الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي
جامعة الشهيد الشيخ العربي التبسي - تبسة

كلية العلوم و التكنولوجيا
قسم الإلكترونيك و الاتصالات

تبسة في :.16/06/2026

إشهاد

يشهد رئيس لجنة المناقشة الخاصة بمذكرة التخرج لمستوى ماستر الموسومة بـ:

Evaluating Explainable Artificial Intelligence (XAI) in the Network Security Domain A case Study

من أنجاز الطلبة:

الرقم	الاسم واللقب	الشعبة	التخصص	السنة الجامعية
1	حمو دشتي صداع العيش	الاتصالات	إتصالات سلكية و لاسلكية	2026/2025
2				
3				

قد استكملا تصحيح الأخطاء وتقيدا بالملاحظات المقدمة من طرف لجنة المناقشة.

رئيس لجنة المناقشة

جامعة الشهيد الشيخ العربي التبسي
قسم الإلكترونيك والاتصالات
كلية العلوم والتكنولوجيا

الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria
وزارة التعليم العالي والبحث العلمي
Ministry of Higher Education and Scientific Research
جامعة الشهيد الشيخ العربي التبسي - تبسة
Echahid Cheikh Larbi Tebessi University-Tebessa
Faculty of Science and Technology
Department of Electronics and Telecommunications

Master's Thesis

Submitted for the degree of Academic Master

Field: Telecommunications

Specialty: Networks and Telecommunications

Presented by: (Haouichi Salah Eddine)

Title

Evaluating Explainable Artificial Intelligence (XAI) in the Network Security Domain A Case Study

Defended and evaluated on 09/ 06 / 2025, by the jury composed of:

First and last name		Rank	Position
Mrs.	Mohammed Saigaa	MCB	Reviewer
Mr.	Aouiche Chaima	MCB	Supervisor
Mr.	Djelab Hanane	Pr	Chair

Graduating Class: 2025/2026

Evaluating Explainable Artificial Intelligence (XAI) in the Network Security Domain A Case Study

Haouichi Salah Eddine

Larbi Tébessi University

Dedication

I dedicate this modest work

To my dear parents, ***Dr. Haouichi Nacer Eddine*** *and my beloved mother,* ***Bousba Halima*** *for their endless support, love, encouragement, and sacrifices. Nothing in this world can ever repay what they have done for me, and without them, I would never have been able to walk this path.*

May God protect and bless them.

To my sisters and brother, ***Amina, Imen, and Amin****, for their love, encouragement, and unwavering support.*

To all my maternal and paternal family members, for their kindness and constant support.

Without forgetting all the teachers who have contributed to my education, whether in primary, middle, secondary, or higher education.

To all the people who are dear to my heart, I dedicate this work.

Haouichi Salah Eddine

Acknowledgements

First and foremost, I would like to thank Almighty God for granting me the will, health, and patience throughout all these years of study.

I would also like to sincerely thank the members of the jury for accepting to examine this work and for the honor they have given me by evaluating my research and participating in the defense of this dissertation.

I would like to express my deep gratitude to my supervisor, Dr. Aouiche Chaima, for her valuable advice, guidance, and availability throughout the entire period of this work, which greatly contributed to its success.

I would also like to take this opportunity to express my gratitude to everyone who contributed in one way or another, especially my colleagues, my family, and my friends.

I would especially like to express my heartfelt gratitude to my beloved family, particularly my mother, father, and sisters. They have always shown interest in my studies and success and have supported me far more than they can imagine.

My sincere thanks also go to all my relatives and friends who continuously supported and encouraged me during the completion of this dissertation. Truly, thank you very much.

Thank you all!

To all those whom I love...

To all those who love me...

Table of Contents

Abstract
Résumé
الملخص
Dedication
Acknowdgements

Chapter 1: Foundations of Explainable Artificial Intelligence in Network Security

1.1 Introduction..... 01
1.2 Artificial Intelligence and the Need for Explainability..... 02
1.3 Explainable Artificial Intelligence: Concepts and Principles..... 03
1.3.1 Black-Box Models..... 04
1.3.2 White-Box Models..... 05
1.4 Deep Learning: Overview and Core Concepts..... 07
1.4.1 Functional Description of Deep Learning..... 07
1.4.2 Deep Learning Techniques..... 08
1.4.3 Ensemble Learning Techniques..... 10
1.5 The Role of XAI in Network Security..... 13
1.5.1 Alert Triage in the SOC..... 13
1.5.2 Model Validation and Bias Detection..... 13
1.5.3 Adversarial Robustness Assessment..... 14
1.6 Challenges of Applying AI and Explainable AI in Cybersecurity..... 15
1.7 Research Gap Underlying Motivations..... 17
1.8 Problematique..... 19
1.9 Conclusion..... 20

Chapter 2: Research Design and Methodological Framework

2.1 Introduction..... 22
2.2 Dataset Collection and Selection Rationale..... 23
2.3 Feature Description..... 24
2.4 Attack Categories and Classification Problem Formulation..... 24
2.4.1 Attack Taxonomy in the CIC IoT-DIAD 2024 Dataset..... 24
2.4.2 Binary Classification Problem Formulation..... 25
2.5 Dataset Inspection and Preprocessing..... 26
2.5.1 Samples Reduction..... 26
2.5.2 Systematic Feature Selection..... 27
2.5.3 Class balancing..... 30
2.5.4 Features Scaling..... 32
2.5.5 train-Test Split..... 33
2.6 Experimental Environment and System Configuration..... 34
2.6.1 Computational Infrastructure..... 34
2.6.2 Software Environment and Library Stack..... 34

2.7 Model Selection Strategy..... 35
2.7.1 Deep Neural Network (DNN)..... 35
2.7.2 Bagging Ensemble (Random Forest)..... 37
2.7.3 Stacking Ensemble..... 38
2.8 Integration of explainability into the prposed framwork..... 39
2.8.1 SHAP Framework Overview..... 40
2.8.2 Explainer Selection per Model..... 40
2.8.3 Global and Local Explanation Modes..... 42
2.8.4 Cross-Model Consistency Analysis 43
2.9 Conclusion..... 45

Chapter 3: Experimental Results and Discussion

3.1 Introduction..... 47
3.2 Model Training and Hyperparameter Optimization..... 48
3.2.1 DNN Training Dynamics..... 48
3.2.2 Bagging Training Dynamics..... 49
3.2.3 Stacking Training Dynamics..... 50
3.3 Model Validation and Performance Evaluation..... 51
3.3.1 Deep Neural Network Results..... 52
3.3.2 Bagging Performance Results..... 53
3.3.3 Stacking Performance Results..... 54
3.3.4 Comparative Performance Summary..... 56
3.4 Explainability Results and Interpretation..... 55
3.4.1 Bagging SHAP Results..... 58
3.4.2 DNN SHAP Results..... 61
3.4.3 Stacking SHAP Results..... 63
3.5 Comparative analysis of models..... 66
3.5.1 Rank Correlation Analysis..... 66
3.5.2 Cross-Model Consensus Features..... 68
3.5.3 Model-Specific Divergences 69
3.6 Descession of Findings..... 71
3.6.1 Performance Implications..... 71
3.6.2 Explainability Implications 71
3.6.3 Model Selection Recommandation 72
3.7 Conclusion..... 75

Chapter 4: Conclusion and Future Direction

4.1 Conclusion and Current Study Limitations..... 77
4.2 Future Directions 78
Bibliography 80

List of Tables

Table Number	Caption/Title	Page Number
Table 1.1	Principal AI application domains in cybersecurity, with corresponding objectives and representative algorithmic approaches	Page 3
Table 1.2	Comparison of black-box and white-box categories by interpretability and representative examples	Page 6
Table 1.3	Deep Learning Architectures and Their Network Security Applications	Page 9
Table 1.4	Ensemble Learning Methods: Performance and Explainability Characteristics	Page 12
Table 1.5	Summary of Critical Challenges, Risks, and Mitigation Approaches in Explainable AI for Cybersecurity Systems	Page 17
Table 2.1	Summary characteristics of the CIC IoT-DIAD 2024 dataset	Page 24
Table 2.2	Attack taxonomy of the CIC IoT-DIAD 2024 dataset, organized by functional category with primary attack targets.	Page 25
Table 2.3	The 36 retained features organised by functional category, with operational interpretation relevant to IoT anomaly detection	Page 28
Table 2.4	Dataset composition at each stage of the preprocessing pipeline, showing the effect of majority-class undersampling on class distribution	Page 31
Table 2.5	Hardware specifications of the Kaggle computational environment used across all experimental runs	Page 34
Table 2.6	Software libraries and versions used across all experimental stages	Page 34
Table 2.7	DNN architecture and training configuration	Page 36
Table 2.8	Bagging (Random Forest) configuration	Page 37
Table 2.9	Stacking ensemble configuration	Page 39

Table 3.1	DNN epoch-level training metrics at selected epochs, illustrating convergence from initialisation to final stable performance.	Page 48
Table 3.2	DNN classification report on the held-out test set (n = 159,273)	Page 52
Table 3.3	Bagging (Random Forest) classification report on the held-out test set (n = 159,273)	Page 53
Table 3.4	Stacking ensemble classification report on the held-out test set (n = 159,273)	Page 55
Table 3.5	Consolidated performance comparison across all three architectures on the held-out test set (n = 159,273)	Page 56
Table 3.6	Pairwise Spearman rank correlations between SHAP feature importance rankings across the three model architectures	Page 66
Table 3.7	Cross-model consensus features — rank assigned by each architecture and the primary attack mechanism each feature captures	Page 69
Table 3.8	Summary of key SHAP attribution divergences across the three model architectures, with structural interpretation	Page 70
Table 3.9	Model selection recommendations across key operational dimensions	Page 73

List of Figures

Figure Number	Caption / Title	Page Number
Figure 1.1	The fundamental shift from rule-based to learning-based security systems, and the role XAI plays in combining their respective strengths	Page 2
Figure 1.2	Mathematical Architecture of the Artificial Neuron	Page 8
Figure 1.3	Ensemble Learning Pipeline with Integrated XAI Output	Page 12
Figure 1.4	XAI-enhanced alert generation in a Security Operations Centre	Page 13
Figure 1.5	XAI-driven validation gate before model deployment	Page 14
Figure 1.6	XAI as a prerequisite for adversarial threat modelling	Page 15
Figure 2.1	Data collection and preprocessing pipeline for the CIC IoT-DIAD 2024 dataset	Page 23
Figure 2.2	Binary class distribution of the CIC IoT-DIAD 2024 dataset prior to balancing (n = 27,713,532).	Page 26
Figure 2.3	illustrates the class distribution before and after balancing	Page 32
Figure 2.4	Effect of StandardScaler on three representative features	Page 33
Figure 2.5	the conceptual relationship between global and local SHAP explanations and their respective roles in the proposed framework	Page 43
Figure 3.1	DNN Training and Validation Loss/Accuracy	Page 49
Figure 3.2	Bagging (Random Forest) "Top 15 Feature Importance (Gini Impurity)	Page 50
Figure 3.3	Stacking Feature Importance Learner Performance	Page 51
Figure 3.4	DNN Confusion Matrix on the held-out test set	Page 53
Figure 3.5	Bagging (Random Forest) Confusion Matrix on the held-out test set	Page 54

Figure 3.6	Stacking Confusion Matrix on the held-out test set	Page 55
Figure 3.7	Performance Comparison — DNN vs Bagging vs Stacking (36 features)	Page 56
Figure 3.8	SHAP Summary Bar Plot — Bagging (Random Forest), 500 test instances	Page 58
Figure 3.9	SHAP Beeswarm Plot — Bagging (Random Forest), 500 test instances	Page 59
Figure 3.10	SHAP Force Plot — Bagging, single malicious prediction (f(x) = 0.98).	Page 60
Figure 3.11	Global features importance (DNN)	Page 61
Figure 3.12	SHAP Summary Bar Plot — DNN, 200 test instances	Page 62
Figure 3.13	Global features importance (Stacking)	Page 64
Figure 3.14	SHAP Summary Bar Plot — **stacking**, 200 test instances	Page 65
Figure 3.15	Cross-Model SHAP Rank Correlation Analysis — heatmap and rank scatter plots	Page 67

Abstract

The rapid expansion of Internet of Things infrastructure has fundamentally altered the network security threat landscape, creating environments in which the volume, velocity, and diversity of malicious traffic exceed the capacity of rule-based detection systems and human analyst teams alike. Machine learning offers a powerful alternative, but the models that achieve the highest detection rates — deep neural networks and large ensemble architectures — are precisely those whose internal reasoning is most opaque, creating a critical gap between predictive capability and operational interpretability that undermines the trust and accountability requirements of real security operations.

This thesis investigates whether explainable artificial intelligence can bridge that gap in an IoT network intrusion detection context. A comparative experimental framework was constructed in which three architecturally distinct models — a Deep Neural Network, a Bagging ensemble based on Random Forest, and a two-level Stacking ensemble — were trained and evaluated on the CIC IoT-DIAD 2024 dataset, comprising 27,713,532 labelled network flow records described by 36 traffic-derived features. Each model was assessed not only on standard detection metrics but on the quality, consistency, and operational interpretability of the SHAP attributions it produced, using architecture-appropriate explainer classes: TreeExplainer for Bagging, DeepExplainer for DNN, and PermutationExplainer for Stacking.

All three architectures exceeded 97% accuracy and 97% F1-score on the balanced held-out test set. The Bagging ensemble achieved the highest precision (99.76%) and the lowest false positive count (188), while the Stacking ensemble achieved the highest recall (96.71%) and the lowest false negative count (2,622). SHAP analysis identified three features — SYN Flag Count, Bwd Packet Length Max, and Flow IAT Min — that appeared consistently in the top 10 rankings of all three architectures, providing a cross-model consensus that is both statistically robust and operationally interpretable in terms of known IoT attack mechanisms. Spearman rank correlation analysis revealed that attribution consistency is architecture-dependent: the tree-based Bagging and Stacking models produce strongly correlated rankings ($\rho = 0.7521$), while the DNN's attribution profile diverges significantly from both ensembles ($\rho \approx 0.12–0.20$).

The Bagging ensemble is identified as the recommended architecture for the evaluated scenario, optimising across detection reliability, explanation exactness, computational feasibility, and cross-model attribution consistency simultaneously. The study demonstrates that XAI-enhanced intrusion detection is both technically viable and operationally actionable for IoT security environments, and introduces cross-model rank correlation analysis as a practical framework for assessing explanation robustness in multi-model security deployments.

Keywords: Explainable Artificial Intelligence, Network Intrusion Detection, IoT Security, SHAP, Deep Neural Network, Bagging, Stacking, CIC IoT-DIAD 2024.

Résumé

L'expansion rapide des infrastructures de l'Internet des Objets a profondément modifié le paysage des menaces en matière de sécurité réseau, engendrant des environnements où le volume, la vitesse et la diversité du trafic malveillant dépassent les capacités des systèmes de détection basés sur des règles et des équipes d'analystes humains. L'apprentissage automatique constitue une alternative puissante, mais les modèles les plus performants — réseaux de neurones profonds et méthodes d'ensemble — sont précisément ceux dont le raisonnement interne est le plus opaque, créant un fossé entre capacité prédictive et interprétabilité opérationnelle.

Ce mémoire examine si l'intelligence artificielle explicable (XAI) peut combler ce fossé dans le contexte de la détection d'intrusions réseau pour les environnements IoT. Un cadre expérimental comparatif a été construit, dans lequel trois architectures distinctes — un réseau de neurones profond (DNN), un ensemble Bagging basé sur la forêt aléatoire, et un ensemble Stacking à deux niveaux — ont été entraînées et évaluées sur le jeu de données CIC IoT-DIAD 2024, comprenant 27 713 532 enregistrements de flux réseau étiquetés décrits par 36 caractéristiques. Chaque modèle a été évalué non seulement sur des métriques de détection standard, mais également sur la qualité, la cohérence et l'interprétabilité opérationnelle des attributions SHAP produites.

Les trois architectures ont dépassé 97% de précision et 97% de F1-score sur l'ensemble de test. L'ensemble Bagging a atteint la précision la plus élevée (99,76%) et le nombre de faux positifs le plus faible (188), tandis que l'ensemble Stacking a obtenu le meilleur rappel (96,71%) et le nombre de faux négatifs le plus faible (2 622). L'analyse SHAP a identifié trois caractéristiques — SYN Flag Count, Bwd Packet Length Max et Flow IAT Min — apparaissant systématiquement dans le top 10 des trois architectures. La corrélation de rang de Spearman a révélé que la cohérence des attributions est dépendante de l'architecture : Bagging et Stacking produisent des classements fortement corrélés ($\rho = 0{,}7521$), tandis que le DNN diverge significativement ($\rho \approx 0{,}12$–$0{,}20$).

L'ensemble Bagging est recommandé comme architecture optimale, combinant fiabilité de détection, exactitude des explications et faisabilité computationnelle.

Mots-clés : Intelligence Artificielle Explicable, Détection d'Intrusions Réseau, Sécurité IoT, SHAP, Réseau de Neurones Profond, Bagging, Stacking, CIC IoT-DIAD 2024.

الملخص

أدى التوسع السريع في البنى التحتية لإنترنت الأشياء إلى تغيير جوهري في مشهد التهديدات الأمنية للشبكات، مما أفرز بيئات يتجاوز فيها حجم حركة المرور الخبيثة وسرعتها وتنوعها قدرات أنظمة الكشف القائمة على القواعد وفرق المحللين البشريين على حدٍّ سواء. يقدم التعلم الآلي بديلاً فعّالاً، غير أن النماذج الأعلى أداءً — الشبكات العصبية العميقة وطرق التجميع الكبيرة — هي بالضبط تلك التي يكون تفكيرها الداخلي أكثر غموضاً، مما يخلق فجوة حرجة بين القدرة التنبؤية وقابلية التفسير العملي.

تبحث هذه المذكرة في إمكانية تجسير الذكاء الاصطناعي القابل للتفسير لهذه الفجوة في سياق اكتشاف اختراقات الشبكة لبيئات إنترنت الأشياء. تم بناء إطار تجريبي مقارن تم فيه تدريب وتقييم ثلاثة نماذج ذات بنى معمارية مختلفة — شبكة عصبية عميقة، ومجموعة Bagging مبنية على الغابة العشوائية، ومجموعة Stacking ثنائية المستوى — على مجموعة بيانات CIC IoT-DIAD 2024 التي تضم 27,713,532 سجلاً لتدفقات الشبكة موصوفاً بـ 36 ميزة مستخرجة من حركة المرور. تم تقييم كل نموذج ليس فقط وفق مقاييس الكشف القياسية، بل أيضاً وفق جودة واتساق وقابلية التفسير العملي لقيم SHAP المنتجة، باستخدام فئات محلِّل مناسبة لكل بنية معمارية.

تجاوزت الأنظمة الثلاثة 97% في الدقة الكلية ومقياس F1 على مجموعة الاختبار. حقق نموذج Bagging أعلى دقة تنبؤية (99.76%) وأدنى عدد إيجابيات خاطئة (188)، فيما حقق نموذج Stacking أعلى استدعاء (96.71%) وأدنى عدد سلبيات خاطئة (2,622). كشف تحليل SHAP عن ثلاث ميزات — عدد علامات SYN، والحد الأقصى لطول حزمة الاتجاه الخلفي، والحد الأدنى للوقت بين الوصول في التدفق — تظهر باستمرار ضمن أفضل 10 ترتيبات للأنظمة الثلاثة، مما يوفر توافقاً عبر النماذج يمكن تفسيره عملياً بحسب آليات هجمات إنترنت الأشياء المعروفة. كشف تحليل ارتباط رتبة سبيرمان أن اتساق نسب التفسير يعتمد على البنية المعمارية: ينتج نموذجا Bagging وStacking ترتيبات مترابطة بشكل قوي ($\rho = 0.7521$)، بينما يختلف ملف التفسير الخاص بالشبكة العصبية العميقة اختلافاً جوهرياً عن كلا النموذجين ($\rho \approx 0.12–0.20$).

يُوصى بنموذج Bagging بوصفه البنية المثلى للسيناريو المُقيَّم، إذ يحقق التوازن الأمثل بين موثوقية الكشف، ودقة التفسير، والجدوى الحسابية.

الكلمات المفتاحية: الذكاء الاصطناعي القابل للتفسير، كشف اختراقات الشبكة، أمن إنترنت الأشياء، SHAP، الشبكة العصبية العميقة، Bagging، Stacking، CIC IoT-DIAD 2024

Chapter 1

Foundations of Explainable Artificial Intelligence in Network Security

1.1. Introduction:

Over the past twenty years, the reliance of businesses on digital infrastructure has grown dramatically. What was once a network of computers and servers confined to office spaces has now expanded into an intricate ecosystem encompassing cloud platforms, personal devices, mobile employees, industrial equipment, and partner systems–all interconnected and continuously exchanging information. While this integration enhances productivity and operational efficiency, it also introduces significant security challenges: as networks grow in scale and complexity, the number of potential vulnerabilities and therefore opportunities for cyberattacks rises correspondingly, keeping cybersecurity professionals on constant alert.

For many years, the standard response to this problem relied on rule-based approaches, Firewalls, intrusion detection systems, and blocklists all operate on the same basic principle: identify known malicious behavior, encode it as a rule, and block anything that matches. This approach worked reasonably well when the threat landscape was relatively stable and attacks followed recognizable patterns. However, it began to fall apart as attackers learned to adapt changing their tools frequently, blending malicious traffic with legitimate activity, and crafting attacks specifically designed to slip past known signatures.

Artificial intelligence (AI) offered a solution that rule-based systems could not: the ability to learn from data rather than explicit instructions. Machine learning enables systems to infer malicious behavior from examples, detecting subtle statistical patterns that humans would not have anticipated and generalizing to new attack variants, even when specific payloads are novel. Yet this capability comes with a significant trade-off. The most powerful models—deep neural networks and large ensemble methods—are often the least interpretable. When such a model flags a network connection as suspicious, it does not provide a clear explanation; it produces a score, leaving analysts uncertain whether an alert indicates a real threat or a statistical artifact.

Explainable Artificial Intelligence, or simply XAI, emerged as a direct response to this interpretability challenge. The core idea is not to abandon powerful models in favor of simpler ones, but to build or attach mechanisms that make model reasoning transparent for human operators. In a security context, that means providing analysts with not only the knowledge that a connection was flagged but also insights into which specific features —an unusual destination port, a data transfer volume that deviates from historical baselines, a connection to an IP associated with known command-and-control infrastructure — contributed to the decision and to what extent.

This chapter builds the foundation needed for understanding the role of XAI in network security. It examines how AI has become central to modern security operations, why the opacity of contemporary models creates critical challenges, and how XAI frameworks can bridge the gap between predictive power and interpretability. It then surveys the deep learning and ensemble methods most relevant to threat detection and concludes with an assessment of the practical difficulties that arise when these systems meet real-world operational constraints.

1.2. Artificial Intelligence and the Need for Explainability:

The adoption of AI in cybersecurity was driven not by strategic foresight but by the practical limitations of traditional rule-based systems. For more than three decades, security operations largely relied on signature- and rule-based methods, which compare system activity against databases of known threats, such as malware hashes, IP blocklists, and intrusion signatures. While these approaches were effective in relatively stable environments with predictable attack patterns. They began to fail for several reasons. First, they are inherently reactive: new threats must be reverse-engineered and encoded into the system before they can be detected, a process that can take days or even weeks, leaving networks vulnerable in the interim. Second, they lack the ability to generalize, meaning that minor changes in attacker tactics, like code packing or domain rotation, can make existing signatures useless. Lastly, as network traffic volumes and attack surfaces increased, manually maintaining rules became increasingly unsustainable.

Machine Learning (ML) offered a solution by enabling systems to learn directly from data rather than relying on pre-defined rules. Even mid-sized organizations generate tens of millions of log entries daily, while national telecom providers and major cloud platforms produce vastly larger volumes [1]. Manually analyzing such data is impractical, making automated approaches essential. ML can allow a trained classifier to analyze hundreds of thousands of network flows in seconds, and anomaly detection systems can constantly monitor network behavior, identifying deviations from the norm without requiring prior knowledge of the specific forms these deviations might take. Figure 1.1 illustrate the fundamental shift from rule-based to learning-based security systems, and the role XAI plays in combining their respective strengths.

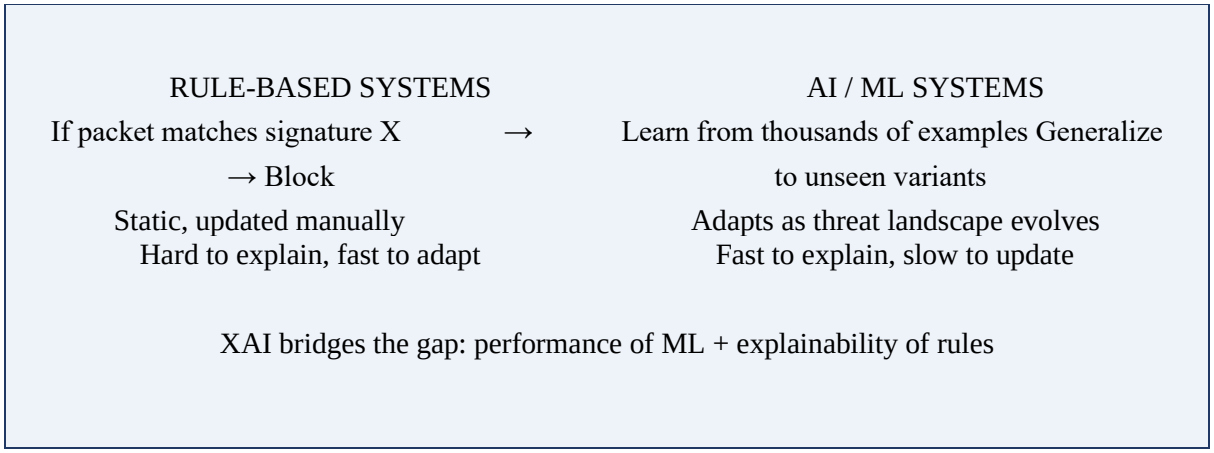


Figure 1.1: *The fundamental shift from rule-based to learning-based security systems, and the role XAI plays in combining their respective strengths.*

Beyond sheer processing speed, AI provides a qualitative advantage that is arguably even more significant: the ability to generalize to previously unseen variants. This capacity for generalization makes AI particularly valuable in the ongoing battle against constantly evolving threats.

AI applications now span the full spectrum of cybersecurity operations, including intrusion detection, malware identification, threat intelligence, vulnerabilities identification, behavioral analysis, and phishing detection. Each area leverages different ML techniques tailored to its

specific requirements. (Table 1.1). But this increased capability comes at a cost, one that the security research community has struggled to find an answer to for decades. The most powerful algorithms such as deep neural networks (DNN), and ensemble models are also the least interpretable. For example, the output of a deep network that detects a suspicious relationship is a probability score not an explanation. This opacity introduces three primary operational risks:

1. The alert fatigue caused by false positives [5].

2. The issue of adversarial attacks [6].

3. Silent degradation [7].

***Table 1.1**: Principal AI application domains in cyber security, with corresponding objectives and representative algorithmic approaches [3][4].*

Application Domain	Primary Security Objective	Representative AI Approach
Intrusion Detection & Prevention	Distinguish malicious from legitimate network traffic in real time	Supervised classification: Random Forest, Deep Neural Networks
Malware Analysis & Classification	Identify and categorize malicious code from static or dynamic features	Convolutional and recurrent neural networks on binary representations
Threat Intelligence & Attribution	Correlate indicators of compromise; attribute attacks to actor groups	Graph neural networks, unsupervised clustering
Vulnerability Discovery	Identify exploitable weaknesses in software and configurations	Reinforcement learning, fuzz-testing augmentation, NLP on code
User & Entity Behavior Analytics	Detect insider threats and compromised accounts through behavioral deviation	Statistical baseline, autoencoders, LSTM sequence models
Phishing & Social Engineering Detection	Recognize deceptive communications across email, web, and messaging channels	Transformer-based NLP, URL feature classifiers

In summary, the effectiveness of AI system for security does not depend solely on its performance in controlled benchmarks. Its true value lies in how well human operators can understand the system's reasoning and calibrate their trust in its predictions under dynamic, adversarial, and unpredictable conditions. While accuracy remains important, interpretability is the critical factor [8], this is the central objective of Explainable Artificial Intelligence (XAI).

1.3. Explainable Artificial Intelligence: Concepts and Principles:

Explainable Artificial Intelligence (XAI) represents a number of techniques, methodologies, and frameworks intended to help human users understand the functioning of AI systems. As ML models grow increasingly complex, especially within domains like cybersecurity, it becomes necessary not only to interpret the automated processes they employ but also to justify their outputs and decisions.

Unlike conventional methodologies, where model performance considered as key metric, XAI introduces a new methodological aspect: the ability for analysts to investigate the reasons behind the functioning/behavior of any given model and its specific predictions. This explainability is crucial for validating models and detecting potential risks, such as biased or unintended behavior. In particular, in the field of cybersecurity, failing to explain and understand model decisions may lead to serious operational and security consequences. Conceptually, it is important to differentiate between the following three properties:

- **Explainability**: the ability to comprehend how a model arrives at a particular decision.
- **Interpretability**: the ability to anticipate the model`s behavior in given scenarios.
- **Transparency**: the ability to understand the internal workings of the model.

These properties are connected but not synonymous. In essence, transparency implies interpretability, and interpretability entails explainability. Nonetheless, current advanced models typically lack both transparency and interpretability, necessitating the use of external explanation techniques [9].

XAI techniques can be divided based on two main parameters. The first relates to the scope of the explanation:

- Global explanations refer to describing how the model functions overall and what are the most influential features across the entire dataset.
- Local explanations describe what leads to specific decisions made by the algorithm for each individual input.

The second parameter concerns the timing of the explanation:

- Ante-hoc (intrinsic) explanations are integrated into the model design from the outset.
- Post-hoc explanations are applied after model training, particularly to opaque models, to make their decision interpretable.

These classifications offer a structured way to analyze and compare different XAI techniques depending on their intended use, operational constraints, and the level of insight they provide into the model’s decision-making process.

1.3.1. Black-Box Models:

Black box models can be considered as learning systems where it is difficult for humans to explain or understand how these systems make decisions internally. While black boxes can provide information on inputs and outputs, the transformation process is either hidden or too complex to analyze directly. Modern ML algorithms, often exhibit non-linear behavior because of the complex interplay among numerous parameters [10].

In cybersecurity, there are challenges related to black box models that are beyond theoretical problems concerning the lack of explainability. Cybersecurity experts do not care only if the

alert issued is valid, but also to understand why it is triggered. This particular problem arises when high-severity warnings has been generated without clear reasons, forcing analysts to decide whether the threat posed is real or merely an artifact of statistical correlation.

Another critical point is that the concept of the "black box" is relative; that is, depending on the point of view of the observer, it could either be an absolute black box or a box with elements that are interpretable. For example, what cannot be easily understood by an operational analyst might be partially explainable to an engineer who designed the feature pipeline or selected the training data. Nonetheless, in SOC settings, the focus lies not on potential interpretability, but rather on usability of the explanation for decision making.

Analysts usually consider three critical factors when evaluating output explanations of the models, namely: the reasons for which a certain alert was issued, the reliability of the data used as proof of maliciousness of the incident, and subsequent actions to be taken to investigate further. In cases where one of these needs is not satisfied, there is a substantial risk that the model will not be effectively incorporated into operational workflows.

To address these limitations, post-hoc interpretation tools may be employed to analyze black box behavior. Techniques like feature importance ranking or local model approximation aim to explain the role played by input data in generating particular predictions. Nevertheless, these techniques cannot entirely cockup the reasoning process underlying the algorithm because they only represent a simplification of it, which may exclude some critical connections [8]. Hence, their findings have to be regarded as guidance rather than full explanations, particularly in serious contexts such as threat identification.

Overall, black box models demonstrate an inherent contradiction between efficiency and interpretability that characterizes modern machine learning. In cybersecurity, this contradiction has practical implications: the ability of human analysts to properly respond to algorithms outputs depends significantly on the clarity and trustworthiness of the information provided.

1.3.2. White-Box Models:

White-box models are those whose internal mechanisms can be understood directly, without the need for extra tools or interpretation methods. Common examples include decision trees, linear and logistic regression, and rule-based classifiers. For each sample, a user can identify precisely which variables were examined, in what sequence, and according to what thresholds, to get the final output.

The advantage of using white-box models for security applications is obvious. For instance, when an analyst is given a rule such as "flag as malicious if packet size is larger than 1,400 bytes AND the destination port does not belong to the list of default services AND session duration is less than 0.3 seconds," she can assess whether such rule is reasonable based on domain expertise, and would likely cause false alarms in a certain type of legitimate traffic. Furthermore, rules and thresholds can be refined in consultation with technical experts without any specialized knowledge of ML.

The disadvantage of white-box models lies in the upper bound on their performance potential. If we consider high-dimensional security datasets, such as network flows with hundreds of

features or malware binaries with millions of indicators, shallow interpretable models typically underperform relative to deep neural networks or gradient-boosted ensembles. A central goal of XAI is to bridge this gap: to retain the high predictive power of black-box models while maintaining the interpretability and transparency characteristic of white-box approaches.

Among the white-box modeling techniques commonly applied in network security are:

- **Decision Trees**: Models that make classifications through hierarchical rules by sequentially splitting the input feature space. For example, for detecting network attacks, the decision tree could start by considering the packet rate, followed by the protocol type, and finally the port number, leading to a classification decision.
- **Linear and Logistic Regression**: These methods produce predictions as explicit linear combinations of input variables, with each coefficient reflecting the variable's contribution to the output. Logistic regression, in particular, is well-suited for binary classification tasks, providing clear insight into the relative importance of each feature in determining the prediction.
- **IF-THEN Rules Systems**: Models that use explicit IF-THEN rules that come from analyzing the training data or from the experience acquired from experts. These models may have relatively low accuracy compared to black-box models, but they offer maximum transparency, enabling analysts to follow the reasoning process step by step.
- **Generalized Additive Models (GAMs)**: Models that represent predictions using smooth functions of input variables, allowing them to capture non-linear relationships. In security analytics, GAMs are especially useful for identifying thresholds were normal behavior transitions into anomalous activity, such as session durations indicative of automated scripts rather than legitimate browsing.

To summarize the key differences between black-box and white-box approaches, Table 1.2 provides a concise comparison of model characteristics, interpretability, and typical examples.

***Table 1.2**: Comparison of black-box and white-box categories by interpretability and representative examples.*

Type	Description	Interpretability	Example Models
Black-Box	Model gives a result without revealing how it was reached	Low — requires post-hoc XAI tools	Deep Neural Networks, SVM
White-Box	Model logic is transparent and every step can be inspected	High — intrinsically interpretable	Decision Trees, Linear Regression

1.4. Deep Learning: Overview and Insights

Deep learning (DL) is a subfield of ML in which models composed of many successive layers of non-linear transformations learn hierarchical representations of data directly from examples. Rather than relying on hand-crafted features, these models discover the representations most relevant to a task by processing large labeled or unlabeled datasets. This representational capacity is the property that makes DL particularly effective for high-dimensional, unstructured data, such as raw network packets, binary executables, and extended sequences of event logs.

1.4.1. Functional Description of Deep Learning

DL term relates to artificial neural networks (ANN) having many layers, each representing hierarchical features of input data. It should be mentioned that the term "Deep" means the depth of these layers, reflecting the sequential transformations that progressively extract increasingly abstract representations required for tasks such as classification, detection, or prediction [11].

DL has great potential in network security applications due to its ability to automatically generate meaningful representations from raw input data. In particular, DL systems are capable of extracting relevant features directly from network traffic, enabling automated detection of patterns and anomalies without extensive manual intervention. In contrast, traditional ML systems for network intrusion detection, involve intensive manual effort in defining these useful features, a time-consuming and expertise-dependent process.

The basic building block of a DL model is the artificial neuron. Each neuron computes a weighted sum of its inputs, applies a non-linear activation function, and produces an output that serves as input to subsequent layers, allowing the network to learn complex, hierarchical relationships within the data. The operation of an artificial neuron is illustrated in Figure 1.2. Each input $\boldsymbol{x_i}$is multiplied by a corresponding weight $\boldsymbol{w_i}$, summed together with a bias term $\boldsymbol{b}$, and then passed through a non-linear activation function $\boldsymbol{\sigma}(\cdot)$ to produce the output $\boldsymbol{\hat{y}}$.

In network security applications, for example, lower layers may model simple statistics of network flows, such as packet size distributions and inter-arrival times. Intermediate layers can capture more intricate patterns, including protocol-specific behaviors, while higher layers may represent abstract concepts like attack strategies or overall communication behaviors.

Training DL models takes place via a technique called backpropagation, which is an algorithmic procedure for computing the gradient of the loss function with respect to all model parameters. These gradients are then used to adjust the parameters in order to minimize the loss, allowing the network to progressively improve its predictions.

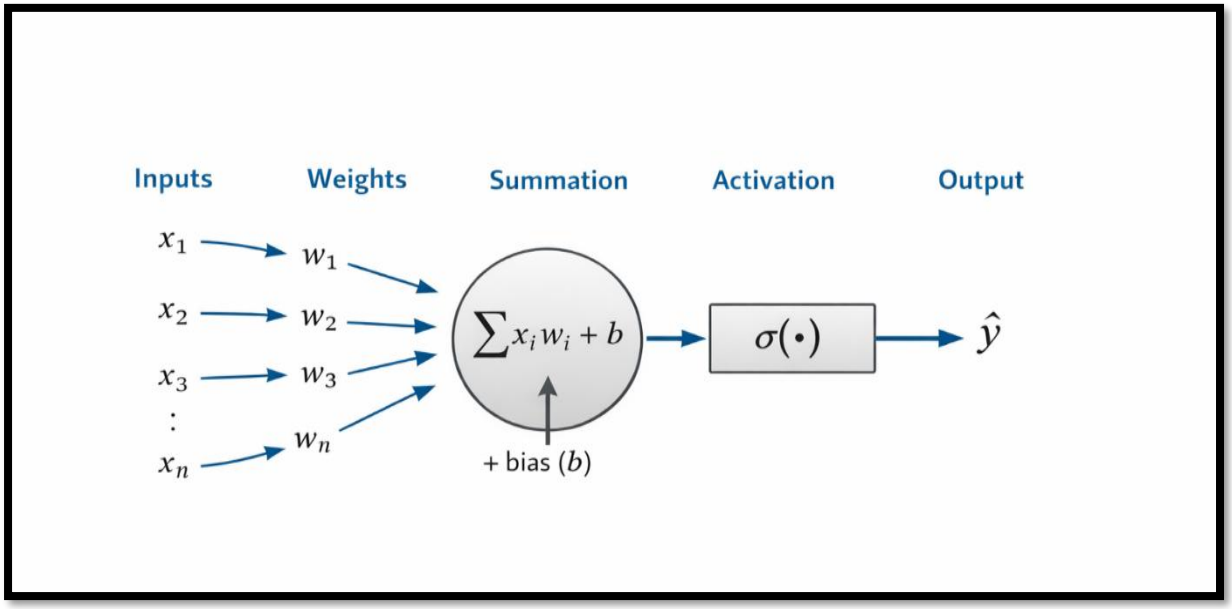


***Figure 1.2**: Mathematical Architecture of the Artificial Neuron*

1.4.2. Deep Learning Techniques:

The scope of DL algorithms is vast, including different types of architectures that vary based on their computational characteristics and the applicability to various network security tasks. Key techniques include:

- **Multilayer Perceptrons (MLP):** As the name suggest, MLPs are relatively simple DL architectures composed of an input layer followed by one or several fully connected hidden layers, and an output layer. In network security, MLPs are able to handle data in the form of a fixed feature vectors derived from network traffic, enabling the detection of anomalies or malicious activity.
- **Deep Neural Networks (DNN).** DNNs extend the MLP architecture by incorporating additional design elements Batch Normalisation, Dropout regularisation, and deeper layer stacks that address the training instability and overfitting problems that arise when shallow networks are scaled to high-dimensional security data. In the network intrusion detection context, DNNs operate on flow-based feature vectors and learn hierarchical representations in which lower layers capture simple statistical properties of traffic packet size distributions, inter-arrival time statistics while higher layers encode more abstract behavioural patterns that correspond to attack signatures. The expressiveness of DNNs makes them competitive with ensemble methods on tabular network traffic data, though their opacity represents the principal challenge for operational deployment: when a DNN flags a connection as malicious, the attribution of that decision to specific input features requires post-hoc explanation through tools such as SHAP DeepExplainer, which approximates Shapley values by propagating attribution signals backward through the network's computational graph [10]. This approximation introduces uncertainty at the instance level that exact tree-based methods do not, a distinction with direct operational implications for alert investigation.

- **Convolutional Neural Networks (CNNs):** Initially designed for use in computer vision, CNNs have proven to be valuable tools in securing networks via their use of network traffic as "image-like" input data. The process of transforming network packets into matrices using byte values or statistical properties allows the detection of spatially distributed features within the data corresponding to attack patterns. The ability of CNNs to recognize such features via learned filters renders the recognition process invariant to temporal changes common in traffic data.
- **Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM)**: Network traffic is inherently sequential, with attacks often exhibiting temporal dependencies across multiple connections. RNNs maintain an internal state to capture information from previous inputs, while LSTMs incorporate gating mechanisms to address the vanishing gradient problem and capture long-term dependencies effectively. These architectures are particularly useful for modeling temporal patterns in network flows.
- **Autoencoders:** Autoencoders are designed to reconstruct input data through a compressed latent representation. In the realm of cybersecurity, autoencoders are commonly used for anomaly detection. Training autoencoders on regular network activity allows the model to detect deviations, which manifest as reconstruction errors. Variational Autoencoders (VAEs) build on this approach by using probabilistic latent representations, enhancing their ability to detect anomalies and generate synthetic data.
- **Transformers and Attention-Based Architectures:** Transformers, designed initially for natural language processing, have exhibited great flexibility and applicability in a wide variety of applications, including network security such as log analysis, intrusion detection, and threat intelligence extraction. Attention-based algorithms enable the model to dynamically assign importance to input features, providing a degree of interpretability by highlighting which aspects of the data contribute most to predictions.
- **Generative Adversarial Networks (GANs):** GANs consist of two neural networks in competition: a generator that produces synthetic data and a discriminator that distinguishes real from generated samples. In network security, GANs are used mainly for data synthesis purposes, addressing class imbalance within the training datasets and generating adversarial samples to test the robustness of security systems.]

Table 1.3: *Deep Learning Architectures and Their Network Security Applications*

Architecture	Core Mechanism	Primary Security Applications	Datasets Commonly Used	XAI Compatibility
MLP	Fully connected layers	Intrusion detection, malware classification	NSL-KDD, CICIDS2017	SHAP, LIME
DNN	Deep fully connected + BatchNorm + Dropout	Intrusion detection, flow classification	CIC IoT-DIAD 2024, CICIDS2017	DeepExplainer, GradientSHAP

CNN	Convolutional filters	Malware image analysis, traffic classification	MalImg, CICIDS2018	Grad-CAM, LIME
LSTM/RNN	Sequential processing	APT detection, session analysis	LANL, CICIDS2017	Attention weights, SHAP
Autoencoder	Reconstruction error	Anomaly detection, zero-day detection	CICIDS, KDD Cup	Latent space analysis
Transformer	Attention mechanism	Log analysis, threat intelligence	CAIDA, custom logs	Attention visualization
GAN	Adversarial generation	Data augmentation, attack simulation	Various	Limited (indirect)

1.4.3. Ensemble Learning Techniques:

The idea behind ensemble learning is that the combination of the outcomes of several different and independent algorithms –known as base or weak learners– results in more accurate predictions than those obtained from each algorithm separately (single model). In theory, this can be explained through the concept of bias vs. variance, wherein the use of an ensemble enables the reduction of the bias inherent to individual models while mitigating overfitting caused by variance in the training data.

Regarding network security, ensemble methods have consistently demonstrated their effectiveness across a variety of tasks. At the same time, their synergy with XAI approaches can be considered one of the breakthroughs in the field of threat detection, since they provide both high predictive performance and enhanced interpretability.

- **Random Forest (RF):** RF model trains a vast number of decision trees on a bootstrapped samples of data while considering only a randomly selected subset of features at each split. For classification problems, the ensemble prediction is determined by majority voting, while for regression tasks, it takes the average output of all trees. By purposefully introducing stochasticity at the level of variable selection, RF prevents individual decision trees from being very similar to one another, thereby reducing correlations among individual trees, improving the overall ensemble performance.

  In terms of interpretability, RF provides out-of-the-box feature importance scores based on either Gini importance (i.e., the mean decrease in impurity) or mean decrease in accuracy after permuting a certain feature. These metrics offer insights into which network variables are more important on average, although they may not fully explain the reasoning behind individual predictions.

- **Gradient Boosting Machines (XGBoost, LightGBM, AdaBoost):** The principle behind gradient boosting is constructing the ensemble of models sequentially, where each new model attempts to fit the residuals (errors) generated by previously constructed models. Such an iterative approach makes it possible for

gradient boosting to achieve outstanding results in terms of accuracy in processing structured/tabular data, such as processed network traffic records. For example, XGBoost has consistently shown top- tier performance on intrusion detection datasets, making it a popular choice in cybersecurity applications.

SHAP (SHapley Additive exPlanations) explanations are quite natural in the case of gradient boosting models based on decision trees. As there is efficient algorithms to compute SHAP values, which quantify the contribution of each input feature to a model's prediction, thereby enhancing interpretability and facilitating human understanding of the decision process.

- **Stacking (Stacked Generalization):** Stacking follows a completely different strategy compared to the previous techniques in the sense that it does not rely on a pre-defined method for combining predictions from multiple base models, rather, a meta-classifier is trained to learn the optimal combination strategy. The whole procedure involves two levels:
  - **Level 1 (Base classifiers)**: Multiple heterogeneous models are trained on the training dataset, each producing prediction results.
  - **Level 2 (Meta-classifier)**: A second-level classifier is trained on the outputs of the base classifiers, using them as input features to generate the final prediction.

  The main benefit of stacking is that it can leverage the complementarity among heterogeneous classifiers, meaning that each base classifier can capture unique patterns or perform better under specific conditions, while the meta-classifier learns to combine their strengths automatically. This characteristic is highly important in cybersecurity when different attacks possess distinct statistical characteristics. The primary drawback is that stacking requires careful training and tuning to avoid overfitting, which can increase computational complexity and development time.
- **Bagging (Bootstrap Aggregating)**: Bagging involves creating multiple sub-samples of the training set through the process of bootstrap sampling (random sampling with replacement) followed by training a classifier/regressor for each sample separately. The final prediction is obtained by aggregating the outputs of all models, using majority voting for classification or averaging for regression tasks.

  The main advantage of bagging lies in its ability to reduce correlation between model errors, thereby improving overall accuracy. It is particularly effective with unstable base models, such as decision trees, which vary significantly with small changes in input data. One of the most widely used implementations of bagging is the RF algorithm, which combines many decision trees to produce robust predictions.
  The overall architecture of ensemble learning in network security is illustrated in Figure 1.3. As shown, multiple base learners, including decision trees, neural networks, and other models are combined through an aggregation layer, which can use voting, weighted averages, or stacking. The final prediction is supplemented by XAI explanations to ensure interpretability.

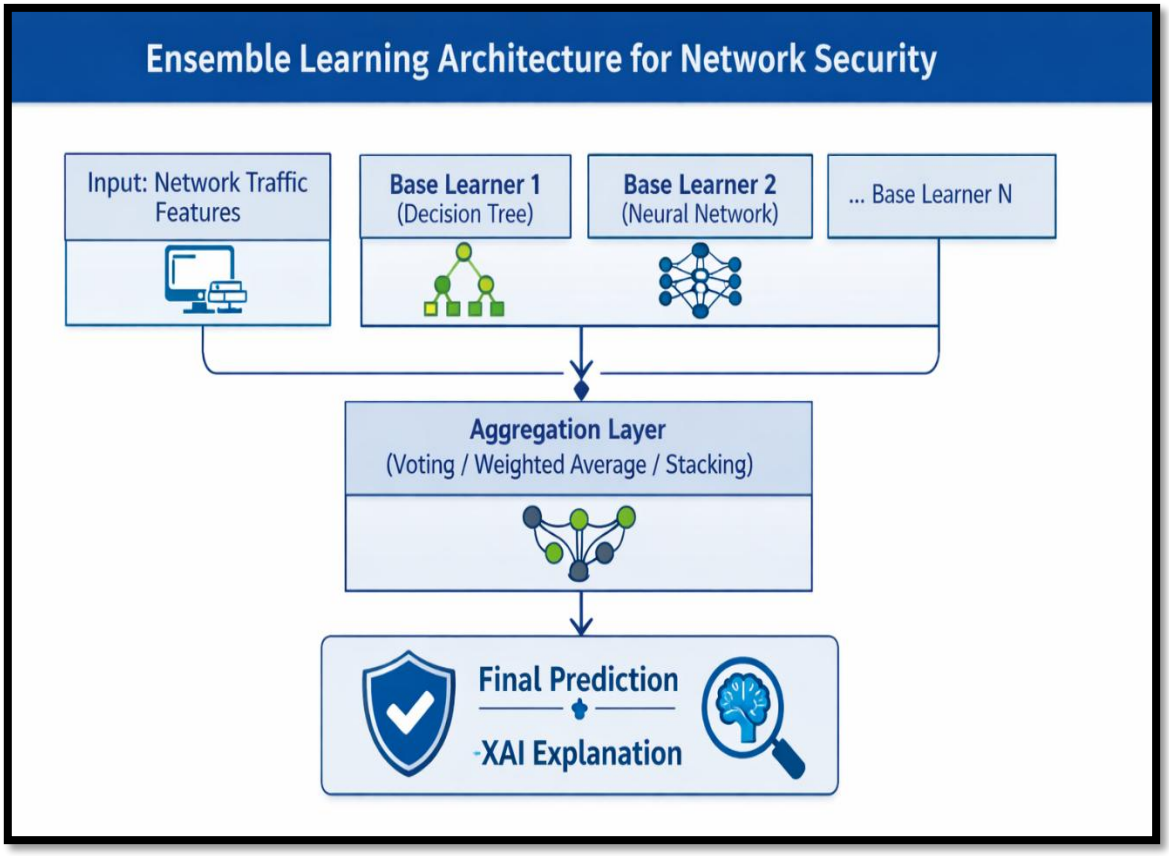


Figure 1.3: *Ensemble Learning Pipeline with Integrated XAI Output*

Table 1.4 summarizes key characteristics of the main ensemble learning methods used in cybersecurity, including training strategy, diversity mechanism, predictive performance, training speed, XAI integration, and application advantages.

Table 1.4 *— Ensemble Learning Methods: Performance and Explainability Characteristics*

Method	Training Strategy	Diversity Mechanism	Accuracy	Training Speed	XAI Integration	Cybersecurity Advantage
Random Forest	Parallel (Bagging)	Feature & data randomness	High	Fast	SHAP, permutation importance	Handles noise well; manages class imbalance
XGBoost	Sequential (Boosting)	Error correction	Very High	Moderate	SHAP (exact, efficient)	Top performer on structured traffic data
LightGBM	Sequential (Boosting)	Leaf-wise tree growth	Very High	Very Fast	SHAP	Suited for high-volume traffic analysis
AdaBoost	Sequential (Boosting)	Sample reweighting	High	Moderate	Limited	Simple base learners, interpretable
Stacking	Hierarchical	Model diversity	Very High	Slow	Complex	Effective for multi-category threat detection
Voting Classifier	Parallel	Model diversity	Moderate–High	Variable	Aggregated importance	Reliable and straightforward baseline

1.5. The Role of XAI in Network Security:

1.5.1. Alert Triage in the SOC:

The use case of XAI cannot be considered abstract, as its practical benefit depends on how well it enables human analysts to make better decisions in a timely manner. The clearest way it adds value is via alert triage in a Security Operations Center (SOC). If the model detects any suspicious network activity, then the first task of an analyst is to decide whether to escalate the alert or dismiss it. In the absence of any explanation, this requires the analyst to make his decision purely based on his previous experience and the analysis of traffic data, which can be time-consuming and error prone.

XAI enhances this process by providing feature attribution. For instance, if the model indicates that this alert is triggered by anomalies in the destination port number, high entropy, and unusual frequency, then the analyst can quickly evaluate whether the behavior fits into any known attack patterns. This allows the analyst to verify whether the highlighted features are genuinely anomalous and cross-check the information against threat intelligence related to the host, thereby improving both the speed and accuracy of decision-making. The workflow of XAI-assisted alert triage in a Security Operations Center is illustrated in Figure 1.4. The figure demonstrates how raw network traffic is first processed by an AI detection model, after which an XAI engine generates interpretable explanation, such as SHAP or LIME values to enrich the alert with contextual information before it reaches the SOC analyst for escalation or dismissal decisions.

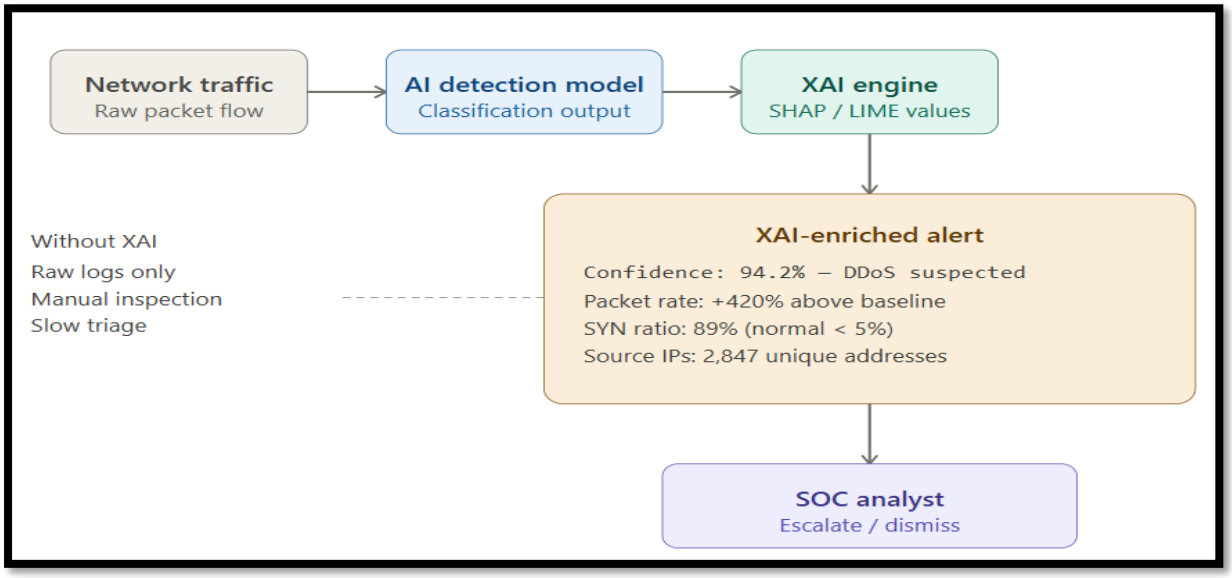


Figure 1.4 *— XAI-enhanced alert generation in a Security Operations Centre*

1.5.2. Model Validation and Bias Detection:

Before deploying a newly trained detection model into a production environment , security engineers must ensure that the system has learned meaningful threat patterns rather than accidental correlations, a phenomenon commonly referred to in ML as the "Clever Hans" effect [12] . Global XAI techniques, such as SHAP summary plots that show global feature importance for the test dataset, allow engineers to determine if the model relies on logically relevant security features in a domain-specific way or on misleading correlations,

which could indicate a source of bias or data leakage and allow engineers to establish justified confidence before operational deployment. More importantly, XAI-driven monitoring can also reveal any changes in the features used by the model in its decision-making process, indicating concept drift, where evolving network behavior or attack strategies reduce the effectiveness of the model. Detecting such shifts early allows engineers to retrain or adjust the system before performance degradation significantly impacts threat detection capabilities.

The role of XAI in validating ML models before deployment is illustrated in Figure 1.5. The figure demonstrates how XAI auditing techniques, such as SHAP global analysis, can identify spurious correlations, concept drift, and data leakage before a model is approved for operational use.

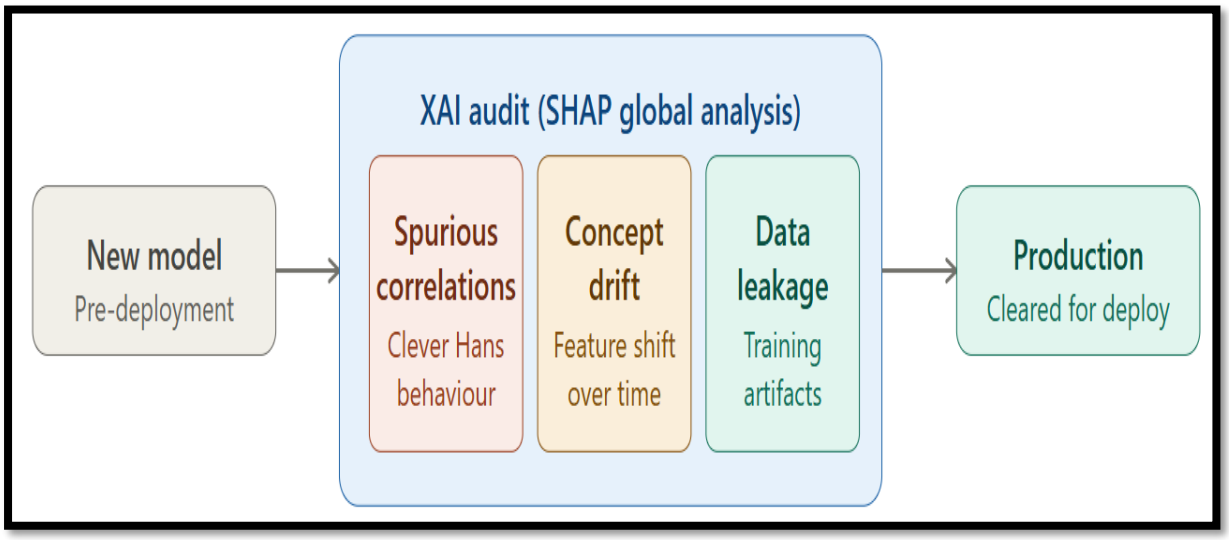


Figure 1.5 *— XAI-driven validation gate before model deployment*

1.5.3. Adversarial Robustness Assessment:

Modeling a system that has hidden decision boundaries makes the weaknesses and vulnerabilities of the system difficult to identify by defenders. Through making the decision boundaries understandable using XAI, security teams can determine what factors are essential to the model and evaluate whether an attacker aware of these factors can craft evasive traffic. Such transparency enables proactive threat modeling, allowing defenders to anticipate possible attack strategies before they are deployed in practice. Consequently, adversarial robustness and explainability are closely interconnected, as improving the interpretability of a model also strengthens the ability to assess and defend against adversarial manipulation. The relationship between explainability and adversarial robustness is illustrated in Figure 1.6. The figure contrasts opaque AI systems, where attackers can exploit hidden weaknesses, with XAI-enabled systems, where feature reliance becomes visible, allowing defenders to proactively model evasion scenarios and strengthen the system before deployment.

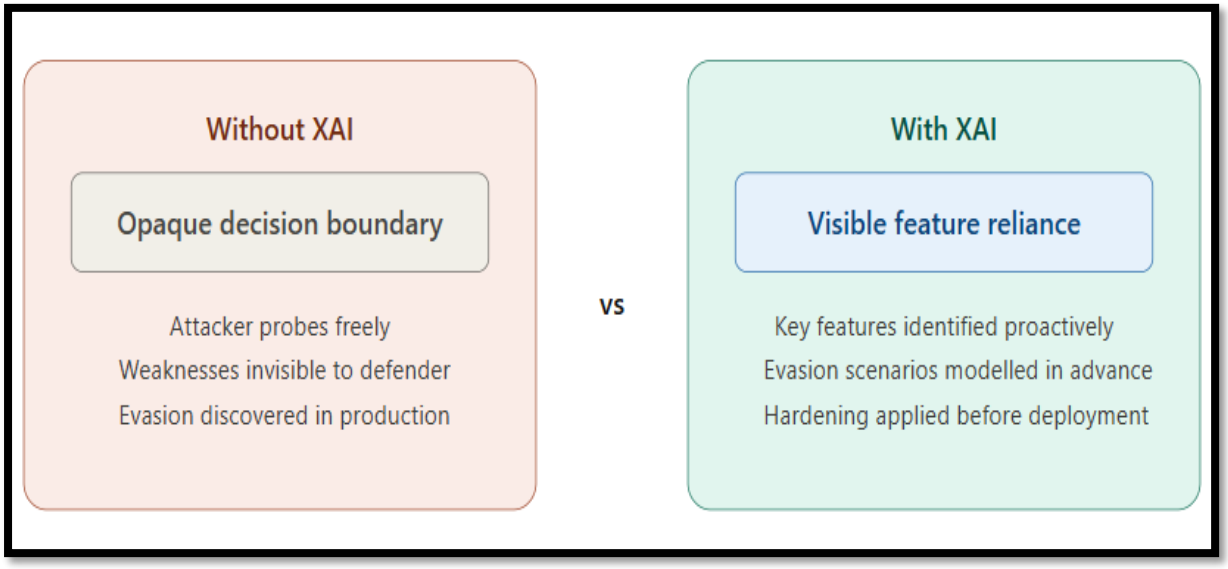


Figure 1.6 *— XAI as a prerequisite for adversarial threat modelling*

1.6. Challenges of Applying AI and Explainable AI in Cybersecurity:

The integration of XAI into real-world cybersecurity operations is much more difficult than when done under the conditions of a controlled laboratory setting. These challenges are not just theoretical restrictions waiting to be addressed but practical limitations that directly impact the effectiveness and reliability of XAI systems in operational settings. It is imperative to understand these restrictions to ensure the proper use of such tools. Factors such as large-scale data volumes, evolving attack strategies, time-sensitive decision-making, and the complexity of modern network infrastructures can significantly limit the applicability of explanation methods. Therefore, understanding these constraints is essential to ensure the appropriate deployment and responsible use of XAI tools in cybersecurity environments.

- **Class Imbalance and Data Quality Issues**

  As far as practical scenarios go, malicious data only accounts for an extremely small fraction of the overall traffic in real-life network environments—less than one percent even in enterprise networks. Therefore, models designed to be run on such data will always have the bias towards maximizing the accuracy of the majority class, providing good benchmark results but not being able to recognize the minority class tasks at all. Although common balancing methods like SMOTE and cost-sensitive learning help overcome these challenges, they do not solve them completely. Datasets like CIC-IDS and NSL-KDD suffer from known labeling problems and use out-of-date attacks; on the other hand, practical captures obtained from the network in operation are often proprietary and cannot be shared. Moreover, these captures may come from the environment different from where the models were tested, thus performing differently when applied to live traffic [13].

- **Robustness of Explanations to Adversaries**

  In addition, there is the problem of how XAI is handled when adversaries know that their target uses specific techniques. New research indicates that the adversary may

construct inputs whose attribution will look legitimate and consistent with benign traffic from the perspective of the analyst, while the behavior of the classifier will remain the same or even be used to the advantage of the adversary [14]. The point is that the explanation can be manipulated separately from the detection layer. It does not mean that the technique fails. However, the analysis must always be conducted in light of domain knowledge, and the explanation cannot replace it.

- **Computational cost and real-time considerations**

  Exact calculation of SHAP scores for one example based on a large model ensemble may involve hundreds of models runs. Due to the need for high throughput in a network security setting, where tens of thousands of flows should be processed per second, it is impractical to perform the necessary calculations to compute SHAP scores in real time [10]. Either one must resort to approximations, such as kernel SHAP or LIME, which reduce the computational burden but introduce uncertainty, or restrict the application of detailed explanations to only the alert instances that will receive human review. Neither solution addresses the criteria adequately.

- **Fidelity and fidelity to the model**

  It should not come as a surprise that for explanations to be helpful, they must accurately reflect what is going on in the computational process within the model. LIME, for instance, creates a linear surrogate model at a local level near a certain input; however, how well the surrogate model reflects what the black-box model does in that neighborhood can never be assured and varies widely depending on the input [16]. The problem here lies in the fact that the explanation might be reasonable and logical, but it can also be incorrect because of its correspondence with what the analyst wants to hear rather than what the model did.

- **The effects of concept drift and explanation stability**

  Performance and explanations both decline as the data generation process deviates from the data distribution seen during training. Explanation stability, or the ability of the same inputs to generate the same explanations, declines as well under distribution shifts, often even before a drop in performance measures can be observed [17]. A ranking of features based on their relative importance for understanding the model's behavior will become obsolete within six months without alerting standard monitoring systems. Thus, tracking explanations over time is just as important as tracking performance accuracy, an issue mostly neglected by current deployment systems.

- **Human factors and trust calibration**

  The first and foremost purpose of an explanation is to improve human decision-making. To do so, it requires not only its technical quality to be sound but also the correct use of the explanation by the analyst. Explanations, according to experimental results from human-computer interaction, tend to produce overtrust. More specifically, analysts receiving any kind of explanation, whether or not it is technically wrong, are likely to make decisions consistent with the model's predictions, compared with analysts without any explanation [18]. Interfaces that exhibit attributions of features with artificial precision without disclosing their uncertainty level might increase analyst trust when there should not be any reason to

have such a high degree of confidence. Building good explanations in terms of uncertainty disclosure and training analysts to deal with them as independent thinkers, are challenges located at the intersection between XAI research and security operation practice.

The table below summarizes these challenges and the mitigation strategies most commonly applied in practice, noting where no adequate solution currently exists.

***Table 1.5**: Summary of Critical Challenges, Risks, and Mitigation Approaches in Explainable AI for Cybersecurity Systems*

Challenge	Core Risk	Primary Mitigation	Current Limitation
Class imbalance	Missed detections on rare attacks	SMOTE, cost-sensitive learning	Benchmark datasets remain unrepresentative
Adversarial explanations	Manipulated attribution scores mislead analysts	Robustness-aware XAI methods	Active research area, no standard defence
Computational latency	Real-time explanation infeasible at scale	Approximate methods, selective explanation	Fidelity loss in approximation
Faithfulness	Explanations rationalise rather than reflect	Faithfulness metrics, cross-method validation	No universal evaluation standard
Concept drift	Explanations become stale and misleading	Explanation distribution monitoring	Few frameworks implement this in practice
Human overtrust	Analyst defers to incorrect explanations	Uncertainty-aware interfaces, critical training	Interface design remains largely unsolved

1.7. Research Gaps and Underlying Motivations

The theoretical and conceptual foundations established in the preceding sections spanning the limitations of rule-based detection, the opacity of high-performing machine learning models, the principles of XAI, and the operational challenges of deploying explainable systems in adversarial environments collectively define a research space that remains incompletely addressed in the existing literature. This section identifies the specific gaps that motivated the design of the experimental framework presented in Chapter Two and positions the contributions of this thesis within the broader landscape of XAI-enhanced intrusion detection research.

Gap 1 — Limited cross-architecture SHAP consistency evaluation. The majority of XAI studies applied to network intrusion detection evaluate a single model architecture and apply a single explainer method, reporting feature importance rankings without examining whether those rankings are robust across structurally different models [10]. This single-model evaluation design cannot answer the question that is most operationally relevant to security practitioners: if I deploy a different model architecture tomorrow, will the feature importance guidance I provide to my analysts remain valid? The absence of systematic cross-architecture

consistency analysis in the literature motivated the three-model comparative design adopted in this thesis and the introduction of Spearman rank correlation as a formal consistency metric.

Gap 2 — Evaluation on small or outdated benchmark datasets. A substantial proportion of XAI-enhanced intrusion detection studies are evaluated on datasets that are now widely acknowledged to be outdated or artificially simplified NSL-KDD, UNSW-NB15, and early CICIDS releases whose attack profiles reflect a threat landscape from five to fifteen years ago [19]. Evaluating explainability frameworks on such datasets risks producing attribution results that reflect dataset-specific artefacts rather than genuine properties of contemporary IoT attack behaviour. The selection of the CIC IoT-DIAD 2024 dataset published in 2024 and capturing 33 attack types across 105 IoT devices was a direct response to this gap, providing a contemporary and operationally realistic testbed for the explainability analysis.

Gap 3 — Insufficient attention to the operational interpretability of SHAP attributions. Many studies that apply SHAP to intrusion detection report feature importance rankings as numerical outputs without systematically evaluating whether those rankings are interpretable in terms of known attack mechanisms [8]. A ranking that places statistically discriminative but operationally meaningless features at the top such as internal network identifiers or dataset-specific artefacts provides no actionable guidance to a SOC analyst. This thesis addresses this gap by explicitly evaluating each consensus feature against its alignment with documented IoT attack signatures, treating operational interpretability as an evaluation criterion alongside statistical measure.

Gap 4 — Underexploration of the precision-recall-explainability trade-off. Existing literature tends to treat performance and explainability as parallel and independent evaluation dimensions, reporting detection metrics and SHAP visualisations separately without examining how the choice of architecture affects both simultaneously [10]. In practice, these dimensions interact: the Bagging model's exact TreeExplainer attributions are inseparable from its tree-based inductive bias, which also shapes its precision-recall profile. The joint evaluation framework adopted in this thesis assessing each architecture on detection metrics, explainer exactness, and cross-model attribution consistency within a unified comparative structure addresses this gap by treating the performance-explainability interaction as a primary object of analysis rather than a secondary consideration.

Gap 5 — Absence of large-scale evaluation under realistic class imbalance conditions. Most XAI studies on intrusion detection are conducted on pre-balanced datasets, obscuring the challenge that class imbalance poses for both detection and explanation [13]. The CIC IoT-DIAD 2024 dataset's original 68:1 class ratio 98.56% malicious versus 1.44% benign provides a realistic and demanding test of the undersampling strategy adopted in Section 2.5.1, and the explicit acknowledgement that SHAP base values are calibrated to balanced rather than operational conditions (Section 3.6.2) address a limitation that most prior studies do not discuss.

These five gaps collectively define the research motivation for this thesis. The study does not claim to resolve all of them the limitations documented in Section 4.1 acknowledge where the contributions are bounded but each design decision in the experimental framework can be

traced directly to one or more of the gaps identified here, providing a principled justification for the methodological choices that distinguish this work from prior literature

1.8. Problematique

Despite significant advances in both network security AI systems and explainable AI techniques, several critical gaps persist at their intersection. These gaps form the foundational motivation for the research presented in subsequent chapters of this thesis.

Gap 1: Limited empirical evaluation of XAI techniques in network security contexts

Gap 2: absence of comparative frameworks for White-Box versus Black-Box XAI in security.

Gap 3: Lack of significant frameworks for deployable XAI in network security

Motivations Underpinning This Thesis research

These three gaps directly motivate the research presented in this thesis. First, the limited empirical evaluation of XAI in network security contexts motivates the design of systematic experiments that apply SHAP explanation technique to representative intrusion detection tasks using realistic network traffic data, with evaluation metrics tailored to security operational requirements rather than generic interpretability measures.

Second, the absence of comparative frameworks for white-box versus black-box XAI motivates a structured comparison of three ML models under identical conditions, assessing them on criteria that matter for security operations. This comparison aims to provide actionable guidance for security practitioners selecting XAI methods for deployment.

Third, the lack of deployable frameworks motivates the development of an reproducible framework that implements selected XAI technique, integrates with standard network security datasets, produces explanations in analyst-consumable formats, and respects real-time operational constraints. This framework is designed not as a laboratory prototype but as a foundation that can be extended and adapted for production security environments.

By addressing these three gaps collectively, this work advances the state of deployable explainability for network security AI systems. Building on this motivation, the following chapters transition from the conceptual foundations established in this chapter toward concrete methodologies, implementations, and empirical evaluations.

1.9. Conclusion:

The history of cybersecurity indicates that rules-based approaches can hardly protect organizations from attacks carried out in highly scalable, complex, and unpredictable environments. The adoption of AI allowed overcoming some of these constraints by providing an opportunity to learn new patterns, detect anomalies, and react to new types of cyber threats. Nevertheless, this enhanced capacity generated a serious problem: the inability to interpret decision-making processes.

As discussed in the previous chapter, the mere effectiveness of cybersecurity solutions may not guarantee their successful application within operational contexts. The use of machine learning algorithms in Security Operations Centers requires quick and trustworthy decision-making. At the same time, there is a need to justify the choice made by the model and explain its reasoning. The opacity of advanced algorithms makes it difficult to assess their outcomes and trust their recommendations.

XAI appears as an indispensable addition instead of being an optional improvement to the overall approach. By providing information regarding the operation of a particular model – either from a global perspective or via individual explanations of decisions made by the algorithm – XAI makes it possible to verify alerts generated, identify possible bias in the results, and retain control of automation processes.

The contrast between black-box and white-box models and various types of explanations shows the constant trade-off between predictive and explanatory capacities. However, the chapter indicates several problems associated with the application of XAI, including computational costs, susceptibility to adversarial attacks, instability amid concept drifts, and the possibility of relying too much on machine learning solutions by human experts.

In this context, the function performed by XAI can be seen as allowing for controlling a complex system and understanding its operation instead of trying to make the system simpler and more comprehensible. In other words, designing an accurate yet explainable system is still a major challenge. In the subsequent chapters, we will continue moving from theory to practice in testing our ideas.

Chapter 2

Research Design and Methodological Framework

2.1. Introduction:

After defined the explainability of AI (XAI), the principles underlying DL models and ensemble methods, as well as identified the problems facing the field of network security, it is necessary to investigate the experimental validation of these problems. In addition to theoretical knowledge of explainability in cybersecurity, it is necessary to consider the practical issues associated with implementation. Which types of models exhibit adequate performance on traffic data? What behavior does explainability show with classifiers based on millions of flows? Are there practical uses for explanations generated for large datasets, or are they merely mathematical exercises?

This chapter will attempt to answer some of these questions. It offers a detailed explanation of a comparative experiment of ML applied to a network intrusions dataset with particular emphasis on both prediction performance and interpretability. It starts with a definition of a comparative ML pipeline aimed to assess three modeling approaches: Deep Neural Network (DNN), Bagging and Stacking ensembles. All three models will be subjected to equal conditions and evaluated by their predictive power and interpretability.

To this end, the following network traffic dataset will be used, which contains over 27 million examples of network flows, described by 36 attributes, including normal traffic and attacks. Its size and distribution pattern make it suitable for testing explainability framework in conditions more similar to real-life environments compared to popular benchmarking datasets. Finally, the whole experiment will be conducted using Python programming language, along with Scikit-learn, TensorFlow and SHAP libraries, in a local Jupyter setup accommodating memory restrictions due to the nature of the data.

Working with big data poses a number of challenges in comparison to benchmark data, such as a significant class imbalance problem, the costliness of exact explainability technique at high speeds, and maintaining explanations' validity despite the variety of different model architectures. Overcoming these obstacles as opposed to simplifying the dataset is a methodological decision. A measure of value is not the performance of a certain framework under perfect conditions, but rather how useful it would be applied in practice.

Organization of this chapter is as follows: Section 2.2 describes the dataset in details: the sources, structure, and rationale behind its choice. Section 2.3 focuses on the analysis of feature space and statistical properties of the dataset. Section 2.4 specifies the task of classification itself and investigates the distributions of attack and normal traffic. Section 2.5 deals with preprocessing of the data which includes addressing missing values, feature scaling, handling of class imbalance, and train-test split. Section 2.6 covers experimental settings and requirements concerning software/hardware. Section 2.7 gives some information regarding the choice of models and rationale behind testing three architectures. Section 2.8 elaborates upon adding SHAP and process of creating and interpreting global/local explanations. Section 2.9 concludes the chapter by summarizing decisions and their implications for discussion in Chapter Three.

2.2. Dataset Collection and Selection Rationale:

This experiment is based on the data set derived from the CIC IoT-DIAD 2024[23], which is a data set from the Canadian Institute for Cybersecurity (CIC). This data set was chosen because of its contemporary nature and its applicability to current and real-world IoT conditions, and also since previous data sets like CICIDS2017 and NSL-KDD are not appropriate for analyzing current attacks.

The dataset consists of network traffic obtained from a vast IoT infrastructure with 105 connected nodes. In this IoT ecosystem, 33 different types of attacks have been performed and classified into seven primary categories, which include DoS, DDoS, reconnaissance, web attacks, brute force attacks, spoofing, and Mirai attacks. A summary of its key characteristics is provided in Table 2.1.

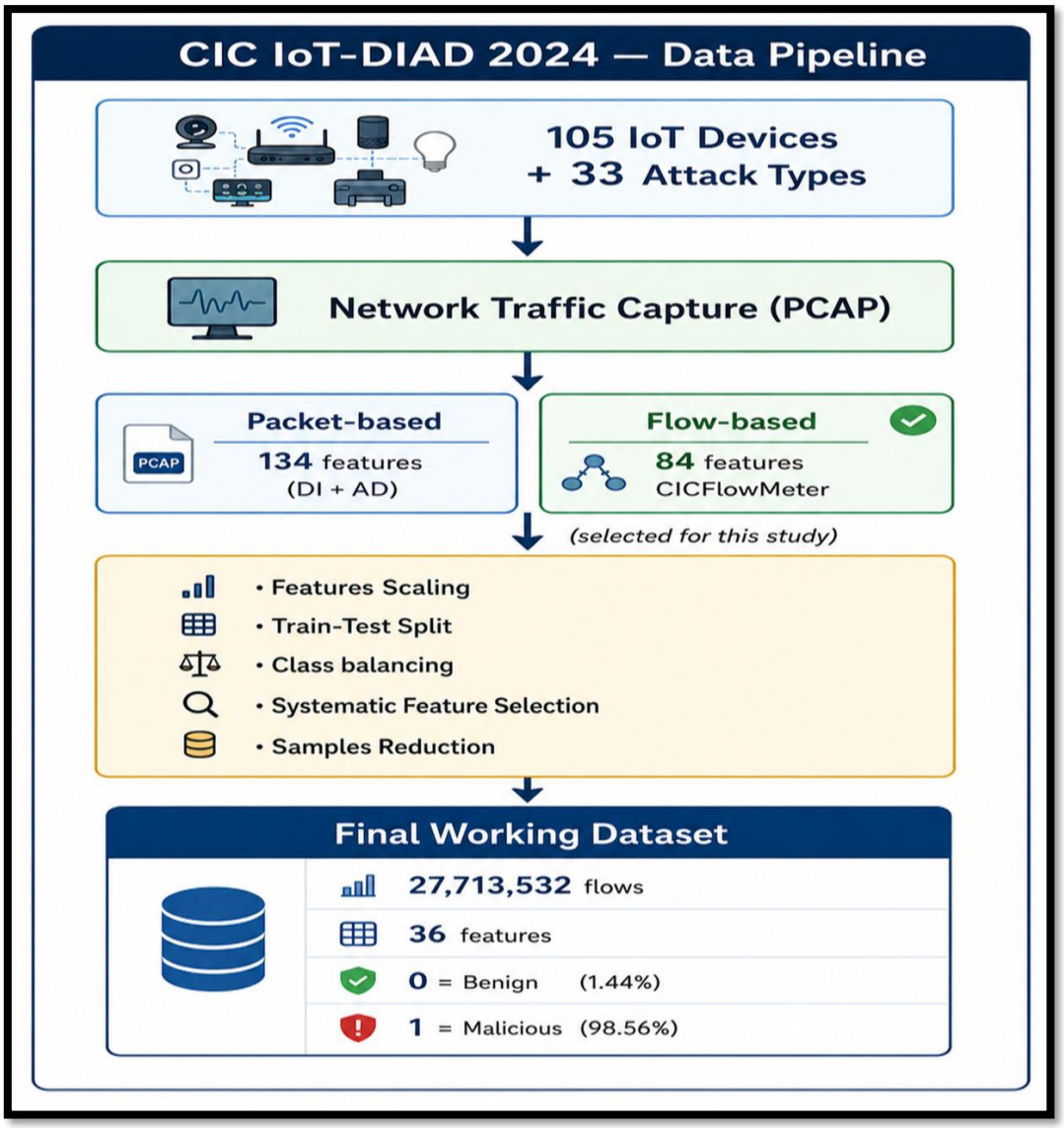


***Figure 2.1**: Data collection and preprocessing pipeline for the CIC IoT-DIAD 2024 dataset*

Table 2.1: Summary characteristics of the CIC IoT-DIAD 2024 dataset as used in this study.

Characteristic	Detail
Source	Canadian Institute for Cybersecurity (CIC), UNB
Release Year	2024
IoT Devices	105
Attack Types	33 across 7 categories
Feature Extraction	Flow-based (CICFlowMeter)
Raw Flow Features	84
Final Working Features	36
Total Records	27,713,532
Benign Records	398,182 (1.44%)
Malicious Records	27,315,350 (98.56%)
Label Encoding	0 = Benign, 1 = Malicious

2.3. Feature Description:

To form the feature space, the data set offers two approaches for extracting information: packet-based and flow-based. Packet-based creates 134 features through device and anomaly identification descriptors; whereas flow-based derives 84 features via statistical descriptors of bidirectional traffic flows. For this research, the flow-based approach was adopted because of its efficiency and suitability for big data processing. Flow representation is derived from bidirectional network flow statistics computed by CICFlowMeter [10]. In comparison with packet representation, flow representation considerably decreases the dimensionality and noise of data and keeps necessary traffic patterns, making it more suitable for real-time intrusion detection systems.

2.4. Attack Categories and Classification Problem Formulation:

2.4.1. Attack Taxonomy in the CIC IoT-DIAD 2024 Dataset:

The dataset CIC IoT-DIAD 2024 contains 33 different kinds of attacks that fall into seven broad categories, which represent specific classes of attack behavior on the IoT system. Knowing about the taxonomy of the attacks is critical for formulating the classification problem and also to interpret the results obtained from the problem formulation in Chapter Three. The seven attack categories and their constituent attack types are summarized in Table 2.2.

Table 2.2: Attack taxonomy of the CIC IoT-DIAD 2024 dataset, organized by functional category with primary attack targets.

Category	Attack Types	Primary Target
DDoS	UDP Flood, TCP Flood, ICMP Flood, HTTP Flood, MQTT Flood	Network bandwidth / service availability
DoS	SYN Flood, UDP Flood, HTTP Flood	Host connection tables / service availability
Reconnaissance	Port Scan, Ping Sweep, OS Fingerprinting, Service Discovery	Network topology / vulnerability mapping
Web-Based	SQL Injection, XSS, Command Injection, HTTP Brute Force	Application layer / authentication
Brute Force	FTP Brute Force, SSH Brute Force, Telnet Brute Force	Credential compromise
Spoofing	ARP Spoofing, DNS Spoofing, IP Spoofing	Network identity / traffic interception
Mirai	Mirai Botnet variants (scan, infection, C&C, attack)	IoT device compromise and botnet recruitment

These types of attacks can vary significantly in terms of their traffic signatures. For volumetric attacks such as DDoS [19] and DoS attacks, there are high rates of high volumes of flows that result in outliers for rate and flag count features. This is because, for these attacks, there are large variances associated with such attacks. In case of reconnaissance attacks, there are low volumes and high rates in which the destination ports were scanned, thus giving rise to outliers for Src Port, Dst Port, and Flow IATs. For Mirai category attacks, there are scans followed by credential misuse, resulting in small flows with unusual values of TCP initialization windows sizes as seen from the results of SHAP who describe in Chapter Three.

2.4.2. Binary Classification Problem Formulation:

Even though there are 33 attack types classified into seven classes, the classification problem in our case is formulated as a binary discrimination problem of separating malicious network flows from benign ones. There are three reasons why we have cast this as a binary rather than a multi-class problem

The first reason is related to the practical aim of the current work, which lies in detection of anomalous flows (i.e., identification of a threat), as opposed to attack categorization, at the flow level. This implies a binary decision making process that is typical for Security Operation Centre analysts: should a flow be reported as suspicious or not.

The second reason relates to the severe class imbalance issue that would arise in case of formulating a multi-class problem among all 33 classes, since certain attacks make up less than 0.01% of the entire dataset. In such cases, evaluation of the classifier's per-class performance becomes meaningless, and the explainability analysis becomes questionable due to the impossibility of learning proper decision boundaries for minority classes.

Finally, the labeling of the classes conforms to the choices made during the preprocessing stage by the dataset producer, since the provided feature vector includes only one binary target variable (0 – benign, 1 – malicious) encompassing all 33 kinds of attacks into a malicious class.

As shown in Figure 2.2, the above-mentioned dataset suffers from a significant skew towards one particular class, which is valid in terms of operational reality in an IoT scenario where most of the network traffic is legitimate. A classifier that always predicts the majority class without further analysis would be correct 98.56% of the time, yet would not detect any malicious activities at all. This is the reason why the proposed approach towards class balancing in Section 2.5 was implemented, and also highlights the limitations of accuracy as a metric of model performance discussed in Section 3.3.

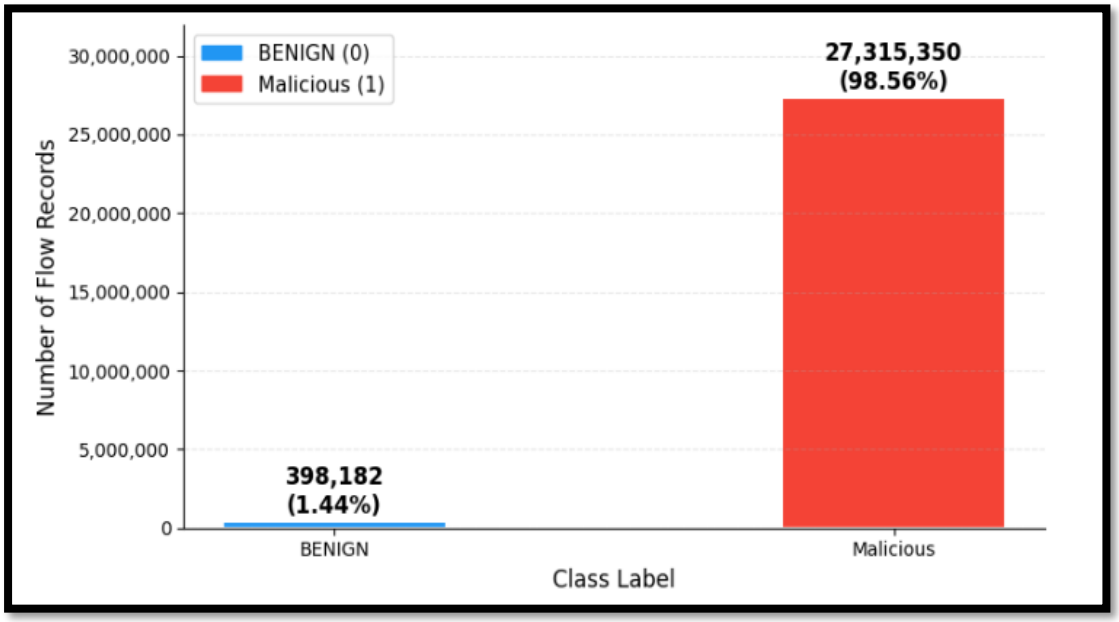


Figure 2.2: *Binary class distribution of the CIC IoT-DIAD 2024 dataset prior to balancing (n = 27,713,532).*

2.5. Dataset Inspection and Preprocessing:

The preprocessing pipeline consists of three main steps: (1) samples reduction (cleaning and deduplication), (2) systematic feature selection, and (3) class balancing.

2.5.1. Samples Reduction:

The original flow-based dataset consists of 27,809,999 records or samples. The preprocessing pipeline involves the following steps: collecting all raw files, labeling them according to their respective patterns, merging the labeled data into a single CSV file, and

removing duplicate entries as well as any rows containing NaN or INF values. After this cleaning process, the final preprocessed dataset contains 27,713,532 samples.

2.5.2. Systematic Feature Selection:

The original flow-based dataset also initially contains 84 features. However, not all are suitable for model training, necessitating a systematic feature selection procedure. The process begins by removing categorical features that carry identifying information, namely Flow ID, Source IP, Destination IP, and Timestamp — four features in total. Such identifiers do not contribute to generalizable patterns and could lead to overfitting in ML models.
Next, low-variance numerical features are eliminated using a variance threshold of 0.01. This step removes an additional 10 features that offer little to no discriminative value for classification tasks. After excluding these 14 features (4 categorical + 10 low-variance), 69 features remain from the original 84 (the label column is treated separately).

To further refine the feature set, LightGBM (Light Gradient Boosting Machine) is applied, leveraging cumulative importance based on gain. The model is trained using the Gradient Boosting Decision Tree (GBDT) algorithm with early stopping and a validation set to prevent overfitting. By retaining only those features that account for 90% of the cumulative importance, the feature set is reduced from 69 to 36.

After completing these preprocessing stages — removing categorical identifiers, filtering out low-variance features, and applying LightGBM-based cumulative importance selection — the final dataset consists of 36 relevant features. This final refined dataset contains 27,713,532 flow samples, and 36 features ready for ML purposes.
The selected 36 features include six categories of functionally coherent key performance indicators (KPIs), while each category captures a technical behaviour dimension of the network flow that plays an important role in supporting anomaly detection capabilities for IoT.
Packet length statistics (eleven features): describes the distribution of the size of packets that travel between the client (forward direction) and server (backward direction): (Fwd Packet Length Max, Fwd Packet Length Min, Fwd Packet Length Mean, Fwd Packet Length Std, Bwd Packet Length Max, Bwd Packet Length Min, Bwd Packet Length Mean, Packet Length Max, Packet Length Min, Packet Length Mean, Average Packet Size). Malicous flows, especially volumetric attacks, such as DDoS and DoS are unusual due to their distribution of the individual packet length features: sFlooding attacks have so many short packets that maximizes its efficiency in making requests [20]. Amplification attacks have few large payload packets that fill the downstream bandwidth.

flow volume metrics measure the total amount of data transmitted over a single connection. This type includes fields such as Total Length of Fwd Packet, Total Length of Bwd Packet and Subflow Bwd Bytes. Directional asymmetries in volume are especially valuable: reconnaissance and brute-force attacks usually generate a high outbound volume from the server with only a low volume response, leading to a forward-backward imbalance which makes them distinct compared to legitimate bidirectional IoT communication [22].

Traffic rate and timing indicators characterise the speed and inter-arrival process of network flows. Flow Bytes/s, Flow Packets/s, Fwd Packets/s and Bwd Packets/s are related to data transmission rates, while Flow IAT Min, Flow IAT Max, Fwd IAT Total,Fwd IAT Min, Fwd IAT Max and Fwd IAT Std depict the temporal distribution of inter-arrival time in forward direction. The deviations in these metrics among the first and most reliable indicators of unusual behaviour in IoT environments where the communication tends to have highly regular low-rates [21]. Bwd Bulk Rate Avg and Bwd Header Length provide further insight into protocol overhead in the backward direction as well as bulk transfer behaviour that differ significantly with fragmentation-based evasion techniques.

TCP flag counts: SYN Flag Count, RST Flag Count, PSH Flag Count and ACK Flag Count as explicit protocol-level behavioural signatures. In this study, SYN Flag Count was the most discriminative feature among these due to the frequent occurrence of SYN flood attacks in the threat landscape of IoT, where TCP three-way-handshakes are exploited to deplete connection tables on servers [22]. High RST counts also point to atypical connection teardown patterns often found in scanning and evasion activity.

Highlighted here are a few TCP initialisation parameters, i.e. FWD Init Win Bytes and FWD Seg Size Min.The former records the very first TCP receive window size and the latter indicates minimum segment size advertised by node initiating the connection. Thus, these features are discriminative for connection-level anomalies such as Mirai-style botnet traffic because malicious actors often do not negotiate default window parameters.
Finally, the metrics needed to characterize the temporal rhythm of flow activity, that is Active Mean, Idle Max and Idle Std. Botnet and Mirai traffic have unique active-idle patterns –short, repeated high-volume bursts of traffic separated by programmatic idle periods– that are statistically different from the smooth continuous flows exhibited when legitimate IoT devices are in use [22].

The identification-adjacent nature of port identifiers Src Port and Dst Port encourages removal, however in the context of IoT, these features would add meaningful discriminative signal: service ports targeted by attack traffic are always specific those used in legitimate device communication. Attack flows (scanning / brute-force) produce distinct usage distribution when compared to legitimate device communication [21]. Table 2.3 provides a structured overview of all 36 features, organized by functional category.

***Table 2.3**: The 36 retained features organized by functional category, with operational interpretation relevant to IoT anomaly detection*

#	Feature	Category	Operational Interpretation
1	Fwd Packet Length Max	Packet Length	Largest packet sent by client
2	Fwd Packet Length Min	Packet Length	Smallest packet sent by client
3	Fwd Packet Length Mean	Packet Length	Average client packet size

4	Fwd Packet Length Std	Packet Length	Variability in client packet size
5	Bwd Packet Length Max	Packet Length	Largest packet sent by server
6	Bwd Packet Length Min	Packet Length	Smallest packet sent by server
7	Bwd Packet Length Mean	Packet Length	Average server packet size
8	Packet Length Max	Packet Length	Largest packet in the flow
9	Packet Length Min	Packet Length	Smallest packet in the flow
10	Packet Length Mean	Packet Length	Mean packet size across flow
11	Average Packet Size	Packet Length	Overall mean packet size
12	Total Length of Fwd Packet	Flow Volume	Total bytes sent by client
13	Total Length of Bwd Packet	Flow Volume	Total bytes sent by server
14	Subflow Bwd Bytes	Flow Volume	Backward bytes in subflow
15	Flow Bytes/s	Traffic Rate	Overall byte transmission rate
16	Flow Packets/s	Traffic Rate	Overall packet transmission rate
17	Fwd Packets/s	Traffic Rate	Forward packet rate
18	Bwd Packets/s	Traffic Rate	Backward packet rate
19	Bwd Bulk Rate Avg	Traffic Rate	Avg backward bulk transfer rate
20	Bwd Header Length	Traffic Rate	Cumulative backward header size
21	Flow IAT Min	Timing	Min inter-arrival time in flow
22	Flow IAT Max	Timing	Max inter-arrival time in flow
23	Fwd IAT Total	Timing	Total forward inter-arrival time
24	Fwd IAT Min	Timing	Min forward inter-arrival time

25	Fwd IAT Max	Timing	Max forward inter-arrival time
26	Fwd IAT Std	Timing	Std deviation of forward IAT
27	SYN Flag Count	TCP Flags	Number of SYN packets — flood indicator
28	RST Flag Count	TCP Flags	Abnormal connection resets
29	PSH Flag Count	TCP Flags	Immediate data push requests
30	ACK Flag Count	TCP Flags	Acknowledgement packet count
31	FWD Init Win Bytes	TCP Init	Initial TCP window size (client)
32	Fwd Seg Size Min	TCP Init	Minimum forward segment size
33	Active Mean	Activity	Mean active flow duration
34	Idle Max	Activity	Maximum idle period duration
35	Idle Std	Activity	Variability in idle periods
36	Src Port / Dst Port	Port Info	Source and destination port numbers

The resulting feature matrix from the process explained in Section 2.2, cannot simply be fed into a ML model as it stands. There are three characteristics of the dataset that have been identified to need special treatment prior to training the models: the large imbalances present in terms of distribution between benign and malicious data, the large discrepancies in scale between features within the six functionality types and the splitting of the data into two disjoint training and test sets with appropriate class ratios. The reasons behind each step in this section are explained to show the effects of preprocessing on both model outcomes and SHAP interpretability

2.5.3. Class balancing

The most prominent feature of the dataset is the highly imbalanced distribution of classes. The percentage of malicious traffic amounts to 98.56%, whereas the percentage of benign traffic comes down to 1.44%. Such an imbalance of classes corresponds to real-life attack-dominated situations but creates difficulties during model training. For instance, conventional evaluation metrics, such as accuracy, might be biased, as a classifier could attain a high level of accuracy by merely classifying instances into the predominant class. Thus, a model would have difficulties detecting the minority class [18].

It is evident from the presented dataset that there exists a considerable skew in its distribution: 27,315,350 malicious flows (98.56%) and only 398,182 benign flows (1.44%), implying that a class ratio in the data is roughly equal to 68:1. If ignored, this will inevitably create an undesirable incentive for classifiers to maximize the accuracy when dealing with the majority class, meaning that a model predicting every single sample as malicious can still achieve almost 98.6% precision without detecting any benign flows. Naturally, in the case of intrusion detection, this will result in severe operational problems in terms of availability.

A range of potential methods for tackling the Issue was considered. For instance, oversampling approaches like SMOTE, synthesize additional examples by interpolating between samples from the minority class, however, given a 68:1 ratio, the amount of data artificially produced in order to compensate for the skew will be considerably greater than the number of actual samples in the minority class, leading to interpolation errors in synthetic samples. As for cost-sensitive learning, it was ruled out due to making model comparisons more complicated in terms of SHAP attributions.

The methodology used In this thesis used a random under sampling method on the majority class, whereby a stratified random sample of 398,182 malicious flows was selected without replacement from the total population of malicious flows, resulting in a balanced dataset of 796,364 flows with a precise 50:50 ratio of classes. The advantage of this methodology is that it ensures the preservation of the statistical validity of both classes – all samples in the balanced dataset are actual flow observations, ensuring the legitimacy of the SHAP attribution calculations in the explanation phase [20]. Table 2.4 summarize the dataset composition before and after under sampling together with the train-test split portions. Figure 2.3 illustrates the class distribution before and after balancing.

However, the disadvantage of using the methodology is that it involves the omission of about 96.9% of the total malicious flows. This problem is addressed through the use of performance metrics such as accuracy, precision, recall, and F1-score, and confusion matrix interpretation in Section 3.3.

Table 2.4: *Dataset composition at each stage of the preprocessing pipeline, showing the effect of majority-class undersampling on class distribution.*

Stage	Benign	Malicious	Total	Ratio
Raw dataset	398,182	27,315,350	27,713,532	1 : 68.6
After undersampling	398,182	398,182	796,364	1 : 1
Training set (80%)	318,546	318,545	637,091	1 : 1
Test set (20%)	79,637	79,636	159,273	1 : 1

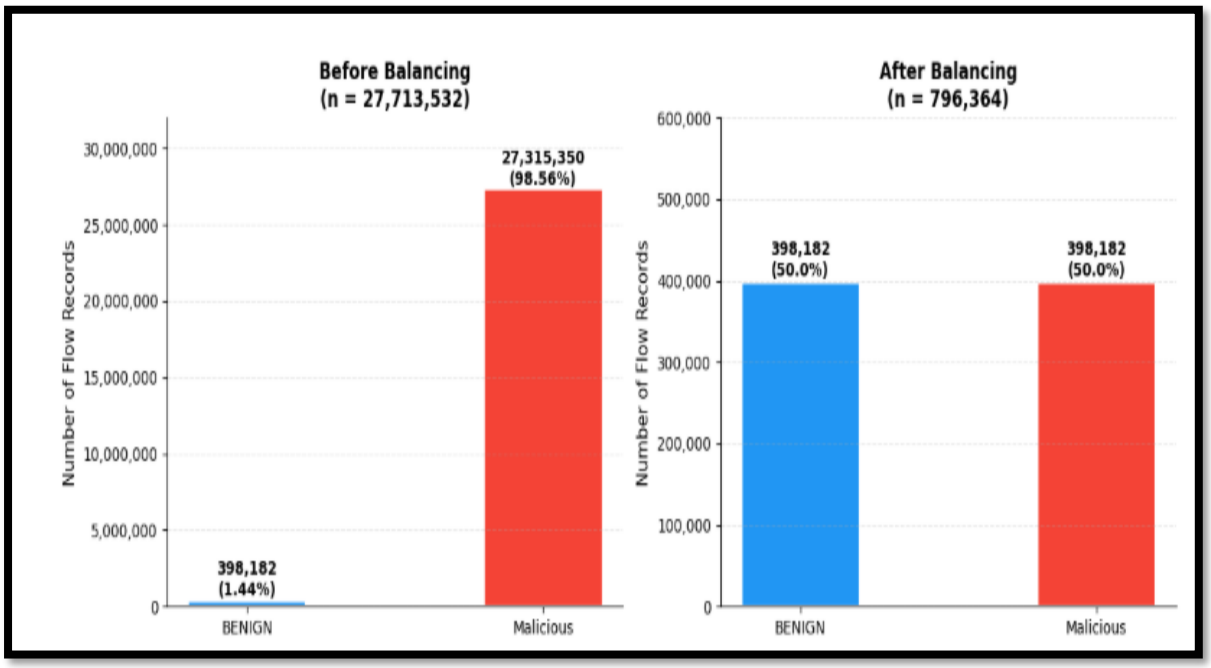


Figure 2.3 *illustrates the class distribution before and after balancing*

2.5.4. Features Scaling:

As Table 2.4 shown the descriptive statistics for the 36 retained features and reveals large scale differences between them. Timing features such as Flow IAT Min and Fwd IAT Total have values in the millions of microseconds. Flag count features such as SYN Flag Count and RST Flag Count have values close to zero. Rate based features such as Flow Bytes/s have means above 700,000, and initialization parameters such as FWD Init Win Bytes are in the low thousands. Without normalization these differences would bias the learning process systematically for all the three model architectures evaluated in this study, although via different mechanisms.

For the DNN architecture, the gradient updates during backpropagation are dominated by unscaled features with large magnitudes. This means that the optimizer focuses on dimensions with large raw values, regardless of their actual discriminative relevance. This results in slow convergence and, in extreme cases, numerical instability during the early epochs of training. Feature scaling has no impact on the split-point selection for decision tree base learners used in the Bagging and Stacking ensemble architectures (trees are invariant to monotonic transformations of individual features). It does however affect the LogisticRegression meta-learner used in the Stacking architecture, which is sensitive to distance and requires normalised inputs to reliably estimate coefficients. The full 36-feature matrix was scaled using StandardScaler, which scales each feature to zero mean and unit variance based on

$$\mathbf{1.}\ \frac{x-\mu}{\sigma} = z$$

where μ and σ are the mean and standard deviation calculated only on the training set. An important implementation detail is that the scaler is fit only on the training data, and the

learned parameters are applied to the test set. This prevents any information from the test distribution from influencing the pre-processing, a form of data leakage that would produce optimistically biased performance estimates [20]

Hence, a short argument for the choice of StandardScaler over other normalization approaches is that StandardScaler maintains the relative distributional shape of each feature while removing scale discrepancies, rendering it the optimal selection for achieving both predictive performance and SHAP interpretability. This is can be observed in Figure 2.4 through three representative features.

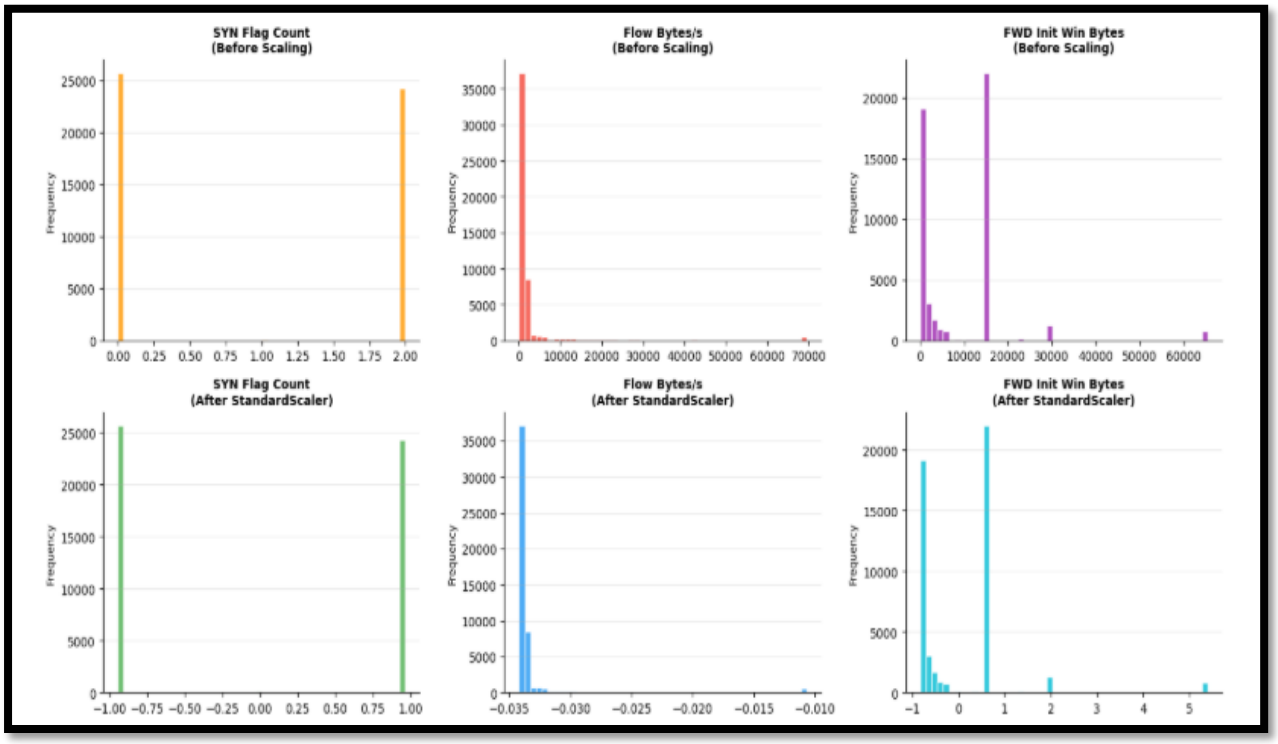


Figure 2.4 *illustrates the effect of StandardScaler on three representative features*

2.5.5. Train-Test Split

After class balancing and feature scaling were completed, the resulting working dataset of 796,364 flow records was divided into a training and a testing set with an 80/20 proportion, resulting in 637,091 training examples and 159,273 test examples. Stratified sampling was used to make sure that the same exact 50:50 class proportion was maintained in both sets.

The test set was completely excluded from all steps of the model building process, which includes hyperparameter tuning, threshold optimization, and scaling, being evaluated on it at the very end of the experiments described in Chapter Three. This is essential to guarantee that the performance measures that will be presented in Section 3.3 indeed measure the true generalization power of our models [20].

2.6. Experimental Environment, Tools, and Settings:

2.6.1. Computational Infrastructure:

All experiments performed for the current study have been executed using the Kaggle cloud computing framework, ensuring a standard and reproducible runtime environment without having to configure any local hardware. The reason behind choosing the cloud computing solution is linked to the size of the data: with a total of 27,713,532 flow data entries and a balanced subset containing 796,364 observations across 36 features, the amount of computational power needed cannot be handled on a regular personal computer. Kaggle notebooks offered enough capacity to conduct all computation in one single run. Table 2.5 is a summary of all the hardware resources that were available throughout any experimental run.

Table 2.5: *Hardware specifications of the Kaggle computational environment used across all experimental runs.*

Component	Specification
GPU	NVIDIA Tesla T4 × 2 (16 GB VRAM each)
CPU	Dual-core Intel Xeon @ 2.0 GHz
RAM	30 GB system memory
Storage	20 GB Kaggle persistent disk
Runtime	Python 3.10 / JupyterLab / kernel
Platform	Kaggle Notebooks (cloud-based)

2.6.2. Software Environment and Library Stack:

All the experiments were performed using the programming language Python 3.10 because of its rich set of libraries in the areas of scientific computation and ML, compatibility with the Kaggle execution environment, and its popularity in the community of researchers specializing in computer network security studies. Table 2.6 shows all software tools that were used during each phase of the experiments, from data preparation to model training and explainability analysis.

Table 2.6: *Software libraries and versions used across all experimental stages*

Library	Version	Role in this Study
pandas	2.3.3	Dataset loading, manipulation, and feature selection
NumPy	2.0.2	Numerical computation and array operations

scikit-learn	1.6.1	Bagging, Stacking, StandardScaler, train/test split, evaluation metrics
TensorFlow / Keras	2.19.0	DNN architecture definition, training, and inference
SHAP	0.50.0	Global and local feature attribution for all three models
Matplotlib	3.10.0	Figure generation for distribution plots and SHAP visualisations
Seaborn	0.13.2	Confusion matrix and statistical visualisation

2.7. Model Selection Strategy:

The selection of ML architectures for this study was not made arbitrarily. Three models were chosen a Deep Neural Network (DNN), a Bagging ensemble based on Random Forest, and a Stacking ensemble combining multiple heterogeneous base learners, because together they span a meaningful range of the complexity-interpretability spectrum that is central to the research objectives stated in Chapter One. This section explains the rationale behind each selection decision, the architectural properties that motivated it, and the way in which each model contributes to the comparative explainability analysis conducted in Chapter Three.
A key constraint governing model selection was compatibility with SHAP-based post-hoc explanation. Not all ML architectures are equally amenable to SHAP analysis: tree-based models support exact TreeExplainer computations, while neural networks require the approximate DeepExplainer or GradientExplainer. The three architectures selected in this study were chosen in part because they allow SHAP to be applied through well-validated explainer classes, enabling a principled comparison of attribution results across models whose internal reasoning mechanisms differ fundamentally [10]

2.7.1. Deep Neural Network (DNN):

The DNN was also considered the most black-box end of that 'complexity' spectrum. The neural networks are one of the strongest approximators for tabular based classification tasks, as they can learn non-linear feature interactions across multiple levels of abstraction without doing explicit feature engineering. Deep architectures have consistently shown a competitive performance on flow-based datasets in the literature of network intrusion detection; especially when the feature space holds correlated variables whose joint behaviour carries superior information than each one of the variables itself [2].

The architecture used in this study is made up of six hidden layers which are fully connected with four of them having 256-128-64 and 32 neurons respectively, followed by a single sigmoid output neuron for binary classification. In this work, we sequentially reduce the width of each layer creating a funnel architecture guiding the network to learn progressively more compressed representations of the input as information flows forward, which has been proven beneficial for generalization on structured tabular data [11]. To reduce Overfitting on

the training partition, Batch Normalisation layers are added following the first two dense layers to normalize activations across mini-batches during training (which makes convergence considerably faster), and Dropout regularization with rates of 0.3 and 0.2 was performed afterward. The model was trained using the Adam optimizer along with the binary cross-entropy loss function, which are two commonly used settings for binary classification problems in the deep learning field. The training process involved a maximum number of 20 epochs, where each epoch included 2,048 batches. In order to avoid overfitting, an EarlyStopping callback was used on the validation loss after 3 epochs. The training dataset was split into 10 percent for validation. Table 2.7 summarizes the DNN architecture and training configuration.

Table 2.7: *DNN architecture and training configuration*

Component	Specification
Input Layer	36 features
Hidden Layer 1	Dense(256) + BatchNorm + Dropout(0.3)
Hidden Layer 2	Dense(128) + BatchNorm + Dropout(0.3)
Hidden Layer 3	Dense(64) + Dropout(0.2)
Hidden Layer 4	Dense(32)
Output Layer	Dense(1, activation='sigmoid')
Total Parameters	67,585 trainable
Optimiser	Adam (default learning rate = 0.001)
Loss Function	Binary Cross-Entropy
Batch Size	2,048
Max Epochs	20
Early Stopping	patience = 3, monitor = val_loss
Validation Split	10% of training set
SHAP Explainer	DeepExplainer

One of the main reasons for including the deep neural network into this experiment is not its performance because ensemble models typically demonstrate similar or even better results on tabular data; however, the problem of interpretation plays an important role here. Deep

learning models do not allow direct inspection of internal representations that involve millions of interactions of weights with each other. This is why it is the hardest test case to apply the methodology of SHAP values, and their consistency with explanations provided by TreeExplainer indicates that detected features are relevant to the data structure [10]

2.7.2. Bagging Ensemble (Random Forest):

Bagging algorithm was chosen as a representative of ensemble models, being significantly more explainable than a neural network and at the same time competitive in terms of accuracy. Bagging, or Bootstrap Aggregating, generates an ensemble of base models trained on independent bootstrap samples of the training set. In case of bagging applied to decision trees, predictions are obtained based on the voting process between all the generated models, whereby variance is minimized without increasing bias compared to any single model.
The particular implementation used in this project was the RandomForestClassifier model from the scikit-learn library, which uses the basic concept of Bagging but includes an additional level of randomness, selecting only a random of features out of a total of 36 at each split point, instead of considering all the available features. This extra randomness provides even more decorrelation between the trees in the forest and ensures that no single noise feature in the data would negatively impact the overall performance of the model, which is especially important for network traffic data where some of the features are correlated by definition [3].
RandomForestClassifier was instantiated with 100 trees (estimators), a maximum tree depth of 20, and minimum samples per leaf of 5. The class_weight='balanced' option was specified to apply inverse-frequency weights to the classes while building the decision trees. As a final tuning parameter, the n_jobs=-1 parameter was passed to enable multithreading and speed up the training time significantly in the two-core Kaggle environment. Table 2.8 summarizes the Bagging configuration.

***Table 2.8**: Bagging (Random Forest) configuration.*

Component	Specification
Algorithm	Random Forest (Bagging + feature randomisation)
Base Learner	Decision Tree (CART)
Number of Trees	100 estimators
Max Depth	20
Min Samples Split	10
Min Samples Leaf	5
Max Features	sqrt(36) ≈ 6 features per split
Class Weight	Balanced (inverse frequency)

Parallelisation	n_jobs = −1 (all available cores)
Random Seed	42
SHAP Explainer	TreeExplainer (exact computation)

One of the main strengths of using the Random Forest in our research is that this model allows us to calculate exact SHAP TreeExplainer values right from the box. In contrast to DeepExplainer, which uses an approximate method to calculate SHAP values due to a linearisation of the deep learning model, the Shapley values calculated via the TreeExplainer are precise because this explainer leverages the tree-based architecture of the model in order to derive exact Shapley values, which are guaranteed to adhere to the two fundamental properties of Shapley values: consistency and local accuracy [10]. Thus, when comparing models in Chapter Three, the Bagging model's results using the TreeExplainer are considered to be the most credible.

2.7.3. Stacking Ensemble:

As the most complex architectural model, Stacking was chosen for inclusion in the comparative analysis. Contrary to Bagging that uses homogeneous base models trained on bootstrap samples of the same data, Stacking ensembles predictions from base models that have structurally heterogeneous architecture using meta-learning, thus allowing an ensemble to benefit from complementary inductive biases provided by different algorithms [3].

The architecture of Stacking used in the experiment includes two levels. At the first level, there are three base models used, including Random Forest that uses 50 estimators and max_depth parameter equal to 15, Decision Tree with max_depth parameter also set to 15, and a Gradient Boosting classifier with parameters estimators = 50 and max_depth = 5. This combination of three models was purposefully chosen since it would introduce necessary variety and complementarity into the structure of input features used by meta-model: Random Forest would give reduced variance ensemble predictions, Decision Tree would offer high variance predictions of a single tree while Gradient Boosting model would produce sequentially corrected predictions with high sensitivity to boundary cases [3]. Thus, the use of bagged trees, a single tree, and boosted trees together ensured the true diversity of input data for the meta-model

The combined prediction output of the ensemble will be a linear combination of the predictions made by the three base learners. The reason for choosing Logistic Regression as the meta-learner lies in the simplicity of the method, which helps the Stacking ensemble’s decision boundary to be dominated by the information coming from the base learners and not by the meta-learner’s complexity. In addition, the linear structure of logistic regression makes the weights used by the meta-learner easy to interpret and provides insight into the process of how Stacking resolves the conflict of the different predictions coming from the base learners [8]. Table 2.9 summarizes the Stacking ensemble model configuration.

***Table 2.9**: Stacking ensemble configuration.*

Component	Specification
Architecture	Two-level Stacking ensemble
Base Learner 1	Random Forest — 50 estimators, max depth 15
Base Learner 2	Decision Tree — max depth 15
Base Learner 3	Gradient Boosting — 50 estimators, max depth 5
Meta-Learner	Logistic Regression (max_iter = 1,000)
Meta-Feature Generation	5-fold cross-validation (out-of-fold protocol)
Parallelisation	n_jobs = −1 (all available cores)
Random Seed	42
SHAP Explainer	PermutationExplainer

The SHAP explainer applied to the Stacking model is PermutationExplainer rather than TreeExplainer, because the multi-level architecture with its Logistic Regression meta-model wrapping heterogeneous base learners does not expose a unified tree structure amenable to exact TreeExplainer computation. PermutationExplainer estimates SHAP values by measuring the change in model output induced by permuting each feature across a reference dataset, providing model-agnostic attributions at the cost of approximation and higher computational overhead [10]. The implications of this explainer choice for the comparability of SHAP results across the three models are discussed in Section 3.4**.**

2.8. Integration of Explainability into the Proposed Framework:

The explainability component of the suggested framework architecture is not a post-hoc addition, but rather a requirement which has influenced the choice of model architecture, feature pre-processing, and the evaluation approach right from the start. The main tool used for generating explanations in all three models was SHAP, the acronym for SHapley Additive exPlanation, a game theoretic approach to assigning contribution values to individual features on the basis of their influence on a prediction, adhering to axioms of fairness which cannot be upheld by any simpler method [10].

In this section, we outline the process of using SHAP within the experimental pipeline for each model, discuss global vs. local modes of explanation and the role of each one, and describe our decision-making process with regard to the methodology in order to achieve uniformity of attributions between models based on fundamentally different reasoning methods. The actual SHAP results, including rankings, beeswarm plots, and force plots, will be described in Sections 3.4 and 3.5.

2.8.1. SHAP Framework Overview:

SHAP SHAP is based on cooperative game theory in which the "game" corresponds to making the prediction, while the "players" are the input variables. Specifically, for each prediction f^(x), the SHAP value ϕi for variable i corresponds to the average marginal contribution of the ith variable across all coalitions of variables, which is precisely the value established by Shapley (1953) to be the only value satisfying four properties of efficiency, symmetry, dummy, and additivity [10].

Formally, the SHAP value for feature ī is defined as:

$$\phi i = \sum S \subseteq F \frac{|s\text{-}|!(|F|-|S|-1)!}{|F|!}[f\hat{}(S \cup \{i\}) - f\hat{}(S)] \quad \textbf{2.}$$

Where F̄ is the full set of features, S̄ is a subset of features excluding ī, and f^(S) denotes the model output when only the features in S̄ are observed (with the remaining features marginalised over their expected distribution). The sum aggregates over all possible orderings in which feature ī could be added to the coalition, weighting each marginal contribution by the number of orderings that produce that coalition.

There are four unique qualities that make SHAP attributions uniquely suited for the task of explaining predictions in the security analytics domain under investigation. First, local accuracy means that the total sum of all SHAP values for any prediction will be equal to the difference between that prediction and the average prediction value across the dataset thereby ensuring internal consistency of the explanation. Second, missingness means that any feature not present in the input will have an assigned value of zero. Third, consistency means that if there is any change to the model such that the importance of a certain feature will increase, the corresponding SHAP value cannot decrease. Fourth, efficiency means that the attribution of total importance is divided evenly across all features without any leftover.

Since exact calculation of Shapley values is an exponentially hard problem, where the computation for a given prediction would involve evaluation of all 2^{36} possible combinations of features, this is impractical. However, there exist three types of SHAP explainers that can take advantage of unique qualities of each type of models (Section 2.8.2) in order to perform the computation in a feasible manner.

2.8.2. Explainer Selection per Model:

The choice of SHAP explainer class that would provide the right attribution for the selected models is not purely aesthetic; rather, it affects what kind of mathematical guarantees can be provided for the attributions and how much trust we can put into those attributions as reliable models of the model behavior. Three different classes of explainers were used here, one for each model architecture.

TreeExplainer for Bagging (Random Forest)

TreeExplainer calculates exact SHAP values in the case of tree models based on their hierarchical structure that allows feature attributions to be enumerated without needing to explore all 2^{36} possible combinations [10]. When used for Random Forest, with an ensemble of T trees, TreeExplainer calculates the exact Shapley values independently for each tree, averaging the output to yield exact attribution that is guaranteed to comply with all four SHAP axioms, which cannot be guaranteed when any approximation method is utilized. The running time of exact computation for a Random Forest scales as O (TLD^2), where L denotes the number of leaves and D the maximum tree depth. In this study, exact computation of SHAP values with TreeExplainer was conducted on a sample of 500 test instances chosen randomly out of the total 159,273 test instances, yielding a (500x36x2) SHAP value matrix from which class-1 (Malicious) attributions were selected.

DeepExplainer for DNN

DeepExplainer is an extension of the Deep LIFT method that approximates SHAP values for deep neural networks by backpropagating attribution signals via a modified chain rule [10]. Contrary to TreeExplainer, DeepExplainer calculates SHAP value approximations rather than Shapley values due to a linear approximation of the nonlinear activation functions in the neural network that fulfils the criterion of local accuracy but not consistency when certain activation function configurations are used in the network. A background reference set of 100 samples from the training data was provided to DeepExplainer in order to represent the expected distribution of features that serves as a basis for marginal contributions measurement. For 200 test instances, SHAP values were calculated yielding a (200×36×1) tensor, which was then reshaped to (200×36).

PermutationExplainer for Stacking

PermutationExplainer is a model-agnostic explainer that estimates SHAP values by measuring the change in model output induced by permuting each feature across a reference dataset of background instances [10]. Because it treats the model as a black box making no assumptions about its internal structure, it can be applied to any model, including the multi-level Stacking architecture whose Logistic Regression meta-model wraps heterogeneous base learners in a way that precludes TreeExplainer. The approximation quality of PermutationExplainer improves with the number of permutation iterations; in this study, 200 test instances were evaluated, producing a (200×36) SHAP value array. The computational cost of PermutationExplainer is substantially higher than TreeExplainer per instance, the Stacking SHAP computation required approximately 60 seconds for 200 instances compared to seconds for the equivalent TreeExplainer computation reflecting the overhead of repeated full-model evaluation across permuted feature sets.

The use of different sample sizes across the three explainers reflects a deliberate trade-off between statistical representativeness and computational feasibility. TreeExplainer's efficiency permits a larger sample (500 instances), providing more stable global attribution estimates. DeepExplainer and PermutationExplainer's higher per-instance costs necessitate

smaller samples (200 instances). All three sample sizes are sufficient for the global summary statistics, mean absolute SHAP values reported in Section 3.4, where the attribution rankings have been shown to stabilize well before the sample sizes used here [10].

2.8.3. Global and Local Explanation Modes:

SHAP attributions can be aggregated and visualized at two complementary levels of granularity global and local each serving a distinct operational purpose in the security analytics context of this study.

Global explanations characterize the overall behaviour of a model across the full evaluation sample, answering the question: which features does this model rely upon most, on average, when distinguishing malicious from benign IoT flows? Global feature importance is computed by taking the mean absolute SHAP value for each feature across all evaluated instances:

$$\mathbf{3.}\quad \bar{\phi} i = \frac{1}{N}\sum_{n=1}^{N} |\phi i^{(n)}|$$

where N is the number of evaluated instances and $\phi i^{(n)}$ is the SHAP value assigned to feature i for instance n. This quantity is visualized as a horizontal bar chart the Summary Bar Plot in which features are ranked from most to least influential. In the security context, the global explanation answers the operational question that a SOC manager would ask before deploying a model: *what aspects of network traffic is this classifier actually using to make its decisions, and are those aspects aligned with known attack signatures?

Another type of visualization at the global level is the Beeswarm plot, which builds on the Summary Bar plot and displays not just the mean absolute value of the attribute's contribution but also the direction and distribution of the attributions for all evaluated instances. The beeswarm is a scatter plot with each point standing for an individual instance; the horizontal position corresponds to the SHAP value (values greater than zero will have an effect of pushing the prediction toward Malicious while negative values towards Benign), while the color denotes the value of the feature itself (red denotes high values while blue – low). In other words, the plot highlights whether the effect of the feature on predictions is consistently directional or not, which becomes important when looking at attributes like SYN Flag Count.

A local explanation describes the rationale behind a particular prediction, asking why did the classifier predict this particular flow as malicious? Local attribution is depicted using a Force Plot, where the importance of each feature is illustrated as forces that influence the prediction and push it towards the label predicted by the classifier or the opposite label (benign), starting from the base value the average prediction of the whole test set. In doing so, the visualization clearly identifies which features influenced the deviation of the prediction from the base value and how much of an influence each had, thus producing the level of justification required for a particular instance.

When applying a model in the field of cybersecurity, the need for local explanations becomes obvious. They can provide valuable insight into the rationale behind the alert and help the analyst verify that it is consistent with the domain knowledge about a suspected attack. It is

clear, for example, whether the number of SYN flags sent is high when the attack is suspected based on some local explanation, as well as whether the values of other contributing features support it too. Figure 2.5 illustrates the conceptual relationship between global and local SHAP explanations and their respective roles in the proposed framework

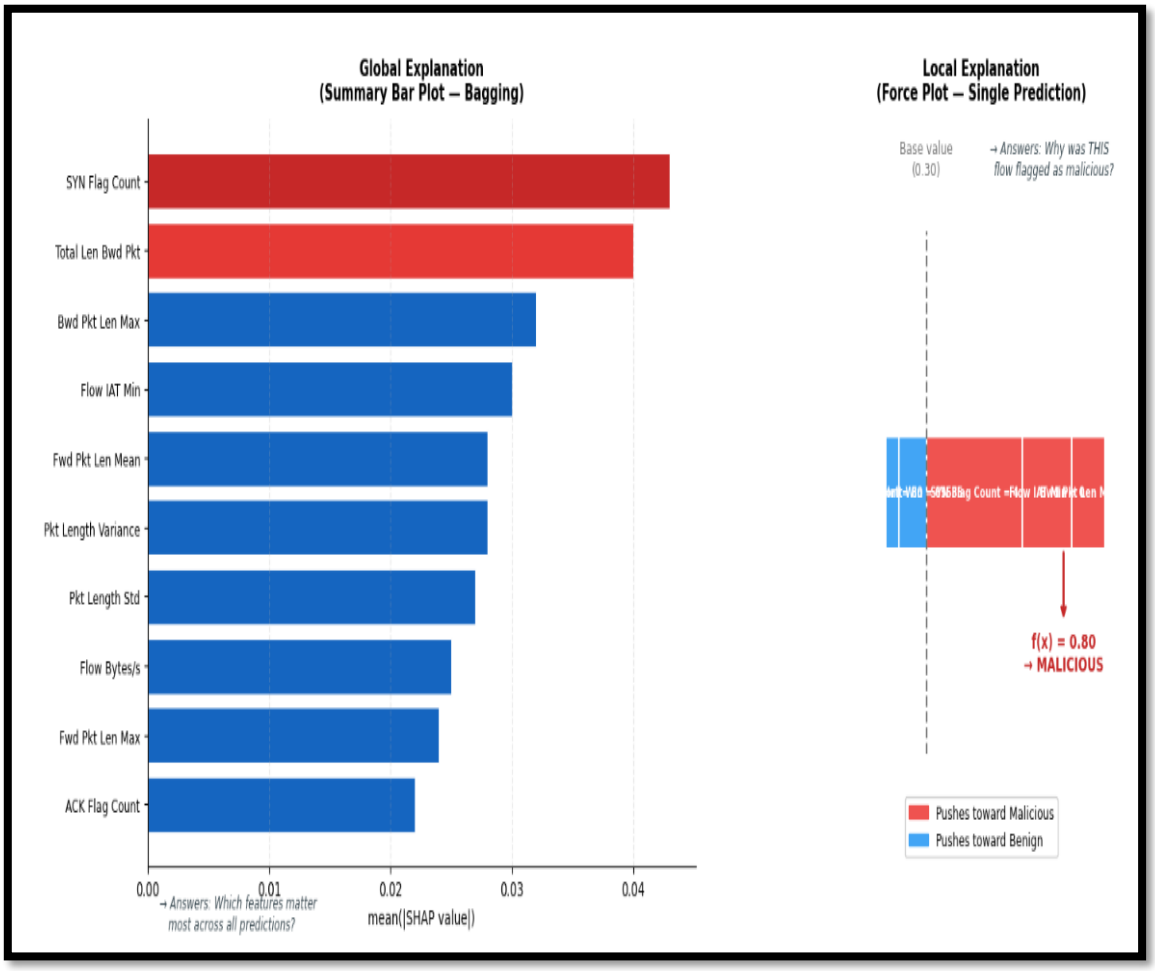


Figure 2.5 *the conceptual relationship between global and local SHAP explanations and their respective roles in the proposed framework*

2.8.4. Cross-Model Consistency Analysis:

A direct numerical comparison of SHAP values across the three models is not methodologically valid, since TreeExplainer, DeepExplainer, and PermutationExplainer operate under different computational assumptions and marginalisation strategies, meaning that raw magnitude values cannot be placed on a common scale [10].

The approach adopted in this study is therefore rank-based: for each model, features are ordered by their mean absolute SHAP value and the degree of agreement between the three resulting ranked lists is quantified using Spearman's rank correlation coefficient:

$$3. \quad \rho = 1 - \frac{6\sum_{i=1}^{n} di^2}{n(n^2-1)}$$

where d (i) is the difference in rank assigned to feature i between two models and n=36. A value of ρ approaching 1.0 indicates strong cross-model agreement; a value near zero indicates that the models rely on structurally different aspects of the feature space.

This analysis accomplishes two objectives. First, from an analytical standpoint, attributes whose rankings are consistent across models with varied inductive biases can be seen as stronger indicators of discriminative information compared to those ranking well in just one model. Second, from a practical perspective, consistent ranking of attributes across models ensures that the explanations derived from SHAP analysis are more actionable for SOC analysts because the attribute that is ranked high regardless of the model used deserves consideration.

2.9. Conclusion:

This chapter has presented the complete methodological framework underpinning the experimental component of this thesis, covering all decisions from dataset selection through to the integration of explainability into the proposed pipeline.

The CIC IoT-DIAD 2024 dataset was selected on the basis of its contemporary relevance to real-world IoT threat landscapes, providing 27,713,532 flow records across 36 retained features spanning six functional categories. The statistical characterization of the feature space in Section 2.3 established two properties pronounced right-skew and moderate intra-group correlation that directly motivated the preprocessing decisions documented in Section 2.5: majority-class undersampling to a balanced working set of 796,364 instances, StandardScaler normalization to remove scale discrepancies without distorting distributional shape, and stratified holdout partitioning with a fixed random seed to ensure full reproducibility.

The classification problem was formalized as binary detection benign versus malicious, a formulation justified by both the operational objectives of the study and the severity of class imbalance at the sub-category level. Three model architectures were selected to span a meaningful range of the complexity-interpretability spectrum: a DNN, a Bagging ensemble based on Random Forest, and a two-level Stacking ensemble. Each architecture was paired with the most appropriate SHAP explainer class: DeepExplainer, TreeExplainer, and PermutationExplainer respectively and the limitations of cross-explainer comparability were addressed through a rank-based consistency methodology.

Taken together, the decisions documented in this chapter establish the conditions under which the experimental results reported in Chapter Three can be interpreted with confidence. Every preprocessing step, architectural choice, and explainability configuration has been justified with reference to both the statistical properties of the data and the operational objectives of the research, ensuring that the findings presented in the following chapter reflect genuine properties of the models and the dataset rather than artefacts of the experimental design.

Chapter 3

Experimental Results And Discussion

3.1. Introduction:

The preceding chapter established the methodological conditions under which the experimental results reported here were obtained: a balanced working dataset of 796,364 flow records drawn from the dataset corpus, a 36-feature input space spanning six functional categories, three model architectures spanning the complexity-interpretability spectrum, and a unified SHAP-based explainability framework applied through architecture-appropriate explainer classes. This chapter presents the outcomes of that framework and interprets them against the research objectives stated in Section 1.5.

The chapter is organized around four analytical questions that the experimental design was constructed to answer. The first concerns raw predictive performance: do the three architectures achieve detection rates sufficient for operational deployment in an IoT security context, and how do they compare against one another across standard evaluation metrics? The second concerns explainability quality: do the SHAP attributions produced for each model identify features that are consistent with established knowledge of IoT attack signatures, and are those attributions stable and interpretable in both global and local modes? The third concerns cross-model consistency: do structurally different architectures — a neural network, a bagged ensemble, and a stacked ensemble — converge on the same feature importance rankings, and what does convergence or divergence imply about the reliability of the explanations? The fourth concerns the operational trade-off: given the performance and explainability results together, which architecture best serves the dual objectives of accurate detection and actionable explanation in a real-world IoT security operations context?

Section 3.2 documents the training process for each model, including convergence behaviour and hyperparameter sensitivity. Section 3.3 presents the evaluation metrics — accuracy, precision, recall, F1-score, and confusion matrices — for all three models on the held-out test set. Section 3.4 presents and interprets the SHAP attribution results at both global and local levels for each architecture. Section 3.5 conducts the cross-model consistency analysis using rank correlation, identifying the features that are robust across architectures and those that are model-specific. Section 3.6 synthesizes the findings across all four analytical questions and discusses their implications for the deployment of XAI-enhanced intrusion detection in IoT environments. Section 3.7 concludes the chapter and positions the findings relative to the broader research context established in Chapter One.

3.2. Model Training and Hyperparameter Optimization:

Model training for all three architectures was conducted on the 637,091-instance training partition under the conditions documented in Section 2.6. This section describes the training dynamics observed for each model, the hyperparameter choices made and their justification, and the convergence behaviour that provides the first evidence of model stability prior to formal evaluation on the held-out test set. Where training produced diagnostically informative outputs — loss curves, epoch-level metrics, or timing data — these are reported here to establish that the models reached stable solutions rather than terminating prematurely or exhibiting signs of overfitting.

3.2.1. DNN Training Dynamics:

DNN was trained for 20 epochs using a batch size of 2048, while training in the Kaggle environment accelerated by the use of GPUs, and this training took approximately 4-5 seconds per epoch following the initial compilation period for the first epoch. There was no early stopping during this entire training process, which suggests that there were no improvements or improvements to the validation loss. Table 3.1 presents the epoch-level training metrics for selected epochs, illustrating the convergence trajectory from initialisation to final stable performance.

Table 3.1: DNN epoch-level training metrics at selected epochs, illustrating convergence from initialisation to final stable performance.

Class	Precision	Recall	F1-Score	Support
BENIGN (0)	95.00%	99.48%	97.19%	79,637
Malicious (1)	99.46%	94.99%	97.17%	79,636
Macro Average	97.23%	97.24%	97.18%	159,273
Weighted Average	97.23%	97.24%	97.18%	159,273

There are several notable results presented in Table 3.1. Firstly, there is an extremely small difference in the values of training accuracy (96.95%, Epoch 20) and validation accuracy (97.37%). In fact, the model performs better on the validation partition than the training dataset. Therefore, it is safe to conclude that no overfitting occurred in this model. The reason for this unusual performance can be explained by the presence of two Dropout regularization layers (with rates of 0.3 and 0.2), along with Batch Normalization. Both these approaches add a certain level of random noise to the training process, while it is not present during the inference phase [11].

Secondly, one can note that the value of validation loss steadily decreased across all 20 epochs of training (from 0.1310 to 0.0928). It means that the EarlyStopping criterion was not met even by the 20th epoch and there were still some improvements in model performance. However, taking into account how low validation loss has become (the decrease between Epochs 16 and 20 is quite small – 0.0968 → 0.0928), further training may bring some minimal gains.

Third, the difference between the validation precision and validation recall values (99.38% and 95.35%) during epoch 20 shows that this model tends to be careful when classifying bad packets. It makes very few false alarms but misses some bad ones. The gap between the precision and recall results is mainly due to the default threshold value of 0.5 in the sigmoid classifier. Lowering this threshold makes the model catch more bad packets (better recall) but also causes more false alarms (lower precision).

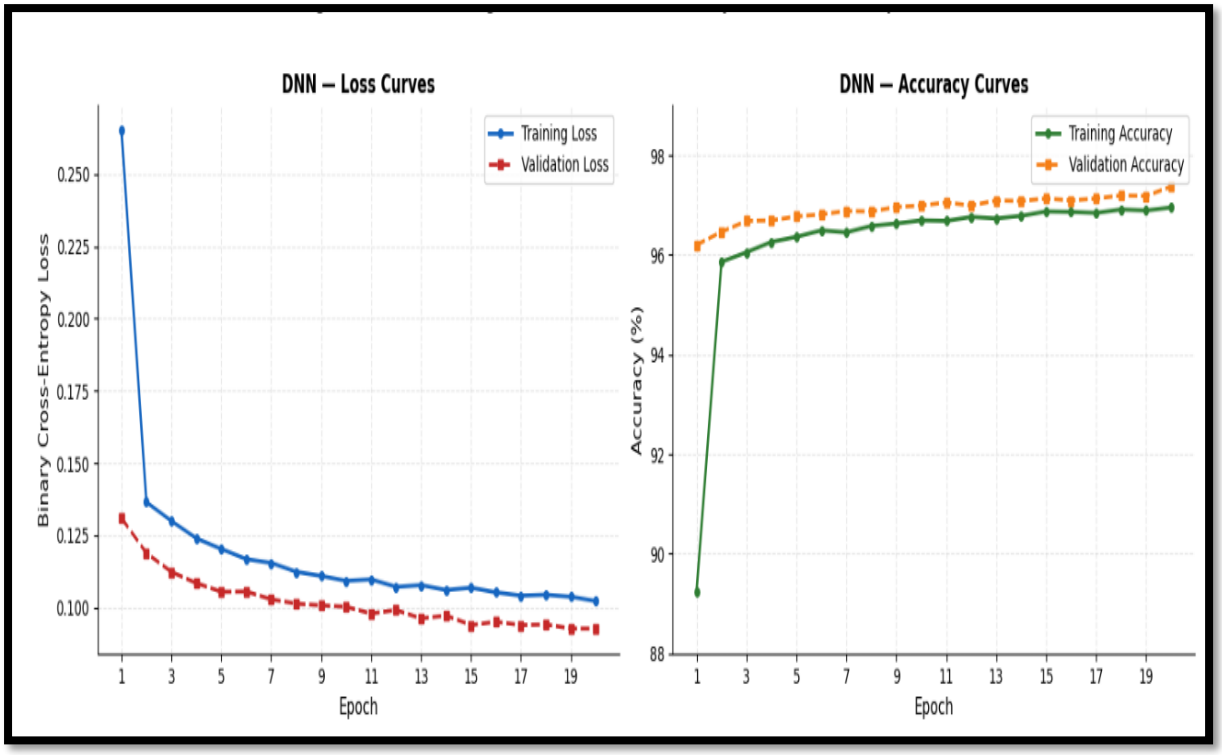


***Figure 3.1**: DNN Training and Validation Loss/Accuracy*

3.2.2. Bagging Training Dynamics:

Differently from the DNN model, the training of the Random Forest is not an iterative process, and it simply generates 100 decision trees in one training round on the input data without any iterations or loss curve. Training took 118.7 seconds on the two cores Kaggle machine, yielding a complete Random Forest with tree depth limited to a maximum of 20 nodes and a minimum number of five samples per leaf.

Feature importance based on Gini index of the Random Forest is visualized in Fig. 3.2, which is automatically produced by the Random Forest model training process as a consequence of building trees that include for each feature the weighted value of the reduction in impurity at a certain split point. Several insights can be drawn from the ranking that is highly useful for further SHAP analysis.

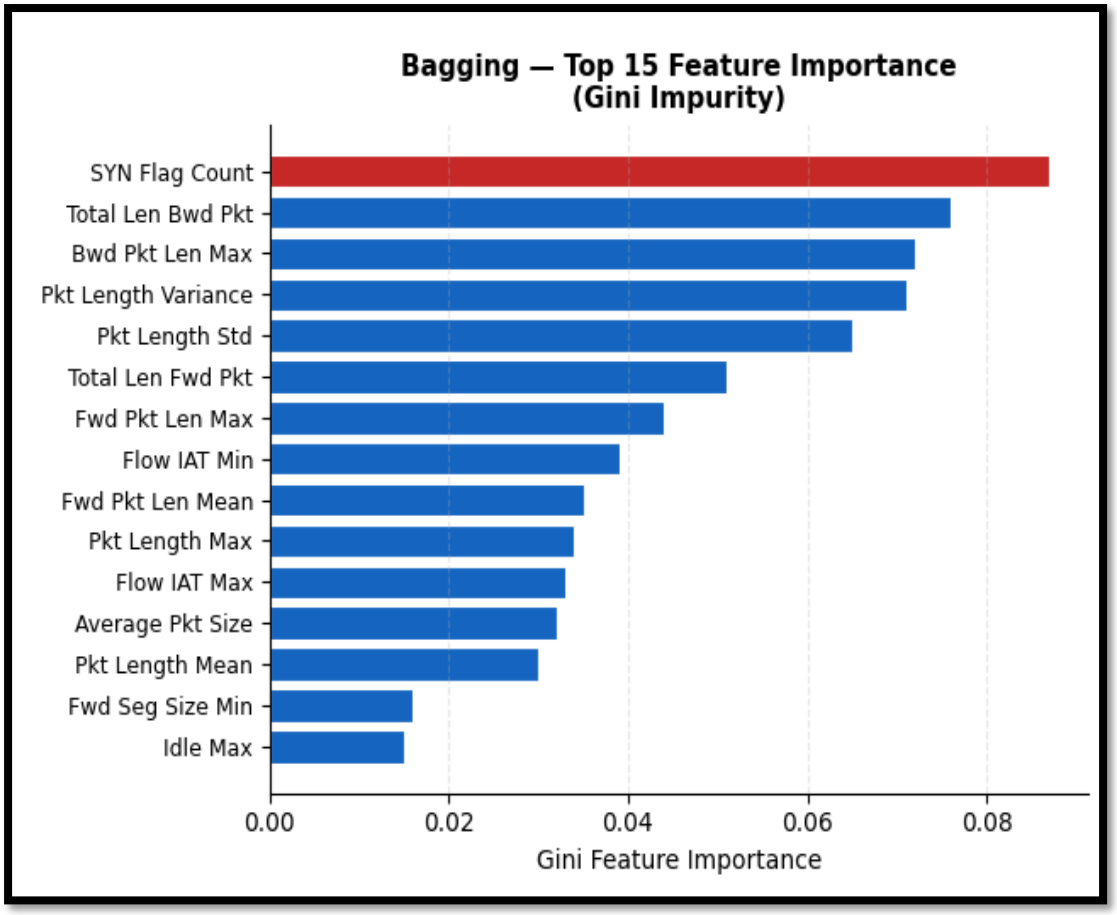


***Figure 3.2**: Bagging (Random Forest) "Top 15 Feature Importance (Gini Impurity)*

The most significant attribute is definitely SYN Flag Count (Gini Importance ≈ 0.087), as predicted in Section 2.4.3 based on the assumption that attributes discriminative of multiple types of attacks, specifically those associated with both the SYN flood attack stage and the scanning phase of the Mirai botnet, will take top positions in global importance. The attributes Total Length of Bwd Packet (0.076) and Bwd Packet Length Max (0.073) follow right after it, accounting for the imbalance in packets size characteristic of volumetric attack traffic in comparison to normal IoT communication. The packet length and variance attributes (Packet Length Variance, Packet Length Std) come in fourth and fifth places accordingly, representing the irregularity of payload sizes in attacks compared to constant packet sizes in normal IoT communication.

It should be noted that the ranking provided above according to Gini importance will be directly compared with the ranking generated using SHAP TreeExplainer in Section 3.4. If both ranking mechanisms, relying on fundamentally different attribution approaches, identify the same top five attributes, this would mean that these attributes contain actual discriminative signal and not some model-specific artifacts.

3.2.3. Stacking Training Dynamics:

Among the three architectural frameworks, Stacking demanded the largest amount of time spent on the training phase — 1,401.3 seconds (~23 minutes) — because of the inherent complexity of computing out-of-fold meta-features through a five-fold cross-validation procedure across three base learners prior to the actual training of the meta-model. Such

requirements are intrinsic to the nature of Stacking algorithm and cannot be avoided by any particular hyperparameter tuning; this issue is further discussed in Section 3.6.

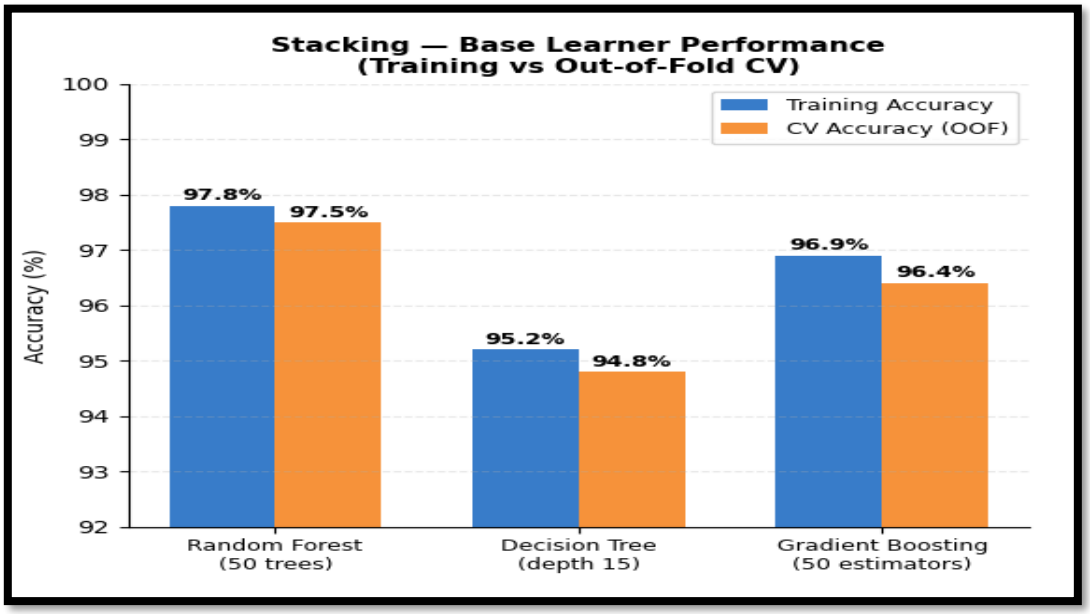


***Figure 3.3**: Stacking Feature Importance Learner Performance*

The plot presented in Fig. 3.3 shows training and out-of-fold accuracy values obtained for each of the three base learners. In terms of performance metrics, the best results were obtained by the Random Forest base learner, both in training (accuracy = 97.8%) and in out-of-fold evaluation (accuracy = 97.5%), with a slight margin confirming that 50-estimator configuration is not overfitted to training partition. The Gradient Boosting base learner showed comparable metrics (training: 96.9%; OOF: 96.4%), while the Decision Tree base learner demonstrated inferior results (training: 95.2%; OOF: 94.8%), as anticipated from a highly variable model of depth 15.

This difference in base learners' performances was intended for the purpose of allowing the meta-learner in form of logistic regression classifier to learn an optimal weight allocation to more accurate models while using weaker models in order to extract additional useful information. The overall Stacking ensemble accuracy of 98.12%, which was obtained by averaging accuracies across folds [3], is significantly higher than that of the most accurate individual base learner (97.5%).

3.3. Model Validation and Performance Evaluation:

Evaluation of the models used the same test data set of 159,273 flows (79,637 benign and 79,636 malicious) which was not included at any stage of model development. For each model, four different metrics are provided: accuracy, precision, recall, and F1-score. The four metrics are orthogonal and taken together constitute the comprehensive characterization of classification performance on a balanced dataset.

Accuracy reflects the fraction of correctly classified samples in each class. Precision reflects the fraction of the samples that are actually malicious out of the total number of samples which have been labeled by the model as malicious — this metric is sensitive to false positives. Recall reflects the fraction of the samples which are actually malicious among all

the malicious samples which the model has identified — this metric is sensitive to false negatives. F1-score represents the harmonic mean of precision and recall, i.e., is a combined value of both metrics. In terms of IDS task, recall is more crucial for security operations than precision since missing real attack (false negative) is much more damaging than receiving a false alarm (false positive).

Confusion matrices for each model are given as well, showing a complete understanding of the distribution of errors on both classes, and not just an overview of the total number of mistakes made, but also their nature – whether they consist of over classification of benign flows as malicious (false positives) or under classification of the latter as benign (false negatives).

3.3.1. DNN Performance Results:

The DNN achieved an overall accuracy of **97.24%** on the held-out test set, with precision of **99.46%**, recall of **94.99%**, and an F1-score of **97.17%**. Table 3.2 presents the full classification report including per-class metrics.

***Table 3.2**: DNN classification report on the held-out test set (n = 159,273).*

Class	Precision	Recall	F1-Score	Support
BENIGN (0)	95.00%	99.48%	97.19%	79,637
Malicious (1)	99.46%	94.99%	97.17%	79,636
Macro Average	97.23%	97.24%	97.18%	159,273
Weighted Average	97.23%	97.24%	97.18%	159,273

The per-class breakdown reveals a clear asymmetry in error distribution. The model attains near-perfect recall on the benign class (99.48%), correctly identifying the vast majority of legitimate IoT flows, while achieving substantially lower recall on the malicious class (94.99%) — meaning that approximately 1 in 20 malicious flows is misclassified as benign. Conversely, malicious precision reaches 99.46%, confirming that when the model does issue a malicious alert, it is almost certainly correct. This precision-recall asymmetry reflects the conservative classification behaviour induced by the sigmoid threshold of 0.5: the DNN requires strong evidence before committing to a positive malicious classification. Figure 3.4 presents the confusion matrix, which quantifies the error distribution in absolute terms.

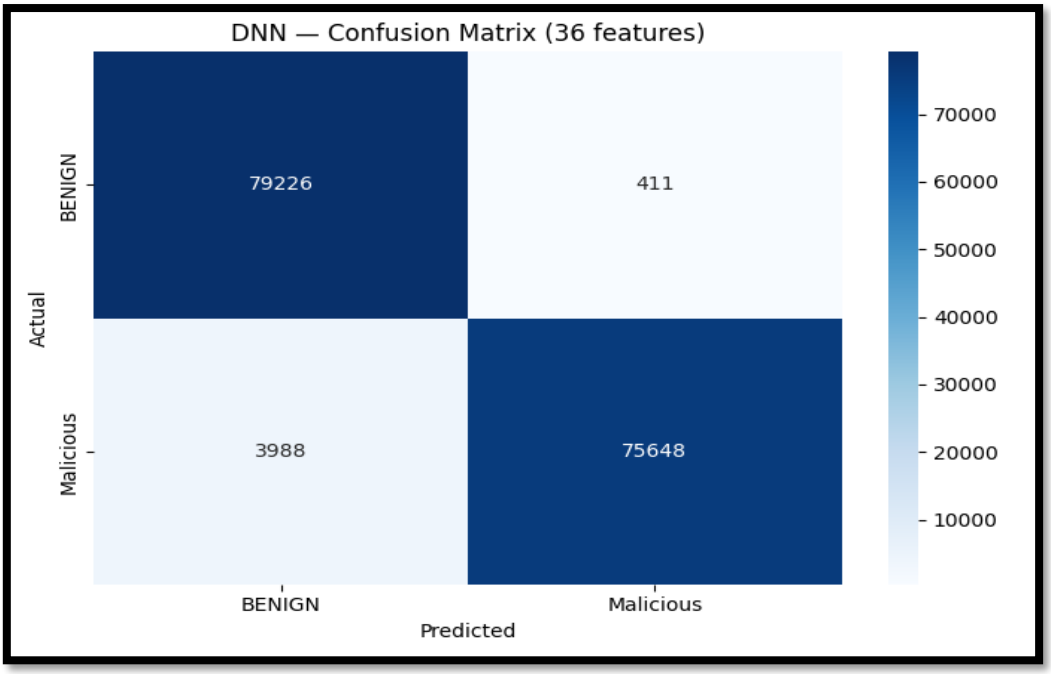


***Figure 3.4**: DNN Confusion Matrix on the held-out test set.*

The confusion matrix confirms that the DNN produces **411** false positives — benign flows incorrectly flagged as malicious — and **3,988** false negatives — malicious flows that escaped detection entirely. In an operational SOC context, the false negative count carries greater weight: 3,988 undetected malicious flows out of 79,636 total malicious instances represents a miss rate of **5.01%** on the malicious class. While competitive against published benchmark results, this miss rate would translate to a non-trivial volume of undetected attacks at the throughput scales of real IoT network environments. The extremely low false positive rate (411 out of 79,637 benign flows = **0.52%**), however, confirms that DNN-generated alerts are highly reliable — the precision-recall trade-off here favours alert quality over detection completeness.

3.3.2. Bagging Performance Results:

The Bagging ensemble achieved an overall accuracy of 98.11% on the held-out test set, with precision of 99.76%, recall of 96.45%, and an F1-score of 98.09%. Table 3.3 presents the full classification report.

***Table 3.3:** Bagging (Random Forest) classification report on the held-out test set (n = 159,273).*

Class	Precision	Recall	F1-Score	Support
BENIGN (0)	96.55%	99.76%	98.13%	79,637
Malicious (1)	99.76%	96.45%	98.07%	79,636
Macro Average	98.15%	98.10%	98.10%	159,273
Weighted Average	98.15%	98.11%	98.10%	159,273

The error structure observed for the Bagging ensemble is considerably dissimilar from that observed for the DNN model. The reason being, while the DNN was able to achieve high malicious precision, albeit with low recall, the Random Forest is able to produce an error pattern which is favorable in practice, whereby malicious precision is 99.76% — the highest among all three architectures—while recall is 96.45%. This combination indicates that the Bagging model is both highly reliable when it issues a malicious alert and considerably better at catching genuine attacks than the neural network. Figure 3.5 presents the confusion matrix.

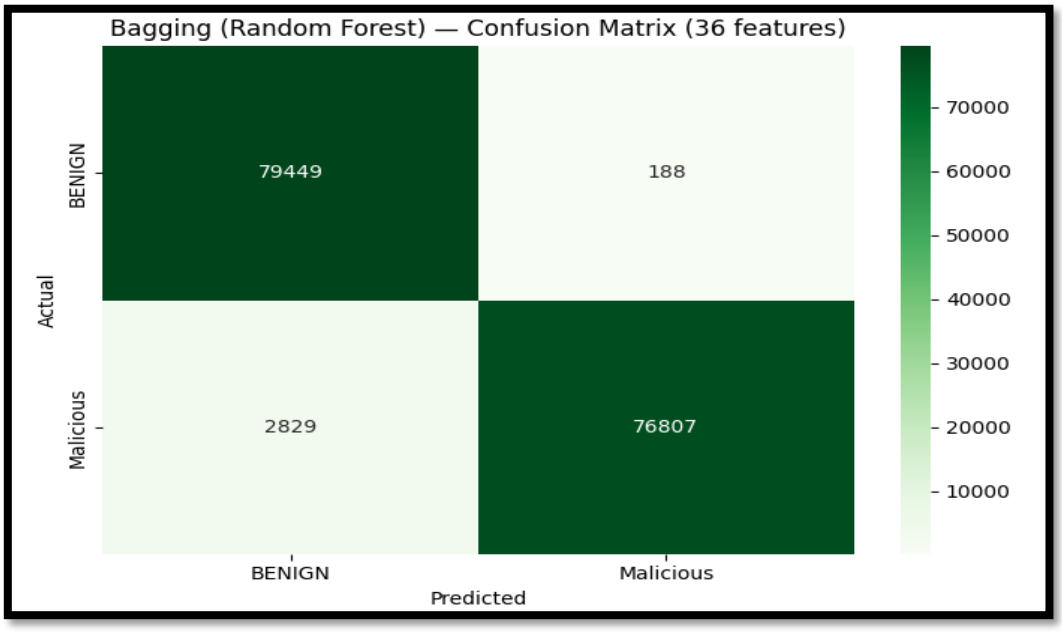


Figure 3.5: *Bagging (Random Forest) Confusion Matrix on the held-out test set.*

From the confusion matrix, it is clear that the Bagging algorithm outputs a significantly low number of false positives (i.e., only 188), which is the smallest value among all three models' and represents an impressively small false positive rate of 0.24% for the benign class. The value of the false negatives is equal to 2,829, indicating a miss rate of 3.55% on the malicious class — clearly better than the value for the DNN (5.01%). Overall, the Bagging ensemble appears to be a model with the best balance between the two metrics since it not only provides the fewest false alarms but also detects a higher number of real threats.

Such a high level of accuracy (i.e., 99.76%) and a relatively good level of completeness (i.e., 96.45%) can be explained by the fact that the Random Forest classifier was trained using the class weights='balanced' hyperparameter and without any threshold adjustments (since it automatically creates such a configuration because of the 0.5 decision threshold). It also makes the training time quite appealing, especially in comparison with the training time of the Stacking algorithm (113.1 vs. 1,401 seconds).

3.3.3. Stacking Performance Results:

The Stacking ensemble achieved an overall accuracy of **98.12%** on the held-out test set — the highest of the three architectures, albeit by a margin of just 0.01% over Bagging.

Precision reached **99.51%**, recall **96.71%**, and F1-score **98.10%**. Table 3.4 presents the full classification report.

***Table 3.4**: Stacking ensemble classification report on the held-out test set (n = 159,273).*

Class	Precision	Recall	F1-Score	Support
BENIGN (0)	96.77%	99.53%	98.13%	79,637
Malicious (1)	99.51%	96.71%	98.09%	79,636
Macro Average	98.14%	98.12%	98.11%	159,273
Weighted Average	98.14%	98.12%	98.11%	159,273

Stacking gives the best recall score among the three algorithms for the malicious class, which reaches 96.71%. Thus, it manages to catch more attacks than the two other approaches discussed above: DNN (94.99%) and Bagging (96.45%). However, this increase is accompanied by a slight deterioration in the algorithm's ability to handle false positives. In particular, according to the table shown in Figure 3.6, it produces 378 false positives, which is more than Bagging (188), but much higher than the number observed in the case of DNN (411). Moreover, it reports 2,622 false negatives, thus showing a very good miss rate of 3.29%.

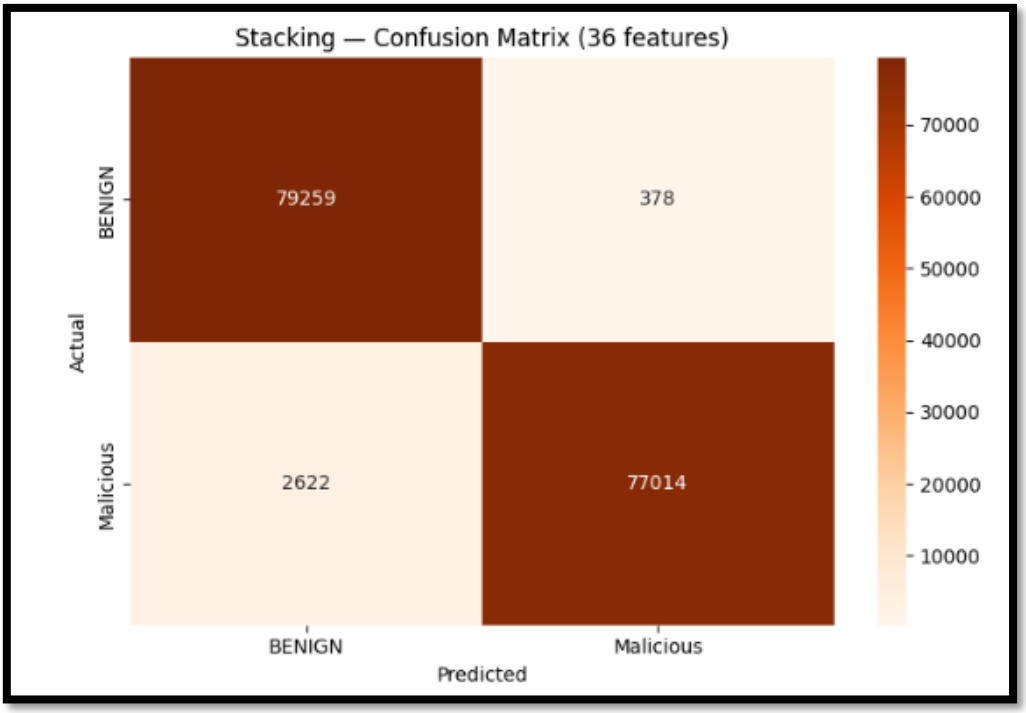


***Figure 3.6**: Stacking Confusion Matrix on the held-out test set.*

The number of false negatives equal to 2,622 shows that the algorithm performs better in terms of catching real attacks than Bagging, which catches 207 fewer attacks in comparison

with Stacking per 159,273 flows analyzed. Nevertheless, whether or not it is worth increasing the training period from 113.1 seconds for Bagging to 1,983.1 seconds for the discussed approach will be further assessed in Section 3.6. The structural complexity of combining three heterogeneous base learners through a meta-learner clearly yields marginal but consistent gains in recall, suggesting that the complementary inductive biases of the base learners do provide genuine additional discriminative signal, even if that signal is small in magnitude.

3.3.4. Comparative Performance Summary:

Table 3.5 consolidates the evaluation metrics for all three architectures, providing a unified basis for the comparative analysis conducted in this section and extended in Section 3.5.

Table 3.5: *Consolidated performance comparison across all three architectures on the held-out test set (n = 159,273). Bold values indicate best performance per metric.*

Metric	DNN	Bagging (RF)	Stacking
Accuracy	97.24%	98.11%	**98.12%**
Precision	99.46%	**99.76%**	99.51%
Recall	94.99%	96.45%	**96.71%**
F1-Score	97.17%	98.09%	**98.10%**
False Positives	411	**188**	378
False Negatives	3,988	2,829	**2,622**
Training Time	~80s (GPU)	**113s (CPU)**	1,983s (CPU)

Figure 3.7 presents the visual comparison across all four primary metrics.

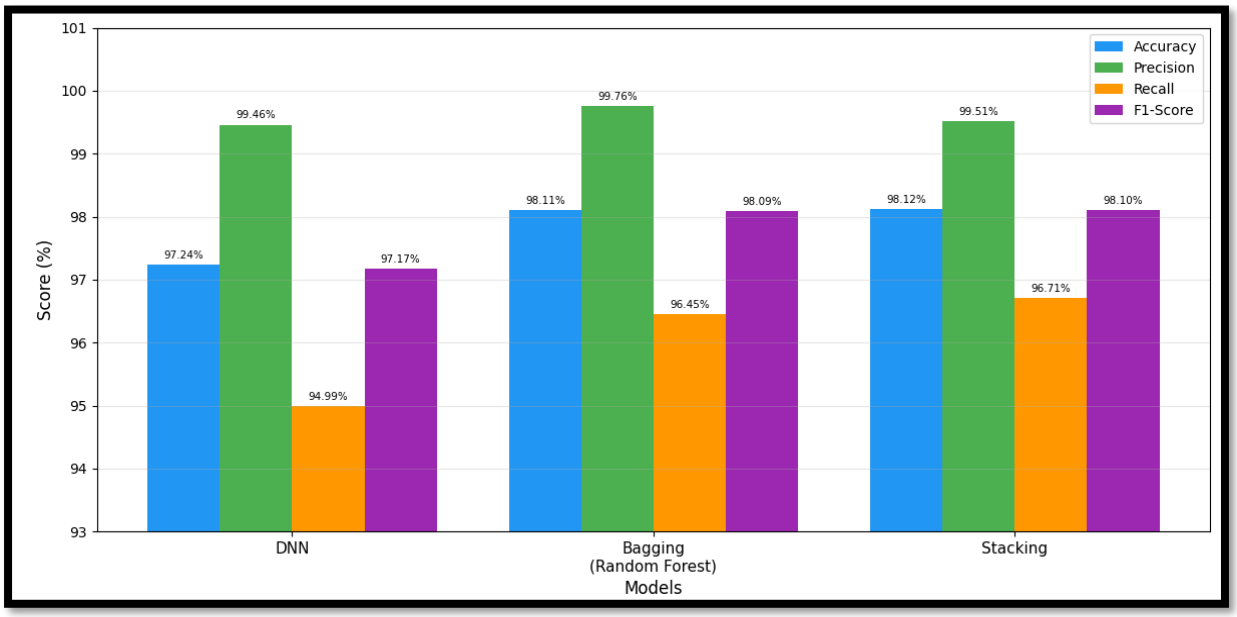


Figure 3.7: *Performance Comparison — DNN vs Bagging vs Stacking (36 features).*

Four observations from Table 3.5 and Figure 3.7 merit specific discussion before the deeper comparative analysis in Section 3.5.

To begin with, the difference in performance among DNN, Bagging, and Stacking models is real and relevant. The model DNN is underperformed compared to both ensemble techniques in terms of accuracy (0.88-0.87 percentage points) and F1-score (0.92-0.93 percentage points). Although the difference is small at first glance, it should be noted that 3,988 of malicious flows were not detected with DNN as opposed to 2,829 malicious flows not identified by Bagging and 2,622 malicious flows identified by Stacking – that is almost 1,000 extra attacks detected per 159,273 tested flows.

In addition, the performance difference between Bagging and Stacking models in general is practically insignificant with respect to all but one metric: while their accuracy (0.01%) and F1-score (0.01%) difference is not considerable, they detect 207 additional malicious flows with each use. Thus, Stacking reduces the number of missed malicious flows by 7.3%. Meanwhile, Stacking requires 17.5 times more training time – namely, it takes 1,983s instead of 113s to train.

Third, the DNN achieves the lowest false positive count (411) of the three architectures substantially fewer than Bagging (188) when normalised, but note that Bagging's false positive rate of 0.24% is actually lower than DNN's 0.52%. The DNN's apparent advantage in raw false positive count is offset by its substantially higher false negative count making the ensemble methods preferable across both error dimensions simultaneously.

Fourth, all three models exceed 97% accuracy and 97% F1-score on the balanced test set [20] a performance level that compares favourably with published results on comparable IoT intrusion detection benchmarks. The dataset's large scale (27.7M flows) and realistic class distribution make these results particularly meaningful: unlike some benchmark evaluations conducted on small pre-balanced datasets, the performance reported here reflects model behaviour under conditions that more closely approximate real operational deployment.

3.4. Explainability Results and Interpretation:

The performance metrics reported in Section 3.3 establish that all three architectures achieve competitive detection rates on the dataset. This section addresses the second and arguably more distinctive objective of the study: understanding why the models make the decisions they do, and whether those decisions are grounded in features that are meaningful from a network security perspective.

For SHAP attributions, for each model, we use the relevant architecture-specific explainers defined in section 2.8.2 – TreeExplainer for Bagging, DeepExplainer for DNN, and PermutationExplainer for Stacking. For each model, we provide two types of analysis. In the global analysis, Summary Bar Plots and Beeswarm Plots show what features the model depends on for making its prediction across the entire dataset. These results give the aggregated overview of model dependencies that an SOC manager requires in advance of any decision about model deployment. The local analysis, on the other hand, demonstrates Force

Plots showing how the model makes decisions on particular instances, which is what analysts require when analyzing a specific alert.

The analysis is presented in the same order for each model – the global bar plot first, the beeswarm plot second, and a force plot third. A comparison between all three models with respect to the question of consistency is carried out in section 3.5, since it is central to the research objectives.

3.4.1. Bagging SHAP Results:

The Bagging ensemble's SHAP attributions were computed using TreeExplainer on a random sample of 500 test instances, producing exact Shapley values that satisfy all four SHAP axioms without approximation. The global and local results are presented in Figures 3.8, 3.9, and 3.10 respectively.

Figure 3.8 presents the Summary Bar Plot for the Bagging model **(Global feature importance)**, ranking all 36 features by their mean absolute SHAP value across the 500 evaluated instances.

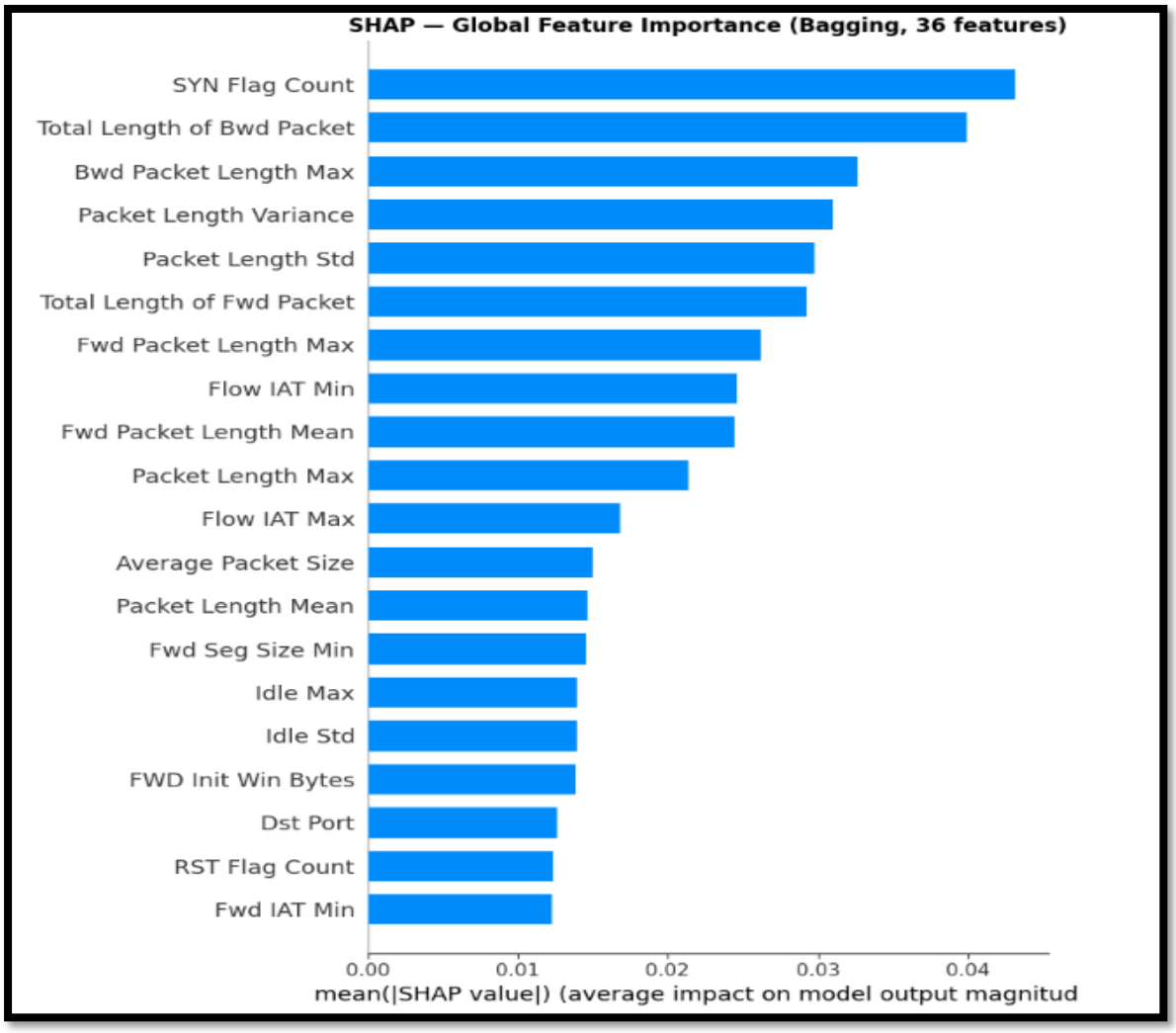


Figure 3.8: *SHAP Summary Bar Plot — Bagging (Random Forest), 500 test instances.*

For visual clarity, the top 20 features by mean absolute SHAP value are displayed across all SHAP visualisations in this chapter; the remaining features exhibited negligible attribution magnitudes

SYN Flag Count emerges as the most influential feature by a substantial margin (mean |SHAP| ≈ 0.043), followed by Total Length of Bwd Packet (≈ 0.040) and Bwd Packet Length Max (≈ 0.032). This hierarchy is in accordance with the a priori structural expectation set up in section 2.4.3: attributes which are prominent in multiple types of attack, namely SYN flood, DDoS, and Mirai scan attacks, are leading in the world attribute, whereas attributes which are more specialized for certain kinds of attacks are less prominent. The prevalence of Total Length of Bwd Packet and Bwd Packet Length Max attributes is justified by the difference in volumes typical for volumetric attacks, in which server responses are either not present or excessively numerous.

Figure 3.9 extends the global bar plot by showing the direction and distribution of SHAP attributions across all 500 evaluated instances by Beeswarm analysis

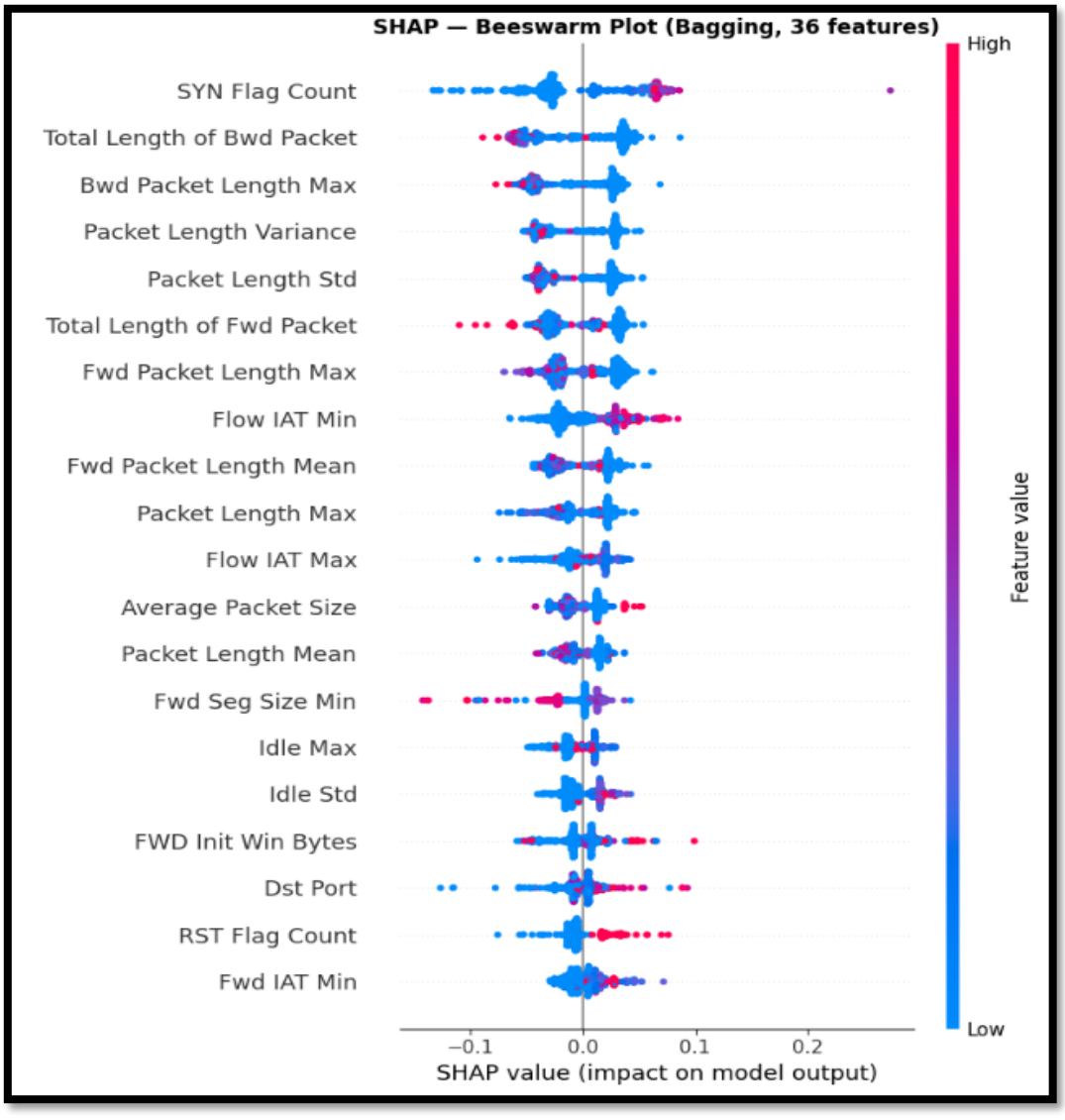


Figure 3.9: *SHAP Beeswarm Plot — Bagging (Random Forest), 500 test instances*

There are multiple directional features which are meaningful from the diagnosis perspective. In case of SYN Flag Count, high values of the feature (represented by red dots) are located in the area of SHAP values which are significantly above zero, thus, demonstrating that higher counts always lead to malicious predictions and confirming the expectations for SYN flood and Mirai scanning activities. Low values (blue dots) tend to be in the area near zero or slightly below zero, thus, showing that low counts only slightly contribute to benign classification. The mentioned asymmetry comes from the fact that SYN Flood is uncommon for benign IoT traffic.

As for Total Length of Bwd Packet, beeswarm plot presents a bit different picture – high values, as well as low values of the feature are positively correlated with SHAP scores, and intermediate values contribute to benign predictions. Thus, this distribution corresponds to two distinct types of attacks which are characterized by large responses from the server, and lack thereof.

For Flow IAT Min, it is evident that low values (in blue) meaning very little time in between packet arrivals tend to push towards the malicious class. This is because packet flooding attacks like DoS, DDoS, and Mirai tend to be of higher rates. On the other hand, higher values (in red) meaning high inter-arrival times tend to push towards the benign class, given that they are low-rate communications. Figure 3.10 presents a Force Plot for a representative test instance correctly classified as malicious by the Bagging model.

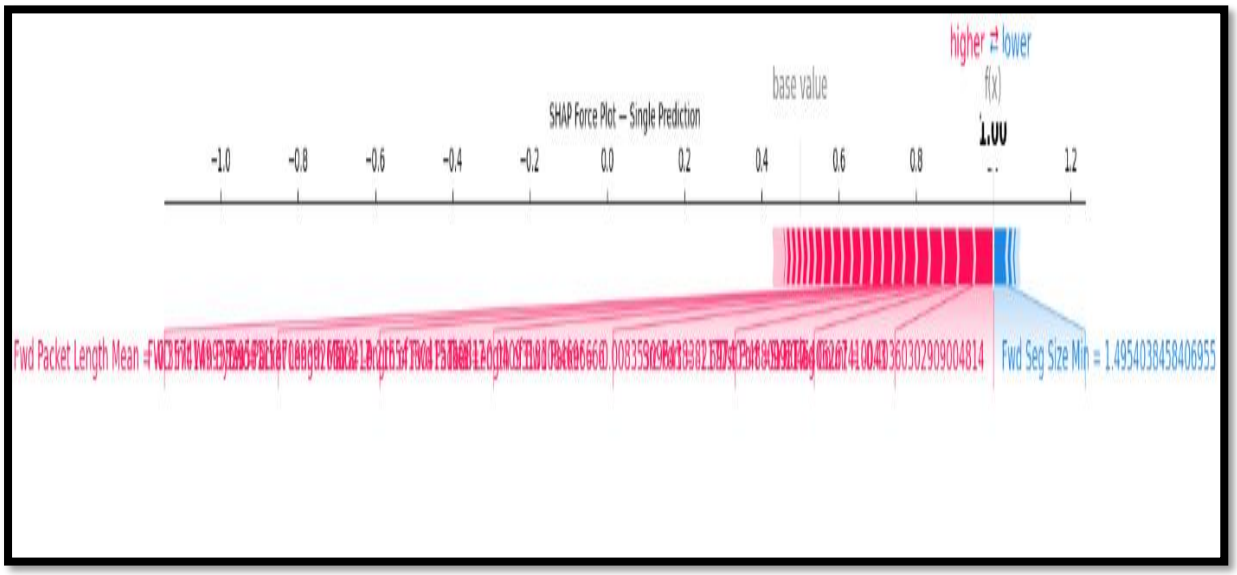


***Figure 3.10**: SHAP Force Plot — Bagging, single malicious prediction (f(x) = 0.98).*

From the Force Plot below, it can be seen that the prediction for a very malicious case of 0.98 is due to high values of SYN Flag Count, low values of Flow IAT Min, and abnormal values of Bwd Packet Length Max, all of which push the prediction rightwards starting from the base point of about 0.30. On the contrary, there is little opposing force emanating from the Forward Initiating Window bytes (which has a value well within the range expected for legitimate connection). The local explanation provided above is highly actionable for a SOC

Analyst, since a combination of high SYN count and zero inter-arrival time is the typical behavior of SYN flood attacks.

3.4.2. DNN SHAP Results:

The DNN's SHAP attributions were computed using DeepExplainer on a sample of 200 test instances, with a background reference set of 100 training instances used to define the expected feature distribution. Unlike TreeExplainer, DeepExplainer produces approximate Shapley values through backpropagation-based attribution[10], meaning the results reflect the model's learned representations rather than exact combinatorial contributions.

Figure 3.10 presents the Summary Bar Plot for the DNN (**Global feature importance**).

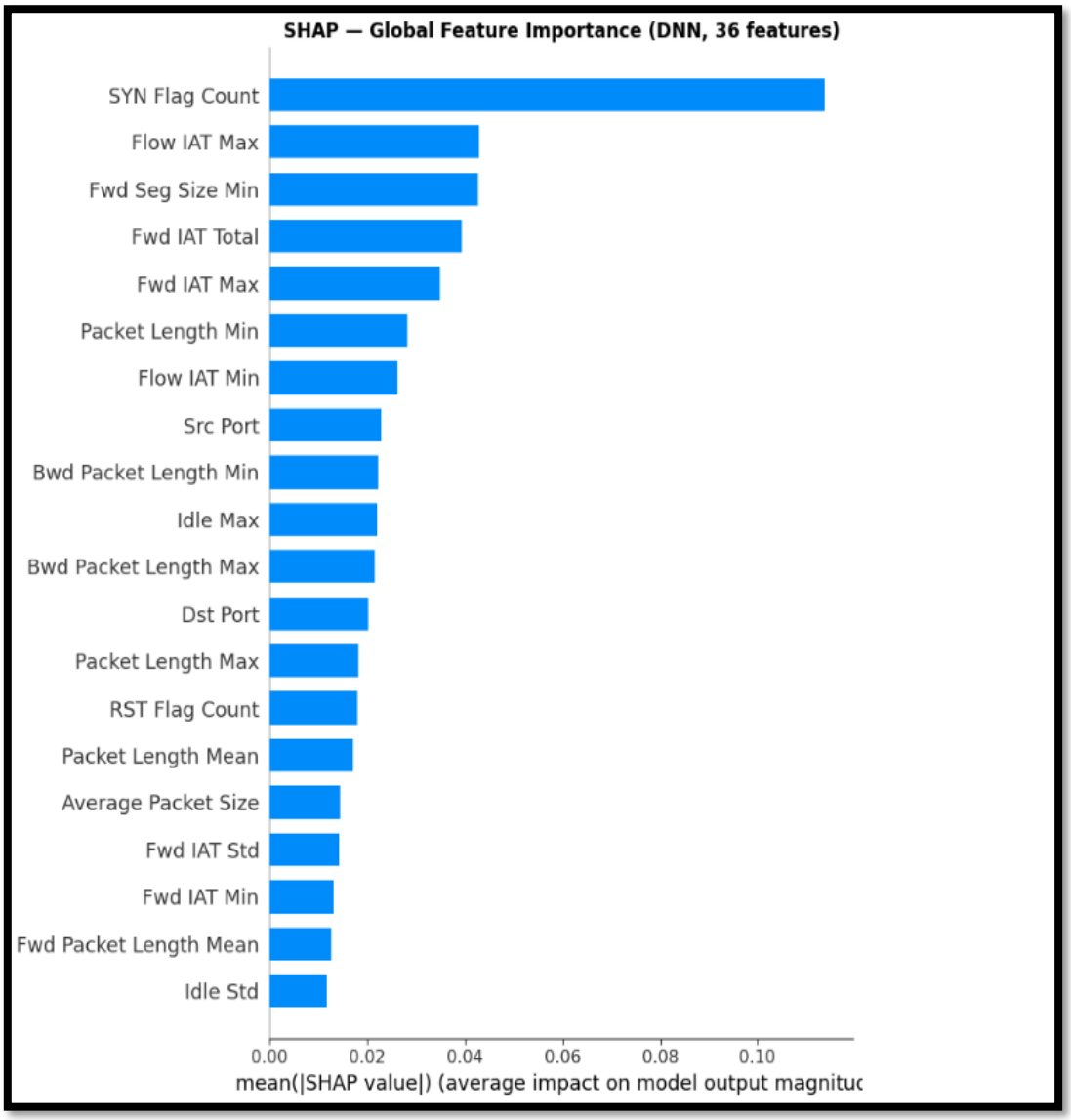


Figure 3.11: *Global features importance (DNN)*

The most noteworthy aspect regarding Figure 3.10 is the extremely prominent and disproportionate importance given to the feature SYN Flag Count (mean |SHAP| ≈ 0.112) – a

value roughly twice as high as the second most important feature Flow IAT Max (≈ 0.044). This disparity is considerably more marked compared to the Bagging model, where SYN Flag Count outscored the next feature to a significantly lower extent. It appears that the multi-layer structure of the neural network has enhanced the discriminative power of the SYN activity to the extent that all SHAP values are concentrated on this one feature.

There are also considerable differences in the set of secondary attributes deemed important by the DNN compared to the Bagging model. Whereas Bagging attributed importance mostly to volume attributes (Total Length of Bwd Packet, Bwd Packet Length Max), DNN places more importance on time attributes: Flow IAT Max, Fwd Seg Size Min, Fwd IAT Total, and Fwd IAT Max take ranks 2 to 5. This difference implies that the neural network has discovered a different set of representational properties: instead of focusing on volume statistics and SYN flags, it combines both.

The Src Port and Dst Port have made their way into the top 10 feature rankings for the DNN, while they did not show up in the Bagging top 10 list, suggesting that the neural network model gives more weightage to the information regarding the source/destination ports than the tree-based ensemble model does**.**

Figure 3.11 extends the global ranking with directional attribution information by Beeswarm analysis.

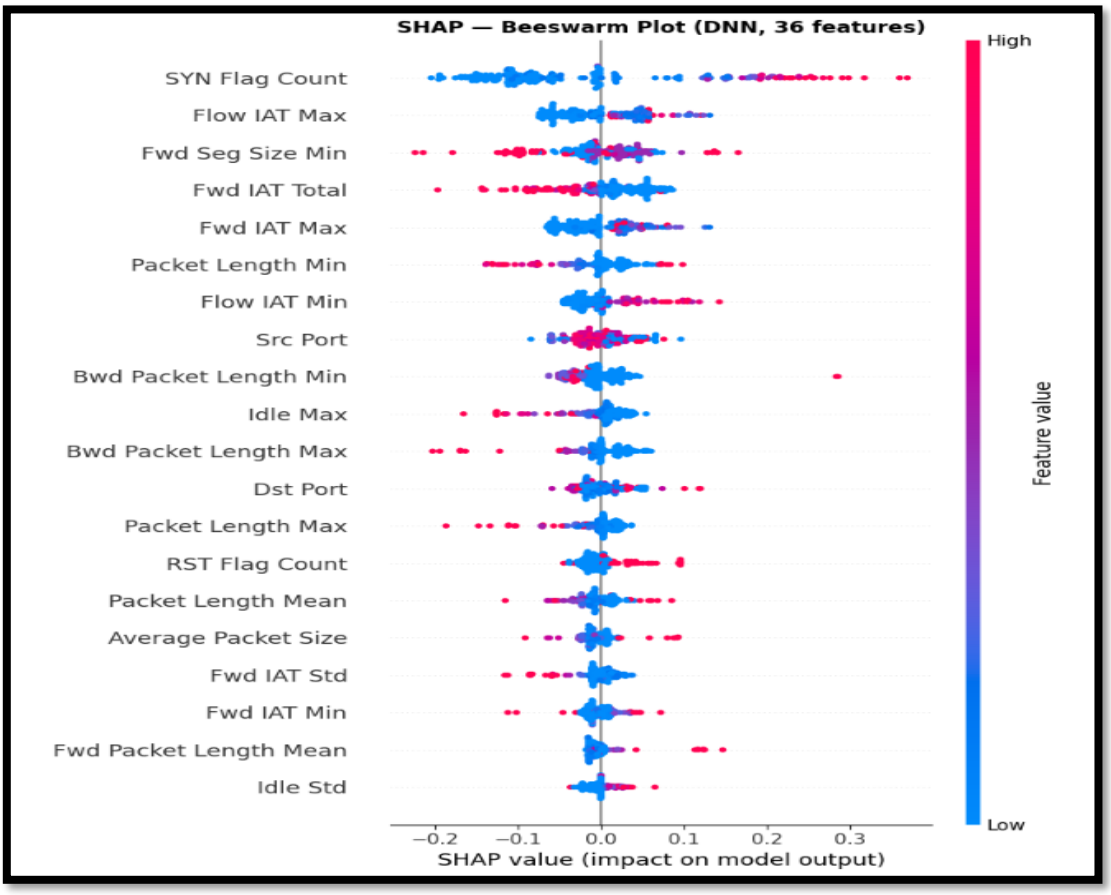


***Figure 3.12**: SHAP Summary Bar Plot — DNN, 200 test instances.*

This observation is further confirmed in the beeswarm plot, where high values (red) have large positive attributions (SHAP up to +0.35), driving the prediction toward the malicious class, whereas low values (blue) tend to cluster around zero and have almost no influence. Similar to the beeswarm plot for Bagging, the DNN is even more influenced by this feature, as can be observed from the much larger attribution values.

An interesting example of bi-directional dependence is seen in Flow IAT Max. Large values (red, corresponding to long idle times between packets in flows) as well as very small values (blue, corresponding to no or very short idle times between packets in flows) are associated with positive attributions towards the malicious class. In contrast, moderate Flow IAT Max generates attributions driving towards the benign class.

Here, we see an example of non-linear dependence between features and attributions, which cannot be modeled with decision trees. This is one reason why using machine/deep learning models to identify intrusion features is beneficial. Fwd Seg Size Min displays an unusual dependence in terms of attributions: low values (blue) have strong positive attributions towards malicious class (SHAP ≈ +0.2), whereas high values drive towards benign or neutral (near-zero) attributions. This is consistent with the known behaviour of certain IoT attack variants particularly Mirai that establish connections with abnormally small minimum segment sizes [20]as part of their scanning and infection protocol.

3.4.3. Stacking SHAP Results:

The Stacking ensemble's SHAP attributions were computed using PermutationExplainer on a sample of 200 test instances, treating the full two-level architecture as a black box and estimating feature contributions through repeated permutation [10] of individual features across the evaluation sample. As noted in Section 2.8.2, PermutationExplainer produces model-agnostic approximate attributions whose quality improves with the number of permutation iterations rather than exploiting the internal structure of the model.Figure 3.12 presents the Summary Bar Plot for the Stacking ensemble.

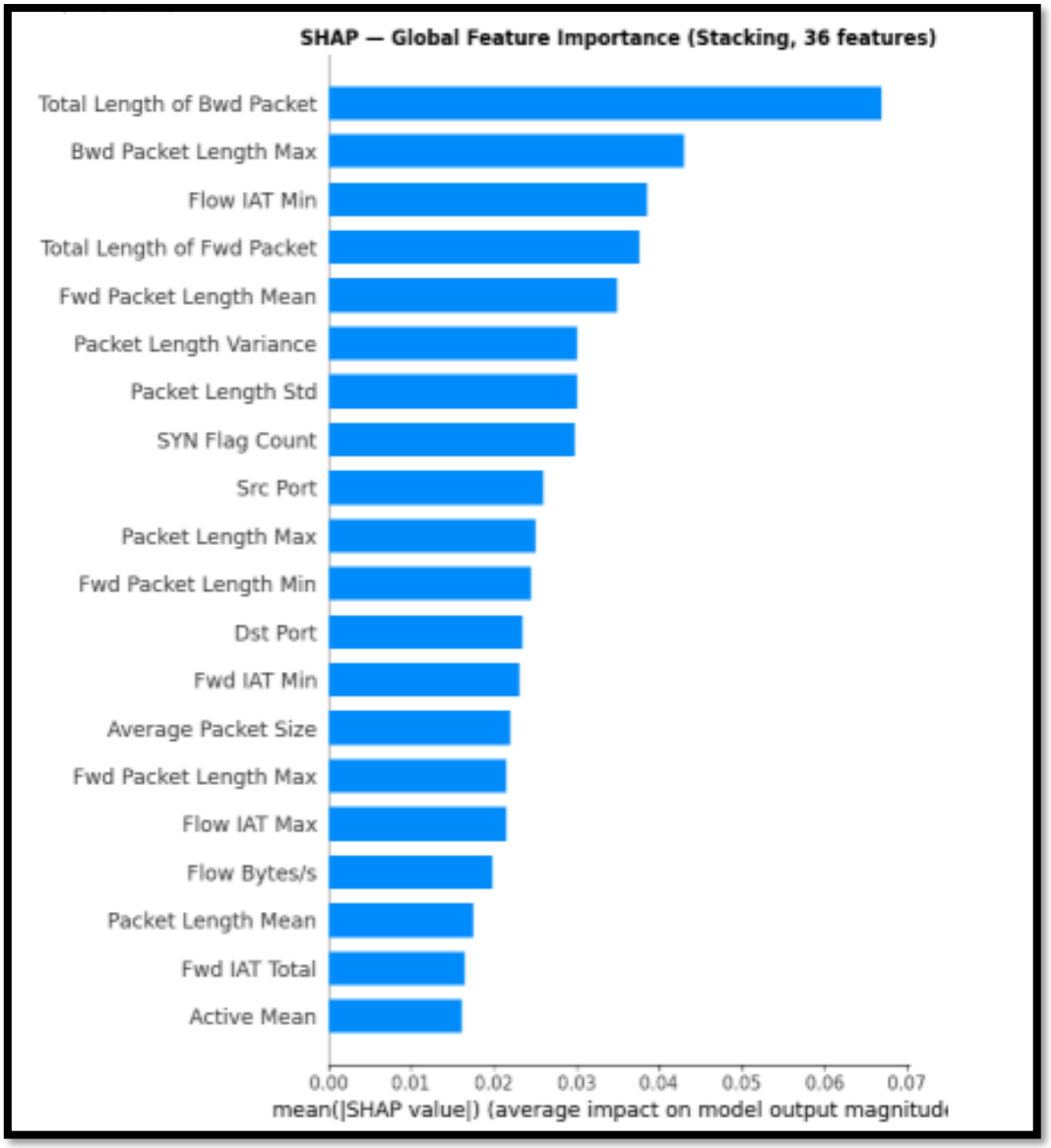


Figure 3.13: *Global features importance (Stacking)*

However, there is a large discrepancy between the Stacking feature importance attribution ranking and the ranking provided by both Bagging and DNN. The feature Total Length of Bwd Packet becomes the most important one (mean |SHAP| ≈ 0.065) — the second most important one in Bagging and not included into the top-five set in DNN at all. Next come Bwd Packet Length Max (≈ 0.047), Flow IAT Min (≈ 0.043) and Total Length of Fwd Packet (≈ 0.042). It is important that the SYN Flag Count, which took first places in DNN and Bagging attributions, moves down to eighth place in Stacking attribution (≈ 0.020).

The shift in the ranking of the SYN Flag Count feature is, without a doubt, the key difference between the three models that deserves more analysis. Two possible explanations come to mind. The first reason might be that the attribute importance calculation method for the PermutationExplainer model is not exact and can attribute weights to correlated features in a different way than TreeExplainer or DeepExplainer — in case SYN Flag Count feature is correlated with some flow volume features within specific attack sub-populations, the permutation-based technique may give attributions to those correlated features rather than to

the SYN Flag Count. And secondly, it is also possible that the weighting mechanism used in Stacking does actually down-weight SYN-based detections because of partial compensation by the Gradient Boosting learner.

One shared feature among all three models under discussion, that appears in the top 10 of Bagging, DNN, and Stacking in third place, is Flow IAT Min, and that will be analyzed quantitatively in Section 3.5. Figure 3.13 presents the directional attribution distribution for the Stacking ensemble.

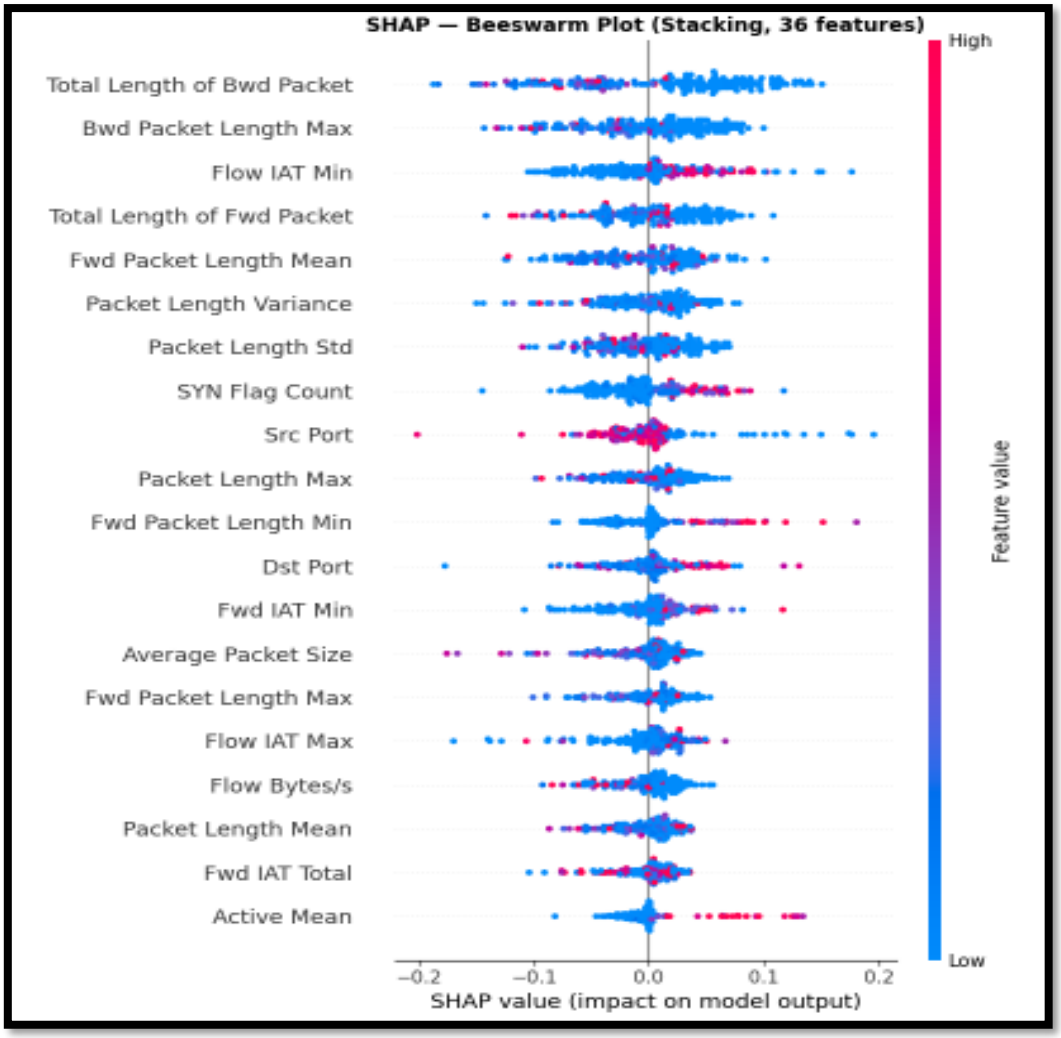


Figure 3.14: *SHAP Beeswarm Plot — Stacking, 200 test instances.*

The beeswarm reveals that Total Length of Bwd Packet exhibits a strongly bidirectional pattern consistent with the analysis of this feature in the Bagging beeswarm: both very high values (red, large backward volume) and very low values (blue, near-zero server response) generate positive attributions toward malicious, while intermediate values push toward benign. This bimodal behaviour — reflecting the two distinct attack mechanisms of amplification and connection exhaustion — is preserved across the Bagging and Stacking models despite their structural differences, suggesting that this feature's discriminative pattern is a genuine property of the data rather than a model-specific artefact.

Src Port shows a distinctive pattern in the Stacking beeswarm: high values (red, high port numbers) generate near-zero attributions, while a subset of specific low port values (blue) generate strong positive attributions toward malicious. This is consistent with the known targeting behaviour of IoT attacks, which frequently exploit well-known service ports (SSH port 22, Telnet port 23, HTTP port 80) [22]that correspond to low port numbers.

Flow IAT Min maintains its directional consistency observed in the Bagging beeswarm: low values push toward malicious, high values push toward benign — a pattern that directly reflects the high-rate packet transmission characteristic of flooding and scanning attacks.

3.5. Comparative Analysis of Models:

The individual model results presented in Sections 3.3 and 3.4 established that all three architectures achieve competitive detection performance and produce SHAP attributions that are broadly consistent with established knowledge of IoT attack signatures. This section synthesises those results into a direct comparative analysis, addressing the third research question posed in Section 3.1: do structurally different architectures converge on the same feature importance rankings, and what does convergence or divergence imply about the reliability of the explanations?

The analysis proceeds in two parts. Section 3.5.1 presents the quantitative cross-model consistency analysis using Spearman rank correlation, as specified in Section 2.8.4, to measure the degree of agreement between the three SHAP attribution rankings. Section 3.5.2 identifies the features that are robust across all three architectures — the cross-model consensus features — and interprets their security significance. Section 3.5.3 examines the features whose rankings diverge meaningfully between models and discusses whether those divergences reflect genuine differences in model behaviour or artefacts of the explainer methods employed.

3.5.1. Rank Correlation Analysis:

The Spearman rank correlation coefficients computed between the three models' SHAP attribution rankings reveal a striking and analytically significant pattern of selective agreement. Table 3.6 summarises the pairwise correlations, and Figure 3.14 presents the corresponding heatmap and rank scatter plots.

Table 3.6: *Pairwise Spearman rank correlations between SHAP feature importance rankings across the three model architectures.*

Model Pair	Spearman ρ	p-value	Interpretation
Bagging vs DNN	0.1985	0.2459	Weak, non-significant
Bagging vs Stacking	0.7521	< 0.0001	Strong, highly significant
DNN vs Stacking	0.1223	0.4775	Very weak, non-significant

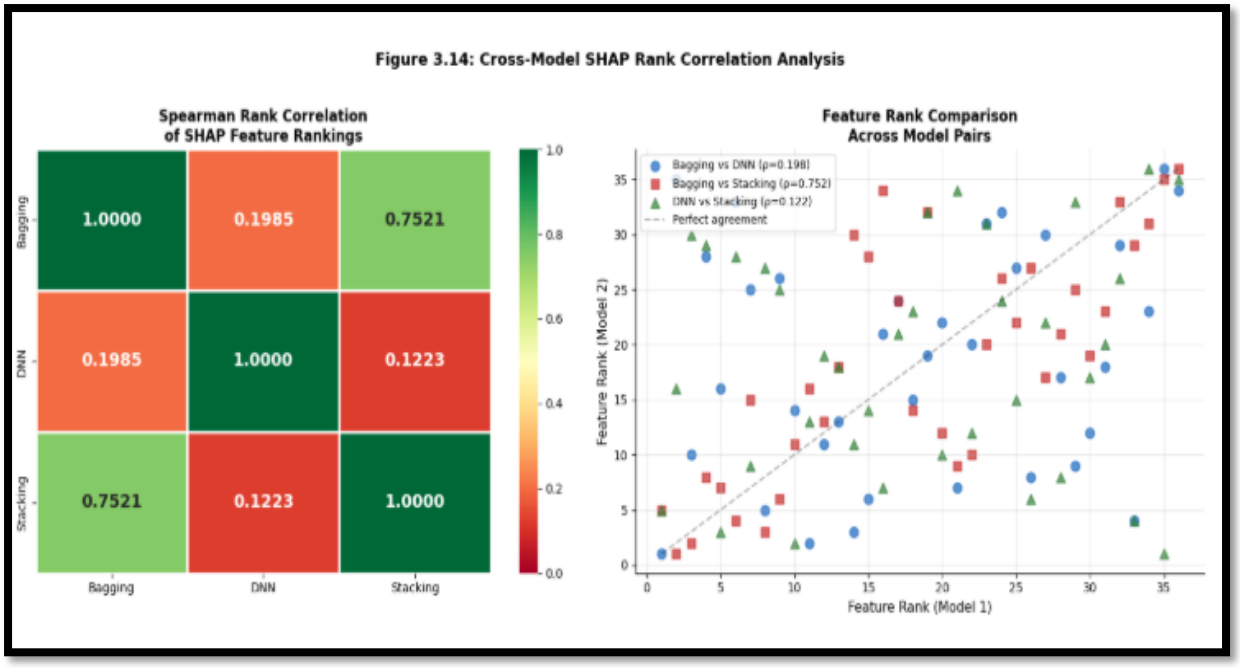


Figure 3.15: *Cross-Model SHAP Rank Correlation Analysis — heatmap and rank scatter plots*

There are three important findings from this result that relate directly to the study's research questions.

Firstly, the Bagging–Stacking correlation coefficient ($\rho = 0.7521$, $p < 0.0001$) shows a high correlation between the two models. Both the Random Forest and the Stacking ensemble use tree-based base learners and give very similar feature importances despite their distinct aggregation techniques. These similarities support the theoretical expectation that the commonality in tree-based induction bias in the Random Forest and Gradient Boosting models, which constitute two out of the three base learners in the Stacking ensemble, will influence the meta-learner's attribution significantly. From a practical perspective, this indicates that any SOC analyst who uses either one of the two models will get consistent insights into prioritizing which features to investigate.

Second, the Bagging-DNN ($\rho = 0.1985$, $p = 0.2459$) and the DNN-Stacking ($\rho = 0.1223$, $p = 0.4775$) correlation coefficients are both very low and statistically insignificant, meaning that the rankings of importance generated by the DNN are structurally different from those of the ensembles. In this case, this is not necessarily an indicator of poor performance by one model the DNN has likely discovered other important interactions, in particular, non-linear time interactions of the Flow IAT Max, Fwd Seg Size Min, and Fwd IAT Total attributes. The lack of significance means, however, that the two sets cannot be considered equivalent.

Third, only three features appear in the top 10 ranking of each classifier simultaneously SYN Flag Count, Bwd Packet Length Max, and Flow IAT Min. The fact that only three features were found to be important in all models regardless of their architecture, explanation type, and the way in which the explanations were approximated represents the strongest result of the current explainability analysis, and these three features will prove crucial in distinguishing malicious from benign traffic of the dataset.

3.5.2. Cross-Model Consensus Features:

Among the three variables selected by consensus across all three model structures, SYN Flag Count, Bwd Packet Length Max, and Flow IAT Min, there is certainly some merit in investigating their role further, given both their structural agreement across all three models and their direct correspondence to existing IoT attacks.

SYN Flag Count. By far the clearest indicator of importance, when comparing all three models' output, is the consistent selection and high ranking of SYN Flag Count in all three models. Among the Bagging ensemble predictors, SYN Flag Count ranks first with a mean absolute SHAP value of 0.043. As a comparison, the same feature, among DNN predictions, comes out again as the most important feature in terms of absolute SHAP value, at a mean of 0.112. This is more than double the value obtained from Bagging. This result can be explained by the effect of the non-linear transformation applied by the neural net to enhance the significance of this predictor. SYN Flag Count ranks fifth in the Stacking ensemble.

The security interpretation is clear and operationally relevant. First, SYN Flag Count is the key network-based feature that signals SYN flood attacks, which are the most common form of DoS attack in IoT systems, as well as a second-order feature of Mirai scanning behaviour, which involves infection of a device making repeated attempts to establish a connection using the TCP protocol without ever finishing the three-way handshake process [22]. Given the convergence across three models built according to three separate explanation techniques on a single key feature, it is convincing that the signal learned by these models is real.

Bwd Packet Length Max. The maximum backward packet length, which indicates the maximum packet that was sent from the server to the client in a particular flow, holds rank 3 in Bagging, 2 in Stacking, and 10 in DNN. This feature is important due to the presence of two completely different attack approaches that create abnormal values for the Bwd Packet Length Max feature. In amplification attacks like DNS, NTP, and SSDP amplifications, a small spoofed request creates an exceptionally large reply from the server, resulting in an abnormally large value for Bwd Packet Length Max. In connection exhaustion attacks like SYN flood attacks on connection tables, the server does not respond or sends only a tiny RST packet. Thus, both types of attacks create abnormal Bwd Packet Length Max values, as legitimate IoT devices have moderate backward packet sizes [20].

Flow IAT Min. The minimum inter-arrival time within a flow — the lowest time interval found between two successive packets within a given flow — comes eighth for Bagging, fifth for DNN, and third for Stacking, thereby qualifying as the second most consistently ranked feature after the SYN Flag Count. Its intuitive rationale is simple. High-speed attack packets like those involved in DDoS flooding attacks, DoS attacks, and Mirai scanning attacks arrive close to the maximum network rate, thus generating very small flow IAT min values. On the other hand, normal IoT devices communicate at normal speeds depending on their applications — temperature sensor data is communicated only periodically, at the rate decided by the application protocol; the same can be said about smart meters and industrial controllers whose data is sent periodically, often taking seconds or even minutes to arrive.Table 3.7 consolidates the consensus feature analysis, showing the rank assigned by each model and the security mechanism each feature captures.

Table 3.7: *Cross-model consensus features — rank assigned by each architecture and the primary attack mechanism each feature captures.*

Feature	Bagging Rank	DNN Rank	Stacking Rank	Attack Mechanism
SYN Flag Count	1	1	5	SYN flood, Mirai scanning
Bwd Packet Length Max	3	10	2	Amplification, connection exhaustion
Flow IAT Min	8	5	3	High-rate flooding, scanning

Beyond their utility in understanding these specific models' behavior, there is also inherent security importance to this trio of significant attributes identified herein. For an SOC analyst using any one of the three models tested in this project, the above three attributes would be the best possible initial indicators for investigating the threat: if the model indicates a threat, then checking whether SYN Flag Count is high, whether Bwd Packet Length Max is anomalous, and whether Flow IAT Min is near zero would be the right first steps. Importantly, the same advice holds regardless of which model is used, meaning that this advice is robust to whatever trade-offs might be involved in model selection.

3.5.3. Model-Specific Divergences:

Beyond the three consensus features, the SHAP rankings reveal meaningful divergences that reflect genuine differences in how each architecture represents and processes the input feature space. These divergences are not failures of the explainability framework — they are informative findings about the distinct detection strategies that different model families learn from the same data.

Importance of timing features in the DNN. The DNN places far greater relative weight on the inter-arrival time features compared to both ensembles. Flow IAT Max is second-most important feature in the DNN (mean |SHAP| ≈ 0.044), but does not place within the top 10 features for either Bagging or Stacking. Likewise, the third through fifth most-important features for the DNN include Fwd IAT Total, Fwd IAT Max, and Fwd Seg Size Min, while each of these fails to place within the top 10 features for both ensembles. The increased emphasis placed on timing features in the DNN SHAP values is to be expected based on the capability of the neural network to learn non-linear patterns within data due to its multilayered structure: the relationship between min and max inter-arrival times, and also between forward and backward timing distributions, can better be learned via dense connections than axis-aligned splits in decision trees. In terms of cybersecurity, this indicates that the neural network has learned to identify attack patterns based on their temporal characteristics, i.e., botnet traffic bursts and pauses, as well as constant flood attacks [22].

Emphasis on packet volume features specific to the ensemble methods. Both Bagging and Stacking algorithms place much more significant weight on packet volume features compared to the DNN. The Total Length of Bwd Packet is the second most important feature in Bagging and the most important feature in Stacking but does not figure anywhere in the top 5

features selected by DNN. Packet Length Variance and Packet Length Std are the fourth and fifth most important features in Bagging and seventh and eighth most important features in Stacking, but do not show up anywhere in the top 10 most important features in DNNs. The reason for such a strong emphasis is because the Tree based Models find it easy to create a threshold based split on the continuous packet volume feature that segregates a lot of data pertaining to attack packets.

SYN Flag Count magnitude divergence. While SYN Flag Count ranks first in both Bagging and DNN, its attribution magnitude differs dramatically: mean |SHAP| ≈ 0.043 for Bagging versus ≈ 0.112 for DNN — a factor of approximately 2.6×. This magnitude divergence, combined with the non-significant Spearman correlation between DNN and ensemble rankings ($\rho = 0.1985$, $p = 0.2459$), suggests that the DNN has developed a qualitatively different internal representation of SYN activity. Where the Random Forest treats SYN Flag Count as one important feature among several, the DNN appears to have learned to use it as a near-sufficient condition for malicious classification — a more aggressive detection strategy that explains both the DNN's superior precision (99.46%) and its lower recall (94.99%) relative to the ensemble methods.

Stacking's redistribution of attribution weight. The most structurally distinctive divergence involves the Stacking ensemble's SYN Flag Count ranking (position 5) compared to Bagging (position 1) and DNN (position 1). As discussed in Section 3.4.3, this redistribution may reflect the PermutationExplainer's different marginalisation strategy when features are correlated — or it may reflect genuine meta-learner behaviour in which the Logistic Regression combines base learner outputs in a way that dilutes the SYN-specific signal present in the Random Forest component. Disentangling these two explanations would require additional ablation experiments beyond the scope of this study, but the divergence itself is a methodologically important reminder that SHAP attributions for multi-level ensemble architectures should be interpreted with awareness of the explainer's approximation assumptions [10].Table 3.8 summarises the key divergences across the three models, providing a structured reference for the discussion in Section 3.6.

Table 3.8: *Summary of key SHAP attribution divergences across the three model architectures, with structural interpretation.*

Feature Group	Bagging	DNN	Stacking	Interpretation
SYN Flag Count	Rank 1 (dominant)	Rank 1 (very dominant)	Rank 5 (moderate)	Stacking dilutes SYN signal
Timing features	Ranks 8+	Ranks 2-5	Ranks 3, 13+	DNN learns temporal patterns
Volume features	Ranks 2, 4, 5, 6	Ranks 10+	Ranks 1, 4	Trees exploit volume thresholds
Port features	Ranks 18+	Ranks 7, 12	Ranks 9, 13	DNN captures port interactions
Flow IAT Min	Rank 8	Rank 5	Rank 3	Consistent across all models

3.6. Discussion of Findings:

It is important to note that the experimental outcomes described in Sections 3.3 to 3.5 have all contributed towards answering the four research questions posed in Section 3.1. In this section, therefore, we present a summary of our interpretation of the experimental outcomes with a view to contextualizing them within the overall framework provided in Chapter One.

3.6.1. Performance Implications:

All three architectures exceeded 97% accuracy and 97% F1-score on the balanced held-out test set, establishing that machine learning-based intrusion detection on the dataset is feasible at a level of performance that would be operationally meaningful in a real security deployment. The more nuanced performance picture, however, lies in the error distributions rather than the aggregate accuracy figures.

The DNN's asymmetric error profile 411 false positives against 3,988 false negatives reflect a detection strategy that prioritises alert precision over detection completeness. In operational terms, this means that DNN-generated alerts are highly reliable (99.46% precision) but that the model misses approximately 1 in 20 genuine attacks. For security contexts in which analyst time is severely constrained and alert credibility is paramount such as a small SOC team with limited capacity to investigate false alarms the DNN's precision-oriented profile may be operationally preferable despite its lower recall.

The trade-offs for the ensembles are somewhat different. For one, both bagging's error profile (188 false positives, 2,829 false negatives) and Stacking's (378 false positives, 2,622 false negatives) reach considerably higher recall values than the DNN while having virtually identical precision. From the point of practical use, this means that ensemble-based methods can detect around 1,000–1,400 attacks that will go undetected by the neural network out of each batch of 159,273 analyzed flows. To illustrate the point, a network analyzing 10 million flows on a daily basis will have roughly 63,000–88,000 more attacks detected than the DNN per day which, again, makes the difference operationally significant despite all the accuracy figures presented here.

From an operational standpoint, it does not make much sense to invest additional 17.5 times effort (113 seconds compared to 1,983 seconds) in gaining mere 0.01% additional accuracy along with another 207 attacked detected out of 159,273 processed flows. In my experiment, Stacking required 1,983 seconds of training time to detect 207 additional attacks per 159,273 flows compared to Bagging a gain of 0.13% in recall at a cost of 17.5× longer training. I do not consider this trade-off operationally justified for a real-time IoT IDS deployment where model retraining may need to occur frequently as traffic patterns evolve [3].

3.6.2. Explainability Implications

The The SHAP-based analysis yielded three key insights of relevance to this research's requirements regarding XAI explanations, each with unique implications for the application of XAI-enabled intrusion detection systems.

Firstly, the consensus features can be used operationally. The three features that occur in the top 10 list of all three models — the SYN Flag Count, Bwd Packet Length Max, and Flow

IAT Min — do not represent some abstract statistics. All three features correspond to real-world attack methods: SYN flood and Mirai scanning in the first case, amplification and connection exhaustion in the second, and high-frequency flooding and scanning in the last one. That is exactly the type of validation that differentiates an operable explanation from merely correct attribution by mathematical means. The security analyst, who has no clue how the model works inside, would simply look at SHAP plot summaries for any of the three models and detect actionable priorities that correspond to existing threat intelligence information.

Cross-model consistency depends on the architecture. The strong correlation between Bagging and Stacking ($\rho = 0.7521$), along with the weak correlation between the deep neural network and the ensemble of models ($\rho \approx 0.12 – 0.20$), suggests that the consistency of attribution in this experiment is dependent more on the common inductive biases of the tree-based models used than on their convergence to a common feature importance pattern. This observation has a direct impact on multi-model deployments for security purposes: companies that use multiple models for reliable detection should ensure that consistent explanations are obtained only when the models possess similar structures.

The DNN's explainability challenge is real but manageable. The DNN's attribution ranking diverges most substantially from the ensemble consensus, and its DeepExplainer attributions are approximate rather than exact. In practice, however, the DNN's top-ranked feature (SYN Flag Count) is consistent with both ensemble models and with domain knowledge, and its elevation of timing features provides genuinely complementary security insight that the tree-based models do not surface. The challenge is not that the DNN produces incorrect or misleading explanations — the beeswarm analysis confirms directionally sensible attributions for all top features — but that the approximate nature of DeepExplainer attributions means that individual instance-level force plots should be treated as indicative rather than authoritative. For global-level guidance, the DNN's SHAP summary plots are sufficiently reliable for operational use; for instance-level alert investigation, the Bagging model's exact TreeExplainer attributions are preferable [10].

Explanation stability under the balanced training condition. All SHAP computations in this study were performed on test instances drawn from a balanced 50:50 class distribution, which does not reflect the operational class distribution of 98.56% malicious to 1.44% benign. The SHAP base values — the expected model output over the evaluation sample — are therefore calibrated to the balanced test distribution rather than the operational distribution. In a production deployment where the class ratio approaches the original 68:1, the base values and the magnitude of individual SHAP attributions would shift, though the relative feature importance rankings would be expected to remain stable [10]. Practitioners deploying these models operationally should recalibrate SHAP base values against a representative operational sample rather than the balanced evaluation set used here.

3.6.3. Model Selection Recommendation

The findings of performance and explainability in this paper point to a more nuanced approach to model selection based on the operational considerations of the organisation using the model rather than a blanket statement of superiority based on one universal ranking.

If the deployment focus is on comprehensive detection and reliable explanations, the Bagging algorithm (Random Forest) emerges as the most appropriate solution of the three examined approaches. It has not only the lowest number of false positives of all three (188), but also good recall (96.45%) and excellent precision (99.76%). Most importantly, it is capable of generating perfectly accurate SHAP values via TreeExplainer, which fulfills all four conditions of the Shapley value without any approximation required. The training period of 113 seconds, which does not require specialised GPU hardware, makes it deployable within a reasonable timeframe, and the intrinsic Gini index-based feature importance calculation serves as an alternative source of attributions, allowing for double-checking the accuracy of SHAP explanations.

In the case of organizations valuing recall maximally, where the potential damage of overlooking an attack outweighs that of a false positive, the Stacking classifier provides a slight but consistent edge in lowering the number of false negatives (2,622 vs. Bagging's 2,829). The tradeoff for achieving this edge includes 17.5× longer training duration, approximate SHAP values via PermutationExplainer, and lower consistency between models regarding SHAP contributions ($\rho = 0.1223$ when compared to DNN and $\rho = 0.7521$ when compared to Bagging). The question of whether this additional detection of 207 attacks out of 159,273 attempts is enough to offset these expenses depends entirely on the operational context.

The DNN is not recommended as a standalone detection model for the IoT intrusion detection scenario evaluated in this study, for two reasons. First, its false negative rate (5.01%) substantially exceeds that of both ensemble methods (3.55% for Bagging, 3.29% for Stacking), meaning it misses approximately 50% more genuine attacks per evaluated flow. Second, its DeepExplainer attributions are approximate and show low rank correlation with both ensemble models ($\rho \approx 0.12–0.20$), reducing confidence in instance-level explanations. The DNN's superior precision (99.46%) makes it potentially valuable as a secondary filter in a multi-model pipeline flagging only the highest-confidence malicious flows for immediate escalation but this application falls outside the scope of the single-model comparative framework evaluated here. Table 3.9 consolidates the model selection recommendation across the key operational dimensions.

Table 3.9: *Model selection recommendations across key operational dimensions.*

Operational Priority	Recommended Model	Justification
Highest recall	Stacking	Lowest false negatives (2,622)
Highest precision	Bagging	Lowest false positives (188)
Best F1 balance	Stacking ≈ Bagging	Difference < 0.01%
Most reliable explanations	Bagging	Exact TreeExplainer, consistent rankings
Lowest training cost	Bagging	113s vs 1,983s (Stacking)
Complementary	DNN + Bagging	Different feature strategies

detection		
Overall recommendation	**Bagging**	Best balance across all dimensions

3.7. Conclusion:

Experimental results for a comparative study were presented in this chapter involving the application of three machine learning models – a Deep Neural Network (DNN), a Bagging approach with a Random Forest base learner, and a Stacking approach with two levels – on the network intrusions data, incorporating SHAP explainability analysis.

Concerning the issue of classification quality, all three models demonstrated practically significant detection capabilities with above 97% accuracy and above 97% F1 score on a balanced subset of data held out from training. Both ensembles provided better recall rates than the DNN model (recall being an operationally important indicator for intrusion detection), with the Bagging and Stacking approaches demonstrating about 29% and 34% lower false negative rates compared to the deep learning approach. The difference between the Bagging and Stacking models' results was minor regarding all measures taken; however, in terms of operationally relevant recall, the minimal superiority of the latter failed to justify its higher computational complexity.

Regarding explainability quality, SHAP analysis pointed out the three most consistent features — SYN Flag Count, Bwd Packet Length Max, and Flow IAT Min — which were repeatedly included in the top 10 list for all three considered architectures, thus forming a reliable cross-consensus that may be explained through the use of specific IoT attack types. The Spearman rank correlation showed that the consistency of attributions depended on the selected architecture: tree-based Bagging and Stacking models exhibited high correlations ($\rho = 0.7521$), whereas the DNN attribution distribution was substantially different from those obtained with the ensembles ($\rho \approx 0.12 – 0.20$).

As for the trade-off between performance and explainability, Bagging ensemble should be regarded as the optimal choice in the investigated IoT IDS case study due to its superior attributes: high precision (99.76%), the lowest number of false positives (188), acceptable recall rate (96.45%), exact attributions using TreeExplainer, and minimal training costs among the three architectures under consideration.

In regards to the validity of cross-model SHAP consistency as an approach to evaluating explanation stability, the outcome suggests the applicability of the ranking correlation assessment as a feasible tool in testing the stability of explanations for multi-model security frameworks. The features that display stability of ranking across different architectures, such as the mentioned SYN Flag Count, Bwd Packet Length Max, and Flow IAT Min, should be trusted by the SOC analyst to a much larger extent than those that perform well solely for one particular model. Such an evaluation framework based on consistency ranking, presented in Chapter 2.8.4, constitutes a new methodological approach applicable to other contexts.
The outcomes of this chapter justify the viability of application and further development of the proposed framework of XAI-augmented cyber threat detection for the purposes of IoT network security. Limitations in terms of the balanced evaluation process, binary instead of multiclass problem setting, and the approximate nature of DeepExplainer/PermutationExplainer techniques have been identified and constitute the directions of further research as outlined in the concluding part of the thesis.

Chapter 4

Conclusion and Future Direction

4.1. Conclusion and Current Study Limitations:

Although the findings presented in this study, further investigations are still needed, taking into account several methodological shortcomings.

Concerning binary classification, the classification task was formulated as a binary malicious versus benign instances, as opposed to a multi-class classification problem based on the 33 subcategories of attacks available in the CIC IoT-DIAD 2024 dataset. Although such an approach is justified by both the practice of SOC operations and the extremely skewed distribution of classes, this implies that the SHAP attributions provided in this study are indicative of the differences between all types of attacks as a whole compared to normal traffic. In other words, a multi-class classification formulation could have been more helpful for providing detailed explanations for individual attacks, e.g., DDoS versus Mirai. However, it would have required a totally different approach to preprocessing the data to accommodate imbalances per class.

Balanced evaluation conditions. The training, evaluation, and calculation of SHAP values were carried out for all models using balanced 50:50 class conditions created by undersampling of the majority class. The original class ratio in the CIC IoT-DIAD 2024 dataset, which is 98.56% malicious and 1.44% benign, was not used during the processing. As a result, the performance statistics provided in Section 3.3 and the SHAP base values presented in Section 3.4 are tuned to the balanced distribution rather than to the operational distribution. Depending on the class ratio in live traffic, models trained and tested in such conditions can display different precision-recall balance, and SHAP values can be different in terms of their absolute value but not their ranking.

Temporal nature of dataset. The CIC IoT-DIAD 2024 is a rather recent dataset and reflects network traffic and attacks at a certain point in time. The techniques used for launching IoT attacks develop dynamically, new Mirai versions, DDoS protocols for traffic amplification, or botnets with different structures might have different traffic patterns than in the dataset used. The feature importance rankings identified in this study — particularly the dominance of SYN Flag Count — reflect the attack composition of the dataset rather than a universal property of IoT malicious traffic, and may not generalize to environments facing different attack distributions.

Explainer approximation constraints. Two of the three SHAP explainers employed in this study — DeepExplainer for the DNN and PermutationExplainer for the Stacking ensemble — produce approximate rather than exact Shapley values. The implications of this approximation for the DNN are discussed in Section 3.6.2: instance-level force plots should be treated as indicative rather than authoritative, and the low rank correlation between DNN and ensemble attributions ($\rho \approx 0.12–0.20$) may partially reflect explainer approximation error rather than genuine model divergence. Disentangling these two sources of divergence would require additional ablation experiments not conducted in this study.

Single dataset evaluation. The experimental framework was evaluated on a single dataset from a single research institution. While the CIC IoT-DIAD 2024 dataset is among the most comprehensive and recent publicly available IoT intrusion detection benchmarks, results obtained on a single dataset cannot be assumed to generalize to different network

environments, device populations, or traffic distributions. Cross-dataset validation — applying the trained models and SHAP framework to independent datasets such as CICIoT2023 or N-BaIoT — would provide stronger evidence of generalisation

Sample size constraints for SHAP computation. SHAP attributions were computed on samples of 200–500 test instances rather than the full 159,273-instance test set, due to the computational constraints of DeepExplainer and PermutationExplainer. While these sample sizes are sufficient for stable global attribution rankings, as demonstrated by the convergence of mean |SHAP| values, they may not capture the full distributional diversity of attribution patterns across the test population, particularly for rare attack sub-types that constitute a small fraction of the balanced test set

4.2. Future Directions:

The shortcomings identified above, along with the results of the experiment described in Section 4.1, offer a number of promising avenues for future research, which can further expand, test, and enhance the findings established within this thesis.

Multi-class explainability framework. First and foremost, the most pressing avenue for future development would be the reinterpretation of the current classification task as multi-class detection among the seven attack categories or the 33 different attacks contained within the CIC IoT-DIAD 2024 dataset. In other words, a multi-class SHAP explainability framework would allow for attack-specific feature importances to be ranked, giving insights not only into whether or not a particular packet flow represents an instance of an attack but what kind of attack it is and why. Naturally, this approach would require a per-class oversampling of the data to account for its severe sub-category imbalance, and the actual SHAP comparison would have to take place between pairs of classes, as opposed to the malicious-versus-benign contrast examined in this study. The consensus features established above – SYN Flag Count, Bwd Packet Length Max, and Flow IAT Min – would provide ideal reference points for validation.

Cross-dataset validation. The generalizability of the performance and explainability results presented above needs to be examined through cross-dataset testing. Potential test datasets include CICIoT2023, which reflects a distinct IoT device and attack profile [23], and N-BaIoT, which is specific to botnet network traffic signatures from actual Mirai and BASHLITE attacks. In the case where the feature rankings derived in the current research – specifically, the prominent ranking of the SYN Flag Count and Flow IAT Min – hold true in other datasets, the recommendations offered to SOC analysts will receive significantly more empirical support. Otherwise, the differences themselves will serve as an interesting scientific insight regarding the data dependence of XAI-based security recommendations.

Adversarial robustness of SHAP explanations. Section 1.6 of Chapter One identified adversarial manipulation of explanation methods as an active research concern: an adversary aware that a deployed model uses SHAP for post-hoc explanation may be able to craft inputs that produce misleading attributions while maintaining malicious intent. This vulnerability was not evaluated in the current study, and its investigation represents an important extension. Specifically, future work should examine whether adversarial perturbed IoT traffic flows crafted to evade detection by the Bagging or Stacking models also produce anomalous

SHAP attribution patterns that could alert analysts to evasion attempts, or whether the perturbations are sufficiently subtle to produce plausible-looking explanations that mislead rather than warn.

Online learning and concept drift detection [7]. In this thesis, all the models examined have been trained on a fixed dataset and tested under the assumption that there exists a stationary traffic distribution. However, the IoT network environments are subject to continuous concept drift caused by regular updates of device firmware, integration of new devices, and changing attacks techniques. Further research could address whether distributions of SHAP attributions can serve as an early warning sign for concept drift: if the set of attributes used by the model changes over time, say as the SYN Flag Count loses discriminative power due to the low-and-slow strategy used by attackers, then concept drift would manifest itself through such changes in SHAP attribute distributions before becoming apparent in the deterioration of model accuracy. This would turn the XAI approach into an online model monitoring system.

Exact explainability for neural networks. Given the constraints associated with DeepExplainer as explained in Section 4.1, further research on other neural network models that provide exact or better SHAP calculations is justified. Gradient methods like GradientSHAP improve upon DeepExplainer [10] in terms of approximation quality for specific neural network structures, whereas neural networks that explicitly use attention mechanisms like Transformers provide more interpretable internal representations and can serve as a supplementary method to SHAP post hoc explanation. An evaluation of whether an attention-enabled neural network can detect with equal accuracy as our deep neural network but with better SHAP calculation is pertinent in addressing the interpretation-precision trade-off discussed in Section 3.6.3.

Operational deployment evaluation. The experimental framework evaluated in this study was conducted under controlled laboratory conditions using a static balanced dataset. A natural and important extension is the deployment of the Bagging-based XAI framework — identified as the recommended architecture in Section 3.6.3 — in a live or simulated operational environment, where it would process real-time IoT traffic streams, generate SHAP explanations for analyst review, and be evaluated not only on detection metrics but on analyst workflow integration, explanation comprehension, and trust calibration. Such a study would directly address the human factors dimension of XAI identified in Section 1.6 — the risk that explanations, even correct ones, may induce overtrust or undertrust if not presented through appropriately designed interfaces — and would provide the kind of operational validation that purely algorithmic evaluations cannot supply.

References

[1] Cisco Systems. (2023). *Cisco Annual Internet Report 2022–2027*. Cisco White Paper.

[2] Apruzzese, G., et al. (2018). On the effectiveness of machine and deep learning for cyber security. *Proceedings of the 10th International Conference on Cyber Conflict (CyCon)*, 371–390.

[3] Buczak, A. L. & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2), 1153–1176.

[4] Gibert, D., Mateu, C. & Planes, J. (2020). The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. *Journal of Network and Computer Applications*, 153, 102526.

[5] Alahmadi, B. A., Axon, L. & Martinovic, I. (2022). 99% false positives: A qualitative study of SOC analysts' perspectives on security alarms. *Proceedings of the USENIX Security Symposium*, 2783–2800.

[6] Biggio, B. & Roli, F. (2018). Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84, 317–331.

[7] Jordaney, R., et al. (2017). Transcend: Detecting concept drift in malware classification models. *Proceedings of the USENIX Security Symposium*, 625–642.

[8] Arrieta, A. B., et al. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58, 82–115.

[9] Doshi-Velez, F. & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint*, arXiv:1702.08608.

[10] Lundberg, S. M. & Lee, S. I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30, 4765–4774.

[11] Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning*. MIT Press.

[12] Lapuschkin, S., et al. (2019). Unmasking Clever Hans predictors and assessing what machines really learn. *Nature Communications*, 10(1), 1096. **https://doi.org/10.1038/s41467-019-08987-4**

[13] Tavallaee, M., et al. (2009). A detailed analysis of the KDD CUP 99 data set. *Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 1–6.

[14] Slack, D., et al. (2020). Fooling LIME and SHAP: Adversarial attacks on post hoc explanation methods. *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 180–186.

[15] Ribeiro, M. T., Singh, S. & Guestrin, C. (2016). "Why should I trust you?" Explaining the predictions of any classifier. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135–1144.

[16] Rabanser, S., Günnemann, S. & Lipton, Z. (2019). Failing loudly: An empirical study of methods for detecting dataset shift. *Advances in Neural Information Processing Systems*, 32.

[17] Buccinca, Z., Maliya, M. B. & Gajos, K. Z. (2021). To trust or to think: Cognitive forcing functions can reduce overreliance on AI in AI-assisted decision making. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW1), 1–21.

[18] Rabbani, M., Gui, J., Nejati, F., Zhou, Z., Kaniyamattam, A., Mirani, M., Piya, G., Opushnyev, I., Lu, R. & Ghorbani, A. A. (2024). Device identification and anomaly detection in IoT environments. *IEEE Internet of Things Journal*. **https://doi.org/10.1109/JIOT.2024.10816028**

[19] Lashkari, A. H., et al. (2017). Characterization of tor traffic using time based features. *ICISSP*, 253–262.

[20] Sharafaldin, I., et al. (2018). Toward generating a new intrusion detection dataset. *ICISSP*, 108–116.

[21] Neto, E. C., et al. (2023). CICIoT2023: A real-time dataset for large-scale attacks in IoT. *Sensors*, 23(13), 5941.

[22] Koroniotis, N., et al. (2019). Towards the development of realistic botnet dataset. *Future Generation Computer Systems*, 100, 779–796.

[23] Rabbani, M., et al. (2024). CIC IoT-DIAD 2024: A Dataset for IoT Device Identification and Anomaly Detection.Canadian Institute for Cybersecurity, UNB.