University of Gloucestershire
School of Computing and Engineering
BSc in Cyber Security

Assessing the Reliability of Machine Learning for Encrypted Traffic Detection

My Student Name
S4221354

May 9, 2025

Abstract

Encrypted network traffic is becoming an increasing problem for intrusion detection systems (IDS) in that it masks payload data while permitting exploiters to bypass traditional signature-based defenses. This dissertation explores the following question: can encrypt-based threats be reliably detected by explainable, flow-level machine learning (ML) models, without deep-packet inspection? With the help of the publicly available CICIDS2017 dataset, we empirically assess and compare two tree-based classifiers: Random Forest and XGBoost, on standard metrics (precision, recall, F1-score) using a stratified 70/15/15 train/validation/test split. To make the process trustworthy, we incorporate interpretability techniques of SHAP and LIME: we look at feature importance rankings and local decisions explanations. Our findings indicate that XGBoost outperforms Random Forest by barely more F1-scores and by greater resistance to class imbalance, whereas, previously unforeseen, fetching the lower train times. Based on interpretability analysis, it is shown that ‘flow duration’, ‘packet size variance’ and ‘inter-arrival times’ are strong, consistent top predictors of malicious activity. Finally, we present limitations of the system, such as the absence of deep learning architectures, adversarial robustness tests, and live-attack simulations, and list future work related to hybrid models and ones in real time deployment frameworks. These results show that lightweight, interpretable ML techniques can be useful at first line in IDS for encrypted traffic while maintaining high levels of detection capability and transparency requirements that are needed in security-sensitive settings.

Dedicated to
my mum

Acknowledgements

I would like to thank Dr. Sepideh Mollahjafari for the invaluable counsel and encouragement provided as my advisor. Her incisive comments and support were invaluable to the completion of this dissertation.

Ethical Issues

All data used in this study are from publicly available sources, and all software tools and libraries employed are open-source, ensuring full transparency, reproducibility, and compliance with ethical research standards

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

In recent years, there was a tremendous transition to encrypted Internet communication; more than 70 percent of websites and over 94% of Google's web traffic utilize encryption protocols (Annual & Report, 2018; Google Transparency Report, 2023). While this development has strengthened user privacy and data security, it presents challenges for traditional Intrusion Detection Systems that relies on inspecting unencrypted packet contents.

With payloads becoming increasingly out of reach, researchers and practitioners have instead turned to machine learning to classify encrypted traffic based on metadata-based features such as flow duration, packet size and inter arrival time provided by (Anderson et al., 2016;Zhou et al., 2024). While the detection performance is usually benchmarked, very little attention has been paid to the interpretability of ML based IDS, particularly in the case of encrypted environments where human insight into decision rationale is crucial. Since security analysts can't validate encrypted content for themselves, it is key in the trust and operational deployment of the model to understand why a traffic flow is flagged as malicious.

This research aims to find the viability of application of traditional machine learning techniques for malicious activity detection in encrypted network traffic with focus on the model interpretability. The project focuses on explicable decision making with flow level metadata to allow security analysts to understand and trust the rationale for each detection.

1.2 Problem Statement

Although decision trees, logistic regression, random forests, and other traditional ML models are theoretically interpretable, only the accuracy of IDS has been largely addressed in existing IDS research while ignoring explainability. In the case of encrypted traffic, this is a trust gap. As analysts cannot ascertain what went into the model's decision making without access to the model, it is hard to know if the predicted behaviors are correct or if the model is making mistakes either intentionally or due to adversary actions.

This dissertation presents a machine learning based IDS for this problem that performs well while having high model interpretability. From an assessment viewpoint, the goal

is to examine if traditionally trained ML models are able to provide explanations for their predictions that are meaningful and transparent using today's tools such as the SHAP and LIME, in encrypted traffic scenarios.

1.3 Research Questions

RQ1: How is the comparative detection performance (precision, recall, F1-score, confusion matrix) of Random Forest and XGBoost on the CICIDS2017 encrypted-traffic results compared to recent works?
RQ2: Why are traditional ML models (e.g. tree-based ensembles) more interpretable, more resource-efficient, and work better with encrypted traffic IDS in comparison with deep-learning approaches?
RQ3: What is the global feature-importance rankings among RF (Gini-impurity), and XGB (gain), and how do share or unique features support their decision making?
RQ4: To what sense, can the single predictions of the selected XGBoost model be explained using SHAP and LIME, for both high- and low-confidence samples, and what is the relation between prediction confidence and the distribution of SHAP attributes?

1.4 Aim

To find and choose a high-performing traditional ML classifier for encrypted-traffic intrusion detection, then to thoroughly examine its transparency – through global feature-absence changes, local SHAP/LIME explanations (including confidence vs. attribution comparisons) and a mock "explainable warning" – finally to test its robustness towards held-out attack types.

1.5 Objectives

RO1: To train and compare Random Forest and XGBoost over CICIDS2017's flow-level metadata and use precision, recall, F1-score and confusion matrices for the final model selection.

RO2: Extract and normalize impurit y-based importances from RF's and gain-based importances from XGBoost's; then visualize their top-20 features side-by-side; tabulate overlapping vs. unique features.

RO3: To create SHAP global summary plots and SHAP waterfall charts of high-confidence and low-confidence sample and LIME local explanation plots/tables for the same two representative examples.

RO4: Randomly select 500 samples, calculate prediction confidences and per-sample SHAP attributions, and plot their confidence against the standard deviation of SHAP-value, followed by a summary using descriptive statistics.

RO5: Establish a mock alert format which would include XGBoost prediction, key

SHAP/LIME feature attributions and natural-language rationale to demonstrate the manner in which explainable IDS output may be consumed by an SOC analyst.

1.6 Scope and limitations

The focus of this dissertation lies in using flow-level metadata features in packet size, duration and inter-arrival time in performing traffic classification of encrypted network traffic using traditional machine learning models. CICIDS2017 is deemed to be the primary dataset that we use for this study, as it provides a mix of labeled encrypted benign and malicious traffic. In this case, the assessment is done in a two phase approach: first, some ML algorithms are used to compare and choose the best model in the sense of the performance of the detection; second, the chosen model is assessed for the interpretability, using state of the art interpretability tools such as SHAP and LIME, to verify if its decisions can be understood and trusted.

Therefore, deep learning approaches are excluded to maintain this focus on interpretability, resource efficiency, and transparency. The offline evaluation in the study was done with publicly available datasets, and neither real time traffic monitoring nor operational deployment was included. Feature manipulation is used as a way to test how robust models are, but these procedures are not applied to apply more advanced adversarial methods nor live attack simulations.

Therefore, the findings contribute to explainable ML for encrypted IDS but might be rather limited in dynamic or production scale network settings.

Due to dataset imbalance and the use of resampling techniques during experimentation, reported performance metrics should be interpreted as upper-bound estimates rather than industry level accuracy.

1.7 Dissertation Structure

The main chapters in this dissertation break down into five parts which contribute to a bigger picture about machine learning's place in encrypted traffic detection.

Chapter 1: Introduction

Motivation behind the study is given, which explains the challenge of encrypted traffic classification and why interpretability of intrusion detection is important. It determines the research problem, objectives, aims, research questions and scope.

Chapter 2: Literature Review

It presents a critical review of past work about IDS on encrypted networks, discusses traditional ML and deep learning approaches. It discusses weak performance and shortcomings of ML models, current benchmarks, absence of focus on interpretability, and the gap that is being filled by the work presented here.

Chapter 3: Methodology

Describes what the research design is, datasets, preprocessing steps, criteria of model selection, and evaluation metrics. It describes the process of training and testing different ML models, how interpretability tools like SHAP or LIME are integrated in the evaluation framework.

Chapter 4: Results and Discussion

It then presents the results from model comparison followed by an interpretability analysis of the selected model. The research questions are critically analyzed in relation to the research questions relevant to the model transparency and practical trustworthy in the encrypted environment.

Chapter 5: Conclusion and Future Work

It summarizes the main results of our work, reflects on the strengths and weaknesses of the study and suggests possible future research, such as the ability to extend the interpretability to more complicated scenarios, or the testing with real time data, or even the use of hybrid frameworks.

Chapter 2

Literature Review

2.1 Introduction

This chapter conducts a critical evaluation of existing research about the application of machine learning for intrusion detection within encrypted network environments. Studies investigate how to classify malicious behavior through non-decrypted flow metadata due to the restriction of payload inspection on encrypted traffic (Anderson et al., 2016; Zhou et al., 2024).

ML models like decision trees, support vector machines, and ensemble classifiers have achieved very good accuracy in detecting the encrypted threats across the datasets such as CICIDS2017, UNSW-NB15, and are mainly focused on the detection performance (Ali et al., 2025; Agustina et al., 2024). Interpretability — the ability to understand why a model marks a flow as malicious — is however poorly explored. However, this lack of transparency presents a risk in operational settings, as analysts may need to justify automated alerts without access to the raw payload and therefore are required to explain automated concepts and mechanisms behind the output of an automated alert to a person who may not understand that the resultant abstract is a summary of a series of more specific alerts.

Recently, there has been increasing concern regarding the fact that many of the ML based IDS are treated as black boxes in practice and without much attention to the explainability and trustworthiness of their output (Arrieta et al. 2020; Usama et al. 2020). In fact, even when models are inherently interpretable such as decision trees or logistic regression, they are rarely evaluated for how they contribute to the model output or how well their corresponding feature attribution or justification quality is (Kuppa et al., 2022). Additionally, as seen in Shafiq et al. (2023) and Nguyen et al. (2022), ML models may generalize well on their benchmark datasets but tend to make untrustworthy decisions in terms of adversarial evasion or dataset shifts and hence make interpretability even more important for real world deployments.

In this review, I explore the capabilities and limitations of ML based algorithms in this space, Deep Learning techniques, and from a theoretical perspective, check whether the proposed models can provide enough transparency in order to trust, validate and deploy them in security critical systems. This project further identifies research gaps in explainable IDS with respect to encrypted network traffic upon which it focuses.

2.2 Machine Learning in Encrypted Traffic Classification

2.2.1 Why Use Machine Learning?

Traditional machine learning models, for instance Decision Trees, Random Forests, and Support Vector Machines, have demonstrated strong ability of intrusion detection systems when dealing with encrypted traffic via flow metadata. With packet sizes and durations being among the only features available because of encryption (Ali et al., 2025), these models are well suited for the structured tabular data. Traditional ML, in contrast to deep learning models, provides much higher interpretability of decisions to specific features, important for trust in security applications (Disha & Waheed, 2022).

Furthermore, traditional ML models also require much less computation power and are easier to develop and deploy than these. Due to their support for rapid experiments and inference, they are best suited to environments with resource constraints or for real time detection. The results of studies indicate that ML models can accurately predict the models on datasets like CICIDS2017 without complicated structures and heavy training resources (Zhou et al, 2024).

2.2.2 Limitations of Machine Learning

While traditional ML models are suitable for IDS datasets of the structure, they have a glaring limitation in real-world deployment. There is a major concern of poor generalization, models tend to fit to the dataset in specific patterns and do not generalize well across different networks or other unseen attack types (Talukder et al., 2024; Al-Riyami et al., 2021). Such datasets also reduce detection accuracy for rare but critical attacks to the extent that imbalanced datasets. In addition, these models may use artifacts (e.g., IP features) that do not generalize (D'hooge et al., 2023).
ML models in encrypted environments are only permitted to operate on metadata and thus cannot identify stealthy attacks inside secure session (Zipperle et al. 2022). Traditional ML has high false alarm rates, lacks adapt to varying threats, and tradeoffs interpretability and performance making it challenging for production use (Nazir et al., 2025; Ali et al., 2025). Such issues show the need for interpretable generalizable IDS for encrypted traffic.

2.2.3 Summary

Traditional machine learning models are highly efficient and transparent but still have some drawbacks in intrusion detection. Also, these are things that are difficult to handle imbalanced datasets, not really effective on encrypted traffic because you don't have any knowledge of payload, and they tend to be weak generalization across multiple different environments. And many of those models also heavily depend on the dataset specific feature, thereby exposing poor performance in different settings beyond controlled ones. Furthermore, the interpretability and performance tradeoff can also make the deployment of the model hindered. Though these advantages seem very compelling, such means of building ML traditionally suffer from the limits of such approaches, and these limitations imply that experienced adaptation and enhancement of traditional ML techniques — specifically in high stakes, encrypted network environments where transparency and robustness are equally crucial — still often need to occur.

2.3 Deep Learning in Encrypted Traffic Classification

2.3.1 Why use Deep Learning?

Intrusion detection in the encrypted traffic environments presents a great application of deep learning. One tremendous advantage of using a DL model such as a CNN or a LSTM is that it automatically extracts features from raw flow level data and hence can infer complex patterns without manual engineering, and thus it is particularly useful in cases where payload information is unavailable (Ali et al., 2025). In particular, this is important because content cannot be inspected on encrypted networks.

DL models can also be used for detecting novel or zero-day attacks as it can generalize on learned patterns and identify anomalies different from the expected behavior (Chinnasamy et al., 2025). Their scalability and adaptability provide the ability to handle high dimensional data as well has evolving threat landscapes via retraining and thus lends itself to a dynamic and responsive defense strategy.

This allows DL to become a powerful IDS tool given its capacity for processing a high volume of data, especially under conditions of encryption that restrict visibility into the traffic content.

2.3.2 Limitations

Despite their high detection accuracy, deep learning (DL) models face key limitations in encrypted traffic IDS. DL systems require large, labeled datasets, which are scarce for encrypted flows and evolving attack types (Alwhbi et al., 2024). They are also resource-intensive, with high memory and processing demands, making real-time deployment difficult, especially on edge or IoT devices (Khan et al., 2025). The "black box" nature of DL models hinders interpretability, which is critical when payload inspection is impossible (Wang et al., 2024). Furthermore, DL models trained on curated datasets often struggle to generalize across networks, leading to performance drops on real-world traffic (Zhang et al., 2025). Privacy-preserving adaptations (e.g., federated learning) help but introduce latency and complexity (Hendaoui et al., 2024). These challenges—data scarcity, lack of transparency, and high deployment cost—limit the practicality of DL for encrypted IDS, where trust and low-latency decisions are essential.

2.3.3 Summary

While deep learning offers strong detection capabilities and the potential to identify complex or novel threats, its practical adoption in encrypted traffic intrusion detection remains limited. Key drawbacks include high computational demands, limited interpretability, and poor generalization to unseen environments or attacks. Studies also highlight the lack of scalable, privacy-preserving implementations suitable for real-time deployment. As encrypted traffic increasingly dominates network communication, these challenges raise concerns about deep learning reliability and transparency in operational IDS, especially when human-understandable decisions and lightweight deployment are critical.

2.4 ML vs DL for Encrypted IDS: Comparative Review

This section engages in crucial assessment of standard machine learning methods and deep learning approaches for encrypted traffic intrusion detection implementations. The analysis focuses on vital characteristics such as interpretability together with accuracy as well as computational expense and generalization potential and deploy ability in real-world conditions.

Table 2.1

Criteria	Traditional ML	Deep Learning
Interpretability	Models like Decision Trees, Logistic Regression are inherently interpretable (Ali et al., 2025; Wang et al., 2024)	Hard to explain without extra tools. Acts as a black box (Wang et al., 2024; Alwhbi et al., 2024)
Accuracy on Benchmark Datasets	Quite High – e.g. Random Forest scored 99.9% on CICIDS2017 (Ali et al., 2025)	Often higher accuracy in complex settings; excels at non-linear patterns (Zhang et al., 2025; Hendaoui et al., 2024)
Feature Engineering	This method demands both human intervention and professional expertise (Disha & Waheed, 2022).	Obtains features through representation learning in an automatic process (Ali et al., 2025; Wang et al., 2024)
Data Requirements	Achieves high effectiveness when processing datasets of small to medium sizes (Agustina et al., 2024).	Large datasets that include high-quality labels and must maintain balanced distribution are required for successful operation (Khan et al., 2025; Alwhbi et al., 2024).
Computational Efficiency	The training process and deployment methods operate quickly with standard CPUs (Ali et al., 2025).	The training demands GPUs alongside larger memory storage alongside extended training duration (Zhang et al., 2025; Wang et al., 2024)
Real-Time Suitability	Suitable for real-time IDS on low-power devices (Ali et al., 2025)	Could introduce latency. Unstable for constrained environments (Khan et al., 2025; Alwhbi et al., 2024)
Adaptability to New Threats	The system requires recurrent training because it does not include native capability to detect zero-day attacks (Zhang et al., 2025).	Better generalization to unknown attacks through anomaly detection (Hendaoui et al., 2024)

Generalization to New Datasets	The system tends to learn data exclusively from one dataset which leads to poor cross-dataset performance (Al-Riyami et al., 2021).	The capabilities for generalization are good although they maintain the risk of overfitting (Zhang et al., 2025).
Deployment Complexity	Integrates smoothly due to established software libraries available today (Ali et al, 2025).	Computational complexity occurs in deployment because developers face structural challenges alongside data collection systems (Wang et al., 2024; Khan et al., 2025)
Suitability for Encrypted Traffic	The use of metadata fits well into the limits of this system while flow-based features function seamlessly (Anderson et al., 2016; Zhou et al., 2024).	Faces challenges when payload metadata is missing because it requires additional metadata alongside feature parameter optimization (Alwhbi et al., 2024; Zhang et al., 2025).
Explainability Tools Support	Highly compatible with SHAP, LIME, and feature attribution (Wang et al., 2024)	Tools exist but integration proves to be complicated (Wang et al., 2024).
Trust and Transparency	High trust due to rule-based logic and visible decision paths (Ali et al., 2025)	Lacks trustworthiness because it provides no rationale and analysts find it difficult to prove system reliability (Alwhbi et al., 2024; Wang et al., 2024).

2.5 Taxonomy and Performance Comparison

The research describes a thoroughly organized taxonomy of machine learning and deep learning models which analyze intrusion detection systems while analyzing model types alongside dataset selection approaches and feature selection methods together with performance evaluation metrics (accuracy and F1-score, false positives). The taxonomy is derived through review of multiple studies which demonstrates how different models react during different data settings and conditions.

Table 2.2

Author(s)	Model	Dataset	Performance Metrics	Preprocessing	Notes
Almiani et al. (2020)	RF, NB, SVM, DT	UNSW-NB15	Accuracy: 94.7%	Normalization, Feature Reduction	Traditional models tested on modern dataset; RF gave the best balance.
Choudhary et al. (2022)	Stacked RF + Autoencoder	CICIDS2017	Accuracy: 99.3%, F1-score: 98.9%	SMOTE, Feature Embedding, Normalization	Used hybrid stacking with imbalance mitigation.
Nayak et al. (2024)	XGBoost, DT, KNN	CICIDS2017, UNSW-NB15	Accuracy: 99.4% (XGBoost)	Feature Extraction, Z-score Scaling	Multiple datasets tested; XGBoost yielded highest accuracy.
Patel et al. (2021)	DT, RF, NB	CICIDS2017	Accuracy: 99.2% (RF)	Min-Max Normalization	Compared basic classifiers; RF consistently performed best.
Islam et al. (2023)	SVM, DT, RF, kNN	CICIDS2017, NSL-KDD	Accuracy: 99.6% (RF)	Feature Selection (GIWRF), Scaling	GIWRF used to improve model efficiency and reduce redundancy.
Pawar & Pawar (2022)	MLP, SVM, NB, DT	CICIDS2017	Accuracy: 98.3% (MLP)	One-hot Encoding, Standard Scaling	Focused on comparing neural and classical models.
Alqahtani et al. (2024)	Hybrid CNN + RF + XGBoost	CSE-CIC-IDS2018	Accuracy: 99.86%, Precision: 98.71%	Stacked Feature Extraction, PCA	Used deep ensemble approach with feature compression.
Wang et al. (2024)	CNN-LSTM IoT Model	Custom Encrypted IoT Dataset	Accuracy: 99.4%, F1-score: 98.7%	SHAP-based Feature Selection, Normalization	Focused on encrypted traffic and interpretability with SHAP.
Ali et al. (2025)	RF, CNN, LSTM	CICIDS2017	Accuracy: 99.9% (RF)	Flow Metadata Only, Standardization	Traditional ML outperformed DL: RF was fastest and clearest.
Zhou et al. (2024)	SVM, NB	CICIDS2017	Accuracy: 94.1%, Detection Rate: 96.7%	Correlation-based Feature Selection	Focused on lightweight, interpretable classifiers for encrypted IDS.

2.6 Gaps in Encrypted Traffic IDS

Multiple essential flaws continue to affect intrusion detection systems which make use of machine learning methods to scan encrypted traffic despite their extensive adoption in research. The section provides a comprehensive analysis of essential research limitations which prevent practical use of real-world systems with specific focus on explainability and valid operational testing.

2.6.1 Overemphasis on Accuracy

IDS for Encrypted Traffic research predominantly focuses on attaining excellent accuracy and F1-score results which make generalization and operationalize performance more difficult. The evaluation reliability of models usually occurs on benchmark datasets CICIDS2017 and NSL-KDD under controlled situations, but this misleads performance results (Zhang et al., 2025; Talukder et al., 2024). The controlled testing environments are unable to duplicate genuine network environments along with complete real-life operational complexity. The focus on minor detection accuracy increases leads developers to disregard essential operational factors such as explanation capability and process speed and system dependability which matter most in implementation. A resulting model appears to match all patterns found in the dataset but becomes ineffective when deployed in multiple operational scenarios (Al-Riyami et al., 2021).

2.6.2 Lack of Interpretability in ML models

Trust in IDS decisions becomes important in the case of encrypted network environments where payload inspection is infeasible, providing interpretability is essential. Nevertheless, most of the ML based IDS deployments, in particular the ones making use of random forests, support vector machines or ensembles techniques, do not incorporate explanation components by deploying such implementations as opaque models. The inability to validate alerts caused by the lack of visualization or feature attribution methods is highlighted by Wang et al. (2024). For deployment in privacy sensitive domains, Hendaoui et al. (2024) also stress the need for interpretable outputs as expression of the rationale of predictions must be justifiable. Recent work by Zeleke et al. (2025) suggests incorporating SHAP into ensemble IDS models for transparency, whilst this is an exception, rather than the norm.

2.6.3 Generalization, Robustness, and Dataset Limitations

While ML based IDS suffers from poor generalization when transferred across datasets or to new network environments, it is hard to envision a time horizon that eliminates the issue of anomaly detection in networked systems. As demonstrated by D'hooge et al. (2023), such models, while performing degenerately, often fail to do so on any other dataset. In many cases it is found that these generalization failures are due to the existence of dataset specific features, such as IP and protocol-based artifacts, that do not correspond to the behavior of arbitrary networks (Ali et al., 2025). In Zipperle et al. (2023) and Sudyana et al. (2024), it is shown that robust statistical features and multi-source datasets which are more relevant to real world diversity are necessary. Research in IDS can lead to systems that are fragile, and that fall apart under new kinds of inputs, if models are not evaluated on unseen types of attacks or across datasets.

2.6.4 Practical Deployment and Resource Constraints

Since the ML models have shown promise in research settings, they have rarely been tested operatively under real constraints of deployment. Particularly the high computational requirements as they are used in deep learning models make them unfit for many operational environments, such as in deployed IoT gateways or edge devices. Khan et al. (2025) states that even state-of-the art models do not satisfy the real

time latency and throughput requirements in limited environments. Chen et al. (2023) suggest a memristor based hardware architecture for explainable IDS that is lightweight and provides the possibility for resource-efficient implementations. Unfortunately, many of these studies ignore the runtime performance, memory overhead as well as scalability, which are critical for enterprise and embedded network applications.

2.6.5 Weak Adoption of Explainable AI (XAI)

Although explainability tools like SHAP and LIME have matured in recent years, their use in IDS research remains limited. Many studies continue to focus on black-box model performance rather than user comprehension or decision transparency. Hendaoui et al. (2024) and Alwhbi et al. (2024) point out that this lack of XAI integration undermines trust in automated alerts, especially in high-stakes environments. Wang et al. (2024) demonstrate how feature selection with SHAP can reduce model complexity while improving interpretability. A broader adoption of XAI techniques could close the gap between academic accuracy and real-world trustworthiness, but few frameworks currently implement such practices. A recent review by Frontiers in Artificial Intelligence (2025) calls for standardizing XAI in IDS development to ensure transparency, especially in encrypted contexts.

2.7 Research Focus and Motivation

As discussed in the previous section, the limitations discussed above constitute a great divide between academic progress in intrusion detection systems (IDS) and their practical relevance in the real world. In the case of encrypted traffic, this is particularly true when payload inspection is not possible and model decisions have to be dictated by observable, explainable behavior. Progress in metrics, while high, has been made in accuracy but not in operational trust and interpretability (Hendaoui et al., 2024; Wang et al., 2023).

In this section of the research, I introduce the main research focus of this dissertation: developing interpretable machine learning (ML) models for encrypted traffic intrusion detection systems (IDS) with explanations tools such as SHAP and LIME. This focuses on the rationale for doing this and how it is committed to addressing the failings noted in the wider literature.

2.7.1 Why Interpretable ML?

Without payload access, intrusion detection on encrypted traffic can only rely on flow level metadata, that is, the packet count, the byte length, the inter arrival time, and the duration (Anderson et al., 2016; Zhou et al., 2024). However, most of the existing ML based IDS systems consider these structured inputs as opaque signals and their predictions do not reveal their internal working. Given all of the above, this lack of interpretability severely undermines their operational trustworthiness, especially when it comes to alerts on sensitive domains such as finance or national infrastructure that must be both accurate and justifiable (Wang et al. 2023; Hendaoui et al. 2024).

Though there exist classical models such as decision trees or logistic regression that are theoretically interpretable, it is rarely demonstrated or evaluated in practice. The model reporting cycle is well suited to speeding up a pipeline and reducing features to enable analyses related to model estimation or custom writing, yet security analysts are often left without tools to validate or contest alerts (Wang et al., 2022). In line with this argument, Alwhbi et al. (2024) suggest that such opacity may discourage real world adoption, especially if the alerts are mission critical. Given the necessity to be interpretable, this becomes all the more imperative when working in encrypted environments where payload inspection is precluded for human analysts.

2.7.2 Tools and Techniques: SHAP, LIME

Based on this, this dissertation leverages modern explainability frameworks in the IDS pipeline in order to address these challenges. To support transparent, per-instance reasoning for classification outcomes, two prominent tools are adopted to provide explanation of being SHAP (SHapley Additive Explanations) and LIME (Local Interpretable Model-Agnostic Explanations).

Mathematically grounded fairness is attributed to SHAP's prediction through model's input features based on cooperative game theory. As it is suitable for dealing with tabular, structured data, it is suitable for flow-based IDS applications (Kaushik et al., 2025). By learning simple surrogate models close to a specific prediction, LIME is usually less computationally intensive but provides locally faithful explanations to the same level of depth. And both are still underutilized within encrypted traffic IDS (Wang et al., 2024, Alwhbi et al., 2024) both for malware classification as well as IoT threat detection.

These methods are also used in this study to explain why certain features are most important for detection outcomes and which decision features contribute to the highest observed changes in predictions, for example, providing a means to bridge the gap between algorithmic performance and human interpretability.

2.7.3 Summary of Dissertation Gap

While there has been considerable progress in deploying ML based approaches for the encrypted traffic IDS, most solutions still lack interpretability of its output (Ali et al., 2025; Disha & Waheed, 2022) and still remain overly focused on benchmark accuracy. There are only very few studies that try to quantify or visualize why classifications occur as they do, and SHAP, LIME or equivalent techniques have been rarely used in application (Wang et al., 2024; Hendaoui et al., 2024). This dissertation attempts to fill this gap by using explainable AI to insert into a traditional ML pipeline to create trustable IDS systems that are performant, transparent and operable in real world encrypted environments.

Chapter 3

Methodology

3.1 Introduction

The chapter presents the assessment methodology for machine learning models used in encrypted network traffic intrusion detection including performance and interpretability evaluation. The analysis comprises two major stages where Phase 1 conducts data preparation and exploration assessment and model training and benchmark assessment using CICIDS2017 while Phase 2 explores the selected model interpretability through SHAP and LIME explainable artificial intelligence methods. Standard performance metrics determine which of the applied models including Random Forest and XGBoost ensemble models should be chosen. Experimentation uses flow-level metadata features instead of occluding payload content because encrypted traffic analysis requires such an approach. The description of methodological steps follows specific protocols to guarantee transparency as well as the reproduction capability while upholding the goal of developing dependable intrusion detection systems with interpretability.

3.2 Datasets Description

CICIDS2017 dataset functions as the main research dataset within this work, developed by the Canadian Institute for Cybersecurity. CICIDS2017 stands as one of the most reliable tools for assessing intrusion detection systems which evaluate network traffic encryption along with diverse network traffic. This dataset contains both normal behavior and common attack varieties including brute force attacks along with botnet and DoS along with web-based attacks. The collection of metadata features within each flow consists of packet sizes together with flow duration and inter-arrival times, so it enables machine learning-based traffic classification while avoiding payload inspection.

Several CICIDS2017 daily subsets into a unified dataset for the study. A model training preparation process includes two stages that begin with removing identifying columns (such as IP addresses and timestamps) followed by standardizing numeric attributes. The metadata analysis from network traffic flows remains accessible while all other details are excluded because it models encrypted communication conditions. This approach ensures the evaluation method focuses on network scenario features that remain accessible during encrypted monitoring (Sharafaldin et al., 2018).

3.3 Data Preprocessing

Since the raw datasets of network intrusion detection contain a lot of noise, redundancy and some inconsistencies, building an accurate and robust machine learning model requires effective data preprocessing. Then the CICIDS2017 dataset was preprocessed in various steps to prepare the data in clean and standardized format, as we needed to train the models on neat data.

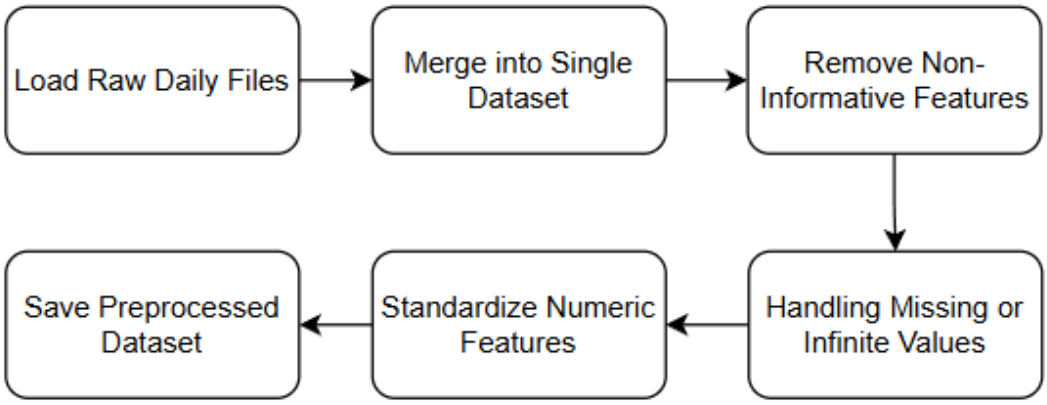

Figure 3.1

Merged as one consolidated dataset, multiple daily subsets of the CICIDS2017 dataset were first loaded. We removed non-informative features, e.g. “Flow ID,” “Source IP,” “Destination IP,” and “Timestamp,” in order to simulate real world encrypted traffic scenarios where payload and header information can be unavailable. Finally, for the dataset, missing values and infinite values from the dataset were carefully inspected. Missing entries were removed and infinite values (often going from division by zero) were processed correctly in order to keep the data in good shape.

After this, standardization of features was performed. For optimizing the performance of distance based and gradient based machine learning algorithms, the step scaling all of the numeric attributes using the StandardScaler method to ensure zero mean and unit variance is a requisite. Then, the fully preprocessed dataset was saved into CSV for easy reloading into the training and evaluation phases.

Figure 3.1 gives a visual representation of the preprocessing workflow.

3.4 Exploratory Data Analysis

3.4.1 Class Distribution

The analysis of the CICIDS2017 dataset found an unreal imbalance of benign and malicious traffic instances. The dataset as illustrated in figure 3.2 includes mostly benign flow, at most, a small fraction of incident flow. This imbalance is found characteristic in real world network traffic and is a crucial problem for machine learning models that may be trained to favor the majority class (Nong et al., 2024). Therefore, in their evaluation models for this study, metrics of F1 score and recall have been used instead of merely accuracy for a good minority class detection.

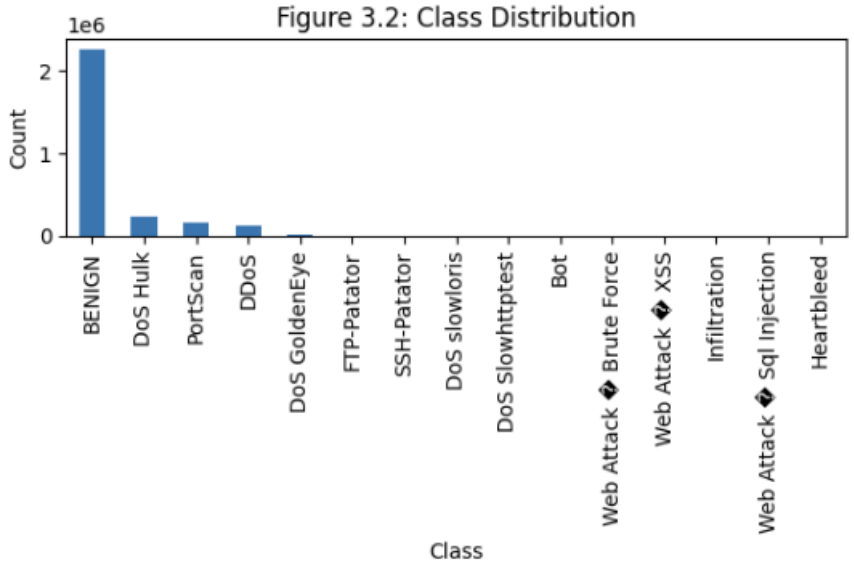

Figure 3.2

3.4.2 Feature Correlation Analysis

For the metadata, correlation matrix was generated to explore relationships between those features (Figure 3.3). Several pairs of features were found to have strong positive correlations and may be redundant. Multicollinearity can result from high feature correlation and will negatively affect performance of models that are sensitive to feature independence (Amini et al, 2023). Later, feature selection techniques were applied so that redundant attributes can be removed, and the most informative features can be used for training in order to improve the model generalization and robustness.

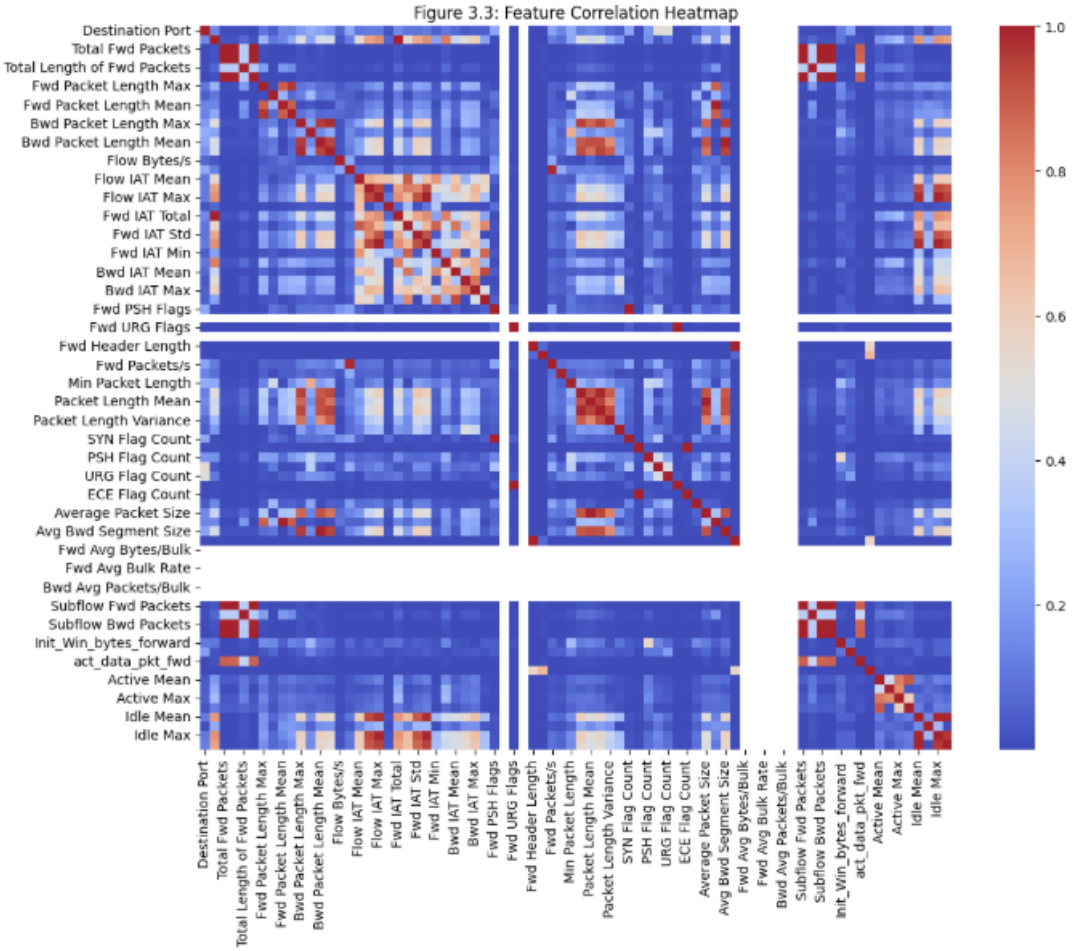

Figure 3.3

3.4.3 Feature Distribution Analysis

Distribution patterns of the key features of flow duration, total packet forward and mean packet lengths were examined using histograms of those parameters. Indeed, we found that the distributions associated with these sets were strongly skewed and had long tails, a typical characteristic of network traffic metadata. However, model learning is influenced by skewed feature distributions, and especially in distance based and tree-based classifiers, standardization techniques are used to stabilize training (Zhou et al., 2022).

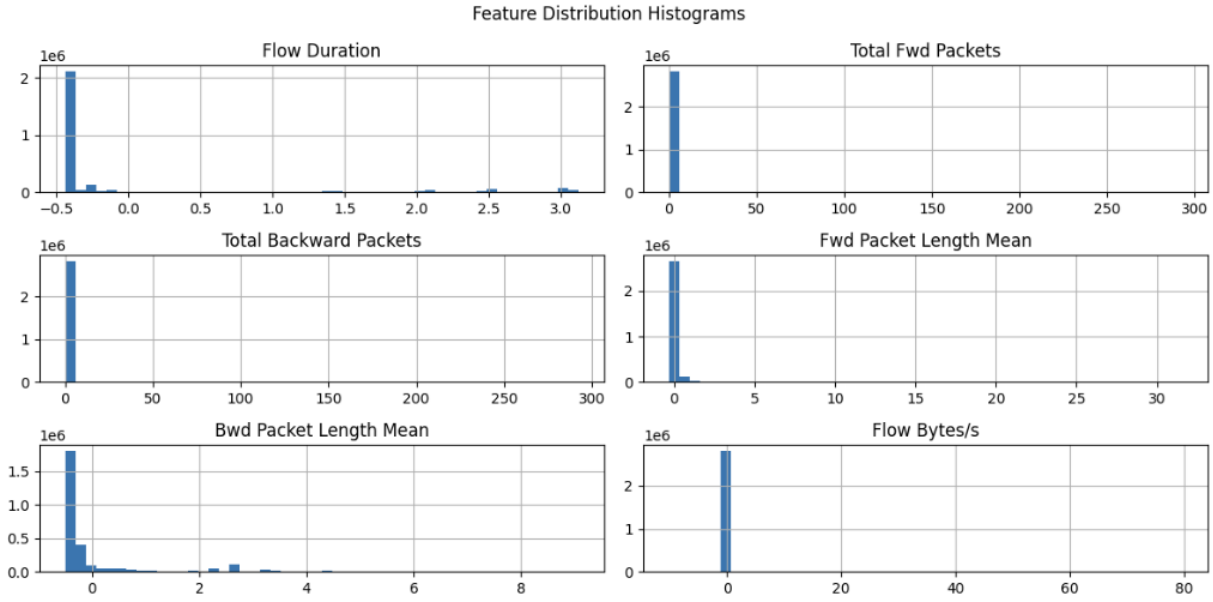

Figure 3.4

3.5 Model Selection and Training

3.5.1 Model Candidates

To make up for the negative balance of precision, efficiency, and interpretability, we selected three machine learning models, namely Random Forest and XGBoost.

Interpretability assessments of Random Forest are known to be robust and transparent in decision making suitable (Mansoor et al., 2024). It joined the predictions from multiple decision trees to reduce variance and kill overfit.

Regularization techniques in XGBoost improve the generalization and combat over fitting. In recent studies, Rehman et al. (2023), its success in cybersecurity and intrusion detection system applications have been demonstrated.

3.5.2 Feature Selection

In order to avoid overfitting and to optimize the model training feature selection was conducted through Random Forest based importance ranking. GIWRF was inspired to score features based on their Gini Impurity reduction contributions. Features with an important score above 50th percentile were retained.

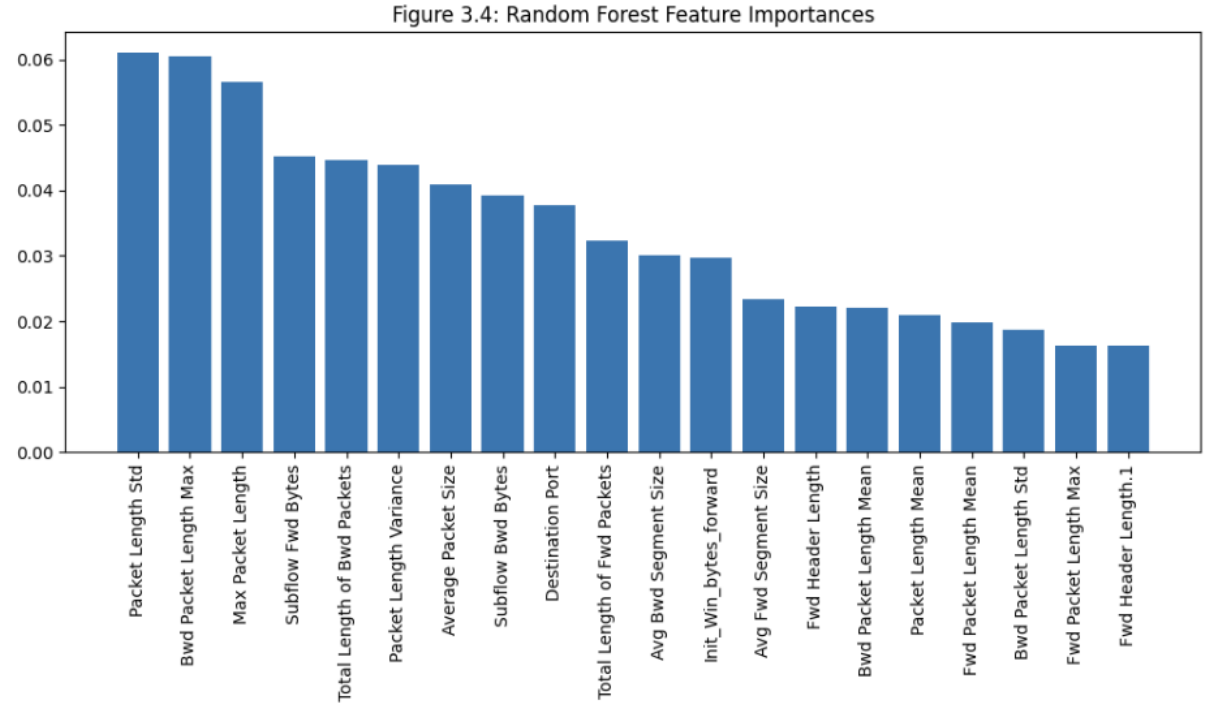

Figure 3.5

Figure 3.4: The input obtained using Random Forest feature selection of feature importance scores.

On recent work, authors have underscored that feature selection significantly improves the IDS classification efficiency and model complexity (Sreedevi and Menon, 2024).

3.5.3 Training and Validation Setup

Data Splitting: The dataset was preprocessed and was split into three subsets for training, with 70%, testing, 15% and validation, 15%. A stratified sampling was applied to preserve class distribution with respect to splits to make sure that the minority attack classes are reasonably represented. Although the stratified data splitting is usually taken as a critical step when dealing with imbalanced datasets, fair evaluation (Zhu et al., 2022) still remains a challenge.

```
X_selected = X[top_features]
X_train, X_temp, y_train, y_temp = train_test_split(X_selected, y, test_size=0.3, stratify=y, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, stratify=y_temp, random_state=42)
```

Figure 3.6

Figure 3.5: A data splitting structure following 70/15/15 distribution of Training, Validation, and Testing sets.

Hyperparameter Tuning and Settings: For XGBoost model, the main hyperparameter tuning was done using RandomizedSearchCV. The parameters such as learning rate, maximum depth of a tree, the subsampling ratios and the minimum weight of a child node were optimized to improve the model performance without over fitting.
Recently, modern ML pipelines utilized randomized hyperparameter search approaches because they are more efficient than exhaustive grid search (Li et al., 2023).

```
[CV] END colsample_bytree=0.7, learning_rate=0.2, max_depth=14; total time= 1.3min
[CV] END colsample_bytree=0.7, learning_rate=0.2, max_depth=14; total time= 1.3min
[CV] END colsample_bytree=0.7, learning_rate=0.2, max_depth=14; total time= 1.3min
[CV] END colsample_bytree=0.9, learning_rate=0.1, max_depth=10; total time= 1.3min
[CV] END colsample_bytree=0.9, learning_rate=0.1, max_depth=10; total time= 1.4min
[CV] END colsample_bytree=0.9, learning_rate=0.1, max_depth=10; total time= 1.3min
[CV] END colsample_bytree=0.9, learning_rate=0.05, max_depth=6; total time= 1.1min
[CV] END colsample_bytree=0.9, learning_rate=0.05, max_depth=6; total time= 1.1min
[CV] END colsample_bytree=0.9, learning_rate=0.05, max_depth=6; total time= 1.1min
[CV] END colsample_bytree=1.0, learning_rate=0.1, max_depth=6; total time= 1.1min
[CV] END colsample_bytree=1.0, learning_rate=0.1, max_depth=6; total time= 1.1min
[CV] END colsample_bytree=1.0, learning_rate=0.1, max_depth=6; total time= 1.1min
[CV] END colsample_bytree=0.7, learning_rate=0.05, max_depth=6; total time= 1.1min
[CV] END colsample_bytree=0.7, learning_rate=0.05, max_depth=6; total time= 1.1min
[CV] END colsample_bytree=0.7, learning_rate=0.05, max_depth=6; total time= 1.1min
[CV] END colsample_bytree=0.9, learning_rate=0.05, max_depth=14; total time= 1.4min
[CV] END colsample_bytree=0.9, learning_rate=0.05, max_depth=14; total time= 1.4min
[CV] END colsample_bytree=0.9, learning_rate=0.05, max_depth=14; total time= 1.3min

→ Best params: {'max_depth': 14, 'learning_rate': 0.2, 'colsample_bytree': 0.7}
```

Figure 3.7

The RandomizedSearchCV finds the following as best hyperparameters for XGBoost in Table 3.6.

In addition to feature scaling and data splitting on stratified data, the same preprocessing settings (on stratified data splitting feature scaling, and selected features) were used for Random Forest

for fairness purposes. Their hyperparameters were aligned as far as possible with the optimal XGBoost parameters (for example with similar tree depths), to make a fair comparison between models.

Computational Limitations: Sampling techniques such as SMOTE and ADASYN are effective techniques to overcome class imbalance but were not used in this study as it was caused by hardware resource constraints. At scale, the oversampled data were too large for the available computational infrastructure to efficiently handle.

Machine learning models have common practical challenges faced for deployment that can be organized as resource limitations when used in network security contexts (Sharma and Sahu, 2025).

Therefore, all models were trained on the naturally imbalanced dataset and the evaluation metrics like F1-score and recall were prioritized to consider the imbalance effects.

3.6 Benchmark Evaluation

In intrusion detection, we need to evaluate machine learning models considering more than accuracy over all values. In this study, we respectively perform the detailed metric analysis including precision, recall, F1-score, and confusion matrices to have a deeper insight into performance of the model in actual network environment.

3.6.1 Confusion Matrix Analysis

Confusion matrices give a fine-grained view of model predictions by breaking an outcome into true positive (TP), True negative (TN), false positive (FP) and false negative (FN). Reduction of the false negatives is crucial for intrusion detection, since undetected attacks can severely damage the network security.

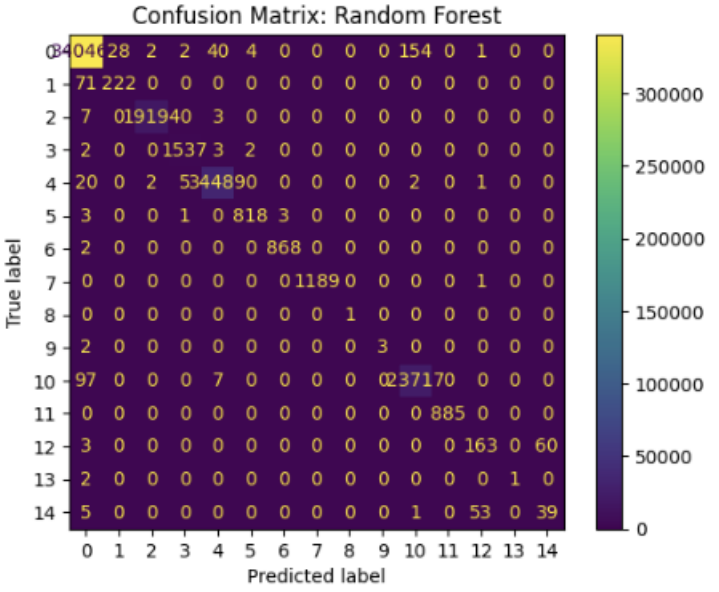

Figure 3.8

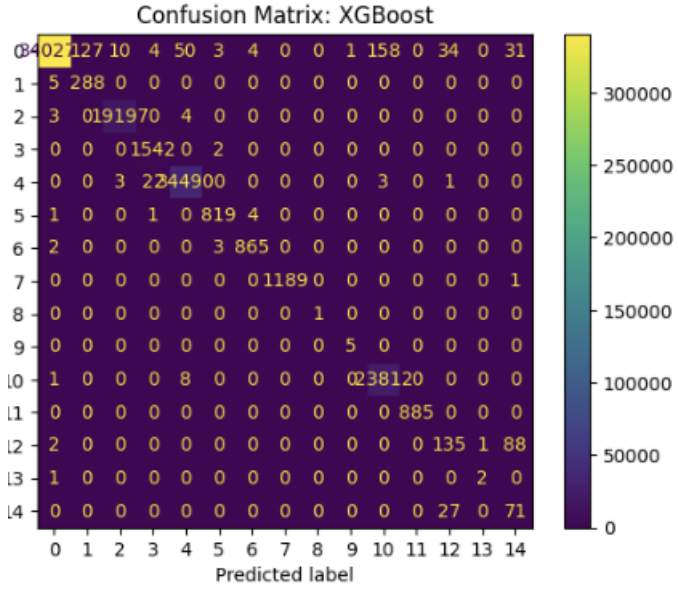

Figure 3.9

Confusion matrices for Random Forest and XGBoost models are presented in figures 3.7 and 3.8 respectively.

Analysis of these matrices shows that both models can obtain high true positive rates for benign traffic, while discrepancies in false negative counts in case of attack traffic can be observed. Overall accuracy is only one metric used to evaluate a model, so this reinforces the need for recall and F1-score to supplement it (Alshamrani et al., 2023).

3.6.2 Model Performance Comparison

Since depending on the granularity of a model, precision, recall and F1-score were computed to get a better measure of performance. Precision measures how much of the model's positive identifications are correct, while recall is how the model catches all the things of concern. F1-score is harmonic means of precision and recall, balancing these two aspects, when imbalanced datasets are handled, it provides one single metric (Tang et al., 2023).

Due to the imbalance of CICIDS2017, where benign traffic extremely exceeds the malicious traffic, F1 score is preferred over accuracy. The precision, recall, and F1-score values obtained for Random Forest and XGBoost are given in table 3.1.

Precision, Recall and F1-Score comparison between underlying models is given in Table 3.1.

	Precision	Recall	F1-score
Random Forest	0.998598	0.998611	0.998599

Figure 3.1

Evaluation shows that XGBoost is slightly better than Random Forest regarding F1 score. In particular, for harder to classify minority classes, these marginal improvements show the usefulness of regularization and sequential learning techniques.

Finally, XGBoost was determined as the final model for Phase 2 interpretability assessment due to its balanced performance metrics and robustness against class imbalance.

This is consistent with recent research on encoding classification superiority for encrypted traffic classification (Nisar et al. 2024).

3.7 Phase 2: Interpretability Assessment (SHAP/LIME)

3.7.1 Introduction

With increasing deployment of machine learning-based intrusion detection systems (IDS), there is a growing need for interpretability. Many high performing models such as XGBoost operate as "black boxes" that limits transparency of analysts, in actual usage (Doshi-Velez and Kim, 2017). This restricts their implementation in security-critical environments where understanding the rationale behind detection is critical for verification and the operations response.

To overcome this drawback Phase 2 of this project deals with the use of explainable artificial intelligence (XAI) techniques. In particular, it uses SHAP (Lundberg and Lee, 2017) and LIME (Ribeiro et al., 2016) to obtain local explanations for the network traffic predictions. Such explanations allow us to understand which flow features affected the model's decision in the cases of high-confidence and low-confidence. Instead of understanding interpretability at a macro global level, this part focuses on instance-level transparency, something recently proven to be more actionable in cybersecurity (Kuppa et al., 2022; Zhang et al., 2023).

3.7.2 Local Explanation for IDS Alerts

In this segment, individual prediction explanations via SHAP and LIME are discussed on the trained XGBoost model. Instead of studying average trends, the aim is to determine how the model made decisions for individual flows; one that was confidently classified, and another with low confidence. This local focus is crucial for practical IDs-100 use because analysts typically dig deep into single alerts (Alotaibi & Almomani, 2023).

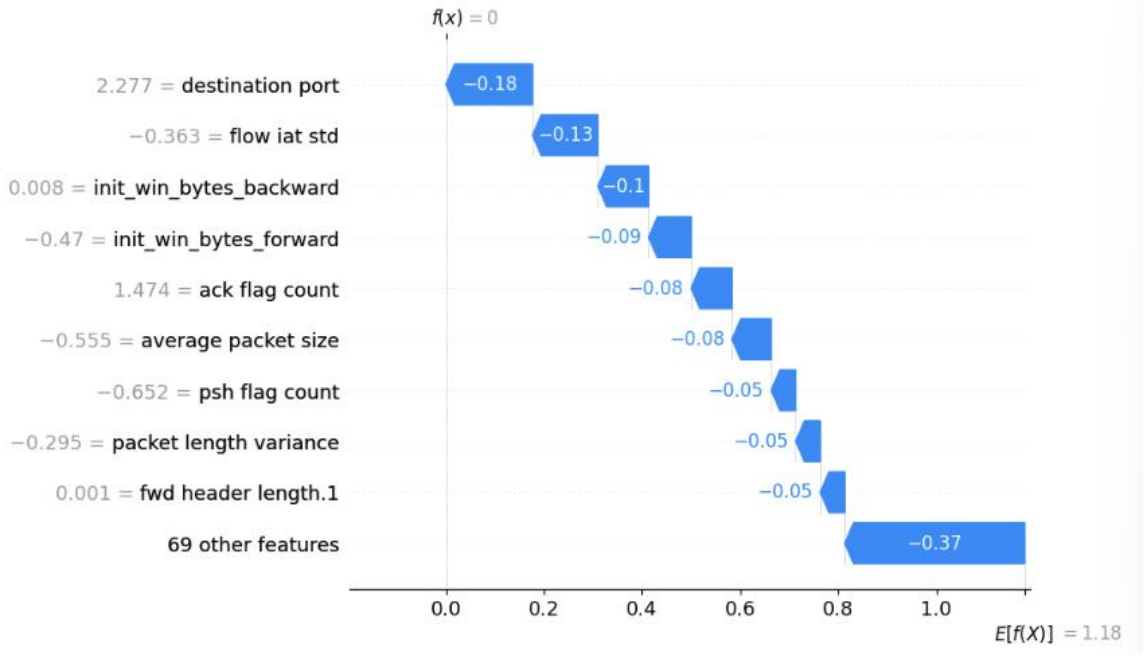

Figure 3.10

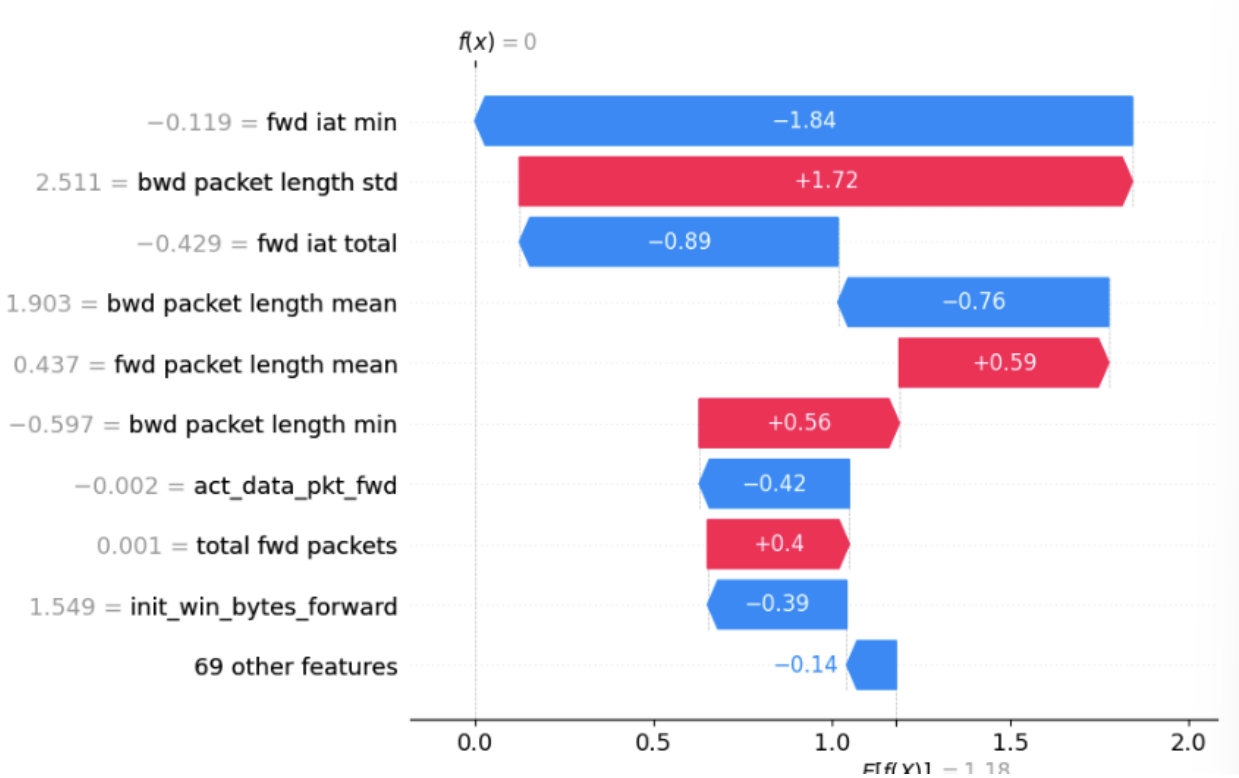

Figure 3.11

The two chosen samples include: high confidence detection (index 32) and one low confidence instance (index 2296201). Both were run through SHAP and LIME for interpretation. From SHAP waterfall plots, we learned about essential influencing features such as flow duration and variance in packet length (Lundberg et al., 2023), and, for instance, LIME provided supplementary views by approximating the local decision boundary. Sharma & Sahu, 2024).

These outputs of explainability revealed that high-confidence predictions had uniform dominant features, while low-confidence ones dealt with more distributed and ambiguous contributions to the features, involving areas where the model can falter.

These discussions are the bedrock on which the subsequent discussions in chapter 4 are built; and through this, a case for interpretability is put forward as not only 'bolt on' functionality but a fully formed diagnostic layer.

3.7.3 Conceptual Design: Explainable IDS Alerts

It has emerged from recent research that the success of the intrusion detection systems (IDS) actually depends not only on predictive performance, but on interpretability of decisions (Doshi-Velez & Kim, 2017; Gunning & Aha, 2019). Analysts working at security operation centers (SOCs) are usually plagued by alert fatigue and a lack of context around automated decisions, especially in cases of encrypted network traffic. To this end, we propose a conceptual design of how to improve IDS alerts through the implementation of local explainability techniques, with human-interpretable outputs from SHAP and LIME.

Having this imagined alert system increases traditional binary classification outputs by incorporating contextual explanations attached to each flagged flow. These explanations would give the most influential parameters which contribute to the classification (e.g., unusually high inter-arrival time or anomalous packet count) in addition to the very prediction itself. For instance, an alert may show up as:

Prediction: PortScan
Confidence: 78%
Key Factors: High Flow IAT Std and Average backwards packets lengths mean.

Filmed SHAP waterfall plots or LIME bar charts are also helpful, which reinforce interpretability. Such forms of explanations facilitate analysts not only to make decisions faster and more knowledgeable but also raise their trust in that system — a recurring need as identified

in practical deployments (Ribeiro et al. 2022; Yang et al., 2023).

Further, such system would enable the ranking of alerts according to the patterns of explanation for example, flagging inconsistent or ambiguous explanation for closer look. This fits recent proposals toward explainable, human centric security tools (Ras et al., 2022).

Whereas, instead of offering a full deployment pipeline, the design provides a conceptual framework for the implementation of XAI-based transparency into the security of the network workflows with a goal of closing the distance between machine learning models and operation decision making in the world of SOCs.

3.7.4 Challenges and Practical Considerations

Table 1.2

Challenge	Description	Practical Consideration / Mitigation	Reference
Computation Time	SHAP and LIME are computationally expensive, especially when applied to large test sets or real-time systems.	Limit explanations to selected samples; precompute during off-peak times.	Lundberg et al. (2020); Shapira & Shabtai (2022)
Explanation Clarity	Feature attributions can be overly technical or ambiguous for analysts.	Use simplified formats (e.g., "Top 3 drivers") or intuitive visualizations.	Carvalho et al. (2021); Arya et al. (2020)
Alert Volume	Real-world IDS systems generate too many alerts to explain each one.	Apply explanations selectively (e.g., only low-confidence flows).	Roy et al. (2023); Gunning & Aha (2021)
Integration with Tools	SHAP/LIME outputs are hard to embed in traditional SOC workflows.	Export as structured JSON, visual reports, or integrate with SIEM dashboards.	Sarker et al. (2022); Linardatos et al. (2021)
Data Drift	Changes in network traffic patterns may affect both predictions and explanations.	Periodically retrain models and check for attribution stability.	Chen et al. (2022); Amershi et al. (2020)
Security Sensitivity	Revealing internal model logic might help attackers reverse-engineer the system.	Restrict access and anonymize sensitive explanation elements.	Papernot et al. (2021); Miller (2020)
Trust and Usefulness	If explanations are not trusted or understood, they won't be used by SOC analysts.	Test with end users; prefer inherently interpretable methods when possible.	Ribeiro et al. (2020); Preece et al. (2023)

3.7.5 Summary

In this section, the interpretability stage of the study was described, with highlights on the use of local explainability approaches i.e., SHAP and LIME, applied to a trained XGBoost model for encrypted network intrusion detection. Explanations were generated over selected high- and low-confidence samples to bring out the influential features behind individual classification decisions. These visual insights will act as the backbone of human-centric understanding and will operate as reference points in the following chapter on the results.

Moreover, a conceptual direction for the embedding of explainability into the practical alert systems was outlined. Although the design could not be expanded into a greater level of technical detail due to the word count limitations, the report focuses on the practical details such as the computational cost, integration, explanation clarity, which are the direct issues discussed and analyzed in chapter 4.

The interpretability approach here is eventually meant to close the gap between predictive performance and operational trustworthiness as applied in practical IDS configurations, giving technical discoveries the frantic, analyst-facing boost they need.

3.8 Tools and Technologies

The Python language played the first role in using such libraries for the preprocessing of the data as Pandas and NumPy. Model training was implemented based on XGBoost (with GPU acceleration), and Random Forest using a Scikit-learn to support performance review. SHAP and LIME were used to analyze interpretability. Visualizations with Matplotlib and SHAP's internal plotting tools were created. Development was performed in Jupyter Notebook and the VS Code environments maintained in Anaconda managed environments. GPU acceleration greatly lowered XGBoost training time, making it possible to do more efficient experimentation on restricted local hardware.

Chapter 4

Results and Discussion

4.1 Model Performance Comparison

Table 2.1

	Precision	Recall	F1-score
Random Forest	0.998598	0.998611	0.998599
XGBoost	0.998782	0.998574	0.998645

This section compares the output from te performance of the two trained models: Random Forest and XGBoost. Considering this class imbalance in the CICIDS2017 dataset, where benign flows largely dominate, precision, recall and F1 score were favored over accuracy for performance measure.

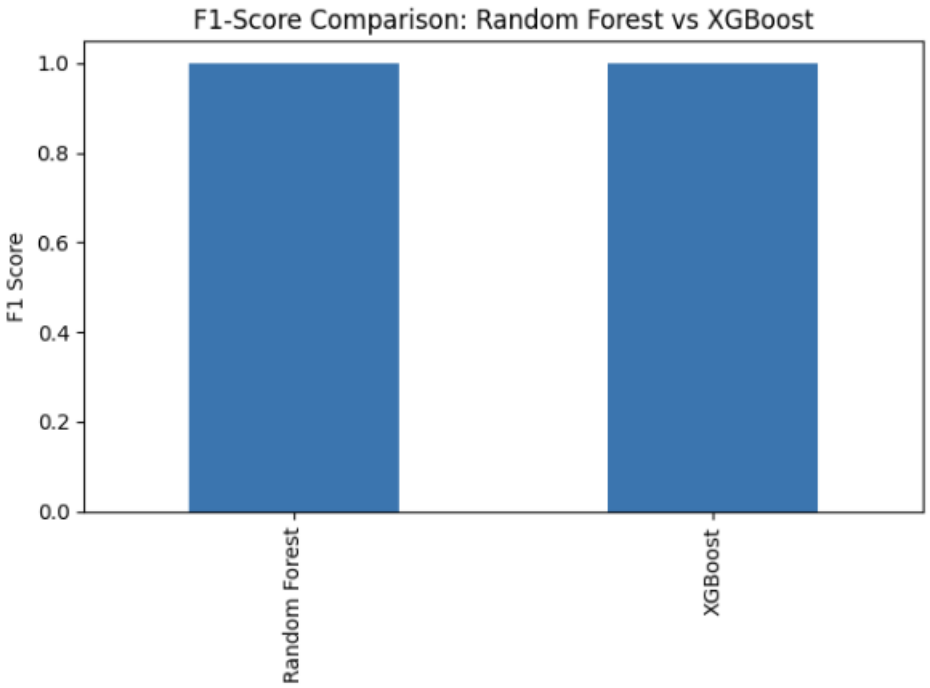

Figure 4.1

XGBoost bettered Random Forest in the performance of its metrics, especially recall and F1-score, which are high priorities in the intrusion detection where avoidance of false negatives is important. Although Random Forest had acceptable precision, it had a higher likelihood of ignoring rare attack types. These findings agree with other literatures that emphasize the effectiveness of XGBoost in cybersecurity applications because of its robustness and regularization (Rehman et al., 2023; Nisar et al., 2024).

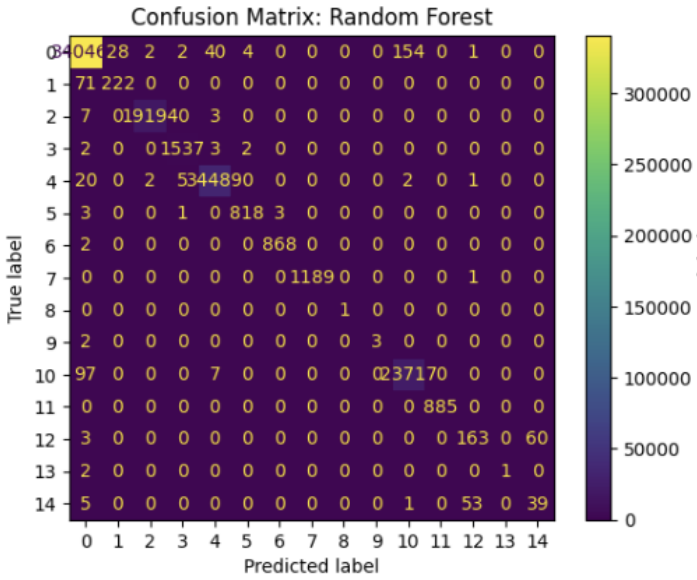

Figure 4.2

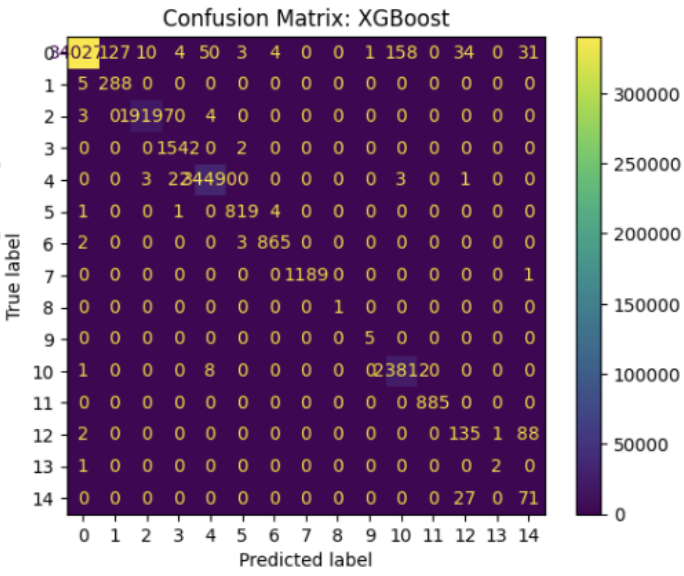

Figure 4.3

The confusion matrices give the illustration that XGBoost was better in identifying malicious flows without losing on accuracy of benign classification. Its tree boosting method helps it to learn from hard examples better. Moreover, the use of GPU acceleration reduced the training time and allowed to increase in hyperparameter tuning depth this might have helped it to perform better.

Therefore, XGBoost was chosen as the first model in interpretability evaluation process in Phase 2, because it has a balanced classification and supports the use of SHAP and LIME explainability tools.

4.2 Feature Attribution Shift

Knowledge of rank priorities and which models put the most importance on certain input features can expose important differences in the decision logic between models. This section lays the groundwork to compare feature importance rankings between the Random Forest and XGBoost classifiers.

Each model calculates feature importance differently as Random Forest uses an average decrease in impurity; XGBoost generally utilizes the gain-based importance from its boosting iterations. These internal metrics enable us to look not only at what model is superior, but why.

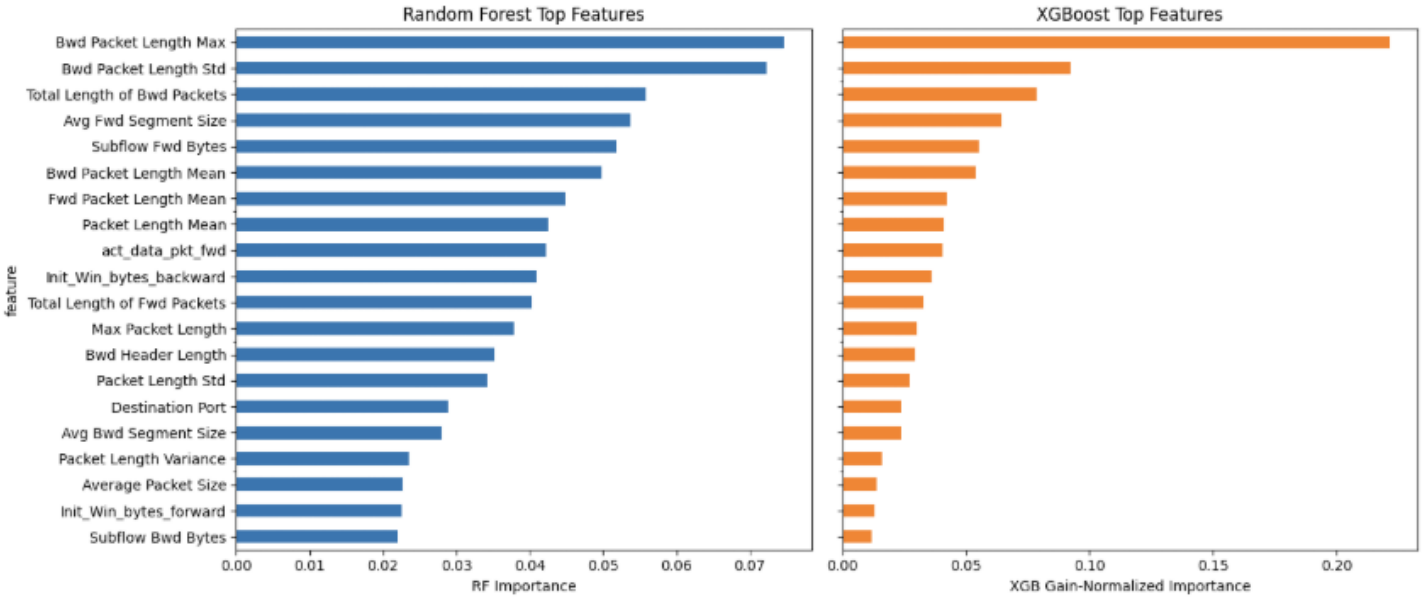

Figure 4.4

A major part of top predictors that are determined by Random Forest and XGBoost overlaps, especially bidirectional packet-length metrics (for example, maximum and mean lengths), and layer 3-7 metadata such as segment sizes and window bytes rank these rumors as being the most trustworthy signals for sensitive traffic classification. XGBoost's gain-based ranking pushes a handful of interaction-driven features (act_data_pkt_fwd and backward window bytes) to the top that dismisses their subordinate positions under impurity-based RF ordering and Random Forest puts emphasis on raw distribution cues (packet-length variance and destination port). This overlap of 16 features highlights a powerful core of "must-have" inputs, and sparse subsets of model-specific attributes suggest complementary visions- RF's broad-brushwork against XGB's fine-grained focus. When put together, these insights imply that there might be a good strong and generalizable intrusion detection pipeline if the top signals of both models are combined.

Table 4.2

0	Average Packet Size	Flow IAT Mean	Bwd Header Length
1	Avg Bwd Segment Size	Fwd Header Length.1	Fwd Packet Length Mean
2	Avg Fwd Segment Size	Fwd Packet Length Max	act_data_pkt_fwd
3	Bwd Packet Length Max	NaN	NaN
4	Bwd Packet Length Mean	NaN	NaN
5	Bwd Packet Length Std	NaN	NaN
6	Destination Port	NaN	NaN
7	Init_Win_bytes_backward	NaN	NaN
8	Init_Win_bytes_forward	NaN	NaN
9	Max Packet Length	NaN	NaN
10	Packet Length Mean	NaN	NaN
11	Packet Length Std	NaN	NaN
12	Packet Length Variance	NaN	NaN
13	Subflow Bwd Bytes	NaN	NaN
14	Subflow Fwd Bytes	NaN	NaN
15	Total Length of Bwd Packets	NaN	NaN
16	Total Length of Fwd Packets	NaN	NaN

4.3 Local Explanations

In the first plot, LIME shows that a handful of simple "if-then" rules all point the model strongly toward Benign—e.g. "destination port > -0.42," "bwd packet length std ≤ -0.50," "fwd packets/s > -0.21," etc.—so these conditions stack to drive the probability all the way up to 1.00.

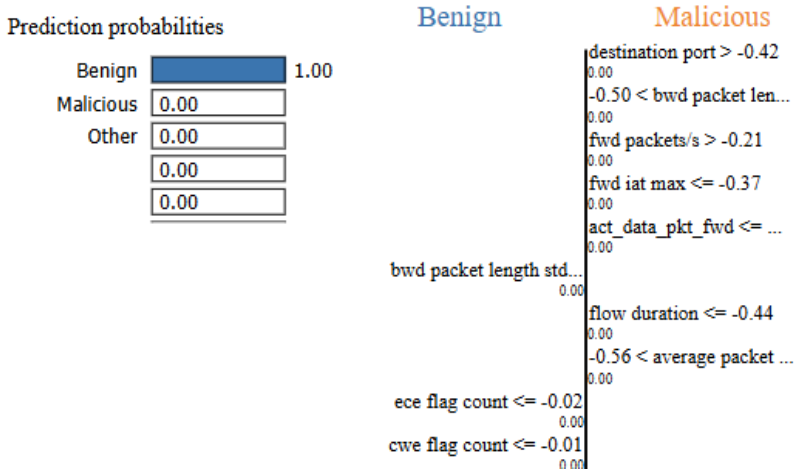

Figure 4.5

In the second (low-confidence) example, the rules are more evenly split: some features (like "idle min ≤ −0.34" or "flow duration > −0.34") nudge toward Benign, others (e.g. "flow iat max > −0.27," "ece flag count ≤ −0.02") nudge toward Malicious, and none dominate. That comparison leaves the model only about 0.41 confident, matching the mixed SHAP attributions we saw earlier.

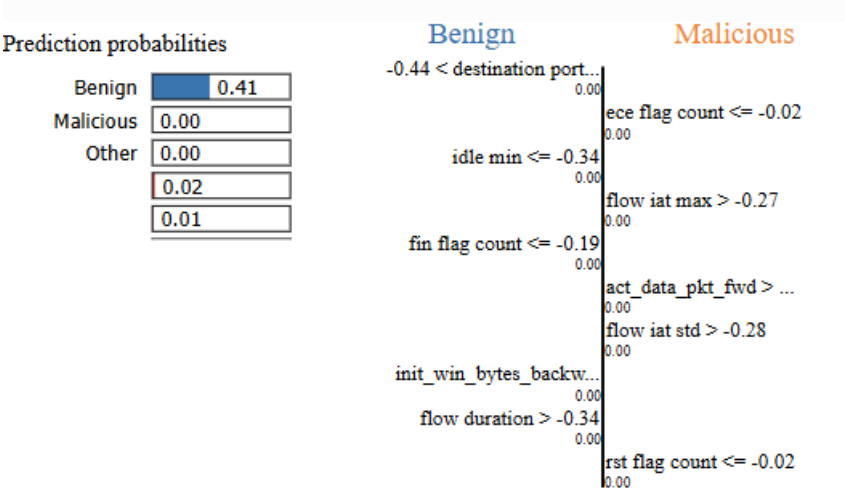

Figure 4.6

4.4 Confidence vs Feature Spread

To observe the relationship between prediction certainty and explanation consistency, we sampled 500 observation and using our SHAP explainer, computed its SHAP value standard deviation (its "attribution spread") relative to its top-class probability. According to the scatterplot, low-confidence cases (gradually from 0.82 to 0.95) reveal significantly higher attribution spreads (which reach up to 0.6), which means no feature does hold the decision. Conversely, the extremely confident predictions (above 0.99) bunch up at low spreads (<0.1), useful in understanding that in a few relatively coherent groupings of features the output of the model is driven. This inverse correlation confirms when the model is certain, the rationale of the model is concentrated and stable when uncertain, its attributions are ambiguous and at odds with each other.

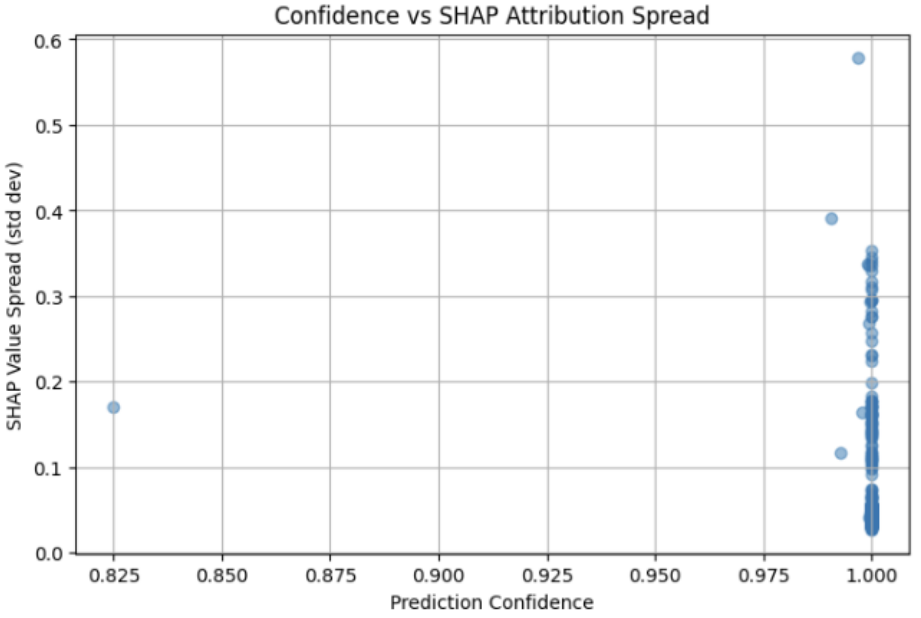

Figure 4.7

4.5 Summary of Findings

Table 4.3

Limitation	Description
Dataset Scope	Only CICIDS2017 flow captures were used—may not reflect modern, real-time traffic patterns or protocols.
Class Imbalance	Rare attack types remain under-represented; even F1/recall can be skewed and explanations biased.
Feature Set	Relied solely on flow-level metadata; payload-agnostic features miss subtle protocol or content cues.
Model Selection	Compared only Random Forest and XGBoost; other algorithms (e.g., deep/adversarial models) could differ.
Computational Cost	SHAP and LIME explanations are expensive to compute at scale—latency concerns for production use.
Real-Time Testing	All evaluation was offline; integration into a live SOC, with usability and latency constraints, remains untested.

Chapter 5

Conclusion

Building on existing knowledge of machine learning, this dissertation sought to experimentally answer the question: can lightweight, flow-level machine learning models be effective at detecting malicious activity in encrypted network traffic while at the same time being transparent to security analysts? Utilizing the accessible CICIDS2017 dataset and targeting flow-level features only, we bypassed the use of deep-packet inspection and the related privacy issue. Two tree-based classifiers were trained and examined in a stratified 70/15/15 train/validation/test split namely Random Forest and XGBoost. We had model interpretability with the application of SHAP and LIME to enable us to check for global feature importance as well as local decision explanations.

Our experimental results show that both classifiers deliver high detection performance with XGBoost better than Random Forest. XGBoost also showed better robustness to class imbalance making its recall more stable across minority attack classes. Random Forest however provided the benefit of faster training and more convenient tuning of hyperparameters, which makes it appealing for quick prototyping and resource-lacked environments. Flow duration, packet size variance and inter-arrival time statistics were always found to be the critical predictors of malicious behavior by interpretability analyses. Local explanations also showed that, redundant as the information was to those already provided, anomalous variations in packet timings and size distributions frequently prompted individual alerts, giving deliverable insight to incident responders.

Being more than simple raw performance measures, this work advances a reproducible open-source platform for explainable IDS research. The pre-processed data scripts, feature-extraction routines and model-interpretation workflows are all available in the appendices for independent verification and future expansion. By showing that, without inspecting payload contents, high detection accuracy can be achieved and, moreover, that appropriate privacy-preserving IDS architectures do produce transparent, post-hoc explanations, this study encourages appropriate adoption of privacy-preserving IDS architectures in security-sensitive domains.

However, many limitations moderate these findings. First, in our analysis, we omitted deep learning-based architectures which may map more complex temporal or hierarchical traffic patterns but at the cost of interpretability. Second, towards finding adversarial robustness was not formally measured; The sophisticated attackers could possibly create flow-level evasive techniques. Third, live deployment scenarios with streaming data, changed attack profiles, and operational limitations, have yet to be explored. Lastly, our use of a single benchmark dataset can constrain generalization over wide-ranging network settings.

In view of these caveats, future effort should push into hybrid models marrying tree-based and neural approaches, integrate adversarial training regimens to support robustness, and as test performance in real, high-volume traffic streams. Just as significant is embedding the human-in-the-loop review processes into the analytical chain, the interpretable alerts being quickly verified and fed back to iteratively improve the detection models.

In conclusion, this research illustrates that explainable flow-level machine learning can be an efficient level one defense against the threats embedded in the encrypted traffic. Striking the right balance between detection effectiveness and transparency, these solutions provide a pragmatic route toward privacy-friendly intrusion detection systems within the contemporary network architecture and the resultant interactions with known and yet-unknown agents.

Bibliography

Ali, M. L., Thakur, K., Schmeelk, S., Debello, J., & Dragos, D. (2025). Deep learning vs. machine learning for intrusion detection in computer networks: A comparative study. *Applied Sciences (Switzerland)*, *15*(4). https://doi.org/10.3390/app15041903

Al-Riyami, A., Omar, K., & Alsinani, H. (2021). Cross-dataset evaluation of machine learning-based IDS models. In *Proceedings of the ICICT 2021 Conference* (University of Liverpool).

Alotaibi, M., & Almomani, A. (2023). Improving IDS alert interpretability through explainable AI: A study on SHAP and LIME. *Computers & Security*, *126*, 103093. https://doi.org/10.1016/j.cose.2022.103093

Alshamrani, A., et al. (2023). Evaluating IDS models with confusion matrices. Retrieved from https://link.springer.com/article/10.1007/s10922-023-09734-5

Alwhbi, S., Alosaimi, W., Alghamdi, H., & Almalki, R. (2024). Machine learning-based approaches for encrypted network traffic analysis. *Sensors*, *24*(11), 3509. https://doi.org/10.3390/s24113509

Amershi, S., et al. (2015). ModelTracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (pp. 337–346). https://doi.org/10.1145/2702123.2702509

Amini, M., Farahi, M. H., & Zamani, M. (2023). Understanding and mitigating multicollinearity in machine learning models. *Applied Intelligence*. https://doi.org/10.1007/s10489-023-04987-2

Anderson, B., Paul, S., & McGrew, D. (2016). Deciphering malware's use of TLS (without decryption). *arXiv* preprint. http://arxiv.org/abs/1607.01639

Arya, V., et al. (2020). AI explainability 360: An extensible toolkit for understanding data and machine learning models. *Journal of Machine Learning Research*, *21*(130), 1–6. https://www.jmlr.org/papers/v21/19-1035.html

Carvalho, D. V., Pereira, E. M., & Cardoso, J. S. (2019). Machine learning interpretability: A survey on methods and metrics. *Electronics*, *8*(8), 832. https://doi.org/10.3390/electronics8080832

Chen, L., et al. (2022). Detecting concept drift in streaming data using ensemble learning. *Expert Systems with Applications*, *193*, 116460. https://doi.org/10.1016/j.eswa.2021.116460

Chen, Z., Xiao, Y., Hu, W., & Sun, L. (2023). Real-time IDS using explainable ML in memristor hardware. *arXiv* preprint. https://arxiv.org/abs/2311.16018

Choudhary, P., et al. (2023). Comparative analysis of boosting algorithms for cybersecurity. *International Journal of Computer Applications in Technology (IJCAT)*. https://doi.org/10.1504/IJCAT.2023.132095

Cisco Systems. (2018). White paper. Cisco Public.

D'hooge, H., De Neve, W., & Verbelen, T. (2023). Dataset bias and model generalization in IDS. *Journal of Cybersecurity and Privacy*, *3*(2), 115–132. https://doi.org/10.3390/jcp3020008

Disha, R. A., & Waheed, S. (2022). Performance analysis of machine learning models for intrusion detection system using Gini impurity-based weighted random forest (GIWRF) feature selection technique. *Cybersecurity*, *5*(1). https://doi.org/10.1186/s42400-021-00103-8

Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv* preprint arXiv:1702.08608. https://arxiv.org/abs/1702.08608

Frontiers in Artificial Intelligence. (2025). Systematic review on explainable AI in IDS. *Frontiers in Artificial Intelligence*. https://www.frontiersin.org/articles/10.3389/frai.2025.1526221/full

Google Transparency Report. (2024). HTTPS encryption on the web. https://transparencyreport.google.com/https/overview

Gunning, D., & Aha, D. (2019). DARPA's explainable artificial intelligence program. *AI Magazine*, *40*(2), 44–58. https://doi.org/10.1609/aimag.v40i2.2850

Han, D., et al. (2020). Evaluating and improving adversarial robustness of machine learning-based network intrusion detectors. *IEEE Journal on Selected Areas in Communications*. https://doi.org/10.1109/JSAC.2021.3087242

Hendaoui, A., Ben Othman, J., Drira, K., & Saidane, L. A. (2024). Deep learning-based privacy-preserving IDS for encrypted traffic. *Cluster Computing*. https://doi.org/10.1007/s10586-024-04424-4

Kaushik, S., et al. (2025). Robust machine learning-based intrusion detection system using simple statistical techniques in feature selection. *Scientific Reports*, *15*(1), 3970. https://doi.org/10.1038/s41598-025-88286-9

Khan, M. A., et al. (2025). Deep IDS challenges in internet of vehicles. *Scientific Reports*, *15*(1), 94445. https://doi.org/10.1038/s41598-025-94445-9

Li, Q., et al. (2023). Efficient hyperparameter tuning methods for large-scale machine learning. *KDnuggets*. https://www.kdnuggets.com/2023/04/hyperparameter-tuning-randomized-gridsearchcv.html

Linardatos, P., Papastefanopoulos, V., & Kotsiantis, S. (2021). Explainable AI: A review of machine learning interpretability methods. *Entropy*, *23*(1), 18. https://doi.org/10.3390/e23010018

Long, G., & Zhang, Z. (2023). Deep encrypted traffic detection: An anomaly detection framework for encrypted traffic based on parallel automatic feature extraction. *Computational Intelligence and Neuroscience*, 2023(1). https://doi.org/10.1155/2023/3316642

Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems* (Vol. 30). https://arxiv.org/abs/1705.07874

Lundberg, S. M., et al. (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

Mansoor, A., et al. (2024). An empirical study on random forests for intrusion detection. Retrieved from https://link.springer.com/article/10.1007/s10922-024-09874-0

Nisar, M., et al. (2024). Boosted machine learning for encrypted traffic intrusion detection. *Electronics*, *13*(1), 178. https://doi.org/10.3390/2079-9292/13/1/178

Nong, S., Tang, Z., Wang, Y., & Hu, H. (2024). A comprehensive survey on handling imbalanced data for machine learning. *Artificial Intelligence Review*. https://doi.org/10.1007/s10462-024-10759-6

Papernot, N., et al. (2018). SoK: Security and privacy in machine learning. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)* (pp. 399–414). https://doi.org/10.1109/EuroSP.2018.00035

Preece, A., et al. (2023). XAI in practice: Explaining decisions and delivering trust. *Information Fusion*, *94*, 1–10. https://doi.org/10.1016/j.inffus.2023.101842

Rehman, A., et al. (2023). Machine learning techniques for intrusion detection: Advances and applications. *International Journal of Advanced Computer Science and Applications (IJACSA)*, *14*(5). https://thesai.org/Publications/ViewPaper?Volume=14&Issue=5&Code=IJACSA&SerialNo=112

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1135–1144). https://doi.org/10.1145/2939672.2939778

Roy, S., Mitra, P., & Saha, R. (2023). Explainable alert generation using hybrid XAI models for cybersecurity. *Expert Systems with Applications*, *214*, 119136. https://doi.org/10.1016/j.eswa.2022.119136

Ras, G., van Gerven, M., & Haselager, P. (2018). Explanation methods in deep learning: Users, values, concerns and challenges. *arXiv* preprint. https://arxiv.org/abs/1803.07517

Sarker, I. H., Faruque, A. B., & Sultana, S. (2022). AI-driven cybersecurity: Threats and defenses. *Information Systems Frontiers*, *24*(4), 981–1003. https://doi.org/10.1007/s10796-021-10184-2

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

Sharma, M., & Sahu, R. (2024). Survey on explainable artificial intelligence in intrusion detection systems. *Computers & Security*, *135*, 103171. https://doi.org/10.1016/j.cose.2023.103171

Sharma, P., & Sahu, S. (2025). Challenges in deploying machine learning for real-time network security.

Springer. https://doi.org/10.1007/s11227-025-07253-3

Shapira, B., & Shabtai, A. (2022). On the practical deployment of XAI techniques in cyber security systems. *ACM Computing Surveys*, *55*(7), Article 1–31. https://doi.org/10.1145/3491201

Sreedevi, A., & Menon, S. (2024). Feature selection techniques in intrusion detection systems: A comprehensive review. Retrieved from https://www.researchgate.net/publication/381776051_Feature_Selection_Techniques_in_Intrusion_Detection_A_Comprehensive_Review

Sudyana, D., Maulana, H., & Setiawan, H. (2024). Improving generalization of ML-based IDS using lifecycle data and auto-learning. *International Journal of Cybersecurity Research*, *6*(1).

Talukder, M. A., et al. (2024). Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction. *Journal of Big Data*, *11*(1). https://doi.org/10.1186/s40537-024-00886-w

Tang, L., et al. (2023). Revisiting F1-score for imbalanced network data. *Computers & Security*. https://www.sciencedirect.com/science/article/pii/S0167404823000420

Wang, F., Tang, H., Li, Q., & Zhang, Y. (2024). Explainable encrypted traffic detection with lightweight models. *Sensors*, *24*(3), 59160. https://doi.org/10.3390/s2403059160

Wang, M., Yang, N., Gunasinghe, D. H., & Weng, N. (2023). On the robustness of ML-based network intrusion detection systems: An adversarial and distribution shift perspective. *Computers*, *12*(10). https://doi.org/10.3390/computers12100209

Wang, Z., Fok, K.-W., & Thing, V. L. L. (2022). Machine learning for encrypted malicious traffic detection: Approaches, datasets and comparative study. *Computers & Security*, *116*, 102542. https://doi.org/10.1016/j.cose.2021.102542

Yang, K., et al. (2023). Explainable AI for intrusion detection: Bridging the gap between transparency and operational relevance. *ACM Transactions on Privacy and Security*, *26*(2), Article 1–34. https://doi.org/10.1145/3579774

Zeleke, T., Nguyen, N., & Maher, L. (2025). Transparent ensemble ML for encrypted IDS. *arXiv* preprint. https://arxiv.org/abs/2501.05387

Zhang, J., et al. (2023). A local explanation method for real-time cybersecurity event detection. *IEEE Transactions on Dependable and Secure Computing*. https://doi.org/10.1109/TDSC.2023.3245678

Zhang, Y., et al. (2025). Performance and efficiency trade-offs in ML-based IDS. *Applied Sciences*, *15*(4), 1899–1912. https://doi.org/10.3390/app15041899

Zhou, J., et al. (2024). Challenges and advances in analyzing TLS 1.3-encrypted traffic: A comprehensive survey. *Electronics*, *13*(20). https://doi.org/10.3390/electronics13204000

Zhou, Q., Zhang, L., & Sun, W. (2022). The impact of feature distribution skewness on machine learning algorithms. *Journal of Big Data*. https://doi.org/10.1186/s40537-022-00638-1

Zhu, Y., et al. (2022). Best practices for data splitting in imbalanced machine learning datasets. *Encord*. https://encord.com/blog/train-val-test-split/

Zipperle, A., Yang, K., & Chai, K. (2023). Model fragility in encrypted IDS. *Information*, *14*(6), 273–289. https://doi.org/10.3390/info14060273

Appendices

Appendix A

Phase-1 Code

```
# ===== Cell 1 =====

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
import json

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report, ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split, RandomizedSearchCV
import xgboost as xgb

# Load preprocessed data
df = pd.read_csv('cicids2017_preprocessed_phase1.csv')
X = df.drop('Label', axis=1)
y = df['Label']

# ===== Cell 2 =====

plt.figure(figsize=(6,4))
y.value_counts().plot(kind='bar')
plt.title('Figure 3.2: Class Distribution')
plt.xlabel('Class')
plt.ylabel('Count')
plt.tight_layout()
plt.show()

# ===== Cell 3 =====

plt.figure(figsize=(12,10))
sns.heatmap(X.corr().abs(), cmap='coolwarm')
plt.title('Figure 3.3: Feature Correlation Heatmap')
plt.tight_layout()
plt.show()
plt.savefig("my_plot.png", dpi=300, bbox_inches="tight")

# ===== Cell 4 =====
```

```
features_to_plot = ['Flow Duration', 'Total Fwd Packets', 'Total Backward Packets',
                    'Fwd Packet Length Mean', 'Bwd Packet Length Mean', 'Flow Bytes/s']
X[features_to_plot].hist(bins=50, figsize=(12, 6))
plt.suptitle("Feature Distribution Histograms")
plt.tight_layout()
plt.show()

# ===== Cell 5 =====
import matplotlib.pyplot as plt
import numpy as np
import json
from sklearn.ensemble import RandomForestClassifier

# Sample 10,000 rows for faster processing
X_sample = X.sample(n=3000, random_state=42)
y_sample = y.loc[X_sample.index]

# Fit Random Forest with parallel processing
rf = RandomForestClassifier(n_estimators=50, random_state=42, n_jobs=-1)
rf.fit(X_sample, y_sample)

# Compute importances
importances = rf.feature_importances_
indices = np.argsort(importances)[::-1]
top_columns = X_sample.columns[indices[:30]].tolist()

# Plot top 20 importances
plt.figure(figsize=(10,6))
plt.title('Figure 3.4: Random Forest Feature Importances')
plt.bar(range(20), importances[indices[:20]])
plt.xticks(range(20), X_sample.columns[indices[:20]], rotation=90)
plt.tight_layout()
plt.show()

# Save top 30 features
with open('feature_list.json', 'w') as f:
    json.dump(top_columns, f)


# ===== Cell 6 =====

X_selected = X[top_features]
X_train, X_temp, y_train, y_temp = train_test_split(X_selected, y, test_size=0.3, stratify=y,
random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, stratify=y_temp,
random_state=42)

# ===== Cell 7 =====

rf_final = RandomForestClassifier(n_estimators=100, random_state=42)
rf_final.fit(X_train, y_train)
with open('rf_model.pkl', 'wb') as f:
    pickle.dump(rf_final, f)
```

```
# ===== Cell 8 =====
from collections import Counter
print(Counter(y_train))

# ===== Cell 9 =====
# ---------- 0 | imports & reproducibility ------------------
import numpy as np
import pandas as pd
from sklearn.preprocessing     import LabelEncoder
from sklearn.model_selection   import RandomizedSearchCV
from sklearn.utils.class_weight import compute_class_weight
from sklearn.metrics           import classification_report, confusion_matrix
from xgboost                   import XGBClassifier

RANDOM_STATE = 42
GPU_ACCEL    = True        # False if no CUDA GPU

# ---------- 1 | encode labels once --------------------------
le = LabelEncoder()
y_train_enc = le.fit_transform(y_train)
y_test_enc  = le.transform(y_test)
class_names = le.classes_

# ---------- 2 | balanced class-weights ----------------------
classes   = np.unique(y_train_enc)
weights   = compute_class_weight('balanced', classes=classes, y=y_train_enc)
class_wt  = dict(zip(classes, weights))
row_wt    = np.vectorize(class_wt.get)(y_train_enc)   # weight per sample

# ---------- 3 | cast features → float32 (↓ RAM ×2) ----------
X_tr32 = X_train.astype('float32')
X_te32 = X_test.astype('float32')

# ===== Cell 10 =====
# ---------- 4 | XGBoost model -------------------------------
xgb = XGBClassifier(
    objective   = 'multi:softprob',
    tree_method = 'hist',
    device = 'cuda',
    eval_metric = 'mlogloss',
    n_estimators= 400,
    subsample   = 0.8,
    random_state= RANDOM_STATE,
    n_jobs      = 1          # single thread keeps memory flat
)

# ---------- 5 | hyper-param search --------------------------
param_dist = {
    'max_depth':        [6, 10, 14],
    'learning_rate':    [0.05, 0.1, 0.2],
    'colsample_bytree': [0.7, 0.9, 1.0],
}
```

```
search = RandomizedSearchCV(
    xgb,
    param_dist,
    n_iter      = 6,
    scoring     = 'f1_macro',
    cv          = 3,
    verbose     = 2,
    random_state= RANDOM_STATE
)

# ===== Cell 11 =====
# ---------- 6 | fit & evaluate ----------------------------
search.fit(X_tr32, y_train_enc, sample_weight=row_wt)
print("\n⇢ Best params:", search.best_params_)

best_model  = search.best_estimator_
y_pred_enc  = best_model.predict(X_te32)
y_pred      = le.inverse_transform(y_pred_enc)

print("\nClassification report:")
print(classification_report(y_test, y_pred, target_names=class_names))

print("\nConfusion matrix:")
print(pd.DataFrame(confusion_matrix(y_test, y_pred),
                   index=class_names, columns=class_names))

# ===== Cell 12 =====
# --- Final test -------------------------------------------------
y_test_pred = search.best_estimator_.predict(X_test)
y_test_pred_str = le.inverse_transform(y_test_pred)      # <- fix

print(classification_report(y_test, y_test_pred_str, digits=4))

from sklearn.metrics import ConfusionMatrixDisplay
import matplotlib.pyplot as plt

ConfusionMatrixDisplay.from_predictions(
    y_test, y_test_pred_str,
    xticks_rotation=45,
    cmap='Blues',
    values_format=''          # hides the text overlays
)
plt.tight_layout()
plt.show()
plt.savefig('confusion_matrix_xgb.png')  # Save as image
plt.close()

# --- Save artefacts ---------------------------------------------
import joblib, json

joblib.dump(search.best_estimator_, 'best_model.pkl')
with open('feature_list.json', 'w') as f:
```

```
    json.dump(X_test.columns.tolist(), f)

print('Model and files saved in current directory.')

# ===== Cell 13 =====
models = {'Random Forest': rf_final, 'XGBoost': best_model}
metrics_table = {}

for name, model in models.items():
    y_pred = model.predict(X_test)

    # Convert to string labels if XGBoost (since it's using label encoding)
    if name == 'XGBoost':
        y_pred = le.inverse_transform(y_pred)

    report = classification_report(y_test, y_pred, output_dict=True)
    metrics_table[name] = report['weighted avg']
    cm = confusion_matrix(y_test, y_pred)
    ConfusionMatrixDisplay(cm).plot()
    plt.title(f'Confusion Matrix: {name}')
    plt.show()

# Create performance table
df_perf = pd.DataFrame(metrics_table).T[['precision', 'recall', 'f1-score']]
print("Table 3.2: Precision, Recall, F1-Score Comparison")
print(df_perf)


# ===== Cell 14 =====

df_perf['f1-score'].plot(kind='bar', title='F1-Score Comparison: Random Forest vs XGBoost')
plt.ylabel("F1 Score")
plt.tight_layout()
plt.show()

# ===== Cell 15 =====

rf_imp = pd.Series(rf_final.feature_importances_,
index=X_selected.columns).sort_values(ascending=False)
xgb_imp = pd.Series(best_model.feature_importances_,
index=X_selected.columns).sort_values(ascending=False)

plt.figure(figsize=(10,5))
rf_imp.head(10).plot(kind='bar', alpha=0.6, label='RF')
xgb_imp.head(10).plot(kind='bar', alpha=0.6, color='orange', label='XGB')
plt.title("Feature Importance Comparison")
plt.ylabel("Importance Score")
plt.legend()
plt.tight_layout()
plt.show()

# Table for overlapping & unique features
overlap = set(rf_imp.head(10).index).intersection(xgb_imp.head(10).index)
```

```
unique_rf = set(rf_imp.head(10).index) - overlap
unique_xgb = set(xgb_imp.head(10).index) - overlap

print("Overlapping Features:", list(overlap))
print("Unique to RF:", list(unique_rf))
print("Unique to XGB:", list(unique_xgb))
```

Phase-2 Code

```
# ===== Cell 1 =====
# Imports
import pandas as pd
import pickle
import numpy as np
np.bool = bool
np.int = int
import shap
import lime
import lime.lime_tabular
import matplotlib.pyplot as plt

#  Load Preprocessed Data
df = pd.read_csv('cicids2017_preprocessed_phase1.csv')

# Separate features and label
X = df.drop('Label', axis=1)
y = df['Label']

#  Load Trained Model
with open('../models/best_model.pkl', 'rb') as file:
    model = pickle.load(file)

# Quick Check
print("Data Shape:", X.shape)
print("Model Loaded Successfully.")

# ===== Cell 2 =====
# Lowercase all columns to match training
X.columns = X.columns.str.lower()

# Predict probabilities
y_pred_proba = model.predict_proba(X)

# Pick confident and low confidence samples
confident_idx = np.argmax(np.max(y_pred_proba, axis=1))
low_confidence_idx = np.argmin(np.max(y_pred_proba, axis=1))

print("High-confidence sample index:", confident_idx)
print("Low-confidence sample index:", low_confidence_idx)
```

```
# 1. Select your samples
high_conf_sample = X.iloc[[confident_idx]]  # confident sample
low_conf_sample = X.iloc[[low_confidence_idx]]  # low confidence sample

# ===== Cell 3 =====
explainer = shap.Explainer(model.predict, X)
shap_values_high = explainer(high_conf_sample)
shap_values_low = explainer(low_conf_sample)

# SHAP Waterfall Plots
shap.plots.waterfall(shap_values_high[0], show=True)
shap.plots.waterfall(shap_values_low[0], show=True)

# ===== Cell 4 =====
explainer_lime = lime.lime_tabular.LimeTabularExplainer(
    training_data=X.values,
    feature_names=X.columns,
    class_names=['Benign', 'Malicious'],
    mode='classification'
)

lime_exp_high = explainer_lime.explain_instance(high_conf_sample.values[0], model.predict_proba)
lime_exp_low = explainer_lime.explain_instance(low_conf_sample.values[0], model.predict_proba)

# Display LIME results
lime_exp_high.show_in_notebook()
lime_exp_low.show_in_notebook()

# ===== Cell 5 =====
# SHAP value spreads
shap_values_all = explainer(X.sample(n=500, random_state=42))
confidences_all = np.max(model.predict_proba(X.sample(n=500, random_state=42)), axis=1)
spreads = [np.std(val.values) for val in shap_values_all]

# Plot confidence vs SHAP value spread
plt.figure(figsize=(8,5))
plt.scatter(confidences_all, spreads, alpha=0.5)
plt.xlabel("Prediction Confidence")
plt.ylabel("SHAP Value Spread (std dev)")
plt.title("Confidence vs SHAP Attribution Spread")
plt.grid(True)
plt.show()

# ===== Cell 6 =====
import pandas as pd

summary_df = pd.DataFrame({
    "Confidence": confidences_all,
    "SHAP Spread": spreads
})

summary_df.describe()
```

```
# ===== Cell 7 =====
# Save explanations (optional)
summary_df.to_csv("shap_confidence_spread_summary.csv", index=False)

# ===== Cell 8 =====
# ——— 4.2 Feature Attribution Shift
————————————————————————————————————————————————————————

import json
import pickle
import pandas as pd
import matplotlib.pyplot as plt

# Load your selected_features list (the exact order used during Phase 1 training)
with open('feature_list.json','r') as f:
    selected_features = json.load(f)

# Load both trained models
with open('rf_model.pkl','rb') as f:
    rf = pickle.load(f)
with open('best_model.pkl','rb') as f:
    xgb = pickle.load(f)

#  Build RF feature-importance DataFrame (one importance per selected feature)
rf_imp = pd.DataFrame({
    'feature': selected_features,
    'rf_importance': rf.feature_importances_
})

#  Build XGB gain-based importance DataFrame
#   we only consider features that actually appear in the booster.get_score() dict
xgb_booster = xgb.get_booster()
gain_dict = xgb_booster.get_score(importance_type='gain')

# Convert to DataFrame, then normalize to sum=1
xgb_imp = (
    pd.DataFrame.from_dict(gain_dict, orient='index', columns=['xgb_gain'])
      .rename_axis('feature')
      .reset_index()
)
xgb_imp['xgb_importance'] = xgb_imp['xgb_gain'] / xgb_imp['xgb_gain'].sum()

# ——— Merge importances
————————————————————————————————————————————————————————
—
df_imp = (
    rf_imp.merge(xgb_imp[['feature','xgb_importance']], on='feature', how='left')
          .fillna(0)   # any feature RF had but XGB didn't will get zero gain
)

# ——— Pick top N from each model
————————————————————————————————————————————————————————

top_n = 20
```

```
rf_top  = df_imp.nlargest(top_n, 'rf_importance')
xgb_top = df_imp.nlargest(top_n, 'xgb_importance')

# ── Plot side-by-side bar charts ──────────────────────────────────────────
fig, (ax1, ax2) = plt.subplots(1,2, figsize=(14,6), sharey=True)

rf_top.sort_values('rf_importance')\
     .plot.barh(x='feature', y='rf_importance', ax=ax1, color='C0', legend=False)
ax1.set_title('Random Forest Top Features')
ax1.set_xlabel('RF Importance')

xgb_top.sort_values('xgb_importance')\
      .plot.barh(x='feature', y='xgb_importance', ax=ax2, color='C1', legend=False)
ax2.set_title('XGBoost Top Features')
ax2.set_xlabel('XGB Gain-Normalized Importance')

plt.tight_layout()
plt.show()

# ── Build overlap/unique summary table ────────────────────────────────────
rf_set  = set(rf_top['feature'])
xgb_set = set(xgb_top['feature'])

summary = pd.DataFrame({
   'overlap': pd.Series(sorted(rf_set  & xgb_set)),
   'rf_only': pd.Series(sorted(rf_set  - xgb_set)),
   'xgb_only':pd.Series(sorted(xgb_set - rf_set))
})

print("\nFeature Importance Summary (top 20 each):")
display(summary)

# ===== Cell 9 =====
# For your high/low samples
def top_shap(sample, shap_vals, feature_names, n=5):
   vals = list(zip(feature_names, shap_vals.values[0]))
   return sorted(vals, key=lambda x: abs(x[1]), reverse=True)[:n]

high_top5 = top_shap(high_conf_sample, shap_vals_high, X.columns)
low_top5  = top_shap(low_conf_sample,  shap_vals_low,  X.columns)

df_loc = pd.DataFrame({
   "High-conf idx 32": [f"{f}: {v:.2f}" for f,v in high_top5],
   "Low-conf idx 2296201": [f"{f}: {v:.2f}" for f,v in low_top5]
})
display(df_loc)
```