-bus system is used. This system is a widely used benchmark that contains 14 buses, 5 generators, and 20 transmission lines which allows realistic simulation of the interdependencies between power and communication infrastructures. The generated dataset includes seven types of attack: DOS, DDOS, FDI, Aurora, Replay, Coordinated, and Ransomware attacks. Preprocessing phase performs by Min-Max normalization followed by PCA-based dimensionality reduction. The result is maintaining 30% of original features while preserving 92% of data variance. Three ML algorithms were compared: Logistic Regression, Random Forest, and XGBoost. XGBoost consistently achieved the highest prediction accuracy across all attack types by the mean of 92.75.

The work in (Pai, Kang, & Chung, 2024) introduces a novel approach for cyber-attach detection in IDS that prioritize both interpretability and generalization (IG). Conventional ML-based IDS methods rely on feature selection or resampling however IG preserves the full original feature set and learns coherent patterns to distinguish normal from anomalous network traffic. The authors evaluate IG on three real-world datasets: NSL-KDD, UNSW-NB15, and UKM-IDS20, using training to test ratios ranging from 10-90% up to 90-10%. Preprocessing

involves five steps: label conversion, missing value preservation, Z-score discretization, column configuration, and anti-contradiction filtering. Even at the lowest training ratio (10-90%), IG achieves AUC = 0.94, Precision = 0.93, and Recall = 0.94 on NSL-KDD; AUC = 0.99, Precision = 0.98, and Recall = 0.99 on UNSW-NB15; and AUC = 0.99, Precision = 0.98, and Recall = 0.98 on UKM-IDS20. Notably, in UNSW-NB15, IG achieves perfect Recall = 1.0 with Precision ≥ 0.98 from the 40-60% ratio onward, and in UKM-IDS20 it successfully identifies all three anomalous instances without any prior exposure during training, showing strong generalization to unknown attacks.

In (Kawaminami, Shao, Hariri, & Tunc, 2026), the authors present ACR, an automated framework for cyberattack classification and response. The framework combines machine learning with ontology-based reasoning to both detect attacks and support appropriate reactions. To describe attack behavior, the study introduces a new representation called Attack Footprint (AFP). This representation is built from host-level metrics collected through WMI. The dataset was created using 14 real-world attack scenarios taken from the Atomic Red Team library. These scenarios were then mapped to the MITRE ATT&CK framework, which helped organize the attacks in a structured and meaningful way. For the classification phase, the authors used an ensemble of machine learning models, including Random Forest, SVM, and Gradient Boosting. Response generation, on the other hand, was supported through a CAPEC-based ontology. The reported results are highly promising. The proposed framework achieved an F1-score of 99.13%, while the average response time was only 4 seconds.

In (Almalki, Alghamdi, & Alabsi, 2025), the authors examined the application of several well-known 2D Convolutional Neural Networks (CNNs), including VGG16, AlexNet, LeNet, ResNet, and DenseNet, for IoT intrusion detection. Because network traffic data is inherently tabular rather than image-based, the BoT-IoT 2020 dataset required careful preprocessing before it could be used with CNN models. This preprocessing stage involved outlier removal, categorical feature encoding, and data normalization. The processed tabular features were then converted into two-dimensional grids to make them compatible with CNN-based analysis. The adapted CNN architectures were subsequently evaluated for their effectiveness in detecting cyberattacks in IoT networks. The experimental results indicated that VGG16, AlexNet, and ResNet achieved accuracy rates close to 99% when trained with a batch size of 128. Among

these models, ResNet yielded the lowest False Positive Rate (FPR), suggesting greater reliability in distinguishing benign traffic from malicious activity. In contrast, LeNet demonstrated the weakest performance among the evaluated CNN models. The study also compared the CNN-based approaches with LightGBM, a machine learning method specifically designed for tabular data. LightGBM achieved a lower FPR of 4.76% and a higher AUC of 94.84%. However, the CNN models generally produced higher overall accuracy. These findings indicate that image-based deep learning architectures can be effectively adapted for tabular IoT intrusion detection and can deliver strong classification performance.

To overcome the challenge of imbalance classes distribution, the authors in (AbdelZaher, Ismail, El-Fishawy, Abd-El-Samie, & Ramadan, 2025) proposed two hybrid intrusion detection models: XGBoost-RNN and Transformer-LightGBM. These models combine the advantages of machine learning and deep learning techniques. Tree-based algorithms such as XGBoost and LightGBM were used for feature selection and classification, while RNN and Transformer networks were used to learn sequential and contextual patterns in network traffic. The models also removed unnecessary features and improved the detection of minority attack classes. The proposed methods were tested on the BoTNeTIoT-L01-v2 and NF-Bot-IoT datasets. Because the datasets were highly imbalanced, the authors evaluated performance using weighted Precision, Recall, F1-score, and ROC-AUC metrics. The results were very strong. The XGBoost-RNN model achieved 99.99% accuracy on the BoTNeTIoT dataset. The Transformer-LightGBM model obtained 99.45% accuracy on the NF-Bot-IoT dataset and showed stable performance across different attack categories. Both models performed better than existing state-of-the-art intrusion detection systems. The study demonstrated that combining machine learning and deep learning methods can significantly improve IoT network security.

In (Leevy, Hancock, Khoshgoftaar, & Peterson, Detecting information theft attacks in the Bot-IoT dataset, 2021), the authors investigated the detection of information theft attacks in IoT networks using the Bot-IoT dataset. While this dataset is widely utilized, the information theft category specifically has received limited academic attention. To address this gap, the researchers utilized the 5% version of the dataset, which includes over 3.6 million records and 43 features. The primary objective was to determine the most effective machine learning model

for identifying these specific attacks. The study compared eight machine learning algorithms, spanning both traditional and ensemble-based methods. To ensure the reliability of the results, the authors employed stratified five-fold cross-validation and repeated the process ten times. Furthermore, because the dataset was highly imbalanced, the evaluation focused on AUC and AUPRC metrics rather than accuracy alone. The results indicated that all models could detect information theft attacks with reasonable success. However, performance varied significantly across different algorithms. LightGBM achieved the highest scores, reaching an AUC of 0.98785 and an AUPRC of 0.99463. CatBoost and XGBoost also demonstrated excellent performance, confirming the effectiveness of gradient boosting techniques for intrusion detection. In contrast, Naive Bayes produced the weakest results among the evaluated models. Overall, the study highlights the strong potential of boosting-based algorithms for securing IoT environments against information theft.

In (Manan, et al., 2023), the authors compared three deep learning models for IoT intrusion detection: Deep Neural Networks (DNNs), Recurrent Neural Networks (RNNs), and Convolutional Neural Networks (CNNs). The experiments were conducted using the 5% subset of the Bot-IoT dataset for both training and testing. Although reduced in size, this subset remained highly imbalanced and reflected realistic IoT network conditions. Most of the records were associated with DDoS and DoS attacks over TCP and UDP protocols. Smaller portions of the dataset represented service scanning, OS fingerprinting, keylogging, data theft, and normal network traffic. This distribution made the classification task more challenging and more representative of real-world attack patterns. The results revealed a clear trade-off between detection accuracy and training time. Among the three models, CNN achieved the best overall performance. With 100 hidden nodes, it reached an accuracy of 98.371%. This value was slightly higher than that of RNN, which achieved 98.311%, and DNN, which achieved 98.221%.

However, the improvement in accuracy came at the cost of higher computational time. DNN was the fastest model to train, with training times ranging from approximately 56 to 712 seconds. In contrast, CNN and RNN generally required substantially longer training times. For larger model configurations, their training time often exceeded 1000 seconds. Overall, CNN provided the highest intrusion detection accuracy and emerged as the strongest model in terms

of predictive performance. By comparison, DNN offered a more balanced trade-off between accuracy and efficiency. This makes it a practical choice in scenarios where training time is an important constraint.

In (Mishra, Rajput, Pandey, & Pathak, 2023), a comparison was carried out among four machine learning methods for botnet detection using the Bot-IoT dataset. The study focused on Decision Tree, Random Forest, Support Vector Machine, and Naive Bayes as the main classification techniques. Before building the models, the dataset was prepared through several preprocessing steps. Missing values were treated, the features were normalized, and the data was divided into training and testing sets. After that, each algorithm was trained and its performance was measured using accuracy, precision, recall, and F1-score. The authors also took into account the computational cost and training complexity of the models, as these aspects are relevant when applying detection systems in IoT environments.

The findings showed noticeable differences among the four methods. Random Forest produced the strongest results overall, reaching 87.45% accuracy, 0.88 precision, 0.94 recall, and 0.91 F1-score. Decision Tree followed as the next most effective model. Support Vector Machine showed acceptable but less competitive performance, while Naive Bayes recorded the weakest results in the comparison.

From these results, the authors suggested that Random Forest was the most suitable method for identifying attacks in the Bot-IoT dataset. They also pointed out that classifier choice has a significant influence on the effectiveness of botnet detection in IoT networks.

In (Sharma & Babbar, 2023), the authors studied whether machine learning could help detect DDoS attacks in IoT-based smart city systems. To run their experiments, they selected 5% of the BoT-IoT dataset. The data was first prepared through a series of preprocessing steps. Missing values were addressed, data types were adjusted, categorical features were encoded, and attributes that did not contribute to the task were removed. They then tested three classifiers: Random Forest, Decision Tree, and Naive Bayes. The comparison showed that Random Forest and Decision Tree gave the strongest results. Each of them reached an accuracy of 91%, although their strengths were slightly different. Random Forest achieved a recall of 96% along with an F1-score of 92%, while Decision Tree produced the highest precision at 97%. Naive Bayes, in contrast, performed much worse and reached only 71% accuracy. From

these results, the authors argued that Random Forest and Decision Tree are better suited for DDoS detection in IoT networks. More broadly, the study suggests that machine learning can make a meaningful contribution to improving security in smart city environments.

2.2. Benchmark Datasets for Cyberattack Detection

The selection of an appropriate benchmark dataset is critical for the validity and comparability of IDS research. This section introduces some of the world-wide used dataset in related studies.

KDDCup 99 is one of the earliest and most widely used intrusion detection datasets, derived from the 1998 DARPA network intrusion detection evaluation. This dataset includes 41 features in basic, traffic, and content. Also, the attack categorizes incorporate Normal, DoS, R2L, U2R, Probe (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020).

NSL-KDD is an improved derivative of KDDCup 99 that removes redundant records and rebalances the class distribution, providing a more reliable benchmark. It remains a simulation-based dataset and may not fully reflect modern network conditions (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020).

WSN-DS (Wireless Sensor Network Dataset) is designed for WSN intrusion detection and contains labeled records of normal traffic and four categories of DoS attacks: Blackhole, Grayhole, Flooding, and Scheduling attacks (Elsadig, 2023).

The Bot-IoT dataset is designed for Internet of Things devices. This dataset contains various attack types including data exfiltration, service scanning, and keylogging operations. The dataset employs Node-RED as its primary tool for simulating network behaviors and utilizes the MQTT (Message Queuing Telemetry Transport) lightweight messaging protocol (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020).

UNSW-NB15 is a modern dataset generated from a realistic network testbed, encompassing 9 attack types including Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms, as well as normal traffic (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020).

CICIDS 2017 (Canadian Institute for Cybersecurity) contains realistic network traffic profiles and labeled instances of common modern attacks including DoS, Web attack, and Botnet (Shaukat, Luo, Varadharajan, Hameed, & Xu, 2020).

2.3. Concluding Remarks

Researches demonstrate that machine learning, particularly lightweight supervised models, provides an effective solution for securing IoT and WSN environments against cyber threats. Supervised methods consistently outperform unsupervised approaches when labeled training data is available. DT-based classifiers strike an optimal balance between detection accuracy and computational efficiency for resource-constrained WSN deployments. Although, deep learning shows high performance in high-resource environments, its computational complexity makes it unsuitable for direct deployment on sensor nodes. Future research should explore efficient architectures that provides lightweight classifiers at the node level with high performance under class imbalance.

CHAPTER 3
METHODOLOGY

3.1. Problem Formulation

In this study, cyber-attack detection in IoT networks is formulated as a supervised multi-class classification problem based on network traffic information as features. As real-world cyber-attack detection datasets suffer from sever class imbalance, the main purpose is to design and evaluate machine learning models capable of accurately indicating normal traffic from different categories of cyber-attacks in the presence of class imbalance.

Let the labeled dataset be defined as:

$$\mathcal{D} = \{(\mathbf{x}_m , y_m) \mid m = 1,2, \dots, M\} \tag{3.1}$$

where:

- $\mathbf{x}_m \in \mathbb{R}^d$ denotes the d-dimensional feature vector extracted from network traffic records.
- $y_m \in \mathcal{Y}$ represents the corresponding true class label.
- $\mathcal{Y} = \{0,1, \dots, K\}$ is the set of class labels, where class 0 corresponds to normal traffic and classes 1 to K stand for different type of attacks.
- M is the total number of samples in the dataset.

The applied dataset in this research is obtained from the BoT-IoT dataset, which class distribution is highly imbalanced in it. Let $n_k = \mid \mathcal{D}_k \mid$ denote the number of samples of class k. If $n_i \ll n_j$ then class i is considered a minority class relative to class j causing the conventional learning algorithms meet significant challenge due to their tendency to be biased toward majority classes.

3.2. Dataset

3.2.1. BoT-IoT dataset

The BoT IoT dataset was developed to provide a realistic benchmark for analyzing botnet-related cyber-attacks in IoT environments. With the rapid growth of IoT devices, networks are

facing more large-scale automated attacks such as distributed denial of service and data exfiltration. Therefore, developing reliable machine learning based intrusion detection systems requires datasets that include realistic traffic patterns, diverse attack scenarios, and well-labeled network flows.

The BoT IoT dataset was generated by using a realistic cyber range testbed environment at the UNSW Canberra Cyber Range laboratory. The testbed contains multiple virtual machines that simulate normal users, IoT services, and attacking machines. Simulated IoT devices, such as temperature, humidity, and pressure sensors were implemented by using Node Red and communicated with external services via the MQTT protocol. Network traffic was captured by using T-Shark, and network flows were extracted by using the Argus tool. The resulting flows were stored and labeled in a MySQL database by using source and destination IP addresses and ports to identify attacks and normal instances.

Attack Categories in BoT-IoT

In the BoT-IoT dataset, cyber-attacks are divided into four main categories: DDoS, DoS, Reconnaissance, and Theft. Each category represents a distinct way attackers can target IoT networks. DDoS attacks happen when multiple compromised devices simultaneously send large volumes of traffic to a target, overwhelming its resources and disrupting its normal operation. Although DoS attacks share the same goal of exhausting network or system resources to make services unavailable, they typically originate from a single source. Reconnaissance attacks focus on collecting information about a target network or device. In this stage, attackers seek active hosts, open ports, and potential vulnerabilities that could be exploited in later stages of an attack. Theft attacks, on the other hand, are designed to steal sensitive information from devices or networks. This can involve unauthorized access to data, user credentials, or private communications.

Collectively, these attack categories provide a comprehensive representation of malicious behaviors within the BoT-IoT dataset and provide the foundation for developing and evaluating multi-class intrusion detection models.

Advantages of BoT-IoT Compared to Existing Datasets

Although several intrusion detection datasets are commonly used in research, many of them have significant limitations when applied to IoT security.

Earlier datasets, such as DARPA98, KDD99, and NSL KDD, are based on outdated traffic patterns and have limited attack diversity. Other datasets, such as DEFCON 8 and CAIDA focus on specific attack types and often lack reliable ground truth labeling. Datasets such as UNIBS and LBNL include packet traces but do not provide features or detailed labeling information. In addition, some datasets, including UNSW NB15 and CICIDS2017 rely on synthetic traffic generation methods or do not provide complete ground truth information.

The BoT IoT dataset addresses these limitations by providing:

- Realistic IoT network traffic generated from a dedicated cyber range environment
- A wide range of botnet-related attack scenarios
- Large scale traffic volumes including both benign and malicious activities
- Clearly labeled flow records with attack categories and subcategories
- Additional generated features suitable for machine learning based analysis

These specific features make the BoT-IoT dataset particularly suitable for evaluating machine learning and deep learning approaches for IoT intrusion detection.

Feature Extraction and Selected Features

After capturing raw network traffic, the Argus tool was used to extract network flow features from packet capture files. These features describe various properties of the communication flows, including packet counts, byte counts, durations, and protocol related information.

Statistical evaluation techniques were applied to identify the most informative features for machine learning models. Specifically, Correlation Coefficient and Joint Entropy were used to measure the relationships between features and evaluate their information content. Features with higher entropy values and lower correlation with other features were considered more informative and less redundant.

Based on the feature evaluation process, a subset of ten features was selected as the most relevant attributes for classification tasks. The selected features include seq, which represents a sequence identifier related to the network flow, stddev, indicating the standard deviation of packet sizes within the flow, N_IN_Conn_P_SrcIP, representing the number of inbound connections per source IP, min, which denotes the minimum packet size observed in the flow, state_number, an encoded representation of the connection state, mean, which describes the average packet size within the flow, N_IN_Conn_P_DstIP, representing the number of inbound connections per destination IP; drate, indicating the destination to source packet rate; srate, representing the source to destination packet rate; and max, which denotes the maximum packet size observed in the flow (Peterson, Leevy, & Khoshgoftaar, 2021).

3.2.2. Dataset Subset Selection and Data Preprocessing

The original BoT-IoT dataset is extremely large in scale, containing tens of millions of network flow records that include both benign and malicious activities. While this extensive volume provides a wide range of behavioral patterns, it also introduces practical challenges during experiment. Particularly, processing the full dataset for model training and testing requires significant computational resources and time. This high computational demand can become a limitation, especially in research environments where hardware resources are limited.

To address these limitations while maintaining the representativeness of the data, a 10% stratified subset of the BoT-IoT dataset was selected for the experimental analysis. The stratified sampling ensures that the proportion of each attack category and normal traffic in the subset remains consistent with the distribution of the original dataset. This approach preserves the statistical properties of the dataset and prevents accidental exclusion of minority classes. Consequently, the machine learning models are trained and evaluated on a realistic sample of the overall traffic distribution.

By using a stratified 10% subset, the experiments achieve a practical trade-off between computational efficiency and data fidelity. The resulting dataset retains the essential behavioral patterns, attack characteristics, and diversity required for robust model training. It also significantly reduces the processing time which is needed for feature preparation, model tuning, and repeated evaluations.

After subset extraction, several preprocessing steps were applied to prepare the data for supervised machine learning. Since the BoT-IoT dataset contains different types of attributes and large-scale network flow records, preprocessing is necessary to improve model reliability. These steps help improve the reliability of the model training, reduce computational complexity, and minimize the risk of biased or unstable model behavior.

First, the feature subset identified by the original BoT-IoT dataset authors was considered for model development. The original authors selected this subset using the Correlation Coefficient and Joint Entropy criteria to remove irrelevant and non-informative attributes. Consequently, features such as source IP address, destination IP address, and other identifiers were excluded because they may not generalize well to new or unseen network environments. Although such attributes may increase classification accuracy on a specific dataset, they can cause the model to learn dataset-specific identifiers rather than generalizable traffic behavior. Therefore, only the feature subset previously selected by the dataset authors was used as the primary input for model training.

Second, the target labels were encoded into numerical class values. Since the problem is formulated as a supervised multi-class classification task, each network traffic record was assigned to one of the predefined classes. Normal traffic was represented as class 0, while the remaining class labels represented different categories of cyber-attacks.

Third, the dataset was carefully examined to identify missing, infinite, and invalid values. Network traffic datasets may contain undefined or infinite values due to abnormal flow durations, division-by-zero operations, incomplete records, or feature extraction errors. Such values can negatively affect the training process, especially for algorithms that rely on numerical optimization. Therefore, invalid values were either removed or replaced using appropriate statistical values depending on the number of affected samples and the nature of the feature.

Fourth, feature normalization was applied to ensure that numerical attributes scaled to a comparable range. The selected features have different magnitudes and units; therefore, normalization is especially important for algorithms such as Support Vector Machine, Logistic Regression, and Artificial Neural Network, which are sensitive to feature scale. In this study, numerical features were standardized using:

$$z = \frac{x - \mu}{\sigma} \tag{3.2}$$

where x represents the original value of the feature, μ is the mean of the feature, and σ is the corresponding standard deviation. This transformation produces features with approximately zero mean and unit variance, improving training stability and convergence.

Finally, the dataset was divided into training and testing subsets by using stratified sampling. Stratification ensures that the class distribution in both subsets remains consistent with the distribution of the selected 10% dataset. This step is important in imbalanced classification problems because random splitting without stratification may lead to severe underrepresentation of minority attack classes in either the training or testing subset.

To prevent data leakage, imbalance handling techniques such as random oversampling, random undersampling, and SMOTE were applied only to the training subset. The testing subset was kept unchanged and imbalanced in order to simulate a realistic cyber-attack detection scenario. This ensures that the reported performance reflects the ability of the trained models to detect cyber-attacks under real-world class distribution conditions.

3.3. Decision Function and Learning Objective

A parametric decision function is defined as:

$$\pi_{\boldsymbol{\theta}}: \mathbb{R}^d \rightarrow \mathcal{Y} \tag{3.3}$$

where $\boldsymbol{\theta} \in \mathbb{R}^N$ denotes the model parameters.

Given an input feature vector **x**, the predicted class label is attained as:

$$\hat{y} = \arg max_{y \in \mathcal{Y}} Pr_{\theta}(y \mid \mathbf{x}) \tag{3.4}$$

Model parameters are learned by minimizing the empirical risk over the training dataset:

$$\min_{\theta} \sum_{m=1}^{M} \ell\left(y_m, \pi_{\boldsymbol{\theta}}(\mathbf{x}_m)\right) \tag{3.5}$$

where $\ell(\cdot)$ represents a suitable loss function.

However, when the class distributions are imbalanced, the standard empirical risk minimization framework tends to favor the majority classes. As a result, model may fail to make reliable decision for uncommon yet critical attack types.

3.4. Imbalance Handling Strategies

To overcome the undesirable impact of class imbalance, two major kinds of imbalance handling strategies are investigated: Data-level methods, Algorithm-level methods. These strategies are systematically evaluated and compared in terms of detection performance and computational complexity.

3.4.1. Data-Level Approaches

Data-level methods seek to rebalance the dataset before the model training phase by modifying the class distribution.

Let the rebalanced dataset be denoted as $\widetilde{\mathcal{D}}$. The optimization problem then becomes:

$$min_{\theta} \sum_{(\mathbf{x}_m, y_m) \in \widetilde{\mathcal{D}}} \ell\left(y_m, \pi_{\boldsymbol{\theta}}(\mathbf{x}_m)\right) \tag{3.6}$$

Random Oversampling

Random oversampling increases the number of minority classes instances by replicating existing samples such that:

$$n_{\text{minority}}^{\text{new}} \approx n_{\text{majority}} \tag{3.7}$$

This method increases recall for minority classes. However, increasing of training time and the chance of overfitting because of duplicated samples is inevitable in this approach (Galar, Fernandez, Barrenechea, Bustince, & Herrera, 2012).

Random Undersampling

Random undersampling balances the dataset by eliminating the instance of the majority classes:

$$n_{\text{majority}}^{\text{new}} \approx n_{\text{minority}} \tag{3.8}$$

This technique reduces training time and it is less hardware intensive but the chance of underfitting is high due to the potential loss of discriminative samples (Galar, Fernandez, Barrenechea, Bustince, & Herrera, 2012).

SMOTE (Synthetic Minority Over-Sampling Technique)

SMOTE generates synthetic minority instances through interpolation between neighboring samples rather than replicating the existed samples. For a minority instance $\mathbf{x}_i$, a synthetic sample is generated as:

$$\mathbf{x}_{\text{new}} = \mathbf{x}_i + \delta.(\mathbf{x}_{\text{nn}} - \mathbf{x}_i) \tag{3.9}$$

Where $\mathbf{x}_{\text{nn}}$ is a random nearest neighbor of $\mathbf{x}_i$ and $\delta \sim \mathcal{U}(0,1)$.

The main advantages are that it reduces the likelihood of overfitting than random oversampling, since it generates new instances rather than duplicating existing ones. Moreover, it helps improve the decision-boundary learning for minority classes by providing more informative training examples. However, it also has certain limitations. It increases the dataset size, which raises computational and memory requirements, and its inherently random nature can introduce noisy or overlapping synthetic samples (Meenakshisundaram & G, 2025).

3.4.2. Algorithm-Level Approaches

Instead of modifying the data distribution before the training phase, algorithm-level methods address class imbalance by adjusting the learning objective. So, the model is trained in a way that gives more weight to minority classes or the misclassification cost of those classes is higher than the majority ones.

Cost-Sensitive Learning

A weighted loss function is employed:

$$min_\theta \sum_{m=1}^{M} w_{y_m} \ \ell\big(y_m, \pi_{\boldsymbol{\theta}}(\mathbf{x}_m)\big) \tag{3.10}$$

where the class weight w_k is defined as:

$$w_k = \frac{M}{n_k} \tag{3.11}$$

This approach also known as Class Weighting, dedicates higher penalties to misclassification errors on minority classes and it does not require expanding the dataset and preserves all informative samples. It also maintains low memory overhead (Khan, Hayat, Bennamoun, Sohel, & Togneri, 2018).

3.5. Multi-Objective Optimization Framework

The main goal of this thesis is:

1. To maximize detection accuracy for minority attack classes
2. To minimize computational cost and complexity, with training time and testing time used as the representative measure of computational cost.

The unified optimization problem is defined as:

$$min_{\theta\ S}[\mathcal{L}_{\text{imbalance}}(\boldsymbol{\theta}, S) + \lambda\, C(\boldsymbol{\theta})] \tag{3.12}$$

Where $\boldsymbol{\theta}$ represents the model parameters, S represents the imbalance handling strategy, $\mathcal{L}_{\text{imbalance}}$ represents the classification loss under class imbalance, $C(\boldsymbol{\theta})$ denotes computational cost, and λ controls the trade-off between performance and complexity.

Finally, the problem can be considered as a Pareto optimization problem, meaning model–strategy combinations that provide optimal trade-offs between accuracy and efficiency.

3.6. Models Considered

The following supervised learning models are evaluated:

3.6.1. Random Forest

Random Forest (RF) is an ensemble machine learning algorithm that combines multiple decision trees to perform classification. Instead of making predictions based on a single decision tree, it builds a large number of trees during the training phase and combines their outputs through a voting mechanism. This improves prediction accuracy and reduces the risk of overfitting.

During the training process, each tree is built using a randomly selected subset of the training data. Moreover, when a split is created within a tree, only a random subset of features is considered. These random selections increase the diversity of the trees and improve the overall robustness of the model.

For classification problems, each tree produces a class prediction, and the final output is determined through majority voting. The class that receives the highest number of votes is selected as the final prediction.

Random Forest was selected for this study because it can effectively model complex and nonlinear relationships that commonly exist in network traffic data. It is also relatively resistant to noise, capable of handling large datasets, and naturally supports multiclass classification problems.

Hyperparameter Optimization

The performance of Random Forest depends on several hyperparameters that determine the structure and complexity of the decision trees. To identify suitable values of parameter, Randomized Search Cross-Validation was employed. Compared with Grid Search, Randomized Search evaluates a limited number of randomly selected parameter combinations, reducing computational cost while maintaining effective optimization performance.

Hyperparameter tuning was performed using three-fold cross-validation, and the weighted F1-score was used as the optimization metric. To reduce computational complexity, the search process was conducted on a representative stratified subset of the training data.

The following hyperparameters were optimized:

Table 3.1 Random Forest Hyperparameters

Hyperparameter	Description
n_estimators	Number of trees in the forest
max_depth	Maximum depth of each tree
min_samples_split	Minimum number of samples required to split a node
min_samples_leaf	Minimum number of samples required in a leaf node
max_features	Number of features considered when searching for the best split
bootstrap	Whether bootstrap sampling is used during training

After determining the optimal parameter combination, the final model was trained using the complete training dataset.

To investigate the impact of class imbalance, Random Forest was evaluated under five experimental settings: standard training, random oversampling, random undersampling,

SMOTE, and cost-sensitive learning. The implementation details of these imbalance handling techniques were presented in Section 3.4 (Ho, 1995).

3.6.2. Gradient Boosting

Gradient Boosting (GB) is an ensemble learning algorithm that improves predictive performance by building a sequence of decision trees. Unlike Random Forest, where trees are trained independently, Gradient Boosting constructs trees in a sequential manner. Each new tree is trained to reduce the errors made by the current ensemble, allowing the model to gradually improve its predictions.

The final model is obtained by combining the outputs of all trees, where each tree contributes to correcting the mistakes of the previous ones. Through this iterative learning process, Gradient Boosting can capture complex patterns and nonlinear relationships within the data.

Gradient Boosting was selected because of its strong predictive capability and its effectiveness when applied to structured datasets. It has been successfully used in many classification problems and often achieves high performance when its hyperparameters are properly optimized. However, the algorithm is sensitive to parameter selection and may overfit if excessive model complexity is allowed.

Hyperparameter Optimization

To identify an effective model configuration, Randomized Search Cross-Validation was employed. Hyperparameter tuning was performed using five-fold cross-validation, and the F1-score was used as the optimization metric. A stratified subset of the training data was used during the search process to reduce computational requirements.

The following hyperparameters were optimized.

Table 3.2: Gradient Boosting Hyperparameters

Hyperparameter	Description
n_estimators	Number of boosting stages
learning_rate	Contribution of each tree to the final model
max_depth	Maximum depth of individual trees
min_samples_split	Minimum number of samples required to split a node
min_samples_leaf	Minimum number of samples required in a leaf node
subsample	Fraction of training samples used at each boosting stage

After selecting the optimal hyperparameter combination, the final model was retrained using the complete training dataset.

Similar to Random Forest, Gradient Boosting was evaluated under five experimental settings: standard training, random oversampling, random undersampling, SMOTE, and cost-sensitive learning (R, Ayachit, Patil, & Singh, 2020).

3.6.3. Logistic Regression

Logistic Regression (LR) is a supervised machine learning algorithm that estimates the probability of a sample belonging to a particular class based on its input features. Although this algorithm is developed for binary classification, it can be extended to multiclass problems. It is also widely used as a baseline classifier due to its simplicity, efficiency, and interpretability.

Unlike tree-based methods such as Random Forest and Gradient Boosting, Logistic Regression assumes a linear relationship between the input features and the decision boundary. The model predicts class probabilities using the logistic function and assigns each sample to the class with the highest estimated probability.

For binary classification, Logistic Regression first computes a linear combination of the input features:

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n a_n x_n \tag{3.13}$$

where $(x_1, \ldots, x_n)$ are the input features, β_0 is the intercept, and $(\beta_1, \ldots, \beta_n)$ are the model coefficients. The resulting value is transformed into a probability using the logistic, or sigmoid, function:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.14}$$

The logistic function maps any real-valued input (z) to a value between 0 and 1, allowing the output to be interpreted as the estimated probability of belonging to the positive class.

Logistic Regression was selected in this study because of its low computational complexity, fast training time, and strong interpretability. Furthermore, it provides an effective baseline for evaluating whether more complex models offer meaningful performance improvements in cyber-attack detection.

Since Logistic Regression is sensitive to feature magnitude, feature standardization was incorporated into the training pipeline. Standardization ensures that all features contribute equally during the optimization process and improves stability and convergence of the model.

Hyperparameter Optimization

The performance of Logistic Regression depends on several hyperparameters related to regularization and optimization. To identify suitable values of parameter, Grid Search Cross-Validation was employed. Unlike Randomized Search, Grid Search evaluates all parameter combinations within a predefined search space, ensuring an exhaustive exploration of candidate configurations.

Hyperparameter tuning was performed using five-fold cross-validation. The optimal parameter combination was selected based on the highest cross-validation accuracy.

The following hyperparameters were optimized.

Table 3.3 Logistic Regression Hyperparameters

Hyperparameter	Description
penalty	Specifies the regularization method used to reduce model complexity and prevent overfitting.
C	Controls the strength of regularization. Smaller values apply stronger regularization, while larger values allow the model to fit the training data more closely.
solver	Optimization algorithm used to estimate the model parameters during training. Different solvers support different regularization methods.
max_iter	Maximum number of iterations allowed for the optimization process to converge.

The search space included L1 and L2 regularization methods, multiple values of the regularization parameter (C), and different optimization solvers. After the optimal parameter combination was identified, the final model was trained using the complete training dataset (Pathan, Prabhu, & Siddalingaswamy, 2018).

To ensure a fair comparison, Logistic Regression was evaluated using the same experimental framework and imbalance handling strategies applied to the other models.

3.6.4. Artificial Neural Network

An Artificial Neural Network (ANN) is a supervised learning model inspired by the basic principles of biological neural systems. It is typically organized into three main components: an input layer, one or more hidden layers, and an output layer. Each layer contains multiple interconnected units, commonly called neurons, which process and transmit information. As data passes through the network, these neurons apply weighted transformations that allow the model to learn relationships between input features and output classes. During training, the network adjusts the connection weights using backpropagation, an optimization process that iteratively minimizes prediction errors and improves the model's performance.

Each neuron first computes a weighted sum of its inputs and then passes the result through a nonlinear activation function. This mechanism allows the network to learn and represent complex, nonlinear patterns in the data. In multiclass classification tasks, the output layer usually uses a softmax activation function to convert the network's outputs into class probabilities, and the class with the highest probability is chosen as the final prediction.

ANN was selected for this study because of its ability to model highly nonlinear decision boundaries and to learn hierarchical feature representations directly from raw input data. Unlike tree-based methods, ANNs do not rely on feature splitting and can generalize effectively across diverse and high-dimensional network traffic patterns. Their flexibility in architecture design also makes them well-suited for complex multiclass classification tasks such as cyber-attack detection.

Since ANN is sensitive to feature magnitude, feature standardization was incorporated into the training pipeline to ensure stable and efficient convergence during optimization.

Hyperparameter Optimization

To identify an effective network configuration, Randomized Search Cross-Validation was employed. Hyperparameter tuning was performed using five-fold cross-validation, and the weighted F1-score was used as the optimization metric. A stratified subset of the training data was used during the search process to reduce computational requirements (Pal & Mitra, 1992). The following hyperparameters were optimized.

Table 3.4 Artificial Neural Network Hyperparameters

Hyperparameter	Description
hidden_layer_sizes	Number of hidden layers and the number of neurons in each layer
activation	Activation function applied at each hidden layer node (e.g., ReLU, tanh)
solver	Optimization algorithm used to update network weights during training
alpha	L2 regularization parameter controlling the penalty on large weights to prevent overfitting
learning_rate	Strategy for adjusting the learning rate during training
max_iter	Maximum number of iterations allowed for the optimization process to converge
batch_size	Number of training samples used in each weight update step

After selecting the optimal hyperparameter combination, the final model was retrained using the complete training dataset.

Similar to the other models, the Artificial Neural Network was evaluated under five experimental settings: standard training, random oversampling, random undersampling, SMOTE, class weighting, and cost-sensitive learning.

3.7. Evaluation Metrics

Considering the imbalanced nature of the dataset, overall accuracy is not a sufficient metric to evaluate the models. The following metrics are implemented:

- Precision
- Recall
- F1-score (macro and weighted)
- Learning Curve
- Training time

The optimal model is the one that meets following criteria:

- Maximizes minority-class recall and macro F1-score
- Minimizes training time

3.8. Experimental Environment

All experiments were conducted on the same computing system to ensure a consistent comparison of model performance and computational cost. The system was equipped with an AMD Ryzen 5 7500F processor, 16 GB of DDR5-5200 RAM, and an NVIDIA GeForce RTX 4060 GPU with 8 GB of dedicated memory.

The experiments were implemented in Python 3.14.7, which was the latest stable version available at the time of experimentation. Common machine-learning and data-analysis libraries were used, including NumPy, Pandas, Scikit-learn, imbalanced-learn, and Matplotlib. TensorFlow was used for the implementation and evaluation of deep-learning models.

The same hardware and software environment was maintained throughout the experiments. This was particularly important because training time was considered as a representative measure of computational cost, and using a consistent computational environment allowed the training times of different models and imbalance-handling strategies to be compared fairly.

3.9. Experimental Procedure

The experimental workflow includes the following steps:

1. Data preprocessing and feature normalization
2. Train–test split
3. Application of imbalance handling strategy
4. Hyperparameter optimization
5. Model training
6. Performance evaluation
7. Computational complexity measurement
8. Comparative analysis

3.10. Chapter Summary

In this chapter, a multi-objective optimization framework was designed and formulated as an imbalance-aware supervised multi-class classification problem to detect cyber-attacks. This framework incorporated data-level and algorithm-level techniques to handle imbalances effectively, and these components were formally defined and integrated into a unified evaluation methodology. Finally, the next chapter presents experimental results and a comprehensive comparative analysis of our models with an emphasis on both imbalance handling and hardware efficiency

CHAPTER 4
RESULTS AND DISCUSSION

4.1. Introduction

This chapter presents and discusses the experimental results produced by the machine learning models developed in this study. It examines how different class imbalance handling techniques, namely random oversampling, random undersampling, SMOTE, and class weighting, affect the performance of each classifier. The analysis focuses on predictive performance, class-specific behavior, model convergence, and computational efficiency in order to determine the most effective methods for cyber-attack detection.

4.2. Experimental Setup Summary

The experiments were performed using the stratified 10% subset of the BoT-IoT dataset described in Chapter 3. The data were divided into training and testing sets using an 80:20 stratified split to preserve the original class distribution. Hyperparameter optimization was conducted using Grid search and Randomized Search with cross-validation, and all classifiers were evaluated under identical experimental conditions. Model performance was assessed using accuracy, precision, recall, F1-score, confusion matrices, learning curves, and computational efficiency measures including training time and prediction time.

4.3. Random Forest Results

Random Forest was selected as the first classifier for evaluation due to its strong performance on high-dimensional classification problems and its inherent resistance to overfitting. The optimized hyperparameters configuration used throughout the experiments is presented in Table 4.1. The following sections examine the effect of each imbalance handling strategy on classification performance, class-level prediction behavior, and computational cost.

Table 4.1 Optimized Hyperparameters of Random Forest Classifier

Hyperparameter	Value
n_estimators	**500**
max_depth	**None**
min_samples_split	**5**
min_samples_leaf	**1**
max_features	**sqrt**
bootstrap	**False**

max_depth = None: there is no explicit limit on the depth of each tree. A tree keeps growing until another stopping condition is met, such as the minimum number of samples required to split a node.

max_features = "sqrt": at each split, the model considers only the square root of the total number of input features when searching for the best split.

4.3.1. Standard Random Forest

The baseline Random Forest model achieved very strong classification performance. The model obtained an accuracy of 99.74%, with macro-averaged precision, recall, and F1-score values of 99.64%, 98.76%, and 99.19%, respectively. These high macro-averaged metrics indicate that predictive performance remained consistent across all attack categories, including minority classes.

The relatively small difference between precision and recall suggests that the model maintained a balanced trade-off between false positive and false negative predictions. These results indicate that Random Forest is capable of handling the class distribution present in the dataset, achieving near-perfect classification performance even without the use of a dedicated imbalance handling strategy.

The confusion matrix presented in Figure 4.1 further confirms the effectiveness of the baseline Random Forest model. Most attack categories were classified correctly, with only a small number of misclassifications observed among the minority classes.

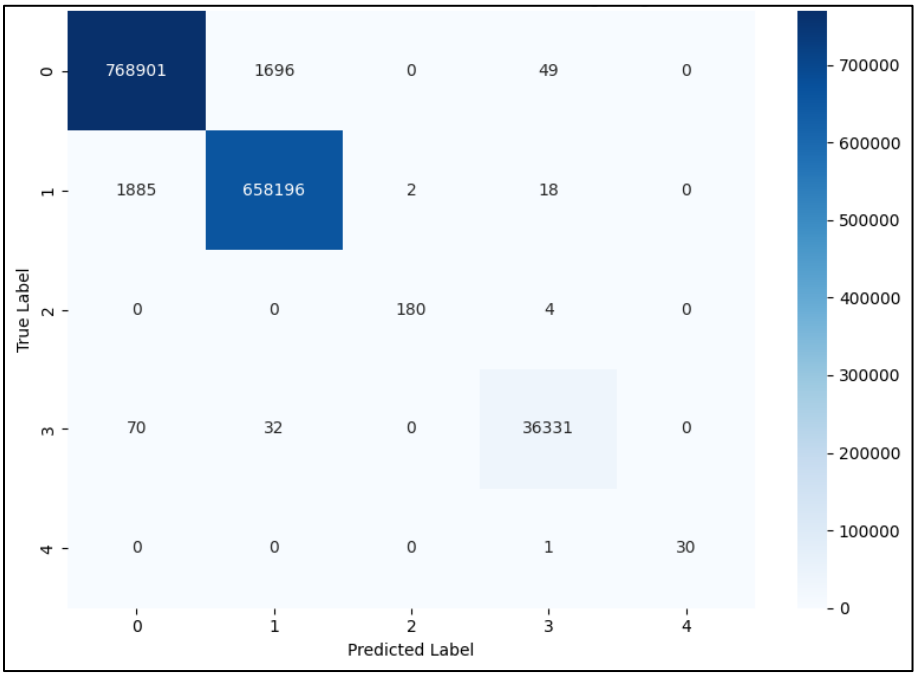

Figure 4.1 Confusion Matrix of Standard Random Forest

Due to the high computational cost of processing the entire dataset, the learning curve was generated using a random subset of 100,000 samples. As shown in Figure 4.2, the curve demonstrates robust model performance and strong generalization capability, with no evident signs of overfitting. The shaded region around the learning curve represents the variability of the validation F1 scores across the cross-validation folds. It is typically calculated as the mean validation score plus and minus one standard deviation. A wider shaded region indicates greater variation and lower stability, whereas a narrower region indicates that the model produces more consistent validation results as the training-set size increases. In particular, the steady increase in F1-score as the training sample size grows indicates that the model effectively benefits from additional data to improve its predictive performance.

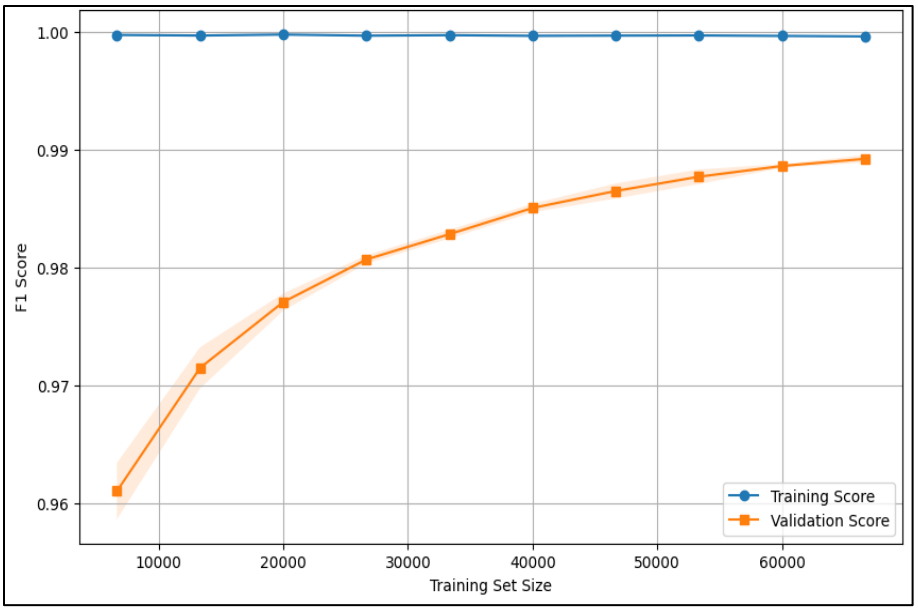

Figure 4.2 Learning Curve of Standard Random Forest

4.3.2. Random Forest with Oversampling Technique

Random oversampling provided only incremental gains, pushing overall metrics to approximately 99.75%. Despite these strong aggregate figures, the confusion matrix (Figure 4.3) reveals a persistent issue: the model remains less efficient at identifying the 'theft information' label. This suggests that simple oversampling may not be enough to overcome the challenges associated with this specific minority class. Furthermore, the learning curve (Figure 4.4) indicates that the model generalizes well, with no clear signs of overfitting or underfitting.

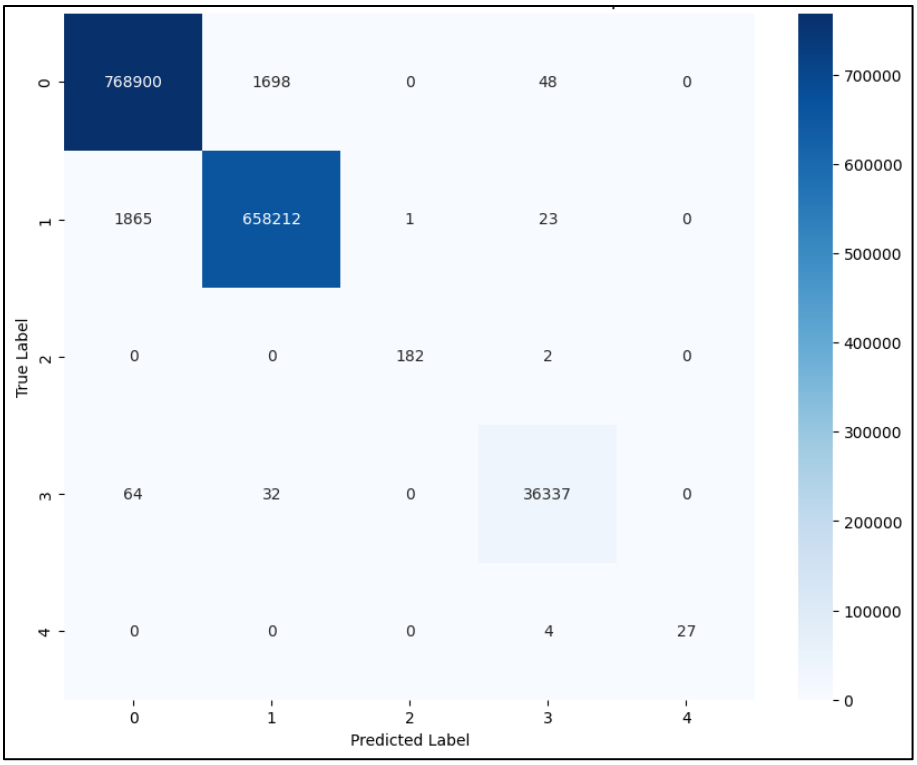

Figure 4.3 Confusion Matrix of Random Forest with Oversampling Technique

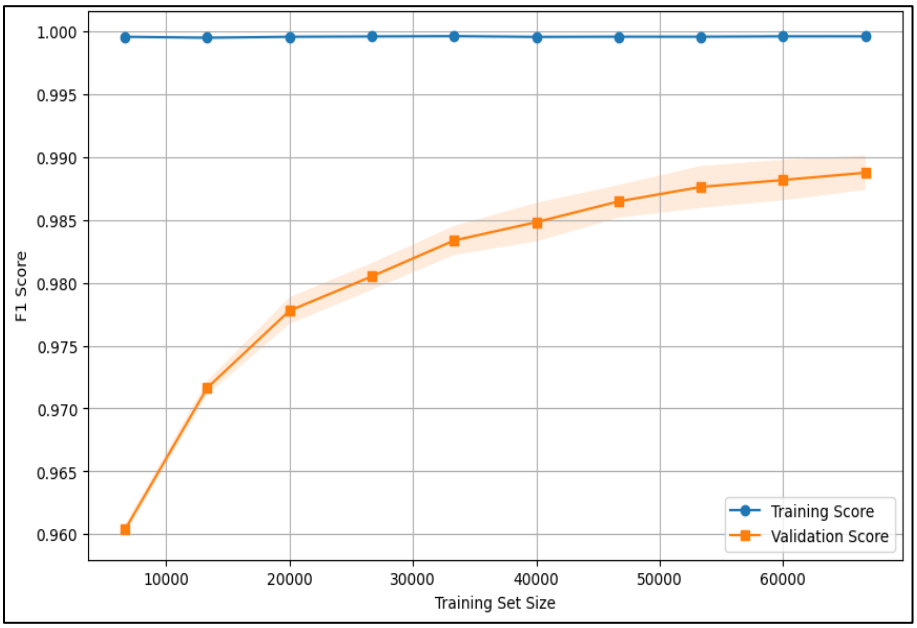

Figure 4.4 Learning Curve of Random Forest with Oversampling Technique

4.3.3. Random Forest with SMOTE Technique

Similarly, SMOTE achieved performance comparable to the baseline model, with all evaluation metrics remaining around 99.74%. These findings suggest that the synthetic minority-class samples provided limited benefit, as the baseline Random Forest was already capable of learning highly discriminative decision boundaries. Furthermore, the learning curves in Figure 4.5 for this approach indicate a well-fit model with no signs of overfitting or underfitting. However, a granular analysis of the confusion matrices (Figure 4.6) reveals that SMOTE was actually less effective than the baseline in detecting Label 4 (the primary minority class), despite the increased computational cost. Conversely, a slight improvement was observed in the detection of Label 2 (the second minority class), suggesting that SMOTE's impact varies significantly across different minority clusters.

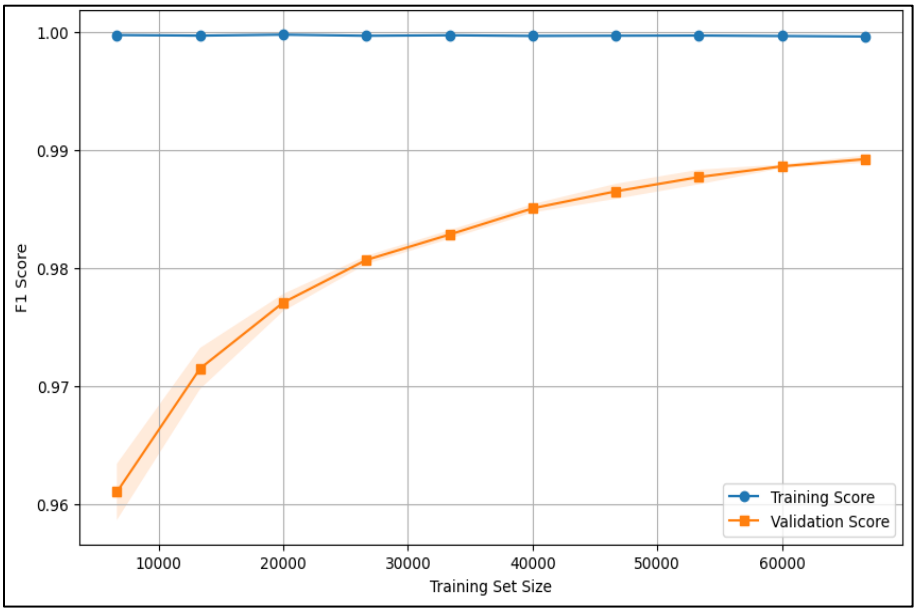

Figure 4.5 Learning Curve of Random Forest with SMOTE Technique

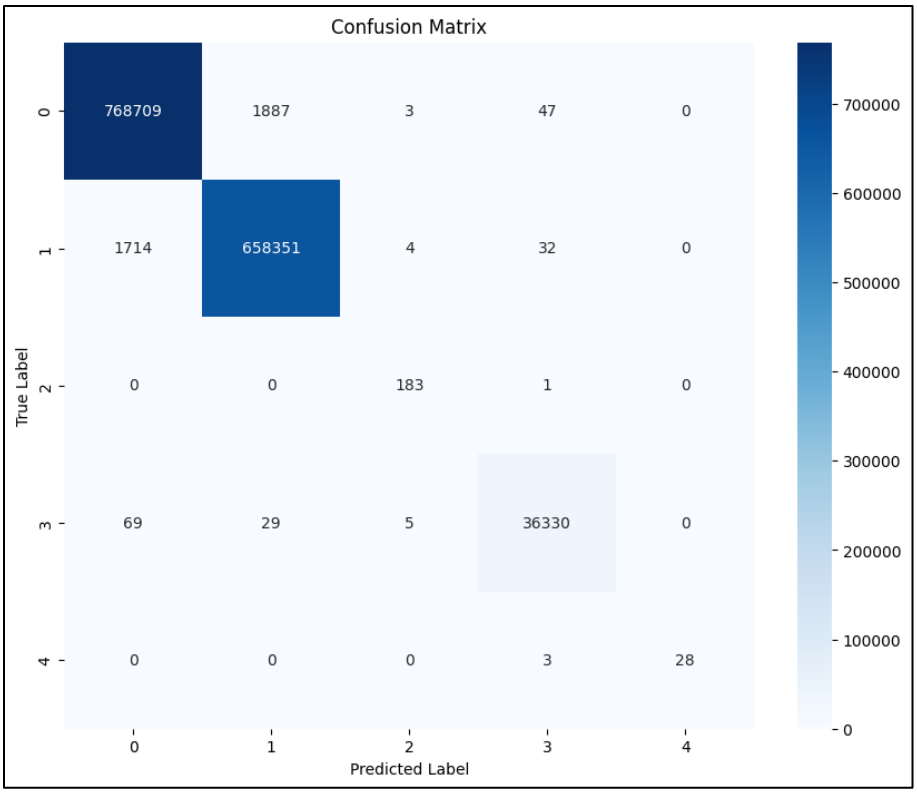

Figure 4.6 Confusion Matrix of Random Forest with SMOTE Technique

4.3.4. Random Forest with Undersampling Technique

The under-sampling method led to a decline in overall model performance, with accuracy dropping to 78.47%, precision to 79.24%, recall to 78.47%, and F1-score to 78.54%. At the same time, it noticeably improved the model's ability to detect minority classes. As shown in the confusion matrix (Figure 4.7), the model correctly classified all samples of Label 2 and Label 4, and it also performed well on Label 3, indicating that under-sampling helped the classifier pay greater attention to the less frequent classes. However, this improvement came with a clear trade-off, the model made considerably more errors between the dominant classes, particularly between Label 0 and Label 1. Therefore, although under-sampling enhanced minority-class detection, it reduced the model's overall stability and accuracy. Also, analysis of the learning curve indicates a state of high variance (Figure 4.8), the model achieves a perfect training F1-score of 1.00 but fails to generalize, with validation scores plateauing

significantly lower. This suggests that the reduced training set lacks the requisite informational density to represent the global feature space.

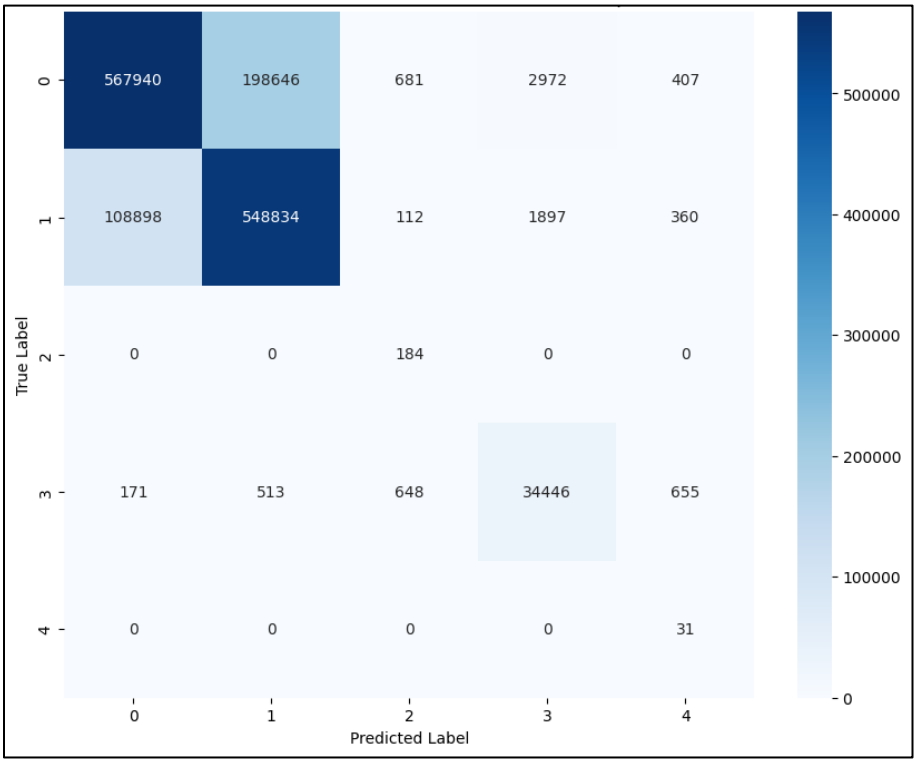

Figure 4.7 Confusion Matrix of Random Forest with Undersampling Technique

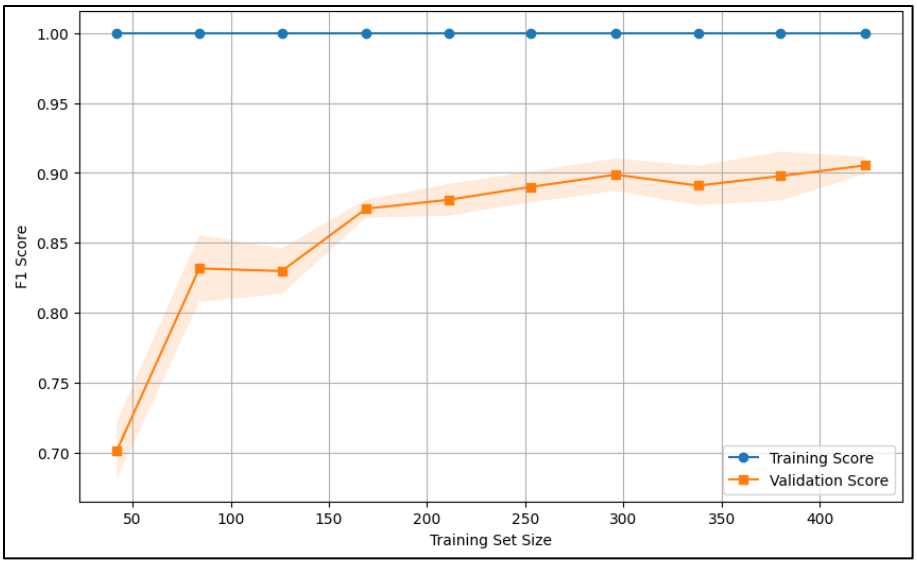

Figure 4.8 Learning Curve of Random Forest with undersampling Technique

4.3.5. Random Forest with Class Weight Technique

Class weighting also maintained very high accuracy (99.74%) and precision (99.62%), but its macro recall declined to 97.16% and the macro F1-score fell to 98.30%. This drop suggests that some minority-class instances remained more difficult to classify correctly, even with the weighting mechanism in place. Although the overall performance was still strong, class weighting was less effective than the baseline, oversampling, and SMOTE approaches in preserving balanced performance across all classes (Figure 4.9 and Figure 4.10).

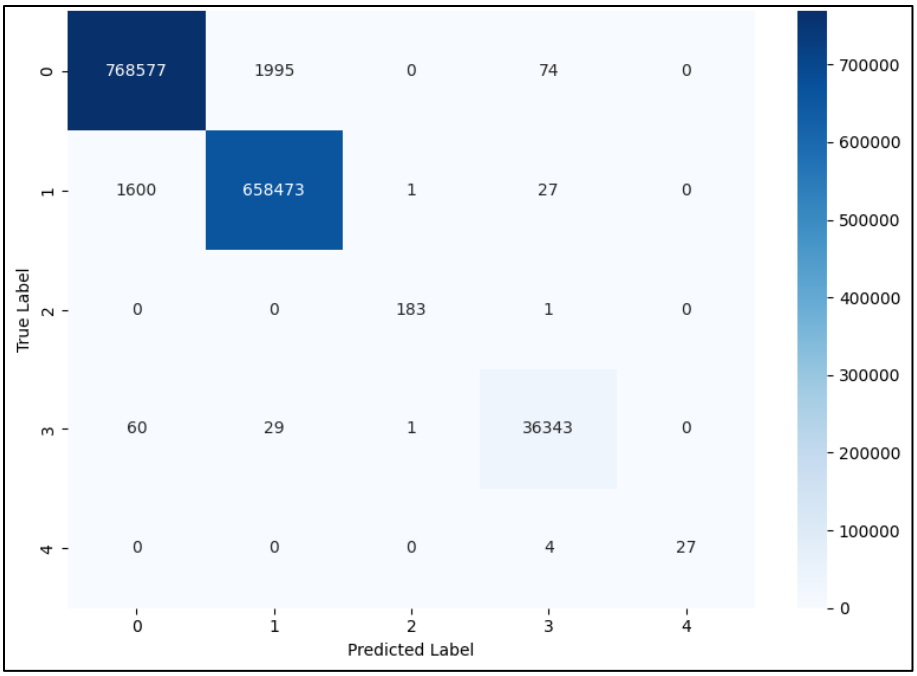

Figure 4.9 Confusion Matrix of Random Forest with Class Weighting Technique

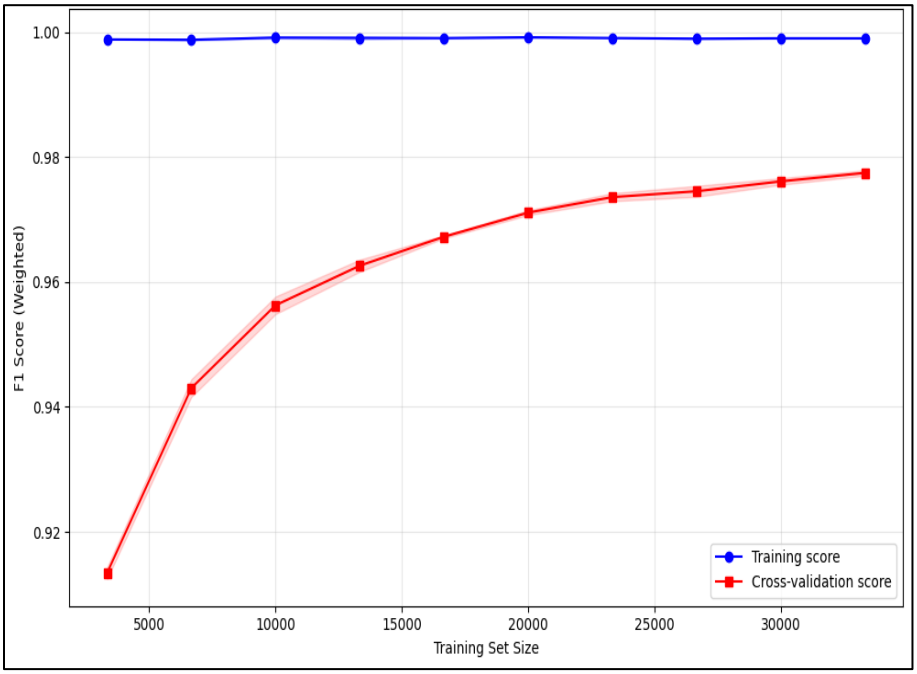

Figure 4.10 Learning Curve of Random Forest with Class Weighting Technique

Figure 4.11 through Figure 4.15 present the accuracy, F1-score, precision, recall, and computational efficiency comparisons across all five Random Forest configurations. As illustrated, the baseline Random Forest, also referred as Standard RF achieved near-perfect performance across all metrics, leaving limited room for improvement through additional balancing strategies. Random undersampling demonstrated the lowest training and inference times (Figure 4.15); however, this computational advantage was accompanied by a notable degradation in overall metrics and poor generalization for the dominant classes, as previously evidenced by its learning curve behavior. Conversely, oversampling and SMOTE introduced substantially higher computational costs while paradoxically underperforming the baseline in minority-class detection. Class weighting offered no meaningful metric gains over the baseline, yet remained computationally competitive. Overall, these results confirm that Random Forest is inherently robust to class imbalance in the studied dataset, with the most favorable trade-off between classification effectiveness and computational efficiency, and random undersampling proving the least suitable strategy due to the significant information loss it introduces.

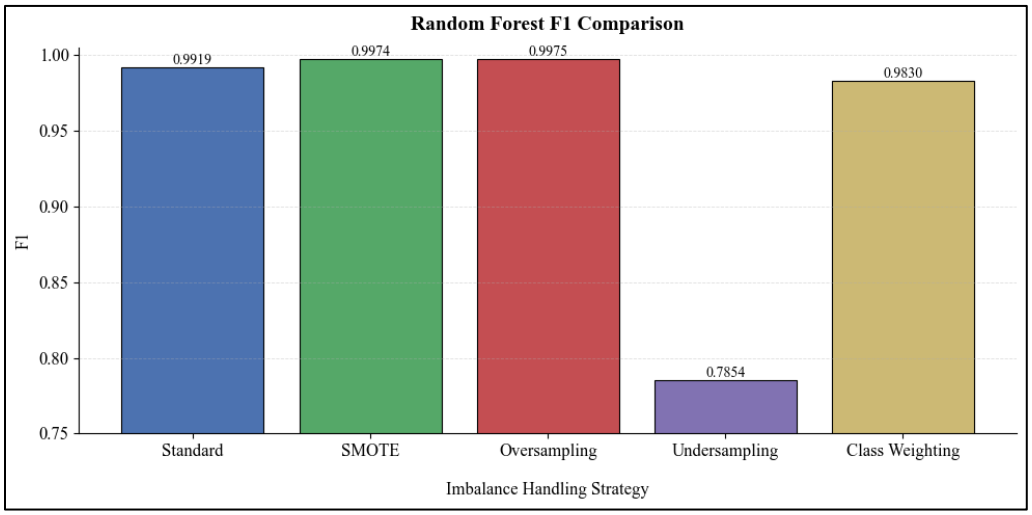

Figure 4.11 Random Forest F1 Score Comparison

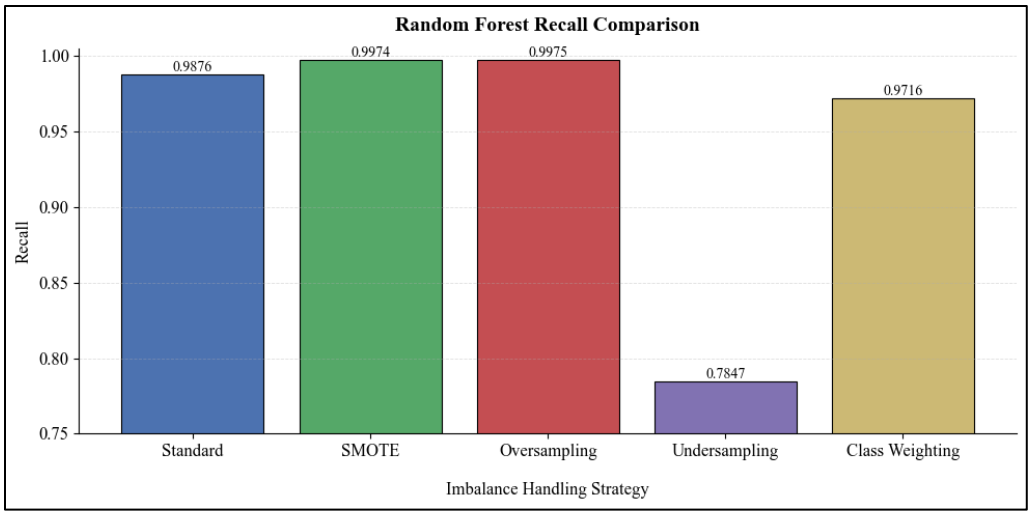

Figure 4.12 Random Forest Recall Comparison

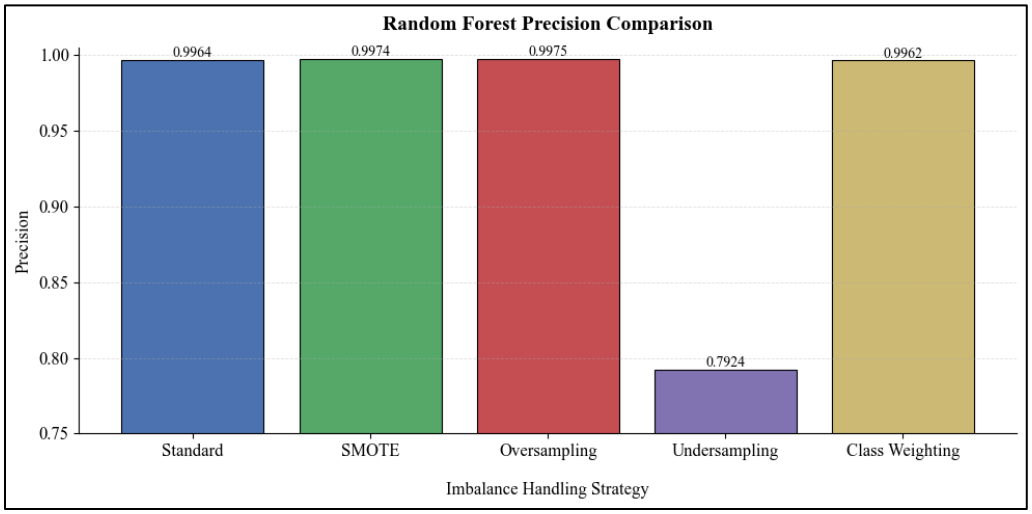

Figure 4.13 Random Forest Precision Comparison

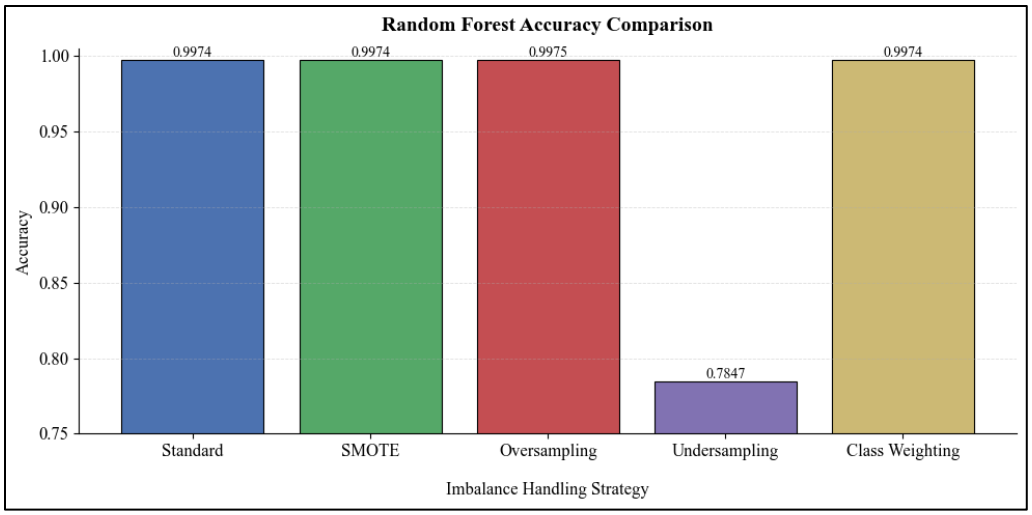

Figure 4.14 Random Forest Accuracy Comparison

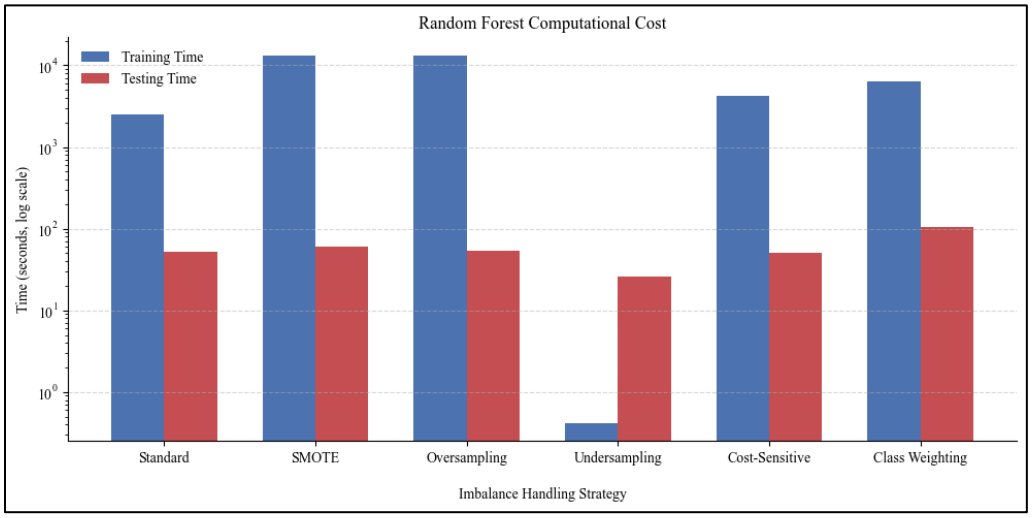

Figure 4.15 Random Forest Training Time and Test Time Comparison

4.4. Gradient Boosting Results

Gradient Boosting was selected as the second classifier for evaluation. The optimized hyperparameter configuration used throughout the experiments is presented in Table 4.2. The following sections investigate the influence of different class imbalance handling strategies on classification performance, class-level prediction behavior, generalization capability, and computational efficiency.

Table 4.2 Optimized Hyperparameters of Gradient Boosting Classifier

Hyperparameter	Value
n_estimators	**171**
learning_rate	**0.0849**
max_depth	**2**
min_samples_split	**12**
min_samples_leaf	**15**
subsample	**0.8395**

4.4.1. Standard Gradient Boosting

The model achieved an overall accuracy of 84.15%, with macro-averaged precision, recall, and F1-score values of 72.67%, 76.32%, and 61.98%, respectively. In comparison with the Random Forest classifier, the overall predictive performance of Gradient Boosting was notably lower. While the model maintained reasonable classification capability for the dominant classes, as it is shown in Figure 4.16 the relatively low macro F1-score indicates difficulties in achieving balanced performance across all attack categories, particularly for second minority class. The difference between macro precision and recall further suggests that the classifier produced a noticeable number of false positive and false negative predictions for the less represented attack classes.

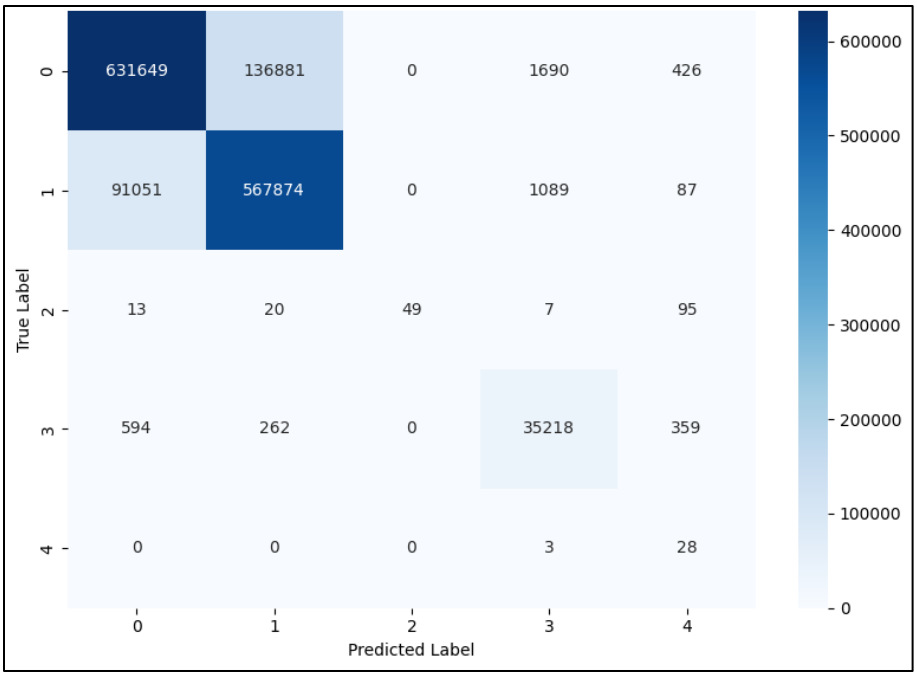

Figure 4.16 Confusion Matrix of Standard Gradient Boosting

As illustrated in Figure 4.17, despite minor fluctuations, the training and cross-validation curves maintain a consistent and relatively narrow gap throughout the experiment. These variations, particularly the notable dips at certain training sizes, are likely attributable to stochastic sampling effects and the intrinsic class imbalance within the dataset, rather than fundamental instability. The convergence of both curves at a weighted F1-score of approximately 84% signifies that the model has reached its asymptotic performance limit. This suggests a degree of structural bias in the Gradient Boosting configuration, which, despite good generalization, prevents it from capturing the higher-order complexities successfully identified by the Random Forest classifier.

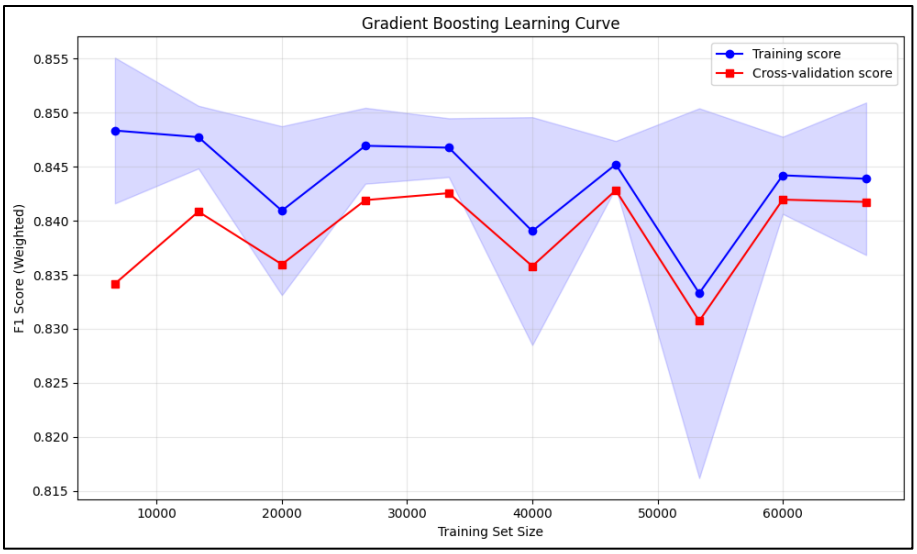

Figure 4.17 Learning Curve of Standard Gradient Boosting

4.4.2. Gradient Boosting with Oversampling Technique

The application of random oversampling produced mixed effects on the performance of the Gradient Boosting classifier. The model achieved an overall accuracy of 84.12%, which was almost identical to that of the baseline Gradient Boosting model. However, macro recall increased markedly from 76.32% to 90.96%, while the macro F1-score rose from 61.98% to 71.47%. These improvements suggest that oversampling helped the classifier detect minority-class instances more effectively by increasing their representation during training. At the same time, macro precision declined to 67.18%, indicating that the gain in minority-class detection came with a higher number of false positive predictions.

The confusion matrix presented in Figure 4.18 further illustrates this behavior. The model achieved perfect detection of Label 2 and maintained near similar performance as baseline Gradient Boosting on Labels 3 and 4, demonstrating a significant improvement in second minority-class recognition. However, this gain was achieved at the expense of increased confusion between the two dominant classes (Label 0 and Label 1), resulting in a larger number of misclassified majority-class samples. Consequently, oversampling shifted the classifier toward a more balanced treatment of all classes while slightly reducing its discrimination capability for the dominant category.

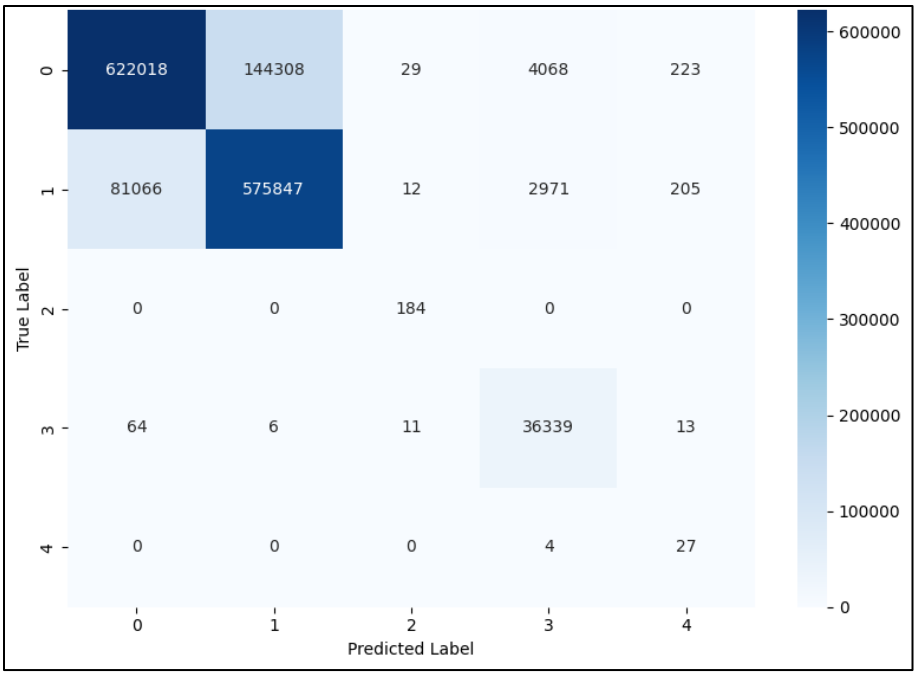

Figure 4.18 Confusion Matrix of Gradient Boosting with Oversampling Technique

The learning curve illustrated in Figure 4.19 demonstrates that while the introduction of Random Oversampling has successfully narrowed the generalization gap compared to the baseline model, it introduces a distinct trade-off between generalization and local stability. On one hand, the training and cross-validation curves remain closely aligned across all training set sizes, indicating that the model generalizes consistently to unseen data without suffering from classic, severe overfitting. On the other hand, the classifier still exhibits localized performance volatility and substantial variance across different training size most notably around the 40,000-sample mark. This instability is highly likely a consequence of exact sample duplication inherent to Random Oversampling; rather than learning a broader representation of the underlying distribution, the sequential Gradient Boosting algorithm becomes overly sensitive to these repeated minority class instances. Although the curves recover quickly from these fluctuations as more samples are added, they eventually level off and converge at a weighted F1-score of approximately 83% to 85%. This sustained plateau further supports the conclusion that, under the current hyperparameter configuration, the model has likely reached

its structural capacity limit. As a result, further expansion of the dataset is unlikely to produce substantial performance improvements.

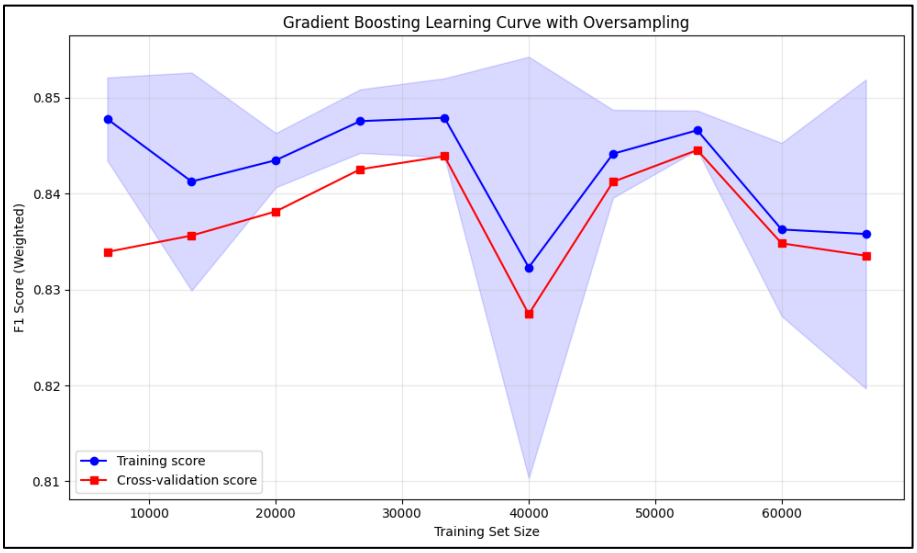

Figure 4.19 Learning Curve of Gradient Boosting with Oversampling Technique

Overall, oversampling improved the classifier's ability to detect minority classes and produced a considerable increase in macro recall and F1-score. However, the associated reduction in precision and the continued confusion between the dominant classes suggest that the technique primarily shifts the model's decision boundaries toward minority-class recognition rather than improving overall classification effectiveness.

4.4.3. Gradient Boosting with SMOTE

The use of SMOTE as an oversampling strategy produced results that were largely similar to those obtained with random oversampling. There were only minor differences across the evaluation metrics. The model achieved an overall accuracy of 84.02%, a macro precision of 66.75%, a macro recall of 90.72%, and a macro F1-score of 70.60%. In comparison with the random oversampling variant, both accuracy and macro precision remained nearly unchanged. The macro recall changed only slightly from 90.96% to 90.72%, and the macro F1-score showed a small decrease from 71.47% to 70.60%. Overall, these findings indicate that SMOTE

did not provide any meaningful improvement over random oversampling for this particular classifier and dataset.

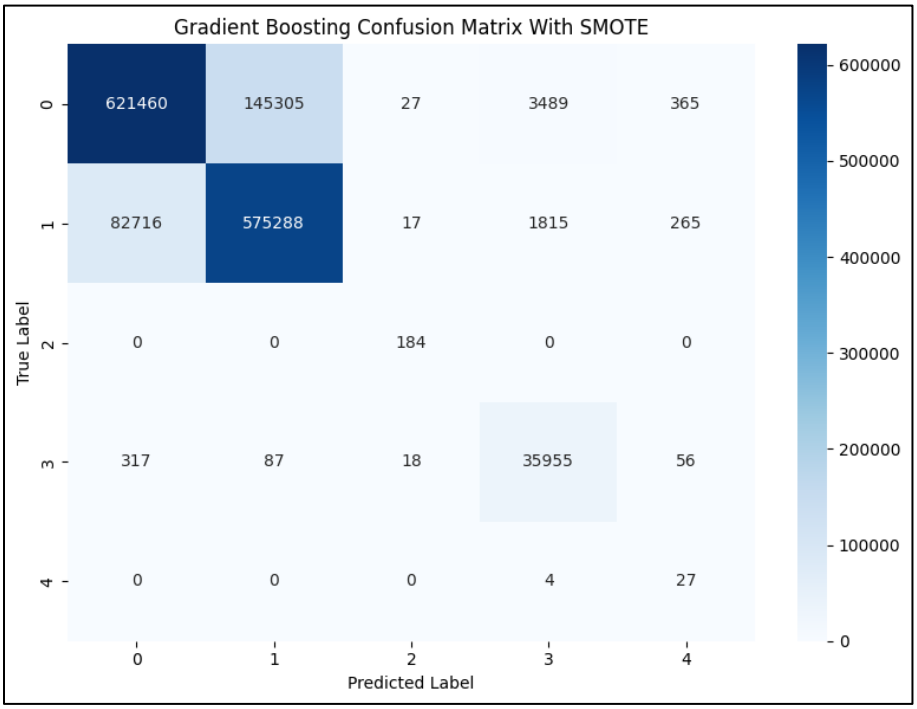

Figure 4.20 Confusion Matrix of Gradient Boosting with SMOTE

The confusion matrix (Figure 4.20) shows a pattern similar to that observed with random oversampling. Label 2 was classified with perfect accuracy, while Labels 3 and 4 also achieved high detection rates, indicating that SMOTE effectively preserved the improvements in minority-class recognition introduced by oversampling. However, the dominant classes again displayed noticeable mutual confusion: many Label 0 instances were misclassified as Label 1, and vice versa. This behavior reflects the same trade-off observed earlier, where improved sensitivity to minority classes comes at the expense of reduced discriminability between the majority classes.

The slight reduction in macro F1-score relative to random oversampling suggests that the synthetic samples generated by SMOTE did not provide additional discriminative signal beyond what simple replication achieved. This may reflect the nature of the feature space in

the dataset, where minority-class instances are sufficiently clustered that interpolation between nearest neighbors does not meaningfully enrich the training distribution.

The transition to SMOTE, as illustrated in the learning curve (Figure 4.21), further challenges the model's stability. Unlike random oversampling, SMOTE introduces sever performance volatility, with weighted F1-scores plummeting as low as 0.70 at certain training intervals. The extreme variance observed, particularly the expansive confidence interval at the 64,000-sample mark, indicates that the synthetic data has severely destabilized the learning process.

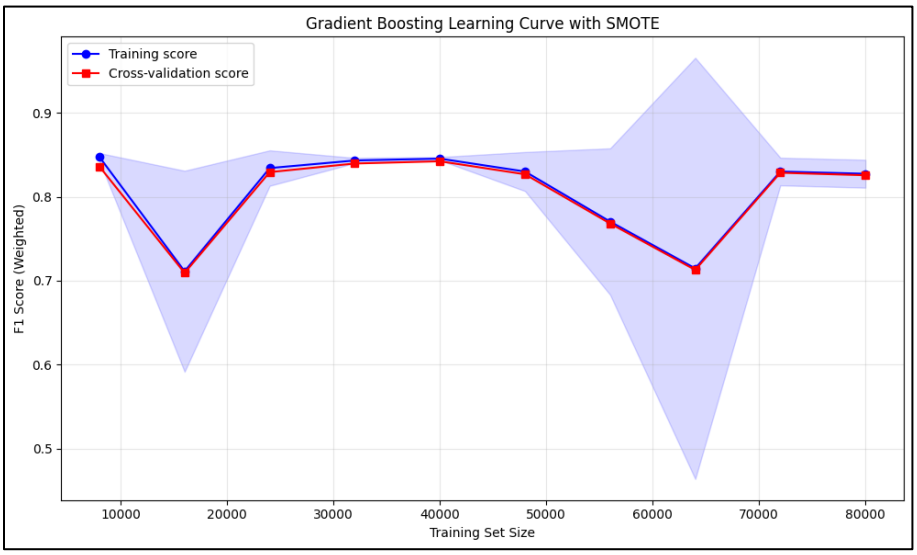

Figure 4.21 Learning Curve of Gradient Boosting with SMOTE

4.4.4. Gradient Boosting with Undersampling

The application of random undersampling resulted in a notable decline in overall classification performance compared to all previously evaluated variants. The model achieved an accuracy of 77.85%, representing a reduction of approximately 6.2 percentage points relative to the baseline Gradient Boosting model. The macro precision stood at 62.92%, macro recall at 87.92%, and macro F1-score at 66.07%. While the macro recall remains relatively high, consistent with the pattern observed across oversampling techniques, the substantially lower accuracy and macro F1-score indicate that the reduction of majority-class samples introduced a considerable amount of misclassification among the dominant classes.

The confusion matrix (Figure 4.22) confirms this interpretation. As with SMOTE and random oversampling, Label 2 achieved perfect classification, and Label 4 maintained a comparable detection rate to previous variants. Label 3 performance remained largely stable, though a slight increase in false negatives is observable. However, the most notable deterioration occurred for Labels 0 and 1, where the number of misclassified majority-class samples increased considerably compared with all oversampling variants. This outcome is consistent with the typical behavior of undersampling. By reducing the number of majority-class training instances, the classifier loses important discriminative information needed to distinguish between the two dominant categories, which ultimately leads to increased confusion between them.

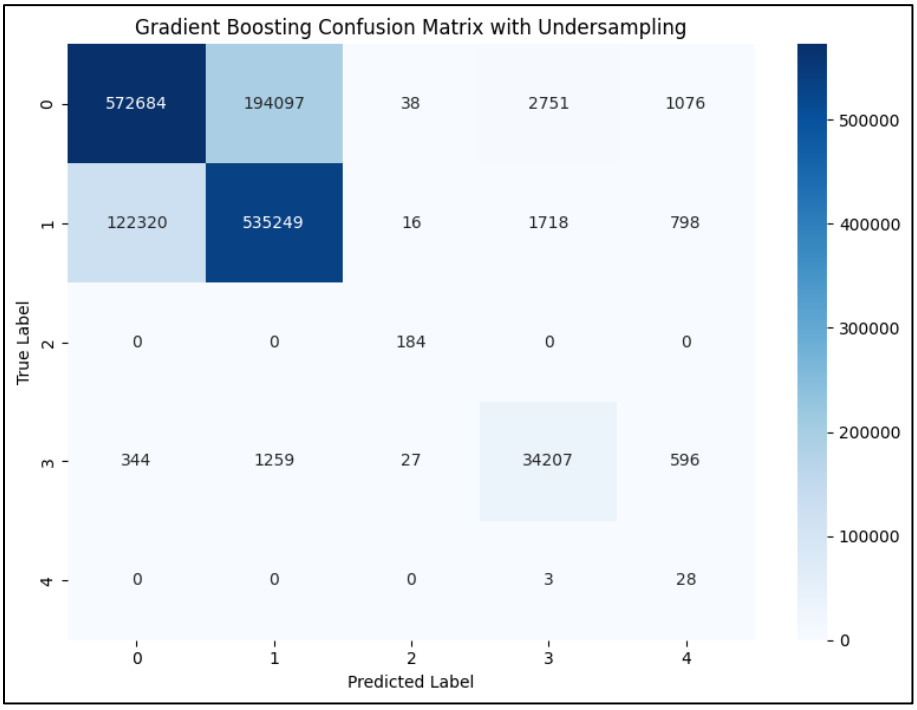

Figure 4.22 Confusion Matrix of Gradient Boosting with Undersampling

The learning curve presented in Figure 4.23 reveals a pronounced overfitting pattern, distinctly different from all previously evaluated variants. The training score remains at or near 1.00 across the entire range of training set sizes, while the cross-validation score begins at approximately 0.68 and gradually improves, stabilizing at around 0.89. The substantial and

persistent gap between the two curves is a direct consequence of the severely reduced training set produced by undersampling: with only a few hundred samples available after downsampling, the model memorizes the training data almost perfectly while failing to generalize equivalently to the original imbalanced distribution.

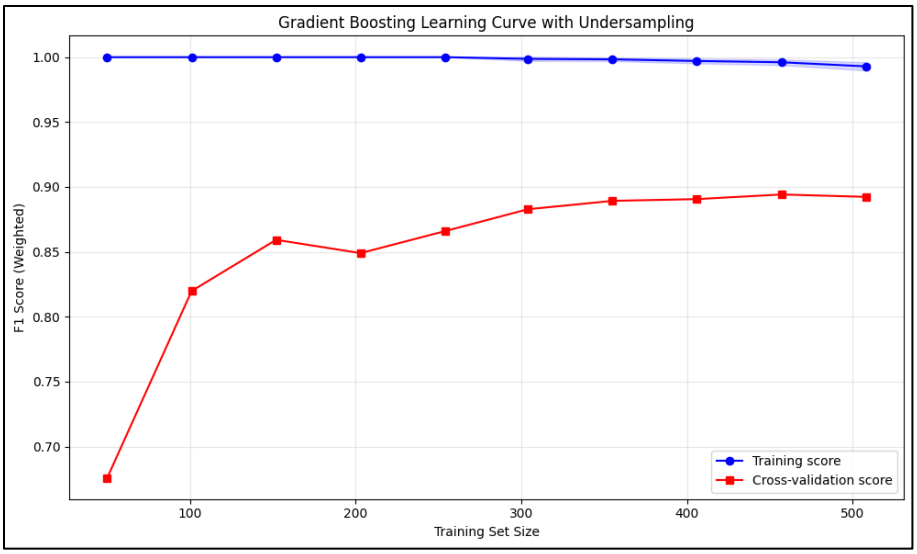

Figure 4.23 Learning Curve of Gradient Boosting with Undersampling

Overall, undersampling produced the weakest classification performance among all Gradient Boosting variants evaluated thus far. The results indicate that discarding majority-class samples is a less effective strategy than augmenting minority-class representation for this dataset and classifier combination, as the information lost through undersampling cannot be compensated for by the improved class balance during training.

4.4.5. Gradient Boosting with Class Weighting

The application of class weighting produced results closely comparable to those obtained with SMOTE and random oversampling. The model achieved an accuracy of 84.00%, a macro precision of 66.83%, a macro recall of 90.90%, and a macro F1-score of 70.64%. These figures differ from the SMOTE variant by less than one percentage point across all metrics, suggesting that rebalancing the loss function through class weights achieves a similar redistribution of the

model's decision boundaries as synthetic sample generation, without modifying the training data itself.

The confusion matrix (Figure 4.24) reflects the familiar pattern: Label 2 achieved perfect classification, and Label 3 delivered strong performance, benefiting here from a reduced false negative count compared to previous variants but less effective than what we achieved with Oversampling. Label 4 again suffered from small precision due to the small number of true instances relative to false positives. The dominant classes showed the same mutual confusion observed under oversampling, with a substantial number of Label 0 and Label 1 samples misclassified across the boundary.

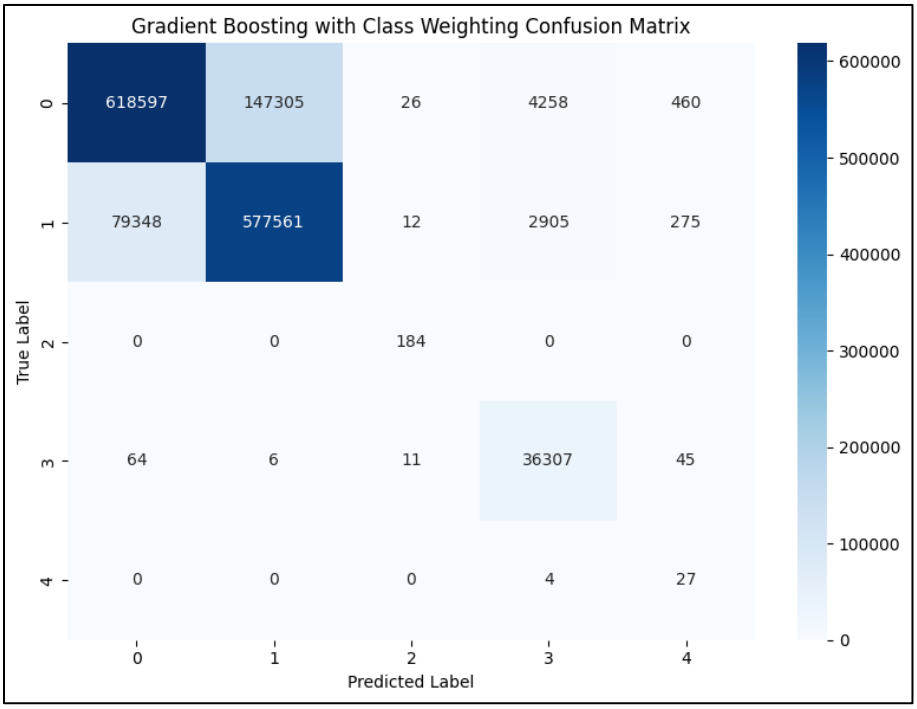

Figure 4.24 Confusion Matrix of Gradient Boosting with Class weighting

Learning curve in Figure 4.25 reveals a highly distinct and revealing behavior. For the majority of training set sizes, cost-sensitive boosting exhibits remarkable stability, closely aligning the training and cross-validation curves at the asymptotic limit of ~0.84. This indicates that scaling the loss function gradients prevents the generalization gap seen in oversampling techniques. However, a catastrophic stochastic collapse is observed around the 46,000-sample interval. At

this point, the weighted cross-validation score drops sharply to approximately 0.57, accompanied by extremely high variance (ranging from about 0.17 to nearly 0.97 across different folds).

Under specific data partitioning at 46,000 samples, the high weight multipliers applied to the minority classes likely induced localized gradient explosion and step-size overshooting. This caused the sequential boosting optimization to diverge heavily in certain folds, reducing performance to near-random guessing, while managing to converge normally in others.

As the training size increases beyond 54,000, this instability is mitigated. The larger volume of majority instances acts as a regularizing force, diluting the destabilizing impact of high-gradient minority samples. Nevertheless, the post-recovery convergence back to the strict 0.84 plateau reinforces that adjusting sample weights merely manipulates the optimization path without expanding the model's fundamental representation capacity.

Overall, class weighting achieved competitive performance relative to the oversampling techniques while avoiding the additional computational cost and data modification they entail. The results further reinforce the observation that for this Gradient Boosting configuration, the method of class rebalancing has limited influence on final performance, with all techniques converging toward a similar accuracy and macro F1-score ceiling.

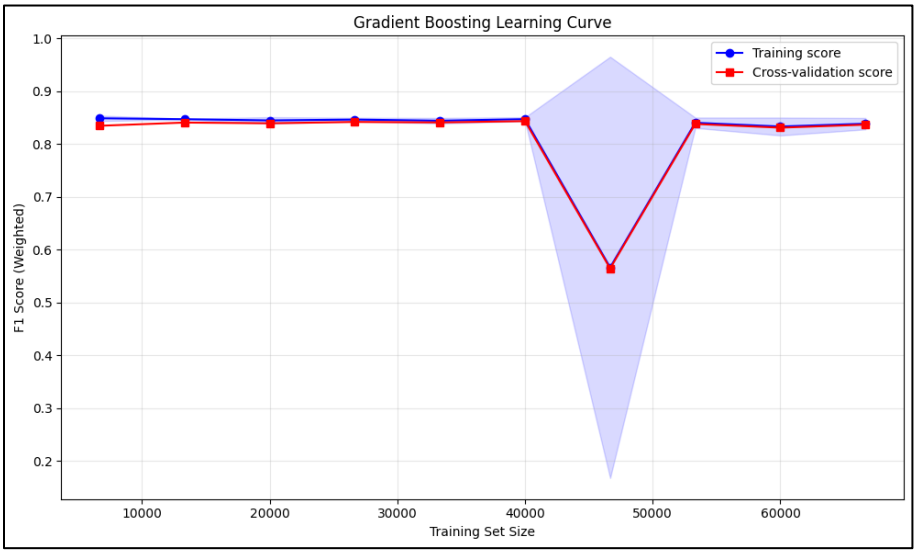

Figure 4.25 Learning Curve of Gradient Boosting with Class weighting

Based on the results of all five Gradient Boosting variants summarized in Figure 4.26 to Figure 4.30, Random Oversampling offers the most favorable overall balance between classification performance and computational cost. Although SMOTE, Oversampling, and Class Weighting achieved very similar accuracy (≈84%) and macro recall (≈91%), Random Oversampling produced the highest macro F1-score (71.47%), indicating the best balance between precision and recall across all classes. It also showed more stable learning behavior than SMOTE, whose learning curve displayed noticeable fluctuations and high variance, suggesting lower reliability.

While both Oversampling and SMOTE introduced considerable training-time overhead due to dataset expansion, SMOTE additionally required extra computational effort for synthetic sample generation without delivering meaningful performance gains.

Class Weighting emerged as the most computationally efficient imbalance-handling method, as it required no modification of the dataset and achieved performance nearly comparable to Oversampling and SMOTE. However, its slightly lower macro F1-score and the instability observed at certain training sizes suggest that the improvement mainly resulted from optimization adjustments rather than improved class representation. In contrast, undersampling produced the weakest results, leading to a noticeable drop in accuracy and clear signs of overfitting due to the loss of important majority-class information. Therefore, if predictive performance is the main objective, Random Oversampling is the preferred option. However, when computational efficiency is also an important consideration, Class Weighting offers the most practical alternative, providing near-equivalent classification performance with significantly lower training cost.

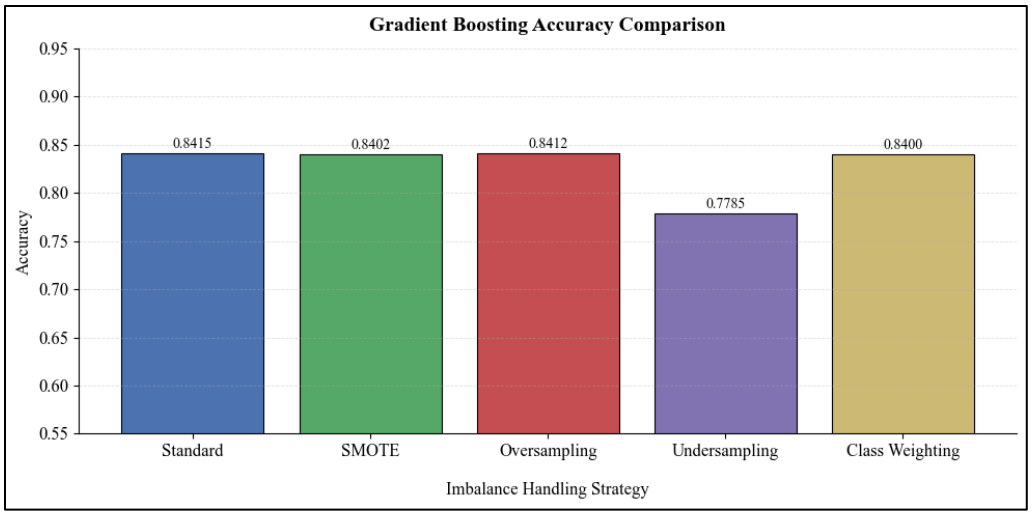

Figure 4.26 Gradient Boosting Accuracy Comparison

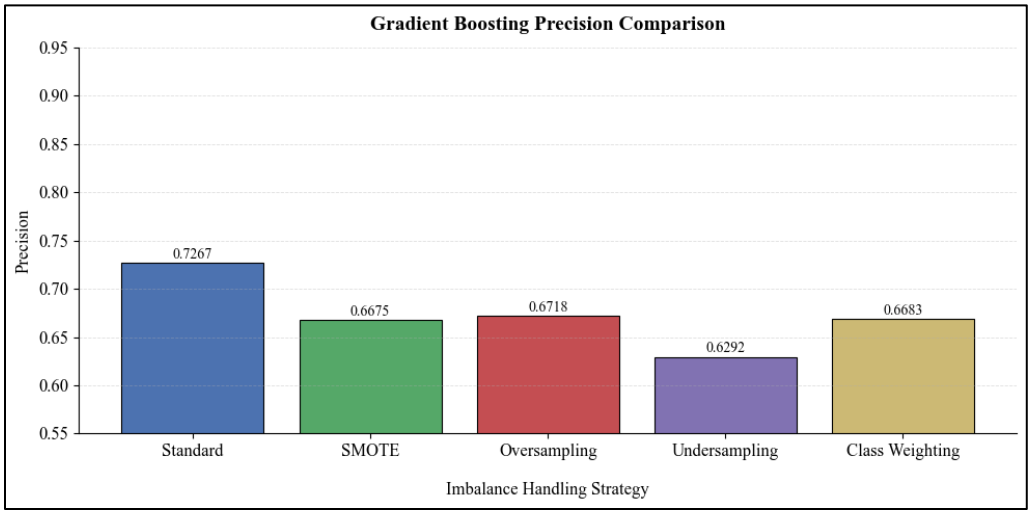

Figure 4.27 Gradient Boosting Precision Comparison

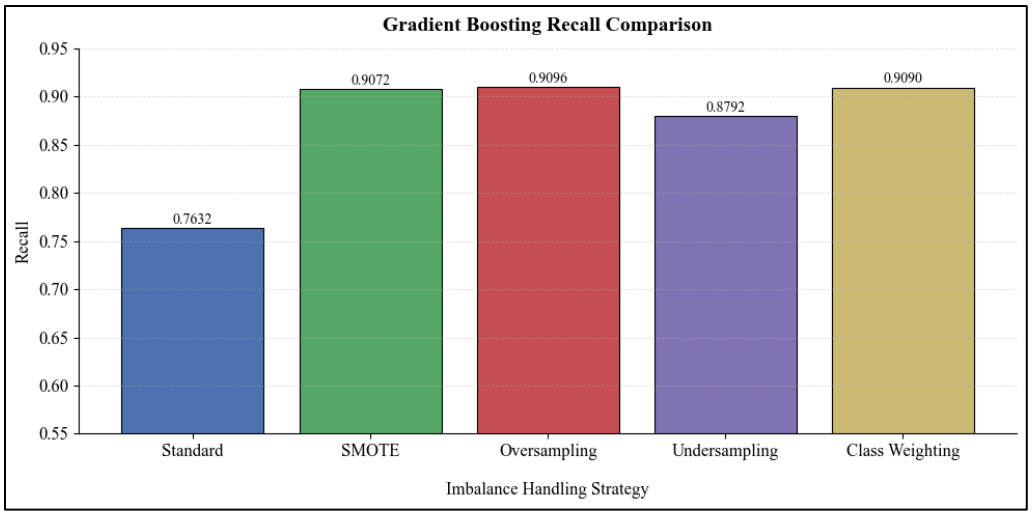

Figure 4.28 Gradient Boosting Recall Comparison

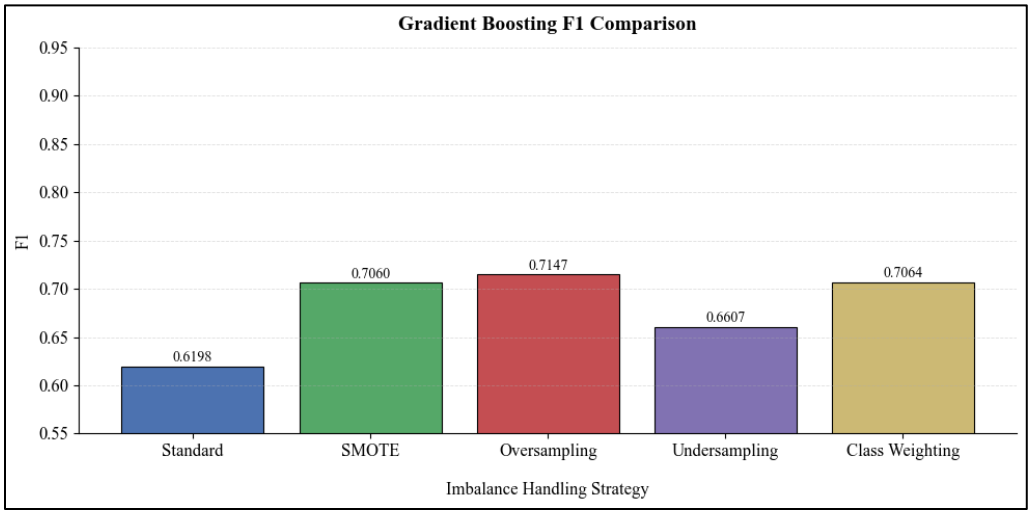

Figure 4.29 Gradient Boosting F1 Score Comparison

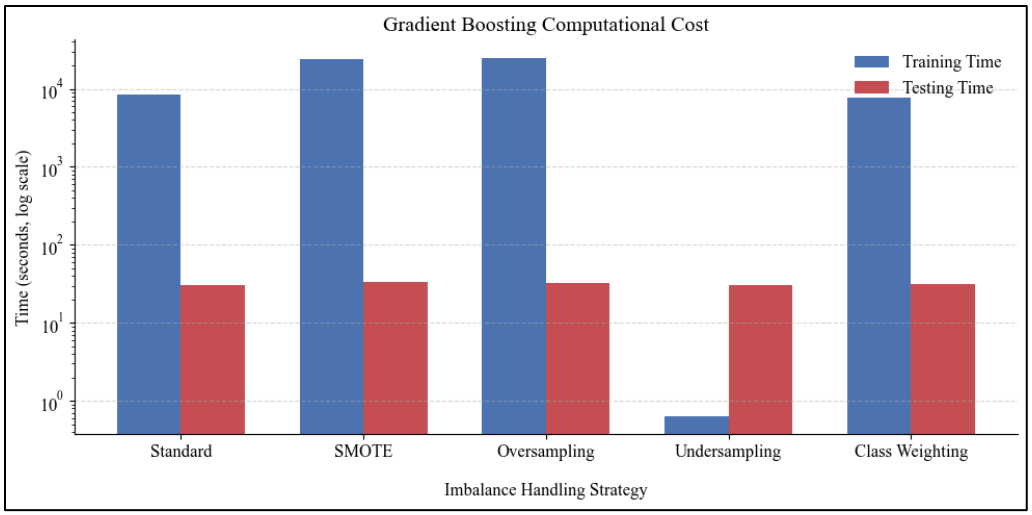

Figure 4.30 Gradient Boosting Training Time and Test Time Comparison

4.5. Artificial Neural Network Results

The optimized hyperparameters configuration for five Artificial Neural Network classifiers used in this study is presented in Table 4.3.

Table 4.3 Optimized Hyperparameters of Artificial Neural Network

Hyperparameter	Value
Hidden Layers	(128, 64)
Activation Function	tanh
Learning Rate	0.001
Regularization Parameter (α)	0.0001
Maximum Iterations	1000
Random State	42

4.5.1. Standard Artificial Neural Network

The baseline Artificial Neural Network model achieved an overall accuracy of 66.40%. The macro-averaged precision, recall, and F1-score values of 69.44%, 66.40%, and 67.34%, respectively. Compared with the Random Forest and Gradient Boosting, the ANN showed

noticeably lower classification performance. Although the precision and recall values remain relatively balanced, the modest F1-score indicates that the network had difficulty maintaining consistent predictive performance across all attack categories. These results suggest that the selected ANN architecture was less effective at capturing the complex decision boundaries present in the BoT-IoT dataset, particularly under the highly imbalanced class distribution.

The confusion matrix shown in Figure 4.31 provides further insight into the model's behavior. The ANN successfully identified the rarest class (Label 4) and achieved near-perfect recognition of Label 2. However, a considerable degree of confusion can be observed between the two dominant classes, Label 0 and Label 1. A large number of Label 0 samples were misclassified as Label 1, while many Label 1 instances were also predicted as Label 0. This mutual misclassification played an important role in reducing the overall accuracy of the model. Furthermore, although Label 3 was detected relatively well, a noticeable number of its instances were incorrectly assigned to neighboring classes, further reducing balanced classification performance.

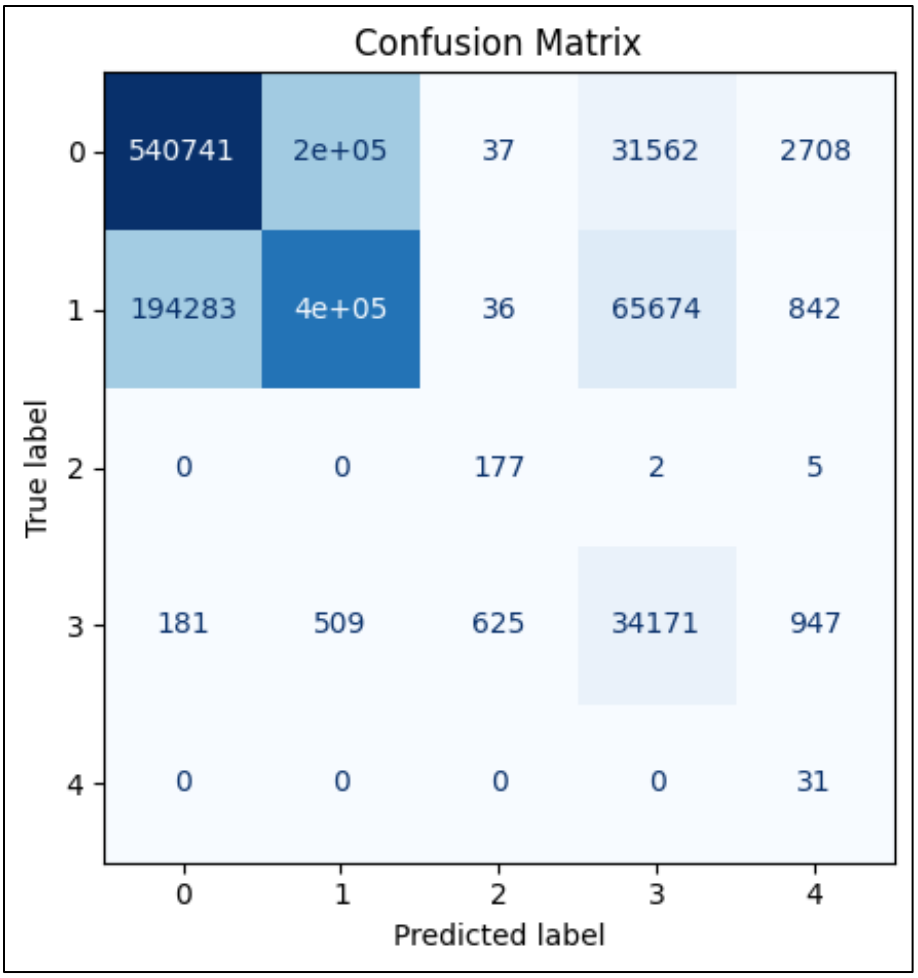

Figure 4.31 Confusion Matrix of ANN

Figure 4.32 illustrates the training loss curve of the ANN. The loss decreases during the initial epochs, dropping from about 1.8 to below 0.5 within the first few iterations, before converging to a stable value near 0.13. The smooth and continuous decline in loss indicates that the optimization process remained stable and that the network successfully learned feature representations from the training data. Furthermore, the absence of significant oscillations suggests that the chosen learning rate provided effective convergence behavior. However, despite the steady reduction in training loss, the relatively modest classification metrics indicate that improved optimization alone was insufficient to overcome the challenges associated with the dataset's class imbalance and complex class boundaries.

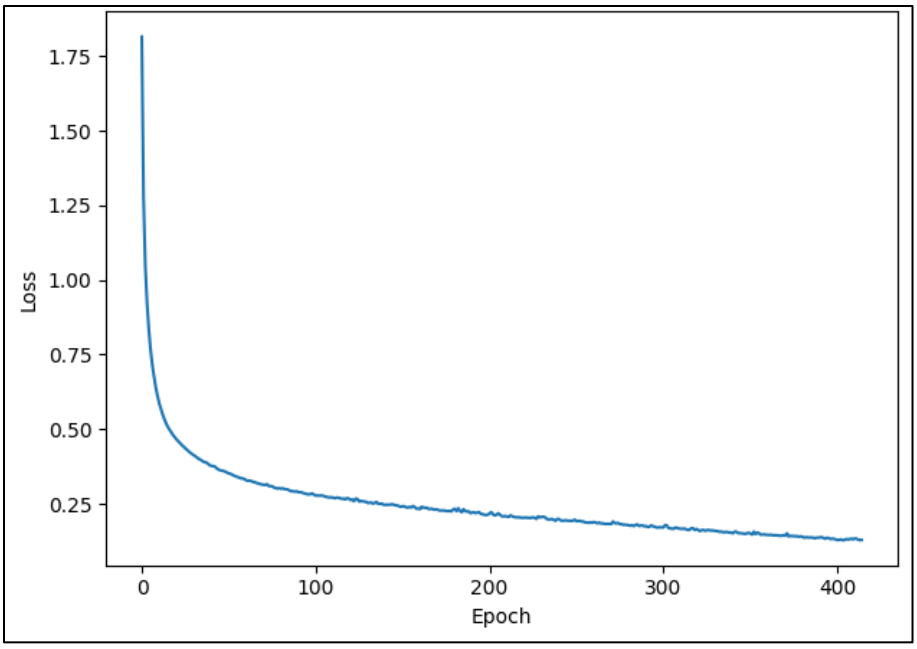

Figure 4.32 Training Loss Curve of ANN

Overall, the baseline ANN achieved moderate classification performance and demonstrated stable convergence characteristics. However, its predictive performance remained lower than that of the ensemble-based classifiers evaluated later. This suggests that additional imbalance-handling techniques may be necessary to improve the recognition of minority classes and the overall effectiveness of the model.

4.5.2. Artificial Neural Network with Oversampling Technique

The application of random oversampling resulted in a substantial improvement in the performance of the Artificial Neural Network classifier. The model achieved an overall accuracy of 92.22%, with macro-averaged precision, recall, and F1-score all reaching approximately 92.22%. Compared with the baseline ANN the improvement is remarkable. These results indicate that increasing the representation of minority-class samples enabled the neural network to learn more balanced decision boundaries and significantly enhanced its classification capability across all attack categories.

The confusion matrix presented in Figure 4.33 further demonstrates the effectiveness of the oversampling strategy. The model achieved strong classification performance across all classes, with only a limited number of misclassifications observed. Labels 2 and 4 were

identified almost perfectly, while Label 3 also achieved a high detection rate with relatively few errors. Although some confusion remained between the two dominant classes (Labels 0 and 1), the number of incorrect predictions was lower than what is typically observed in imbalanced learning scenarios. These results indicate that the oversampling strategy helped the network learn a more balanced representation of the class distribution without compromising its ability to recognize the majority classes.

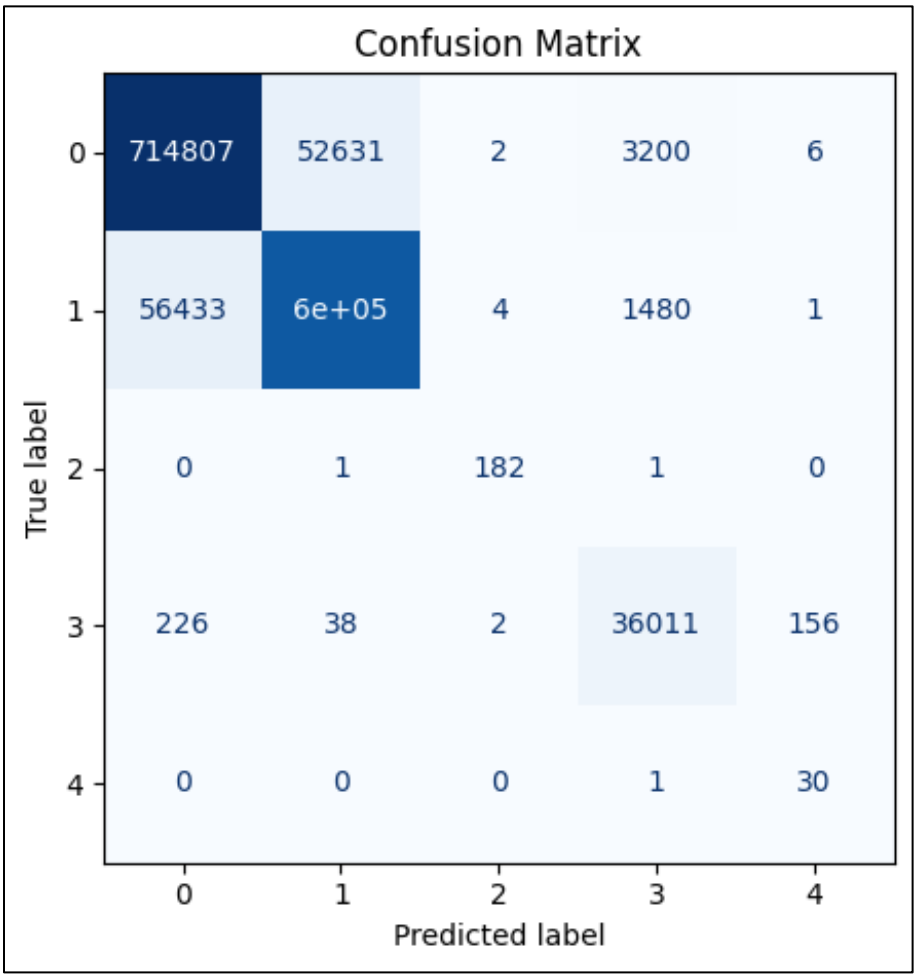

Figure 4.33 Confusion Matrix of ANN with Oversampling

The training loss curve shown in Figure 4.34 illustrates stable and effective convergence throughout the learning process. The loss decreases smoothly from about 0.156 at the beginning of training to nearly 0.05 by the final epoch, without noticeable oscillations or abrupt

fluctuations. This steady decline indicates that the optimization process remained stable and that the network continued to extract useful patterns as training progressed. Furthermore, the absence of sudden increases in loss suggests that the selected architecture and learning parameters provided a well-behaved training process with no apparent convergence difficulties.

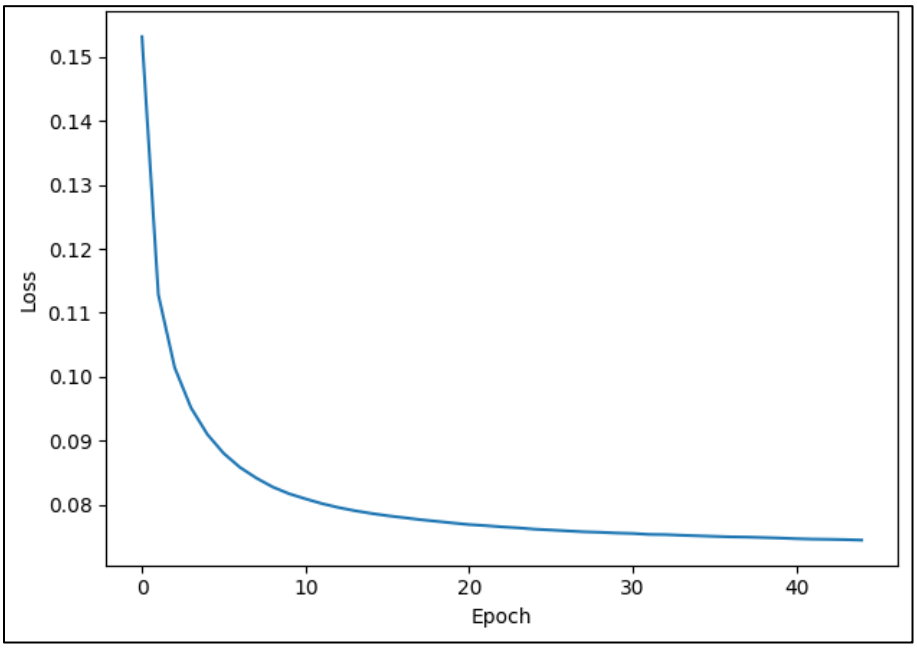

Figure 4.34 Training Loss Curve of ANN with Oversampling

Overall, random oversampling proved to be effective for the ANN classifier, producing strong performance across all evaluation metrics while maintaining stable training behavior. The results suggest that increasing the representation of minority-class samples allowed the network to learn more balanced decision boundaries and improved its ability to distinguish between different attack categories.

4.5.3. Artificial Neural Network with SMOTE Technique

The application of SMOTE produced the strongest overall performance among the ANN variants evaluated. The model achieved an accuracy of 92.38%, while macro-averaged precision, recall, and F1-score each reached 92.38%. The identical values across all evaluation metrics indicate a balanced classification performance, suggesting that the classifier

maintained an effective trade-off between false positive and false negative predictions across the different attack categories. These results demonstrate that the synthetic samples generated by SMOTE provided the network with a more representative training distribution, enabling it to learn more robust decision boundaries.

The confusion matrix presented in Figure 4.35 further highlights the effectiveness of this approach. Labels 2 and 4 were classified almost perfectly, with only a single misclassification observed for Label 2 and no misclassifications for Label 4. Label 3 also achieved a high detection rate, with few samples assigned to other categories. Similar to the oversampling variant, the majority of classification errors occurred between Labels 0 and 1, which remained the most challenging classes to distinguish. However, the number of classified instances in both categories exceeded the number of misclassifications, indicating strong discrimination capability for the dominant traffic classes. Overall, the confusion matrix confirms that SMOTE enabled the ANN to achieve balanced performance across both majority and minority classes.

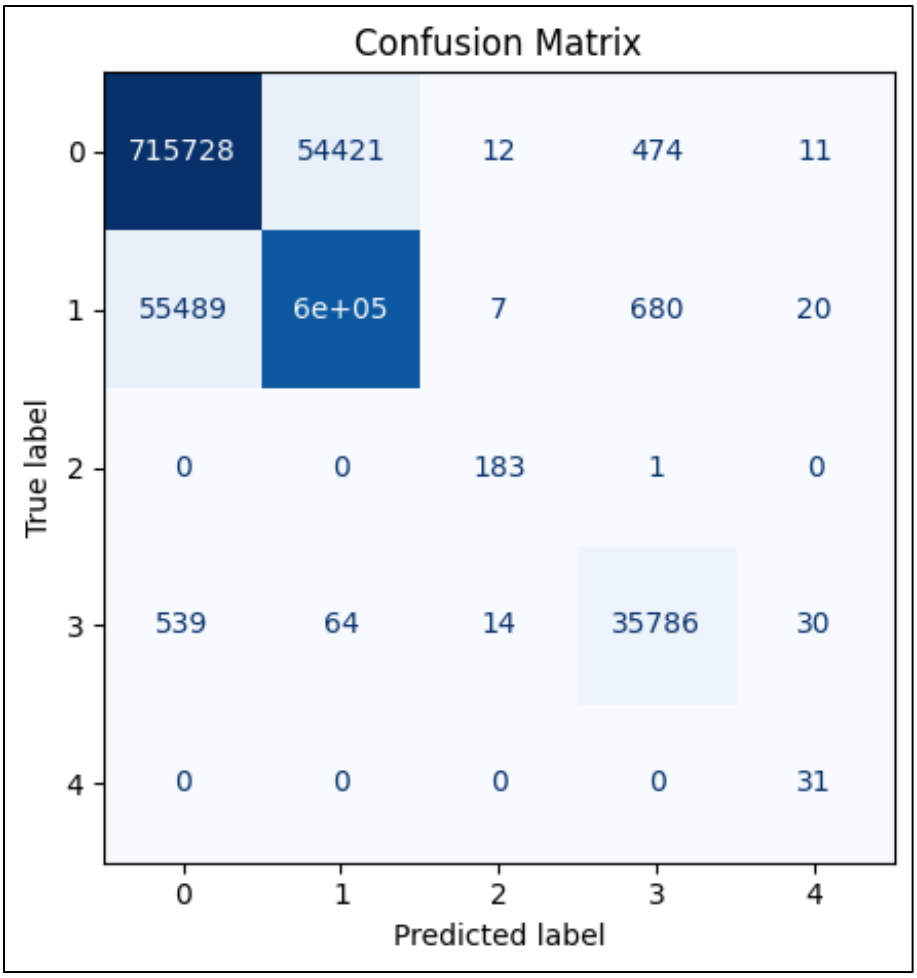

Figure 4.35 Confusion Matrix of ANN with SMOTE

The training loss curve in Figure 4.36 shows stable convergence throughout the optimization process. The loss gradually declines from about 0.146 at the start of training to nearly 0.07 in the final epoch, with no major fluctuations. This steady decrease suggests that the network gradually improved its internal representation of the data during training and that the optimization process remained stable. The smooth convergence pattern also indicates that the synthetic samples generated by SMOTE did not introduce instability into the learning process.

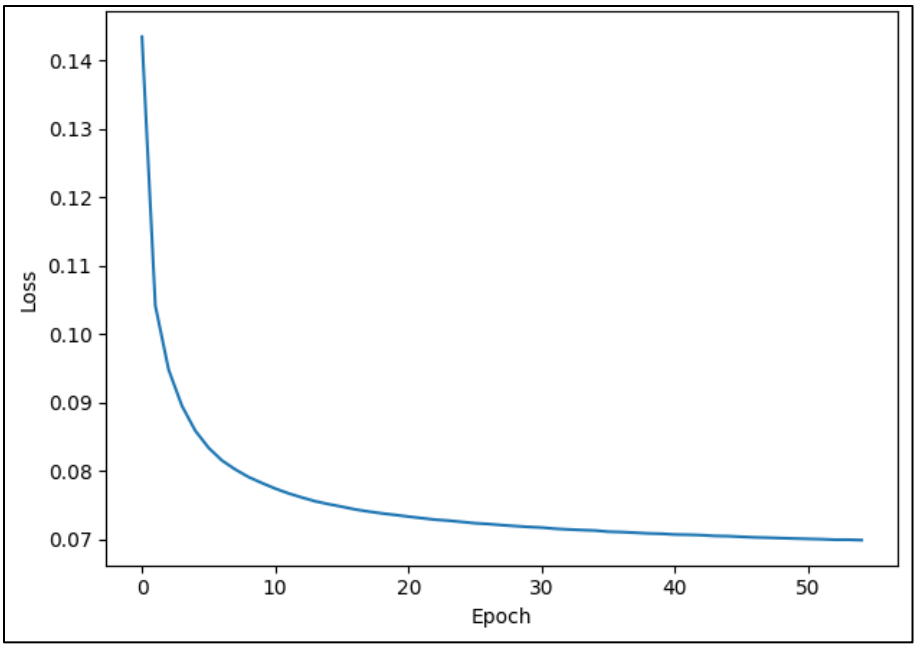

Figure 4.36 Training Loss Curve of ANN with SMOTE

All in all, SMOTE proved to be a highly effective imbalance-handling strategy for the ANN classifier. The technique achieved excellent classification performance while maintaining stable training behavior and strong recognition capability across all classes. These findings indicate that generating synthetic minority-class samples can significantly enhance the learning capacity of neural networks when dealing with highly imbalanced intrusion detection datasets.

4.5.4. Artificial Neural Network with Undersampling Technique

The application of random undersampling resulted in a significant reduction in overall classification performance. The model achieved an accuracy of 66.40%, with macro-averaged precision, recall, and F1-score values of 69.44%, 66.40%, and 67.34%, respectively. These results indicate that balancing the class distribution through the removal of majority-class samples was considerably less effective than the oversampling-based approaches. Although the network was exposed to a more balanced training set, the loss of a substantial portion of the majority-class information negatively affected its ability to learn representative decision boundaries.

The confusion matrix presented in Figure 4.37 illustrates this behavior clearly. The model maintained excellent detection of the minority classes, correctly classifying almost all

instances of Labels 2 and 4. Label 3 also achieved a relatively high recognition rate, with the majority of its samples assigned to the correct class. However, the dominant classes experienced severe degradation in performance. A large number of Label 0 samples were incorrectly classified as Label 1, while a similarly substantial number of Label 1 instances were misclassified as Label 0. Additional confusion was also observed between the dominant classes and Label 3. Consequently, the gains in minority-class detection were offset by the considerable increase in majority-class classification errors, leading to a substantial reduction in overall predictive performance.

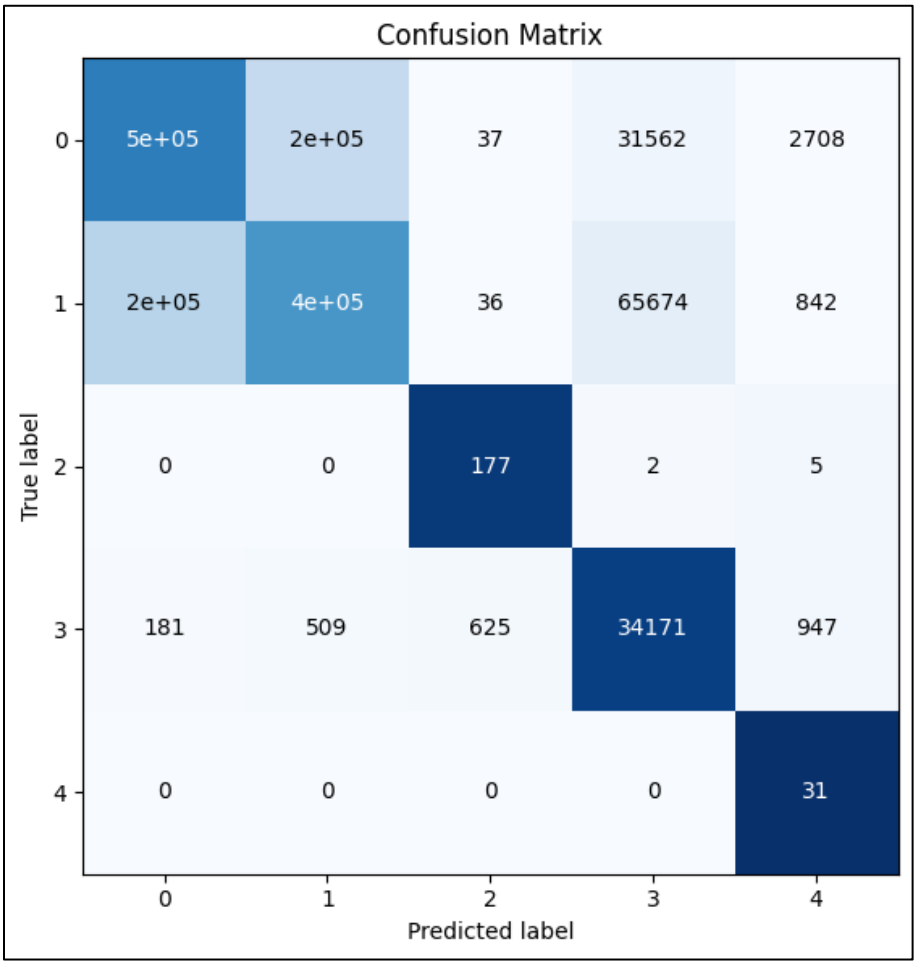

Figure 4.37 Confusion Matrix of ANN with Undersampling

The training loss curve shown in Figure 4.38 demonstrates stable convergence throughout the optimization process. The loss decreases steadily from approximately 1.85 at the beginning of

training to nearly 0.10 by the final epoch, indicating that the network successfully learned the patterns present within the reduced training set. The smooth decline in loss suggests that the optimization process remained stable and free from convergence difficulties. However, the significant drop in training loss did not lead to similar performance on the test set, suggesting that the reduced dataset did not contain enough information for the network to generalize properly to the original class distribution.

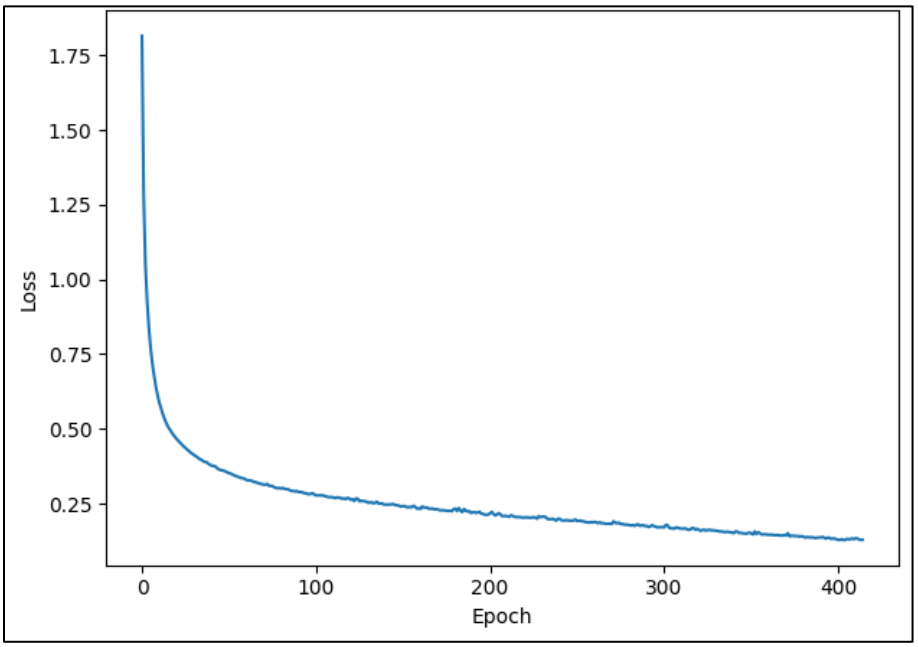

Figure 4.38 Training Loss Curve of ANN with Undersampling

Altogether, random undersampling proved to be one of the least effective imbalance-handling strategies for the ANN classifier. While the technique improved the representation of minority classes during training and preserved strong detection rates for the rare attack categories, the substantial loss of majority-class information resulted in extensive confusion among the dominant classes. These findings suggest that the disadvantages associated with information loss outweigh the benefits of class balancing, making undersampling a less suitable approach for ANN-based intrusion detection on the BoT-IoT dataset.

4.5.5. Artificial Neural Network with Class Weighting Technique

Applying class weighting produced the best overall results among the evaluated ANN variants. The model reached an accuracy of 92.64%, with macro-averaged precision, recall, and F1-score of 92.66%, 92.64%, and 92.65%. The very close values of these metrics show a balanced classification performance across all attack classes. These results indicate that modifying the loss function to assign higher penalties to errors on minority classes helped the network improve class discrimination without changing the original training dataset.

The confusion matrix presented in Figure 4.39 further confirms the effectiveness of this approach. Labels 2 and 4 were classified almost perfectly, with all instances of Label 2 correctly identified and only a single misclassification observed for Label 4. Label 3 also achieved a high detection rate, with relatively few errors distributed among the remaining classes. Although some confusion persisted between the dominant classes, Labels 0 and 1, the number of misclassifications was lower than that observed in the undersampling approach and comparable to the oversampling-based methods. The classifier therefore maintained strong recognition capability across both majority and minority classes while preserving a balanced distribution of prediction errors.

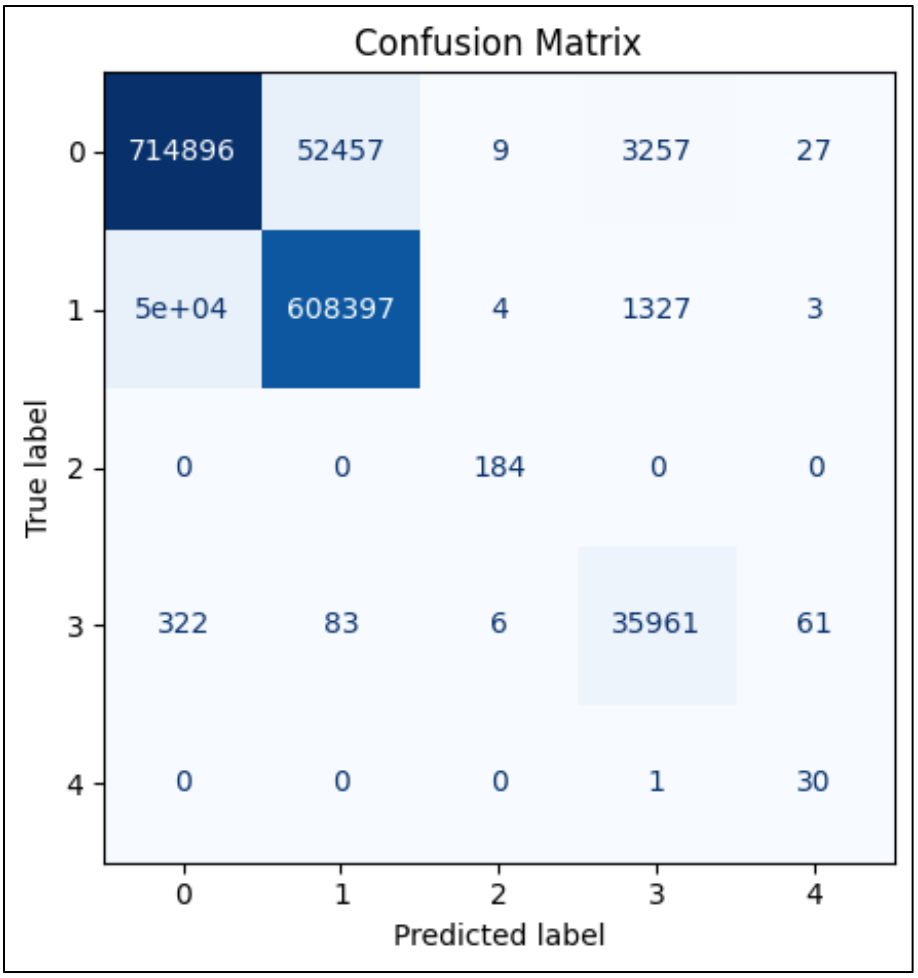

Figure 4.39 Confusion Matrix of ANN with Class Weighting Technique

The training loss curve in Figure 4.40 shows stable and steady convergence during the training process. The loss decreases smoothly from about 0.30 at the start of training to around 0.125 at the final epoch, without large oscillations or sudden changes. This pattern shows that the optimization process remained stable even after adding class-dependent penalties to the loss function. The gradual reduction in loss further suggests that the network effectively adapted its decision boundaries to account for minority-class samples while maintaining good overall generalization.

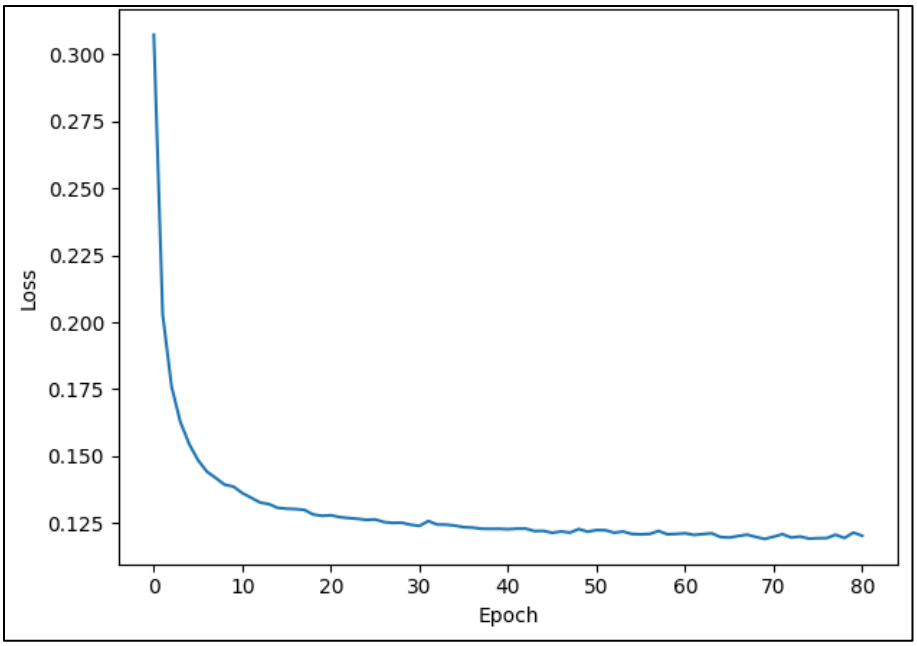

Figure 4.40 Training Loss Curve of ANN with Class Weighting Technique

Overall, class weighting proved to be a highly effective imbalance-handling strategy for the ANN classifier (Figure 4.41 to Figure 4.45). The technique achieved excellent classification performance while avoiding the additional computational cost associated with generating or duplicating training samples. By modifying the learning process directly rather than altering the dataset, class weighting successfully improved the network's ability to recognize minority classes while maintaining strong performance on the dominant categories. These findings suggest that class weighting offers an effective and computationally efficient solution for handling class imbalance in ANN-based intrusion detection systems.

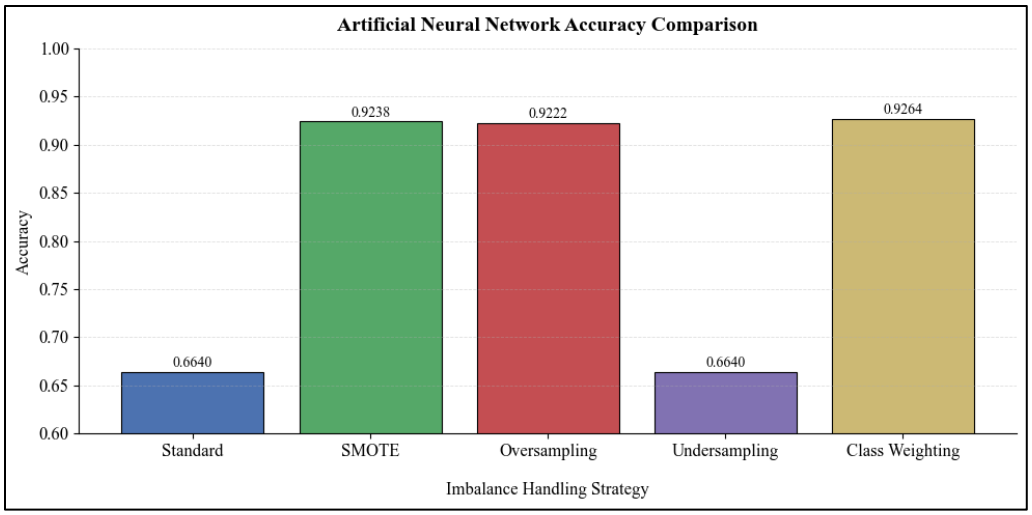

Figure 4.41 ANN Accuracy Comparison

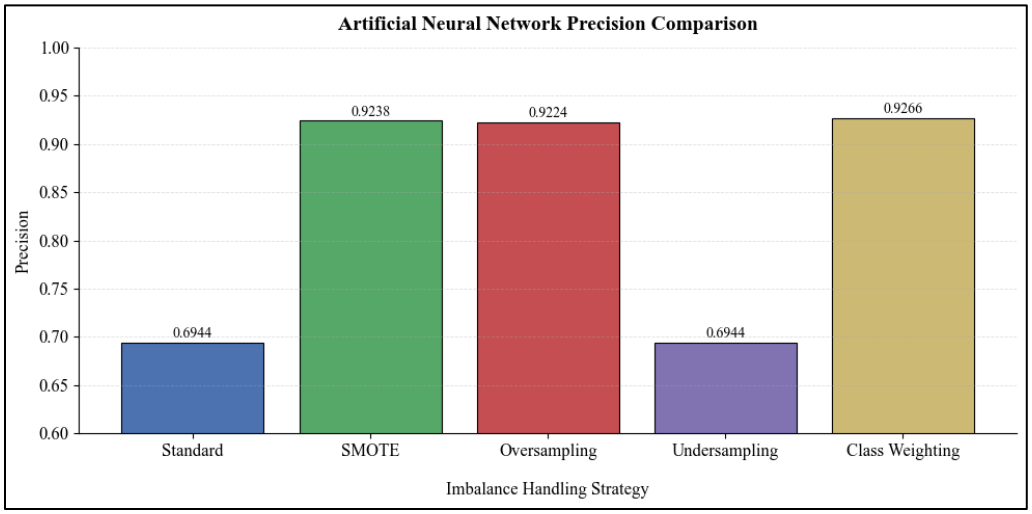

Figure 4.42 ANN Precision Comparison

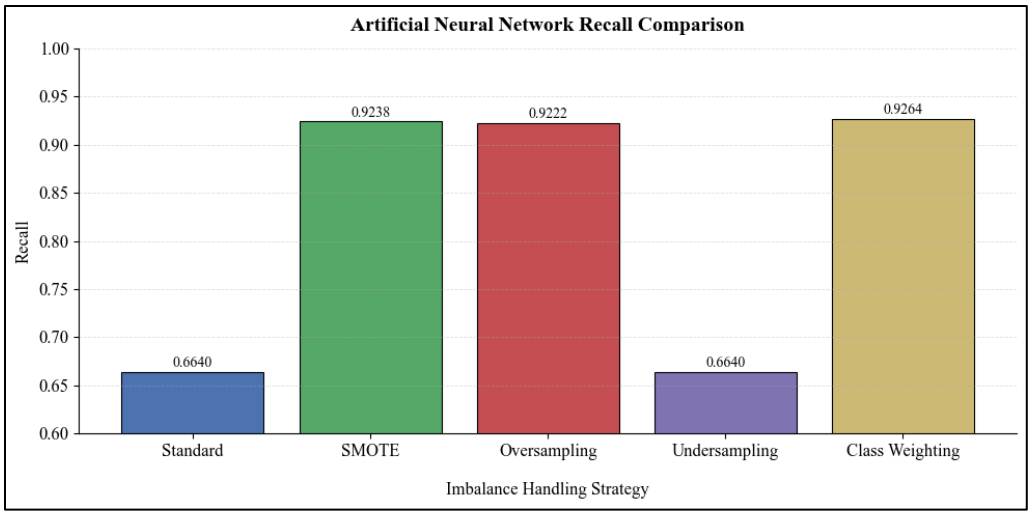

Figure 4.43 ANN Recall Comparison

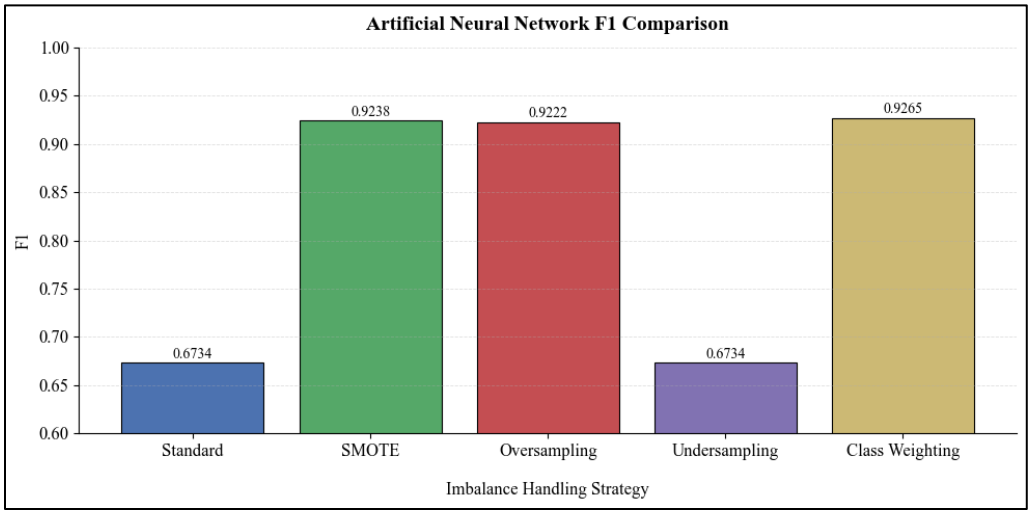

Figure 4.44 ANN F1 Score Comparison

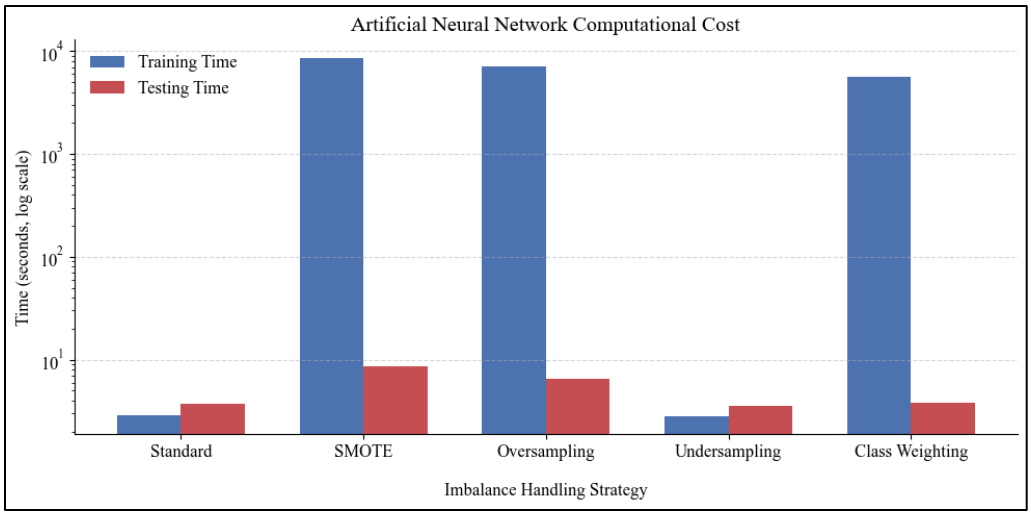

Figure 4.45 ANN Training Time and Test Time Comparison

4.6. Logistic Regression Results

Logistic Regression was selected as the final classifier for evaluation due to its simplicity, interpretability, and widespread use as a baseline model for classification tasks. The optimized hyperparameter configuration used throughout the experiments is presented in Table 4.4.

Table 4.4 Optimized Hyperparameters of Logistic Regression Classifier

Hyperparameter	Value
Solver	lbfgs
Maximum Iterations (max_iter)	1000

4.6.1. Standard Logistic Regression

The baseline Logistic Regression model achieved an overall accuracy of 66.57%, with macro-averaged precision, recall, and F1-score of 66.51%, 66.57%, and 66.44%. Compared with the ensemble classifiers evaluated earlier, the predictive performance of Logistic Regression was noticeably lower. The very similar values of precision, recall, and F1-score show that the classifier produced a balanced distribution of errors. However, the overall results

indicate that the linear decision boundaries of Logistic Regression could not fully capture the complex relationships in the BoT-IoT dataset.

The confusion matrix in Figure 4.46 gives a clearer view of the model's classification behavior. The main source of error comes from strong confusion between Label 0 and Label 1. Many samples from each of these dominant classes were predicted as the other class, showing that the model had difficulty separating their feature distributions. In contrast, Label 3 achieved a relatively high recognition rate, with most of its samples classified correctly. The performance on the minority classes was considerably weaker. Although a reasonable number of Label 2 samples were detected correctly, several instances were misclassified into neighboring classes. Most notably, the classifier completely failed to identify Label 4, with none of its test samples being correctly recognized. This inability to detect the rarest class significantly contributed to the limited overall effectiveness of the baseline Logistic Regression model.

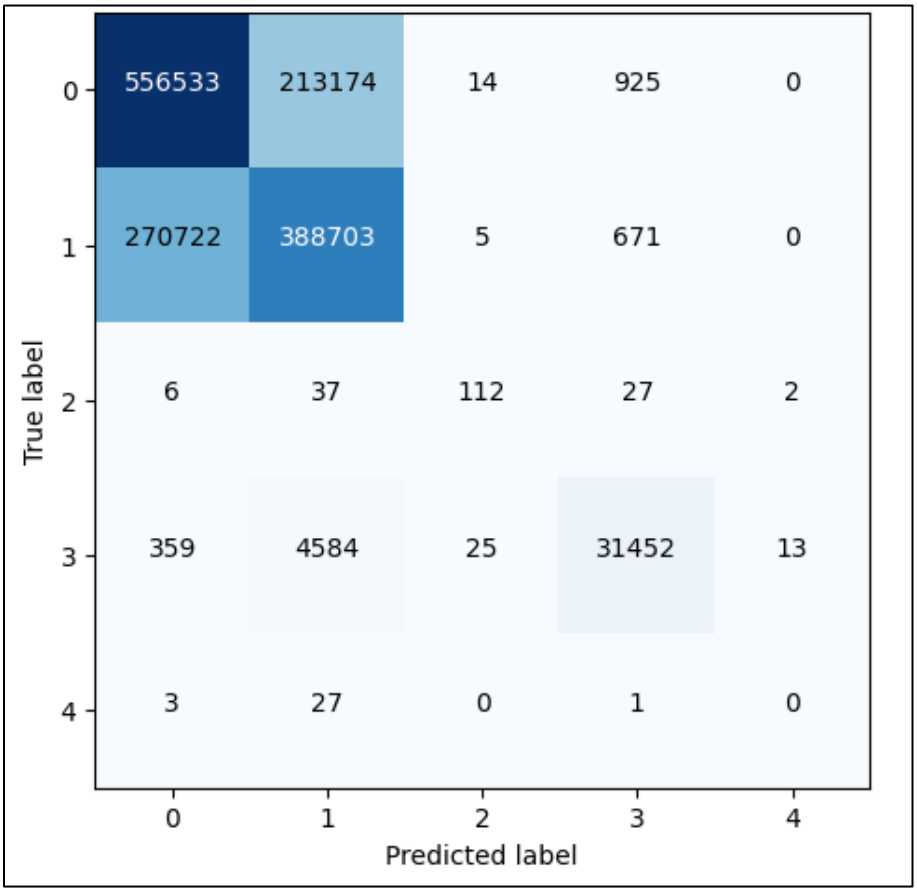

Figure 4.46 Confusion Matrix of Standard Logistic Regression

Figure 4.47 shows that the training and validation scores follow very similar trajectories and converge early, leaving only a small generalization gap. This small gap indicates a low-variance condition, which means the model generalizes well to unseen data and does not exhibit overfitting. However, both curves reach a plateau, and a slight decline appears after the training size passes 2.0×10^6. This pattern indicates a high-bias (underfitting) regime. Since increasing the amount of training data does not lead to noticeable performance improvement, the model appears to have reached the limit of its current representational capacity.

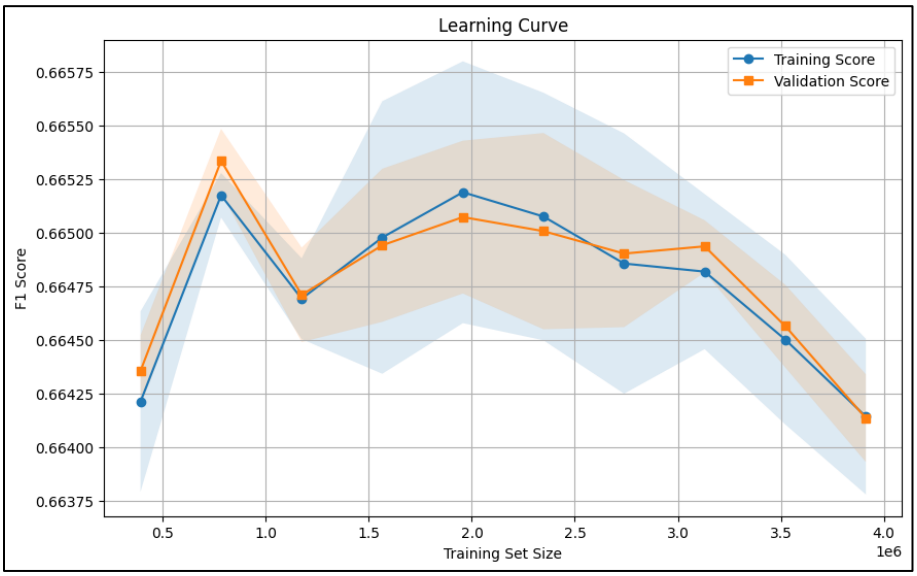

Figure 4.47 Learning Curve of Standard Logistic Regression

4.6.2. Logistic Regression with Oversampling Technique

Random oversampling had a mixed effect on the performance of the Logistic Regression classifier. The model reached an overall accuracy of 63.45%, with macro-averaged precision, recall, and F1-score of 67.15%, 63.45%, and 64.62%. Compared with the baseline model, accuracy and recall decreased, while precision increased. These results show that duplicating minority-class samples did not improve the overall predictive performance of the classifier. Unlike the ANN model, which benefited from stronger minority-class representation, Logistic

Regression remains limited by its linear decision boundaries and cannot fully use the additional training samples created through oversampling.

The confusion matrix in Figure 4.48 shows several changes in the classification results. The model detected Label 4 perfectly, with all samples of the rarest class classified correctly. Label 2 was also recognized with very high accuracy, with only two samples misclassified. This indicates that oversampling increased the presence of minority-class patterns during training. Label 3 also showed strong performance, with most of its instances predicted correctly.

These improvements came with increased confusion among the dominant classes. Many Label 0 samples were predicted as Label 1, and a large number of Label 1 instances were classified as Label 0 or Label 3. The number of Label 1 samples assigned to Label 3 also increased compared with the baseline model. This pattern indicates that the decision boundaries shifted to improve detection of minority classes, but this change reduced separation between the dominant classes. As a result, oversampling improved recognition of rare attack categories but weakened the classifier's ability to distinguish the main traffic classes.

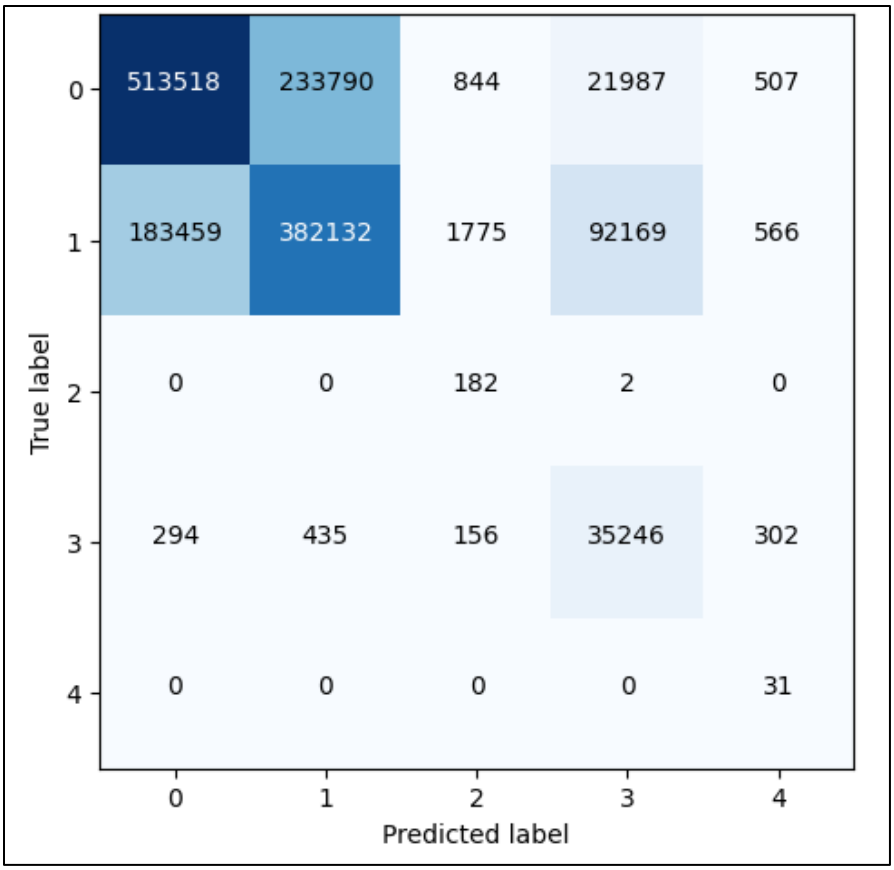

Figure 4.48 Confusion Matrix of Logistic Regression with Oversampling Technique

As depicted in Figure 4.49, the learning curve exhibits a high-bias regime characterized by early convergence and a negligible generalization gap. The model demonstrates significant sensitivity to early data increments, showing a sharp peak at 0.8×10^6samples followed by a transient performance dip. After a recovery around 2.0×10^6samples, the curve reaches a plateau and then starts to decline. This pattern shows that the model has reached its saturation point. Increasing the training set size up to 3.5×10^6samples does not produce any improvement in the F1-score.

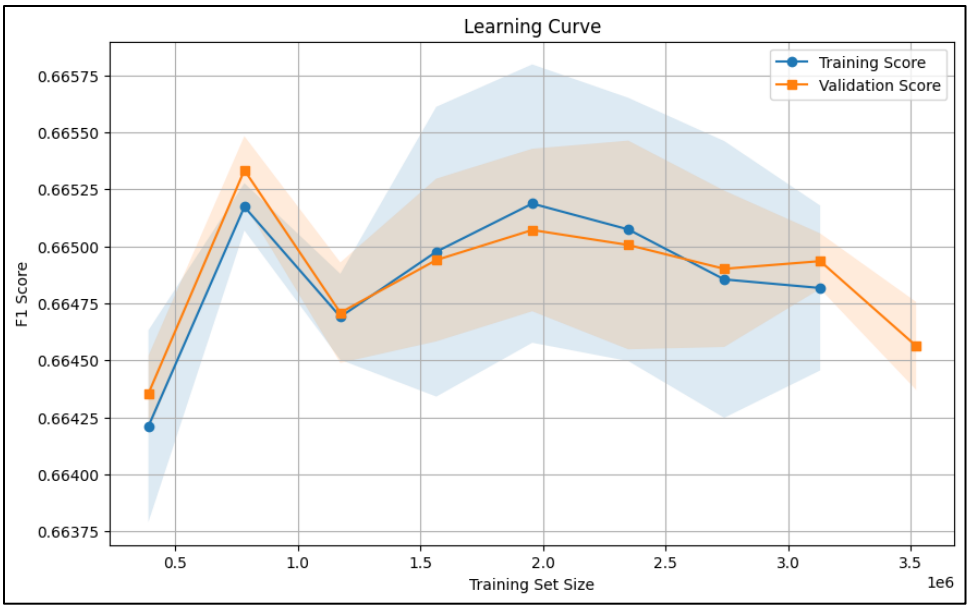

Figure 4.49 Learning Curve of Logistic Regression with Oversampling Technique

4.6.3. Logistic Regression with SMOTE

Applying SMOTE to Logistic Regression resulted in an accuracy of 65.00%. The macro-averaged precision, recall, and F1-score were 67.63%, 65.00%, and 65.75%, respectively. Compared with the baseline, SMOTE led to a small increase in precision but caused slight drops in both overall accuracy and F1-score. Even so, this approach performed better than random oversampling in all evaluation metrics. This suggests that generating synthetic samples provides a more informative and generalizable training signal than simply duplicating existing instances. Despite these gains, the model's linear constraints continue to limit its discriminative capacity.

The confusion matrix (Figure 4.50) highlights SMOTE's effectiveness in minority-class visibility, notably achieving perfect recall for Label 4 (31/31 instances) and near-perfect recognition for Label 2 (182/184). Label 3 also remained robust. Nevertheless, the persistent bottleneck remains the mutual confusion between Label 0 and Label 1. SMOTE failed to resolve this overlap, as synthetic diversity alone could not redefine the linear decision boundary within the interleaved feature space of these two dominant classes.

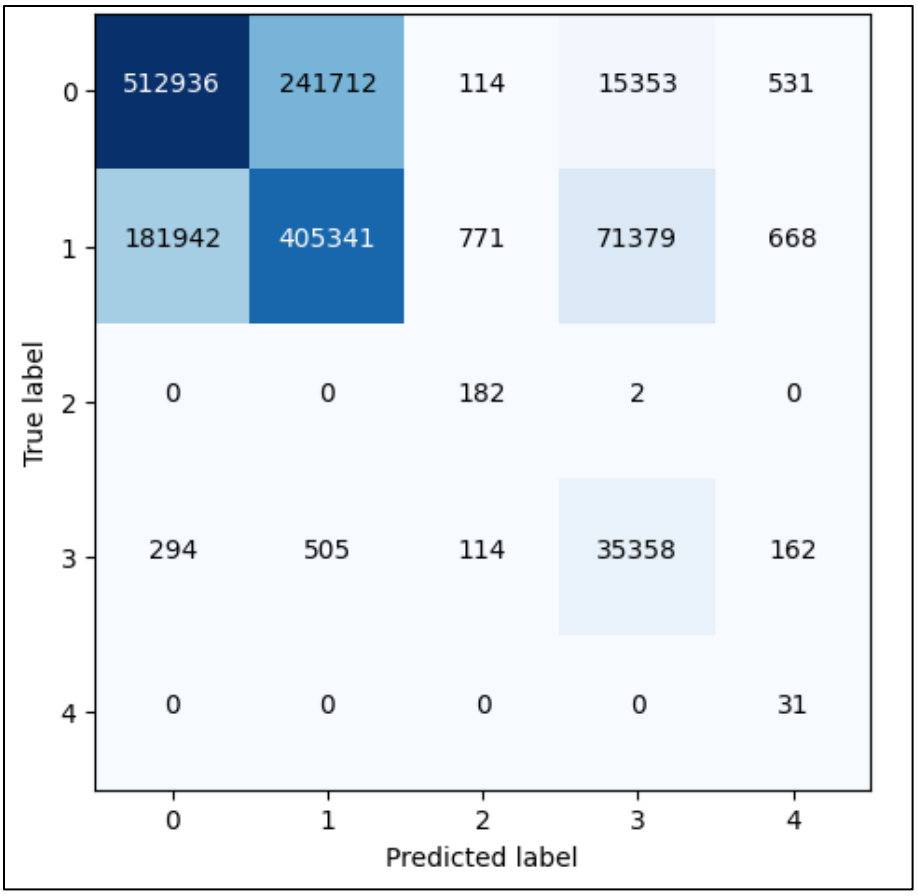

Figure 4.50 Confusion Matrix of Logistic Regression with SMOTE

As shown in Figure 4.51, the learning curve confirms a high-bias regime with a negligible generalization gap. Following an early peak at 0.75×10^6samples and a transient dip, the performance stabilizes before entering a sustained decline beyond 2.4×10^6samples. The wider validation confidence intervals, compared to the baseline, reflect increased cross-fold variability introduced by the synthetic samples. Ultimately, the stagnant and declining trajectory reinforces that the model has reached its representational ceiling; further data scaling, real or synthetic, will not yield improvements without transitioning to a more expressive architecture.

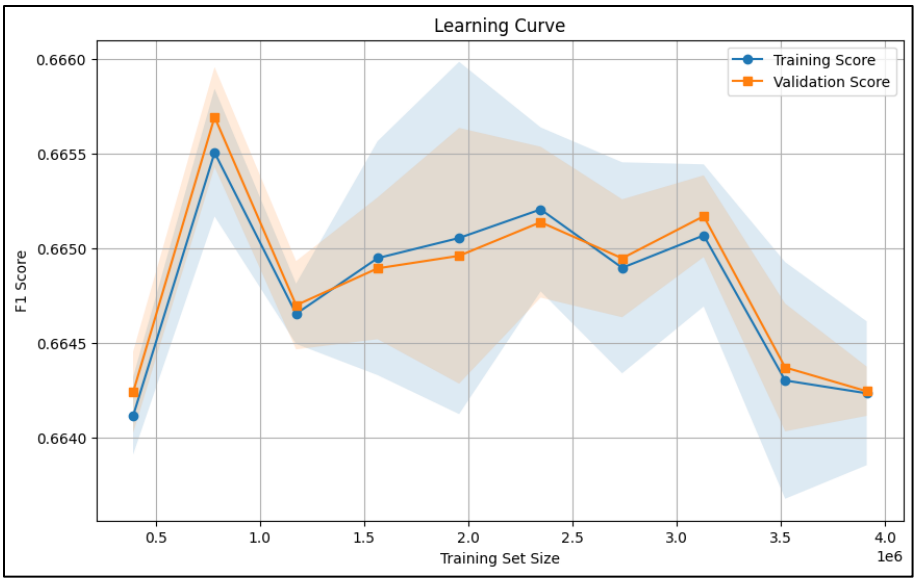

Figure 4.51 Learning Curve of Logistic Regression with SMOTE

4.6.4. Logistic Regression with Undersampling

Random undersampling resulted an overall accuracy of 57.31%, macro-averaged precision of 62.25%, recall of 57.31%, and F1-score of 58.47%. This variant exhibited the weakest performance among all Logistic Regression configurations and showed a degradation of 9.26 percentage points in accuracy and 7.97 in F1-score relative to the baseline. The lower precision compared to both SMOTE and random oversampling confirms that aggressive undersampling introduced a less favorable class balance for this classifier.

The confusion matrix in Figure 4.52 shows the trade-offs of this approach. Detection of the minority classes improved. Label 2 reached a recall of 95.1%, with 175 of the 184 samples classified correctly. Label 3 was also detected well, with 34,025 out of 36,433 instances identified correctly. Label 4 had almost perfect detection, with 29 of its 31 samples correctly predicted. These results indicate that balancing the class distribution helped the model focus more on the minority classes. As a result, the classifier achieved sensitivity levels similar to those obtained with oversampling and SMOTE.

However, this improvement came with a noticeable decline in the model's ability to distinguish the dominant classes. The recall for Label 1 dropped sharply to 42.5%, with only 280,314 out of 660,101 samples correctly identified. A large portion of these instances were misclassified,

including 265,045 assigned to Label 0 and another 110,440 predicted as Label 3. Although Label 0 achieved a relatively higher recall of 68.3%, a significant number of its samples (188,201) were still incorrectly classified as Label 1. This uneven decline in performance suggests that undersampling removed too much information from the training data and made it harder for the model to properly learn the decision boundary associated with Label 1.

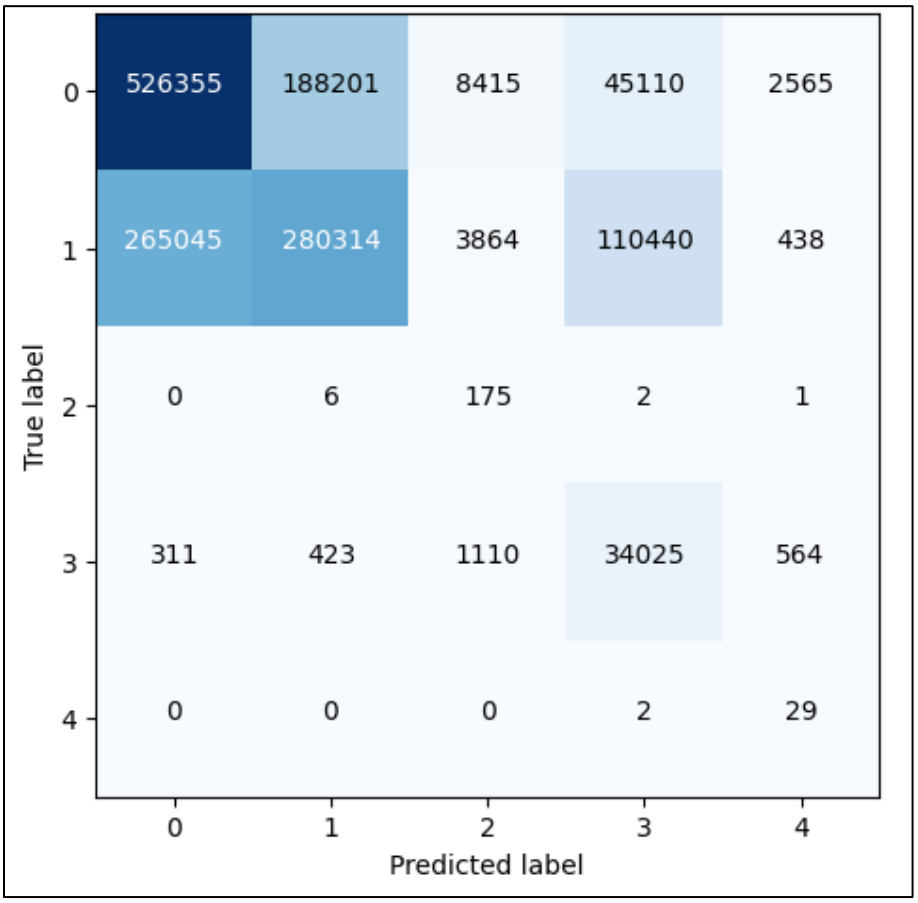

Figure 4.52 Confusion Matrix of Logistic Regression with Undersampling

As illustrated in Figure 4.53, the learning curve confirms a high-bias regime. The training and validation scores remain tightly coupled, peaking at 0.75×10^6samples before a transient dip at $1.2 \times 10^6$and a sustained decline toward the 4.0×10^6mark. The wider validation confidence intervals reflect cross-fold variability similar to SMOTE. Ultimately, the stagnant trajectory reinforces that the model has reached its representational ceiling; undersampling merely shifts the learning focus at the expense of the dominant classes' predictive integrity.

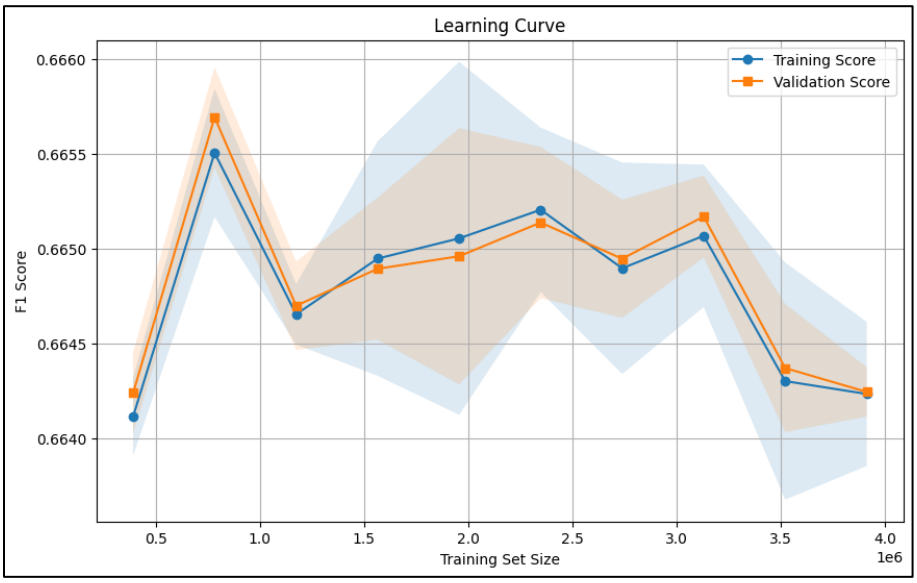

Figure 4.53 Learning Curve of Standard Logistic Regression with Undersampling

4.6.5. Logistic Regression with Class Weighting

Applying class weighting resulted in an overall accuracy of 62.83%, with macro-averaged precision, recall, and F1-score of 66.59%, 62.83%, and 64.02%, respectively. Relative to the baseline model, this configuration produced decreases of 3.74 percentage points in both accuracy and recall, and 2.42 percentage points in F1-score. Although its performance remained higher than that of the undersampling approach—indicating that penalizing misclassifications through weighted loss is less damaging than removing training data—it still ranked as the second weakest strategy, performing below both SMOTE and random oversampling.

The confusion matrix shown in Figure 4.54 reflects a pattern similar to what is typically observed when resampling methods are applied. Label 4 was detected with perfect recall, as all 31 samples were classified correctly. Label 2 was also identified with very high accuracy, with 182 out of 184 instances predicted correctly. Likewise, Label 3 achieved a strong recall rate, with 35,324 of its 36,433 samples successfully recognized. These findings suggest that increasing the penalty weights encouraged the model to pay greater attention to minority

classes, allowing it to detect these less frequent categories with sensitivity comparable to that achieved through oversampling approaches.

However, this gain significantly compromised dominant-class discrimination. Label 1 recall fell to 56.9% (375,361/660,101), which is inferior to SMOTE and comparable to the degradation in undersampling. Notable misclassifications included 188,685 instances assigned to Label 0 and 93,754 to Label 3. The latter suggests that the high penalty for Label 3 caused its decision boundary to over-extend into Label 1's feature space a systematic bias more pronounced here than in resampling approaches. Mutual confusion between Label 0 and Label 1 remained the primary bottleneck, with 234,421 Label 0 instances incorrectly attributed to Label 1.

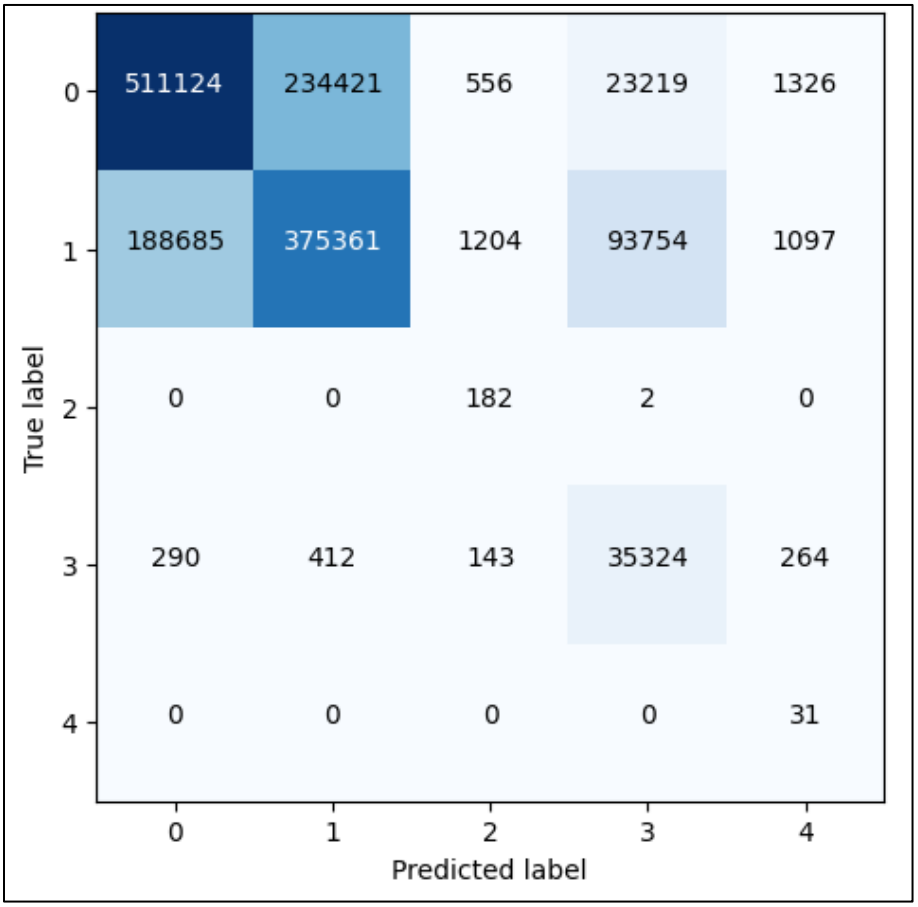

Figure 4.54 Confusion Matrix of Logistic Regression with Class Weighting

As shown in Figure 4.55, the learning curve demonstrates a behavior that differs noticeably from the previous model variants. It covers a narrower data range, beginning at 1.5×10^6samples, and generally follows a downward trend. The F1-score decreases from 0.6475 to 0.6392 at approximately 2.75×10^6samples, followed by a brief and unstable recovery to 0.6425, before settling at 0.6405. Unlike the early peak and stabilization pattern observed in other models, the use of class weighting does not lead to performance improvements as the training data increases. This suggests that, under linear constraints, adding more samples does not significantly refine the weighted decision boundaries. The training and validation curves remain very close to each other throughout the process, indicating a high-bias learning regime without signs of overfitting. However, the validation confidence interval is noticeably wider, particularly in the range between $2.0 \times 10^6$and 3.5×10^6samples. This instability reflects the sensitivity of the weighted loss function to distributional shifts across folds: as effective class frequencies change per partition, the weighted gradient signals shift unpredictably, leading to inconsistent convergence. Ultimately, this reinforces that class weighting cannot overcome the representational limits of the linear boundary in the overlapping class structure of the BoT-IoT dataset.

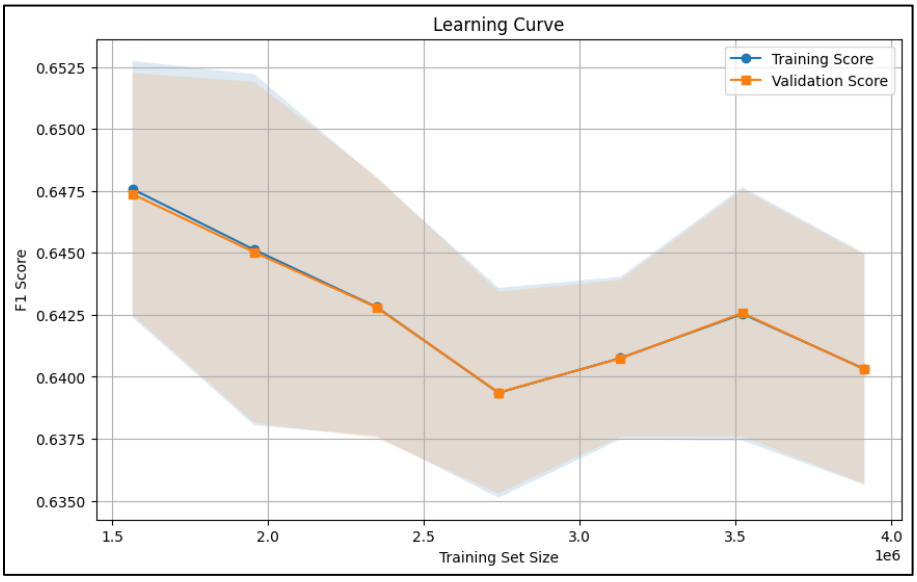

Figure 4.55 Learning Curve of Logistic Regression with Class Weighting

Across all five Logistic Regression configurations, a consistent high-bias, low-variance regime was observed, with training and validation scores remaining tightly coupled regardless of the class-balancing strategy applied. This pattern suggests that the performance limit observed in these experiments is not caused by insufficient data or overfitting. Instead, it reflects the fundamental limitation of a linear decision boundary, which cannot adequately represent the complex, nonlinear, and overlapping class distributions present in the BoT-IoT dataset.

Although the standard baseline achieved the best overall performance metrics, its complete inability to detect the rarest class makes it unsuitable for intrusion detection scenarios, where identifying minority classes is operationally critical. Among the class-balancing approaches examined (Figure 4.56 to Figure 4.60), SMOTE provided the most balanced overall performance. It achieved the highest macro-averaged precision, a competitive F1-score, and perfect recall for the minority class, while still maintaining strong discrimination for the dominant classes. Random oversampling produced broadly comparable minority-class improvements but with slightly weaker overall metrics, whereas class weighting introduced greater instability without a commensurate gain in performance. Undersampling proved the least effective strategy, yielding the weakest results across all metrics and producing a severe degradation in dominant-class recall due to the loss of informative majority-class training instances.

Overall, SMOTE is identified as the optimal configuration within the Logistic Regression family. Nevertheless, the modest absolute performance shared across all variants underscores the need for more expressive, non-linear classifiers, which are evaluated in the sections that follow.

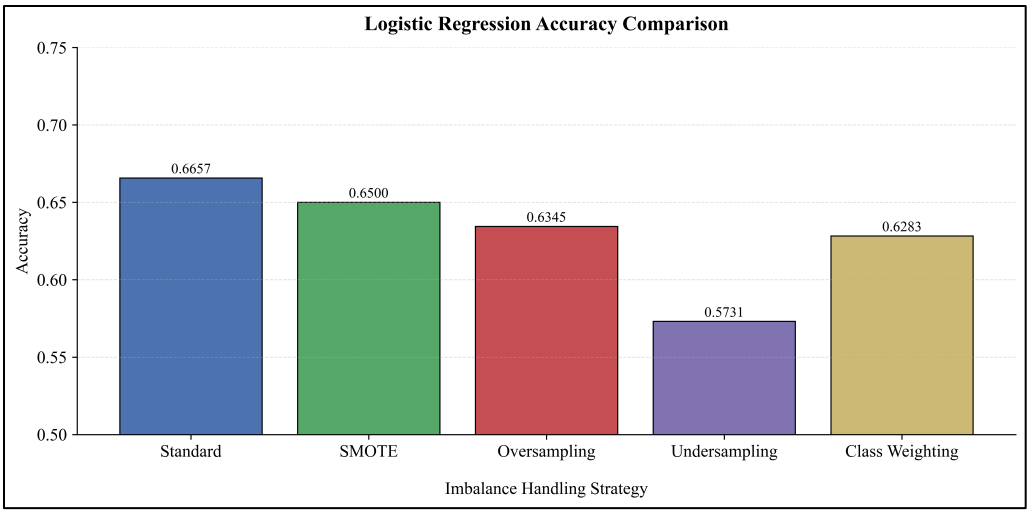

Figure 4.56 Logistic Regression Accuracy Comparison

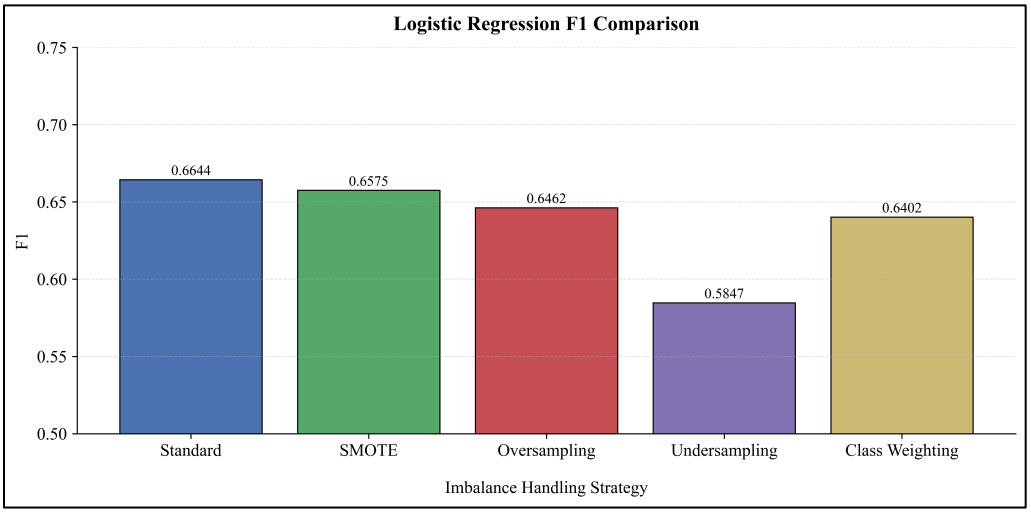

Figure 4.57 Logistic Regression F1 Score Comparison

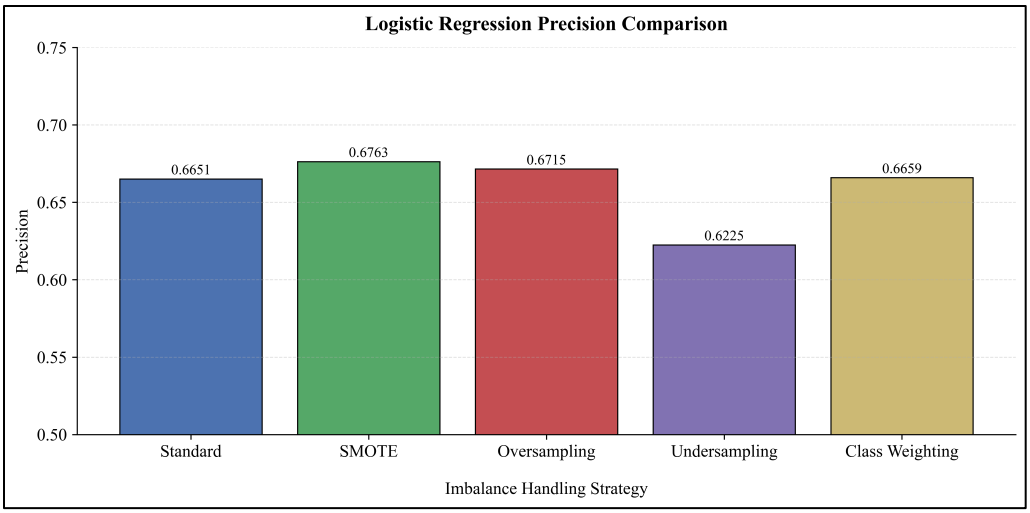

Figure 4.58 Logistic Regression Precision Comparison

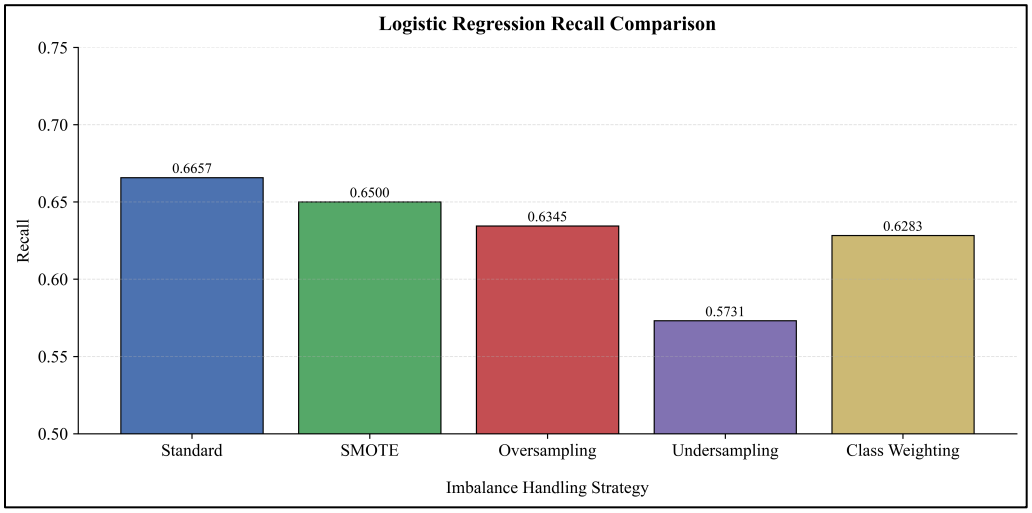

Figure 4.59 Logistic Regression Recall Comparison

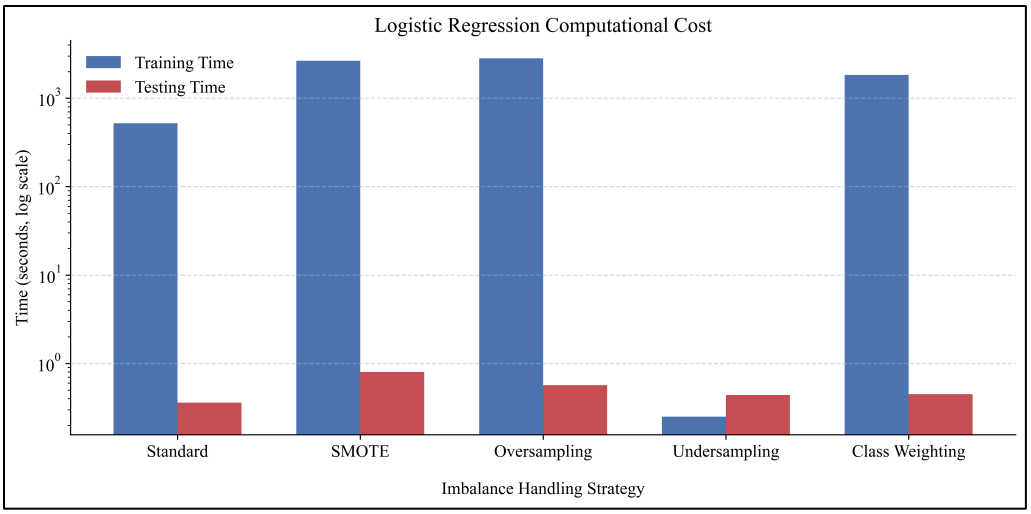

Figure 4.60 Logistic Regression Training Time and Test Time Comparison

CONCLUSION

5.1. Introduction

This chapter summarizes the overall conclusions of the study and discusses the main implications of the experimental findings reported in Chapter 4. The primary aim of this research was to evaluate the effectiveness of different machine learning classifiers for multi-class classification under class-imbalanced conditions, with particular focus on the BoT-IoT dataset. To achieve this, the study compared the behavior of several classifiers, including Logistic Regression, Random Forest, Gradient Boosting, and Artificial Neural Networks, under different imbalance-handling strategies such as random oversampling, SMOTE, random undersampling, and class weighting.

The results demonstrate that classifier performance is not only determined by the balancing technique applied, but also by the representational capacity of the underlying learning algorithm. In this respect, the study provides a comparative understanding of how linear, tree-based ensemble, and neural models respond to severe class imbalance and overlapping class distributions.

5.2. Summary of Key Findings

The results of the experiments indicate that the performance of classification models depended on both algorithms and balancing strategies. Among all evaluated classifiers, Random Forest achieved the strongest and most consistent overall performance. The results showed an almost perfect level of classification performance, with high accuracy and macro-averaged metrics. This indicates that the model was able to distinguish between both majority and minority classes even without requiring any specific techniques to address the class. This suggests that Random Forest was inherently well suited to the structural characteristics of the dataset.

Gradient Boosting also demonstrated strong predictive performance; however, its behavior was more sensitive to certain imbalance-handling configurations, particularly when class weighting was applied. The analysis of the learning curves revealed instability at specific training sizes, which may indicate that cost-sensitive optimization can create convergence challenges when

minority classes receive large weights. Therefore, although Gradient Boosting remained a strong classifier, its robustness was lower than that of Random Forest under some conditions.

The results of Artificial Neural Networks showed that balancing strategies played a much more significant role in performance improvement. In particular, ANN performed best when SMOTE and class weighting were applied, showing more balanced overall performance and a stronger ability to detect minority classes. On the other hand, random undersampling significantly reduced predictive quality, mainly because the removal of majority-class samples caused a loss of important structural information. These findings indicate that ANN-based models can benefit from data balancing, provided that the selected strategy preserves class structure and does not excessively distort the original data distribution.

A further important contribution of this study lies in the evaluation of Logistic Regression as a baseline linear classifier. Across all five LR configurations, the model consistently exhibited a high-bias, low-variance learning regime, with training and validation curves remaining closely aligned. This pattern indicates that LR did not suffer from overfitting; rather, its limited performance stemmed from the inability of a linear decision boundary to capture the complex, non-linear, and overlapping class distributions of the BoT-IoT dataset.

The standard Logistic Regression model achieved the highest overall performance within the LR family, with an accuracy of 66.57% and a macro F1-score of 66.44%. However, despite these stronger overall metrics, the model was unable to identify the rarest class, which makes it unsuitable for practical intrusion detection tasks where minority-class recognition is operationally important. Among the balancing strategies applied to LR, SMOTE provided the most effective compromise. It improved minority-class visibility, achieved perfect detection of the rarest class, and maintained the strongest macro-level balance among the imbalance-aware LR variants, with an accuracy of 65.00% and a macro F1-score of 65.75%. Random oversampling led to similar improvements for minority-class detection, although its overall performance was slightly lower. Class weighting, on the other hand, introduced additional instability without sufficient gains. Undersampling was the weakest LR strategy, leading to the greatest drop in the model's ability to distinguish dominant classes and the lowest performance across all metrics.

Across multiple classifiers, one recurring pattern remained evident: the most persistent source of error was the mutual confusion between Label 0 and Label 1. Although balancing methods often improved the recognition of minority classes such as Label 2 and Label 4, they did not fully resolve the overlap between the dominant classes. This finding suggests that the main difficulty of the task was not merely imbalance itself, but also the intrinsic similarity and overlap between the feature distributions of the major categories.

5.3. Interpretation of the Findings

The results of this study confirm that imbalance-handling strategies cannot be evaluated independently from the representational characteristics of the classifier. In the case of Logistic Regression, balancing methods improved minority-class sensitivity, but they could not overcome the inherent limitations of a linear model. This explains why all LR variants remained within a relatively modest performance range, regardless of whether oversampling, SMOTE, undersampling, or class weighting was applied. The learning curves further reinforced this interpretation by showing that additional data did not substantially improve performance once the model had reached its representational ceiling.

In contrast, the stronger performance observed for Random Forest and, to a lesser extent, Gradient Boosting indicate that non-linear ensemble methods are more capable of capturing the underlying structure of the dataset. These models can represent more complex decision boundaries and therefore handle overlap between classes more effectively. The superior ANN results under SMOTE and class weighting similarly demonstrate that models with higher representational flexibility can benefit from balancing methods when the data distribution is carefully adjusted.

Taken together, the findings suggest that the effectiveness of imbalance treatment depends on two interacting factors:

1. the severity and structure of the class imbalance, and
2. the expressive capacity of the classifier.

In other words, balancing alone is not sufficient if the classifier lacks the ability to model the complexity of the data.

5.4. Research Implications

This study has several important implications for the design of intrusion detection and other imbalanced multi-class classification systems.

First, the findings show that classifier selection is more critical than the balancing technique alone. A well-chosen non-linear classifier, such as Random Forest, may outperform a balanced but structurally limited linear model. This is particularly important in security-related applications, where the accurate detection of minority attack classes is often more valuable than maximizing overall accuracy alone.

Second, the results demonstrate that balancing methods should be tailored to the classifier type. For Logistic Regression, SMOTE offered the best compromise among imbalance-aware strategies, but its gains remained limited by linear separability constraints. For ANN models, by contrast, class weighting and SMOTE delivered substantial improvements, indicating that these techniques are better matched to models capable of learning more complex patterns.

Third, the results emphasize the need to evaluate imbalanced classification systems using macro-averaged precision, recall, and F1-score, in addition to accuracy. This is clearly illustrated by the standard LR model, which achieved the strongest aggregate LR metrics yet failed to detect the rarest class entirely. Such a model might appear acceptable based on accuracy alone, but would be operationally inadequate in real-world anomaly or intrusion detection scenarios.

5.5. Limitations of the Study

Despite the significance of the findings, several limitations should be acknowledged.

First, the experiments were conducted using a single dataset, namely BoT-IoT, which has its own class distribution, feature characteristics, traffic patterns, and degree of overlap among attack categories. Consequently, the conclusions may not be directly generalizable to other intrusion detection datasets, IoT environments, or imbalanced classification domains.

Second, model validation was based on a fixed training–test split, and cross-validation was not performed. Therefore, the reported results may be influenced by the selected data partition and may not fully reflect the variability, robustness, or generalizability of model performance across different subsets of the dataset.

Third, although the study considered several important classifiers and imbalance-handling strategies, it did not include some advanced state-of-the-art algorithms, such as XGBoost, LightGBM, CatBoost, or deep-learning architectures specifically optimized for imbalanced learning. These models may provide additional insights into the trade-off between interpretability, computational efficiency, and predictive performance.

Fourth, the study primarily focused on predictive metrics, learning behavior, and training time, with training time used as the representative measure of computational cost. Other deployment-related criteria, including inference time, memory usage, energy consumption, scalability, and feasibility in real-time environments, were not examined in depth. These factors are particularly important in operational cybersecurity systems and resource-constrained IoT devices.

Finally, although Logistic Regression served as an informative baseline because of its simplicity, computational efficiency, and interpretability, its consistently high-bias behavior suggests that linear models may be fundamentally limited for this dataset. Therefore, the role of Logistic Regression should be understood primarily as a comparative benchmark rather than as a practically competitive final solution.

5.6. Future Work

Several directions can be proposed for future research.

One promising extension is the evaluation of more advanced non-linear ensemble methods, such as XGBoost, LightGBM, and CatBoost, which may provide improved minority-class discrimination while maintaining high overall performance. Given the strong results of Random Forest, further exploration of ensemble learning appears especially justified.

A second direction involves the development of hybrid imbalance-handling strategies that combine synthetic oversampling, cost-sensitive learning, and adaptive sampling mechanisms. Such approaches may help achieve a better balance between minority-class recognition and dominant-class discrimination, particularly in cases where class overlap remains unresolved.

Third, future work could investigate feature selection, feature transformation, or representation learning techniques to reduce the overlap between Labels 0 and 1. Since this confusion

persisted across several classifiers, improving feature separability may be as important as selecting the classifier itself.

Another relevant extension would be to explore more expressive deep learning architectures and optimization methods specifically designed for imbalanced multi-class problems. Attention-based models, focal loss functions, and class-aware objective functions may offer further gains in minority-class recognition.

Future studies should also apply k-fold or stratified cross-validation rather than relying on a single training–test split. This would provide a more reliable estimate of model performance, reduce the influence of a particular data partition, and allow the stability and robustness of the evaluated models to be assessed across multiple folds.

Finally, future studies should evaluate the proposed approaches across multiple datasets with different imbalance ratios and attack distributions in order to test the robustness and external validity of the present findings.

5.7. Chapter Summary

This chapter presented the final conclusions of the study and summarized the main findings obtained from the comparative evaluation of multiple classifiers under imbalanced learning conditions. The results showed that Random Forest achieved the most robust and consistent overall performance, indicating that it was the most effective classifier for the BoT-IoT dataset among those evaluated. Gradient Boosting also performed well but showed sensitivity to certain weighted training settings. ANN models benefited substantially from SMOTE and class weighting, whereas undersampling reduced predictive quality because of the loss of important majority-class information.

The findings related to Logistic Regression further highlighted the importance of classifier expressiveness. Although LR provided a useful interpretable baseline, all of its variants remained constrained by a high-bias regime caused by the inadequacy of linear decision boundaries for modeling the non-linear and overlapping class structure of the dataset. Within the LR family, the baseline model achieved the strongest aggregate metrics, while SMOTE delivered the most effective balancing-based trade-off by substantially improving minority-

class detection. Nonetheless, the overall performance of LR remained clearly below that of the stronger non-linear models.

In conclusion, this study demonstrates that addressing class imbalance is essential, but the success of any balancing strategy depends fundamentally on the learning capacity of the classifier and the structural properties of the dataset. The results therefore support the use of more expressive non-linear models for complex intrusion detection tasks and provide a practical basis for future research on robust classification under imbalanced conditions.

BIBLIOGRAPHY

AbdelZaher, H., Ismail, N., El-Fishawy, A., Abd-El-Samie, F., & Ramadan, K. (2025). Hybrid deep learning architectures for multiclass IoT intrusion detection: Evaluation on BoT-IoT datasets. *2025 4th International Conference on Electronic Engineering (ICEEM)*, (pp. 1-8).

Agarwal, A. (2022). Load forecast anomaly detection under cyber-attacks using a novel approach. *2022 IEEE 4th International Conference on Cybernetics, Cognition and Machine Learning Applications (ICCCMLA)*, (pp. 1-6).

Aissaoui, R., Deneuville, J.-C., Guerber, C., & Pirovano, A. (2023). A survey on cryptographic methods to secure communications for UAV traffic management. *44*, p. 100661.

Alajmi, N., & Elleithy, K. (2015). Selective forwarding detection (SFD) in wireless sensor networks. *2015 Long Island Systems, Applications and Technology*, (pp. 1-5).

Almalki, S., Alghamdi, T., & Alabsi, B. (2025). Optimizing 2D CNN architectures for tabular IoT intrusion data: A comparative study using the BoT-IoT 2020 dataset. *13*, pp. 177785-177796.

Ashfaq, M., Malik, M., Fatima, U., & Shahzad, M. (2022). Classification of IoT based DDoS Attack using Machine Learning Techniques. *2022 16th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, (pp. 1-6).

Bello, O., & Zeadally, S. (2022). Toward efficient and intelligent Internet of Things: Design and implementation of an adaptive and context-aware communication middleware. *9*(11), pp. 8496-8511.

Elsadig, M. (2023). Detection of denial-of-service attack in wireless sensor networks: A lightweight machine learning approach. *11*, pp. 83537-83552.

Faker, O., & Dogdu, E. (2019). Intrusion detection using big data and deep learning techniques. *Proceedings of the 2019 ACM Southeast Conference*, (pp. 86-93).

Feng, Z., & Hua, C. (2018). Machine Learning-based RF Jamming Detection in Wireless Networks. *2018 Third International Conference on Security of Smart Cities, Industrial Control System and Communications (SSIC)*, (pp. 1-6).

Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., & Herrera, F. (2012). A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches. *42*(4), pp. 463-484.

Hameed, S., Khan, F., & Hameed, B. (2023). Understanding security requirements and challenges in the Internet of Things: A review. *2023*, pp. 1-14.

Ho, T. (1995). Random decision forests. *Proceedings of 3rd International Conference on Document Analysis and Recognition*, (pp. 278-282).

Kandhro, I., & et al. (2023). Detection of real-time malicious intrusions and attacks in IoT empowered cybersecurity infrastructures. *11*, pp. 9136-9148.

Kawaminami, I., Shao, S., Hariri, S., & Tunc, C. (2026). Attack classification and response framework (ACR) based on machine learning and CAPEC ontology. *14*, pp. 11560-11581.

Khan, S., Hayat, M., Bennamoun, M., Sohel, F., & Togneri, R. (2018). Cost-Sensitive Learning of Deep Feature Representations From Imbalanced Data. *29*(8), pp. 3573-3587.

Leevy, J., Hancock, J., Khoshgoftaar, T., & Peterson, J. (2021). An easy-to-classify approach for the Bot-IoT dataset. *2021 IEEE Third International Conference on Cognitive Machine Intelligence (CogMI)*, (pp. 172-179).

Leevy, J., Hancock, J., Khoshgoftaar, T., & Peterson, J. (2021). Detecting information theft attacks in the Bot-IoT dataset. *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, (pp. 807-812).

Li, M., & Dou, Z. (2023). Active eavesdropping detection: a novel physical layer security in wireless IoT. *119*.

Magotra, S., & Kumar, K. (2014). Detection of HELLO flood attack on LEACH protocol. *2014 IEEE International Advance Computing Conference (IACC)*, (pp. 193-198).

Manan, I., Rehman, F., Sharif, H., Ali, C., Ali, R., & Liaqat, A. (2023). Cyber security intrusion detection using deep learning approaches, datasets, Bot-IoT dataset. *2023 4th International Conference on Advancements in Computational Sciences (ICACS)*, (pp. 1-5).

Meenakshisundaram, N., & G, S. (2025). Evaluating Oversampling Strategies for Imbalanced Cervical Cancer Risk Prediction: A Comparative Analysis of SMOTE, Borderline-

SMOTE and ADASYN. *2025 Third International Conference on Networks, Multimedia and Information Technology (NMITCON)*, (pp. 1-6).

Mirsadeghi, S., Bahsi, H., Vaarandi, R., & Inoubli, W. (2023). Learning from few cyber-attacks: Addressing the class imbalance problem in machine learning-based intrusion detection in software-defined networking. *11*, pp. 140428-140442.

Mishra, A., Rajput, K., Pandey, N., & Pathak, A. (2023). Comparative analysis of classification algorithms using Bot_IoT dataset. *2023 International Conference on Sustainable Communication Networks and Application (ICSCNA)*, (pp. 1775-1780).

Mothukuri, V., Khare, P., Parizi, R., Pouriyeh, S., Dehghantanha, A., & Srivastava, G. (2021). Federated learning-based anomaly detection for IoT security. *9*(4), pp. 2545-2554.

Nadim, M., Eugenio, J., & Chennamaneni, A. (2026). A literature review on sinkhole attack detection for Internet of Things. *182*, p. 104081.

Pai, H., Kang, Y., & Chung, W. (2024). An interpretable generalization mechanism for accurately detecting anomaly and identifying networking intrusion techniques. *19*, pp. 10302-10313.

Pal, S., & Mitra, S. (1992). Multilayer perceptron, fuzzy sets, and classification. *3*(5), pp. 683-697.

Pathan, S., Prabhu, K., & Siddalingaswamy, P. (2018). Techniques and algorithms for computer aided diagnosis of pigmented skin lesions—A review. *39*, pp. 237-262.

Peterson, J., Leevy, J., & Khoshgoftaar, T. (2021). A Review and Analysis of the Bot-IoT Dataset. *2021 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, (pp. 20-27).

Pu, G., Wang, L., Shen, J., & Dong, F. (2021). A hybrid unsupervised clustering-based anomaly detection method. *26*(2), pp. 146-153.

Qureshi, K., Ahmad, A., Iqbal, R., & Aloqaily, M. (2022). A survey on intrusion detection in Internet of Things-based sensor networks. *22*(14), p. 5297.

R, S., Ayachit, S., Patil, V., & Singh, A. (2020). Competitive Analysis of the Top Gradient Boosting Machine Learning Algorithms. *2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, (pp. 191-196).

Salehpour, A., & Al-Anbagi, I. (2024). RTAP: A real-time model for attack detection and prediction in smart grid systems. *12*, pp. 130425-130443.

Sharma, A., & Babbar, H. (2023). BoT-IoT: Detection of DDoS attacks in Internet of Things for smart cities. *2023 10th International Conference on Computing for Sustainable Global Development (INDIACom)*, (pp. 438-443).

Sharma, S., & Kaul, A. (2018). A survey on Intrusion Detection Systems and Honeypot based proactive security mechanisms in VANETs and VANET Cloud. *12*, pp. 138-164.

Shaukat, K., Luo, S., Varadharajan, V., Hameed, I., & Xu, M. (2020). A survey on machine learning techniques for cyber security in the last decade. *8*, pp. 222310-222354.

Stellios, I., Kotzanikolaou, P., & Grigoriadis, C. (2022). Assessing IoT-enabled cyberattacks against critical infrastructure. *37*, p. 100523.

Thamilarasu, G., Odesile, A., & Hoang, A. (2020). An intrusion detection system for Internet of Things networks. *8*, pp. 185475-185488.

Tsogbaatar, E., Bhuyan, M., Taenaka, Y., Fall, D., Gonchigsumlaa, K., Elmroth, E., & Kadobayashi, Y. (2021). DeL-IoT: A deep ensemble learning approach to uncover anomalies in IoT. *14*, p. 100391.

Zarandi, Z., & Sharifi, I. (2020). Detection and identification of cyber-attacks in cyber-physical systems based on machine learning methods. *2020 11th International Conference on Information and Knowledge Technology (IKT)*, (pp. 107-112).

Zhao, D., Yang, B., Li, Y., & Zhang, H. (2025). Replay Attack Detection for Cyber-Physical Control Systems: A Dynamical Delay Estimation Method. *72*(1), pp. 867-875.

Zhukabayeva , T., Mardenov E. M, & Abdildaeva A. A. (2020). Sybil Attack Detection In Wireless Sensor Networks. *2020 IEEE 14th International Conference on Application of Information and Communication Technologies (AICT)*, (pp. 1-6).