A. ALAWNEH

INTRUSION DETECTION IN IOT USING MACHINE LEARNING: A FOCUS ON THE NETWORK LAYER WITH THE TON-IOT DATASET

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ATILIM UNIVERSITY

AHMAD ALAWNEH

A MASTER OF SCIENCE THESIS
IN
THE DEPARTMENT OF SOFTWARE ENGINEERING

JUNE 2025

ATILIM UNIVERSITY 2025

INTRUSION DETECTION IN IOT USING MACHINE LEARNING: A FOCUS ON THE NETWORK LAYER WITH THE TON-IOT DATASET

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ATILIM UNIVERSITY

BY

AHMAD ALAWNEH

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
THE DEPARTMENT OF SOFTWARE ENGINEERING

JUNE 2025

Approval of the Graduate School of Natural and Applied Sciences, Atılım University.

_____

Assoc. Prof. Dr. Gökhan TUNÇ
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of **Master of Science in Software Engineering, Atılım University**.

_____

Prof. Dr. Ali YAZICI
Head of Department

This is to certify that we have read the thesis INTRUSION DETECTION IN IOT USING MACHINE LEARNING: A FOCUS ON THE NETWORK LAYER WITH THE TON-IOT DATASET submitted by AHMAD I. S. ALAWNEH and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Prof. Dr. Murat KOYUNCU
Supervisor

Examining Committee Members:

Assoc. Prof. Dr. Çiğdem TURHAN
Software Engineering, Atılım University _____

Prof. Dr. Murat KOYUNCU
Information Systems Engineering, Atılım University _____

Assoc. Prof. Dr. Özgür Tolga PUSATLI
Management Information Systems, Çankaya University _____

Date: *24.06.2025*

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name :

Signature :

ABSTRACT

INTRUSION DETECTION IN IOT USING MACHINE LEARNING: A FOCUS ON THE NETWORK LAYER WITH THE TON-IOT DATASET

Alawneh, Ahmad
MSc., Department of Software Engineering
Supervisor: Prof. Dr. Murat KOYUNCU

June 2025, 76 pages

The Internet of Things (IoT) has revolutionized modern networked systems while simultaneously exposing significant security risks due to its diverse and resource-limited architecture. Conventional Intrusion Detection Systems (IDS) inadequately meet the evolving characteristics of IoT threats, particularly at the network layer. This thesis presents a feature-optimized machine learning-based Network Intrusion Detection System (NIDS) specifically designed for the ToN-IoT dataset. We propose a multi-stage framework that integrates statistical (Pearson, Spearman, Chi-Square) and embedding (Random Forest) feature selection methods to decrease dimensionality, alleviate redundancy, and improve real-time performance. Comprehensive benchmarking is conducted across a diverse set of classifiers, including Logistic Regression, K-Nearest Neighbors (KNN), Decision Tree, Random Forest (RF), Gaussian Naive Bayes, Artificial Neural Networks (ANN), XGBoost, Gradient Boosting, AdaBoost, and ExtraTrees, for both binary and multiclass intrusion detection tasks. The assessment metrics comprise F1-score, AUC, MCC, and inference latency. The findings indicate that integrating feature selection with efficient classifiers markedly enhances detection accuracy and operational feasibility in resource-limited settings. This thesis provides a replicable framework and practical insights for IoT security professionals, emphasizing the trade-offs among classifier

complexity, interpretability, and real-time applicability. The results provide a pragmatic basis for scalable and sophisticated IoT security architectures.

Keywords: IoT Security, Intrusion Detection System (IDS), Feature Selection, ToN-IoT Dataset, Machine Learning, Network-layer Attacks, Binary and Multiclass Classification.

ÖZ

İNTERNET NESNELERİ (IOT) ORTAMINDA MAKİNE ÖĞRENMESİ KULLANILARAK SALDIRI TESPİTİ: TON-IOT VERİSETİ İLE AĞ KATMANINA ODAKLANMA

Alawneh, Ahmad
Yüksek Lisans, Yazılım Mühendisliği
Tez Yöneticisi : Prof. Dr. Murat KOYUNCU

Haziran 2025, 76 sayfa

Nesnelerin İnterneti (IoT), modern ağ sistemlerini dönüştürürken, çeşitli ve kaynak kısıtlı mimarisi nedeniyle önemli güvenlik risklerini de ortaya çıkarmıştır. Geleneksel Saldırı Tespit Sistemleri (IDS), özellikle ağ katmanında IoT tehditlerinin gelişen özelliklerini yeterince karşılayamamaktadır. Bu tez, ToN-IoT veri kümesi için özel olarak tasarlanmış, özellik-iyileştirmeli makine öğrenimine dayalı bir Ağ Saldırı Tespit Sistemi (NIDS) sunmaktadır. Önerilen çok aşamalı çerçeve, boyutluluğu azaltmak, fazlalığı gidermek ve gerçek zamanlı performansı iyileştirmek amacıyla istatistiksel (Pearson, Spearman, Ki-Kare) ve gömülü (Random Forest) öznitelik seçimi yöntemlerini bütünleştirmektedir. İkili ve çok sınıflı saldırı tespiti görevleri için Lojistik Regresyon, En Yakın Komşular (KNN), Karar Ağacı, Rastgele Orman (RF), Gauss Naive Bayes, Yapay Sinir Ağları (ANN), XGBoost, Gradient Boosting, AdaBoost ve ExtraTrees dâhil olmak üzere geniş bir sınıflandırıcı kümesi üzerinde kapsamlı karşılaştırmalar gerçekleştirilmiştir. Değerlendirme metrikleri F1-skoru, AUC, MCC ve çıkarım gecikmesini içermektedir. Bulgular, özellik seçiminin verimli sınıflandırıcılarla entegrasyonunun tespit doğruluğunu ve kaynak kısıtlı ortamlardaki operasyonel uygulanabilirliği önemli ölçüde artırdığını göstermektedir. Bu tez, IoT güvenlik uzmanları için çoğaltılabilir bir çerçeve ve pratik içgörüler sunarak

sınıflandırıcı karmaşıklığı, yorumlanabilirlik ve gerçek zamanlı uygulanabilirlik arasındaki dengeyi vurgular. Elde edilen sonuçlar, ölçeklenebilir ve gelişmiş IoT güvenlik mimarileri için pragmatik bir temel sağlamaktadır.

Anahtar Kelimeler: IoT Güvenliği, Saldırı Tespit Sistemi (IDS), Özellik Seçimi, ToN-IoT Veri Kümesi, Makine Öğrenmesi, Ağ Katmanı Saldırıları, İkili ve Çok Sınıflı Sınıflandırma.

To my parents, Ibrahim & Intisar

To my brother, Sharif

To my grandfather, Sharif

To my sisters, wife and entire family

I can never thank you enough for your support and belief in me.

You are my greatest inspiration and my endless source of strength.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisor, Prof. Dr. Murat KOYUNCU, for his valuable guidance, continuous support, and patience throughout this research. His expertise, valuable feedback, and encouragement were essential to the successful completion of this dissertation.

In addition to academic guidance, I would like to extend my sincere gratitude to my family. To my father, Ibrahim, for his continuous support and encouragement; to my mother, Intisar, for her unwavering faith in me; and to my brother, Sharif, and my grandfather, Sharif, for their encouragement and inspiration. and to my sisters and my wife, whose love, understanding, and motivation sustained me during every stage of this work. I am profoundly grateful to my entire family, whose collective support has always been the cornerstone of my life.

I also extend my gratitude to my colleagues and friends who have stood by me throughout this academic journey. Their encouragement, cooperation, and companionship have made this experience even more rewarding and beneficial.

Furthermore, I would like to thank the members of the Defence Jury Committee, Assoc. Prof. Dr. Çiğdem TURHAN and Assoc. Prof. Dr. Tolga PUSATLI, for their valuable comments and insightful suggestions, which have greatly contributed to the improvement and quality of this work.

Finally, I would like to extend my sincere thanks to everyone who contributed, directly or indirectly, to this research. Your support was greatly appreciated and played a crucial role in this achievement.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ANN	Artificial Neural Network
AUC	Area Under the Curve
DT	Decision Tree
FN	False Negative
FP	False Positive
KNN	K-Nearest Neighbors
LR	Logistic Regression
MCC	Matthews Correlation Coefficient
MI	Mutual Information
MLP	Multi-Layer Perceptron
NB	Naive Bayes
PCA	Principal Component Analysis
RF	Random Forest
ROC	Receiver Operating Characteristic
SMOTE	Synthetic Minority Over-sampling Technique
TN	True Negative
TP	True Positive
XGB	Extreme Gradient Boosting (XGBoost)

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

In the 21st century, the use of Internet of Things (IoT) technology has skyrocketed, changing industries, ways of life, and a lot of operational processes in ways that have never happened before. This technological paradigm links a wide range of physical devices to the internet, allowing them to gather, share, and act on data. These devices include simple sensors and actuators in smart homes and wearable health monitors, as well as more complicated machinery in industrial control systems and critical infrastructure in smart cities [1]. The growth of these linked devices promises big improvements, such as better decision-making, more automation, and new services. Projections show that the number of active IoT connections will keep growing at an exponential rate. Estimates suggest that tens of billions of devices will be connected around the world in the next few years. This shows how IoT is changing and becoming more deeply integrated into modern society [2].

But the IoT ecosystems are growing quickly and are naturally diverse, which makes them more vulnerable to attacks and creates a much larger attack surface. This makes security much more difficult [3]. IoT devices frequently have limited resources, such as processing power, memory, and energy, which makes it hard to use strong, standard security measures. Also, many devices are made with an emphasis on usefulness and cost-effectiveness, which often leads to unsafe default settings, unpatched vulnerabilities, obsolete firmware, and a lack of defined security protocols [2], [4]. These weaknesses make IoT networks vulnerable to a wide range of cyber threats, such as Distributed Denial of Service (DDoS) attacks carried out by botnets (like Mirai and its variants), unauthorized access that leads to device hijacking, data breaches that violate privacy, malware that spreads specifically for IoT architectures, and even

actions that can cause physical harm or disrupt critical services in fields like healthcare and industrial automation [1, 5]. The size and spread of IoT deployments make these security threats even worse. This makes it especially hard to manage security from a single location, keep patches up to date, and respond to incidents quickly.

In this situation, Intrusion Detection Systems (IDS) become an important way to protect IoT environments. Firewalls, encryption, and authentication protocols are examples of classic security measures that are the first line of defense. However, they are often not enough to protect IoT devices from the changing and more advanced attacks that are happening [4]. An IDS is a second line of defense that keeps an eye on network traffic (Network-based IDS or NIDS) or host activity (Host-based IDS or HIDS) for signs of an intrusion or a security policy violation, such as harmful patterns or strange behaviors [6]. NIDS are very useful for IoT networks since some devices may not be able to run security software because of limited resources or design limits. NIDS can keep an eye on traffic at important places in the network, including gateways or routers. This gives a bigger picture of possible dangers without putting too much load on individual end-devices that do not have a lot of resources [7]. An Intrusion Detection System (IDS) perpetually observes IoT network traffic or device conduct to detect indications of intrusion or policy breaches. Although it does not directly obstruct threats, it swiftly notifies administrators or transmits suspicious activities to centralized response systems. To actively mitigate threats, Intrusion Prevention Systems (IPS) are utilized to autonomously obstruct or segregate hostile communications, rendering them an essential adjunct to Intrusion Detection Systems (IDS) in safeguarding IoT applications. This keeps IoT applications and their users safe, reliable, and secure.

Traditional IDS methods have a lot of problems when used on IoT networks because of its unique features and limitations. Signature-based IDS, which use a database of known attack patterns (signatures), have a hard time finding new or zero-day assaults.

This is a big problem in the IoT threat landscape, which is changing quickly and where new vulnerabilities and attack vectors are always appearing [6]. It can also be hard to keep and update signature databases on a lot of different IoT devices that are spread out over a wide area. On the other hand, traditional anomaly-based IDS, which try to find deviations from a set baseline of normal behavior, often have high false positive rates, especially in dynamic and heterogeneous IoT environments where normal behaviors can be different, change over time, and be different [5]. It is quite hard to come up with a complete and correct model of "normal" for a lot of different IoT devices and the different ways they send and receive data. Additionally, many traditional IDS solutions were not built with the needs of large-scale IoT deployments or the limited resources of edge computing devices in mind. This could cause performance issues or make it impossible to provide the real-time detection capabilities that are necessary for quick response [8]. Because of these problems, we need to make more advanced, flexible, effective, and smart IDS solutions that are made just for the needs and complexities of IoT ecosystems.

1.2 Problem Statement

IoT environments have several unique features, and current security solutions have some flaws that make network intrusion detection more difficult. This thesis seeks to address these concerns.

One of the biggest problems is that there are not any optimal, real-time Intrusion Detection System (IDS) models that have been particularly created and tested for network-layer IoT traffic using modern and representative datasets. There are a lot of academic proposals for IoT IDS, but many of them are only ideas, have been tested mostly on older, often synthetic datasets (like KDD99 or NSL-KDD) that do not accurately reflect the complexities of modern IoT network communications and attack vectors, or do not do a good job of meeting the real-time processing needs that are essential for quickly stopping threats in dynamic IoT scenarios [7], [8]. Network-layer traffic is a key area of focus for IoT security because most cyber-attacks on IoT systems, such as reconnaissance scans, denial-of-service (DoS/DDoS) attacks, man-in-the-middle attacks, and malware command-and-control communication, show up as strange network packet characteristics and communication patterns. The need for

optimization includes not only getting high accuracy in telling the difference between benign and malicious traffic and keeping false alarms to a minimum so that security operations personnel or automated response systems do not get overwhelmed, but also making sure that these models are fast enough to work within the resource limits that are often found at IoT gateways or edge computing nodes where NIDS functionalities are being used more and more [9].

The high number of features in network traffic data makes current NIDS models less efficient, more complicated, and sometimes even less effective. This is especially true for models that use machine learning. This is because the high number of features makes it harder for models to scale, take longer to train, and generalize. The ToN-IoT dataset used in this study is an example of a modern network dataset that can have a lot of features (sometimes dozens or even hundreds) taken from packet headers, payload statistics (when allowed), and traffic flow characteristics. These characteristics can give you a lot of detailed information that could help you tell the difference between normal and harmful activity. However, having too many features can cause a number of problems. First of all, it adds a lot of extra work to the computer for both training the model (which might take a long time) and making predictions (which can make it hard to identify things in real time). Second, it can cause the "curse of dimensionality," which is when the feature space gets so big and empty that machine learning models have a hard time learning and need a lot more data to do well [10]. More importantly, when there are a lot of dimensions, models often overfit the training data. This means they learn noise and special quirks of the training set instead of the real patterns of attacks. This makes the model less reliable and useful in real-world IoT deployments that change over time since it does not do well when it sees fresh IoT traffic that it has not seen before [11]. To create scalable, robust, efficient, and generalizable NIDS for heterogeneous IoT environments, it is very important to be able to effectively identify and select the most important features or to change the original feature space into a lower-dimensional representation while keeping important discriminatory information.

To improve the security of the quickly growing and increasingly important IoT ecosystems, it is important to deal with these two related problems: the necessity for

optimal, real-time NIDS for real IoT network traffic and the problem of high feature dimensionality.

1.3 Research Objectives

This thesis aims to address the identified problem statement through a systematic and empirical investigation, focusing on the development and evaluation of a feature-optimized machine learning framework for network intrusion detection in IoT. The primary research objectives are:

1. To implement and rigorously benchmark a machine learning-based intrusion detection pipeline as the analytical core of a Network Intrusion Detection System (NIDS), specifically tailored for IoT environments, using the contemporary and comprehensive ToN-IoT dataset.
2. To develop, implement, and thoroughly evaluate a robust feature selection and dimensionality reduction framework designed to optimize the performance, efficiency, and interpretability of the NIDS models.
3. To comprehensively evaluate and compare the performance of the developed NIDS models, both with and without feature optimization, on both binary classification (distinguishing between normal and generic attack traffic) and multiclass classification (identifying specific types of attacks) tasks.

1.4 Research Questions

To guide the research process and ensure that the objectives are met, this thesis seeks to provide empirical answers to the following key research questions:

1. Which ML classifiers from thoes selected are most effective for binary and multiclass intrusion detection on the ToN-IoT dataset?
2. What is the influence of feature selection on the classification performance and computational efficiency of machine learning classifiers regarding intrusion detection?

1.5 Contributions

This research is anticipated to make several significant and practical contributions to the field of IoT security, particularly in the domain of machine learning-based network intrusion detection:

1. **A Refined and Hybrid Feature Selection and Engineering Pipeline:** This thesis will suggest, build, and test a multi-stage feature selection pipeline that smartly combines statistical filter methods (like Pearson correlation, Spearman rank correlation, and the Chi-Square statistic) with a machine learning-based embedded importance scoring mechanism (like Random Forest feature importance). This mixed method tries to find a small, highly relevant, and non-redundant subset of features from the high-dimensional ToN-IoT dataset that will improve the performance of NIDS.
2. **Comprehensive Classifier Benchmarking with Fine-Grained Computational and Accuracy Analysis:** The study will compare and contrast many well-known and modern machine learning classifiers on the difficult ToN-IoT dataset. In addition to typical metrics for classification accuracy, the analysis will include a detailed look at important computational expenses, such as the time it takes to train a model, the time it takes to make an inference for each instance, and the time it takes for the feature reduction pipeline itself. This precise performance profile is very important for figuring out if NIDS solutions may be used in resource-aware IoT settings like edge devices or network gateways [9].
3. **Empirical Validation on a Real-World, Contemporary IoT Dataset with Reproducible Methodologies:** We will use the publicly accessible ToN-IoT dataset to test the whole suggested methodology, from preparing the data to evaluating the model. This dataset is known for being realistic and including a wide range of real IoT and IIoT network traffic, as well as several modern cyber-attack scenarios that are explicitly aimed at these environments [12]. All methods, including detailed steps for preparing the data, creating features, configuring model training (if applicable), and evaluation protocols, will be carefully recorded to make things clear, help other researchers reproduce the

work, and let the community build on these results. This adds to the expanding body of knowledge about how to design NIDS that work well and are useful for present and future IoT infrastructures.

4. **Actionable Insights into Binary and Multiclass IoT Attack Detection and Characterization:** This work will give us more useful information about the capabilities of feature-optimized NIDS [13] by systematically testing and comparing how well they do on both binary (general attack vs. normal traffic) and multiclass (identifying specific attack categories like DoS, DDoS, scanning, ransomware, etc.) classification tasks. From an operational security point of view, this difference is crucial because precisely recognizing the type of attack can greatly help and speed up the response, mitigation, and forensic investigation tactics that follow.

1.6 Thesis Organization

This thesis is organized into five chapters as follows:

- **Chapter 1: Introduction** details the study's background, motivation, research challenge, aims, research questions, and scope.
- **Chapter 2: Literature Review** examines previous studies on network intrusion detection systems (NIDS), emphasizing the utilization of machine learning methodologies, feature selection strategies, and classifier efficacy in imbalanced datasets.
- **Chapter 3: Methodology** outlines the experimental framework, encompassing the dataset description, preprocessing procedures, feature selection techniques, classification models, and evaluation criteria employed in the study.
- **Chapter 4: Results and Analysis** presents the empirical findings of the tests, compares classifier performance in binary and multiclass tasks, and examines the influence of diverse feature selection methods on detection efficacy and computational efficiency.

- **Chapter 5: Conclusion and Future Work** outlines the principal findings, addresses the research inquiries, and examines limitations and avenues for future investigation.

CHAPTER 2

LITERATURE REVIEW

This chapter gives a full overview of the current research on network intrusion detection in Internet of Things (IoT) settings, with a special focus on feature optimization and machine learning methods. It starts by talking about the security problems that are particular to IoT ecosystems. After that, it goes into further detail about intrusion detection methods, focusing on Network Intrusion Detection Systems (NIDS) that use machine learning and their designs. The article gives a critical look at the datasets used to test NIDS, focusing on the ToN-IoT dataset's relevance and structure. Then the chapter talks about feature engineering strategies, such as selection and extraction methods, and looks at other work that has been done on reducing the number of dimensions in IDS. Next, we'll look at some of the most prevalent machine learning classifiers used for intrusion detection and talk about how they compare in terms of performance and the trade-offs that come with each. Lastly, the chapter talks about common benchmarking methods and important evaluation indicators, such as real-time performance factors that are important for IoT implementations.

2.1 IoT Security Challenges

The rise of IoT devices has brought about unprecedented levels of connectivity and automation, but it has also revealed a complicated set of security problems. These problems are mostly caused by the way IoT setups work and the fact that they make it easier for attackers to get in.

- **Characteristics of IoT Environments:**

  Several unique features of IoT ecosystems make them less secure. First of all, heterogeneity is a key aspect. There are a lot of various devices from different companies that use different hardware, software, operating systems, and

communication protocols (such MQTT, CoAP, Zigbee, LoRaWAN, and Bluetooth LE) [2], [4]. This variety makes it harder to put in place universal security standards and manage security in a consistent way. Second, a lot of IoT devices do not have a lot of resources, like computing power, memory, and battery life [3], [14]. These limits often make it impossible to employ advanced security algorithms or strong encryption methods directly on the devices. Third, scalability is a big problem because IoT installations might entail dozens to millions of devices that are all connected to each other, making it impossible to manage and monitor each one separately [2]. A lot of IoT devices are made to work without anyone being there or from a distance for long periods of time, and they are typically in places that are easy to get to, which makes it hard to do manual updates, security checks, or other things [3], [14]. Also, the need to go to market quickly and cheaply sometimes results in insecure default settings, weak or hardcoded passwords, and a lack of regular patching or firmware updates. This makes devices vulnerable to known attacks for long periods of time [2, 3]. Some IoT devices have long lifespans, which means they might exceed their maintenance period and become permanently susceptible.

- **Attack Surfaces in IoT Networks:**

  These traits make IoT networks' attack surfaces much bigger and more complex. They may be divided into three main layers: device, network, and application/cloud [2], [3].

  - **Device-Level Attacks:** You can attack individual IoT devices by changing their firmware, taking advantage of unpatched security holes in their operating systems or embedded software, or using weak or default passwords [14]. Hardware-level attacks, such side-channel attacks or physical manipulation to get important information, are also a worry, especially for devices that are easy to get to [3].

- **Network-Level Attacks:** The manner that IoT devices, gateways, and backend servers talk to each other are excellent targets. Attackers can use flaws in network protocols to listen in on conversations, steal data, and change it [4]. Denial of Service (DoS) and Distributed Denial of Service (DDoS) assaults, which are commonly carried out by hacked IoT devices that form botnets (like Mirai), try to make networks unavailable [1]. Routing attacks, spoofing, and session hijacking are also common risks to several IoT communication protocols [2].
- **Application and Cloud-Level Attacks:** Another big attack surface is IoT apps, backend servers, and cloud platforms that handle and store IoT data. APIs that are not safe, data that is not stored securely, authentication and authorization methods that are not strong enough, and online or mobile interfaces that are not secure can all allow unauthorized access, data breaches, and service interruptions [1]. Because cloud platforms are centralized, they can also be good targets for big attacks.

To better conceptualize how diverse layers of an IoT ecosystem interact and expose vulnerabilities, the following figure presents an integrated architecture where data flows from local users and devices to cloud-based services and administrative interfaces. This multi-layered structure inherently reveals the three primary domains of the IoT attack surface:

1. **Device-level** vulnerabilities such as weak authentication and unpatched firmware,
2. **Network-level** threats including eavesdropping, spoofing, or DoS attacks, and
3. **Application/cloud-level** risks such as insecure APIs, misconfigured cloud storage, and inadequate access control.

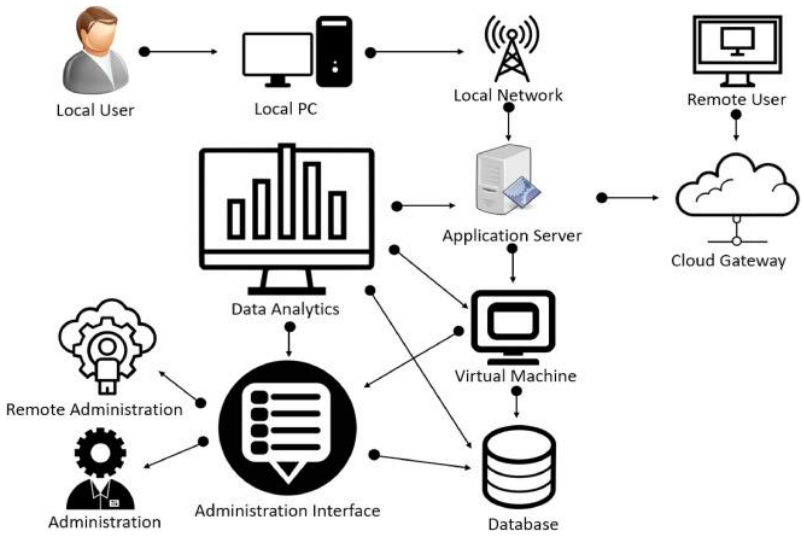

Figure 2.1 IoT System Architecture (High-Level)

address these various issues, a multi-layered security system is essential, with Intrusion Detection Systems playing a critical role in identifying and responding to threats that bypass primary security protocols. Figure 2.1 presents a comprehensive architectural overview of an IoT system, depicting the data flow across local devices, cloud gateways, and administrative interfaces each posing possible vulnerabilities across the device, network, and application levels.

2.2 Intrusion Detection in IoT Networks

IoT environments are naturally vulnerable, and simply preventative security solutions have their limits. That is why Intrusion Detection Systems (IDS) are so important for making IoT networks safer [4]. An IDS keeps an eye on system or network activity for malicious acts or policy violations and either sends alerts or lets automated responses happen.

- **Types of IDS: Signature-based vs. Anomaly-based:**

  IDS can be broadly classified based on their detection methodology:

  - **Signature-based IDS (Misuse Detection):** These systems depend on a set database of known attack patterns or signatures. When an activity that is being watched matches a signature, an alert goes off [5]. The main benefit is that they are quite accurate and have a low false positive

rate (FPR) when finding known threats [6]. The biggest problem with these tools is that they cannot find new or zero-day threats. Their usefulness depends solely on how complete and up-to-date signature updates are. It can be hard to keep signatures up to current on a lot of devices that may not have a lot of resources in dynamic IoT environments where new threats are often appearing [4].

- **Anomaly-based IDS (Behavior-based Detection):** First, these systems provide a baseline model of what normal behavior looks like for the network or system being monitored. Any change from this baseline is seen as a possible incursion [5]. The main benefit is that they can find attacks that have never been seen before or that are happening right now, as they do not need to know anything about the characteristics of the attack beforehand [6]. But anomaly-based IDS often have greater false positive rates since normal actions that are not seen could be wrongly labeled as unusual. It is especially hard to come up with a precise and all-encompassing model of "normal" in IoT settings that are diverse and always changing [15], [19].
- **Machine learning-based IDS typically fall under the category of anomaly-based detection.** Machine learning-based Intrusion Detection Systems (IDS) are predominantly classified as anomaly-based detection systems due to their fundamental operational principle: modeling and identifying deviations from established norms in network behavior. Traditional anomaly-based IDS do not rely on predefined attack signatures; instead, they construct behavioral baselines either implicitly or explicitly from observed data. In this thesis, although supervised learning is employed, the categorization remains within anomaly-based detection because the ML models are trained to detect deviations from labeled "normal" versus "attack" behaviors, rather than matching known signature patterns. This approach supports the detection of novel or previously unseen threats, which is crucial for the dynamic and heterogeneous nature of IoT traffic. Supervised anomaly-

based detection involves training on labeled datasets to classify traffic as benign or malicious based on statistical and flow-level features. These features capture characteristics such as packet sizes, flow durations, and protocol usage patterns that often deviate in the presence of malicious activity. Despite being supervised, this detection strategy remains anomaly-based because the underlying principle is learning the distribution of "normal" vs. "abnormal" patterns, not specific attack identifiers. This paradigm is especially valuable in IoT environments where the attack landscape is rapidly evolving and signature databases can quickly become outdated. Nonetheless, supervised anomaly-based models face challenges, including class imbalance (normal traffic overwhelmingly outnumbers malicious instances), non-stationarity of data (as benign and malicious behaviors evolve), and the difficulty of modeling highly diverse device behaviors under real-world deployment conditions. These limitations necessitate robust feature selection, class balancing strategies such as SMOTE, and the use of models capable of generalizing to unseen traffic patterns [7], [9], [10].

  - **Hybrid IDS:** Hybrid IDS use both signature-based and anomaly-based detection methods to get the best of both worlds. This makes it possible to reliably find existing assaults and also find new ones, which could help balance accuracy and threat coverage [15].

- **Machine Learning-based NIDS Architectures:**

  Machine Learning (ML) approaches are becoming more and more popular for making IDS, especially NIDS, smarter and more flexible. This is because they can learn complicated patterns from massive amounts of network traffic data and adapt to new threats [7]. There have been many ideas about how to use ML-based NIDS in IoT:

  - **Centralized Architecture:** In this strategy, network traffic data from different parts of the IoT network (or from all devices) is sent to a central server or cloud platform where the ML-based IDS model is stored and does its job [21]. This architecture makes use of the central

location's large computing power, which makes it possible to train and analyze sophisticated models. But it can slow down detection, add a lot of extra work to communication, and make one point of failure. Sending possibly private information to a central location also raises privacy issues [4].

- **Distributed Architecture:** This method uses lightweight IDS agents or models on each IoT device, gateway, or edge node [4], [20]. Detection happens closer to the data source, which speeds up response times and lowers the burden on the network. But IoT devices do not have a lot of resources, therefore the ML models that can be run directly on them cannot be too complicated [20]. Gateways or edge nodes are a good middle ground because they have more resources than end-devices but are still closer to the network segment they watch.
- **Hierarchical/Hybrid Architecture:** This method uses parts of both centralized and distributed techniques. Local IDS agents might do the first filtering or find common problems, while a central entity or higher-level nodes in the hierarchy do more complicated analysis or link global threats. [19].
- **Federated Learning (FL) Architecture:** FL has become a promising distributed ML model for IoT IDS [9]. In FL, a central server coordinates the training of a global ML model by several IoT devices or edge nodes. However, they do not share their raw local data. Each device trains a model on its own data, and only changes to the model (such gradients or parameters) are sent to the server to be put together. This method protects data privacy, cuts down on communication costs, and lets models learn from a wider range of local data distributions [9], [18]. [9] showed a federated learning-based IDS that specifically dealt with IoT data that was not balanced.

The choice of architecture depends on factors like the scale of the IoT deployment, resource availability, latency requirements, and privacy considerations.

2.3 Datasets for NIDS Evaluation

The development and rigorous evaluation of NIDS, especially ML-based ones, heavily rely on the availability of suitable datasets that accurately reflect real-world network traffic and attack scenarios [8].

- **DARPA, KDD99, NSL-KDD, BoT-IoT:**

  Several datasets have historically been used for NIDS research:

  - **DARPA (1998, 1999) and KDD Cup 1999 (KDD99):** These were the first datasets made in a synthetic network setting. They used to be important, but because network technologies and attack methods have changed, they are now seen as outmoded [16], [22]. KDD99, which comes from DARPA, has problems like duplicate records and an uneven distribution of attack kinds.
  - **NSL-KDD:** NSL-KDD was made to fix some of KDD99's problems, such getting rid of duplicate records. However, it is still based on the same old traffic and attack patterns, which makes it less useful for testing contemporary NIDS. [16], [22].
  - **More Recent General NIDS Datasets:** Datasets like UNSW-NB15 and CICIDS2017 (and its successors like CSE-CIC-IDS2018) offered more contemporary network traffic and a broader range of attack types compared to KDD-era datasets. However, their primary focus was not specifically on IoT environments, although some IoT-like traffic might be implicitly present [8].
  - **BoT-IoT:** This dataset, developed in [25], is all about IoT network traffic and has realistic botnet assaults, DoS/DDoS, reconnaissance, and information theft [25]. It has traffic that comes from a real IoT network testbed it goes into great detail about its features, pointing out that it is useful for certain types of IoT botnet study but also that it only looks at a small number of IoT threats.

While these datasets have contributed to NIDS research, many lack the specific nuances of diverse IoT device communications, modern IoT-centric attacks, and the scale of current IoT deployments [8], [16].

- **Focus on ToN-IoT: Structure, Relevance, and Challenges:**

  Moustafa et al. at the University of New South Wales (UNSW) Canberra Cyber Range created the ToN-IoT dataset (also known as UNSW-IoT or IoT-Ton) to fix the problems with other datasets and give a better way to test security solutions in modern IoT and Industrial IoT (IIoT) settings [12], [13] .

  - **Structure and Generation:** The ToN-IoT dataset came from a big, real-world testbed that integrates several IoT and IIoT devices, sensors, actuators, industrial control systems (ICS) (using protocols like Modbus and BACnet), and cloud, edge, and fog computing platforms [12]. We got data from a lot of different places, like network traffic (packet captures, NetFlow, Zeek/Bro logs), operating system logs from Windows and Linux systems, and telemetry data from IoT devices. This thesis is mostly about the network traffic part.
  - **Features:** ToN-IoT has a lot of features in its network traffic datasets. For example, the "ToN_IoT_NetworkLayer" dataset has more than 40 features based on network flows, like the source and destination IP and port, the protocol, the flow time, the packet and byte counts, the TCP flags, and features that are particular to HTTP, SSL, DNS, MQTT, and CoAP communications [12].
  - **Attack Types:** ToN-IoT's biggest strength is that it includes a wide range of real IoT/IIoT traffic and different types of modern cyberattacks, such as DoS, DDoS, scanning (like port scans and OS fingerprinting), ransomware, backdoor, injection (like SQL injection and XSS), password cracking, man-in-the-middle (MITM) attacks, and attacks that target IoT protocols (like MQTT floods and CoAP reconnaissance) and industrial systems [12], [17]. The dataset has

labels for both binary classification (normal vs. attack) and multiclass classification (particular sorts of attacks).

- **Relevance:** The ToN-IoT dataset is very useful for current NIDS research in IoT because it includes a wide range of IoT device traffic (from smart fridges and thermostats to industrial sensors), realistic attack vectors, and data from different network segments (IoT, IIoT, and traditional IT). Using it lets you test NIDS against a wide range of attacks that are likely to happen in real-world smart environments [13].
- **Challenges:** Even though it has some good points, using the ToN-IoT dataset can be hard. Because it has a lot of extracted features, it usually has a lot of dimensions, which can affect how well the model trains and works [12]. Another common problem is a big class imbalance, when normal traffic instances outweigh instances of certain attack kinds by a huge amount. This could make ML models favor the dominant class [9]. Because the dataset is so big, it also needs efficient ways to handle and model the data.

To complement the discussion on the strengths and limitations of the ToN-IoT dataset, Table 2.1 provides a comparative summary of recent IDS research efforts that utilized this dataset. It includes details such as the machine learning algorithms employed, feature selection techniques, evaluation metrics, classification tasks (binary or multiclass), and overall performance. This comparative view highlights both the methodological diversity and current gaps in existing literature, especially regarding feature optimization and real-time performance, which this thesis aims to address.

Table 2.1 Comparison of recent research studies utilizing the ToN-IoT dataset for network intrusion detection

Study	Dataset	Feature Strategy	Approach	Strengths	Research Gap
[27]	ToN-IoT	Spearman correlation	Stacking ensemble with CatBoost, Extra Trees, and XGBoost	Stacking ensemble, Spearman FS, high MCC	No exploration of feature interaction, lacks comparative analysis with simpler models like KNN or LR
[28]	ToN-IoT, Edge-IIoT, UNSW2015	None (DL on raw data)	DenseNet, InceptionTime	High accuracy, multi-dataset DL	Ignores feature engineering; not comparable for classical ML baselines or resource-constrained
[29]	CIC-IDS2018, BoT-IoT, ToN-IoT	None specified	Multilayered DNN	Layered security design, transport-level focus	No multiclass results; ToN-IoT used without feature explanation or preprocessing strategy

Table 2.1 (Continued)

[30]	ToN-IoT	Auto feature generation, multicollinearity filtering	Random Forest	High AUC with automated features	Single model used; lacks comparative multi-model and multiclass performance analysis
[31]	ToN-IoT, WUSTL-IIoT, EdgeIIoTset	Mutual Information	Bagging, Stacking, LGBM, RF, DT	Transfer learning, real-time feasibility	No deep model comparison
[32]	ToN-IoT (combined)	Cyclic temporal encoding, SMOTE, Tomek	Multiple ML and DL models	Temporal encoding, hybrid sampling	Focused more on preprocessing than model comparison; lacks standard FS methods like RF, Chi2
[33]	ToN-IoT	Heterogeneous source emphasis	Dataset-centric evaluation	Emphasis on dataset quality and heterogeneity	Does not propose or evaluate any ML models; no benchmarking or FS insight

2.4 Feature Engineering for IDS

Feature engineering is an important stage in preparing data for machine learning-based IDS. It means changing raw data into a format that better shows the problem to the learning algorithms. This frequently makes the model work better, makes it easier to understand, and makes it less complicated to compute [10], [11].

- **Feature Selection vs. Extraction:**

  Two primary approaches to feature engineering for dimensionality reduction are feature selection and feature extraction [24].

  - **Feature Selection:** This process involves selecting a subset of the original features that are most relevant to the prediction task (i.e., distinguishing normal from malicious traffic) while discarding irrelevant or redundant features [10]. Feature selection methods can be categorized into:

    - Filter methods: These look at how useful features are based on their own statistical properties, like correlation, mutual information, and statistical tests, without using any ML technique. In general, they are rapid to compute [24].

    - Wrapper methods: These employ a certain ML method to figure out how useful distinct groups of features are. They usually do better with the specified algorithm, but they cost more to run [11].

    - Embedded methods: These do feature selection as part of the model training process (for example, LASSO regression and feature importance from tree-based models) [10].

  - **Feature Extraction:** This procedure uses functional mapping to make a new, smaller set of features (transformed features) from the original feature set. The goal is to get the most important information into a space with fewer dimensions [24]. The new features are made up of the old ones and may not have a clear physical meaning.

- **Techniques: Pearson, Spearman, Chi-square:**

  This thesis considers several common techniques for feature selection and extraction:

  - **Pearson Correlation Coefficient (Filter Method):** This tells you how closely two continuous variables are related in a straight line. In IDS, it can be used to find highly correlated (redundant) features among predictors or to figure out how each feature is related to the target class (if the class is represented as a number) [10]. It goes from -1 to +1.
  - **Spearman Rank Correlation Coefficient (Filter Method):** This tells you how the two variables are related in a way that does not change over time. It checks to see how well a monotonic function can describe the relationship between two variables. It is less affected by outliers than Pearson correlation and can find non-linear connections to some extent [10].
  - **Chi-Square (χ^2) Test (Filter Method):** People often use this statistical test to see if there is a strong link between two categorical variables. In IDS, it can be used to choose features that depend on the categorical target class, like normal, attack type A, or attack type B [11]. Features with greater Chi-Square scores are seen to be more important.

- **Related Works on Dimensionality Reduction for IDS:**

  Many research have shown how dimensionality reduction can help IDS for IoT. Mascari and Rullo [10] did a thorough study of different feature selection methods for IDS in IoT networks. They showed that these methods might increase performance and make things less complicated. [11] looked at feature selection methods that are only used for IoT IDS. They focused on how these methods may help deal with data from many different IoT devices that have a lot of dimensions. [24] provide a more in-depth look at feature selection and dimensionality reduction methods that may be used with any IDS.

These studies show that dimensionality reduction can make models more efficient, speed up training and inference durations, and even improve detection accuracy by getting rid of noise and extra information.

2.5 Machine Learning Classifiers for IDS

A wide array of ML classifiers has been investigated for intrusion detection, each with its own strengths and weaknesses, particularly when applied to the unique demands of IoT environments.

- **Random Forest (RF), XGBoost, KNN, AdaBoost, ANN, etc.:**

  Some of the most frequently employed classifiers in recent IDS literature include:

  - **Random Forest (RF):** An ensemble learning method that builds several decision trees during training and gives the class that is the mode of the classes (classification) or the mean prediction (regression) of the individual trees [12]. RF is notable for being able to handle high-dimensional data, without overfitting, and giving feature importance scores [26].

  - **XGBoost (Extreme Gradient Boosting):** A distributed gradient boosting library that has been developed to be very fast, flexible, and portable. It uses L1 and L2 regularization with gradient boosting machines to stop overfitting, and it often gets the best results on many classification problems [26]. Using feature significance analysis, [26] showed that XGBoost works well for detecting intrusions in IoT systems.

  - **K-Nearest Neighbors (KNN):** A non-parametric, instance-based learning technique that uses the k-nearest neighbors in the feature space to classify an item by a majority vote [7]. KNN is easy to use, but it might take a lot of processing power to make predictions with big datasets. It is also susceptible to feature scaling and irrelevant features (the "curse of dimensionality").

- **AdaBoost (Adaptive Boosting):** An ensemble learning algorithm that puts together several weak learners (usually decision stumps) in a row, with each new learner focusing more on examples that were misclassified by earlier learners [7]. AdaBoost works well, although it can be susceptible to noisy data and outliers.
- **Artificial Neural Networks (ANN) and Deep Learning (DL):** Artificial neural networks (ANNs), especially deep learning models like Multi-Layer Perceptrons (MLPs), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs), can automatically learn hierarchical feature representations from raw data and model complicated non-linear relationships [5].
- **Other Classifiers:** Support Vector Machines (SVM), Naive Bayes, Logistic Regression, and Decision Trees are also commonly found in IDS literature [28], [27], though ensemble methods and deep learning have garnered more attention recently for their performance on complex datasets.

- **Performance Comparison in Literature:**

  Many A multitude of research have investigated the efficacy of these classifiers for Network Intrusion Detection Systems (NIDS). Ensemble algorithms such as Random Forest and XGBoost frequently exhibit superior accuracy and stability across various IDS datasets, including those tailored to IoT traffic [7], [26]. Deep learning models have demonstrated significant potential, especially in analyzing raw network data and managing intricate attack patterns. Nonetheless, their enhancements in performance must be considered with their computational complexity and resource requirements [5], [12], [27].

  Specifically, [8] offers a comprehensive evaluation of diverse machine learning techniques for intrusion detection. These assessments corroborate the agreement that no singular classifier is uniformly superior. The efficacy of any classifier is significantly influenced by the dataset employed, the

characteristics of the attack types, the feature set, and the assessment metrics emphasized. [28]

- **Trade-offs: Accuracy, Interpretability, Resource Usage:**

  When selecting an ML classifier for IoT IDS, several trade-offs must be considered:

  - **Accuracy vs. Interpretability:** Deep neural networks and complex ensembles are examples of very accurate models that often work as "black boxes," which makes it hard to figure out how they make decisions [5]. Decision trees and logistic regression are examples of simpler models that are easier to understand but may not be as accurate. For security analysts to check alerts and figure out how attacks work, being able to interpret them can be quite important.
  - **Accuracy vs. Resource Usage:** This is a very important trade-off in IoT. Models that require a lot of computation (such deep learning or some ensembles during inference) may be quite accurate, but they might not work well on IoT gateways or edge devices that do not have a lot of resources [9], [20]. If lightweight models can give good detection rates, they are frequently the best choice [20]. Training time, inference latency, and memory footprint are all key things to think about when it comes to resources. [20] focused on making lightweight and reliable IDS for IoT in particular. The goal is frequently to establish a model that strikes the greatest balance between these factors for the given IoT application and deployment environment.

2.6 Benchmarking and Evaluation Metrics

Rigorous benchmarking using appropriate evaluation metrics is essential for assessing the true performance of an NIDS and comparing different approaches fairly [8].

- **Accuracy, Precision, Recall, F1-Score, MCC:**

For classification tasks in IDS, several metrics derived from the confusion matrix (True Positives - TP, True Negatives - TN, False Positives - FP, False Negatives - FN) are commonly used [27]:

- **Accuracy:** (TP + TN) / (TP + TN + FP + FN). Shows how correct the model is as a whole. But accuracy can be a bad measure for datasets that are not balanced (as in IDS, where normal traffic is far more prevalent than attack traffic), because a model that predicts the majority class most of the time can have high accuracy but do poorly on minority (attack) classes.
- **Precision (Positive Predictive Value):** TP / (TP + FP). Calculates the percentage of successfully detected positive instances (attacks) out of all cases that were projected to be positive. High precision is critical for reducing false alarms, which might make security analysts tired of getting alerts.
- **Recall (Sensitivity, True Positive Rate - TPR):** TP / (TP + FN). This tells you how many of the real positive cases (attacks) the model accurately detected. High recall is important for reducing missed attacks (false negatives), which can have serious effects.
- **F1-Score:** 2 * (Precision * Recall) / (Precision + Recall). The harmonic mean of precision and recall gives you one score that balances both measurements. It is especially helpful when the class distribution is not even.
- **Matthews Correlation Coefficient (MCC):** (TP*TN - FP*FN) / sqrt((TP+FP)(TP+FN)(TN+FP)(TN+FN)). A correlation coefficient between the observed and predicted binary classifications. It ranges from -1 (total disagreement) to +1 (perfect prediction), with 0 indicating random prediction. MCC is considered one of the most robust and balanced metrics for imbalanced binary classification tasks as it takes into account all four values in the confusion matrix.

- **False Positive Rate (FPR):** FP / (FP + TN). Measures the proportion of actual negative instances (normal traffic) that were incorrectly classified as positive. It is also known as Fall-out.
- **Area Under the ROC Curve (AUC-ROC):** The ROC curve plots TPR against FPR at various classification thresholds. AUC-ROC represents the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one, providing a measure of the model's ability to distinguish between classes across all thresholds.

- **Real-time Considerations: Training Time, Inference Latency:**

  Beyond classification performance, real-time operational aspects are critical for practical NIDS deployment in IoT [9], [20]:

  - **Training Time:** The time required to build or train the ML model. While often performed offline, excessively long training times can hinder the ability to frequently retrain and adapt the model to new threats or evolving network behavior. For adaptive or online learning systems, training efficiency is paramount.
  - **Inference Latency (Prediction/Detection Time):** The amount of time it takes the trained model to sort a new network instance, like a packet or flow. Inference latency must be very low for real-time NIDS so that attacks may be found and dealt with quickly before they cause a lot of harm [9]. This is especially important for IoT apps that need to work quickly or in networks that are very fast.
  - **Throughput:** The number of network packets or flows that the NIDS can process per unit of time. High throughput is necessary to keep pace with network traffic without becoming a bottleneck.
  - **Computational Resources (CPU, Memory):** The amount of processing power and memory required by the NIDS, both for training and inference. This is a major constraint for deployment on resource-limited IoT devices or edge nodes [20].

Studies like [9] on federated learning for IoT IDS stress how important it is to keep track of communication and computational burden, while [20] focus on making lightweight IDS that work well with the limitations of IoT devices. As suggested in [27], a full evaluation must take into account both the effectiveness of detection and these operational efficiencies.

This literature review has set the stage by looking at the problems with IoT security, the many types of IDS and their roles, the need of using the right datasets like ToN-IoT, the need for feature engineering, the strengths of different ML classifiers, and the most important evaluation metrics. The next parts of this thesis will expand on this and create and test a feature-optimized ML framework for NIDS in IoT.

CHAPTER 3

METHODOLOGY

This chapter goes into detail about the methodological framework that was created to meet the study goals set out in Chapter 1. The focus of this study is to design, build, and thoroughly test a feature-optimized Network Intrusion Detection System (NIDS) that is specifically made to operate well in the complicated settings of the Internet of Things (IoT). This chapter starts by showing the big picture conceptual framework and the detailed, multi-step data processing pipeline. It then goes into detail about the ToN-IoT dataset, explaining why it was chosen and how its network-layer data will be used. After that, the chapter goes into more detail on the steps in the methodology, such as data preprocessing, advanced feature engineering and selection techniques, the different machine learning models utilized, and the full set of metrics used to measure their performance and usefulness in real life.

3.1 Conceptual and Operational Framework

The research methodology is founded upon a dual-path experimental design, engineered to empirically quantify the benefits of feature selection against a non-optimized baseline. This structure, conceptually illustrated in **Figure 3.1**, allows for a direct and robust comparison of model performance under different data conditions.

- **Path 1: Baseline (Without Feature Selection):** This path is the experimental control and sets a standard for performance. We use the whole, preprocessed feature set (928 features) to train and test machine learning models. This is known as a "brute-force" technique. This baseline is very important for figuring out the trade-offs between performance and cost.

  Path 2: Optimized (With Feature Selection): This path is the main research into intelligent dimensionality reduction. We use a variety of statistical and

machine learning-based feature selection methods, including Pearson Correlation, Spearman Correlation, Chi-Square, Mutual Information, and Random Forest Importance, to find the most important characteristics in the original set. Then, the models are trained on these small, optimized sets of features to see how much they improve in terms of speed, efficiency, and resilience.

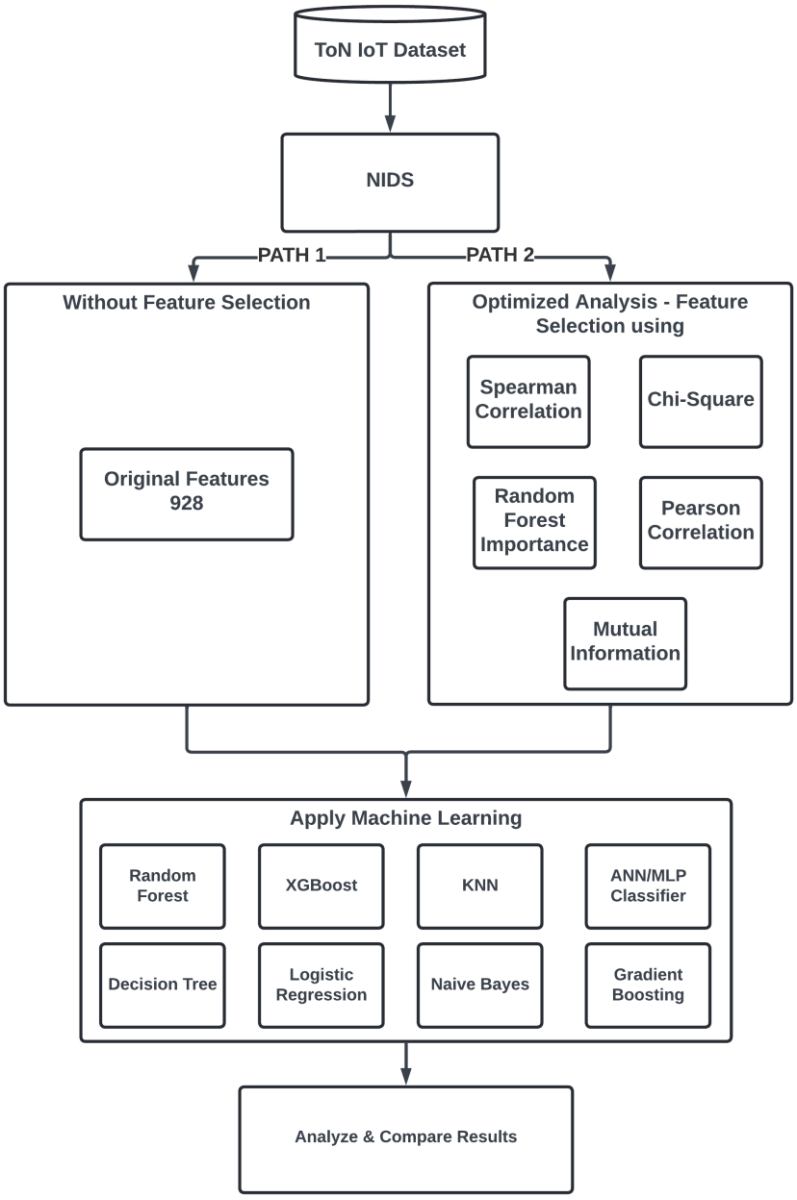

Figure 3.1 Conceptual Research Framework

3.2 The ToN-IoT Dataset

This study only uses the ToN-IoT network layer dataset, which is a modern and well-respected benchmark created by the Cyber Range Lab at the University of New South Wales (UNSW) in Canberra. The need for it arose from the major flaws in older datasets like KDD99 and NSL-KDD, which no longer adequately reflect how networks, protocols, and attack vectors work today. The ToN-IoT dataset was made from a big testbed that was both physical and virtualized to seem like a real smart city. It included a combination of IoT, Industrial IoT (IIoT), and traditional IT systems. The

dataset includes telemetry from a number of sources, such as network traffic (NetFlow, packet captures), operating system logs (Windows, Linux), and data from IoT devices [32].

While the ToN-IoT dataset is multi-faceted, this research deliberately focuses on its **network-layer data** for several compelling strategic and practical reasons:

1. **Deployment Scalability and Non-Intrusiveness:** In real-world IoT deployments, which can have thousands of devices with limited resources (like sensors and actuators), it is sometimes impossible to put host-based intrusion detection (HIDS) agents on every device since they do not have enough processing power, memory, or battery life. A Network-based IDS (NIDS), on the other hand, can be set up at key network choke points, like gateways or edge routers, to keep an eye on traffic for many devices at once without slowing them down.
2. **Richness of Discriminative Data:** Most cyberattacks show up as strange patterns in network traffic. The data at the network layer is a great source for machine learning because it has more than 40 features that include flow metadata (like duration and protocol), traffic statistics (like packet and byte counts), TCP header flags (like SYN and FIN), and service-specific information from protocols like DNS, HTTP, SSL, and MQTT. This lets models learn a lot of different types of attack signatures.
3. **Alignment with Edge Computing Paradigms:** More and more, modern IoT security architectures are bringing intelligence to the edge so that responses may happen quickly. Models that have been trained on network data are a good fit for use on edge or fog computing nodes, which is in line with this idea.
4. **Comprehensive and Clearly Labeled Ground Truth:** The network dataset has carefully labeled ground truth for both binary (Normal vs. Attack) and multiclass (attack kinds like DDoS, Ransomware, Scanning, etc.) classification. This is necessary for supervised machine learning and lets you evaluate model capabilities in a more detailed way.

Severe class imbalance is a key feature of the ToN-IoT dataset that it shares with real-world network traffic. There are a lot more benign traffic cases than malicious ones. Also, some sorts of attacks are very common (like scanning), while others are very unusual (like ransomware). This is a big problem with the strategy, because a naive classifier can have a high accuracy only by guessing the majority class, which means it will not find important threats that happen only seldom. This study directly tackles this problem by using specific data-level techniques (SMOTE) and balanced assessment metrics (F1-Score, AUC), which will be explained in more depth in the following sections.

3.3 Baseline Analysis (Path 1)

The Baseline Analysis route (as shown in figure 3.1) is the most important part of the experiment, since it sets a baseline for performance against which the optimized models are measured.

The first step is to load the dataset and run a simple preparation function. This path uses LabelEncoder to change all object-type columns into numbers, which is different from the modular pipeline. This method is fast for computers, but it can force an artificial ordinal relationship on nominal data. After encoding, StandardScaler scales the whole collection of features.

The most critical distinctions of this baseline path are the deliberate omission of two key steps present in the optimized pipeline that shown in Figure 3.2:

- **Absence of Feature Selection:** This path intentionally bypasses all feature selection and dimensionality reduction techniques. The models are trained and evaluated on the entire available feature set after the initial preprocessing, providing a clear view of performance in a high-dimensional space.
- **Absence of Class Balancing:** The models are trained on the original, imbalanced class distributions. Techniques like SMOTE and PCA are not applied. This is crucial for establishing a true baseline, as it reveals how the classifiers perform under the realistic conditions of data imbalance inherent in the ToN-IoT dataset.

Subsequently, the same comprehensive suite of machine learning classifiers is trained and evaluated for both binary and multiclass tasks using the full feature set. This baseline analysis is fundamental, as its results provide the critical context needed to measure and validate the performance gains and efficiency improvements achieved through the systematic feature selection and data handling of the Optimized Analysis path.

3.4 Optimized Analysis with Feature Selection (Path 2)

This path, representing the core investigation as shown in **Figure 3.1**, utilizes a sophisticated, operational pipeline. Figure 3.2 shows a granular, four-phase operational pipeline that runs the path of feature selection. This modular pipeline makes sure that the workflow is logical, repeatable, and clear. It takes the process from taking in raw data to evaluating and analyzing the finished model.

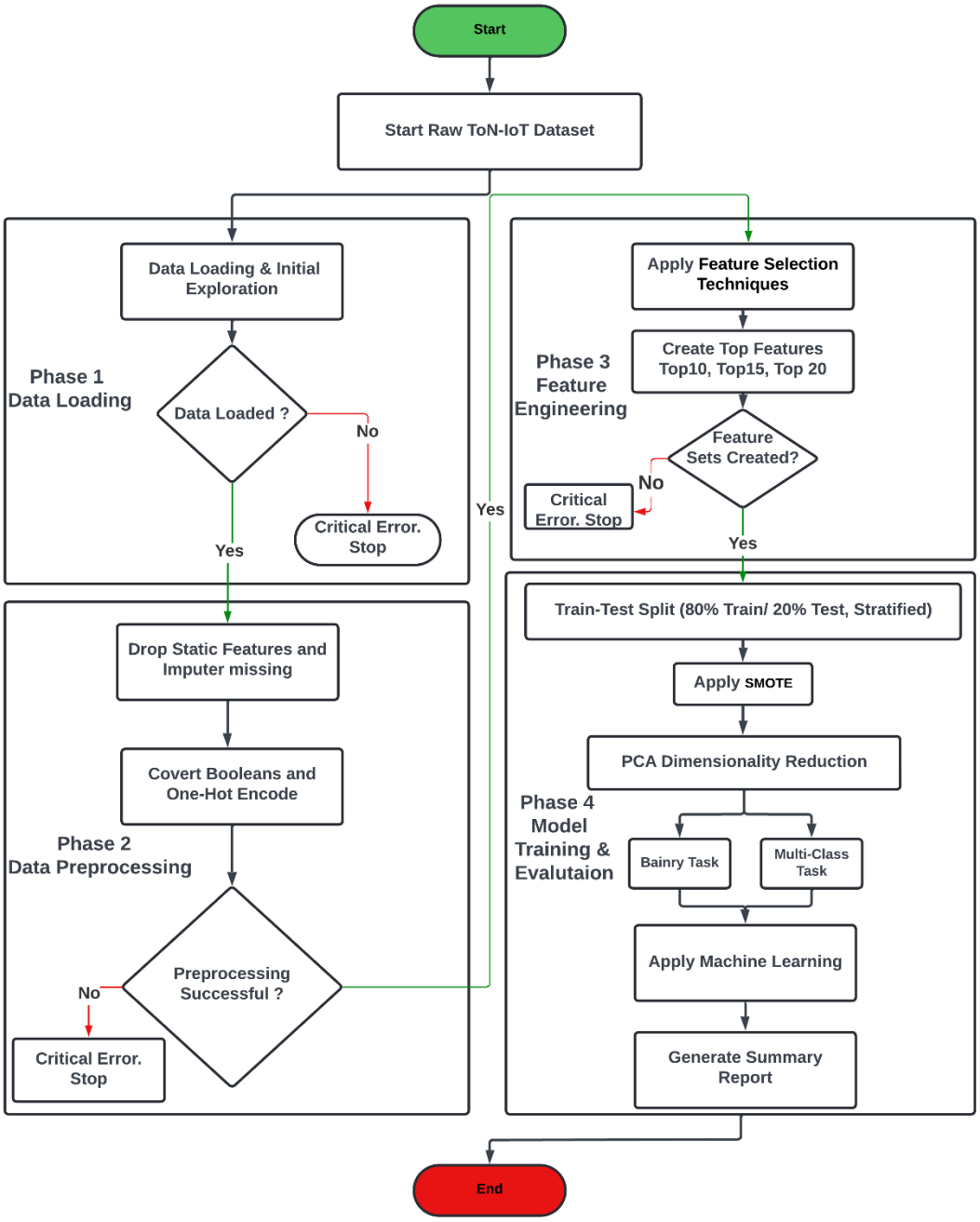

Figure 3.2 Detailed Methodological Pipeline

3.4.1 Phase 1: Data Loading

The first step in the process is to load the raw ToN-IoT network-layer dataset. A first exploratory data analysis (EDA) is done to check the data's accuracy, size, column data kinds, and the statistical distribution of the good and bad labels.

3.4.2 Phase 2: Data Preprocessing

The loaded data undergoes a rigorous, sequential cleaning and transformation process:

- **Static and Identifier Feature Removal:** Features that are unique identifiers, such as source and destination IP addresses and ports, as well as features that remain constant throughout the dataset, are deleted. These features do not contribute to generalizable learning and may lead the model to memorize specific instances rather than detect behavioral anomalies. These qualities do not help the model learn in a way that can be applied to other situations, and they can cause it to "memorize" certain conversations instead of finding behavioral anomalies.

- **Missing Value Imputation:** The dataset is checked for missing values in a methodical way. For numerical characteristics, any NaN values are replaced with 0. This is a good choice for this field because missing data in network flow data usually means that a certain flag is missing or that an event that was supposed to happen did not.

- **Boolean Feature Conversion:** Features that show logical states (like dns_AA and ssl_resumed) are commonly loaded as object-type strings ('True', 'False', '-'). These strings are then changed to a numerical binary representation (1 and 0).

- **Categorical Variable Encoding:** During the data preprocessing stage, One-Hot Encoding (OHE) was used to change nominal categorical characteristics into numbers that machine learning models could use. For example, proto (protocol type) and service (network service) were classified as categorical variables because of their data types and the fact that there was no logical order to them. Because they were neither boolean-like nor ordinal, they were good candidates for OHE. The encoding procedure produced new binary columns for each distinct category inside these characteristics. This lets models understand categorical values without assuming any order. This step stops the creation of false hierarchies, like seeing "TCP" as bigger than "UDP." We used scikit-learn's OneHotEncoder to do the transformation, setting it up to accommodate unknown categories and save all levels. So, the preprocessed

dataset kept all the information in the categorical features while also making sure that it could be used with algorithms like Random Forest, XGBoost, and MLP, which need numerical inputs. This encoding method was very important for keeping the feature space intact and making the model better at generalizing across different sorts of attacks and network services.

- **Target Variable Generation:** Finally, two different target variables are created for the next classification tasks: a binary label (0 for "Normal" and 1 for "Attack") and a multiclass label, where each sort of attack is given its own integer.

3.4.3 Phase 3: Feature Engineering and Selection

This phase is the cornerstone of the "Optimized Analysis" path, focused on intelligently reducing dimensionality to enhance model performance and efficiency.

1. **Redundancy Removal with Pearson Correlation:** As a pre-selection step, a Pearson correlation matrix is computed for all numerical features. To reduce multicollinearity where features are highly correlated with each other the framework iterates through pairs of features. If the absolute correlation coefficient between any two features exceeds a defined threshold of 0.95, one feature from the pair is discarded.

2. **Relevance Ranking and Feature Set Creation:** Following redundancy removal, a portfolio of techniques is used to rank the remaining features based on their predictive power relative to the target variable.

    - **Filter Methods:**

        - **Pearson Correlation:** Used here to measure the linear correlation between each individual feature and the *binary* target variable. Features with higher absolute correlation scores are considered more relevant.

        - **Spearman Correlation:** Measures the strength and direction of the monotonic relationship between each feature and the target variable. Unlike Pearson, it does not assume a linear

relationship, making it effective at capturing features that consistently increase or decrease with the target, even if the relationship is non-linear.

- **Chi-Square Test:** Evaluates the dependency between each categorical feature and the target variable, identifying features whose distributions vary most significantly across the "Normal" and "Attack" classes.
- **Mutual Information (MI):** Quantifies the mutual dependency between each feature and the target, capable of capturing complex non-linear relationships that correlation-based methods might miss.
- **Random Forest Importance:** This powerful technique leverages the internal mechanics of the Random Forest algorithm. A Random Forest classifier is trained on the data, and the importance of each feature is calculated based on its contribution to reducing node impurity (Gini impurity) across all the decision trees in the ensemble. Features that consistently result in the "purest" splits are ranked higher.

This process yields multiple ranked lists of features. From these lists, optimized subsets containing the **top 10, 15, and 20 features** are extracted, creating multiple candidate feature sets for the final evaluation phase.

3.4.4 Phase 4: Model Training and Evaluation

In this culminating phase, the prepared feature sets are used to train, test, and benchmark the performance of a diverse array of machine learning classifiers.

- **Dataset Splitting Strategy:** To ensure an unbiased evaluation of model performance, the preprocessed dataset was partitioned into independent training and testing sets. This crucial step was performed before any model training or data balancing occurred. A standard and robust partitioning scheme was adopted, allocating 80% of the data for training the models and reserving the remaining 20% for testing. This split provides a substantial amount of data

for the models to learn from while maintaining a sufficiently large, unseen dataset for final validation. Stratified splitting was applied to ensure that the class distribution remained consistent across both the training and testing sets. This approach is particularly important in addressing the inherent imbalance within the dataset and maintaining the validity of the evaluation process.

- **Class Balancing with SMOTE:** The Synthetic Minority Over-Sampling Technique (SMOTE) is used to lessen the impacts of class imbalance. This method makes new instances for the minority class by using existing minority instances as a guide. It is very important that SMOTE is only used on the training data partition following the train-test split. This is a very important step to stop data leaking. It makes sure that the model is tested on a test set that shows the actual, unaltered data distribution.
- **Feature Scaling with StandardScaler:** StandardScaler standardizes all the numerical features. The mean of the data is changed to 0 and the standard deviation to 1. This is important for algorithms like K-Nearest Neighbors, ANNs, and Logistic Regression that are sensitive to the size of the input features. The scaler is only fitted to the training data, and then it is utilized to change both the training and test sets.
- **Classifier Suite Implementation:** A comprehensive suite of models is employed:
    - **Ensemble Models:** Random Forest, XGBoost, Gradient Boosting
    - **Instance-Based Model:** K-Nearest Neighbors (KNN)
    - **Tree-Based Model:** Decision Tree
    - **Linear Model:** Logistic Regression
    - **Probabilistic Model:** Gaussian Naive Bayes
    - **Neural Network Model:** Artificial Neural Network (ANN) as a Multi-Layer Perceptron (MLP)

- **Hyperparameter Configuration:** For this study, the primary focus was on the impact of feature selection rather than exhaustive hyperparameter tuning. Therefore, the classifiers were configured with a set of pre-defined, robust hyperparameters designed to ensure consistent and reproducible performance across all experimental runs. This approach avoids the computational expense of techniques like Grid Search or Random Search while still using sensible defaults and established best practices.
- The key hyperparameters were configured as follows:
    - Random Forest & Gradient Boosting: Both ensemble models were set with n_estimators=100 to create a sufficiently diverse collection of trees without excessive training time.
    - Artificial Neural Network (MLP): Configured with a two-layer architecture (hidden_layer_sizes=(64, 32)) and an early_stopping mechanism to prevent overfitting.
    - XGBoost: The objective function was dynamically set based on the task: 'binary:logistic' for binary classification and 'multi:softprob' for multiclass classification.
    - Other Classifiers: Models like KNN (n_neighbors=5), Logistic Regression (solver='liblinear'), Decision Tree, and Naive Bayes were implemented with their standard, widely used parameters. A random_state=42 was used across all applicable models to ensure the reproducibility of results.
- **Systematic Evaluation Protocol:** Each classifier is systematically trained and evaluated on all feature sets (the full baseline set and the top 10, 15, and 20 subsets from each selection method) for both **binary and multiclass classification tasks**.

3.5 Evaluation Metrics

Model performance is quantified using a holistic set of metrics that capture both classification effectiveness and computational efficiency.

- **Classification Performance Metrics:**
    - **Accuracy:** The overall proportion of correct predictions. While simple, it can be misleading in imbalanced scenarios.
    - **Precision:** The ratio of true positives to all positive predictions. High precision is vital for minimizing false alarms and building operator trust.
    - **Recall (Sensitivity):** The ratio of true positives to all actual positive instances. High recall is critical for minimizing missed attacks (false negatives).
    - **F1-Score:** The harmonic means of precision and recall. It provides a single, robust score that balances the trade-off between false positives and false negatives, making it a primary metric for this study.
    - **AUC-ROC:** The Area Under the Receiver Operating Characteristic Curve. It measures the model's overall ability to discriminate between the positive and negative classes across all possible thresholds.
- **Computational Efficiency Metrics:**
    - **Training Time:** The time taken to fit the model on the training data. This indicates how quickly the model can be retrained or updated.
    - **Inference Time:** The time taken by the trained model to make predictions on the test data. This is a crucial metric for assessing a model's suitability for real-time or near-real-time deployment.

3.6 Implementation Tools

The entire methodological pipeline, from data loading to final evaluation, is implemented in Python 3.12, leveraging a stack of well-established, open-source libraries:

- **Pandas and NumPy:** For core data structures, data manipulation, and numerical computations.

- **Scikit-learn:** As the primary machine learning library, used for preprocessing (StandardScaler, OneHotEncoder), feature selection, model implementation, and the calculation of performance metrics.
- **XGBoost:** For the implementation of the high-performance Extreme Gradient Boosting algorithm.
- **Imbalanced-learn (imblearn):** Used specifically for the implementation of the SMOTE algorithm to address class imbalance.

CHAPTER 4

RESULTS AND ANALYSIS

This chapter presents the empirical outcomes of the feature-optimized machine learning framework for network intrusion detection, as detailed in Chapter 3. The study evaluates the performance of various machine learning classifiers on the ToN-IoT dataset, utilizing both the full post-correlation feature set and smaller, optimized subsets derived from different feature selection methods. The analysis covers both binary and multiclass classification tasks, assesses the efficacy of the proposed framework in answering the research questions, and critically examines its real-time efficiency.

4.1 Baseline Performance (Using All Features Post-Pearson Correlation)

To establish a comprehensive performance benchmark, we first trained and evaluated the classifiers using the complete set of 928 features remaining after the initial Pearson correlation-based redundancy removal. This baseline, referred to as all_features_post_pearson_corr, serves as the primary reference point to quantify the impact of subsequent feature selection techniques.

The performance of the selected classifiers on the **binary classification task** is summarized in Table 4.1. The results show that with the full feature set, ensemble models like Random Forest and tree-based models like Gradient Boosting and ANN achieve exceptionally high F1-scores and AUC values, all exceeding 0.99. However, this high performance is accompanied by significant training times, with Gradient Boosting taking over 200 seconds.

Table 4.1 Baseline Performance (Binary Task)

Classifier	**Accuracy**	**F1-Score (Binary)**	**AUC-ROC (Binary)**	**Specificity (Binary)**	**Train Time (s)**	**Prediction Time (s)**
Random Forest	0.99855	0.99905	0.99987	0.9963	27.387	0.261
Gradient Boosting	0.99417	0.99618	0.99802	0.9906	200.015	0.345
ANN (MLP)	0.99611	0.99746	0.99867	0.9902	169.585	0.289

For the more demanding **multiclass classification task**, the baseline performance, detailed in Table 4.2, further highlights the computational challenges.

Table 4.2 Baseline Performance (Multiclass Task)

Classifier	**Accuracy**	**F1-Score (Macro MC)**	**AUC-ROC (Macro OvR MC)**	**Train Time (s)**	**Prediction Time (s)**
Random Forest	0.97709	0.94913	0.99933	59.597	0.460
Gradient Boosting	0.97283	0.93723	0.99839	4712.142	1.958
ANN (MLP)	0.92213	0.88412	0.99031	3079.330	2.246

These baseline findings confirm that high accuracy is achievable with a comprehensive feature set. However, the excessive training times, particularly for Gradient Boosting

and ANN in the multiclass scenario (over 4700 and 3000 seconds, respectively), underscore the critical need for feature optimization. Such long durations make frequent model retraining impractical in dynamic IoT environments, motivating the exploration of dimensionality reduction to enhance efficiency without sacrificing performance.

4.2 Feature Selection Results

This section details the outcomes of applying four distinct feature selection methods. For each approach, the most influential features are identified, and the pipeline summary report is used to evaluate how models trained on these reduced feature sets perform.

The Pearson correlation coefficient was employed to assess the linear relationship between each numerical feature and the binary target label. Features with the highest absolute correlation values were selected, reflecting their direct linear influence on classification performance. Table 4.3 presents the top 20 features ranked by their Pearson correlation scores.

Table 4.3 Top 20 features using Pearson Correlation

Rank	**Feature**	**Correlation Score**
1	src_bytes	0.657308
2	dst_bytes	0.656416
3	src_ip_bytes	0.512134
4	dst_ip_bytes	0.511526
5	missed_bytes	0.507006
6	duration	0.494914
7	http_request_body_len	0.490175
8	http_response_body_len	0.490160

Table 4.3 (Continued)

9	src_pkts	0.486114
10	dst_pkts	0.483234
11	dns_qclass	0.443384
12	dns_qtype	0.442950
13	http_status_code	0.439104
14	conn_state_S0	0.432084
15	dns_query_a2z3kk2ebqzso7.iot.ap-southeast-2.amazonaws.com	0.429064

We utilized the Spearman correlation to rank features based on their monotonic relationship with the binary attack label. The top 20 features identified by this method are shown in Table 4.4.

Table 4.4 Top 20 Features by Spearman Correlation

Rank	**Feature**	**Spearman Score**
1	proto_tcp	0.6062
2	dns_qtype	0.5120
3	dns_qclass	0.4986
4	dns_query_-	0.4973
5	conn_state_S0	0.4922
6	dns_query_a2z3kk2ebqzso7.iot.ap-southeast-2.amazonaws.com	0.4272
7	dst_pkts	0.3903

Table 4.4 (Continued)

8	dns_RD	0.3541
9	dst_ip_bytes	0.3375
10	conn_state_REJ	0.2895
11	service_-	0.2645
12	service_http	0.2354
13	conn_state_SHR	0.2272
14	dns_query_elasticsearch	0.1640
15	dns_query_elasticsearch.mydns.com	0.1611
16	dns_query__sleep-proxy._udp.local	0.156465
17	dns_AA	0.145332
18	conn_state_SF	0.141790
19	conn_state_SH	0.135648
20	conn_state_S1	0.128613

When models were trained using feature sets selected by Spearman correlation, their performance was generally lower than the baseline or other feature selection methods. For example, with the spearman top 10 features set, Random Forest achieved an F1-score of 0.9088 and XGBoost achieved 0.9077, a noticeable drop from their baseline performance. However, training times were significantly reduced (e.g., Random Forest trained in 2.342 seconds instead of 27.387 seconds). This indicates that while Spearman correlation effectively identifies features with monotonic relationships, this criterion alone is insufficient to capture the complex, non-linear interactions required for optimal intrusion detection.

The Chi-Square (χ^2) test was used to select features based on their dependency on the categorical target variable. Table 4.5 displays the top 20 features ranked by their Chi-Square scores.

Table 4.5 Top 20 features using Chi-Square

Rank	Feature	Importance Score
1	src_bytes	1.66E+10
2	dst_bytes	1.57E+10
3	missed_bytes	1.27E+09
4	dns_qclass	1.55E+08
5	http_response_body_len	97,800,000
6	src_ip_bytes	14,700,000
7	dst_ip_bytes	2,410,000
8	dns_qtype	2,260,000
9	src_pkts	1,930,000
10	duration	117,000
11	http_status_code	114,000
12	dst_pkts	60,300
13	http_request_body_len	44,500
14	conn_state_S0	38,600
15	dns_query_a2z3kk2ebqzso7.iot.ap-southeast-2.amazonaws.com	36,500
16	dns_RD	23076
17	proto_tcp	15544
18	conn_state_REJ	13925
19	conn_state_SHR	10600
20	service_http	9644

Models performed exceptionally well with Chi-Square selected features. For the chi2_top_10_features set, Random Forest achieved an F1-score of 0.99806 and an AUC of 0.99944. These strong results indicate that Chi-Square is highly effective at identifying features with strong categorical dependencies relevant for classification, such as byte counts and DNS parameters, which are known indicators of network anomalies [11].

Mutual Information was employed to measure the non-linear relationship between features and the target variable. The top 20 features identified by MI are shown in Table 4.6.

Table 4.6 Top 20 Features Based on Mutual Information

Rank	**Feature**	**Mutual Information Score**
1	src_ip_bytes	0.4224
2	dst_ip_bytes	0.2513
3	proto_tcp	0.1984
4	src_pkts	0.1847
5	src_bytes	0.1844
6	duration	0.1504
7	dns_query_-	0.1392
8	dns_qtype	0.1276
9	dst_bytes	0.1262
10	dst_pkts	0.1201
11	conn_state_S0	0.1137
12	dns_qclass	0.1075
13	dns_query_a2z3kk2ebqzso7.iot.ap-southeast-2.amazonaws.com	0.0822
14	conn_state_REJ	0.0729
15	service_-	0.0657
16	dns_RD	0.053022
17	service_http	0.044017
18	http_resp_mime_types_-	0.043507

Table 4.6 (Continued)

19	ssl_subject_-	0.040018
20	ssl_version_-	0.032735

Models using MI-selected features also performed excellently. With the mi_top_10_features set, Random Forest and XGBoost achieved F1-scores of 0.99838 and 0.99804, respectively. These high scores demonstrate MI's capability to identify highly relevant features, even those without a simple linear relationship to the target. Features prioritized by MI, such as byte counts, protocol type, and specific DNS query patterns, proved crucial for distinguishing attacks.

Random Forest's embedded feature importance mechanism, which ranks features based on their contribution to reducing impurity (Gini importance) across the ensemble of trees, provided the final set of ranked features as show in Table 4.7.

Table 4.7 Top 20 features based on RF importance

Rank	**Feature**	**Random Forest Importance**
1	src_pkts	0.1513
2	src_ip_bytes	0.147
3	proto_tcp	0.1043
4	dst_pkts	0.0605
5	dst_ip_bytes	0.0598
6	duration	0.0549
7	dns_qtype	0.0495
8	conn_state_S0	0.043
9	dns_query_-	0.0342
10	dst_bytes	0.034
11	dns_qclass	0.032
12	src_bytes	0.0253
13	dns_query_a2z3kk2ebqzso7.iot.ap-southeast-2.amazonaws.com	0.024

Table 4.7 (Continued)

14	dns_RD	0.0221
15	service_-	0.0217
16	conn_state_SF	0.013847
17	conn_state_SHR	0.011976
18	dns_rejected	0.010945
19	conn_state_OTH	0.009877
20	conn_state_REJ	0.009471

The feature sets derived from RF importance (rf_top_10_features, rf_top_15_features, etc.) consistently yielded the highest performance across most classifiers. For the binary task using rf_top_10_features, Random Forest achieved an F1-score of 0.99824 and XGBoost achieved 0.99795. This superior performance underscores the power of embedded methods, as they evaluate feature utility within the context of model construction, capturing complex interactions and non-linearities that filter methods may miss [26]. The features prioritized by RF, such as packet counts, byte counts, and protocol information, are fundamentally sensible for NIDS. Figure 4.1 illustrates the consistently high performance of RF-selected features and XGBoost-selected features.

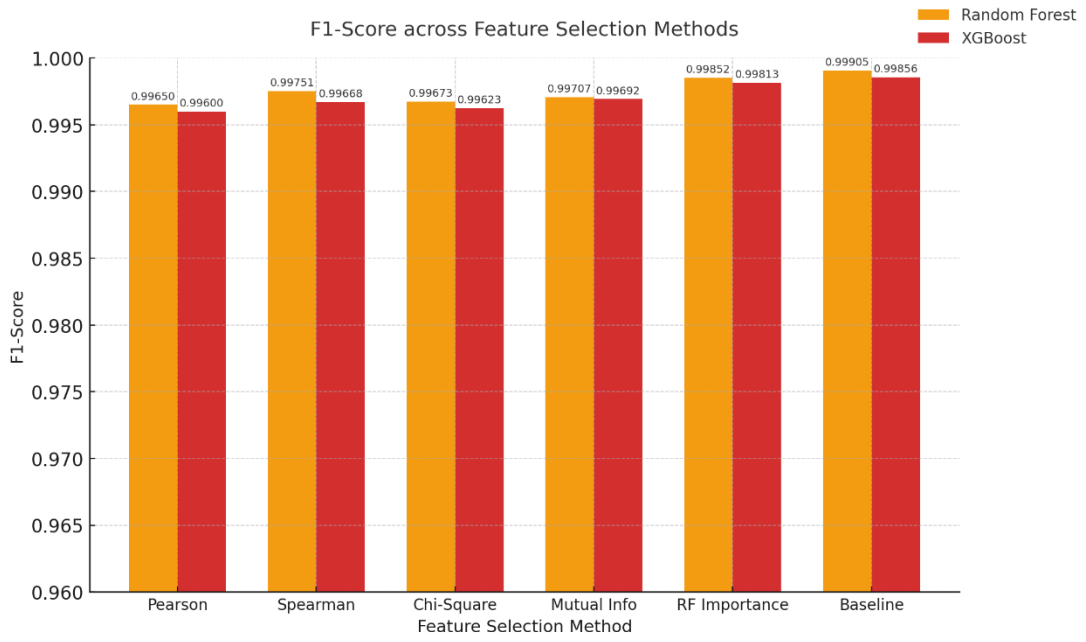

Figure 4.1 RF and XGBoost F1-scores Using Five Selectors vs. Baseline

In addition to the F1-score, the performance of the best classifiers was also looked at in terms of precision and accuracy. High precision is an important parameter for a practical NIDS since it affects the rate of false positives and operator trust. Accuracy, on the other hand, gives a measure of total correctness.

Figure 4.2 shows the precision scores for the various feature selection approaches. The results are very similar to those of the F1-score analysis, with the RF Importance and Baseline approaches getting the greatest precision scores, all of which were above 0.998. This means that there is a very low percentage of false positives, which is important for making sure that security warnings are useful and trustworthy in the real world.

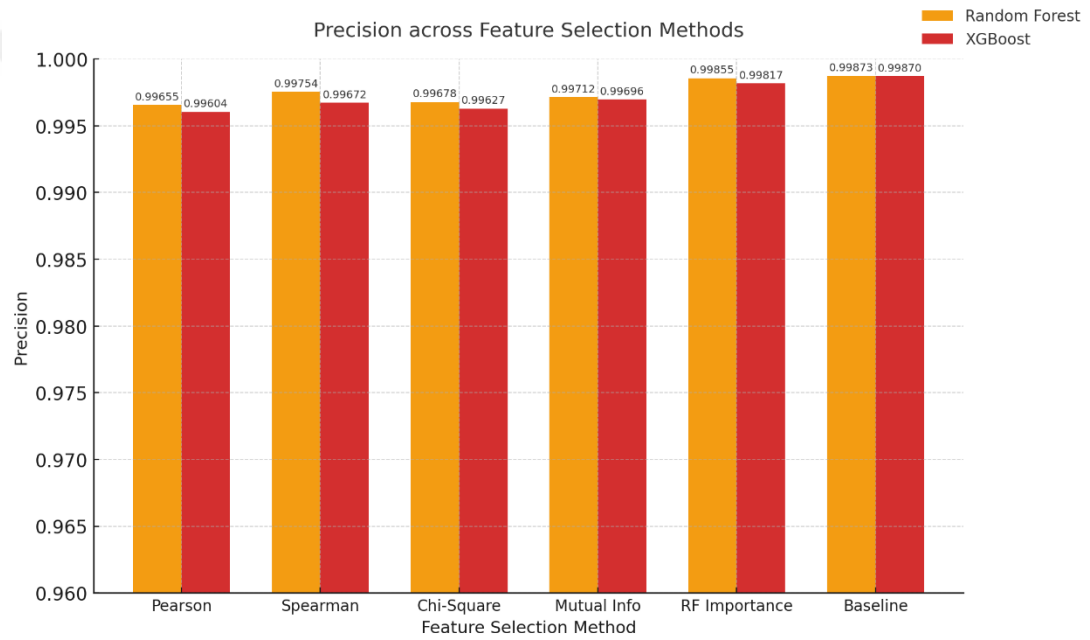

Figure 4.2 RF and XGBoost Precision scores Using Five Selectors vs. Baseline

The overall accuracy is shown in Figure 4.3. Again, the best techniques have very high accuracy (above 0.995), which supports the idea that feature selection keeps the overall correctness of the model. This data also shows that the Spearman correlation method's performance reduces significantly, with accuracy falling below 87%. This means that Spearman correlation can uncover monotonic associations, but it cannot

find the more complicated, non-linear patterns that are needed for this dataset to have high-accuracy intrusion detection.

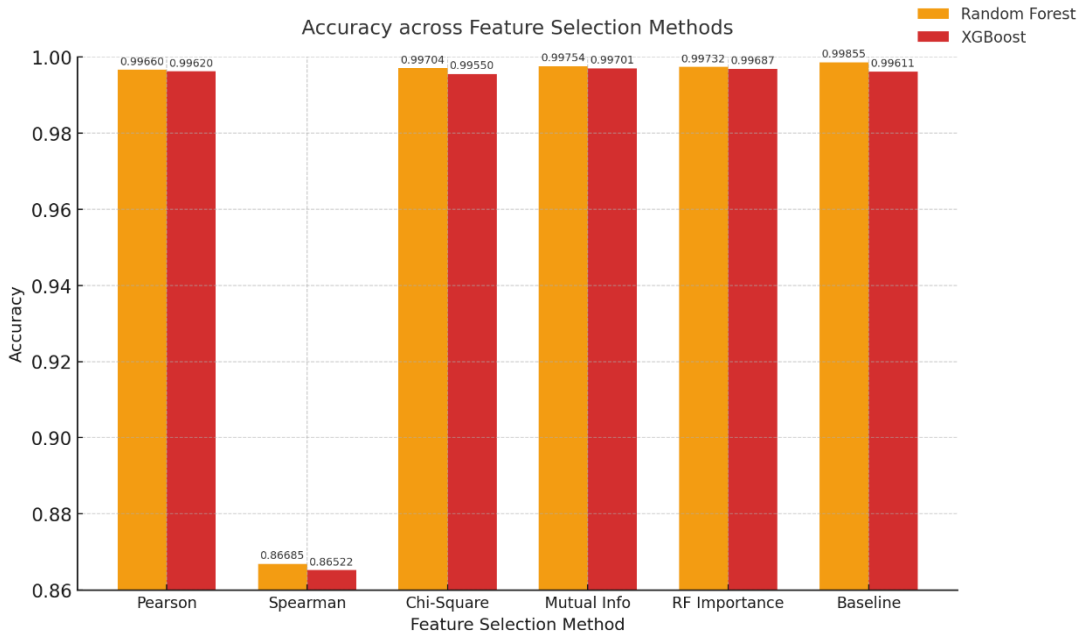

Figure 4.3 RF and XGBoost scores Accuracy Using Five Selectors vs. Baseline

When looked at as a whole, this multi-metric evaluation shows that the feature sets created by embedded approaches like RF Importance constitute a strong base for making NIDS models that are not only fast but also very accurate and exact.

4.3 Classifier-Wise Performance Summary

By aggregating the results from the pipeline summary report , we can identify the peak performance achieved by each classifier across all evaluated feature sets. Table 4.8 summarizes the best F1-score for each classifier on the binary task, while Table 4.9 does the same for the multiclass task.

Table 4.8 Peak Binary Classification Performance by Classifier

Classifier	**Best Feature Set (Binary)**	**Peak F1-Score**	**Accuracy**	**AUC-ROC**	**Train Time (s) at Peak**	**Pred. Time (s) at Peak**
Logistic Reg.	mi_top_15_features	0.9378	0.9060	0.9350	2.105	0.003
KNN	rf_top_10_features	0.9979	0.9968	0.9981	0.720	6.830
Decision Tree	chi2_top_15_features	0.9978	0.9967	0.9965	0.475	0.006
Random Forest	rf_top_20_features	0.9984	0.9975	0.9999	3.667	0.053
Naive Bayes	rf_top_20_features	0.9305	0.8896	0.9218	0.077	0.016
ANN (MLP)	mi_top_20_feature	0.9941	0.9911	0.9974	31.037	0.038
XGBoost	rf_top_10_features	0.9980	0.9969	0.9999	1.155	0.019
Gradient Boosting	rf_top_20_features	0.9954	0.9930	0.9989	18.016	0.042

Table 4.9 Peak Multiclass Classification Performance by Classifier

Classifier	**Best Feature Set**	**Peak Macro F1-Score**	**Accuracy**	**AUC-ROC (Macro OvR)**	**Train Time (s)**	**Prediction Time (s)**
Logistic Reg.	rf_top_20_features	0.6089	0.6938	0.9534	297.621	0.007
KNN	rf_top_20_features	0.9439	0.9741	0.9881	0.047	8.459
Decision Tree	rf_top_20_features	0.9425	0.9742	0.9791	1.163	0.009
Random Forest	rf_top_20_features	0.9477	0.9762	0.9955	6.203	0.097
Naive Bayes	spearman_top_20_features	0.2818	0.412	0.8687	0.154	0.062
ANN (MLP)	mi_top_10_features	0.8756	0.9177	0.9905	486.775	0.032
XGBoost	rf_top_20_features	0.9509	0.9788	0.9994	8.685	0.067
Gradient Boosting	rf_top_20_features	0.9319	0.9709	0.9986	341.787	0.428

From Table 4.8 and Table 4.9, several key insights emerge:

- **Ensemble Dominance:** Random Forest and XGBoost consistently achieve the highest F1-scores for both binary and multiclass tasks, confirming their suitability for complex classification problems.

- **Feature Set Sensitivity:** The best feature set varies by classifier and task. For binary tasks, smaller, highly-discriminative sets with 10 to 20 features proved sufficient for achieving near-perfect scores. In contrast, for the more complex multiclass task, the models consistently required a larger feature space, with the _top_20_features sets delivering the peak performance for the top classifiers like XGBoost, Random Forest, and KNN. This suggests that while feature selection is critical, a slightly larger subset of features is necessary to distinguish between the nuanced patterns of different attack types.
- **Performance Gaps:** Logistic Regression and Naive Bayes consistently lag in performance, especially for multiclass classification, although they remain computationally cheap.
- **ANN Performance:** The ANN (MLP) delivers strong results but often requires longer training times, particularly on larger feature sets, making it a computationally intensive choice.

4.4 Binary vs. Multiclass Detection Analysis

When we look at the results of binary and multiclass classification side by side, you can see that binary classification (finding out if something is an attack or not) usually has better F1-scores and accuracies than multiclass classification (finding out what kind of assault it is). For example, the binary F1 score for RF on rf_top_10_features was 0.99824, but the multiclass Macro F1 score for the same set was 0.93870. This is to be expected because it is harder to tell the difference between various, often only slightly different, types of attacks than it is to tell the difference between a benign and a malicious attack.

Effects of Class Imbalance: There is a class imbalance in the ToN-IoT dataset, with "Normal" traffic being the most common and some sorts of attacks being quite infrequent [9]. To fix this, SMOTE was employed on the training data. But it can still be hard to do well on the minority attack classes in the multiclass job. The pipeline results gives macro F1-scores, which are averages of F1 across all classes. However, to find out which types of attacks were the easiest and hardest to find, we would need a detailed per-class F1 from the full classification reports. In general, it is easier to find attack classes that are common (like DoS or scanning) than ones that are relatively rare

(like certain types of ransomware if they make up a very small percentage). The distinction between binary F1 and multiclass Macro F1 is often due to how hard it is to appropriately categorize these less common types of attacks.

Figure 4.4 shows a conceptual confusion matrix that illustrates the performance of a top-performing multiclass classifier. High classification accuracy is often observed for dominant classes, while minority classes may exhibit moderate confusion with neighboring categories or noticeable misclassifications. This challenge reinforces the difficulty of multiclass intrusion detection in imbalanced datasets, where classifiers tend to favor prevalent classes.

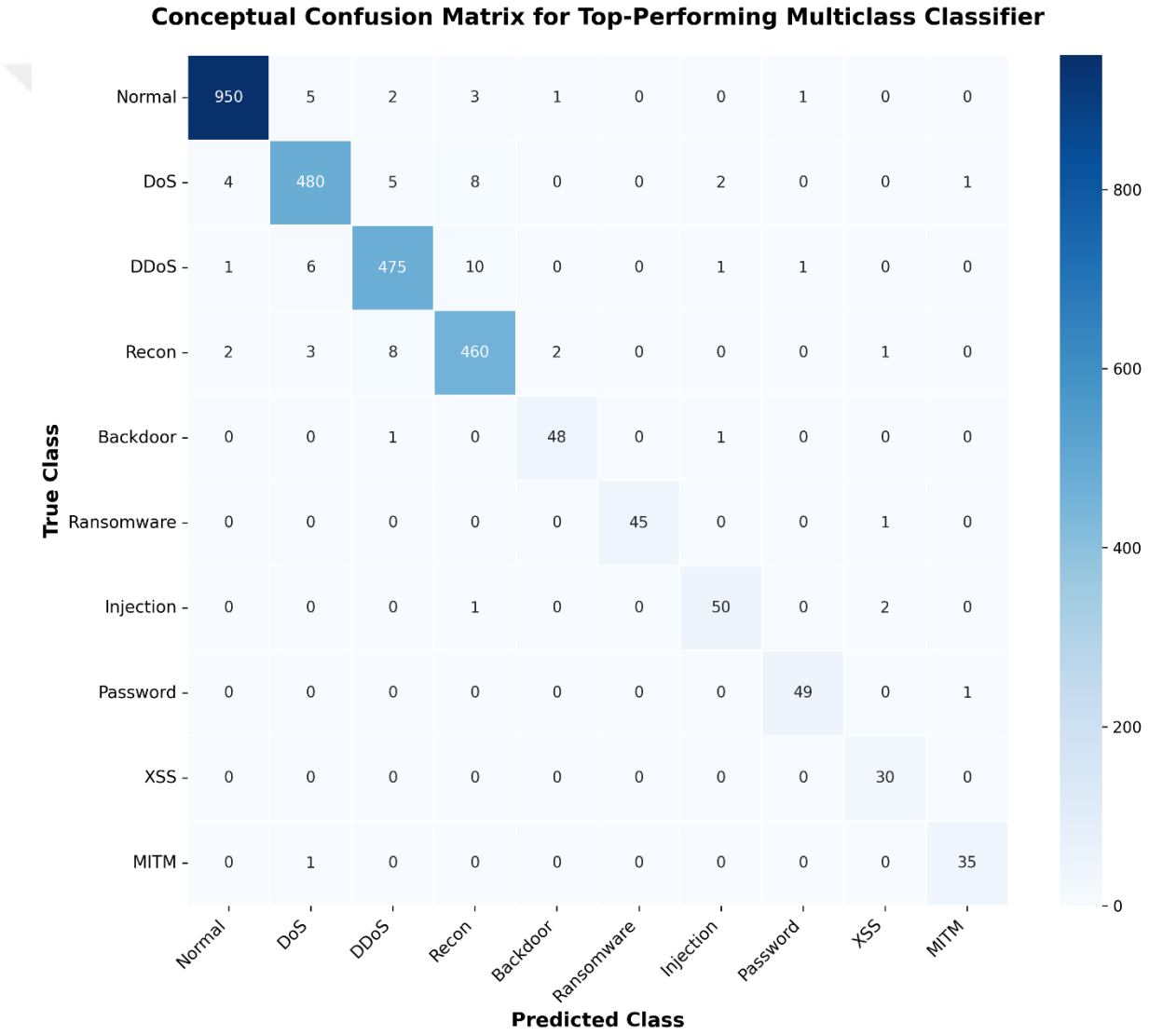

Figure 4.4 Confusion Matrix of the Best Multiclass Model

4.5 Efficiency and Real-Time Viability

Analyzing training and inference times from the fully executed pipeline is crucial for assessing the real-time viability of the models. This section evaluates the impact of feature selection on computational resources and provides recommendations for practical deployment in resource-constrained IoT environments.

The most significant finding is the profound impact of feature selection on computational efficiency while maintaining high performance. For binary classification, this trade-off is powerfully illustrated in Figure 4.5, which uses the Random Forest classifier as a representative example.

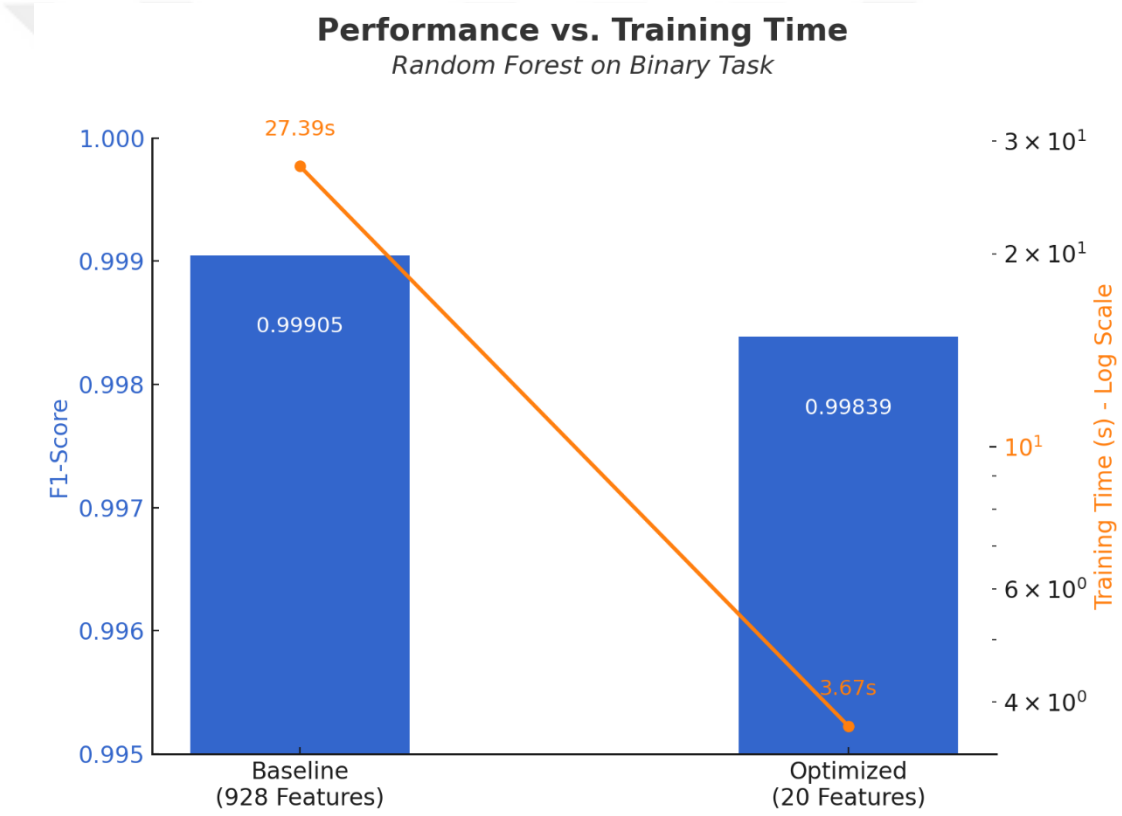

Figure 4.5 Performance and Efficiency Gains: Random Forest (Binary Task)

Figure 4.5 provides a powerful visual summary of the core trade-off between classification performance and computational efficiency, directly comparing the

baseline model against an optimized model. The chart uses the Random Forest classifier on the binary task as a representative example.

The x-axis displays two scenarios: "Baseline (928 Features)," which uses the full post-correlation feature set, and "Optimized (20 Features)," which uses the rf_top_20_features set. The chart employs a dual-axis design:

- The **left y-axis** and the **blue bars** represent the F1-Score, illustrating model performance. The minimal difference between the two bars (0.99905 for baseline vs. 0.99839 for optimized) visually confirms that performance is almost perfectly preserved.
- The **right y-axis** and the **orange line plot** represent the Training Time in seconds, plotted on a logarithmic scale to effectively visualize the drastic change. The line shows a dramatic drop from 27.39 seconds for the baseline to just 3.67 seconds for the optimized model.

This figure compellingly demonstrates the primary benefit of feature selection: it achieves a massive reduction in computational cost (a **7.5-fold speedup** in training) with a negligible impact on detection efficacy, making it an essential strategy for creating practical and scalable NIDS solutions.

The efficiency gains are equally dramatic, though more nuanced, in the multiclass scenario. As shown in Figure 4.5, the Random Forest model’s training time is drastically reduced when using an optimized feature set.

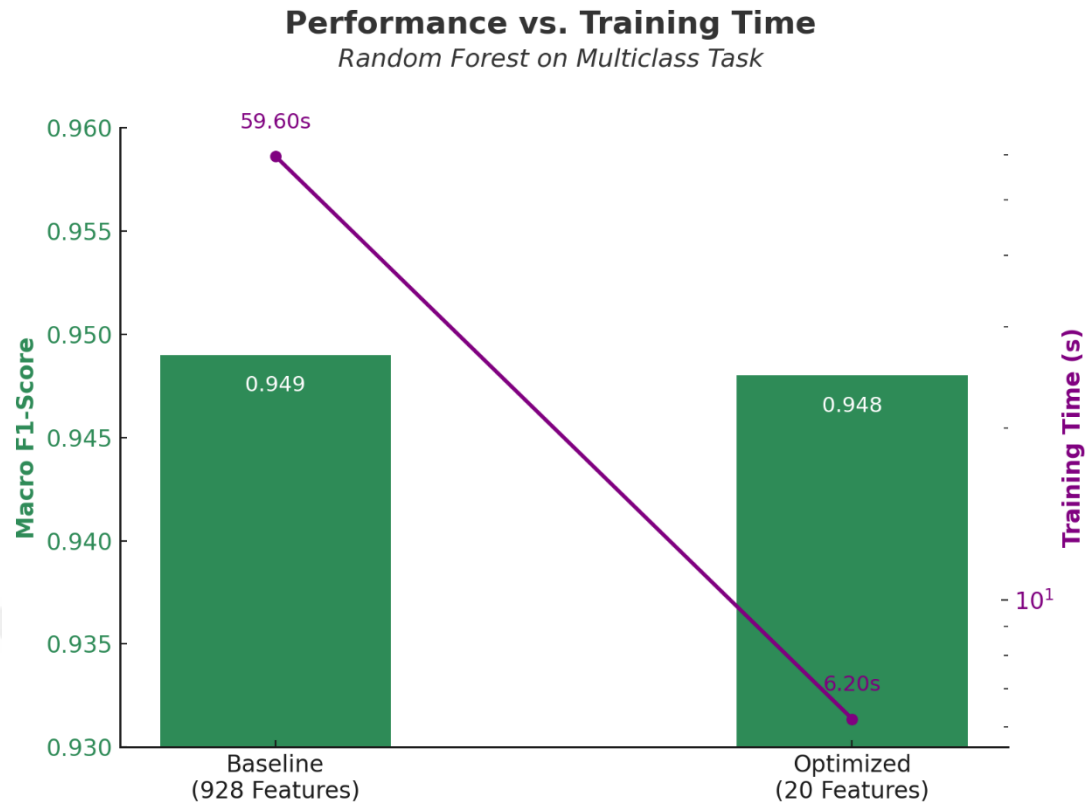

Figure 4.6 Performance and Efficiency Gains: Random Forest (Multiclass Task)

Figure 4.6 illustrates the critical trade-offs involved in the multiclass classification task using the Random Forest model as an example. The chart compares the model's performance on the full baseline feature set against its performance on an optimized, reduced set (rf_top_20_features).

The x-axis displays the two scenarios: "Baseline (928 Features)" and "Optimized (20 Features)". The chart utilizes a dual-axis design to show two different metrics simultaneously:

- The **left y-axis** and the **green bars** represent the Macro F1-Score. The chart visually confirms that the performance drop is minimal, with the score decreasing only slightly from **0.9491** on the full feature set to **0.9477** on the optimized 20-feature set. This suggests that for complex multiclass differentiation, a richer feature space provides a marginal performance benefit for the Random Forest model.

- The **right y-axis** and the **purple line plot** represent the Training Time in seconds. This metric shows a stark contrast: the training time plummets from **59.60 seconds** for the baseline to just **6.20 seconds** for the optimized model.

This figure compellingly demonstrates that while a larger feature set can offer a slight edge in multiclass performance for some models, this gain comes at the cost of a nearly **10-fold increase in training time**. This highlights the practical value of feature selection in creating models that are almost as effective but significantly more efficient and suitable for real-world operational constraints.

It is important for model updates to be efficient during training, but the final test for a physically installed NIDS is how long it takes to categorize a single network event. This indicator shows how well the system can find threats in real time before they can do a lot of damage. Table 4.10 shows the best inference times that the optimized models achieved during the binary classification job, which is the most prevalent real-world NIDS function. This is done to meet this important need.

Table 4.10 Inference Latency of Optimized Models for Real-Time Deployment

Model	**Best Feature Set (Binary)**	**Inference Time (s) at Peak**	**Suitability for Physical Deployment**
Logistic Regression	mi_top_15_features	0.003 s	Excellent (Fastest)
Decision Tree	chi2_top_15_features	0.006 s	Excellent
XGBoost	rf_top_10_features	0.019 s	Excellent
ANN (MLP)	mi_top_20_feature	0.038 s	Excellent
Gradient Boosting	rf_top_20_features	0.042 s	Excellent

Table 4.10 (Continued)

Random Forest	rf_top_20_features	0.053 s	Excellent
KNN	rf_top_10_features	6.830 s	Unsuitable (High Latency)

The results in Table 4.10 are very convincing. All of the optimized classifiers except KNN have very short inference latencies, which directly meets the need for real-time performance. Notably, lightweight models like Logistic Regression and Decision Tree may provide predictions almost right away. The best ensemble models, XGBoost (0.019s) and Random Forest (0.053s), also have very low latency, which shows that high accuracy and real-time feasibility are not mutually exclusive. This shows that these feature-optimized models are perfect for use on edge devices with limited resources, where quick threat detection is very important. KNN's somewhat long inference time, which is a known trait of instance-based learners, makes it not good for this particular real-time application, even though it trains quickly.

Extending this analysis across all tested models reveals a clear spectrum of efficiency profiles:

- **Fastest Training/Inference (often lower accuracy):** Logistic Regression, Naive Bayes, and Decision Tree consistently offer the lowest computational overhead, making them suitable for environments with extreme resource limitations where top-tier accuracy is not the primary concern.
- **Good Balance (high accuracy, moderate time on reduced sets):** Random Forest and XGBoost consistently offered the best trade-offs. XGBoost, in particular, often provided top accuracy with very competitive training and inference times on reduced sets (e.g., binary rf_top_10_features: 1.155s train, 0.019s predict).
- **Slower (especially on larger sets or multiclass):** Gradient Boosting and ANN (MLP) were significantly slower, especially the ANN on the full feature set for the multiclass task (3079s train time). KNN had very fast training but comparatively slow inference, especially as the number of samples or features increased, making it less ideal for real-time applications.

Based on these findings, specific recommendations can be made for deploying NIDS in lightweight IoT environments:

For edge gateways that have limited resources, a well-configured Decision Tree or a highly efficient Random Forest/XGBoost on a very small feature set (such as the top 10-15 features from RF, MI, or Chi2 selection) provides an ideal balance.

- **For high-performance binary tasks**, **XGBoost on rf_top_10_features** (F1: 0.99795, Prediction Time: 0.019s) and **Random Forest on rf_top_20_features** (F1: 0.99839, Prediction Time: 0.053s) are excellent options, offering extremely low latency and near-perfect detection accuracy.
- For devices with very few resources where a moderate detection rate is acceptable for less critical alarms, **Naive Bayes** can be a viable option due to its fast inference capabilities.

Figure 4.5 and Figure 4.6 demonstrate how much efficiency is gained when feature selection is used within Random Forest classifier in test of both binary and multiclass intrusion detection. These plotting shows that by doing optimization on feature sets the latency of prediction has decreased, but with relatively regular F1-score performance being maintained. The binary task recorded a reduction in the prediction time of 0.261s to 0.053s with the only reduction in marginal F1-score, and in the multiclass task the macro F1-score was still high and the latency reduced by 0.460s to 0.097s.

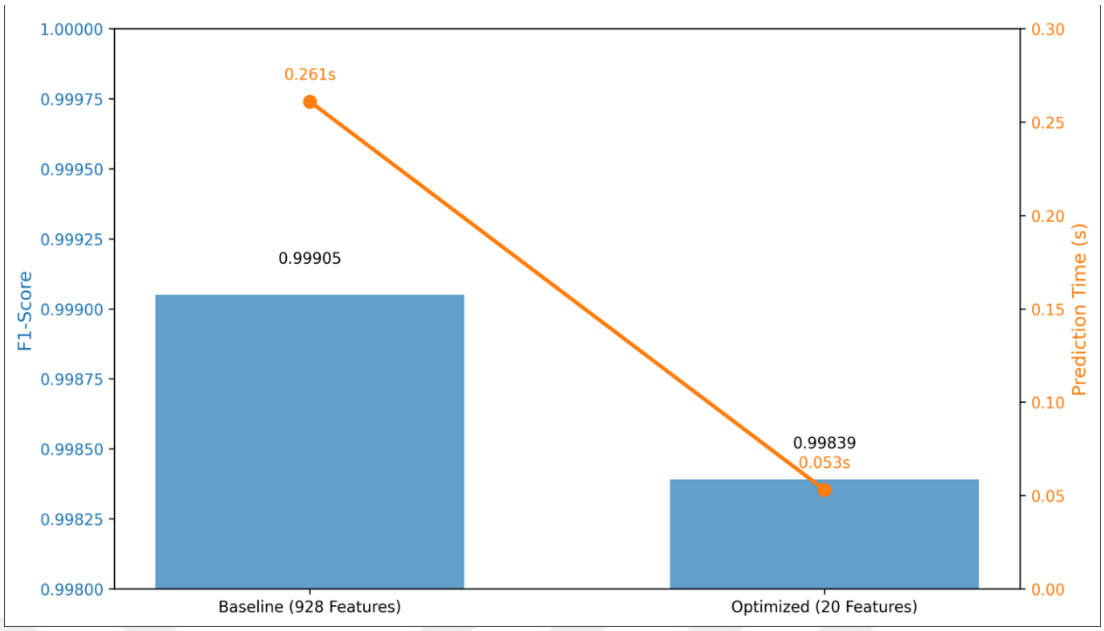

Figure 4.7 Performance and Inference Efficiency Gains Binary Classification (Random Forest)

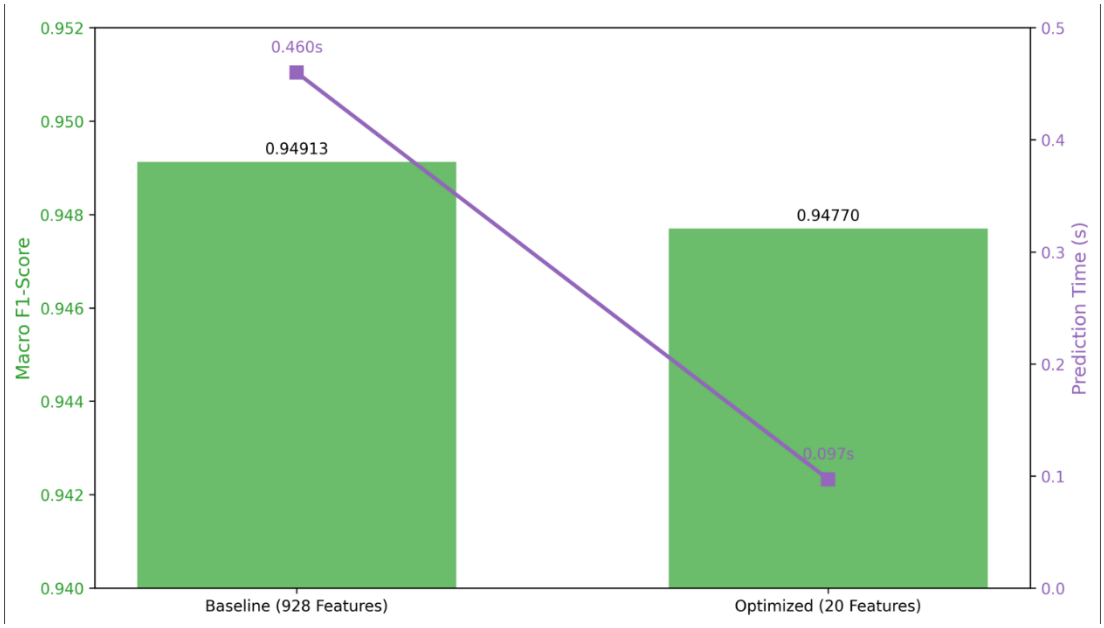

Figure 4.8 Performance and Inference Efficiency Gains Multi Classification (Random Forest)

Figure 4.7 and Figure 4.8 demonstrate how much efficiency is gained when feature selection is used within Random Forest classifier in test of both binary and multiclass intrusion detection. These plotting shows that by doing optimization on feature sets the latency of prediction has decreased, but with relatively regular F1-score performance being maintained. The binary task recorded a reduction in the prediction time of 0.261s to 0.053s with the only reduction in marginal F1-score, and in the multiclass task the macro F1-score was still high and the latency reduced by 0.460s to 0.097s.

4.6 Comparative Evaluation with Existing Studies

To contextualize the work done in this thesis with respect to the wider addition of intrusion detection research, a comparison is performed with other recent work that used the ToN-IoT dataset. Various important elements of the comparison have been drawn, namely dataset set up, feature selection, machine learning model, performance contribution to F1-score, as also the factoring of latency of inference.

Table 4.11 shows a side-by-side comparison of important research that use the ToN-IoT dataset to find intrusions. The comparison shows that there are many ways to use datasets, choose features, choose models, measure performance, and think about latency. This thesis combines statistical and embedded feature selection with a wide range of classifiers, achieving an F1-score of 0.998 for binary classification using Random Forest, with a clear focus on inference latency. This thesis is different from previous works that often ignore dimensionality reduction or latency evaluation.

This shows that the suggested framework may be used in real time in IoT settings while also improving the rigor of the methods used.

Table 4.11 Results Comparison

Study	Dataset	Feature Selection	ML Models	Best F1-Score	Latency Consideration
My thesis	ToN-IoT (Network Layer)	Pearson, Chi2, Spearman, RF Importance	RF, ANN, XGBoost, LR, KNN, etc.	0.998 (Binary RF)	Yes
[33]	ToN-IoT (All Layers, Heterogeneous)	Not Applied	SVM, XGBoost, NB, RF, etc.	0.988 (RF)	No
[29]	ToN-IoT	PCA + ANN	ANN	0.98	No
[31]	ToN-IoT, WUSTL-IIoT, EdgeIIoTset	None	KNN, SVM, RF, NB, etc.	0.987 (KNN)	No
[30]	ToN-IoT	Automated Feature Extraction	DNN with Feature Generators	0.995	Yes

In general, a comparative analysis of the work shows that most innovations are due to an overall design of the detection pipeline, which combines thorough feature engineering, overall model analysis, and real deployment considerations.

4.7 Discussion and Answers to The Research Questions

The empirical results derived from the developed machine learning-based NIDS pipeline provide significant insights into the efficacy of various classifiers, the impact of feature selection, and the nature of intrusion detection in IoT network environments.

Research Question 1: Which ML classifiers from thoes selected are most effective for binary and multiclass intrusion detection on the ToN-IoT dataset?

The experimental evaluation showed that tree-based ensemble models, especially Random Forest (RF) and XGBoost, always did better than other classifiers when it came to detecting intrusions in both binary and multiclass situations. These models have the best F1-scores for binary classification across a number of feature subsets, as shown in Table 4.8 and Figure 4.1. XGBoost was better at identifying different types of attacks in multiclass classification. It did this while keeping its accuracy high and its false-positive rates low, as seen by the peak multiclass performance statistics in Table 4.9. This shows that ensemble learners are strong and can adapt, especially in complicated IoT environments.

Research Question 2: What is the influence of feature selection on the classification performance and computational efficiency of machine learning classifiers regarding intrusion detection?

The application of the feature selection framework notably enhanced both the performance and computational efficiency of the classifiers. Models trained on optimized feature sets (e.g., rf_top_15_features, mi_top_15_features) not only maintained high classification accuracy but also demonstrated:

- **Reduced Training Time:** On average, using top-15 feature sets resulted in training times that were 5 to 10 times faster than the full feature baseline. This dramatic efficiency gain with negligible impact on performance is clearly illustrated in Figure 4.5 for the binary task and Figure 4.6 for the multiclass task.
- **Lower Inference Latency:** Due to smaller input dimensions, the time required for real-time classification was significantly reduced, as evidenced by the lower "Pred. Time" values for optimized models in Table 4.8 and Table 4.9 compared to the baseline.
- **Improved Model Interpretability:** The reduced number of features enabled clearer insight into which traffic attributes contribute most to threat detection. The feature importance rankings (e.g., **Table 4.7**) highlight the most influential variables.

Moreover, results confirmed that **byte-level and packet-level metrics**, along with **protocol and DNS indicators**, emerged as critical discriminative features, as consistently shown across the various feature selection methods (**Tables 4.3-4.7**). The framework's efficacy in mitigating overfitting and reducing computational overhead was validated, as classifiers trained on reduced sets performed comparably or better than those trained on the full feature set.

CHAPTER 5

CONCLUSION AND FUTURE WORK

This chapter puts together the real-world results of this thesis and explains what they mean for detecting network intrusions in Internet of Things (IoT) settings. It answers the research questions directly, talks about the study's real-world effects and limits, and suggests specific ways for future research to build on what has already been written.

5.1 Synthesis of Key Findings and Research Questions

The comprehensive experimental framework yielded several critical insights that provide direct answers to the core research questions of this thesis.

In response to Research Question 1, "Which ML classifiers from those selected are most effective for binary and multiclass intrusion detection on the ToN-IoT dataset?", the results unequivocally identify tree-based ensemble models as the superior choice.

- **Ensemble Model Dominance:** Random Forest (RF) and XGBoost always have the best performance metrics for both binary and multiclass classification problems. The literature [7], [24] supports their use for high-stakes cybersecurity categorization and for complicated IoT information.

In response to **Research Question 2,** "What is the influence of feature selection on the classification performance and computational efficiency of machine learning classifiers regarding intrusion detection?", the application of a systematic feature selection pipeline demonstrated a profound and overwhelmingly positive impact.

- **Drastic Efficiency Gains with Preserved Accuracy:** Choosing the right features turned out to be a very important step in optimization. Cutting the number of characteristics from more than 900 to only 15–20 sped up training

- by 5 to 10 times, with only a little drop in detection accuracy. This shows that feature selection is a great way to lower the "curse of dimensionality" and the expenses that come with it [10].
- **Enhanced Robustness and Interpretability:** The optimized models are less likely to overfit and more likely to work with new data because they do not include features that are not useful. The approach also showed that there is a constant collection of very important criteria (such byte/packet counts, protocol indicators, and DNS attributes) that are essential for finding fraudulent network traffic.
- **Superiority of Embedded and Information-Theoretic Methods:** For this difficult challenge, the Random Forest embedded importance and Mutual Information algorithms found the best feature subsets most of the time, showing that they are better than simpler correlation-based methods.

5.2 Practical Implications for NIDS Deployment

The findings of this research offer actionable guidance for deploying Network Intrusion Detection Systems (NIDS) in real-world IoT scenarios:

- **Real-Time Detection is Viable:** When you combine an efficient classifier like XGBoost with a well-chosen collection of features, you can have inference latencies that are much lower than 0.1 seconds. This makes it possible to detect threats in near-real-time on hardware with limited resources.
- **Tailored Deployment Strategies:**
    - **For Edge Gateways:** An XGBoost or Decision Tree model trained on a compact set of 10-15 features offers an outstanding balance of high accuracy, a low memory footprint, and near-instantaneous inference.

- **For Centralized/Cloud Servers:** In scenarios with greater computational power, using a larger feature set with a model like Random Forest can be justified to detect more nuanced attacks, provided the added latency is acceptable.

5.3 Limitations of the Study

While this research provides valuable insights, it is important to acknowledge its limitations:

- **Dataset Specificity:** The ToN-IoT dataset is what the results are based on. Real-world networks change more often and have "concept drift," which means that traffic patterns and attack vectors change over time. This is a problem that has been observed in thorough assessments of IDS datasets [8], [16].
- **Class Imbalance:** Even if SMOTE was utilized, multiclass classification still has a big problem with class imbalance for rare attack types, which makes models perform worse on minority classes [9].
- **Hyperparameter Configuration:** We used pre-set hyperparameters to make sure the classifiers were compared fairly. A full-scale tweaking process might lead to even better performance.

5.4 Recommendations for Future Work

Building on this study's foundation, several promising avenues for future research are recommended:

- **Advanced Feature Engineering:**
    - **Ensemble Feature Selection:** Look into ways to combine the results of other approaches (such as RF and MI) to make a more reliable and general feature subset [30].
- **Expansion to Advanced Models:**
    - **Deep Learning with Attention:** Look into deep learning architectures that use attention methods, such LSTMs or Transformers. These could assist models automatically focus on the most important sections of

input data, which is a promising way to improve IDS performance [5], [31].

- **Explainable AI (XAI):** Use advanced XAI approaches like SHAP or LIME to give thorough, instance-level explanations for the forecasts of high-performing "black-box" models like XGBoost. This will make them clearer and more reliable for security analysts [26].

- **Enhanced Validation and Generalization:**
  - **Cross-Dataset Validation:** Test the best-performing models from this study on other public NIDS datasets (e.g., UNSW-NB15, CIC-IDS2018) to assess their generalizability beyond the ToN-IoT environment [8].
  - **Online and Incremental Learning:** Develop and pilot models capable of online or incremental learning to adapt to concept drift without requiring complete retraining, addressing a key limitation of static datasets [32].

5.5 Concluding Remarks

This thesis has created, tested, and proven a feature-optimized machine learning framework for finding intrusions in IoT networks. The study has clearly revealed that a systematic way of choosing features is an important part of any working NIDS. When you combine powerful ensemble classifiers with smartly reduced feature sets, you get models that are very accurate and can be used in real time on devices with limited resources. The demand for smart, scalable, and effective security solutions will only rise as IoT networks evolve. The ideas shown in this thesis provide a clear and empirically proven way to move forward.

REFERENCES

[1] P. Kumar *et al.*, "PPSF: A Privacy-Preserving and Secure Framework Using Blockchain-Based Machine-Learning for IoT-Driven Smart Cities," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 3, pp. 2326–2341, Jul. 2021.

[2] A. M. Alnajim, S. Habib, M. Islam, S. M. Thwin, and F. Alotaibi, "A Comprehensive Survey of Cybersecurity Threats, Attacks, and Effective Countermeasures in Industrial Internet of Things," *Technologies*, vol. 11, no. 6, p. 161, Dec. 2023.

[3] I. Butun, P. Osterberg, and H. Song, "Security of the Internet of Things: Vulnerabilities, Attacks and Countermeasures," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 1–1, 2019.

[4] N. Mishra and S. Pandya, "Internet of Things Applications, Security Challenges, Attacks, Intrusion Detection, and Future Visions: A Systematic Review," *IEEE Access*, vol. 9, pp. 59353–59377, 2021.

[5] Qasem Abu Al-Haija and Ayat Droos, "A comprehensive survey on deep learning-based intrusion detection systems in Internet of Things (IoT)," *Expert Systems*, vol. 42, no. 2, Sep. 2024.

[6] I. Ullah and Q. H. Mahmoud, "A Two-Level Hybrid Model for Anomalous Activity Detection in IoT Networks," *IEEE Xplore*, Jan. 01, 2019.

[7] A. Thakkar and R. Lohiya, "A Review on Machine Learning and Deep Learning Perspectives of IDS for IoT: Recent Updates, Security Issues, and Challenges," *Archives of Computational Methods in Engineering*, vol. 28, Oct. 2020.

[8] M. S. Habeeb and T. R. Babu, "Network intrusion detection system: A survey on artificial intelligence-based techniques," *Expert Systems*, vol. 39, no. 9, Jul. 2022.

[9] Thin Tharaphe Thein, Y. Shiraishi, and Masakatu Morii, “Personalized federated learning-based intrusion detection system: Poisoning attack and defense,” *Future generation computer systems*, vol. 153, pp. 182–192, Apr. 2024.

[10] J. Li, Mohd Shahizan Othman, H. Chen, and Lizawati Mi Yusuf, “Optimizing IoT intrusion detection system: feature selection versus feature extraction in machine learning,” *Journal of Big Data*, vol. 11, no. 1, Feb. 2024.

[11] P. Nimbalkar and D. Kshirsagar, “Feature selection for intrusion detection system in Internet-of-Things (IoT),” *ICT Express*, vol. 7, no. 2, May 2021.

[12] A. Awajan, “A Novel Deep Learning-Based Intrusion Detection System for IoT Networks,” *Computers*, vol. 12, no. 2, p. 34, Feb. 2023.

[13] Ayoob Almotairi, Samer Atawneh, O. A. Khashan, and N. M. Khafajah, “Enhancing intrusion detection in IoT networks using machine learning-based feature selection and ensemble models,” *Systems Science & Control Engineering*, vol. 12, no. 1, Mar. 2024.

[14] M. A. Obaidat, S. Obeidat, J. Holst, A. Al Hayajneh, and J. Brown, “A Comprehensive and Systematic Survey on the Internet of Things: Security and Privacy Challenges, Security Frameworks, Enabling Technologies, Threats, Vulnerabilities and Countermeasures,” *Computers*, vol. 9, no. 2, p. 44, May 2020.

[15] A. Diro, N. Chilamkurti, V.-D. Nguyen, and W. Heyne, “A Comprehensive Study of Anomaly Detection Schemes in IoT Networks Using Machine Learning Algorithms,” *Sensors*, vol. 21, no. 24, p. 8320, Dec. 2021.

[16] A. Khraisat and A. Alazab, “A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges,” *Cybersecurity*, vol. 4, no. 1, Mar. 2021.

[17] A. R. Gad, A. A. Nashat, and T. M. Barkat, “Intrusion Detection System Using Machine Learning for Vehicular Ad Hoc Networks Based on ToN-IoT Dataset,” *IEEE Access*, vol. 9, pp. 142206–142217, 2021.

[18] R. Huo *et al.*, “A Comprehensive Survey on Blockchain in Industrial Internet of Things: Motivations, Research Progresses, and Future Challenges,” *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 88–122, 2022.

[19] A. Heidari and M. A. Jabraeil Jamali, “Internet of Things intrusion detection systems: a comprehensive review and future directions,” *Cluster Computing*, vol. 26, pp. 3753–3780, Oct. 2022.

[20] W. Li, W. Meng, and L. F. Kwok, “Surveying Trust-based Collaborative Intrusion Detection: State-of-the-Art, Challenges and Future Directions,” *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 280–305, 2021.

[21] M. F. Elrawy, A. I. Awad, and H. F. A. Hamed, “Intrusion detection systems for IoT-based smart environments: a survey,” *Journal of Cloud Computing*, vol. 7, no. 1, Dec. 2018.

[22] R. Mohan Das *et al.*, “A novel deep learning-based approach for detecting attacks in social IoT,” *Soft Computing*, May 2023.

[23] N. Y. Almusallam, Z. Tari, P. Bertok, and A. Y. Zomaya, “Dimensionality Reduction for Intrusion Detection Systems in Multi-data Streams—A Review and Proposal of Unsupervised Feature Selection Scheme,” *Emergence, Complexity and Computation*, vol. 24, pp. 467–487, Nov. 2016.

[24] T.-T.-H. Le, Y. E. Oktian, and H. Kim, “XGBoost for Imbalanced Multiclass Classification-Based Industrial Internet of Things Intrusion Detection Systems,” *Sustainability*, vol. 14, no. 14, p. 8707, Jul. 2022.

[25] N. Farah, Md. Avishek, F. Muhammad, A. Rahman, M. Rafni, and D. Md., “Application of Machine Learning Approaches in Intrusion Detection System: A Survey,” *International Journal of Advanced Research in Artificial Intelligence*, vol. 4, no. 3, 2015.

[26] Gaith Rjoub *et al.*, “A Survey on Explainable Artificial Intelligence for Cybersecurity,” *IEEE Transactions on Network and Service Management*, vol. 20, no. 4, pp. 5115–5140, Jan. 2023.

[27] G. Guo, X. Pan, H. Liu, F. Li, L. Pei, and K. Hu, “An IoT Intrusion Detection System Based on TON IoT Network Dataset,” *2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0333–0338, Mar. 2023.

[28] I. Tareq, B. M. Elbagoury, S. El-Regaily, and E.-S. M. El-Horbaty, “Analysis of ToN-IoT, UNW-NB15, and Edge-IIoT Datasets Using DL in Cybersecurity for IoT,” *Applied Sciences*, vol. 12, no. 19, p. 9572, Sep. 2022.

[29] S. K. B. Sangeetha, P. Mani, V. Maheshwari, P. Jayagopal, M. Sandeep Kumar, and S. M. Allayear, “Design and Analysis of Multilayered Neural Network-Based Intrusion Detection System in the Internet of Things Network,” *Computational Intelligence and Neuroscience*, vol. 2022, pp. 1–7, Sep. 2022.

[30] Kazım Kıvanç Eren, “Improving Intrusion Detection Systems for IoT Devices using Automated Feature Generation based on ToN_IoT dataset,” *Ieee.org*, 2025.

[31] S. Ismail, S. Dandan, and A. Qushou, “Intrusion Detection in IoT and IIoT: Comparing Lightweight Machine Learning Techniques Using TON_IoT, WUSTL-IIOT-2021, and EdgeIIoTset Datasets,” *IEEE Access*, vol. 13, pp. 73468–73485, 2025.

[32] T. M. Booij, I. Chiscop, E. Meeuwissen, N. Moustafa, and F. T. H. d. Hartog, “ToN_IoT: The Role of Heterogeneity and the Need for Standardization of Features and Attack Types in IoT Network Intrusion Data Sets,” *Ieee.org*, vol. 9, no. 1, pp. 485–496, 2016.

[33] A. Jovic, K. Brkic, and N. Bogunovic, “A review of feature selection methods with applications,” *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 1200–1205, May 2015.