NOVA IMS Information Management School

MDDM

Master's Degree Program in
Data-Driven Marketing

Improving Intrusion Detection Systems: Challenges with Public Datasets and the Role of Explainable AI

A Practical Guide Using NFS-2023-TE and HIKARI-2021

Toscano Ludovico

Master Thesis

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

Improving Intrusion Detection Systems: Challenges with Public Datasets and the Role of Explainable AI

A Practical Guide Using NFS-2023-TE and HIKARI-2021

by

Toscano Ludovico

Master Thesis presented as partial requirement for obtaining the Master's degree in Data-Driven Marketing, with a specialization in Data Science for Marketing.

Supervised by

Supervisor's Fernando Bação. PhD, NOVA Information Management School.

July, 2024

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism, any form of undue use of information or falsification of results along the process leading to its elaboration. I further declare that I have fully acknowledged the Rules of Conduct and Code of Honor from the NOVA Information Management School.

Rome, 27/06/2024

ABSTRACT

Intrusion detection systems (IDS) based on public datasets often show promising results in academic papers but fail to perform effectively in real-world scenarios due to flaws in dataset creation. This discrepancy raises the question of whether explainable algorithms should be essential when building machine learning-based intrusion detection systems. While recent research has highlighted some deficiencies in these studies, new datasets have emerged, and practical guides on addressing these issues are lacking. This thesis extends previous work by evaluating improvements in recent datasets and providing new insights based on our findings. Our study reveals persistent issues in existing datasets and demonstrates, through explainable AI techniques, why building intrusion detection systems with these datasets should be approached with caution. By contributing guidelines on what to avoid when developing an intrusion detection system, we also illustrate how certain aspects of the process require deeper analysis before proposing new models. This research underscores the critical need for more robust and representative datasets in IDS development, paving the way for more reliable and practical cybersecurity solutions.

KEYWORDS

intrusion detection; explainable AI; NIDS; public dataset, HIKARI-2021, NFS-2023-TE, Cybersecurity

TABLE OF CONTENTS

STATEMENT OF INTEGRITY i
Abstract ii
List of Figures 2
List of Tables 3
1. INTRODUCTION 4
2. BACKGROUND: UNDERSTANDING INTRUSION DETECTION SYSTEMS AND NETWORK DATA 6
2.1. **What does an IDS do?** 6
2.2. **What is an IP address, and what is a port?** 6
2.3. **What is a flow?** 7
2.4. **How these datasets are made** 8
2.5. **What is a FIN and RST flag?** 8
3. RELATED WORK 10
4. **METHODOLOGY** 17
4.1. **Explainable Algorithms in Machine Learning: SHAP and Beyond** 19
4.2. **Model Selection and Rationale** 21
4.3. **Data Preprocessing and Sampling Methodology** 22
4.4. **Evaluation Metrics and Handling Class Imbalance in Multi-class NIDS** 23
5. RESULTS 25
5.1. ***Data set analysis*** 25
5.2. **Model Evaluation and Performance Analysis** 25
5.3. ***Model explanations*** 30
6. **CONCLUSION** 40
BIBLIOGRAPHICAL REFERENCES 41
APPENDIX A 46

LIST OF FIGURES

FIGURE 5.1 - NFS-2023-TE - DECISION TREE FEATURE IMPORTANCE WITH SHAP OF BOT CLASS 31
FIGURE 5.2 - HIKARI-2021 - DECISION TREE FEATURE IMPORTANCE WITH SHAP OF BRUTEFORCE-XML CLASS 32
FIGURE 5.3- HIKARI-2021 - DECISION TREE FEATURE IMPORTANCE WITH SHAP OF XMRIGCC CRYPTOMINER 33
FIGURE 5.4- HIKARI-2021 - FEATURE IMPORTANCE WITH SHAP OF PROBING ATTACK OF DECISION TREE 34
FIGURE 5.5 - NFS-2023-TE - CATBOOST IMPACT ON THE MODEL OUTPUT OF DDOS WITH SHAP SORTED BY HIGHEST MAGNITUDE OF IMPACT 36

LIST OF TABLES

TABLE 3.1 - RELATED WORKS * SHAP WAS USED, BUT FOR FEATURE SELECTION ** THEY USE ALL THE FEATURES, BUT THEY GROUP THEM (PANIGRAHI & BORAH, 2018) 11
TABLE 4.1- TYPICAL DATASET ISSUES 17
TABLE 5.1 - NFS-2023-TE AND NFS-2023-NTE PARAMETERS 26
TABLE 5.2 - NFS-2023-TE TRAINING TIMES 27
TABLE 5.3- NFS-2023-TE – AVERAGE F1 SCORE OF 5-FOLD 28
TABLE 5.4 - HIKARI 2021 PARAMETERS 28
TABLE 5.5- HIKARI-2021 TRAINING TIMES FOR 31,952 TRAINING SAMPLES 29
TABLE 5.6 - HIKARI-2021 F1 SCORE AND TIMES 30
TABLE 5.7 - FWD_PKTS_PAYLOAD.MAX QUINTILES 32
TABLE 5.8 - DOWN_UP_RATION MEAN AND STD 34
TABLE 5.9 - NFS-2023-TE BIDIRECTIONAL_FIN_PACKETS STATISTICAL ANALYSIS 36
TABLE 5.10 - NFS-2023-TE - BIDIRECTIONAL_DURATION_MS 37

1. INTRODUCTION

In recent years, the global landscape of internet usage has undergone a dramatic transformation, with the number of users experiencing significant growth (Xiong et al., 2021). This expansion, driven by technological advancements and worldwide digitalization, has brought numerous benefits to society, including enhanced data sharing, network efficiency, and improved decision-making processes (Ahmetoglu et al., 2022).

However, this digital transformation has also introduced new challenges, particularly in cybersecurity. As the reliance on internet-connected systems grows, so does the potential for network threats and cyber-attacks (Walter et al., 2022). This trend has been further exacerbated by recent global events, such as the COVID-19 pandemic, which led to a surge in internet traffic due to increased remote work and stay-at-home activities. In 2022, global internet traffic reached a staggering 997,301 Gbps, a 30% increase from 2018 (TeleGeography, 2023).

The escalating threat landscape is evidenced by alarming statistics, such as a 29% increase in global cyber-attacks in 2021 (check point, 2021). This underscores the need for robust cybersecurity measures to protect individuals, businesses, and critical infrastructure.

In response to these growing threats, cybersecurity has evolved rapidly. Network Intrusion Detection Systems (NIDS) have become integral to organizational cybersecurity strategies. However, traditional signature-based and rule-based defensive mechanisms face challenges in coping with the increasing volume and complexity of information flowing through the internet (Gümüşbaş et al., 2021).

Researchers and computer engineers have turned to more advanced solutions to address these challenges, particularly in Artificial Intelligence (AI). Machine Learning (ML) and Deep Learning (DL) algorithms have shown promise in enhancing the performance of cyber defensive systems (Louk & Tama, 2023). These AI-based approaches offer the potential for more effective prediction and high-accuracy detection of emerging threats that may be difficult to identify using conventional antivirus programs.

While AI techniques have demonstrated impressive performance in various cybersecurity applications, such as intrusion detection, spam filtering, and malware identification (Sahakyan et al., 2021), a significant challenge lies in the trade-off between model performance and explainability. Interpretability and transparency have come to the forefront, particularly in light of regulations like the European Union's General Data Protection Regulation, which emphasizes the importance of understanding the logic behind AI-driven decisions (Goodman & Flaxman, 2017).

The field of Explainable AI (XAI) has emerged to address these concerns, offering strategies to make AI decisions more intelligible to humans. It has already been implemented in various domains, such as healthcare, business processes, financial and legal decisions, and autonomous vehicle (Nazar et al., 2021).

Building on the issue of explainability, this study practically addresses some specific criticisms raised by (Catillo et al., 2023). Starting from the premise that today, there are ML models that perform IDS work with incredible results, this study highlights how these academic results do not translate into real-world implementation of these models. The problem lies in a series of issues related to public datasets.

Besides being a simplified version of reality, these datasets include errors in creating the dataset itself, leading to the easy achievement of good results by machine learning models. A dataset taken as an example in this study is CIC-IDS 2017 (Sharafaldin et al., 2018). The reason this dataset was chosen is that among the various datasets released in recent years, it is the most cited.

Starting from the criticisms developed by (Catillo et al., 2023), we will analyze two new datasets, NFS-2023-TE (Pekar & Jozsa, 2024) and HIKARI-2021 (Ferriyan et al., 2021). The choice of these datasets is not random, as NFS-2023-TE is the latest in a series of datasets created using the same data as CIC-IDS 2017, correcting some issues found in the original dataset. These issues have been addressed by WMTC-2021 (Engelen et al., 2021; L. Liu et al., 2022) and CRiSIS-2022 (Lanvin et al., 2023), leading to NFS-2023-TE, the latest in this series of improvements. Meanwhile, HIKARI-2021 is an entirely new dataset inspired by CIC-IDS 2017 that adds seven new requirements to the eleven used (Gharib et al., 2016) by CIC-IDS 2017.

The need to analyze new datasets instead of a more famous one arises for two reasons. The first is that NFS-2023-TE assumes that CIC-IDS 2017 contains errors, while the other most cited datasets are nowadays outdated. The second reason is that these datasets become obsolete quite quickly (Guerra et al., 2022), so it makes no sense to insist on working on old datasets.

This work aims to minimize the trade-off between explainability of these algorithms and performance. To achieve this result, we will demonstrate how even classical machine learning algorithms can achieve results comparable to deep learning models. After that, we will show how XAI algorithms like Shap (Lundberg & Lee, 2017) are fundamental in demonstrating the validity of a model or the dataset on which this model is based. We will then analyze the HIKARI-2021 dataset, which, to our knowledge, has never been done before. This analysis has led to the discovery of serious problems that make any model developed on these data unsuitable for real-world usage. This analysis will be conducted partly by verifying that the problems found in CIC-IDS 2017 are not present in HIKARI-2021, partly following what is exposed in (Catillo et al., 2023), and finally, use Shap as an XAI algorithm.

Consequently, this study does not want to be yet another attempt to obtain good results with a new model, as starting from flawed data would be a useless exercise. However, it wants to be practical work that brings together some issues already seen in the past to contribute to the drafting of some guidelines that can help develop better datasets in the future.

This thesis is structured to systematically address the challenges in cybersecurity, focusing on developing and evaluating machine learning models for Network Intrusion Detection Systems (NIDS). Chapter 2 provides the necessary background on cybersecurity concepts and the current threat landscape. Chapter 3 reviews related works, examining previous research and identifying gaps this study aims to fill. Chapter 4 details the methodology, including the selection of datasets, an explanation of SHAP for model interpretability, the choice of specific models, evaluation metrics, and preprocessing steps. Chapter 5 presents the results, analyzing dataset characteristics, model performance, and SHAP-based explanations. Finally, Chapter 6 concludes with a summary of findings and discusses their implications for improving dataset quality and model explainability in cybersecurity.

2. BACKGROUND: UNDERSTANDING INTRUSION DETECTION SYSTEMS AND NETWORK DATA

This chapter briefly overviews key concepts in network security and intrusion detection systems (IDS). While not exhaustive, this introduction aims to equip readers with sufficient background knowledge to understand the context of our data science approach to IDS. The following sections offer concise explanations of fundamental ideas and processes in network monitoring and dataset creation for cybersecurity applications.

2.1. WHAT DOES AN IDS DO?

An Intrusion Detection System (IDS) is a security tool that monitors network traffic for suspicious activities. Its main job is to:

1. Watch all network connections passing through a specific point, usually near the router.
2. Analyze this traffic to identify potential threats or attacks.
3. Alert security teams or automatically respond when it detects something suspicious.

Traditionally, many IDS systems used signature-based detection, which could only identify known attacks. However, according to recent research, machine learning (ML) models are increasingly being used to build more effective IDS. The critical advantage of ML-based IDS is their ability to understand the underlying patterns of attacks. This allows them to potentially detect new, previously unseen threats, unlike signature-based systems limited to recognizing only known attack patterns.

In our analysis, we will focus on how to evaluate the performance of ML models used in IDS. Specifically, we will explore:

1. Metrics for model accuracy: We will examine the advantages and limitations of using different evaluation metrics such as accuracy, precision, recall, and F1 score. This comparison will help us understand which metrics are most suitable for assessing IDS performance in different scenarios.

2. Inference time: We will evaluate the speed at which the model can make predictions. This is crucial for an IDS, as faster detection allows for quicker response to potential threats, minimizing the impact of an attack. For instance, in a data theft scenario, reducing detection time directly correlates with reducing the amount of data that could be stolen.

By focusing on these aspects, we aim to develop a comprehensive understanding of building and evaluating effective ML-based Intrusion Detection Systems that can adapt to new threats and provide robust protection against evolving cyber risks.

2.2. WHAT IS AN IP ADDRESS, AND WHAT IS A PORT?

In the context of a dataset for an Intrusion Detection System (IDS), IP addresses and ports are represented as features. Each host (computer or device on a network) is identified by an IP address, which can be called its network "address". However, it is essential to note that a single host can have multiple active ports, which act like "doors" for different types of network traffic.

Ports, ranging from 0 to 65535, serve specific purposes. For example, port 80 is typically used for HTTP traffic, port 443 for HTTPS, and port 22 for SSH. This means one host can communicate using multiple services simultaneously, each through a different port.

In our dataset, we typically find four key pieces of information for each connection: the source IP address, source port, destination IP address, and destination port. While this information is crucial for network analysis, using it directly in machine learning model training can lead to problems.

The main issue is that IP addresses are often dynamically assigned and can change over time. They are also limited in number, especially with IPv4 (though IPv6 vastly expands this range). Similarly, while there are 65,536 possible ports, many are standardized for specific uses. Training a model on these specific values could create a bias where the model associates certain IPs or ports with malicious activity rather than learning the underlying patterns of the attack.

Attackers could exploit this bias. If they discover that the model considers specific IPs or ports as "safe," they could use these to bypass detection. Therefore, using these features for data correlation and initial analysis is generally better than training the detection model's core.

The primary reason for including IP addresses and ports in the dataset is to correlate records with the raw network traffic data, typically stored in pcap (packet capture) files. However, to make this correlation accurate, a timestamp is also necessary. The combination of IP, port, and timestamp uniquely identifies a specific network connection at a particular moment.

In summary, while IP addresses and ports are crucial for understanding network traffic, their direct use in machine learning models for cybersecurity can lead to vulnerabilities. Instead, they should be used carefully, primarily for data correlation and preprocessing, while the models should focus on more stable and generalizable features of network behavior.

2.3. WHAT IS A FLOW?

A flow in these datasets typically represents a single TCP or UDP connection, recorded as one row in the dataset. Each flow is composed of multiple IP packets but is summarized into a single record with various features:

1. Identifier features: These include source and destination IP addresses, ports, and timestamps. They uniquely identify a connection.
2. Statistical features: These are derived from the connection data, such as flag counts, average packet size, maximum payload length, duration, etc. These features capture the behavior of the connection.

The number of features can vary significantly between datasets. Modern public datasets often have 80 or more features, with some new tools potentially offering up to 150 features.

There's a clear start (three-way handshake) and various defined endings for TCP connections. UDP, being connectionless, does not have standard open/close procedures. In datasets, UDP "connections"

are typically defined as exchanges starting with the first observed packet and ending after a predefined timeout.

2.4. How these datasets are made

Creating a dataset for training an IDS involves several key steps, building upon the concepts of IP addresses, ports, and flows discussed earlier. The process begins with setting up a testbed, which involves creating a network environment including various devices like computers and servers. Some datasets may focus on specific environments, such as IoT networks, which can influence the protocols used.

Once the testbed is established, the next step is traffic generation. This involves programming devices to simulate typical user behavior like web browsing for normal traffic, while also performing controlled attacks within the network to generate attack traffic. The data is then captured using tools like tcpdump, which record network traffic during both typical and attack scenarios. This captured data is stored in pcap (packet capture) files containing detailed packet information.

Following data capture, flow generation takes place. Flowmeter tools such as CICFlowMeter (Lashkari et al., 2017) or NFStream (Aouini & Pekar, 2022) process the pcap files, aggregating individual packets into flows, which form the foundation for dataset samples. These tools then extract features from each flow, including statistical information like packet counts, flow duration, and flag counts. This process transforms raw packet data into a structured format suitable for machine learning.

The next crucial step is labeling, where each flow is categorized as either normal or a specific attack type. This is typically done by analyzing the traffic patterns and correlating them with the known attack times and targets. The labeled flows are then compiled into a CSV file, forming the final dataset. Each row in this file represents a single flow, with columns for various features and labels.

The dataset then undergoes validation to ensure accuracy and consistency. Often, both the processed CSV and original pcap files are made available to researchers. It's worth noting that some datasets, like WTMC-2021, CRiSIS-2022, and NFS-2023-TE, are created by reprocessing pcap files from existing datasets such as CIC-IDS 2017. This approach allows for improving and expanding existing datasets without capturing new network traffic.

The field of NIDS dataset creation continues to evolve. For instance, the authors of CICFlowMeter are developing a new Python-based tool with over 50 new features, potentially offering 130 features in total. This development could lead to more standardized dataset creation, allowing for better comparison of models across different datasets.

2.5. What is a FIN and RST flag?

IP packets, in addition to their payload, contain a series of metadata, including so-called flags. These flags perform various functions, including establishing or terminating the connection between two

hosts. Specifically, to understand network traffic datasets, we need to know what the RST and FIN flags do.

There are three ways to close a TCP connection. The first is by exchanging a packet with the RST flag set. The second method involves exchanging two packets with the FIN flag set between the two hosts. The third way is through a timeout if the connection is not used after a certain period.

It is essential to know that when monitoring a TCP connection, we should theoretically see either one RST packet or two FIN packets passing through, but not more than one RST or one FIN. Among all the features used for today's datasets, this is one of the few that contains an absolute value that, if exceeded, indicates a malfunction during the dataset creation.

This behavior of RST and FIN flags provides a unique feature in network datasets. Unlike many other features with varying thresholds or patterns, the count of RST or FIN flags per connection has a clear, absolute limit. Exceeding this limit (more than one RST or more than two FIN packets per connection) strongly suggests either an anomaly in the network behavior or an error in the dataset creation process.

For data scientists working with network traffic datasets, understanding these TCP flag behaviors can be crucial for data cleaning and preprocessing, feature engineering, anomaly detection, and understanding the underlying network processes represented in the data.

3. RELATED WORK

This section reviews recent literature on Intrusion Detection Systems (IDS) using machine learning (ML), focusing on studies that employ the HIKARI-2021 and CIC-IDS 2017 datasets. Our analysis aims to identify common trends and challenges in current IDS research.

We selected six papers using HIKARI-2021 (the total available at the time of writing) and six papers using CIC-IDS 2017 (two most cited papers using SHAP and five most cited overall). Some papers include additional datasets beyond these two.

Our analysis utilizes criteria adapted from (Catillo et al., 2023), focusing on ML implications for IDS. We excluded data partitioning issues as they were irrelevant to our non-sequential analysis and transferability due to dataset limitations. We added an explainability criterion for papers employing post hoc explainability algorithms. Before presenting Table *3.1*, it's important to explain the criteria used in our evaluation.

We consider whether studies avoid attack-revealing features and ease of detection, which refers to features that inadvertently make attack detection easier due to dataset limitations. For example, in NFS-2023-TE, all attacks use TCP protocol, while the dataset also includes UDP connections. This creates a bias where UDP traffic is automatically classified as non-malicious, which doesn't reflect real-world scenarios. We also observed instances where researchers misunderstood certain features, such as the 'Unnamed' features in HIKARI-2021, which we'll discuss later.

Another criterion is avoiding unmotivated complexity. Some papers propose complex deep learning models without comparing them to simpler, classic models like random forests. Even when comparisons are made, the baseline models often lack proper fine-tuning, potentially skewing results in favor of more complex models.

We also evaluate the use of evaluation metrics. Inappropriate metrics are sometimes used to evaluate models. For instance, using accuracy on an imbalanced dataset with 90% benign traffic can easily yield 90% accuracy by classifying everything as benign, which is misleading.

Source code availability is another important factor. Few papers provide source code, which becomes problematic when crucial details like feature selection or class evaluation are omitted from the paper. Source code could clarify these issues and dispel doubts.

Explainability is assessed to determine whether a model's decisions are explained. Section 5 will analyze model explanations, demonstrating their importance.

We consider whether studies use updated datasets. Despite known errors in datasets like CIC-IDS 2017, some researchers continue using them instead of updated versions that correct them. For HIKARI-2021, the authors have released updated data improving class balance, so we will evaluate this as the update.

Lastly, we examine whether all attacks are used. Not all papers use all available attack types. While this can be justified, some studies don't mention excluded attacks, and their absence only becomes apparent in detailed metrics. This is another reason why publishing source code enhances credibility.

Table 3.1 - Related works * shap was used, but for feature selection ** they use all the features, but they group them *(Panigrahi & Borah, 2018)*

Paper	*Avoid attack-revealing features and ease of detection*	*Avoid unmotivated complexity*	*Use of the evaluation metrics*	*Source code*	*Explainability*	*Updated dataset*	*All attacks*	*Dataset used*
(Kwon et al., 2023)	Yes	Yes	Yes	No	No	No	Yes	Hikari-2021
(Noori et al., 2023)	Yes	Yes	Yes	No	No	No	No	Hikari-2021
(Louk & Tama, 2023)	-	No	Yes	No	No	No	No	Hikari-2021
(Rajak et al., 2022)	-	No	Yes	No	No	No	Yes	Hikari-2021
(Fernandes & Lopes, 2022)	No	Yes	Yes	No	No	No	Yes	Hikari-2021
(Fernandes et al., 2023)	No	Yes	Yes	No	No	No	Yes	Hikari-2021
(Chauhan & Shah Heydari, 2020)	Yes	No	Yes	No	No *	No	Yes	CIC-IDS 2017
(Zavrak & Iskefiyeli, 2020)	Yes	Yes	No	No	No	No	Yes	CIC-IDS 2017
(Kurniabudi et al., 2020)	Yes	Yes	No	No	No	No	Yes**	CIC-IDS 2017
(Yulianto et al., 2019)	Yes	Yes	Yes	No	No	No	No	CIC-IDS 2017
(Maseer et al., 2021)	Yes	Yes	Yes	No	No	No	Yes**	CIC-IDS 2017
(J. Liu et al., 2021)	Yes	Yes	No	No	No	No	Yes	CIC-IDS 2017
This work	Yes	Yes	Yes	Yes	Yes	Yes	Yes	

(Kwon et al., 2023) conducted a study analyzing the HIKARI-2021 dataset, focusing on network anomaly detection and addressing challenges posed by unbalanced data and zero-day attacks. Their research aimed to answer three main questions: how Machine Learning (ML) and Deep Learning (DL) models perform on the highly unbalanced HIKARI-2021 dataset, what strategies can improve model performance on this unbalanced dataset, and whether it's possible to detect previously unseen attack types using these models.

The authors used the HIKARI-2021 dataset, which contains six label categories: Background, Benign, Bruteforce, Bruteforce-XML, Probing, and XMRIGCC CryptoMiner. They noted that this dataset is highly skewed towards normal instances, with a normal-to-attack ratio of approximately 13.7:1. To address these challenges, the researchers employed several ML and DL models, including Random Forest (RF), Extreme Gradient Boosting (XGBoost), Multi-Layer Perceptron (MLP), and Convolutional Neural Network (CNN). They evaluated these models under different data preprocessing scenarios, using the unbalanced dataset as-is and applying random sampling to create balanced datasets of different sizes.

The study's key findings revealed that on the unbalanced dataset, none of the models performed satisfactorily, with F-measures ranging from 0.5384 to 0.6998. However, balancing the dataset through sampling significantly improved performance, with F-measures reaching up to 0.9251. Zero-day detection proved highly challenging, with most models failing to detect untrained attack types.

It is important to note that this study did not use the updated version of the HIKARI-2021 dataset, which could potentially improve the models' performance and impact the results of zero-day detection experiments. Additionally, the Random Forest and XGBoost models were used with default settings, without any fine-tuning, which may have affected their performance, especially in comparison to the neural network models (MLP and CNN) that were more carefully constructed. Lastly, the authors did not provide access to their source code.

(Noori et al., 2023) proposed a novel approach called Dynamic Feature Aware Genetic Programming Ensemble (DFA-GPE) to address several challenges in intrusion detection systems, particularly in the context of data streams. Their work aimed to tackle feature drift, high dimensionality, and data imbalance in modern network traffic data. To address these challenges, the authors proposed DFA-GPE, which extends the GP-combiner framework by incorporating a Dynamic Feature Selection (DFS) block based on an improved multi-objective Particle Swarm Optimization (PSO). This approach allows for variable-length feature selection and optimizes for both accuracy and memory usage.

The authors evaluated their method on two datasets: HIKARI 2021 for network traffic analysis and TON_IoT 2020 for IoT and IIoT intrusion detection. Their proposed solution included several key components: an intelligent population initialization algorithm using Bernoulli distribution and symmetric uncertainty, a hybrid transfer function set for versatile searching behavior, a novel exemplar selection method to balance exploration and exploitation, and the use of the entire Pareto front for feature selection.

The authors reported significant improvements in performance and efficiency. On the HIKARI 2021 dataset, DFA-GPE achieved 99.09% accuracy while reducing memory usage by 93.49%. On the TON_IoT 2020 dataset, it achieved 92.64% accuracy with a 62.93% reduction in memory usage. However, it's worth noting that the authors did not use the updated version of the HIKARI 2021 dataset. Furthermore, there is a discrepancy in the attack categories mentioned. In the dataset section related to HIKARI 2021, they only mention three attack categories (Brute Force, Brute Force-XML, and probing) out of the four known categories, without explaining why one has been removed. This missing attack category is also absent in Figure 7, while Figure 4 shows a completely different set of attacks. These unexplained changes highlight the importance of providing source code, which could help clarify these discrepancies and ensure the reproducibility of the research.

(Louk & Tama, 2023) proposed a novel approach called Dual-IDS, which combines bagging with gradient-boosting decision tree (GBDT) models for network anomaly intrusion detection. Their work aimed to improve detection rates and reduce false alarm rates in anomaly-based intrusion detection systems, while enhancing the performance of GBDT algorithms through a dual ensemble technique. The authors evaluated their method on three datasets: HIKARI 2021, NSL-KDD, and UNSW-NB15.

Their proposed solution included a dual ensemble strategy combining bagging with GBDT algorithms, evaluation of multiple GBDT algorithms including GBM, LightGBM, CatBoost, and XGBoost, and hyperparameter tuning using random search for each constituent learner. The authors reported significant improvements in performance across various metrics. For example, on the NSL-KDD KDDTest+ dataset, they achieved 91.57% accuracy, 1.3% false positive rate, 98.67% precision, and an F1 score of 0.915.

However, attempts to reproduce these results were unsuccessful, highlighting the critical importance of providing accessible source code for reproducibility in research. The authors stated that their source code was available upon request, but attempts to obtain it were unsuccessful.

(Rajak et al., 2022) proposed a CNN-LSTM-based Intrusion Detection System (IDS) framework specifically designed for precision farming and Industrial Internet of Things (IIoT) environments. Their work aimed to address security threats to smart agriculture devices and IoT sensors, the need for real-time intrusion detection in precision farming environments, and handling encrypted network traffic in IDS. The authors utilized the HIKARI-2021 dataset to develop and evaluate their model, chosen for its focus on encrypted traffic and representation of modern network behaviors.

The proposed solution included a combined CNN-LSTM architecture for feature extraction and sequence learning, data preprocessing using correlation analysis to select optimal features, and a real-time detection framework for immediate threat identification. The CNN-LSTM model architecture consisted of seven convolutional layers, four max-pooling layers, and four batch normalization layers in the CNN component, two LSTM layers, two dense layers, and two dropout layers in the LSTM component, and a final dense output layer with softmax activation for classification.

The authors reported an accuracy of 93.27% for both training and validation sets, which is competitive but slightly lower than some traditional machine learning approaches they compared against (e.g., SVM and MLP achieved 99% accuracy). However, critical analysis of this work reveals several important points. The authors did not specify using the most recent version of the HIKARI-2021 dataset, which may impact the relevance of their results. Additionally, Table 2 in the paper indicates that both 'traffic_category' and 'Label' were included in the selected features, which is a critical issue as these columns directly contain the target information, potentially leading to data leakage and overfitting.

Without access to the source code, it's impossible to verify if the inclusion of 'traffic_category' and 'Label' in Table 2 was an error in reporting or an actual flaw in the methodology. If the reported results are correct, they demonstrate that this complex CNN-LSTM model does not outperform classical machine learning approaches like SVM and MLP. However, the exceptionally high performance of these classical models, combined with the inclusion of both 'traffic_category' and 'Label' in the feature set, strongly suggests that these results are likely due to incorrect feature selection. This would lead to data leakage, where the model is essentially being given the answers it's trying to predict, resulting in artificially inflated performance metrics.

These issues highlight the importance of transparency in research methodologies and the need for source code availability. The potential inclusion of target variables in the feature set is particularly concerning, as it could lead to unrealistically high performance that wouldn't generalize to real-world scenarios. It also underscores the critical importance of proper feature selection in machine learning studies, especially in the field of intrusion detection where the reliability and generalizability of models are crucial.

(Fernandes et al., 2023; Fernandes & Lopes, 2022) inadvertently included an artificial feature ("unnamed: 0.1") in their analyses, which was not part of the original dataset but rather an artifact created during CSV file loading. This feature, representing row indices, allowed models like Random Forest to learn sequential patterns of attacks, leading to artificially inflated performance metrics. Notably, the inclusion of this feature was not explicitly mentioned in either study. It was only through

careful examination of Figure 2 in the first study that this mistake became apparent. The second study revealed a significant performance drop when excluding this feature but failed to recognize its inappropriate inclusion, again highlighting how the source code could help to recognize that there was a mistake in the first study.

(Chauhan & Shah Heydari, 2020) proposed a novel approach to generate polymorphic adversarial DDoS attacks using Generative Adversarial Networks (GANs). Their work aimed to address vulnerabilities of machine learning-based Intrusion Detection Systems (IDS) against new, unknown types of attacks and the ability of attackers to evade detection by continuously changing attack profiles. The authors focused specifically on DDoS attacks, limiting the scope of their study to this particular type of network intrusion. This focused approach allowed for a more in-depth analysis of DDoS attack characteristics and their detection mechanisms.

Key aspects of their methodology include the use of SHAP (SHapley Additive exPlanations) for feature selection, GAN-based adversarial attack generation, and polymorphic attack generation. The authors evaluated their method using the CICIDS2017 dataset and reported significant success in generating adversarial attacks that could evade detection. Their results indicated that by continuously changing the attack profile, even defensive systems employing incremental learning remained vulnerable to new attacks.

This study contributes to the field by demonstrating the potential vulnerabilities of ML-based IDS to adversarial attacks, providing a method for generating polymorphic DDoS attacks, which could be used to improve the robustness of IDS, and showcasing an alternative use of SHAP for feature selection in the context of network security.

(Zavrak & Iskefiyeli, 2020) focused on detecting network intrusions and anomalies using flow-based data and unsupervised deep learning methods, specifically Autoencoder (AE) and Variational Autoencoder (VAE), along with One-Class Support Vector Machine (OCSVM). They used the CICIDS2017 dataset, which contains flow-based network traffic data including normal traffic and various types of attacks. Their methodology employed a semi-supervised learning approach where models were trained only on normal traffic data, and AE, VAE, and OCSVM were used as anomaly detectors.

The model architecture for both AE and VAE used a simple deep autoencoder architecture with two hidden layers in the encoder and decoder, with the bottleneck layer having 64 dimensions. They evaluated their models using Receiver Operating Characteristics (ROC) curves and Area Under the ROC Curve (AUC).

Their results showed that VAE generally performed better than AE and OCSVM in detecting anomalies, although performance varied across different types of attacks. Some attacks (e.g., various DoS attacks) were well-detected, while others (e.g., SSH Patator, Web Attack - SQL Injection) were poorly detected by all methods. The study highlighted the need for additional supervised learning methods to reduce false alarms and suggested considering flow-based features collected at specified time intervals to better model attack characteristics for future work.

(Kurniabudi et al., 2020) conducted a study on feature selection for anomaly detection using the CICIDS-2017 dataset. Their work aimed to address challenges in intrusion detection systems, particularly high dimensionality of data, impact on computational complexity, and computational time. The authors used

Information Gain for feature selection and experimented with five classifier algorithms: Random Forest (RF), Bayes Net (BN), Random Tree (RT), Naive Bayes (NB), and J48. They analyzed only 20% of the CICIDS-2017 dataset, which was split into 70% training and 30% testing data.

Their methodology included using Information Gain to rank and group features based on weight scores, evaluating classifier performance using metrics like accuracy, TPR, FPR, precision, recall, and execution time, and implementing 10-fold cross-validation. Their results showed that Random Forest achieved the highest accuracy (99.86%) using 22 selected features, while J48 performed best (99.87% accuracy) using 52 features, but with longer execution time. The number of selected features significantly impacted both detection accuracy and execution time.

However, there are several critical points to consider regarding this study. The authors only used 20% of the CICIDS-2017 dataset, which may not fully represent the dataset's complexity and could impact the generalizability of their results. The study adopts a relabeling approach for attack traffic, as suggested by previous research. While this can help address class imbalance issues, it may also obscure the detection performance for individual, less common attack types. Additionally, while Information Gain was used to rank features, the authors manually determined the minimum weight value for feature selection, introducing a subjective element that could affect reproducibility.

(Yulianto et al., 2019) proposed an approach to enhance the performance of AdaBoost-based Intrusion Detection Systems (IDS) on the CIC IDS 2017 dataset. Their work aimed to address the imbalance in training data, inappropriate selection of classification methods, and the need for effective feature selection. To tackle these issues, the authors proposed a framework combining Synthetic Minority Oversampling Technique (SMOTE) to handle the imbalance in training data, Principal Component Analysis (PCA) and Ensemble Feature Selection (EFS) for selecting important attributes from the dataset, and AdaBoost classifier as the core classification method.

The authors evaluated their method on the CIC IDS 2017 dataset, focusing on DDoS attacks. They used 70% of the data for training and 30% for testing, applied SMOTE with a 200% minority oversampling class, and used a threshold value of T = 0.9 for feature selection. Their results showed significant improvements over previous methods, with the AdaBoost classifier using PCA and SMOTE achieving an Area Under the Receiver Operating Characteristic curve (AUROC) of 92%, and the AdaBoost classifier using EFS and SMOTE producing an accuracy of 81.83%, precision of 81.83%, recall of 100%, and F1 Score of 90.01%. These results demonstrated a substantial improvement over the previous AdaBoost implementation on the same dataset, which had achieved precision, recall, and F1 scores of only 77%, 84%, and 77% respectively. However, two limitations of this study regarding the goal that this thesis set are the fact that explaining the PCA is a problem and that they used only SMOTE.

(Maseer et al., 2021) conducted a comprehensive benchmarking study of machine learning algorithms for anomaly-based intrusion detection systems (AIDS) using the CIC-IDS 2017 dataset. Their work aimed to address several issues in existing AIDS research, including the randomness of selected algorithms, parameters, and testing criteria, the use of outdated datasets, and shallow analyses and validation of results. The authors evaluated 10 popular supervised and unsupervised machine learning algorithms, including Artificial Neural Network (ANN), Decision Tree (DT), k-Nearest Neighbor (k-NN), Naive Bayes (NB), Random Forest (RF), Support Vector Machine (SVM), Convolutional Neural Network (CNN), Expectation-Maximization (EM), k-Means, and Self-Organizing Maps (SOM).

They tested multiple models and tuning parameters for each algorithm to achieve optimal classifier performance, using evaluation metrics including accuracy, precision, recall, F1-score, training time, and testing time. Key findings revealed that k-NN, DT, and NB algorithms demonstrated the best overall performance in detecting web attacks, supervised learning algorithms generally outperformed unsupervised ones, and DT and k-NN provided the best balance of accuracy and efficiency when considering both training and testing times. However, many models struggled to detect the rarest attack class (SQL Injection) due to class imbalance in the dataset.

The authors highlighted the importance of using comprehensive evaluation metrics beyond just accuracy, especially for imbalanced multi-class datasets like CICIDS2017. They also emphasized the need to consider both effectiveness (detection accuracy) and efficiency (training/testing times) when evaluating AIDS models. However, it's important to note that they did not use the updated version of the CICIDS2017 dataset, which could impact the applicability of their results. Additionally, they focused on only four categories of attacks (BENIGN, Brute Force, XSS, and SQL Injection) out of the multiple attack types available in the full CICIDS2017 dataset. This limited scope, while allowing for a focused analysis, may not represent the full range of intrusion detection challenges present in the complete dataset.

(J. Liu et al., 2021) proposed a network intrusion detection system using ADASYN (Adaptive Synthetic) oversampling and LightGBM. They compared several resampling techniques, including SMOTE and ADASYN, in combination with LightGBM on three datasets: NSL-KDD, UNSW-NB15, and CICIDS2017. The comparison between SMOTE and ADASYN focused primarily on accuracy and false alarm rate. For NSL-KDD,

In light of these observations from related works, it's clear that while each study has its unique approach and objectives, there are common challenges across the field. These include issues with dataset quality, lack of source code availability, and inconsistent evaluation methods. Our work aims to address these challenges by using more recent datasets, implementing standardized preprocessing and evaluation techniques, incorporating explainable AI, and providing full access to our source code. By doing so, we hope to demonstrate how addressing these common issues can lead to the development of more robust and reliable NIDS models, potentially bridging the gap between academic research and real-world applicability.

4. METHODOLOGY

The evolution of Network Intrusion Detection Systems (NIDS) datasets has seen significant progress over the years. Among the most cited datasets according to Google Scholar are KDD Cup '99 (Lee et al., 1999), NSL-KDD (Tavallaee et al., 2009), UNSW-NB15 (Moustafa & Slay, 2015), and CIC-IDS 2017 (Sharafaldin et al., 2017). KDD Cup '99, one of the earliest and most influential datasets, set the foundation for IDS research by providing a large-scale, labeled dataset of network intrusions. NSL-KDD improved upon KDD Cup '99 by addressing key limitations such as redundant records and statistical issues, enhancing the dataset's reliability for evaluating machine learning-based IDSs.

UNSW-NB15 was created to address the outdated nature of previous benchmarks. It used the IXIA PerfectStorm tool to generate a mix of modern normal and attack traffic, introducing nine contemporary attack categories and 49 features, making it more relevant for current IDS evaluation. CIC-IDS 2017 marked a milestone as it met all 11 criteria set by (Gharib et al., 2016) for IDS datasets, providing a more realistic and comprehensive representation of network behaviors and attack patterns.

As of February 2024, NSL-KDD led with 5,131 citations, followed by CIC-IDS 2017 at 3,149 citations, and UNSW-NB15 at 2,740 citations. Despite their popularity, newer datasets have emerged to address limitations in these earlier versions. Our study focuses on two recent datasets: NFS-2023-TE (Pekar & Jozsa, 2024), derived from CIC-IDS 2017, and HIKARI-2021 (Ferriyan et al., 2021), a new approach to NIDS datasets.

CIC-IDS 2017 was created over five days, simulating various attack scenarios and providing 12 attack labels, 1 benign label, and 80 traffic features. However, subsequent analyses reported several issues, including over 5% corruption in dataset labeling (Engelen et al., 2021; L. Liu et al., 2022) and non-compliance with TCP connection closure standards (Brownlee et al., 1999) in the CICFlowMeter tool used to create the dataset.

The evolution from CIC-IDS 2017 to NFS-2023-TE involved several intermediate steps. WTMC-2021 provided extensive documentation on correct attack labeling and improved CICFlowMeter by fixing various issues and implementing new attributes. CRiSIS-2022 (Lanvin et al., 2023) further improved CICFlowMeter, addressed issues with duplicate packets and packet ordering in pcap files, and added a label for port scan attacks. NFS-2023-TE uses NFStream (Aouini & Pekar, 2022) instead of CICFlowMeter, closes connections after the first FIN or RST flag, aligning with most flow analyzers (Hofstede et al., 2014), and drops duplicates within 10,000 packets instead of 500 microseconds.

HIKARI-2021 focuses on encrypted traffic, particularly application layer attacks delivered via HTTPS. It offers comprehensive data provision, including pcap files, CSV files, and PKL format. The dataset provides an enhanced feature set of 86 traffic features, including 80 adopted from CICIDS-2017 and 6 additional features derived from Zeek.

Table 4.1- Typical dataset issues

Issue	HIKARI-2021	NFS-2023-TE
Simplification of the data collection	While HIKARI-2021 has made improvements, its environment is still simplified compared to	Uses the same environment as CIC-IDS 2017, inheriting its

environment	real-world networks. It focuses primarily on web traffic, which is both a strength and a limitation.	simplifications and limitations.
Contemporaneity and effectiveness of the attack	Focuses on application layer attacks, which the authors claim represent 80% of web attacks. However, the effectiveness of these attacks in real-world scenarios is not thoroughly discussed.	Some attacks in this dataset have been proven ineffective against properly configured systems, as noted in (Catillo et al., 2021).
Representativeness of the normal baselines	Represents an improvement over previous datasets but still falls short of real-world complexity. It only represents web traffic, which aligns with its goals but remains a limitation.	Inherits the limitations of CIC-IDS 2017 in terms of normal traffic representation, which may not fully capture the complexity of real-world network behavior.
Bugs of the feature extractor and incorrect flow records	The authors mention using Zeek for feature extraction but don't provide detailed configuration information. They also reference an unspecified Python tool, raising questions about the reproducibility of their method.	Was specifically created to address the flaws in CICFlowMeter and improve upon CIC-IDS 2017, representing a significant improvement in this area.
Data Labeling (Was the traffic analyzed or labeled based on IP, port, and timestamp?)	The labeling process cannot be fully verified due to the absence of timestamps. However, the authors discovered an unexpected attack during background validation, suggesting a more thorough analysis.	Is the result of multiple iterations and improvements in labeling methodology. From a methodological standpoint, the labeling process should be more robust and less subject to criticism.
Class imbalance	Shows significant imbalance. The majority class (Benign) represents 71.77% of all data, with the smallest attack class (Bruteforce) at only 1.02% of the total dataset.	Exhibits extreme imbalance. The majority class (BENIGN) accounts for 83.71% of all data, while the smallest attack class (Heartbleed) represents only 0.00035% of the total dataset.

Table 4.1 summarizes the comparison between HIKARI-2021 and NFS-2023-TE based on typical dataset issues identified by (Catillo et al., 2023). This comparison highlights the strengths and weaknesses of each dataset across various criteria.

The field of NIDS dataset creation continues to evolve, with the authors of CICFlowMeter developing a new Python-based tool potentially offering 130 features. This development could lead to more standardized dataset creation, allowing for better comparison of models across different datasets. By selecting NFS-2023-TE and HIKARI-2021 for our study, we aim to leverage the most current and refined datasets reported in the literature. A thorough analysis of these datasets will be conducted, providing deeper insights into their actual characteristics and potential limitations, contributing to the ongoing discussion on NIDS dataset quality and suitability.

4.1. EXPLAINABLE ALGORITHMS IN MACHINE LEARNING: SHAP AND BEYOND

As models become increasingly complex in machine learning, the need for interpretability and explainability has grown significantly. SHAP (SHapley Additive exPlanations), introduced by (Lundberg & Lee, 2017), leverages game theory concepts to provide post hoc explanations for machine learning models. At its core, SHAP computes approximations of Shapley values, a concept from cooperative game theory used to fairly distribute each feature's contribution to the model prediction.

Shapley values are calculated by considering all possible combinations of features and assessing how each feature's presence or absence affects the model's output. This approach is considered "fair" because it accounts for all possible feature interactions and orders. For example, if we have features A, B, and C, we would consider scenarios like {A}, {B}, {C}, {A,B}, {A,C}, {B,C}, and {A,B,C} to determine each feature's contribution. However, computing exact Shapley values is computationally expensive, especially for models with many features. SHAP addresses this challenge by providing efficient approximation methods.

SHAP offers several key explainers. Kernel SHAP is a model-agnostic explainer adapted from LIME (Local Interpretable Model-agnostic Explanations) (Ribeiro et al., 2016). Deep SHAP is designed for neural networks based on the DeepLIFT algorithm (Shrikumar et al., 2019). Tree SHAP, introduced in subsequent papers (Lundberg et al., 2018, 2020), is a model-specific explainer for tree-based models and ensembles, notably faster than Kernel SHAP and Deep SHAP. Fast Tree SHAP (Yang, 2021) is not offered by the SHAP library itself but is a separate library that can output Shapley values in the same format as the original library. It can use the original Tree SHAP algorithm and two different versions of their improved algorithm. The compatibility of its output with the original library allows the use of the original SHAP plotting functions.

Our focus on SHAP, particularly TreeExplainer for tree-based models, is motivated by its unique theoretical guarantees and practical efficiency. As stated by (Lundberg et al., 2020), TreeExplainer offers several key advantages, including fast local explanations, guaranteed consistency, and polynomial time complexity. Within the class of additive feature attribution methods, Shapley values are the only approach that satisfies three critical properties: local accuracy (additivity), consistency (monotonicity), and missingness. Local accuracy ensures that the explanation's attribution values sum up to the model's output for a specific input. Consistency guarantees that if a model changes so that a feature's contribution increases or stays the same regardless of other inputs, that feature's attribution will not decrease. Missingness ensures that features that do not contribute to a prediction are assigned zero importance.

The distinction between model-agnostic and model-specific explainers is important to note. Model-agnostic explainers, like Kernel SHAP, typically work with a surrogate model. In the case of Kernel SHAP, like LIME, they use a linear or logistic regression depending on the case. This surrogate model needs to approximate the output of the original model, and the explanation is then based on this surrogate model rather than the original one. In contrast, model-specific explainers like Tree SHAP leverage the model's characteristics. For tree-based models, Tree SHAP can read all the nodes to see how different outputs change based on the boundaries of the nodes.

Another important distinction in explainable AI is between true-to-model and true-to-data explanations (H. Chen et al., 2020). True-to-model explanations, provided by Tree SHAP and Linear SHAP, assign non-zero importance only to features actually used by the model. They break feature dependencies following causal inference principles (Janzing, 2019). True-to-data explanations may assign equal importance to highly correlated features, even if the model only uses one.

SHAP uses different approaches to assess feature importance, depending on the specific explainer and model type. Kernel SHAP uses perturbation with background data, randomly swapping feature values between the explained instance and the background data. Tree SHAP can use tree path-dependent perturbation and does not require separate background data, instead following the nodes of the tree, which contain all the training data. Linear SHAP can use interventional feature perturbation and reads the linear model weights to provide explanations.

The choice of background data is crucial for any model that uses it, not just Kernel SHAP. Background data serves as a reference point against which feature contributions are measured. Poor selection of background data can lead to misleading explanations. Ideally, background data should represent the features' meaningful "average" or baseline state.

SHAP provides both local and global explanations. Local explanations explain individual predictions. For example, in an intrusion detection system, a local explanation might show which network traffic features most contributed to classifying a particular connection as malicious. Global explanations provide overall feature importance across the dataset. These can be computed using either the mean absolute SHAP value or the maximum absolute SHAP value across observations. Global explanations in an IDS context might reveal which features are most important for detecting intrusions across all traffic. Global explanations are typically derived by aggregating local explanations across many instances. While global explanations provide an overview of feature importance, they may obscure nuances captured in local explanations, especially for models with complex decision boundaries.

The last thing we need to know is that not all explainers are post hoc. Post hoc explanations refer to interpretability methods applied after a model has been trained. These techniques aim to explain the decisions or predictions of a model without altering its internal structure or training process. SHAP is a prime example of a post hoc explanation method. Post hoc methods can be model-specific or model-agnostic, are applied as a separate step after model training and prediction, do not impact the model's predictive capabilities, and can be applied selectively to specific predictions or instances of interest. While post hoc methods like SHAP offer powerful explanatory capabilities, they do add an additional computational step after model training and prediction.

In contrast to models requiring post hoc explanations, we also explored alternative approaches that offer inherent explainability, specifically the Explainable Boosting Machine (EBM) (Lou et al., 2013). EBM, a Generalized Additive Model with Interactions (GA2M), represents a significant advancement in interpretable machine learning.

As stated by the authors of EBM, this model aims to achieve predictive power comparable to complex deep learning models while providing clear, interpretable explanations similar to simpler glass box models like linear regression. This combination of high performance and inherent interpretability makes EBM particularly interesting for applications where accuracy and explainability are crucial, such as intrusion detection systems.

Our study included EBM as a glass box model for comparison. Our approach focused on comparing the training time of EBM with other models. This comparison is particularly relevant because traditional black box models require additional time to run SHAP to generate explanations after training. In contrast, EBM provides explanations inherently during the training process, potentially offering a more time-efficient solution for scenarios where both model performance and interpretability are crucial.

By including EBM in our analysis, we aimed to evaluate its predictive performance and efficiency in providing interpretable results. This comparison allows us to assess the trade-offs between post hoc explainability methods like SHAP and inherently interpretable models like EBM in the context of intrusion detection systems, considering both predictive power and the time required to obtain explanations.

4.2. Model Selection and Rationale

The choice of models for this study was driven by several key factors, including the unique advantages of Tree SHAP, the performance of boosted models on imbalanced datasets, and the need to balance explainability with performance. This approach aims to mitigate the trade-off between model performance and explainability, addressing a key challenge highlighted in the survey by (Zhang et al., 2022) on explainable AI for cybersecurity.

Tree SHAP emerged as a crucial factor in our model selection. Its ability to provide fast, consistent, and theoretically sound explanations for tree-based models made it an ideal choice for our explainability needs. Most importantly, Tree SHAP offers true-to-the-model explanations, meaning they directly reflect the model's decision-making process rather than approximating it. This feature is critical for understanding how our models function, especially in the sensitive domain of intrusion detection.

Nearly all the models we selected, except for the Decision Tree, are boosted. This choice was deliberate and essential, especially given the imbalanced nature of our datasets. Boosted models, such as Random Forest, LightGBM, XGBoost, and CatBoost, have demonstrated superior performance on imbalanced datasets. This is crucial for our case with NFS-2023-TE, where some categories have tiny sample sizes. Combining these boosted models with Tree SHAP allows us to leverage their high performance while maintaining true-to-the-model explainability.

The imbalanced nature of our datasets significantly influenced our model selection. While we considered deep learning models, they struggled with the imbalanced data. Moreover, the additional complexity of explaining deep learning models and the increased explanation time made them less suitable for our study. We decided against using oversampling techniques to maintain the integrity of the original data distribution.

We included simpler models like Decision Trees to provide a baseline for comparison. This allows us to assess the trade-off between model complexity and performance. The Decision Tree, combined with Tree SHAP, also offers insights into the dataset structure and model behavior, with explanations that are particularly easy to interpret due to the model's simplicity. We tested other classic machine learning models such as SVM, Logistic Regression, and KNN. However, these models did not provide satisfactory results on our datasets, further justifying our focus on tree-based and boosted models.

The Explainable Boosting Machine (EBM) model was included as it promises to deliver performance comparable to complex models while maintaining inherent explainability. This aligns with our goal of

balancing performance and explainability and provides an interesting contrast to the post-hoc explanations of Tree SHAP.

Based on these considerations, we chose the following models for our study: Decision Tree (DT) as a baseline model and for its inherent interpretability; Random Forest (Breiman, 2001), a powerful ensemble method known for its robustness; LightGBM (Ke et al., 2017), an efficient gradient boosting framework; XGBoost (T. Chen & Guestrin, 2016), another popular and highly effective boosting algorithm; CatBoost (Dorogush et al., 2017), known for its performance and handling of categorical features; and Explainable Boosting Machine for its balance of performance and inherent explainability.

These models, coupled with Tree SHAP for explanation (except for EBM, which provides inherent explanations), allow us to thoroughly explore the trade-off between model performance and explainability in intrusion detection systems. The true-to-the-model nature of Tree SHAP explanations ensures that we can trust the insights gained from our models, which is crucial in a security-critical domain like intrusion detection. This approach enables us to address the challenges of imbalanced datasets while providing meaningful, accurate, and efficient explanations, crucial for the practical application of machine learning in cybersecurity.

4.3. Data Preprocessing and Sampling Methodology

To address the challenge of imbalanced datasets and improve our model performance, we implemented a careful under-sampling strategy for both the HIKARI-2021 and NFS-2023-TE datasets. This approach was chosen to balance the classes while maintaining the integrity of the original data distribution.

For the HIKARI-2021 dataset, we reduced all classes to 7,988 samples each, a number chosen based on the size of the smallest class in the dataset. We also merged the Benign and Background traffic classes to create a single non-malicious category, simplifying the focus on distinguishing between normal and malicious traffic. The NFS-2023-TE dataset underwent a similar process, with most classes reduced to 738 samples each. This number was determined through experimentation as the optimal balance between class representation and overall model performance. Classes with fewer than 738 samples were left unchanged to preserve all available information for rare attack types.

We acknowledge that under-sampling results in a significant reduction of data, which is a limitation of this study. However, we deemed this preferable to removing entire attack classes, which would have limited the model's ability to detect a full range of attacks. Given the extreme imbalance in NFS-2023-TE (with the smallest class having only 11 samples), we decided against oversampling to avoid the risk of overfitting on synthetic data, which could lead to unreliable model performance on real-world data.

Our training and validation methodology involved a stratified split, using 80% of the under-sampled data for training, ensuring that the class distribution was maintained in both training and test sets. We implemented 5-fold cross-validation to ensure robust model evaluation and mitigate the risk of overfitting. This methodology represents a balanced approach to handling the challenges presented by our imbalanced datasets. While it involves some compromise regarding data utilization, it allows us to train models that can recognize a full spectrum of attack types while maintaining a balanced view of the problem space. The use of cross-validation further strengthens the reliability of our results, helping to ensure that our models' performance is consistent across different subsets of the data.

4.4. Evaluation Metrics and Handling Class Imbalance in Multi-class NIDS

NIDS (Network Intrusion Detection System) datasets are typically multi-categorical, which indeed complicates the selection of performance evaluation metrics. This complexity arises from the need to choose among various metrics such as accuracy, recall, precision, and F1 score, while also determining how to obtain a global result that accurately represents the model's performance across all categories.

In multi-categorical datasets, it's possible to evaluate the model's effectiveness not just based on overall correct versus incorrect responses, but for each individual class as well. This granular approach allows for a more nuanced understanding of the model's performance, particularly when dealing with imbalanced datasets where some classes may have significantly fewer instances than others.

To address the challenges of multi-class evaluation, this study compares two methods of aggregating F1 scores: F1 Macro and F1 Weighted. F1 Macro calculates the arithmetic mean of F1 scores across all classes, giving equal importance to each class regardless of its frequency in the dataset. This method is particularly sensitive to performance on minority classes. In contrast, F1 Weighted computes a weighted average of F1 scores, with weights proportional to class frequencies. This approach gives more influence to classes with more samples, reflecting performance in relation to the dataset's actual distribution.

$$Macro\ F1 = \frac{1}{N}\sum_{i=1}^{N} F1_i$$

$$Weighted\ F1 = \sum_{i=1}^{N} w_i \times F1_i$$

Where N is the number of classes, i represents a specific class, and w_i is the weight of a class in the dataset.

These metrics offer valuable insights into model performance, especially in imbalanced datasets. F1 Macro provides a clearer picture of performance on rare attacks, while F1 Weighted might indicate high overall performance even if rare attacks are misclassified, as long as common attacks are correctly identified. For the balanced HIKARI 2021 dataset (post-undersampling), these metrics yield identical results since the weight of all the classes is the same. However, for the imbalanced NFS-2023-TE dataset, a high F1 Weighted score coupled with a lower F1 Macro score would suggest strong performance on common attacks but potential weaknesses in detecting rare ones. Conversely, similar F1 Macro and Weighted scores indicate consistent performance across all classes.

In terms of overall model assessment, F1 Macro offers a more stringent evaluation across all attack types, while F1 Weighted reflects performance as it would be experienced in a real-world scenario given the actual class distribution. A significant discrepancy between these scores can guide improvement efforts: a lower F1 Macro suggests the need to enhance the detection of minority classes, while low scores in both metrics indicate overall performance issues.

By employing this comprehensive evaluation strategy, th study aims to provide a nuanced and accurate assessment of model performance across all attack types. This approach mitigates the risks associated with oversimplified metrics and ensures that the evaluation aligns with the real-world requirements of

intrusion detection systems. In such systems, the ability to detect every type of attack, regardless of its frequency, is crucial for maintaining robust network security.

The combination of F1 Macro and F1 Weighted, supplemented by additional analyses, allows for a balanced assessment of overall effectiveness in real-world scenarios while addressing the critical requirement of detecting all types of network intrusions. This holistic approach not only provides a more accurate picture of model performance but also aligns closely with the practical needs of cybersecurity professionals tasked with protecting networks against a diverse array of potential threats.

5. RESULTS

5.1. DATA SET ANALYSIS

Our analysis of the NFS-2023-TE and HIKARI-2021 datasets revealed essential insights into their quality and potential limitations for intrusion detection research. While our initial examination of NFS-2023-TE did not identify any significant new flaws, it is essential to note that this does not guarantee the absence of issues. Some inherent problems from CIC-IDS 2017, such as outdated attack types and class imbalance, persist in NFS-2023-TE. These issues can only be fully addressed by completely redoing the data collection process, which was beyond the scope of the authors' aim to improve upon CIC-IDS 2017 rather than create an entirely new dataset from scratch.

Acknowledging that our analysis was primarily based on issues previously identified in other datasets is crucial. Consequently, NFS-2023-TE may have additional problems we did not uncover in this initial examination. Further in-depth analysis might reveal new concerns specific to this dataset.

Our analysis of HIKARI-2021 is noteworthy as it represents the first comprehensive examination of this dataset. By applying the criteria used to identify issues in CIC-IDS 2017, we discovered some problems in HIKARI-2021. Key findings include TCP flag anomalies, with some flows showing up to 140 FIN and 110 RST packets. This is a significant deviation from expected TCP behavior, where typically, only one RST or up to two FIN packets should be present per connection.

We also observed extended flow durations, with the longest observed flow lasting approximately 4.9 hours (17,942 seconds), which exceeds the stated 3-5 hour capture sessions. This suggests the flow generation tool did not implement proper timeouts or connection closure mechanisms. Unlike NFS-2023-TE, which closes connections after 120 seconds, HIKARI-2021 lacks a consistent connection termination policy. This leads to unrealistic flow statistics and potential misrepresentation of network behavior.

When filtering for more realistic connection parameters (duration < 181 seconds, RST < 2, FIN < 3), we found only 104 samples for bruteforce and 59 for bruteforce-XML attacks. This severely limits the dataset's utility for training robust intrusion detection models. The absence of detailed documentation on the labeling process and the unavailability of timestamps make it challenging to fully validate the dataset's integrity.

Analysis of the pcap files revealed that 0.03% of the packets were duplicated, and 0.23% were out of order. While these percentages are small, they indicate a lack of preprocessing that could impact the dataset's accuracy. These findings highlight the importance of thorough dataset validation and the need for continuous improvement in dataset creation for intrusion detection research.

5.2. MODEL EVALUATION AND PERFORMANCE ANALYSIS

It is crucial to preface this section by emphasizing its specific role within the broader context of this thesis. The primary purpose of this model evaluation and performance analysis is twofold: firstly, to demonstrate that these models can achieve good performance on the given datasets, and secondly, to maintain transparency regarding our fine-tuning process.

However, it is equally important to note that we are fully aware of the limitations of these datasets for real-world applications. Given this understanding, the objective of this analysis is not to propose models for practical use but rather to provide a basis for comparison with existing works in the field.

That said, even this comparative aim presents challenges. During our literature review, we observed that nearly all analyzed works employed different preprocessing techniques, such as using only a portion of the dataset or eliminating certain classes. These varied approaches make direct and accurate comparisons difficult, if not impossible.

Therefore, while we present detailed performance metrics and analysis, readers should interpret these results within the context of academic exploration rather than as indicators of real-world efficacy. The following evaluation serves primarily as a demonstration of model capabilities on these specific datasets and as a transparent account of our methodology rather than as a proposal for practical implementation.

We evaluated several machine learning models on two datasets: NFS-2023-TE and HIKARI-2021. Our analysis focused on training times, prediction times, and F1 scores (both macro and weighted). All experiments were conducted on a Dell XPS 13 9315 with a 12th gen i7-1250u and 16GB of DDR5 RAM, running Fedora 39 with Linux 6.8. We used Sklearnex and daal4py to optimize inference times, and all models were configured to utilize all available cores.

This introduction clarifies the purpose and limitations of this section, emphasizing its role in academic comparison and methodological transparency rather than real-world application. It also highlights the challenges in directly comparing other works due to varied preprocessing approaches.

The parameter tuning in Table *5.1* was conducted following the suggestions in the documentation for each library and by observing what worked well across models. We used the F1 macro metric from sklearn to evaluate results, and when different parameters yielded similar results, we favored those that led to shorter training times

Table 5.1 - NFS-2023-TE and NFS-2023-nTe parameters

mode	parameter	value
Random forest	n_estimators	10
	max_depth	14
	max_features	None
	bootstrap	False
catboost	iterations	40
	depth	11
	learning_rate	0.4
	loss_function	MultiClass
ebm	learning_rate	1
lightgbm	objective	MultiClass
	num_class	15
	learning_rate	0.01
	num_iterations	250
	max_depth	4
	num_leaves	6

xgboost	n_estimators	60
	max_depth	10
	objective	multi:softprob
	learning_rate	0.3
Dt	max_depth	14

Table 5.2 presents the training time, explanation time (using Fast Tree SHAP), and total time for each model based on 6,087 samples. Key observations include: EBM has the longest training time but is competitive regarding total time when including explanations; Random Forest and Decision Tree are the fastest, likely due to their simpler structures and the Random Forest using only 10 trees; CatBoost is surprisingly slow despite using fewer trees than LightGBM and XGBoost. These times show some variance but indicate the order of magnitude necessary to run each model. It's worth noting that while EBM is the slowest during training, it can be faster than LightGBM and XGBoost when considering the explanation time.

Table 5.2 - NFS-2023-TE training times

model	fit ms	explanation ms	total ms
ebm	76000	0	76000
dt	76	606	682
rf	92	762	854
catboost	51400	10900	62300
lightgbm	4600	1793000	1797600
xgboost	4890	581000	585890

Table *5.3* shows the average F1 scores (macro and weighted) from 5-fold cross-validation and prediction times for 1,522 samples. For the prediction times, we used the %timeit magic function of IPython to compute the mean of different runs. This approach is crucial because it provides more statistically significant results. With such small durations, a warmup of the function is necessary to reduce computation time. For instance, without the warmup run, the classification time of the decision tree was 3 ms because the first run included the time to load the prediction function into memory.

Key findings from this analysis include: EBM achieves the highest F1 macro score but the lowest F1 weighted score, indicating strong performance on minority classes but potential issues with majority classes; LightGBM offers the best balance of F1 scores but has the slowest prediction time; Random Forest presents a good compromise, with near-top F1 scores and faster prediction times.

The discrepancy in EBM's performance highlights the challenge of class imbalance in the dataset. Comparing these results with the decision tree shows that the decision tree has problems with Heartbleed and Web Attack - SQL Injection, misclassifying two samples for the first and 4 for the latter. In contrast, EBM misclassifies 111 samples in the DoS Slowhttptest category. These results expose the need for a balanced dataset. We can argue that EBM is the best because it can handle each attack well, but on the other hand, building a model over just 11 samples will likely lead to something that will not work in the real world.

LightGBM's slow prediction time could be problematic for real-time intrusion detection systems, especially during DoS attacks. It is twice as slow as the Random Forest, which scores nearly the same in the F1 metrics. The classification time is crucial because, in case of a DoS attack that opens and closes connections at a fast enough speed, the model needs to keep up with each new flow generated; otherwise, the NIDS will run out of service. Another reason this classification time is important is that it can make a difference between enabling or not the use on the edge and lead to less expensive devices when a GPU is not required for running an NIDS.

Given these considerations, while LightGBM shows the best F1 scores, the Random Forest emerges as the best overall alternative, balancing accuracy and speed effectively.

Table 5.3- NFS-2023-TE – average F1 score of 5-fold

model	prediction µs	F1 macro	F1 weighted
ebm	6610	96,0%	97,0%
dt	920	92,3%	98,4%
rf	2390	94,4%	98,5%
catboost	2730	94,2%	98,5%
lightgbm	56600	94,5%	98,6%
xgboost	15600	93,7%	98,8%

For HIKARI-2021, Table *5.4* demonstrates how the parameters lead to less complex models, except for EBM. While balanced datasets typically allow for simpler model structures, the extreme simplicity of these parameters is notable. This level of simplicity, achieving high performance with such basic configurations, raises questions about the complexity and quality of the HIKARI-2021 dataset. Our analysis will further explore these concerns using explainability algorithms, which will provide deeper insights into the models' decision-making processes and the dataset's underlying structure.

Table 5.4 - HIKARI 2021 parameters

model	parameter	value
cat	iterations	10
	depth	8
	learning_rate	0.6
	loss_function	MultiClass
dt	max_depth	8
ebm	default	
lightgbm	objective	MultiClass
	num_leaves	16
	n_estimators	5
	max_depth	6
rf	n_estimators	5
	max_depth	10
	max_features	None
	bootstrap	False

xgb	n_estimators	3
	max_depth	8
	learning_rate	1
	objective	multi:softprob

Table 5.5 presents the training and explanation times for HIKARI-2021. Notable points include: Despite using a larger training set (31,952 samples) compared to NFS-2023-TE (6,087 samples), HIKARI-2021 generally shows lower training times. This efficiency can be attributed to two main factors: the balanced nature of the dataset after our under-sampling preprocessing step, and the inherent simplicity of the HIKARI-2021 dataset itself, which may lack the complexity typically found in real-world network intrusion scenarios. EBM is the exception, with increased training time due to maintaining its complex structure. Unlike the other models, building a smaller EBM model through fine-tuning without sacrificing performance was impossible.

Table 5.5- HIKARI-2021 training times for 31,952 training samples

model	fit ms	explanation ms	total ms
ebm	145000	0	145000
dt	888	942	1830
rf	98	1140	1238
catboost	668	1490	2158
lightgbm	236	11000	11236
xgboost	283	79000	79283

Table 5.6 shows the model performance for HIKARI-2021. Key insights include: All models perform similarly regarding F1 scores, suggesting the dataset might be easier to classify after balancing; The Decision Tree is the fastest model while maintaining competitive performance.

The similarity in F1 scores across models is particularly interesting. For this balanced version of HIKARI-2021, the additional complexity of ensemble methods and boosting algorithms may not provide significant advantages over simpler models like the Decision Tree. These results should be interpreted cautiously, considering the previously discussed limitations of HIKARI-2021, including issues with its creation process and test bed simplification. Our findings highlight a critical issue: the lack of sufficiently complex, real-world datasets for intrusion detection research.

Given the already good performance of simpler models like decision trees, we argue that training complex deep learning models on these datasets should be avoided. Such efforts are likely to be unproductive and potentially misleading. Instead, researchers should redirect their focus towards improving the datasets themselves. Future work should concentrate on developing methods to create more realistic and challenging datasets. Our approach of using explainability techniques to uncover dataset flaws is one example of how researchers can contribute to this goal. By focusing on dataset quality and representativeness, we can pave the way for more meaningful advancements in intrusion detection systems that are truly applicable to real-world scenarios.

Table 5.6 - HIKARI-2021 F1 score and times

model	prediction µs	5-fold F1
ebm	21	96,8%
dt	2	96,6%
rf	3	96,6%
catboost	6	96,6%
lightgbm	7	96,5%
xgboost	11	96,7%

5.3. *MODEL EXPLANATIONS*

In this section, we explore the use of SHAP (SHapley Additive exPlanations) to analyze the NFS-2023-TE and HIKARI-2021 datasets. Our primary goal is to leverage SHAP to uncover potential weaknesses in these datasets and assess their suitability for real-world applications. We focused our analysis on decision trees and CatBoost models, as they provided distinct and insightful explanations.

The decision to limit our detailed analysis to these two models was deliberate. Decision trees were chosen for their inherent interpretability, allowing us to examine the underlying tree structure. CatBoost, on the other hand, was selected because its feature importance often diverged significantly from that of the decision tree, providing a valuable contrasting perspective.

It is important to note that while we examined SHAP explanations for all classes in both datasets, we have chosen to present only the most significant and illustrative examples here. This approach aligns with our objective of identifying critical weaknesses in the datasets; even a single problematic explanation can raise valid concerns about the dataset's integrity and real-world applicability

Figure 5.1 shows the feature importance of a decision tree for the class bot of NFS-2023-TE. The model has misclassified only 1 sample out of 737 using only a small subset of the available features. The same pattern of using a few features applies to the other attacks. The only exception is benign traffic, which requires more features to be detected. The benign class assigns positive importance to more features because most of the split leads to either attack or benign traffic.

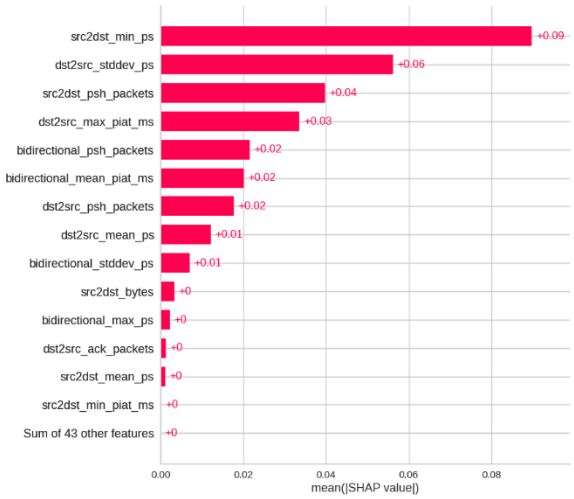

Figure 5.1 - NFS-2023-TE - decision tree feature importance with shap of bot class

Figure 5.2 illustrates the feature importance for the bruteforce-XML class in the HIKARI-2021 dataset, as determined by a decision tree model using SHAP values. The chart reveals a striking imbalance in feature importance, with 'fwd_pkts_payload.max' dominating the model's decision-making process. Upon closer examination of the decision tree structure, we find that just the first two nodes of the tree are sufficient to achieve a remarkably low gini coefficient of 0.041 for this class. Specifically, if we follow the false path from the root node (which likely corresponds to 'fwd_pkts_payload.max' ≤ 737) and then proceed to the second node (likely 'fwd_pkts_payload.max' ≤ 761.5), we reach a point where the tree achieves this high level of classification accuracy. Using primarily these two decision points, the model correctly classifies 6,388 out of 6,526 samples.

While this demonstrates the model's effectiveness on this particular dataset, it raises important questions about the dataset's representation of real-world attack scenarios. The ability to achieve such high accuracy with just two decision points based on a single feature is unusual for complex network attacks. In real-world environments, attacks typically exhibit more varied characteristics that require more complex decision boundaries for accurate detection.

This finding underscores the importance of critically evaluating datasets used for intrusion detection research. While the model performs exceptionally well on this data, its reliance on such a simple decision boundary may limit its ability to detect more sophisticated or varied attacks in real-world applications. It highlights the need for datasets that capture a more diverse range of attack behaviors to train models that are more robust and applicable to real-world cybersecurity challenges.

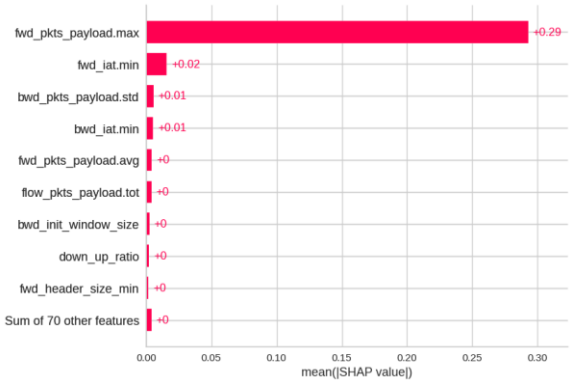

Figure 5.2 - HIKARI-2021 - decision tree feature importance with shap of bruteforce-XML class

Table 5.7 presents the quintile distribution of the 'fwd_pkts_payload.max' feature for different categories in the HIKARI-2021 dataset. This feature represents the maximum payload size of forward packets in a network flow. The table provides valuable insights into why this feature is crucial for detecting bruteforce-XML attacks. The bruteforce-XML category shows a very narrow distribution, with 50% of the samples having values between 746 and 748 bytes. This tight clustering is highly unusual for real-world network traffic. The decision tree's root node splits at 737 bytes, and the subsequent false branch splits at 761.5 bytes. These split points align perfectly with the bruteforce-XML distribution, explaining why this feature is so effective for classification. Other categories show much wider distributions. For instance, background traffic ranges from 0 to 15,741 bytes, and benign traffic from 0 to 3,456 bytes. This stark contrast makes the bruteforce-XML traffic stand out. In real-world scenarios, attack traffic rarely exhibits consistent payload sizes. The uniformity of the bruteforce-XML category suggests this might be an artifact of the dataset generation process rather than a true representation of attack behavior. A model trained on this dataset would be highly susceptible to evasion. An attacker could easily bypass detection by slightly altering their payload size to fall outside the narrow range identified for bruteforce-XML attacks.

This analysis highlights a significant limitation of the HIKARI-2021 dataset. The unrealistic distribution of the 'fwd_pkts_payload.max' feature for bruteforce-XML attacks could lead to developing intrusion detection systems that perform well on this dataset but fail in real-world applications. It underscores the importance of critically evaluating datasets for training security models and the need for more diverse and realistic network traffic data in IDS research.

Table 5.7 - fwd_pkts_payload.max quintiles

category	min	25%	50%	75%	max
Background	0	0	40	127	15741
Benign	0	0	36	426	3456
Bruteforce	0	357	373	425	786
Bruteforce-XML	0	746	747	748	925
Probing	517	517	517	517	517
XMRIGCC CryptoMiner	40	40	50	50	232

In our analysis, we closely examined the features of the XMRIGCC CryptoMiner attack, as well as the probing attack, using decision tree models and SHAP values. Figure *5.3* illustrates the feature importance for the XMRIGCC CryptoMiner attack, revealing that only a select few features are utilized by the model. The most crucial feature for this attack is 'bwd_header_size_min', which consistently has a value of zero for this attack type. Following this, we find 'down_up_ratio' as the second most important feature.

Transitioning to Figure *5.4*, we see a similar analysis for the probing attack. Interestingly, 'down_up_ratio' emerges as the most critical feature for classifying this attack type as well. This connection between the two attack types through a common important feature prompted us to delve deeper into the 'down_up_ratio' characteristic across different attack categories.

Table 5.8 provides this deeper insight, presenting the mean and standard deviation of the 'down_up_ratio' feature for various attack types and benign traffic. This table reveals several important points that tie our observations together. First, it confirms that the XMRIGCC CryptoMiner attack has a consistent 'down_up_ratio' of zero, explaining its importance in Figure 5.3. For the probing attack, we observe a mean value of around 1.3 with a notably low standard deviation compared to other attack types. This distinct pattern explains why 'down_up_ratio' is so crucial for identifying probing attacks, as shown in Figure *5.3*.

Interestingly, Table 5.8 also shows that 'down_up_ratio' is the second most important feature for classifying background traffic. However, the reason for this is not immediately apparent from the statistics provided, highlighting the complex interactions between features in network traffic classification.

By presenting these three elements together - Figure *5.3* and Figure *5.4*, along with Table 5.8 - we provide a comprehensive view of how a single feature can play diverse roles in identifying different types of network activities. This interconnected analysis demonstrates the nuanced nature of feature importance in intrusion detection systems and underscores the value of using explainable AI techniques like SHAP to uncover these intricate relationships.

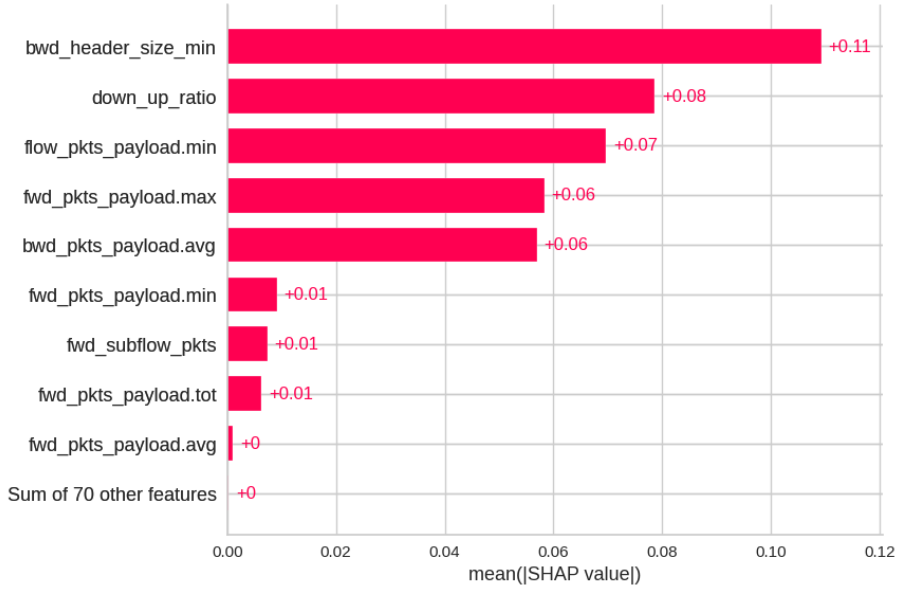

Figure 5.3- HIKARI-2021 - decision tree feature importance with shap of XMRIGCC CryptoMiner

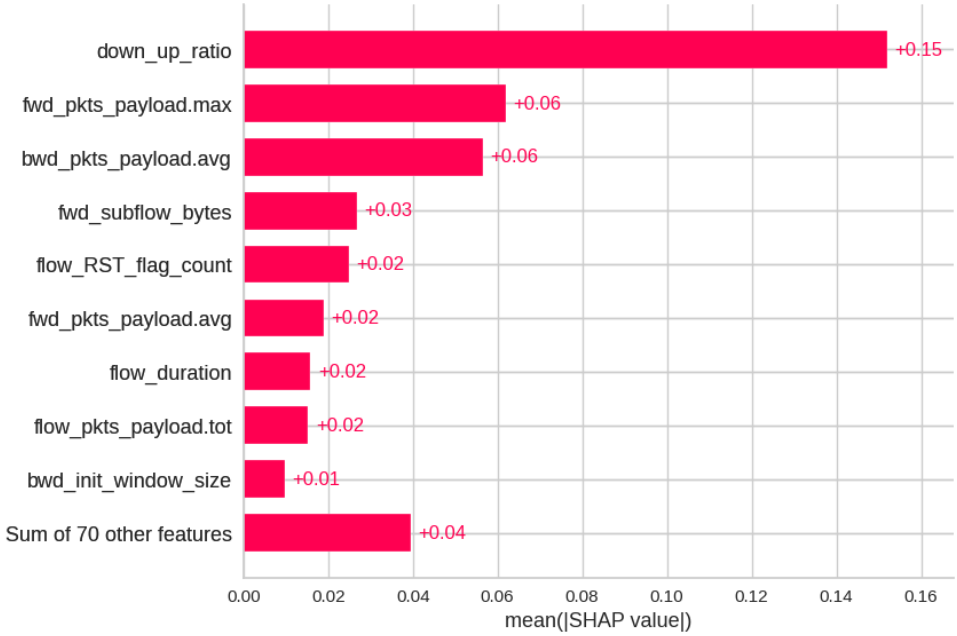

Figure 5.4- HIKARI-2021 - feature importance with shap of probing attack of decision tree

Table 5.8 - Down_up_ration mean and std

category	mean	std
Background	0.810244	0.721923
Benign	2.015250	32.321667
Bruteforce	8.794107	89.440773

Bruteforce-XML	20.071672	128.065872
Probing	1.298040	0.119266
XMRIGCC CryptoMiner	0.000000	0.000000

Before delving into the specific findings, it's important to understand how to interpret the SHAP beeswarm plot shown in Figure 5.5. This visualization technique, known as a SHAP (SHapley Additive exPlanations) beeswarm plot, illustrates the impact of various features on the model's predictions.

SHAP beeswarm plots can be configured to show either the average impact or the maximum impact of features. The average impact relates to how a feature typically influences the model's decision when the feature remains within its usual range. In contrast, the maximum impact shows how much a feature can affect the model's decision when it reaches its extreme values (maximum or minimum). This distinction is crucial as it can reveal features that may not be impactful under normal circumstances but can significantly alter the model's outcome in rare cases.

In this analysis, we've chosen to display the maximum impact to highlight the most significant potential influence of each feature, including those that might be overlooked when considering only average impacts.

In this plot, each row represents a feature, with the features sorted by their maximum impact on the model's output. The most impactful features appear at the top of the plot. Each dot represents a sample from the dataset. The horizontal position of a dot shows the SHAP value - the impact of that feature for that sample on the model's prediction. Dots to the right increase the likelihood of the prediction, while those to the left decrease it. The color of each dot indicates the feature's value, with red representing high values and blue representing low values.

The spread of dots for each feature demonstrates the range of impacts that feature can have across different samples. Features with wider spreads tend to have more significant potential effects on the model's decisions, especially at their extreme values.

With this understanding of how to interpret the SHAP beeswarm plot and the significance of maximum impact analysis, we can now examine the specific patterns revealed by the SHAP analysis of the CatBoost algorithm on the NFS-2023-TE dataset.

Figure 5.5 illustrates the feature importance analysis of the CatBoost algorithm across various attack types in the NFS-2023-TE dataset. A striking pattern emerges: the 'bidirectional_fin_packets' feature consistently ranks as one of the most influential factors for several attack categories, including Bot, DDoS, DoS Slowhttptest, DoS Slowloris, Heartbleed, Infiltration, and Web Attack - XSS.

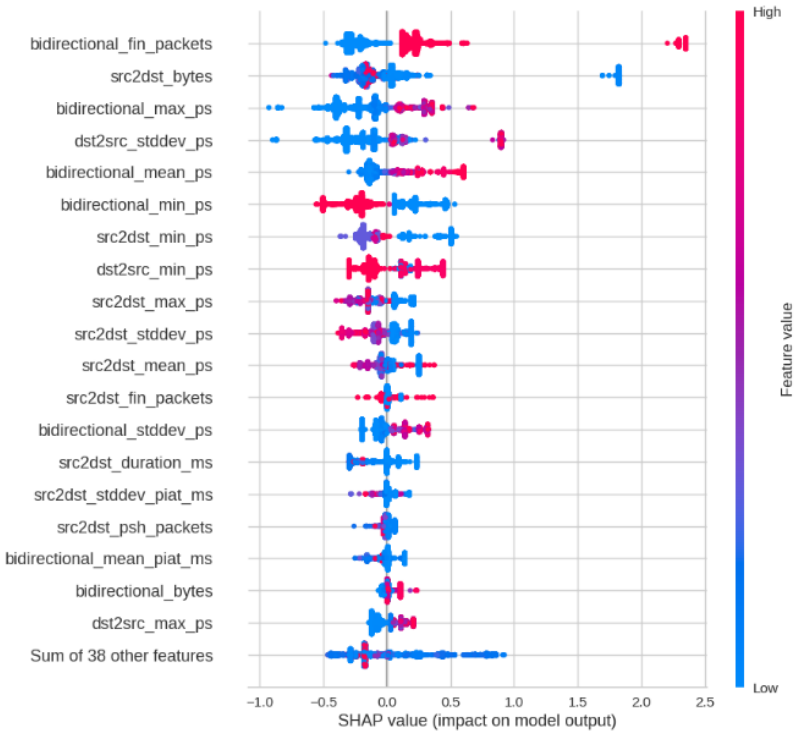

Figure 5.5 - NFS-2023-TE - catboost impact on the model output of DDoS with shap sorted by highest magnitude of impact.

Table 5.9 shows the statistical analysis of the 'bidirectional_fin_packets' feature, which counts the number of FIN packets in a bidirectional connection and shows distinct patterns across different attack types. We observe exactly 1 FIN packet for Bot and Web Attack- XSS in all instances. Similarly, DDoS attacks show 1 FIN packet in 99% of cases. Other attack types (DoS Slowhttptest, DoS Slowloris, Heartbleed, and Infiltration) predominantly show 0 FIN packets, with this pattern occurring at the 85th percentile of their distributions.

Table 5.9 - NFS-2023-TE bidirectional_fin_packets statistical analysis

	mean	**std**	**25%**	**50%**	**75%**
BENIGN	0.306362	0.460982	0.0	0.0	1.0
Bot	1.000.000	0.000000	1.0	1.0	1.0
DDoS	0.999739	0.016162	1.0	1.0	1.0
DoS GoldenEye	0.935203	0.246183	1.0	1.0	1.0
DoS Hulk	0.999823	0.013288	1.0	1.0	1.0
DoS Slowhttptest	0.242679	0.428781	0.0	0.0	0.0
DoS slowloris	0.082435	0.275052	0.0	0.0	0.0
FTP-Patator	0.689690	0.462650	0.0	1.0	1.0
Heartbleed	0.090909	0.301511	0.0	0.0	0.0
Infiltration	0.071429	0.262265	0.0	0.0	0.0
PortScan	0.001956	0.044186	0.0	0.0	0.0

SSH-Patator	0.989597	0.101479	1.0	1.0	1.0
Web Attack - Brute Force	1.000.000	0.000000	1.0	1.0	1.0
Web Attack - Sql Injection	1.000.000	0.000000	1.0	1.0	1.0
Web Attack - XSS	1.000.000	0.000000	1.0	1.0	1.0

To provide further context, we examined the connection durations for these attack types, as presented in Table *5.10*. It is important to note that these durations are in milliseconds. Heartbleed and Infiltration have very long mean durations (110,680 ms and 71,705 ms respectively), with 75% of connections lasting over 119,260 ms and 105,616 ms. This supports the hypothesis that these connections often terminate due to timeouts rather than normal closure, as they last for nearly two minutes or more. DoS Slowhttptest and DoS Slowloris show widely varying durations. Slowhttptest has a mean of 11,832 ms, but 50% of connections last under 6 ms. Slowloris is even more extreme, with a mean of 36,721 ms, but 50% of connections lasting 0 ms. This high variability, with many connections lasting mere milliseconds, could explain the lack of consistent FIN packets. Bot attacks have relatively short durations (mean 262 ms), which aligns with their consistent use of FIN packets for closure. These brief connections suggest rapid, automated interactions. DDoS attacks show moderate durations (mean 700 ms), suggesting more prolonged but still relatively short connections.

Table 5.10 - NFS-2023-TE - bidirectional_duration_ms

	mean	25%	50%	75%	max
BENIGN	8.455.942.641	0.00	23.0	226.00	119999.0
Bot	261.727.642	68.00	76.0	95.00	61003.0
DDoS	699.695.919	103.00	593.0	1180.00	2236.0
DoS GoldenEye	11.028.957.812	10022.00	11312.0	11793.00	106793.0
DoS Hulk	674.306.605	63.00	148.0	182.00	110889.0
DoS Slowhttptest	11.832.117.496	0.00	6.0	3004.00	119800.0
DoS slowloris	36.721.484.592	0.00	0.0	102650.00	105745.0
FTP-Patator	4.520.066.432	0.00	4099.0	8996.00	10780.0
Heartbleed	110.679.636.364	119259.50	119261.0	119297.50	119303.0
Infiltration	71.705.357.143	47004.25	79873.0	105616.25	119992.0
PortScan	2.512.117	0.00	0.0	0.00	11012.0

SSH-Patator	12.131.148.322	11634.00	12109.5	12868.50	19582.0
Web Attack - Brute Force	20.656.476.821	8801.50	9856.0	34065.00	35452.0
Web Attack - Sql Injection	5.023.083.333	5006.75	5009.5	5022.25	5087.0
Web Attack - XSS	46.164.407.407	6155.50	67004.0	68090.00	70204.0

This pattern raises important questions about the dataset's representation of real-world network behavior and the potential for creating robust intrusion detection systems. The consistent importance of this single feature across multiple attack types suggests it may be an artifact of the dataset generation process rather than a true indicator of attack behavior. Future research should investigate whether this feature's prominence could lead to overfitting or vulnerabilities in real-world applications, and explore how it interacts with other important features in the classification process.

Our analysis revealed several key insights. Both datasets showed a significant imbalance in feature importance. For instance, in HIKARI-2021, the 'fwd_pkts_payload.max' feature dominated the decision-making process for the bruteforce-XML class (Figure 5.2). This over-reliance on a single feature raises concerns about the model's robustness in real-world scenarios. The analysis revealed unrealistically consistent patterns for certain attack types. In HIKARI-2021, the bruteforce-XML attacks showed a very narrow distribution of the 'fwd_pkts_payload.max' feature (Table 5.7), which is unlikely in real-world attacks. Similarly, in NFS-2023-TE, the 'bidirectional_fin_packets' feature showed unexpected consistency across multiple attack types (Table 5.9). Some attack classes could be easily identified using just one or two features. For example, the bruteforce-XML class in HIKARI-2021 could be classified with 97.8% accuracy using just two decision points on a single feature. This simplicity is concerning for a dataset meant to represent complex network attacks.

The consistent importance of certain features across multiple attack types, such as 'bidirectional_fin_packets' in NFS-2023-TE, suggests these might be artifacts of the dataset generation process rather than true indicators of attack behavior. Analysis of connection durations (Table 5.10) in NFS-2023-TE revealed patterns that may not accurately represent real-world network behavior, particularly for attacks like Heartbleed and Infiltration with unusually long durations. The models relied on a small subset of available features for many attack classes. This is evident in both datasets and across different model types (decision trees and CatBoost), suggesting a potential lack of complexity in the dataset's representation of attacks. The simplistic nature of some attack signatures, such as the XMRIGCC CryptoMiner in HIKARI-2021 being identifiable by specific values of just two features, suggests that models trained on these datasets could be easily evaded in real-world scenarios.

These findings highlight the limitations of both NFS-2023-TE and HIKARI-2021 for developing robust, real-world intrusion detection systems. While they represent improvements over previous datasets, they still fail to accurately represent the complexity and variability of real-world network environments and attack scenarios. The use of SHAP in this analysis proved valuable for model interpretation and dataset validation, revealing specific features and decision boundaries that models rely on. This allows for critical assessment of whether these align with expert knowledge of network behavior and attack characteristics.

In conclusion, this analysis underscores the ongoing challenge of creating high-quality, realistic datasets for intrusion detection research. It emphasizes the need for continued refinement in dataset generation methodologies to capture better the complexity and diversity of real-world network traffic and attack patterns.

6. CONCLUSION

This work has critically analyzed various aspects of network intrusion detection systems (NIDS), focusing on datasets and methodologies used in current research. The study presents several key findings and contributions, starting with a thorough analysis of the Hikari-2021 dataset. This analysis identified significant issues such as improper handling of TCP connection terminations and class imbalance, highlighting the need for meticulous dataset examination before model development.

A major contribution of this study is the application of SHAP (SHapley Additive exPlanations) for model validation. This method enhances transparency by explaining model decisions clearly, ensuring that selected features are relevant to attack detection. The study also emphasizes the importance of using comprehensive evaluation metrics beyond simple accuracy. Metrics like the F1 score are crucial for a more accurate assessment, especially in imbalanced datasets.

The limitations of current datasets were highlighted, noting that while techniques like random stratified undersampling were employed to address class imbalance in Hikari-2021 and NFS-2023, these measures are insufficient. The need for more robust and representative datasets remains critical. Addressing the lack of reproducibility in many studies, this research ensured the release of source code and the use of up-to-date datasets. This approach promotes transparency and allows for the validation and comparison of different models.

The challenge of ensuring that models trained on public datasets perform well in real-world scenarios was recognized. This ongoing problem highlights the necessity for future research to bridge the gap between academic studies and practical applications. The main contributions of this study include an in-depth analysis of the Hikari-2021 dataset, identifying critical issues and underscoring the importance of thorough dataset examination. The introduction of SHAP for model validation enhances model transparency and explainability by providing deeper insights into decision-making processes. The comprehensive methodology promotes the use of diverse evaluation metrics and addresses the limitations of current datasets, though acknowledging the need for further improvements.

In conclusion, this study calls on the NIDS research community to shift focus from developing new models on flawed datasets to improving dataset quality and representativeness. The methodology provides a foundation for a standardized approach to NIDS research, emphasizing transparency, reproducibility, and practical applicability. By addressing these core issues, advancements can be made towards developing NIDS models that are both academically robust and practically effective in real-world cybersecurity contexts.

BIBLIOGRAPHICAL REFERENCES

Ahmetoglu, S., Che Cob, Z., & Ali, N. (2022). A Systematic Review of Internet of Things Adoption in Organizations: Taxonomy, Benefits, Challenges and Critical Factors. *Appl. Sci. 2022*, *12*, 4117. https://doi.org/10.3390/app12094117

Aouini, Z., & Pekar, A. (2022). NFStream: A flexible network data analysis framework. *Computer Networks*, *204*. https://doi.org/10.1016/j.comnet.2021.108719

Breiman, L. (2001). Random forests. In *Random Forests* (Vol. 45). https://doi.org/10.4324/9781003109396-5

Brownlee, N., Mills, C., & Ruth, G. (1999). *Traffic Flow Measurement: Architecture*.

Catillo, M., Pecchia, A., Rak, M., & Villano, U. (2021). Demystifying the role of public intrusion datasets: A replication study of DoS network traffic data. *Computers & Security*, *108*, 102341. https://doi.org/10.1016/j.cose.2021.102341

Catillo, M., Pecchia, A., & Villano, U. (2023). Machine Learning on Public Intrusion Datasets: Academic Hype or Concrete Advances in NIDS? *Proceedings - 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume, DSN-S 2023*, 132–136. https://doi.org/10.1109/DSN-S58398.2023.00038

Chauhan, R., & Shah Heydari, S. (2020). *Polymorphic Adversarial DDoS attack on IDS using GAN*.

check point. (2021). *CYBER ATTACK TRENDS Mid Year Report 2021*.

Chen, H., Janizek, J. D., Lundberg, S., & Lee, S.-I. (2020). *True to the Model or True to the Data?* https://github.com/slundberg/shap/blob/

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *13-17-August-2016*, 785–794. https://doi.org/10.1145/2939672.2939785

Dorogush, A. V., Ershov, V., & Yandex, A. G. (2017). *CatBoost: gradient boosting with categorical features support*. https://github.com/Microsoft/LightGBM

Engelen, G., Rimmer, V., & Joosen, W. (2021). Troubleshooting an Intrusion Detection Dataset: The CICIDS2017 Case Study. *Proceedings - 2021 IEEE Symposium on Security and Privacy Workshops, SPW 2021*, 7–12. https://doi.org/10.1109/SPW53761.2021.00009

Fernandes, R., & Lopes, N. (2022). Network Intrusion Detection Packet Classification with the HIKARI-2021 Dataset: a study on ML Algorithms. *10th International Symposium on Digital Forensics and Security, ISDFS 2022*, 1–5. https://doi.org/10.1109/ISDFS55398.2022.9800807

Fernandes, R., Silva, J., Ribeiro, O., Portela, I., & Lopes, N. (2023). The impact of identifiable features in ML Classification algorithms with the HIKARI-2021 Dataset. *ISDFS 2023 - 11th International Symposium on Digital Forensics and Security*, 1–5. https://doi.org/10.1109/ISDFS58141.2023.10131864

Ferriyan, A., Thamrin, A. H., Takeda, K., & Murai, J. (2021). Generating network intrusion detection dataset based on real and encrypted synthetic attack traffic. *Applied Sciences (Switzerland)*, *11*(17). https://doi.org/10.3390/app11177868

Gharib, A., Sharafaldin, I., Habibi, A., & Ghorbani, A. (2016). *An Evaluation Framework for Intrusion Detection Dataset*.

Goodman, B., & Flaxman, S. (2017). *European Union Regulations on Algorithmic Decision Making and a "Right to Explanation."*

Guerra, J. L., Catania, C., & Veas, E. (2022). Datasets are not enough: Challenges in labeling network traffic. *Computers & Security*, *120*, 102810. https://doi.org/10.1016/j.cose.2022.102810

Gümüşbaş, D., Yıldırım, T., Genovese, A., & Scotti, F. (2021). A comprehensive survey of databases and deep learning methods for cybersecurity and intrusion detection systems. In *IEEE Systems Journal* (Vol. 15, Issue 2, pp. 1717–1731). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/JSYST.2020.2992966

Hofstede, R., Čeleda, P., Trammell, B., Drago, I., Sadre, R., Sperotto, A., & Pras, A. (2014). Flow monitoring explained: From packet capture to data analysis with NetFlow and IPFIX. *IEEE Communications Surveys and Tutorials*, *16*(4), 2037–2064. https://doi.org/10.1109/COMST.2014.2321898

Janzing, D. (2019). *Causal Regularization*.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *31st Conference on Neural Information Processing Systems*. https://github.com/Microsoft/LightGBM.

Kurniabudi, Stiawan, D., Darmawijoyo, Bin Idris, M. Y. Bin, Bamhdi, A. M., & Budiarto, R. (2020). CICIDS-2017 Dataset Feature Analysis with Information Gain for Anomaly Detection. *IEEE Access*, *8*, 132911–132921. https://doi.org/10.1109/ACCESS.2020.3009843

Kwon, D., Neagu, R. M., Rasakonda, P., Ryu, J. T., & Kim, J. (2023). Evaluating Unbalanced Network Data for Attack Detection. *SNTA 2023 - Proceedings of the 2023 on Systems and Network Telemetry and Analytics*, 23–26. https://doi.org/10.1145/3589012.3594898

Lanvin, M., Gimenez, P. F., Han, Y., Majorczyk, F., Mé, L., & Totel, É. (2023). Errors in the CICIDS2017 Dataset and the Significant Differences in Detection Performances It Makes. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *13857 LNCS*, 18–33. https://doi.org/10.1007/978-3-031-31108-6_2

Lashkari, A. H., Gil, G. D., Mamun, M. S. I., & Ghorbani, A. A. (2017). Characterization of tor traffic using time based features. *ICISSP 2017 - Proceedings of the 3rd International Conference on Information Systems Security and Privacy*, *2017-Janua*, 253–262. https://doi.org/10.5220/0006105602530262

Lee, W., Stolfo, S. J., & Mok, K. W. (1999). *Mining in a Data-flow Environment: Experience in Network Intrusion Detection*.

Liu, J., Gao, Y., & Hu, F. (2021). A fast network intrusion detection system using adaptive synthetic oversampling and LightGBM. *Computers and Security*, *106*, 102289. https://doi.org/10.1016/j.cose.2021.102289

Liu, L., Engelen, G., Lynar, T., Essam, D., & Joosen, W. (2022). Error Prevalence in NIDS datasets: A Case Study on CIC-IDS-2017 and CSE-CIC-IDS-2018. *2022 IEEE Conference on Communications and Network Security, CNS 2022*, 254–263. https://doi.org/10.1109/CNS56114.2022.9947235

Lou, Y., Caruana, R., Gehrke, J., & Hooker, G. (2013). Accurate intelligible models with pairwise interactions. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *Part F1288*, 623–631. https://doi.org/10.1145/2487575.2487579

Louk, M. H. L., & Tama, B. A. (2023). Dual-IDS: A bagging-based gradient boosting decision tree model for network anomaly intrusion detection system. *Expert Systems with Applications*, *213*, 119030. https://doi.org/10.1016/j.eswa.2022.119030

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., & Lee, S. I. (2020). From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence*, *2*(1), 56–67. https://doi.org/10.1038/s42256-019-0138-9

Lundberg, S. M., Erion, G. G., & Lee, S.-I. (2018). *Consistent Individualized Feature Attribution for Tree Ensembles*. http://github.com/slundberg/shap

Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, *2017-Decem*, 4766–4775. https://github.com/slundberg/shap

Maseer, Z. K., Yusof, R., Bahaman, N., Mostafa, S. A., & Foozy, C. F. M. (2021). Benchmarking of Machine Learning for Anomaly Based Intrusion Detection Systems in the CICIDS2017 Dataset. *IEEE Access*, *9*, 22351–22370. https://doi.org/10.1109/ACCESS.2021.3056614

Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). *2015 Military Communications and Information Systems Conference, MilCIS 2015 - Proceedings*. https://doi.org/10.1109/MilCIS.2015.7348942

Nazar, M., Alam, M. M., Yafi, E., & Su'Ud, M. M. (2021). A Systematic Review of Human-Computer Interaction and Explainable Artificial Intelligence in Healthcare with Artificial Intelligence Techniques. In *IEEE Access* (Vol. 9, pp. 153316–153348). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ACCESS.2021.3127881

Noori, M. S., Sahbudin, R. K. Z., Sali, A., & Hashim, F. (2023). Feature Drift Aware for Intrusion Detection System using Developed Variable Length Particle Swarm Optimization in Data Stream. *IEEE Access*, *11*(September), 1–1. https://doi.org/10.1109/access.2023.3333000

Panigrahi, R., & Borah, S. (2018). A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems. *International Journal of Engineering and Technology(UAE)*, *7*(3.24 Special Issue 24), 479–482. https://www.researchgate.net/publication/329045441

Pekar, A., & Jozsa, R. (2024). *Evaluating ML-Based Anomaly Detection Across Datasets of Varied Integrity: A Case Study*. https://intrusion-detection.distrinet-research.be/WTMC2021/tools

Rajak, P., Lachure, J., & Doriya, R. (2022). CNN-LSTM-based IDS on Precision Farming for IIoT data. *Proceedings of 4th International Conference on Cybernetics, Cognition and Machine Learning Applications, ICCCMLA 2022*, 99–103. https://doi.org/10.1109/ICCCMLA56841.2022.9988997

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). *"Why Should I Trust You?" Explaining the Predictions of Any Classifier*. https://doi.org/10.1145/2939672.2939778

Sahakyan, M., Aung, Z., & Rahwan, T. (2021). Explainable Artificial Intelligence for Tabular Data: A Survey. *IEEE Access*, *9*, 135392–135422. https://doi.org/10.1109/ACCESS.2021.3116481

Sharafaldin, I., Gharib, A., Lashkari, A. H., & Ghorbani, A. A. (2017). Towards a Reliable Intrusion Detection Benchmark Dataset. *Software Networking*, *2017*(1), 177–200. https://doi.org/10.13052/jsn2445-9739.2017.009

Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSP 2018 - Proceedings of the 4th International Conference on Information Systems Security and Privacy*, *2018-Janua*(Cic), 108–116. https://doi.org/10.5220/0006639801080116

Shrikumar, A., Greenside, P., & Kundaje, A. (2019). *Learning Important Features Through Propagating Activation Differences*. http://goo.gl/qKb7pL,

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. *IEEE Symposium on Computational Intelligence for Security and Defense Applications, CISDA 2009*, *Cisda*, 1–6. https://doi.org/10.1109/CISDA.2009.5356528

TeleGeography. (2023). *Executive Summary, IP Networks Research Service*.

Walter, L., Denter, N. M., & Kebel, J. (2022). A review on digitalization trends in patent information databases and interrogation tools. *World Patent Information*, *69*, 102107. https://doi.org/10.1016/j.wpi.2022.102107

Xiong, F., Zang, L., & Gao, Y. (2021). *Internet penetration as national innovation capacity: worldwide evidence on the impact of ICTs on innovation development*. https://doi.org/10.1080/02681102.2021.1891853

Yang, J. (2021). *Fast TreeSHAP: Accelerating SHAP Value Computation for Trees*. http://arxiv.org/abs/2109.09847

Yulianto, A., Sukarno, P., & Suwastika, N. A. (2019). Improving AdaBoost-based Intrusion Detection System (IDS) Performance on CIC IDS 2017 Dataset. *Journal of Physics: Conference Series*, *1192*(1). https://doi.org/10.1088/1742-6596/1192/1/012018

Zavrak, S., & Iskefiyeli, M. (2020). Anomaly-Based Intrusion Detection from Network Flow Features Using Variational Autoencoder. *IEEE Access*, *8*, 108346–108358. https://doi.org/10.1109/ACCESS.2020.3001350

Zhang, Z., Hamadi, H. Al, Damiani, E., Yeun, C. Y., & Taher, F. (2022). Explainable Artificial Intelligence Applications in Cyber Security: State-of-the-Art in Research. *IEEE Access*, *10*(July), 93104–93139. https://doi.org/10.1109/ACCESS.2022.3204051

APPENDIX A

Source code available at: https://github.com/ludotosk/tesi

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa