Transferability and Explainability of Machine Learning Models for Network Intrusion Detection

Final Project Report

Author: Ahmed Bedair

Supervisor: Dr. Fabio Pierazzi

Student ID: K2105577

February 24, 2025

Abstract

This dissertation investigates the effectiveness of machine learning-based Network Intrusion Detection Systems (NIDS) in detecting cyber threats by analysing patterns in network traffic data. The study assesses the transferability of Random Forest and Support Vector Machine (SVM) classifiers across the CTU13 and CICIDS2017 datasets. Models trained on one dataset are tested on the other to evaluate how well learned patterns generalise to different network environments. Additionally, the SHAP (SHapley Additive exPlanations) technique is employed to identify critical features influencing the detection of malicious traffic, enhancing the transparency of the models' decision-making processes.

The novelty of this work lies in its comprehensive analysis of model transferability within NIDS and the application of explainable AI to improve interpretability. Findings reveal significant challenges arising from dataset biases and the limitations of directly applying models across datasets. To address these, the study proposes strategies to enhance detection accuracy and model interpretability, strengthening NIDS robustness against evolving cyber threats. This research advances cybersecurity by providing insights into the adaptability of machine learning models and the key factors impacting NIDS performance across diverse network settings.

Originality Avowal

I verify that I am the sole author of this report, except where explicitly stated to the contrary. I grant the right to King's College London to make paper and electronic copies of the submitted work for purposes of marking, plagiarism detection and archival, and to upload a copy of the work to Turnitin or another trusted plagiarism detection service. I confirm this report does not exceed 25,000 words.

Ahmed Bedair

February 24, 2025

Acknowledgements

I extend my sincere gratitude to my supervisor, Dr Fabio Pierazzi, for his invaluable guidance and unwavering support throughout this project. His expertise and encouragement were instrumental in keeping me focused and enabling me to complete this work successfully.

I would also like to express my heartfelt thanks to my family and friends for their constant encouragement and belief in me. Their support has been a source of strength and motivation.

Contents

Chapter 1

Introduction

In an increasingly interconnected digital landscape, safeguarding computer network security is of paramount importance. The relentless evolution of cyber threats challenges the effectiveness of conventional signature-based Network Intrusion Detection Systems (NIDS), which often fail to detect novel and sophisticated attacks [18]. Consequently, the research community has shifted towards machine learning approaches that identify unseen intrusions by analysing network traffic for patterns and anomalies [7].

The efficacy of machine learning-driven NIDS depends significantly on the authenticity and comprehensiveness of the datasets employed for training and evaluation. The CTU13 and CICIDS2017 datasets stand out as prominent examples, providing genuine botnet traffic within regular network flows and a diverse array of contemporary attack vectors, respectively [11, 24]. Despite their extensive use, a critical research gap persists: understanding how machine learning models can generalise patterns learned from one dataset to accurately detect intrusions in another remains underexplored.

This dissertation investigates the effectiveness of machine learning algorithms—specifically Random Forest and Support Vector Machine (SVM) classifiers—trained on the CICIDS2017 dataset, in identifying botnet activities within the CTU13 dataset. By evaluating their performance on CTU13, we assess the transferability and applicability of learned features across distinct datasets. This approach mirrors real-world scenarios where models trained on specific datasets must protect diverse network environments against intrusions.

Following Arp et al.'s guidelines for applying machine learning in cybersecurity [4], this study integrates explainable AI techniques, notably SHAP (SHapley Additive exPlanations) [15], to clarify the decision-making processes of our models. By pinpointing critical features for botnet detection, we reveal the distinctive patterns that differentiate malicious from benign network flows, enhancing interpretability and trust in the results.

Additionally, this dissertation conducts a comparative analysis of network flow features from the CTU13 and CICIDS2017 datasets. Examining statistical properties—such as flow duration, packet counts, and inter-arrival times—we highlight differences between real and synthetic traffic. This analysis underscores the limitations of synthetic datasets in training NIDS models and emphasises the importance of addressing dataset biases in performance evaluations [25].

Through the use of Random Forest and SVM classifiers, this study provides a thorough evaluation of model transferability and generalisability. Performance comparisons, coupled with SHAP-based interpretation, offer valuable insights into the robustness and versatility of these algorithms in detecting botnet attacks across datasets, advancing their potential application in cybersecurity.

This dissertation is structured as follows: Section 2 introduces the CTU13 and CICIDS2017 datasets, machine learning classifiers, and explainable AI methods. Section 3 reviews existing literature on machine learning-based NIDS. Section 4 details the experimental framework's design, while Section 5 describes its implementation. Section 6 presents the evaluation results and their implications. Section 7 addresses the legal, social, ethical, and professional considerations of deploying machine learning in intrusion detection. Finally, Section 8 concludes with a summary of findings and recommendations for future research.

Chapter 2

Background

This chapter provides essential context for understanding the datasets, machine learning classifiers, and explainable AI techniques used in this dissertation. Section 2.1 overviews two benchmark datasets in network intrusion detection: CTU13 [11] and CICIDS2017 [24]. These datasets are widely employed to develop and evaluate machine learning-based Network Intrusion Detection Systems (NIDS). Understanding their characteristics, strengths, and limitations is crucial for designing robust and effective NIDS.

Section 2.2 discusses two prominent machine learning classifiers—Random Forest and Support Vector Machine (SVM)—commonly used in NIDS. Their strengths, limitations, and performance across different datasets are examined, highlighting the importance of selecting appropriate models for intrusion detection tasks.

Section 2.3 introduces explainable AI techniques, focusing on SHAP (SHapley Additive exPlanations) [15], which provides insights into the decision-making processes of machine learning models. These techniques are vital for validating model reliability and understanding the features that influence predictions of benign or malicious network flows.

2.1 CTU13 and CICIDS2017 Datasets

Representative and labelled datasets are fundamental for developing and evaluating machine learning-based NIDS. The CTU13 [11] and CICIDS2017 [24] datasets are two widely used benchmarks in this domain.

The CTU13 dataset, introduced by Garcia et al. [11], comprises real botnet traffic captured in a controlled environment. It includes 13 scenarios, each representing distinct botnet behaviours such as port scanning, DDoS attacks, click fraud, and spam. The dataset's creation involved a detailed methodology, including the use of actual botnet samples and a labelling process based on known botnet behaviour. Its realistic nature and diverse botnet scenarios have made it a popular choice among researchers. Table 2.1 presents the class distribution in CTU13.

Class	**Count**
Benign	213,326
Botnet	15,559

Table 2.1: CTU13 Dataset Class Breakdown

Sharafaldin et al. [24] developed the CICIDS2017 dataset, a more recent and comprehensive resource for evaluating NIDS. It encompasses a variety of modern attacks, including DoS, DDoS, brute force, XSS, SQL injection, and infiltration. The dataset was generated in a controlled lab environment that mimics real-world network infrastructure, using tools and scripts to create realistic benign and attack traffic. It features both manually and time-based labelled data, as shown in Table 2.2.

Class	**Count**
Benign	908,528
Botnet	745
DDoS	51,234
DoS GoldenEye	4,189
DoS Hulk	91,856
DoS Slowhttptest	2,191
DoS slowloris	2,355
FTP-Patator	3,184
Heartbleed	6
Infiltration	18
PortScan	63,633
SSH-Patator	2,342
Web Attack Brute Force	601
Web Attack SQL Injection	9
Web Attack XSS	260

Table 2.2: CICIDS2017 Dataset Class Breakdown

Comparing the datasets, botnet attacks in CTU13 share structural and behavioural similarities with several attack types in CICIDS2017, such as DDoS/DoS, web attacks, and bot attacks. These similarities suggest that machine learning models trained on CICIDS2017 may be transferable to detecting botnet attacks in CTU13. However, both datasets have limitations: synthetic datasets like CICIDS2017 may not fully capture the complexity and diversity of real-

world traffic, lacking crucial contextual information. More representative datasets that reflect operational and deployment challenges are needed to fully assess machine learning-based NIDS.

2.2 Machine Learning Classifiers

Machine learning classifiers are integral to NIDS, automating the distinction between malicious and benign traffic. Random Forest (RF) and Support Vector Machine (SVM) are two widely adopted classifiers in this domain due to their effectiveness in handling complex, high-dimensional data.

Random Forest, an ensemble method, combines multiple decision trees to produce a majority-vote classification [12]. Its strengths include handling high-dimensional data, robustness to noise and outliers, and the ability to model complex feature interactions—key advantages for detecting diverse network intrusions.

Support Vector Machine (SVM) identifies the optimal hyperplane to separate classes in high-dimensional space [9]. It excels in scenarios with limited labelled data, a common challenge in NIDS, and can manage both linear and non-linear classification tasks via kernel functions. SVM's strong generalisation capabilities make it suitable for detecting novel attacks.

The selection of a classifier depends on factors such as dataset characteristics, attack types, and computational resources. Evaluating classifiers using metrics like accuracy, precision, and recall is essential for choosing the most effective model for intrusion detection.

2.3 Explainable AI Techniques

Despite their success, machine learning classifiers often operate as 'black boxes,' offering limited insight into their decision-making processes. Explainable AI techniques address this by providing interpretable explanations for model predictions.

SHAP (SHapley Additive exPlanations) [15] is a leading technique for model interpretation, rooted in cooperative game theory. It assigns importance scores to each feature, quantifying their contribution to a model's prediction for a given instance. In the context of NIDS, SHAP helps identify key features that influence the detection of specific attacks, enhancing transparency and trust in the model's outputs.

Insights from SHAP can validate model reliability, uncover dataset biases, and inform feature engineering. For network security practitioners, these explanations are invaluable for understanding the factors driving attack detection and refining defence strategies.

Chapter 3

Relevant Work

This chapter provides a comprehensive review of existing research in machine learning-based network intrusion detection. It covers state-of-the-art approaches for botnet detection using the CTU13 dataset, intrusion detection using the CICIDS2017 dataset, and the application of Random Forest and Support Vector Machines (SVM) for intrusion detection tasks. Additionally, it explores the use of explainable AI techniques, specifically SHAP, to interpret machine learning models in the context of intrusion detection. The chapter also addresses the challenges and limitations of synthetic datasets and the dynamic nature of network traffic, emphasising the importance of considering dataset biases and representativeness. Throughout, we highlight how this dissertation's focus on **transferability**—the ability of models trained on one dataset to perform well on another—distinguishes it from prior studies.

3.1 Botnet Detection using CTU13 Dataset

Several studies have utilised the CTU13 dataset to develop and evaluate botnet detection systems. Chowdhury et al. [8] proposed a graph-based approach, constructing a graph representation of communication patterns among botnet-infected hosts and applying graph analysis techniques to identify botnets. They extracted features such as in-degree, out-degree, and betweenness centrality and used them to train a Random Forest classifier, achieving high detection accuracy. Their work demonstrates the potential of graph-based features for botnet detection.

Pektaş and Acarman [20] applied deep learning techniques to the CTU13 dataset. They used a

convolutional neural network (CNN) to learn discriminative features from raw network traffic data, converting the traffic into greyscale images for input. Their CNN-based model achieved high accuracy, showcasing the effectiveness of deep learning in capturing complex botnet patterns.

While these studies focus on developing models tailored to the CTU13 dataset, this dissertation investigates the less explored area of **transferability**. Specifically, it assesses whether models trained on the CICIDS2017 dataset can detect botnets in CTU13, contributing to understanding the adaptability of machine learning models across diverse network environments. This focus on cross-dataset performance distinguishes our work from previous CTU13-specific studies.

Having reviewed botnet detection on CTU13, we now turn to intrusion detection research using the CICIDS2017 dataset, which offers a broader range of attack types.

3.2 Intrusion Detection using CICIDS2017 Dataset

The CICIDS2017 dataset has been widely used to evaluate intrusion detection systems. Ustebay et al. [27] proposed a multi-layer perceptron (MLP)-based system, performing extensive preprocessing and using recursive feature elimination with Random Forest for feature selection. Their approach demonstrated the potential of neural networks for intrusion detection.

Aksu and Aydin [2] conducted a comparative study of machine learning algorithms on CICIDS2017, evaluating decision trees, Random Forests, and SVMs. They found that Random Forests outperformed other algorithms, highlighting the effectiveness of ensemble methods.

This dissertation leverages CICIDS2017 to train classifiers and investigate their **transferability** to detect botnet attacks in CTU13. Unlike previous studies that focus on performance within CICIDS2017, our work explores the adaptability of models to a different dataset, providing insights into cross-dataset generalisation.

The following sections focus on Random Forest and SVM, two algorithms central to this dissertation's investigation of transferability.

3.3 Random Forest for Intrusion Detection

Random Forest has been widely applied to intrusion detection. Farnaaz and Jabbar [10] used Random Forest on the NSL-KDD dataset, employing feature selection via the Chi-square test. Their model demonstrated Random Forest's effectiveness in detecting various network attacks.

Belouch et al. [5] applied Random Forest to CICIDS2017, comparing it with other algorithms like Decision Tree and Naive Bayes. Their results confirmed Random Forest's superior performance, further validating its use in intrusion detection.

While these studies demonstrate Random Forest's effectiveness within single datasets, this dissertation explores its **transferability** by training on CICIDS2017 and testing on CTU13. This cross-dataset approach provides new insights into the algorithm's adaptability.

Similarly, the next section examines SVM's role in intrusion detection and its potential for transferability.

3.4 Support Vector Machines for Intrusion Detection

SVM has been extensively used for intrusion detection. Kabir et al. [13] proposed an SVM-based system on the NSL-KDD dataset, using a genetic algorithm for feature selection and grid search for parameter optimisation. Their work highlighted SVM's effectiveness in detecting diverse attacks.

Teng et al. [26] applied SVM with various kernel functions to CICIDS2017, emphasising the importance of kernel selection and parameter tuning for optimal performance.

In contrast to these studies, this dissertation focuses on the **transferability** of SVM models trained on CICIDS2017 to detect botnet attacks in CTU13. By investigating cross-dataset performance, we extend existing research on SVM's generalisation capabilities.

While Random Forest and SVM are effective for intrusion detection, understanding their decision-making processes is crucial for assessing transferability. The next section addresses this through explainable AI techniques.

3.5 Explainable AI Techniques for Intrusion Detection

Explainable AI techniques have gained attention for providing insights into machine learning models' decision-making. Warnecke et al. [28] evaluated explanation methods, including SHAP, for deep learning-based intrusion detection systems. They applied SHAP to a CNN trained on NSL-KDD, demonstrating its ability to identify influential features for specific attack types.

Amarasinghe et al. [3] used SHAP to interpret a deep neural network's predictions on NSL-KDD, highlighting its potential for explaining complex models.

Mane and Rao [17] applied SHAP to a Random Forest classifier on NSL-KDD, visualising feature contributions to understand the model's decisions.

While these studies use SHAP to interpret models within single datasets, this dissertation extends SHAP's application to analyse **cross-dataset transferability**. By employing SHAP to interpret Random Forest and SVM classifiers trained on CICIDS2017 and tested on CTU13, we provide novel insights into why models succeed or fail in new environments. This approach enhances the interpretability of transferability, revealing dataset-specific biases and feature importance across different network settings.

Having reviewed algorithmic and explainable AI approaches, we now discuss the challenges and limitations that motivate this dissertation's focus on transferability.

3.6 Challenges and Limitations

Despite the frequent use of CTU13 and CICIDS2017, their limitations must be acknowledged. Sommer and Paxson [25] critiqued synthetic datasets for failing to capture real-world network traffic's complexity and lacking contextual information. They emphasised the need for datasets that reflect operational challenges in intrusion detection.

This dissertation addresses these limitations by investigating **transferability** across datasets, providing insights into model performance in diverse environments. Additionally, by using SHAP, we mitigate dataset biases by identifying features that are consistently important across datasets.

Moreover, Sommer and Paxson [25] and Buczak and Guven [7] highlighted the challenges posed by evolving attack patterns and dynamic network traffic. To address this, our work focuses on

transferability, assessing whether models trained on one dataset can adapt to new attack types in another, contributing to more robust intrusion detection systems.

These challenges underscore the importance of this dissertation's focus on transferability and explainable AI, which we summarise in the next section.

3.7 Summary and Positioning

This chapter has reviewed key studies in machine learning-based network intrusion detection, covering botnet detection (CTU13), intrusion detection (CICIDS2017), and the use of Random Forest, SVM, and SHAP. While prior research demonstrates the effectiveness of these techniques within single datasets, this dissertation investigates a novel aspect: the **transferability** of models across datasets.

Specifically, this work addresses two primary questions: 1. How effectively can Random Forest and SVM classifiers trained on CICIDS2017 detect botnet attacks in CTU13? 2. What insights can SHAP provide into the transferability of these models across datasets?

By focusing on cross-dataset performance and leveraging SHAP to interpret feature importance, this dissertation extends existing research. It provides valuable insights into the robustness and adaptability of machine learning models in diverse network environments, contributing to the development of practical intrusion detection solutions.

In summary, this chapter establishes the foundation for our investigation of transferability and explainable AI in network intrusion detection, emphasising the importance of dataset representativeness and model interpretability in addressing real-world cybersecurity challenges.

Chapter 4

Specification & Design

This chapter outlines the experimental design of this dissertation, which focuses on three classifiers: the Dummy Classifier, the Random Forest Classifier, and the Support Vector Machine (SVM) Classifier. Informed by a thorough literature review, the experiments address the following research questions:

RQ1 How well do machine learning models trained on one dataset transfer and generalise to another in the context of network intrusion detection?

RQ2 What is the impact of dataset biases and representativeness on the performance of machine learning-based intrusion detection systems?

RQ3 How can explainable AI techniques, such as SHAP, provide insights into the transferability of learned patterns and the most relevant features for detecting specific attack types across datasets?

The experiments utilise the CTU13 and CICIDS2017 datasets, selected for their distinct properties and attack types, making them ideal for assessing transferability (RQ1). CTU13 contains real botnet traffic [11], while CICIDS2017 offers a comprehensive range of modern attacks [24]. Models are trained on CICIDS2017 and tested on CTU13 to evaluate cross-dataset performance. Detailed characteristics of these widely used datasets are provided in Chapter 2.

Pre-processing, implemented via Algorithms 2 and 1, standardises feature names and removes dataset-specific features to ensure consistency and mitigate biases (RQ2), aligning with best

practices from Chapter 3. The classifiers are evaluated using accuracy, recall, precision, F1 score, and confusion matrices, offering a robust assessment of performance and transferability. SHAP (SHapley Additive exPlanations) [15] is employed to interpret predictions and identify key features across datasets (RQ3), enhancing explainability as reviewed in Chapter 3, Section 3.5.

A 60/40 training-testing split balances model training and evaluation, following recommendations by Buczak et al. [7]. Although neural networks were considered, Random Forest and SVM were chosen for their interpretability and effectiveness with imbalanced, high-dimensional data [10, 26].

4.1 Experiments

The experiments involve training and testing the Dummy, Random Forest, and SVM Classifiers on both CICIDS2017 and CTU13 datasets. This setup assesses within-dataset performance and cross-dataset transferability (RQ1). The Dummy Classifier acts as a baseline, while Random Forest and SVM are selected for their ability to handle large, imbalanced datasets, as detailed in Chapter 2, Section 2.2.

4.1.1 Data Pre-processing

Pre-processing ensures compatibility by standardising feature names and removing unique features (Algorithms 2 and 1). CICIDS2017 supports binary and multi-class tasks, while CTU13 is processed for binary classification (botnet vs. benign), aligning with the focus on botnet detection.

4.1.2 Training and Testing Split

A 60/40 split is applied to each dataset. Classifiers are trained on the CICIDS2017 training set and tested on both its testing set and the full CTU13 dataset to evaluate transferability (RQ1).

4.1.3 Evaluation Metrics

Performance is assessed using:

- **Accuracy**: Proportion of correct predictions.

- **Recall**: Ability to detect true positives.
- **Precision**: Accuracy of positive predictions.
- **F1 Score**: Harmonic mean of precision and recall.
- **Confusion Matrix**: Detailed classification breakdown.

These metrics, informed by Chapter 3, evaluate both within-dataset and cross-dataset performance.

4.1.4 Explainable AI

SHAP interprets classifier predictions, identifying critical features for attack detection across datasets (RQ3). This builds on prior work in explainable AI for cybersecurity (Chapter 3, Section 3.5).

4.2 Classifier Configurations

4.2.1 Dummy Classifier

The Dummy Classifier provides a baseline by randomly assigning labels based on class distribution (Figure 4.1).

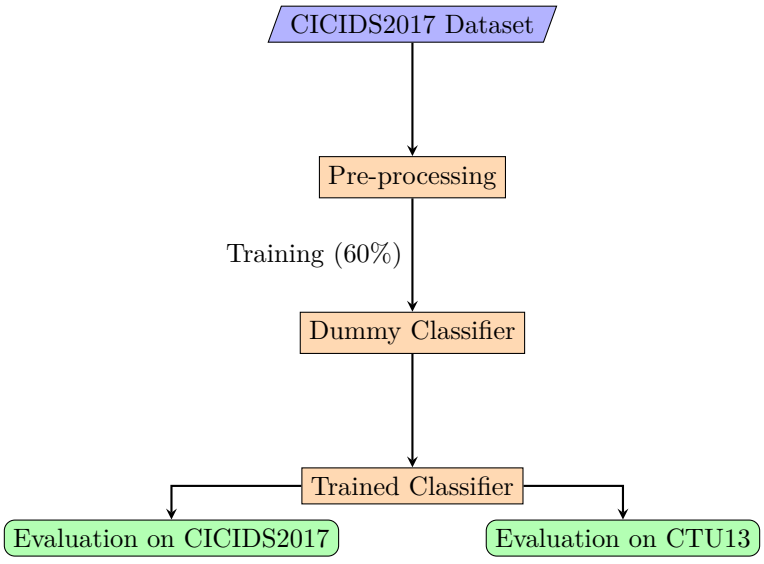

Figure 4.1: Dummy Classifier Configuration

4.2.2 Random Forest and SVM Classifiers

Random Forest and SVM are trained on CICIDS2017 and tested on both datasets, with SHAP enhancing interpretability (Figure 4.2).

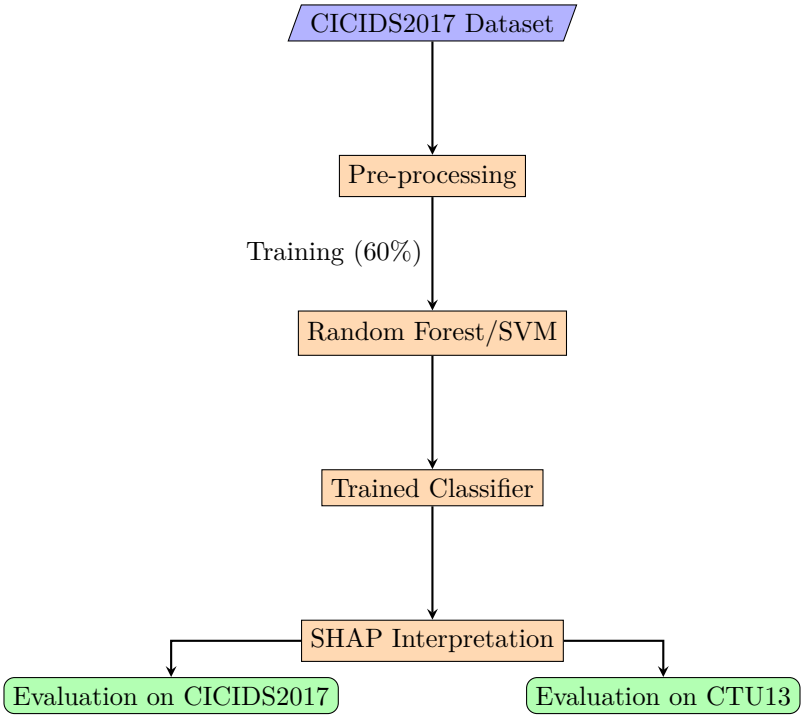

Figure 4.2: Random Forest and SVM Classifier Configuration

4.3 Transferability Evaluation

Transferability is evaluated by training classifiers on CICIDS2017 and testing on CTU13, using the metrics from Section 4.1.3. High performance across both datasets indicates strong transferability, while a decline suggests limitations (RQ1). SHAP values reveal feature importance consistency, addressing dataset biases (RQ2) and explainability (RQ3), and reflecting real-world adaptability as noted in Chapter 1.

4.4 Summary

This chapter details the experimental design for assessing classifier transferability in network intrusion detection. The Dummy, Random Forest, and SVM Classifiers are trained on CICIDS2017 and tested on both CICIDS2017 and CTU13, with pre-processing ensuring compatibility. Performance metrics and SHAP insights address RQ1–RQ3, providing a foundation for

the implementation in Chapter 5.

Chapter 5

Implementation

This chapter provides a comprehensive overview of the experiments conducted in this dissertation, detailing the experimental setup, dataset usage, and the design of each experiment. We also explore the implementation of key package modules, accompanied by relevant code excerpts highlighting essential functionality.

5.1 Experiment Implementation

The experiments in this dissertation involve training and evaluating three classifiers: a Dummy Classifier, a Random Forest Classifier, and a Support Vector Machine (SVM) Classifier. Each classifier is trained on both the CICIDS2017 [24] and CTU13 [11] datasets and tested on both the CICIDS2017 and CTU13 datasets to assess transferability and generalisation capabilities.

Initially, we set up the experiments using scikit-learn's classifier implementations. However, due to the extensive training times required for the CPU-based models, we decided to switch to CUML's GPU-accelerated models [21]. The CUML library provides GPU-accelerated machine learning algorithms, enabling faster training and inference times. By leveraging the power of GPUs, we can significantly accelerate the training process, making it more feasible to train complex models on large datasets. The benefits of using CUML's GPU-accelerated models include reduced training times, improved scalability, and the ability to handle larger datasets efficiently.

The choice of Random Forest and SVM classifiers is justified by their proven effectiveness in

handling large, high-dimensional datasets and unbalanced class distributions [10, 26], which are common in network traffic classification, as discussed in Chapter 2, Section 2.2. The CTU13 and CICIDS2017 datasets exhibit such class imbalances (Tables 2.1 and 2.2), making these classifiers well-suited for this study. We employ the SHAP library [15] for explainability due to its ability to provide insights into the decision-making process of machine learning models and identify essential features, as highlighted in Chapter 2, Section 2.3.

5.1.1 Data Pre-processing

Proper data preprocessing is crucial for ensuring the effectiveness of the trained classifiers. The relabelling scripts, `relabelCICIDS2017.py` (Algorithm 2) and `relabelCTU13.py` (Algorithm 1), standardise feature names and class labels across the datasets. This process involves mapping dataset features to a consistent naming convention, removing features unique to one dataset to avoid overfitting, and ensuring class labels are compatible, as described in Chapter 4, Section 4.1.1. For example, the CTU13 dataset's binary labels '0' and '1' are converted to 'Benign' and 'Botnet' to match the CICIDS2017 labelling scheme.

During the preprocessing stage, several data quality issues and inconsistencies were encountered, such as missing values and differing feature names across datasets. We addressed these issues through careful data cleaning, imputation, and feature mapping techniques to ensure the datasets' compatibility and integrity, aligning with the challenges discussed in Chapter 3, Section 3.6.

5.1.2 Experiment 1: Dummy Classifier

The Dummy Classifier from the scikit-learn library [19] serves as a baseline for evaluating the performance of the more advanced classifiers. It predicts the most frequent class in the training data. We trained three Dummy Classifiers (Algorithm 3): one on the CTU13 dataset for binary classification and two on the CICIDS2017 dataset for binary and multiclass classification. The Dummy Classifiers use the same features as the Random Forest and SVM classifiers, and their performance metrics (accuracy, precision, recall, and F1 score) provide a baseline for comparison, as discussed in Chapter 4, Section 4.2.1.

5.1.3 Experiment 2: Random Forest Classifier

Random Forest, an ensemble learning method constructing multiple decision trees [12], is well-suited for handling large, high-dimensional datasets and unbalanced class distributions [10], which are common in network traffic classification. The CTU13 and CICIDS2017 datasets exhibit such class imbalances (Tables 2.1 and 2.2), as discussed in Chapter 2, Section 2.1.

Three Random Forest Classifiers were trained (`trainRandomForest.ipynb`, Algorithm 4): one on the CTU13 dataset and two on the CICIDS2017 dataset for binary and multiclass classification. The classifiers are evaluated on their respective datasets and then tested on the other dataset to assess transferability, addressing RQ1, as discussed in Chapter 4, Section 4.2.2. The SHapley Additive exPlanations (SHAP) library [15] is employed to explain the classifiers' predictions and identify the essential features, contributing to the interpretability and transparency of the models, as highlighted in Chapter 3, Section 3.5.

5.1.4 Experiment 3: Support Vector Machine Classifier

Support Vector Machines (SVMs) effectively handle large, high-dimensional, non-linear data [9, 22]. They have been successfully applied to network intrusion detection tasks [14, 26], as discussed in Chapter 3, Section 3.4.

The experimental setup for the SVM Classifiers (`trainSVM.ipynb`, Algorithm 5) mirrors that of the Random Forest Classifiers: we trained three SVMs on the CTU13 and CICIDS2017 datasets, which are then evaluated on their respective datasets, and tested on the other dataset, addressing the transferability and generalisability aspects of RQ1, as discussed in Chapter 4, Section 4.2.2. SHAP is used to interpret the SVM predictions and identify essential features, contributing to the understanding of the transferability of learned patterns and the key characteristics distinguishing malicious and benign traffic, as highlighted in Chapter 3, Section 3.5.

5.1.5 Testing and Evaluation

A comprehensive testing and evaluation strategy is employed to ensure the robustness and reliability of the results. We assessed the performance of the classifiers using various metrics, including accuracy, precision, recall, and F1 score, as discussed in Chapter 4, Section 4.1.3. We calculated these metrics for each classifier on their respective test sets and when evaluated on the other dataset for transferability analysis, addressing RQ1 and RQ3, as discussed in Chapter 4, Section 4.3.

5.1.6 Novelty and Originality

The novelty and originality of this research lie in the combination of the chosen datasets (CTU13 and CICIDS2017), classifiers (Random Forest and SVM), and the use of SHAP for explainability. This dissertation provides a unique perspective on network intrusion detection by investigating the transferability and generalisation of machine learning models across different datasets, as discussed in Chapter 3, Section 3.7.

The application of SHAP to interpret the predictions of the trained classifiers and identify the most relevant features for detecting specific types of attacks across datasets is a novel approach. This aspect of the research contributes to a deeper understanding of the transferability of learned patterns and the key characteristics distinguishing malicious and benign traffic, as highlighted in Chapter 3, Section 3.5.

5.1.7 Strengths and Limitations

The chosen methodology has several strengths. Random Forest and SVM classifiers are well-suited for handling the imbalanced and high-dimensional nature of the CTU13 and CICIDS2017 datasets. These classifiers have demonstrated their effectiveness in network intrusion detection tasks [10, 26], as discussed in Chapter 3, Sections 3.3 and 3.4. Using CUML's GPU-accelerated models significantly reduces training times and improves scalability, enabling the efficient handling of large datasets, as discussed in Section 5.1.

However, there are also limitations to consider. The potential impact of dataset biases and the training data's representativeness on the models' transferability is a concern, as discussed in Chapter 3, Section 3.6. The computational complexity of the SHAP explanations may also pose challenges when dealing with large-scale datasets, even with GPU acceleration, as highlighted in Chapter 8.

5.2 Package Implementation

To support the experiments described above, a set of software packages was developed for data preprocessing, model training, and result visualisation. The following subsections detail each of these packages.

5.2.1 relabelCTU13.py

Purpose: This Python script (Algorithm 1) is tasked with preprocessing the CTU13 dataset, ensuring its compatibility with the analysis framework. It specifically focuses on feature normalisation, class relabeling, and reordering of features to align with the structure of the CICIDS2017 dataset.

Functionality:

- *Feature Renaming*: Aligns the CTU13 dataset's feature names with those of CICIDS2017, facilitating direct comparison and joint analysis.
- *Traffic Class Relabeling*: Converts traffic classification labels to a unified schema shared with CICIDS2017, enabling consistent interpretation of traffic types across datasets.
- *Feature Reordering*: Adjusts the order of features in the CTU13 dataset to match that of CICIDS2017, ensuring that subsequent analysis scripts operate correctly without needing dataset-specific adjustments.

5.2.2 relabelCICIDS2017.py

Purpose: Similar to `relabelCTU13.py`, this script (Algorithm 2) prepares the CICIDS2017 dataset for analysis by renaming features, relabeling traffic classes, and reordering features. The adjustments ensure that the CICIDS2017 dataset's structure is compatible with CTU13, facilitating combined analyses.

Functionality:

- *Mapping Feature Names*: Transforms the feature names in CICIDS2017 to align with CTU13's nomenclature, ensuring consistency in feature interpretation.
- *Traffic Class Relabeling and Identification*: Updates the traffic class labels for compatibility and identifies common features between the datasets to focus on comparable data points.
- *Feature Reordering*: Modifies the feature order in CICIDS2017 to conform to CTU13's layout, simplifying cross-dataset analyses.

5.2.3 trainDummyClassifier.ipynb

Purpose: A Jupyter Notebook (Algorithm 3) dedicated to training baseline Dummy Classifiers on the CTU13 and CICIDS2017 datasets. It sets a foundational performance benchmark for comparing more sophisticated Random Forest and SVM classifiers.

Functionality:

- *Training Process*: Outlines the steps for training Dummy Classifiers, including data loading, preprocessing application, and classifier training.
- *Performance Evaluation*: Details the evaluation metrics used to assess the classifiers, providing a baseline for the effectiveness of subsequent, more complex models.

5.2.4 trainRandomForest.ipynb

Purpose: Trains and evaluates Random Forest classifiers (Algorithm 4) on both datasets using GPU-accelerated implementations. It explores the classifiers' transferability and employs the SHAP library for result interpretation.

Functionality:

- *GPU-Accelerated Training*: Leverages CUML's GPU-accelerated Random Forest implementation for efficient model training and evaluation.
- *Transferability Testing*: Assesses the model's performance on both the training and alternative datasets to examine transferability.
- *Feature Importance Analysis*: Utilises SHAP values to interpret the model's predictions and identify significant features, enhancing model transparency and understanding.

5.2.5 trainSVM.ipynb

Purpose: Similar to the Random Forest notebook, this Jupyter Notebook trains SVM classifiers (Algorithm 5) on the datasets, tests their transferability, and applies SHAP for interpretability.

Functionality:

- *GPU-Accelerated SVM Training*: Implements CUML's SVM for rapid model training, facilitating the handling of large datasets.

- *Cross-Dataset Performance Evaluation*: Evaluates SVM classifiers' performance across different datasets to test model generalisability.

- *SHAP-Based Explanation*: Applies SHAP to elucidate SVM classifiers' decision-making, spotlighting critical features that influence predictions.

5.2.6 plotData.ipynb

Purpose: This script (Algorithm 6) provides visualisations of dataset characteristics and model performance metrics, supporting the analysis and discussion of the experimental results presented in Chapter 6.

Functionality:

- *Dataset Visualisation*: Generates descriptive statistics and visualisations of the CTU13 and CICIDS2017 datasets, providing insights into class distributions, feature distributions, and other relevant characteristics.

- *Performance Metric Visualisation*: Creates visualisations of the performance metrics obtained from the experiments, including accuracy, precision, recall, and F1 score, facilitating the comparison of different classifiers and their transferability across datasets.

- *SHAP Value Visualisation*: Visualises the SHAP values obtained from the explainable AI analysis, highlighting the most important features contributing to the classifiers' predictions and their transferability across datasets.

The implementation of these software packages and the experimental setup and evaluation strategies align with the research objectives and methodology outlined in Chapter 4. The preprocessing scripts ensure the consistency and compatibility of the datasets, while the training modules and evaluation strategies address the research questions related to model transferability, dataset biases, and the insights provided by explainable AI techniques.

The use of GPU-accelerated implementations and the SHAP library for explainability reflects the state-of-the-art machine learning and cybersecurity research practices, as discussed in Chapter 3. The visualisations generated by the `plotData.ipynb` script support the analysis and

interpretation of the experimental results, contributing to understanding the transferability of learned patterns and the key characteristics distinguishing malicious and benign traffic across different datasets.

Chapter 6

Evaluation

This chapter presents the findings and results from the experiments outlined in the specification and design chapter (4) as well as the implementation chapter (5). We evaluate the classifiers' performance, first on their respective datasets and then on a different dataset to assess transferability. Additionally, we explore the SHAP (SHapley Additive exPlanations) values for each experiment to reason about the importance of features in the models.

6.1 Performance Evaluation

As described in section (4.3), the classifiers are evaluated on their respective datasets and subsequently on another dataset to assess transferability. The performance metrics assessed include accuracy, precision, recall, and F1 score, as outlined in Chapter 4, Section 4.1.3. Results are illustrated in Figures 6.1 and 6.4.

6.1.1 Performance on the Same Dataset

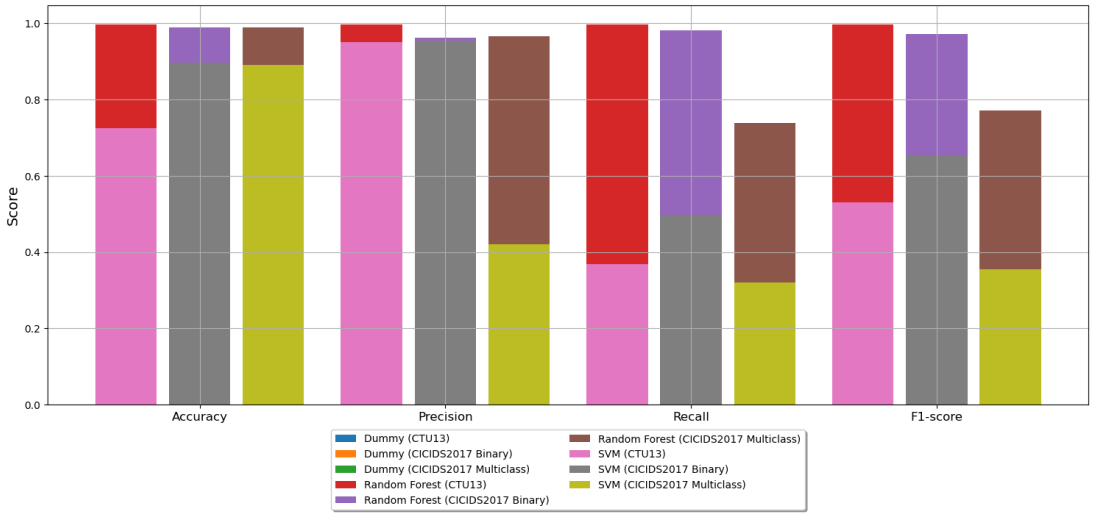

Figure 6.1: Classifiers' performance on the same datasets.

When testing the classifiers' performance on the datasets they were trained on, we observed promising results. The Random Forest model achieved high performance across all metrics on both the CTU13 and CICIDS2017 datasets for binary classification. It recorded an accuracy, precision, recall, and F1 score of 0.99 on CTU13, with comparable scores on CICIDS2017 (accuracy=0.99, precision=0.96, recall=0.98, F1=0.97). These results suggest that the Random Forest classifiers effectively learned patterns to distinguish between benign and malicious traffic, aligning with prior studies on their efficacy in network attack detection, as discussed in Chapter 3, Section 3.3.

For multi-class classification on CICIDS2017, the Random Forest model performed respectably, achieving an F1 score of 0.78 (accuracy=0.99, precision=0.90, recall=0.76). Although there is scope for improvement to achieve deployment-ready performance, the model demonstrated clear learning compared to a dummy classifier baseline of 0.66. Overall, Random Forest classifiers consistently outperformed dummy baselines on their training datasets, underscoring their ability to identify discriminative patterns, as noted in Chapter 2, Section 2.2.

In contrast, the SVM classifiers underperformed relative to both Random Forest models and dummy classifiers when evaluated on their training datasets. The binary SVM on CTU13 achieved an accuracy of 0.73, precision of 0.96, recall of 0.37, and F1 score of 0.53. Similar results were observed on CICIDS2017 (accuracy=0.73, precision=0.96, recall=0.37, F1=0.65). The multi-class SVM on CICIDS2017 recorded an accuracy of 0.89 but exhibited low preci-

sion (0.42), recall (0.32), and F1 score (0.35). These findings indicate that the current SVM implementations struggled to learn distinguishing patterns, even on their training datasets, contrasting with prior studies highlighting SVM effectiveness in network intrusion detection, as discussed in Chapter 3, Section 3.4. Improvements are likely required, as explored in Chapter 8.

6.1.2 Feature Importance Analysis on Same Dataset

To gain insight into how Random Forest models make predictions, we analyse SHAP values. SHAP, an explainable AI technique, assigns importance scores to features for each prediction, elucidating the model's decision-making process, as detailed in Chapter 2, Section 2.3. The appendix listing B provides guidance on interpreting the SHAP value graphs in this section.

Random Forest Model on CTU13

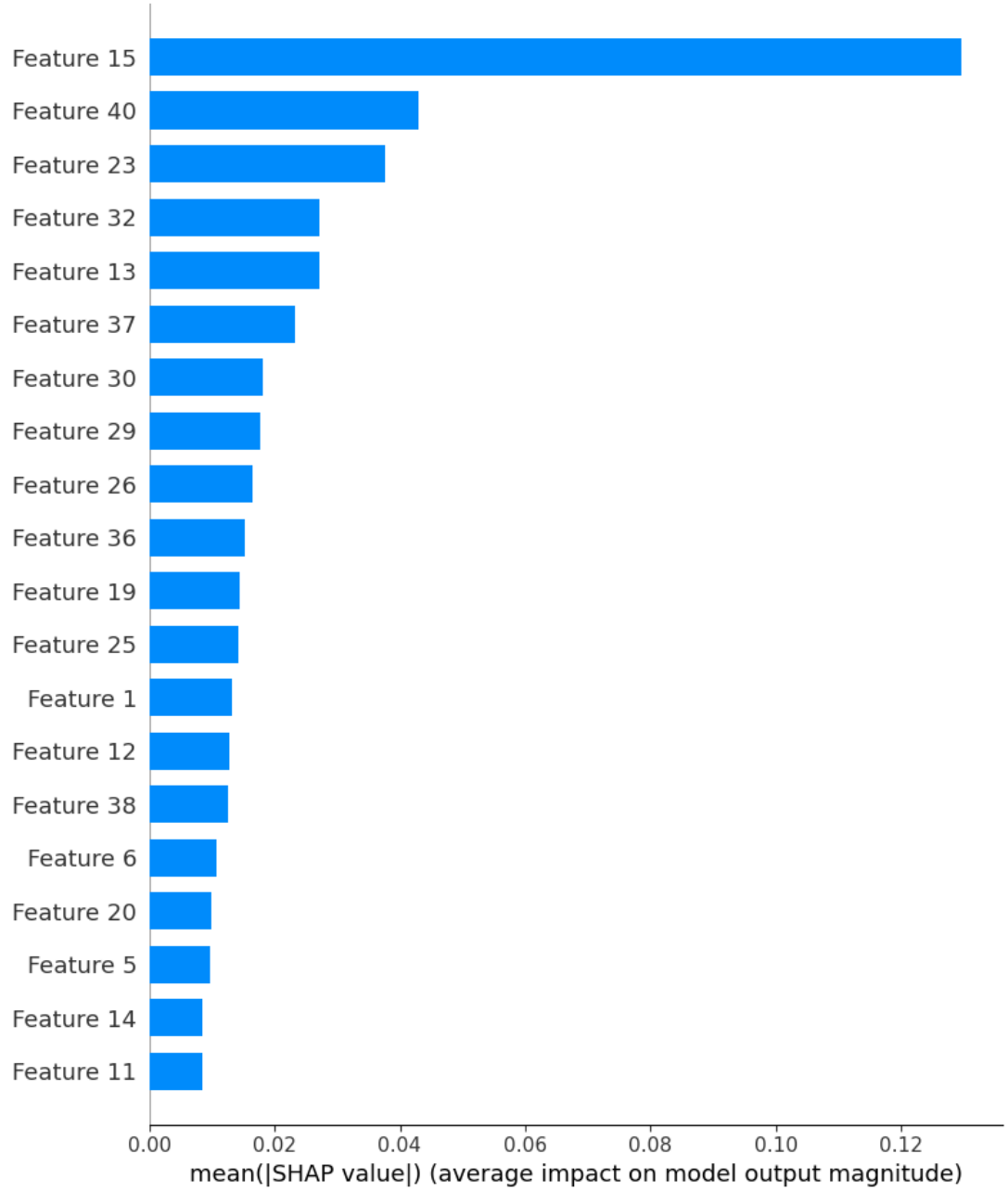

Figure 6.2: SHAP summary plot for the CTU13 Random Forest model tested against CTU13 data.

Figure 6.2 presents the SHAP summary plot for the Random Forest model trained and tested on CTU13. Features are ranked by importance, with ‘Bwd Packet Length Mean‘ and ‘Flow IAT Min‘ exerting the greatest influence on predictions. This suggests that packet size in the backward direction (destination to source) and the minimum inter-arrival time between packets

are key for detecting botnet traffic in CTU13. This aligns with botnets exhibiting abnormal communication patterns, as discussed in Chapter 2, Section 2.1.

Other notable features include ‘Fwd Packet Length Min‘, ‘Fwd Packet Length Max‘, and ‘Fwd IAT Min‘, indicating that forward-direction packet characteristics also contribute significantly. The prominence of both spatial (packet size) and temporal (inter-arrival time) features highlights their role in identifying botnet behaviour, consistent with prior research in Chapter 3, Section 3.3.

Random Forest Model on CICIDS2017

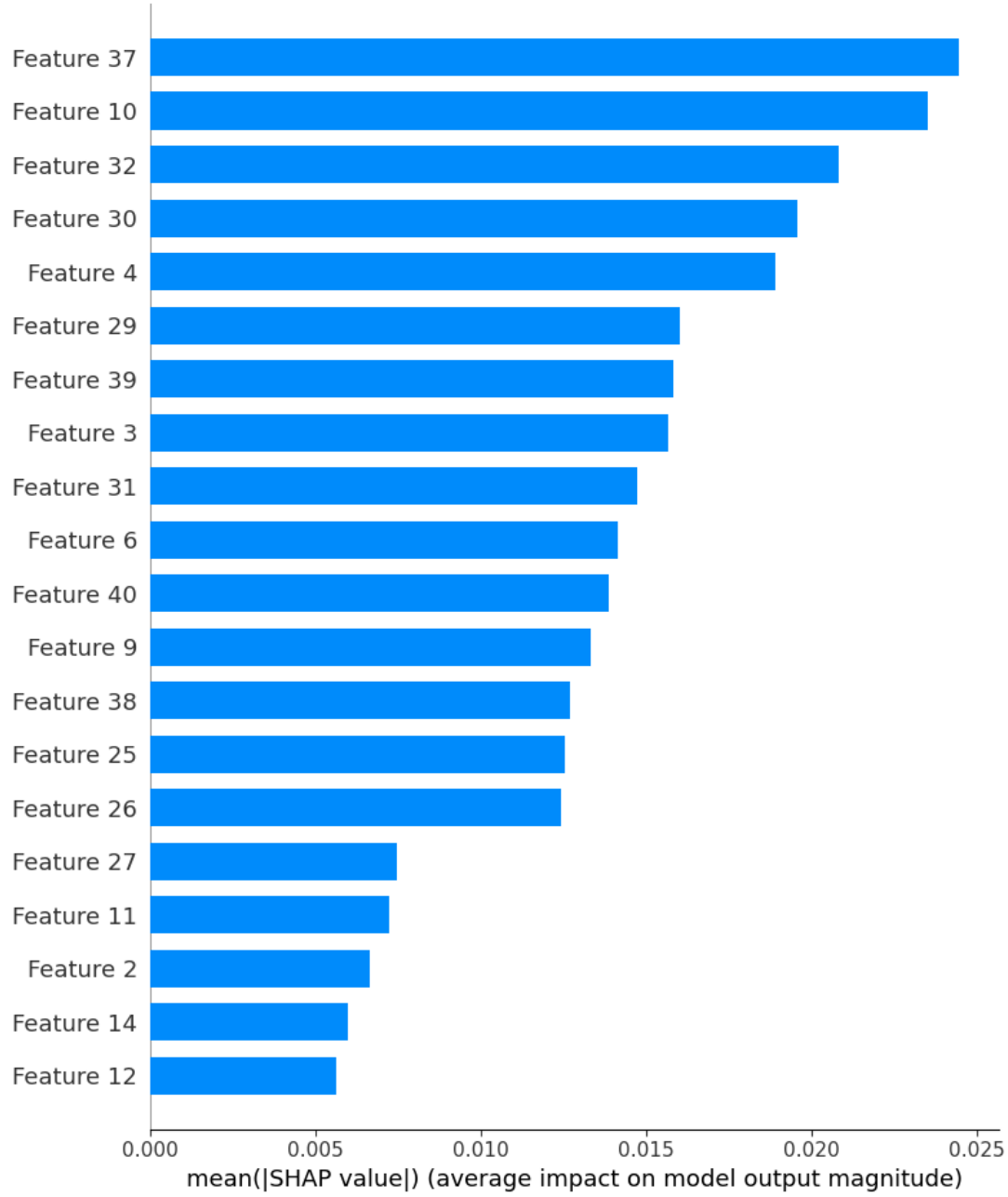

Figure 6.3: SHAP summary plot for the CICIDS2017 Random Forest model tested against CICIDS2017 data.

For the Random Forest model on CICIDS2017 (Figure 6.3), the feature importance differs. ‘Fwd Packet Length Max‘ and ‘Average Packet Size‘ lead, followed by ‘Bwd Packet Length Std‘ and ‘Flow Duration‘. The focus on forward-direction packet sizes underscores their discriminative power for CICIDS2017 attacks, as noted in Chapter 2, Section 2.1. The high ranking of ‘Average

Packet Size‘ suggests deviations from typical profiles signal malicious activity, while ‘Flow Duration‘ emphasises temporal relevance, aligning with prior analyses in Chapter 3, Section 3.3.

The variation in feature importance between CTU13 and CICIDS2017 highlights dataset-specific discriminative attributes, motivating transferability studies to address RQ1 and RQ3, as outlined in Chapter 4, Section 4.3.

6.1.3 Performance on Different Datasets

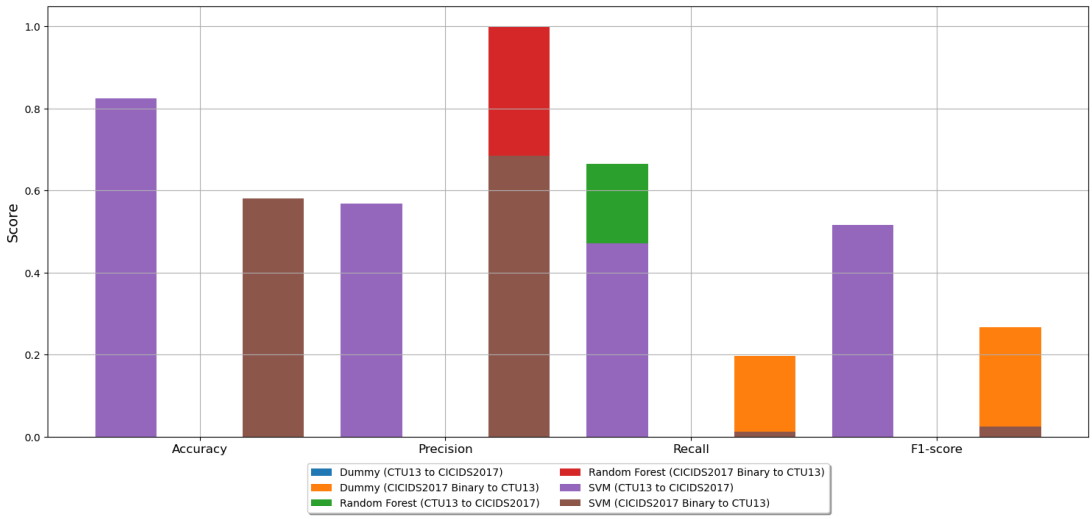

Figure 6.4: Classifiers’ performance across datasets.

To evaluate transferability, we assessed model performance on datasets different from their training sets, as shown in Figure 6.4. The Random Forest model trained on CICIDS2017 and tested on CTU13 exhibited a significant performance drop: accuracy fell to 0.58, precision to 0.65, recall to 0.02, and F1 score to 0.04. This indicates that patterns learned on CICIDS2017 do not translate effectively to CTU13 botnet detection, reflecting challenges in cross-dataset model transfer, as discussed in Chapter 3, Section 3.6.

Low recall suggests the model misses much of the botnet traffic in CTU13, possibly due to differing attack profiles or overfitting to CICIDS2017-specific patterns. This underscores dataset bias impacts and the need for representative training data, addressing RQ2, per Chapter 4, Section 4.3.

Conversely, the SVM trained on CTU13 and tested on CICIDS2017 showed improved transferability, with an accuracy of 0.83, precision of 0.57, recall of 0.49, and F1 score of 0.53. Though

still below dummy classifier benchmarks, this outperforms its CTU13 baseline, suggesting more transferable decision boundaries than the Random Forest in the reverse direction. However, performance remains suboptimal, with a precision-recall balance indicating many false positives and negatives, reflecting network traffic dynamics and attack evolution challenges, as noted in Chapter 3, Section 3.6.

These results highlight cross-dataset performance degradation due to differences in distributions, attack types, and feature profiles. Addressing these requires advanced techniques like transfer learning and domain adaptation, as explored in Chapter 8.

6.1.4 Feature Importance Analysis on Different Datasets

We further examined SHAP feature importance for models applied to different datasets, with interpretation guidance in appendix listing B.

Random Forest Model on CTU13 Tested on CICIDS2017

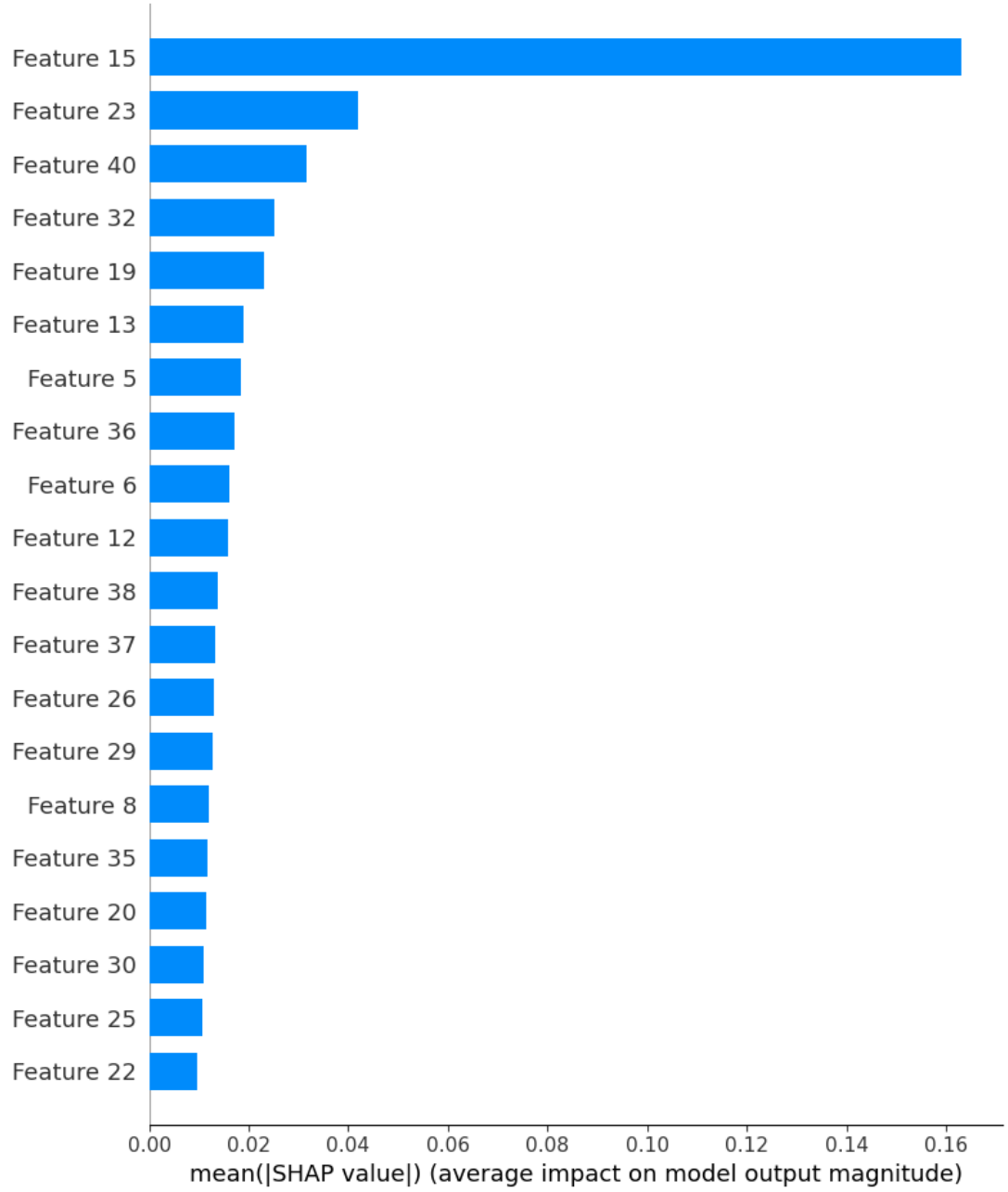

Figure 6.5: SHAP summary plot for the CTU13 Random Forest model tested against CICIDS2017 data.

Figure 6.5 shows SHAP values for the CTU13-trained Random Forest applied to CICIDS2017. Compared to Figure 6.2, feature rankings shift: ‘Bwd Packet Length Mean‘ remains key, but ‘Init Win bytes backward‘ and ‘Flow Packets/s‘ rise in importance. This suggests reliance on different features due to CICIDS2017’s distinct traffic profiles, with ‘Init Win bytes backward‘

potentially better distinguishing attack flows.

Despite this shift, the model's classification performance remains suboptimal with data unlike its training set. This highlights transferability challenges, as discussed in Chapter 8.

Random Forest Model on CICIDS2017 Tested on CTU13

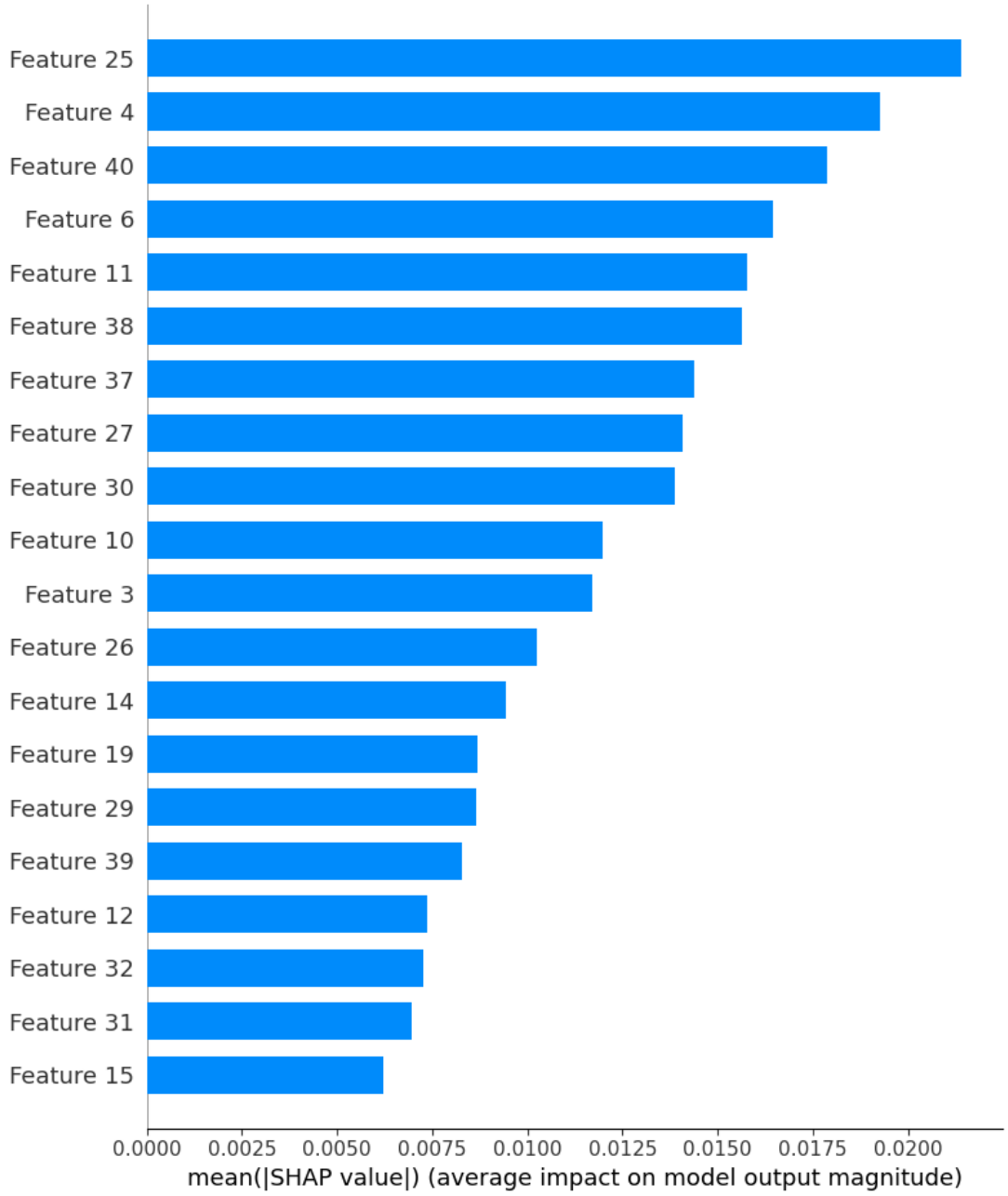

Figure 6.6: SHAP summary plot for the CICIDS2017 Random Forest model tested against CTU13 data.

Figure 6.6 illustrates feature importance for the CICIDS2017-trained Random Forest on CTU13. Compared to Figure 6.3, ‘Bwd Packet Length Std‘ and ‘Fwd IAT Total‘ overtake ‘Fwd Packet Length Max‘ and ‘Average Packet Size‘, reflecting a focus on backward packet variation and forward inter-arrival times. However, this aligns with poor performance, reinforcing cross-dataset application difficulties, as noted in Chapter 3, Section 3.6.

SHAP analysis across datasets offers insights into pattern transferability and feature relevance for attack detection, addressing RQ3 (Chapter 4, Section 4.3). Feature importance shifts and performance impacts highlight dataset biases and direct application limits, per Chapter 3, Section 3.6.

These findings emphasise the need for advanced techniques—transfer learning, domain adaptation, and robust feature engineering—to enhance transferability and generalisation in network intrusion detection models, as discussed in Chapter 8. Developing adaptable, interpretable models can lead to reliable machine learning-based systems for detecting evolving cyber threats across diverse network environments.

Chapter 7

Legal, Social, Ethical and Professional Issues

The British Computer Society (BCS) Code of Conduct [6] establishes the professional standards expected of its members, promoting excellence in competence, conduct, and ethical practice. This research is designed, executed, and disseminated in alignment with these principles.

This chapter examines the potential legal, social, ethical, and professional issues arising from the research process. It also demonstrates how the research complies with the BCS Code of Conduct.

7.1 Legal Considerations

The research utilises two publicly available datasets: CTU13 [11] and CICIDS2017 [24]. Both datasets are freely accessible for research purposes and are not subject to legal restrictions. No personal data is involved, as CICIDS2017 is a synthetic dataset, and CTU13 excludes passive network flows that might contain sensitive information. By avoiding personal data, the research ensures compliance with the UK Data Protection Act 2018.

7.2 Ethical and Social Considerations

This research investigates the transferability and generalisation of machine learning models across diverse datasets. As it does not involve human subjects and relies on publicly available datasets, no direct ethical or social issues arise. Nevertheless, the broader implications of developing effective network intrusion detection systems merit consideration.

By improving the detection and prevention of cyber attacks, this research enhances the security and privacy of individuals and organisations. Robust, transferable machine learning models for intrusion detection can safeguard sensitive information, prevent data breaches, and reduce risks from malicious activities in networked environments.

However, the use of explainable AI techniques, such as SHAP, introduces ethical concerns. While SHAP illuminates the decision-making processes of machine learning models, it may also reveal the most influential features for detecting specific attacks. If misused, this information could enable malicious actors to manipulate or spoof these features to evade detection. Thus, insights from SHAP must be handled with care, protected from unauthorised disclosure, and secured through robust measures to maintain their confidentiality and integrity.

Additionally, highly effective intrusion detection systems risk unintended consequences. Over-reliance on automation might diminish human expertise and judgement, necessitating a balance between leveraging machine learning capabilities and preserving human oversight. False positives, which could disrupt operations and misallocate resources, also require careful management through safeguards and human review processes.

7.3 Professional Considerations

The research reveals limited transferability of machine learning models across datasets, posing challenges for organisations using these models for cybersecurity. It advises caution when deploying models across varied environments, urging organisations to enhance model generalisation to mitigate potential issues. The explainability results, however, offer insights into these limitations, laying the groundwork for future improvements.

Professionally, this research underscores the need for rigorous evaluation and validation of machine learning models in network intrusion detection. Organisations must account for dataset limitations and biases during training and testing. Techniques like SHAP highlight critical

features for attack detection, fostering the development of more reliable systems.

Yet, professionals must recognise the risks of explainable AI in cybersecurity. SHAP-derived insights are sensitive and require protection from unauthorised access. Professionals bear responsibility for their ethical use, preventing inadvertent aid to malicious actors. Clear guidelines and protocols for handling these insights are essential to minimise misuse risks.

Moreover, professionals should engage in ongoing research to enhance model transferability and generalisability. Collaborative efforts within the cybersecurity community can address this dissertation's identified limitations, advancing robust, adaptable intrusion detection solutions.

7.4 Societal Impact and Sustainability

Effective network intrusion detection systems carry profound societal implications. In an interconnected world, network security underpins public trust, protects sensitive data, and ensures critical infrastructure resilience. This research advances robust, transferable models, helping to counter cyber threats and minimise data breaches, thus fostering a sustainable digital environment.

From a sustainability perspective, improved model transferability reduces the need for extensive retraining and resource-heavy development. This adaptability supports long-term cybersecurity resilience amid evolving threats and complex networks.

However, advanced systems may engender overconfidence or complacency. Raising awareness of their limitations and advocating a multi-layered security approach is vital. False positives, which could undermine trust and cause disruptions, must be minimised through clear communication and redress mechanisms.

7.5 British Computing Society Code of Conduct

Conducted with integrity and professionalism, this research adheres to the BCS Code of Conduct. Results are presented transparently using unrestricted, publicly available datasets, and no direct ethical or social issues arise. Findings are communicated clearly, with limitations acknowledged.

The research aligns with BCS principles by promoting responsible technology use and advancing

cybersecurity knowledge. It upholds honesty, integrity, and objectivity while addressing network intrusion detection—a key public interest issue—through effective, reliable solutions.

It also reflects professional competence and a commitment to continuous improvement, building on existing knowledge to enhance understanding of model transferability and explainability. Insights from this work guide future research and professional development in cybersecurity.

Nonetheless, the research acknowledges risks tied to explainable AI, particularly with SHAP. In line with the BCS Code of Conduct, it stresses responsible handling of these insights, advocating guidelines to prevent misuse. By addressing these ethical considerations, the research exemplifies BCS-aligned responsible AI use in cybersecurity.

Chapter 8

Conclusion and Future Work

In this dissertation, we set out to investigate three key research questions related to the transferability and interpretability of machine learning models for network intrusion detection:

1. How well do machine learning models trained on one dataset transfer and generalise to another dataset in the context of network intrusion detection?

2. What is the impact of dataset biases and representativeness on the performance of machine learning-based intrusion detection systems?

3. How can explainable AI techniques, such as SHAP, provide insights into the transferability of learned patterns and the most relevant features for detecting specific types of attacks across different datasets?

We conducted experiments using the CTU13 and CICIDS2017 datasets to address these questions. We trained Random Forest and SVM classifiers on one dataset and evaluated their performance and transferability on the other. We also applied the SHAP explainable AI technique to interpret the models' predictions and identify the most essential features for detecting malicious traffic.

Our results highlight the challenges of transferring machine learning models across network intrusion datasets. The Random Forest classifiers achieved strong performance when trained and tested on the same dataset, with accuracy, precision, recall, and F1 scores above 0.99 for binary classification on CTU13 and CICIDS2017. However, their effectiveness significantly

degraded when we applied the models to a different dataset. For example, the Random Forest trained on CICIDS2017 and tested on CTU13 saw its accuracy drop to 0.58, precision to 0.65, recall to 0.02, and F1 score to 0.04.

The SVM classifiers exhibited similar transferability issues, although they also struggled to learn effective decision boundaries even on their training datasets. These findings underscore the impact of dataset biases and the importance of representative training data for building robust intrusion detection models.

The SHAP analysis provided valuable insights into the factors contributing to the transferability challenges. We observed notable shifts in feature importance when the models were applied to different datasets, indicating that the discriminative power of individual attributes can vary significantly depending on the data distribution. This variability makes it difficult for models to maintain effectiveness when deployed in new environments.

One limitation of this research was our inability to obtain SHAP values for the SVM classifiers due to the high computational cost of the kernelExplainer objects. Running the regular SHAP library on a CPU led to performance issues and memory constraints, while the GPU-accelerated version also encountered memory limitations due to insufficient dedicated GPU memory. Exploring SVM interpretability with more powerful hardware could be an exciting direction for future work.

In order to tackle the transferability challenges highlighted in this dissertation, further research could delve into various techniques such as transfer learning, domain adaptation, and robust feature engineering. Transfer learning methods, such as fine-tuning models on a subset of labelled target data, could aid in adapting learned patterns to novel environments. Moreover, unsupervised domain adaptation techniques could align feature distributions between source and target datasets, rendering the models more generalisable. Furthermore, developing feature learning methods that can identify dataset-invariant representations could enhance the transferability of intrusion detection models.

Another promising direction for future work is the development of more sophisticated explainable AI techniques tailored to the cybersecurity domain. While SHAP provided valuable insights into the importance of features, methods that can capture complex interactions and temporal dependencies in network traffic data are needed. Explainable AI techniques that provide more granular, contextualised explanations of model predictions could significantly

enhance the interpretability and trustworthiness of intrusion detection systems.

Furthermore, future research could explore integrating multiple data sources and modalities to build more comprehensive and robust intrusion detection models. Combining network traffic data with host-based logs, application-level events, and external threat intelligence could provide a more holistic view of the cybersecurity landscape and improve the accuracy and resilience of detection systems.

In conclusion, this dissertation highlights the importance of evaluating model transferability and interpretability in network intrusion detection. Our experiments demonstrate the challenges of applying machine learning models trained on one dataset to another, emphasising the need for representative training data and techniques to address dataset biases. The SHAP analysis provides valuable insights into the factors contributing to these challenges, including shifts in feature importance and potential overfitting to dataset-specific patterns.

We can work towards more practical and reliable machine learning-based network intrusion detection systems by advancing our understanding of these issues and developing more transferable and interpretable models. The results and insights presented in this dissertation lay the foundation for future research in this critical area, contributing to the ongoing efforts to safeguard our digital infrastructure against evolving cyber threats.

References

[1] RAPIDS AI. Rapids download link. `https://docs.rapids.ai/install`.

[2] Dogukan Aksu and M Ali Aydin. Detecting port scan attempts with comparative analysis of deep learning and support vector machine algorithms. In *2018 International congress on big data, deep learning and fighting cyber terrorism (IBIGDELFT)*, pages 77–80. IEEE, 2018.

[3] Kasun Amarasinghe, Kevin Kenney, and Milos Manic. Toward explainable deep neural network based anomaly detection. In *2018 11th international conference on human system interaction (HSI)*, pages 311–317. IEEE, 2018.

[4] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. Dos and don'ts of machine learning in computer security. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3971–3988, 2022.

[5] Mustapha Belouch, Salah El Hadaj, and Mohamed Idhammad. Performance evaluation of intrusion detection based on machine learning using apache spark. *Procedia Computer Science*, 127:1–6, 2018.

[6] British Computer Society. Bcs code of conduct. `https://www.bcs.org/media/2211/bcs-code-of-conduct.pdf`.

[7] Anna L Buczak and Erhan Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications surveys & tutorials*, 18(2):1153–1176, 2015.

[8] Sudipta Chowdhury, Mojtaba Khanzadeh, Ravi Akula, Fangyan Zhang, Song Zhang, Hugh Medal, Mohammad Marufuzzaman, and Linkan Bian. Botnet detection using graph-based feature clustering. *Journal of Big Data*, 4:1–23, 2017.

[9] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20:273–297, 1995.

[10] Nabila Farnaaz and MA Jabbar. Random forest modeling for network intrusion detection system. *Procedia Computer Science*, 89:213–217, 2016.

[11] Sebastian Garcia, Martin Grill, Jan Stiborek, and Alejandro Zunino. An empirical comparison of botnet detection methods. *computers & security*, 45:100–123, 2014.

[12] Trevor Hastie, Robert Tibshirani, Jerome Friedman, Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Random forests. *The elements of statistical learning: Data mining, inference, and prediction*, pages 587–604, 2009.

[13] Md Reazul Kabir, Abdur Rahman Onik, and Tanvir Samad. A network intrusion detection framework based on bayesian network using wrapper approach. *International Journal of Computer Applications*, 166(4):13–17, 2017.

[14] Dong Seong Kim and Jong Sou Park. Network-based intrusion detection with support vector machines. In *Information Networking: International Conference, ICOIN 2003, Cheju Island, Korea, February 12-14, 2003. Revised Selected Papers*, pages 747–756. Springer, 2003.

[15] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[16] Faisal Malik. Ctu13 csv files. `https://github.com/imfaisalmalik/CTU13-CSV-Dataset`.

[17] Shraddha Mane and Dattaraj Rao. Explaining network intrusion detection system using explainable ai framework. *arXiv preprint arXiv:2103.07110*, 2021.

[18] Mirco Marchetti, Fabio Pierazzi, Michele Colajanni, and Alessandro Guido. Analysis of high volumes of network traffic for advanced persistent threat detection. *Computer Networks*, 109:127–141, 2016.

[19] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[20] Abdurrahman Pektaş and Tankut Acarman. A deep learning method to detect network intrusion through flow-based features. *International Journal of Network Management*, 29(3):e2050, 2019.

[21] Sebastian Raschka, Joshua Patterson, and Corey Nolet. Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *arXiv preprint arXiv:2002.04803*, 2020.

[22] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[23] Iman Sharafaldin, Arash Habibi Lashkari, Ali A Ghorbani, et al. Cicids2017 csv files. `http://205.174.165.80/CICDataset/CIC-IDS-2017/`.

[24] Iman Sharafaldin, Arash Habibi Lashkari, Ali A Ghorbani, et al. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1:108–116, 2018.

[25] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE symposium on security and privacy*, pages 305–316. IEEE, 2010.

[26] Shaohua Teng, Naiqi Wu, Haibin Zhu, Luyao Teng, and Wei Zhang. Svm-dt-based adaptive and collaborative intrusion detection. *IEEE/CAA Journal of Automatica Sinica*, 5(1):108–118, 2017.

[27] Serpil Ustebay, Zeynep Turgut, and Muhammed Ali Aydin. Intrusion detection system with recursive feature elimination by using random forest and deep learning classifier. In *2018 international congress on big data, deep learning and fighting cyber terrorism (IBIGDELFT)*, pages 71–76. IEEE, 2018.

[28] Alexander Warnecke, Daniel Arp, Christian Wressnegger, and Konrad Rieck. Evaluating explanation methods for deep learning in security. In *2020 IEEE european symposium on security and privacy (EuroS&P)*, pages 158–174. IEEE, 2020.

Appendix A

Algorithms

Algorithm 1 Relabeling CTU13 Dataset

Require: Raw CTU13 dataset
Ensure: Preprocessed CTU13 dataset with consistent feature names and class labels

```
Import necessary libraries
function LoadDataset(datasetName)
    Load dataset specified by datasetName
    return Loaded dataset
end function
function RenameFeatures(dataset, mappingDict)
    for each feature in dataset do
        if the feature name exists in the mapping dictionary then
            Rename the feature using the corresponding value from the dictionary
        end if
    end for
end function
function ReplaceLabels(dataset, labelMappings)
    for each labelMapping in labelMappings do
        Replace the class label in dataset using labelMapping
    end for
end function
function ReorderFeatures(dataset, desiredOrder)
    Reorder the features in dataset according to desiredOrder
    return Reordered dataset
end function
rawCTU13 ← LoadDataset('CTU13')
mappingDict ← Define a dictionary for mapping feature names
RenameFeatures(rawCTU13, mappingDict)
labelMappings ← {'0': 'Benign', '1': 'Botnet'}
ReplaceLabels(rawCTU13, labelMappings)
desiredOrder ← Define the desired order of features based on the CICIDS2017 dataset
preprocessedCTU13 ← ReorderFeatures(rawCTU13, desiredOrder)
Save the preprocessedCTU13 dataset
```

Algorithm 2 Relabeling CICIDS2017 Dataset

Require: Raw CICIDS2017 and preprocessed CTU13 datasets
Ensure: Preprocessed CICIDS2017 dataset with consistent feature names and class labels

```
Import necessary libraries
function LoadDataset(datasetName)
    Load dataset specified by datasetName
    return Loaded dataset
end function
function RenameFeatures(sourceDataset, targetDataset, mappingDict)
    for each feature in sourceDataset do
        if the feature name exists in targetDataset then
            Rename the feature in targetDataset using the corresponding value from mappingDict
        end if
    end for
end function
function ChangeLabels(dataset, oldLabel, newLabel)
    Change the label in dataset from oldLabel to newLabel
end function
function PreprocessDataset(sourceDataset, targetDataset)
    Get the list of columns in sourceDataset
    Get the common columns between sourceDataset and targetDataset
    Reorder and select the common columns in targetDataset
    return Preprocessed dataset
end function
rawCICIDS2017 ← LoadDataset('CICIDS2017')
preprocessedCTU13 ← LoadDataset('CTU13')
mappingDict ← Define a dictionary for mapping feature names from CTU13 to CICIDS2017
RenameFeatures(preprocessedCTU13, rawCICIDS2017, mappingDict)
ChangeLabels(rawCICIDS2017, '0', 'Benign')
ChangeLabels(rawCICIDS2017, '1', 'Botnet')
preprocessedCICIDS2017 ← PreprocessDataset(preprocessedCTU13, rawCICIDS2017)
Save the preprocessedCICIDS2017 dataset
```

Algorithm 3 Training Dummy Classifiers

Require: Preprocessed CTU13 and CICIDS2017 datasets
Ensure: Trained Dummy Classifiers and performance metrics

```
Import necessary libraries
Read preprocessed CTU13 and CICIDS2017 datasets
Define common features for training and testing
function TrainDummy(dataset, classificationType)
    Train Dummy Classifier on dataset ('classificationType')
    Save the trained classifier
end function
function EvaluateClassifier(classifier)
    Load the trained classifier
    Test the classifier on its corresponding test set as well as the other dataset
    Calculate performance metrics (accuracy, precision, recall, F1 score) for each experiment
    Save performance metrics for analysis and comparison
end function
for dataset in [CTU13, CICIDS2017] do
    if dataset is CTU13 then
        TrainDummy(CTU13, 'binary classification')
    else if dataset is CICIDS2017 then
        TrainDummy(CICIDS2017, 'binary classification')
        TrainDummy(CICIDS2017, 'multiclass classification')
    end if
end for
for each trained Dummy Classifier do
    EvaluateClassifier(classifier)
end for
```

Algorithm 4 Training Random Forest Classifiers

Require: Preprocessed CTU13 and CICIDS2017 datasets
Ensure: Trained Random Forest Classifiers, performance metrics, and SHAP values

```
Import necessary libraries
Read preprocessed CTU13 and CICIDS2017 datasets
Define common features for training and testing
function TrainRandomForest(dataset, classificationType)
    Train Random Forest Classifier on dataset (‘classificationType‘) using CUML
    Save the trained classifier
end function
function EvaluateClassifier(classifier)
    Load the trained classifier
    Test the classifier on its corresponding test set as well as the other dataset
    Calculate performance metrics (accuracy, precision, recall, F1 score) for each experiment
    Save performance metrics for analysis and comparison
    Create a SHAP explainer object for the classifier using CUML
    Calculate SHAP values for the test set
    Save SHAP values for analysis and comparison
end function
for dataset in [CTU13, CICIDS2017] do
    if dataset is CTU13 then
        TrainRandomForest(CTU13, ‘binary classification’)
    else if dataset is CICIDS2017 then
        TrainRandomForest(CICIDS2017, ‘binary classification’)
        TrainRandomForest(CICIDS2017, ‘multiclass classification’)
    end if
end for
for each trained Random Forest Classifier do
    EvaluateClassifier(classifier)
end for
```

Algorithm 5 Training Support Vector Machine Classifiers

Require: Preprocessed CTU13 and CICIDS2017 datasets
Ensure: Trained SVM Classifiers, performance metrics, and SHAP values

```
Import necessary libraries
Read preprocessed CTU13 and CICIDS2017 datasets
Define common features for training and testing
function TrainSVM(dataset, classificationType)
    Train SVM Classifier on dataset (classificationType) using CUML
    Save the trained classifier
end function
function EvaluateClassifier(classifier)
    Load the trained classifier
    Test the classifier on its corresponding test set as well as the other dataset
    Calculate performance metrics (accuracy, precision, recall, F1 score) for each experiment
    Save performance metrics for analysis and comparison
    Create a SHAP explainer object for the classifier using CUML
    Calculate SHAP values for the test set
    Save SHAP values for analysis and comparison
end function
for dataset in [CTU13, CICIDS2017] do
    if dataset is CTU13 then
        TrainSVM(CTU13, 'binary classification')
    else if dataset is CICIDS2017 then
        TrainSVM(CICIDS2017, 'binary classification')
        TrainSVM(CICIDS2017, 'multiclass classification')
    end if
end for
for each trained SVM Classifier do
    EvaluateClassifier(classifier)
end for
```

Algorithm 6 Plotting Performance Metrics and Classifier Generalisability

Require: Performance metrics for trained classifiers
Ensure: Plots comparing classifier performance and generalisability

```
function SetUpPlot
    Import necessary libraries
    Define classifiers and metrics
    Set bar width and spacing
    Create figures and axes with larger sise
    Set positions of bars on the x-axis
end function
function PlotMetrics(data)
    for each classifier in data do
        Plot bars for performance metrics
    end for
    Set x-tick labels and positions
    Add labels and title with a larger font size
    Add a grid for better readability
    Add legend outside the plot
    Adjust the layout to make space for the legend
    Display the plot
end function
SetUpPlot
Define data for classifier performance on the same dataset
for each classifier and dataset combination do
    Store performance metrics in the corresponding data structure
end for
PlotMetrics(classifier performance data)
SetUpPlot
Define data for classifier generalisability across datasets
for each classifier and dataset transfer combination do
    Store performance metrics in the corresponding data structure
end for
PlotMetrics(classifier generalisability data)
```

Appendix B

SHAP Feature List

This appendix lists the features used in the SHAP analysis for the Random Forest classifiers trained on the CTU13 and CICIDS2017 datasets. The features are listed in the order they appear for the models, which is also the order they are numbered in the SHAP summary plots. Use this table to reference the feature number to the corresponding feature name.

List of features:

1. ACK Flag Count
2. Active Max
3. Active Min
4. Active Std
5. Average Packet Size
6. Avg Bwd Segment Size
7. Avg Fwd Segment Size
8. Bwd Header Length
9. Bwd IAT Max
10. Bwd IAT Mean
11. Bwd IAT Min
12. Bwd IAT Std
13. Bwd PSH Flags
14. Bwd Packet Length Mean
15. Bwd Packet Length Min
16. Bwd Packet Length Std
17. Bwd Packets/s
18. Down/Up Ratio
19. Flow Duration
20. Flow IAT Max

21. Flow IAT Mean
22. Flow IAT Min
23. Flow IAT Std
24. Flow Packets/s
25. Fwd Header Length
26. Fwd IAT Max
27. Fwd IAT Mean
28. Fwd IAT Min
29. Fwd IAT Std
30. Fwd Packet Length Max
31. Fwd Packet Length Mean
32. Fwd Packet Length Min
33. Fwd Packet Length Std
34. Idle Max
35. Idle Min
36. Idle Std
37. Init_Win_bytes_backward
38. Max Packet Length
39. Min Packet Length
40. Packet Length Mean
41. Packet Length Std
42. Packet Length Variance
43. RST Flag Count
44. SYN Flag Count
45. Total Backward Packets
46. Total Fwd Packets
47. Total Length of Bwd Packets
48. act_data_pkt_fwd

Appendix C

User Guide

This user guide outlines the structure and requirements for running the provided source code, including file descriptions, directory structure, dataset sources, and setup instructions.

C.0.1 Source Code Files

The project includes the following Python scripts and Jupyter notebooks:

1. `relabelCTU13.py` (Algorithm 1): Script for relabeling the CTU-13 dataset.
2. `relabelCICIDS2017.py` (Algorithm 2): Script for relabeling the CICIDS2017 dataset.
3. `trainDummyClassifier.ipynb` (Algorithm 3): Notebook for training a dummy classifier.
4. `trainRandomForest.ipynb` (Algorithm 4): Notebook for training a RandomForest classifier.
5. `trainSVM.ipynb` (Algorithm 5): Notebook for training an SVM classifier.
6. `plotData.ipynb` (Algorithm 6): Notebook for plotting dataset statistics and results.

C.0.2 Directory Structure

The code is designed to work with the following directory structure:

- A root directory containing the source code files.

- Two subdirectories within the root:
  - `CTU13` — Contains the CTU-13 dataset files.
  - `CICIDS2017` — Contains the CICIDS2017 dataset files.

C.0.3 Datasets

The dataset files are available from the following sources:

- CTU-13 dataset: [16]
- CICIDS2017 dataset: [23]

Note: The CTU-13 CSV files are provided in the correct format. For CICIDS2017, download the dataset and use the CSV files in the `ML` directory.

C.0.4 Installation and Setup

Before running the code, ensure that your environment meets the following requirements:

- IDE can run Python and Jupyter notebooks (e.g., Visual Studio Code with the necessary extensions).
- Python version 3.10.14 (specifically, any 3.10.x version should suffice).
- A Nvidia GPU with CUDA support for running the CUML models.
- A valid RAPIDS AI environment. Follow the installation instructions at [1], choosing the RAPIDS version 24.02 with the CUDA 12 option.
- The following Python packages (compatible versions are listed):
  - `pandas` (2.2.1)
  - `numpy` (1.26.4)
  - `cuml` (24.02)
  - `shap` (0.45.0)
  - `matplotlib` (3.8.3)

- `ipywidgets` (8.1.2)

Install the required packages using the following conda command. It is better to install them all at the same time so conda can resolve dependencies correctly:

```
conda install
pandas=2.2.1 numpy=1.26.4 shap=0.45.0 matplotlib=3.8.3 ipywidgets=8.1.2
```

After setting up the directory structure and installing the necessary packages, run the code files in the order listed above to preprocess the datasets, train classifiers, evaluate their performance, and generate visualisations of the data and results.