Transferability and Explainability of Machine Learning Models for Network Intrusion Detection

Final Project Report

Author: Ahmed Bedair

Supervisor: Dr. Fabio Pierazzi

Student ID: K2105577

April 11, 2024

Abstract

This dissertation explores the efficacy of machine learning-based Network Intrusion Detection Systems (NIDS) in identifying cyber threats through pattern analysis in network traffic data.

By specifically examining the transferability of patterns between the Random Forests and Support Vector Machines (SVMs) across different datasets, this dissertation evaluates flow-based features to differentiate between malicious and benign traffic. It also employs explainable AI techniques to pinpoint critical features for threat detection. The insights into the adaptability of learned patterns and the challenges of dataset biases and model explainability contribute to the enhancement of NIDS. The strategies for improving detection accuracy and model transparency offered in this dissertation can significantly bolster cybersecurity defences, making them more robust and effective in the face of evolving cyber threats.

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

April 11, 2024

Acknowledgements

Thank you to my supervisor, Dr. Fabio Pierazzi, for his guidance and support throughout the project. The supervision and support were very much appreciated and helped me stay on track and complete the project successfully. I would also like to thank my family and friends for their encouragement and support.

Contents

Chapter 1

Introduction

As the digital landscape becomes increasingly interconnected, safeguarding computer network security is crucial. The continuous evolution of cyber threats challenges the efficacy of conventional signature-based Network Intrusion Detection Systems (NIDS), which struggle to identify and mitigate novel and complex attacks [19]. In response, the research community has pivoted towards machine learning strategies that recognise unseen intrusions by analysing network traffic data for patterns and anomalies [6].

A pivotal factor in the success of machine learning-driven NIDS is the authenticity and comprehensiveness of the datasets utilised for training and evaluation purposes [9]. Prominent among these are the CTU13 dataset, encompassing genuine botnet traffic within normal traffic [11], and the CICIDS2017 dataset, which encompasses a broad spectrum of contemporary attack vectors [25]. Despite their widespread usage, the extent to which machine learning models can generalise learned patterns from one dataset to accurately predict intrusions in another dataset still needs to be explored.

This dissertation endeavours to bridge this knowledge gap by meticulously examining the effectiveness of machine learning algorithms, specifically trained on the CICIDS2017 dataset, in detecting botnet activities within the CTU13 dataset. Employing Random Forest and Support Vector Machine (SVM) classifiers trained on CICIDS2017, we evaluate their performance on the CTU13 dataset. This methodology facilitates an investigation into the transferability and applicability of learned features and patterns across distinct datasets, mirroring a scenario wherein a model trained on a particular dataset is applied to safeguard against intrusions in

diverse network environments.

Guided by Arp et al.'s recommendations for the application of machine learning in cybersecurity contexts [4], this research incorporates explainable AI techniques, notably SHAP (SHapley Additive exPlanations) [16], to elucidate our trained models' decision-making processes. Identifying critical features for the detection of botnet traffic enables us to elucidate distinctive patterns and characteristics differentiating malicious from benign network flows.

Furthermore, this dissertation conducts a detailed comparative analysis of network flow features derived from the CTU13 and CICIDS2017 datasets. By examining the statistical properties and distribution of various features, such as flow duration, packet counts, and inter-arrival times, we aim to discern the discrepancies between real and synthetically generated network traffic. This comparison illuminates synthetic datasets' inherent challenges and constraints in training NIDS models, underscoring the imperative of acknowledging dataset biases during performance assessments.

Incorporating both Random Forest and SVM classifiers, this study facilitates a thorough evaluation of model transferability and generalizability. Through performance comparison and prediction interpretation via SHAP, we derive significant insights into the resilience and versatility of these algorithms in identifying botnet attacks across different datasets.

This dissertation is structured as follows: Section 2 provides a foundational understanding of the CTU13 and CICIDS2017 datasets, machine learning classifiers, and explainable AI techniques. Section 3 reviews pertinent literature in the domain of machine learning-based NIDS. Section 4 outlines the specifications and design of our experimental framework, whereas Section 5 elaborates on the implementation. Section 6 discusses the evaluation outcomes and their implications. Section 7 deliberates on the legal, social, ethical, and professional considerations of deploying machine learning in network intrusion detection. The dissertation concludes with Section 8, summarising our findings and suggesting avenues for future research.

Chapter 2

Background

Section 2.1 provides an overview of two widely used datasets in the field of network intrusion detection: the CTU13 dataset [11] and the CICIDS2017 dataset [25]. Researchers have extensively utilised these datasets to develop and evaluate machine learning-based NIDS. Understanding the datasets' characteristics, strengths, and limitations used to train machine learning models is essential for designing robust and effective intrusion detection systems.

Section 2.2 discusses two popular machine learning classifiers, Random Forest and Support Vector Machine, widely used in network intrusion detection systems. The strengths and limitations of these classifiers are highlighted, along with their performance on different datasets. Understanding the capabilities and trade-offs of the classifiers used in a machine learning NIDS is crucial for selecting appropriate models for intrusion detection.

Section 2.3 introduces explainable AI techniques, which aim to provide insights into the decision-making process of machine learning models. These techniques help us understand the factors that contribute to the predictions made by the models and offer interpretable explanations for their outputs. Understanding the importance of the features a model uses to predict benign or malicious network flows is essential for validating the reliability of machine learning models and gaining insights into their decision-making process.

2.1 CTU13 and CICIDS2017 Datasets

The availability of representative and labelled datasets is crucial for developing and evaluating machine learning-based network intrusion detection systems. Two widely used benchmark datasets in this domain are the CTU13 dataset [11] and the CICIDS2017 dataset [25].

Garcia et al. [11] introduced the CTU13 dataset, which contains real botnet traffic captured in a controlled environment. The dataset consists of 13 scenarios, each representing specific botnet behaviours such as port scanning, DDoS attacks, click fraud, and spam. The authors provide a detailed description of the dataset creation methodology, including the setup of the controlled environment, the use of actual botnet samples, and the labelling process based on the known behaviour of the captured botnets. They also present an evaluation of the dataset using various machine learning algorithms, demonstrating its utility for botnet detection research. The realistic nature of the CTU13 dataset and the variety of botnet scenarios it covers have made it a popular choice among researchers.

Table 2.1 shows the breakdown of classes in the CTU13 dataset.

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

Table 2.1: CTU13 Dataset Class Breakdown

Sharafaldin et al. [25] created the CICIDS2017 dataset, a more recent and comprehensive dataset for evaluating network intrusion detection systems. The dataset contains many modern attacks, including DoS, DDoS, brute force, XSS, SQL injection, and infiltration. The authors describe the dataset generation process, which involved creating a controlled lab environment resembling a real-world network infrastructure. They used tools and scripts to generate realistic benign traffic and attack scenarios. The dataset also includes a combination of manually labelled and time-based labelled data. The authors evaluate the dataset using different machine learning algorithms and demonstrate its effectiveness in detecting various types of attacks.

Table 2.2 shows the breakdown of classes in the CICIDS2017 dataset.

Comparing the two datasets, it is evident that botnet attacks in the CTU13 dataset share similar characteristics with various attack types in the CICIDS2017 dataset, such as DDoS/DoS attacks, Web attacks, and Bot attacks. These similarities in attack structure and behaviour suggest that machine learning models trained on the CICIDS2017 dataset may be transferable

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

to detecting botnet attacks in the CTU13 dataset.

Several studies have utilised these datasets to develop and evaluate network intrusion detection systems. Chowdhury et al. [7] proposed a graph-based approach for botnet detection using the CTU13 dataset. They constructed a graph representation of the communication patterns among botnet-infected hosts and applied graph analysis techniques to identify botnets. Their approach achieved high accuracy in detecting botnets and demonstrated the potential of leveraging graph-based features for botnet detection.

Pektaş and Acarman [21] applied deep learning techniques to the CTU13 dataset for botnet detection. They used a convolutional neural network (CNN) to learn discriminative features from raw network traffic data. Their proposed model achieved high detection accuracy and showcased the effectiveness of deep learning in capturing complex patterns and behaviours associated with botnets.

Ustebay et al. [28] proposed an intrusion detection system based on a multi-layer perceptron (MLP) classifier using the CICIDS2017 dataset. They performed extensive preprocessing and feature selection to optimise the input data for the MLP classifier. Their approach achieved high detection accuracy for various attack types in the dataset, demonstrating the potential of neural network-based models for network intrusion detection.

Aksu and Aydin [2] conducted a comparative study of different machine learning algorithms for network intrusion detection using the CICIDS2017 dataset. They evaluated the performance of decision trees, random forests, and support vector machines. Their results showed that random

forests outperformed other algorithms regarding accuracy and false-positive rates, highlighting the effectiveness of ensemble learning methods for intrusion detection.

While these studies demonstrate the utility of the CTU13 and CICIDS2017 datasets, it is crucial to acknowledge their limitations. Sommer and Paxson [26] discuss the challenges of using synthetic datasets for network intrusion detection. They argue that synthetic datasets may only partially capture the complexity and diversity of real-world network traffic and may need more crucial contextual information. They emphasise the need for more realistic and representative datasets considering intrusion detection systems' operational aspects and deployment challenges.

Lashkari and colleagues have presented a solution to the constraints surrounding network traffic dataset generation in their study [12]. Their framework utilises methods for producing authentic benign traffic and modelling user actions, resulting in diverse and comprehensive datasets. Additionally, they have introduced the CICFlowMeter tool, which facilitates the extraction of flow-based attributes from unprocessed network traffic data. This tool has become famous for feature extraction and analysis alongside the CICIDS2017 dataset.

However, Engelen et al. [9] identified limitations and issues in the CICFlowMeter tool that affect the quality of the extracted features in the CICIDS2017 dataset. They conducted an in-depth dataset analysis and discovered problems related to flow construction, feature extraction, and labelling. They proposed improvements to the CICFlowMeter tool and generated a revised version of the dataset to address these limitations, enhancing the reliability and utility of the dataset for intrusion detection research.

Sommer and Paxson [26] underscore the significance of utilising unbiased and representative datasets when training machine learning models in network intrusion detection. They draw attention to the obstacles posed by the ever-changing nature of network traffic and attack patterns, which can impact the models' long-term efficacy. Buczak and Guven [6] present a comprehensive survey of data mining and machine learning techniques for cyber security intrusion detection. They delve into the challenges and considerations of applying machine learning methods to network intrusion detection, such as selecting appropriate algorithms, feature engineering, and model evaluation.

2.2 Machine Learning Classifiers

Machine learning classifiers have been widely used in network intrusion detection systems to identify malicious activities and automatically distinguish them from benign traffic. Random Forest (RF) and Support Vector Machine (SVM) are popular classifiers with promising results in this domain.

Random Forest is an ensemble learning method that combines multiple decision trees to make predictions [13]. It constructs many decision trees during the training phase and outputs the majority vote of the individual trees for classification tasks. Random Forest has advantages such as handling high-dimensional data, robustness to noise and outliers, and capturing complex interactions among features.

Farnaaz and Jabbar [10] proposed a Random Forest-based model for intrusion detection and evaluated its performance on the NSL-KDD dataset. They performed feature selection using the Chi-square test and trained the Random Forest classifier on the selected features. Their model achieved an accuracy of 99.67% and a low false positive rate of 0.06%, demonstrating Random Forest's effectiveness in detecting various types of network attacks.

Belouch et al. [5] applied Random Forest to the CICIDS2017 dataset for network intrusion detection. They compared the performance of Random Forest with other machine learning algorithms, including Decision Tree, Naive Bayes, and k-Nearest Neighbors. Their results showed that Random Forest outperformed other algorithms, achieving an accuracy of 99.98% and a false positive rate of 0.01%. They also analysed the importance of different features in the dataset and identified the most discriminative features for intrusion detection.

Support Vector Machine (SVM) is another widely used classifier in network intrusion detection. SVM aims to find the optimal hyperplane that maximally separates different classes in a high-dimensional feature space [8]. It can handle both linear and non-linear classification tasks using kernel functions. SVM is known for its ability to generalise well, even with limited training data, making it suitable for network intrusion detection scenarios where labelled data may be scarce.

Kabir et al. [14] proposed an SVM-based intrusion detection system and evaluated its performance on the NSL-KDD dataset. They used a genetic algorithm for feature selection and optimised the SVM parameters using grid search. Their proposed system achieved an accuracy

of 99.91% and a detection rate of 99.93%, showcasing SVM's effectiveness in detecting various types of network attacks.

Teng et al. [27] applied SVM with different kernel functions for intrusion detection on the CICIDS2017 dataset. They compared the performance of linear, polynomial, and radial basis function (RBF) kernels. Their results showed that the RBF kernel achieved the highest accuracy of 97.80% and a low false positive rate of 0.12%. They also highlighted the importance of selecting appropriate kernel functions and tuning the SVM parameters for optimal performance.

The choice of a machine learning classifier depends on various factors, such as the dataset's characteristics, the nature of the attacks, and the computational resources available. Buczak and Guven [6] provide a comprehensive survey of machine learning methods for cyber security intrusion detection. They discuss the strengths and limitations of different classifiers and emphasise the importance of selecting appropriate features, handling imbalanced data, and evaluating the performance of classifiers using relevant metrics.

Comparing the performance of different classifiers on multiple datasets provides valuable insights into their generalisation capabilities and transferability. Training classifiers on one dataset and testing them on another helps assess how well the learned patterns and features can be applied to detect intrusions in different network environments. This approach helps understand the robustness and adaptability of the classifiers across various scenarios.

2.3 Explainable AI Techniques

While machine learning classifiers have shown promising results in network intrusion detection, their decision-making process is often considered a 'black box', lacking transparency and interpretability. Explainable AI techniques aim to bridge this gap by providing insights into the reasoning behind the predictions made by these models.

SHAP (SHapley Additive exPlanations) [16] is a popular technique for model interpretation. It is based on the concept of Shapley values from cooperative game theory and provides a unified framework for explaining the output of any machine learning model. SHAP assigns importance scores to each feature, indicating their contribution to the model's prediction for a specific instance. By applying SHAP to trained classifiers, researchers can identify the key features that contribute to detecting particular types of attacks.

Warnecke et al. [29] evaluated various explanation methods for deep learning-based intrusion detection systems, including SHAP They applied SHAP to a convolutional neural network (CNN) trained on the NSL-KDD dataset and analysed the importance of different features in the model's predictions. They demonstrated that SHAP can provide meaningful insights into the decision-making process of deep learning models and help identify the most influential features for detecting specific attack types.

Amarasinghe et al. [3] employed SHAP to interpret the predictions of a deep learning-based NIDS.They trained a deep neural network on the NSL-KDD dataset and applied SHAP to explain the model's predictions. They analysed the SHAP values to understand the impact of different features on the model's decisions and identified the most discriminative features for detecting specific types of attacks. Their study highlights the potential of SHAP in providing interpretable explanations for deep learning-based NIDS.

Mane and Rao [18] used SHAP to explain the predictions of a random forest classifier for network intrusion detection. They trained the classifier on the NSL-KDD dataset and applied SHAP to interpret the model's predictions. They visualised the SHAP values to understand the contribution of each feature towards the model's output and identified the most critical features for detecting various types of network attacks. Their study demonstrates the effectiveness of SHAP in providing interpretable explanations for ensemble learning methods like random forests.

The insights gained from explainable AI techniques can help validate the reliability of the trained models, identify potential biases or limitations in the datasets, and guide future improvements in feature engineering and model development. Moreover, these insights can be valuable for network security analysts and practitioners in understanding the key factors contributing to detecting specific attacks and developing more targeted defence strategies.

Chapter 3

Relevant Work

This chapter comprehensively reviews related work 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 for intrusion detection tasks. Furthermore, it explores the use of explainable AI techniques, such as SHAP, to provide insights into the decision-making process of machine learning models in the context of intrusion detection. The chapter also discusses the challenges and limitations associated with using synthetic datasets and the importance of considering dataset biases and the dynamic nature of network traffic when developing and evaluating intrusion detection systems.

3.1 Botnet Detection using CTU13 Dataset

Several studies have utilized the CTU13 dataset for developing and evaluating botnet detection systems. Chowdhury et al. [7] proposed a graph-based approach for botnet detection using the CTU13 dataset. They constructed a graph representation of the communication patterns among botnet-infected hosts and applied graph analysis techniques to identify botnets. The authors extracted features such as in-degree, out-degree, and betweenness centrality from the graph and used them to train a Random Forest classifier. Their approach achieved an accuracy of 99.86% in detecting botnets, demonstrating the potential of leveraging graph-based features for botnet detection. In contrast, this dissertation investigates the transferability of models trained on the CICIDS2017 dataset to detect botnet attacks in the CTU13 dataset, exploring

the generalization capabilities of machine learning classifiers across different datasets.

Pektaş and Acarman [21] applied deep learning techniques to the CTU13 dataset for botnet detection. They used a convolutional neural network (CNN) to learn discriminative features from raw network traffic data. The authors converted the traffic data of the CTU13 dataset into grayscale images through preprocessing and then fed them as input to the CNN. The proposed model achieved an accuracy of 99.99% and a false positive rate of 0.01% in detecting botnet traffic, showcasing the effectiveness of deep learning in capturing complex patterns and behaviours associated with botnets. While deep learning approaches have shown promising results, this dissertation explores the potential of traditional machine learning classifiers, such as Random Forest and Support Vector Machines, in detecting botnet attacks and investigates their transferability across datasets.

3.2 Intrusion Detection using CICIDS2017 Dataset

Many researchers have widely used the CICIDS2017 dataset to evaluate intrusion detection systems. Ustebay et al. [28] proposed an intrusion detection system based on a multi-layer perceptron (MLP) classifier using the CICIDS2017 dataset. They performed extensive preprocessing on the dataset, including handling missing values, encoding categorical features, and scaling numerical features. The authors also applied recursive feature elimination with random forest to select the most relevant features for intrusion detection. Their MLP-based approach achieved an accuracy of 97.29% and an F1-score of 97.30%, demonstrating the potential of neural network-based models for network intrusion detection. In this dissertation, we leverage the CICIDS2017 dataset to train machine learning classifiers and investigate their transferability to detect botnet attacks in the CTU13 dataset, focusing on the generalization capabilities of the learned patterns and features.

Aksu and Aydin [2] conducted a comparative study of different machine learning algorithms for network intrusion detection using the CICIDS2017 dataset. They evaluated the performance of decision trees, random forests, and support vector machines. The authors preprocessed the dataset by removing redundant records, handling missing values, and applying normalization techniques. They also performed feature selection using information gain and correlation-based methods. The experimental results showed that random forests outperformed other algorithms, achieving an accuracy of 99.77% and a false positive rate of 0.04%, highlighting the effectiveness of ensemble learning methods for intrusion detection. This dissertation takes

inspiration from their findings and employs Random Forest as one of the classifiers to investigate its transferability across datasets.

3.3 Random Forest for Intrusion Detection

Farnaaz and Jabbar [10] utilized Random Forest for intrusion detection tasks and presented a model based on it. They evaluated the model's performance on the NSL-KDD dataset. They performed feature selection using the Chi-square test to identify the most relevant features for intrusion detection, then proceeded to train the Random Forest classifier using the selected features. The proposed model achieved an accuracy of 99.67% and a false positive rate of 0.06%, demonstrating Random Forest's effectiveness in detecting various types of network attacks. While their work focused on a single dataset, this dissertation explores the transferability of Random Forest classifiers trained on the CICIDS2017 dataset to detect botnet attacks in the CTU13 dataset, investigating the generalization capabilities of the learned patterns across different network environments.

Belouch et al. [5] applied Random Forest to the CICIDS2017 dataset for network intrusion detection. They compared the performance of Random Forest with other machine learning algorithms, including Decision Tree, Naive Bayes, and k-Nearest Neighbors. The authors pre-processed the dataset by removing redundant records and handling missing values. They also analyzed the importance of different features in the dataset using the Random Forest feature importance measure. The experimental results showed that Random Forest outperformed other algorithms, achieving an accuracy of 99.98% and a false positive rate of 0.01%, further validating the effectiveness of Random Forest for intrusion detection. This dissertation builds upon their findings by employing Random Forest as one of the classifiers. It investigates its transferability to detect botnet attacks in a different dataset, exploring the robustness and adaptability of the learned patterns.

3.4 Support Vector Machines for Intrusion Detection

Intrusion detection tasks have extensively utilized Support Vector Machines (SVM). Kabir et al. [14] proposed an SVM-based intrusion detection system and evaluated its performance on the NSL-KDD dataset. They used a genetic algorithm for feature selection to identify the most discriminative features for intrusion detection. The authors also optimized the SVM parameters

using grid search to improve the model's performance. The proposed system achieved an accuracy of 99.91% and a detection rate of 99.93%, showcasing SVM's effectiveness in detecting various types of network attacks. In contrast to their work, this dissertation focuses on the transferability of SVM classifiers trained on the CICIDS2017 dataset to detect botnet attacks in the CTU13 dataset, exploring the generalization capabilities of the learned patterns across different datasets.

Teng et al. [27] applied SVM with different kernel functions for intrusion detection on the CICIDS2017 dataset. They compared the performance of linear, polynomial, and radial basis function (RBF) kernels. The authors preprocessed the dataset by removing redundant records and scaling the features. They also applied principal component analysis (PCA) for dimensionality reduction. The experimental results showed that the RBF kernel achieved the highest accuracy of 97.80% and a low false positive rate of 0.12%, highlighting the importance of selecting appropriate kernel functions and tuning the SVM parameters for optimal performance. This dissertation takes inspiration from their findings and employs SVM as one of the classifiers to investigate its transferability in detecting botnet attacks across different datasets, exploring the robustness and adaptability of the learned patterns.

3.5 Explainable AI Techniques for Intrusion Detection

Explainable AI techniques have gained attention in intrusion detection to provide insights into the decision-making process of machine learning models. Warnecke et al. [29] evaluated various explanation methods for deep learning-based intrusion detection systems, including SHAP They applied SHAP to a convolutional neural network (CNN) trained on the NSL-KDD dataset and analyzed the importance of different features in the model's predictions. The authors demonstrated that SHAP can provide meaningful insights into the decision-making process of deep learning models and help identify the most influential features for detecting specific attack types. This dissertation takes inspiration from their approach and employs SHAP to interpret the predictions of Random Forest and SVM classifiers trained on the CICIDS2017 dataset, providing insights into the transferability of the learned patterns and the most relevant features for detecting botnet attacks in the CTU13 dataset.

Amarasinghe et al. [3] employed SHAP to interpret the predictions of a deep learning-based network intrusion detection system. They trained a deep neural network on the NSL-KDD dataset and applied SHAP to explain the model's predictions. The authors analyzed the SHAP

values to understand the impact of different features on the model's decisions and identified the most discriminative features for detecting specific types of attacks. Their study highlights the potential of SHAP in providing interpretable explanations for deep learning-based intrusion detection systems. In this dissertation, we leverage SHAP to interpret the predictions of traditional machine learning classifiers, such as Random Forest and SVM, and investigate the transferability of the learned patterns and the most relevant features for detecting botnet attacks across different datasets.

Mane and Rao [18] used SHAP to explain the predictions of a random forest classifier for network intrusion detection. They trained the classifier on the NSL-KDD dataset and applied SHAP to interpret the model's predictions. The authors visualized the SHAP values to understand the contribution of each feature towards the model's output and identified the most critical features for detecting various types of network attacks. Their study demonstrates the effectiveness of SHAP in providing interpretable explanations for ensemble learning methods like random forests. This dissertation builds upon their findings by employing SHAP to interpret the predictions of Random Forest and SVM classifiers trained on the CICIDS2017 dataset. It investigates the transferability of the learned patterns and the most relevant features for detecting botnet attacks in the CTU13 dataset.

3.6 Challenges and Limitations

Intrusion detection system developers and evaluators frequently employ the CTU13 and CICIDS2017 datasets. Nonetheless, it is necessary to recognize their limitations. Sommer and Paxson [26] have addressed the challenges of utilizing synthetic datasets for network intrusion detection. They contend that these datasets may only partially capture the intricacy and variety of network traffic and require additional contextual information. Furthermore, the authors stress the necessity of more authentic and comprehensive datasets that consider the operational aspects and deployment hurdles of intrusion detection systems. This dissertation acknowledges these limitations and emphasizes the importance of considering dataset biases and the representativeness of training data when developing and evaluating intrusion detection systems.

Furthermore, Sommer and Paxson [26] highlight the importance of using representative and unbiased datasets for training machine learning models in network intrusion detection. They discuss the challenges posed by the dynamic nature of network traffic and the constant evolution of attack patterns, which can impact the long-term performance of the trained models.

Buczak and Guven [6] provide a comprehensive survey of machine learning and data mining methods for cyber security intrusion detection. They discuss the considerations and challenges in applying machine learning techniques to network intrusion detection, including selecting appropriate algorithms, feature engineering, and model evaluation. This dissertation considers these challenges and aims to address them by investigating the transferability of machine learning models across different datasets and employing explainable AI techniques to gain insights into the decision-making process.

3.7 Summary and Positioning

This chapter reviews relevant work in machine learning-based network intrusion detection, focusing on botnet detection using the CTU13 dataset, intrusion detection using the CICIDS2017 dataset, and the application of Random Forests and Support Vector Machines for intrusion detection tasks. While previous studies have demonstrated the effectiveness of various machine learning algorithms in detecting network attacks, this dissertation explores a novel aspect by investigating machine learning models' transferability and generalization capabilities across different datasets.

This dissertation takes inspiration from existing approaches that have utilized Random Forest and Support Vector Machines for intrusion detection and applies these classifiers to the CICIDS2017 dataset. However, this work goes beyond previous studies by evaluating the performance of these classifiers in detecting botnet attacks in the CTU13 dataset, which represents a different network environment. This dissertation aims to assess the robustness and adaptability of the learned patterns and features across datasets, simulating a real-world scenario where a model is trained on one dataset and deployed to detect intrusions in the wild.

Furthermore, while the use of explainable AI techniques, such as SHAP, has been explored in previous studies for interpreting the predictions of intrusion detection models, this dissertation employs SHAP in a novel context. Specifically, it leverages SHAP to provide insights into the transferability of learned patterns and the most relevant features for detecting botnet attacks when applying classifiers trained on the CICIDS2017 dataset to the CTU13 dataset. This aspect of the research distinguishes it from previous studies, as it focuses on understanding the transferability of the learned patterns and the key characteristics that distinguish malicious and benign traffic across different datasets.

In summary, this dissertation builds upon existing research by investigating machine learning models' transferability and generalization capabilities for network intrusion detection, emphasizing the importance of dataset representativeness and biases. By employing Random Forest and Support Vector Machines classifiers and leveraging explainable AI techniques in the context of cross-dataset transferability, this work aims to provide valuable insights into the robustness and adaptability of these algorithms in detecting botnet attacks across different datasets, contributing to the development of more effective and practical intrusion detection solutions.

Chapter 4

Specification & Design

This chapter outlines the experiments conducted in this dissertation, focusing on three classifiers: the Dummy Classifier, the Random Forest Classifier, and the Support Vector Machine Classifier. The experimental setup has been designed based on a thorough literature review to address the following research questions:

RQ1 How well do machine learning models trained on one dataset transfer and generalise to another dataset 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 types of attacks across different datasets?

The CTU13 and CICIDS2017 datasets are chosen for this dissertation as they contain specific types of attacks and possess distinct properties that make them suitable for addressing the research questions. CTU13 consists of real botnet traffic captured in a controlled environment [11], while CICIDS2017 is a more recent and comprehensive dataset designed for evaluating network intrusion detection systems, containing a wide range of modern attacks [25]. By training models on CICIDS2017 and testing them on CTU13, this dissertation aims to assess the transferability of learned patterns and features from one dataset to another, addressing RQ1.

The scripts 2 and 1 are utilised during pre-processing to ensure uniformity and coherence across all datasets. These scripts facilitate mapping dataset features to a standardised naming convention and removing features that only appear in one dataset (which would not be transferrable between datasets), promoting equitable comparisons and analysis. This approach effectively addresses RQ2 by mitigating the impact of dataset biases and representativeness.

This dissertation comprehensively analyses network intrusion detection classifiers, utilising a range of evaluation metrics, including accuracy, recall, precision, the confusion matrix, and F1 score. Additionally, the SHAP library interprets the classifiers' predictions, offering valuable insights into their decision-making process. An essential aspect of this dissertation is the exploration of RQ3, where we utilise explainable AI techniques to uncover the transferability of learned patterns and identify critical features for detecting specific types of attacks across multiple datasets.

A common practice in machine learning is to split data into training and testing sets using a 60/40 ratio. This approach ensures a fair data distribution between the two sets, which helps prevent overfitting and promotes unbiased testing. This partitioning ratio has been widely accepted in the field as it balances training and assessing models, as noted in a research survey by Buczak et al. [6].

Alternative designs and dataset selections were considered, such as using neural network-based classifiers (e.g., MLP and CNN) and other datasets. However, the focus on interpretability using SHAP and the comparative study by Belouch et al. [5] influenced the decision to use Random Forest and SVM classifiers. The combination of CICIDS2017 and CTU13 datasets aligns with the research objectives of assessing the effectiveness of machine learning classifiers in detecting botnet attacks and understanding the challenges and limitations of transferring learned patterns across datasets.

4.1 Experiments

The experiments we run consist of three classifiers: a Dummy Classifier, a Random Forest Classifier, and a Support Vector Machine Classifier. Each classifier is trained on the CICIDS2017 dataset and tested on both the CICIDS2017 and CTU13 datasets to assess its transferability and generalisation capabilities.

4.1.1 Data Pre-processing

During the pre-processing stage, the datasets undergo relabeling using the scripts 2 and 1 to ensure consistency and compatibility. Subsequently, the CICIDS2017 dataset is processed to generate binary and multi-class classification tasks. In contrast, we only process the CTU13 dataset to generate binary classification tasks, aiming to detect botnet attacks and benign traffic because it only includes botnet attacks, whereas CICIDS2017 contains various types of attacks.

4.1.2 Training and Testing Split

We partitioned the datasets into training and testing sets to ensure precise results with a 60/40 ratio. Our team trained the classifiers using the training set from the CICIDS2017 dataset and subsequently tested them on both the testing set of the same dataset and the full CTU13 dataset. This approach allows us to evaluate the classifiers' effectiveness on the dataset we trained them on and their performance on a separate dataset.

4.1.3 Evaluation Metrics

Several metrics evaluate the classifiers' performance, including confusion matrix, accuracy, precision, recall, and F1 score. These metrics comprehensively assess the classifiers' effectiveness in detecting network intrusions and their ability to transfer learned patterns across datasets.

4.1.4 Explainable AI

The SHAP library is utilised to interpret the predictions made by the trained classifiers and offer valuable insights into their decision-making process. We compute SHAP values for the Random Forest and Support Vector Machine classifiers to pinpoint the most significant features that aid in detecting particular types of attacks. This aspect of explainability is critical in comprehending the transferability of learned patterns and the essential attributes that differentiate between malicious and benign traffic, even across diverse datasets.

4.2 Classifier Configurations

4.2.1 Dummy Classifier

Purpose: The Dummy Classifier serves as a baseline for evaluating the performance of the more advanced classifiers. It randomly assigns labels to the data, providing a reference point for comparison.

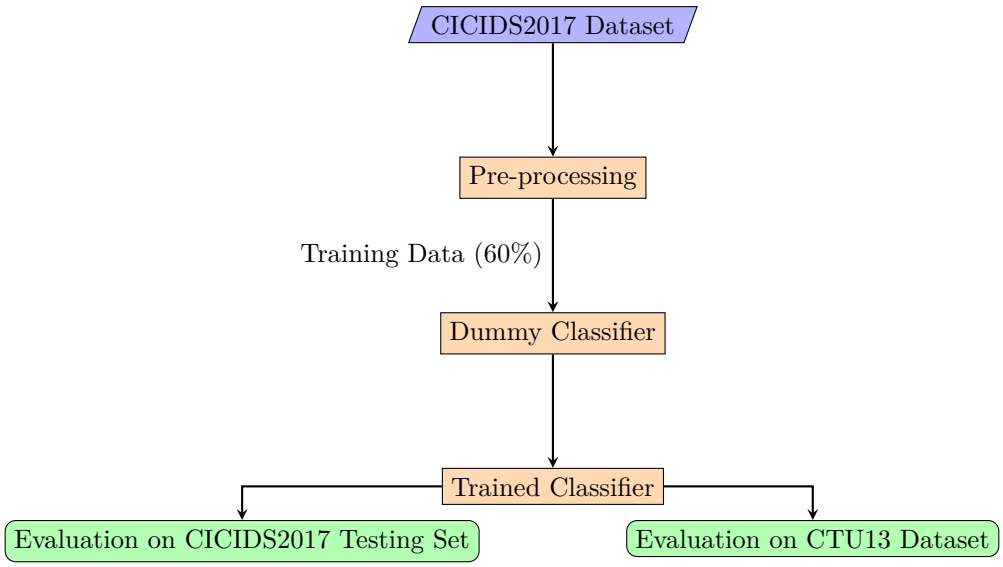

Figure 4.1: Dummy Classifier Configuration

4.2.2 Random Forest and SVM Classifiers

Purpose: The Random Forest and Support Vector Machine (SVM) classifiers are employed to evaluate the performance of more advanced machine learning models in detecting network intrusions. These classifiers are trained on the CICIDS2017 dataset and tested on both the CICIDS2017 testing set and the CTU13 dataset to assess their transferability and generalisation capabilities.

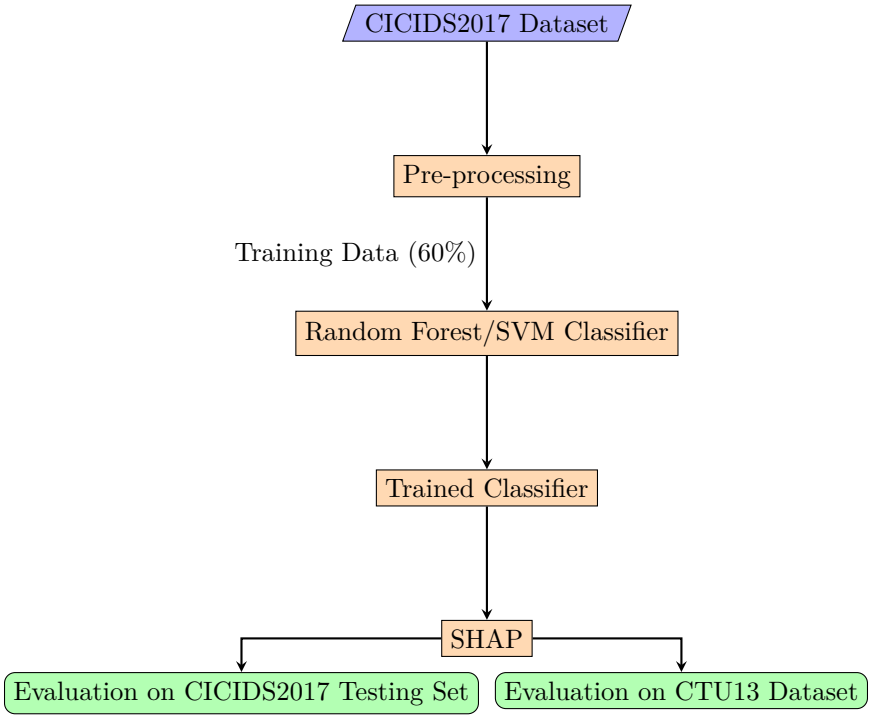

Figure 4.2: Random Forest and SVM Classifier Configuration

4.3 Transferability Evaluation

In evaluating the classifiers' transferability from the CICIDS2017 dataset to the CTU13 dataset, we emphasised the significance of measuring their capability to generalise the learned patterns. To assess this capacity, the classifiers were rigorously trained on the CICIDS2017 dataset and subsequently tested on both its testing set and the entirety of the CTU13 dataset.

To effectively evaluate the transferability of our classifiers, we utilised the following performance metrics, accompanied by their respective mathematical expressions:

- **Accuracy**. Accuracy quantifies the proportion of true predictions, encompassing both positive and negative outcomes, in relation to the total number of observations. It is calculated as:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{4.1}$$

  where TP, TN, FP, and FN represent the counts of true positives, true negatives, false positives, and false negatives, respectively.

- **Recall**. Recall, or the True Positive Rate, measures the proportion of actual positive

cases correctly identified by the model, given by:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.2}$$

- **Precision**. Precision assesses the fraction of correctly predicted positive observations out of all predicted positives, computed as:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.3}$$

- **F1 Score**. The F1 score integrates precision and recall into a single metric by taking their harmonic mean, thus providing a balanced measure of the classifier's precision and recall, calculated as:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.4}$$

Calculating these metrics for each classifier on both the CICIDS2017 testing set and the CTU13 dataset enables evaluation of the transferability of the learned patterns. A classifier that maintains high-performance metrics on both datasets demonstrates good transferability, whereas a significant drop in performance on the CTU13 dataset indicates limited transferability.

Furthermore, the SHAP values generated through the explainable AI analysis provide valuable insights into the essential attributes that facilitate the identification of distinct attack types across a range of datasets. By carefully examining the consistency of the most significant features across these datasets, we can assess the applicability of the acquired patterns. The SHAP values obtained through the explainable AI analysis also offer crucial information on the features that contribute the most to detecting particular attack types across various datasets. By thoroughly examining the top-ranked features' consistency across datasets, we can further evaluate the transferability of the learned patterns.

4.4 Summary

This chapter presents the specification and design of the experiments conducted in this dissertation, focusing on three classifiers: a Dummy Classifier, a Random Forest Classifier, and a Support Vector Machine Classifier. The experimental setup addresses three key research questions related to the transferability and generalisation of machine learning models, the impact of

dataset biases, and the insights provided by explainable AI techniques in the context of network intrusion detection.

We chose the CTU13 and CICIDS2017 datasets for their distinct properties and the presence of specific types of attacks. The data pre-processing step involves relabeling the datasets to ensure consistency and compatibility, while the training and testing split follows a 60/40 ratio. The classifiers are trained on the CICIDS2017 dataset and tested on both the CICIDS2017 testing set and the entire CTU13 dataset to evaluate their transferability.

We assess the classifiers' performance using various evaluation metrics, including accuracy, precision, recall, and F1 score. The SHAP library is employed to interpret the predictions of the trained classifiers and identify the most relevant features contributing to the detection of specific types of attacks across different datasets.

The evaluation of transferability is of immense importance to this dissertation. We utilise quality metrics and SHAP values to assess the classifiers' proficiency in transferring learned patterns from CICIDS2017 to CTU13. By analysing the consistency of the top-ranking features and comparing the classifiers' performance on both datasets, we can gauge the transferability of the learned patterns.

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 the CICIDS2017 dataset [25] and tested on both the CICIDS2017 and CTU13 [11] datasets to assess transferability and generalisation capabilities.

Initially, we set up the experiments using sci-kit-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 [22]. The CUML library provides GPU-accelerated machine learning algorithms, enabling faster training and inference times. By leveraging the power of GPUs, the training process can be significantly accelerated, 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, 27], which are common in network traffic classification. We employ the SHAP library [16] for explainability due to its ability to provide insights into the decision-making process of machine learning models and identify important features.

5.1.1 Data Pre-processing

Proper data preprocessing is crucial for ensuring the effectiveness of the trained classifiers. The relabelling scripts, `relabelCICIDS2017.py` (Section 2) and `relabelCTU13.py` (Section 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. 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 address these issues through careful data cleaning, imputation, and feature mapping techniques to ensure the datasets' compatibility and integrity.

5.1.2 Experiment 1: Dummy Classifier

The Dummy Classifier from the scikit-learn library [20] serves as a baseline for evaluating the performance of the more advanced classifiers. It predicts the most frequent class in the training data. We train three Dummy Classifiers: 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.

5.1.3 Experiment 2: Random Forest Classifier

Random Forest, an ensemble learning method constructing multiple decision trees [13], 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).

Three Random Forest Classifiers are trained (`trainRandomForest.ipynb`, Section 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. The SHapley Additive exPlanations (SHAP) library [16] is employed to explain the classifiers' predictions and identify the essential features.

5.1.4 Experiment 3: Support Vector Machine Classifier

Support Vector Machines (SVMs) effectively handle large, high-dimensional, non-linear data [8, 23]. They have been successfully applied to network intrusion detection tasks [15, 27].

The experimental setup for the SVM Classifiers (`trainSVM.ipynb`, Section 5) mirrors that of the Random Forest Classifiers: we train three SVMs on the CTU13 and CICIDS2017 datasets, evaluated on their respective datasets, and tested on the other dataset. SHAP is used to interpret the SVM predictions and identify important features.

5.1.5 Testing and Evaluation

A comprehensive testing and evaluation strategy is employed to ensure the robustness and reliability of the results. We assess the performance of the classifiers using various metrics, including accuracy, precision, recall, and F1 score. We calculate these metrics for each classifier on their respective test sets and when evaluated on the other dataset for transferability analysis.

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.

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.

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, 27]. Using CUML's GPU-accelerated models significantly reduces training times and improves scalability, enabling the efficient handling of large datasets.

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. The computational complexity of the SHAP explanations may also pose challenges when dealing with large-scale datasets, even with GPU acceleration.

5.2 Package Implementation

This section elucidates the implementation details of the software packages developed as part of this project, focusing on preprocessing scripts for the CTU13 and CICIDS2017 datasets, training modules for various classifiers, and a data visualization tool. Each subsection is dedicated to a specific script or Jupyter Notebook, detailing its purpose, functionalities, and contribution to the overarching project goals.

5.2.1 relabelCTU13.py

Purpose: This Python script (1) is tasked with preprocessing the CTU13 dataset, ensuring its compatibility with the analysis framework. It specifically focuses on feature normalization, 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 the need for dataset-specific adjustments.

5.2.2 relabelCICIDS2017.py

Purpose: Similar to `relabelCTU13.py`, this script (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 (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 (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 dataset and an alternative dataset to examine transferability.
- *Feature Importance Analysis*: Utilizes 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 (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 generalizability.
- *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 (6) provides visualizations of dataset characteristics and model performance metrics

Chapter 6

Evaluation

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

6.1 Performance Evaluation

As described in section (4.3), the classifiers are evaluated on their respective datasets and then on the other dataset to assess transferability. The performance metrics tested include accuracy, precision, recall, and F1 score. The results are presented in Figures 6.1 and 6.2.

6.1.1 Performance on the Same Dataset

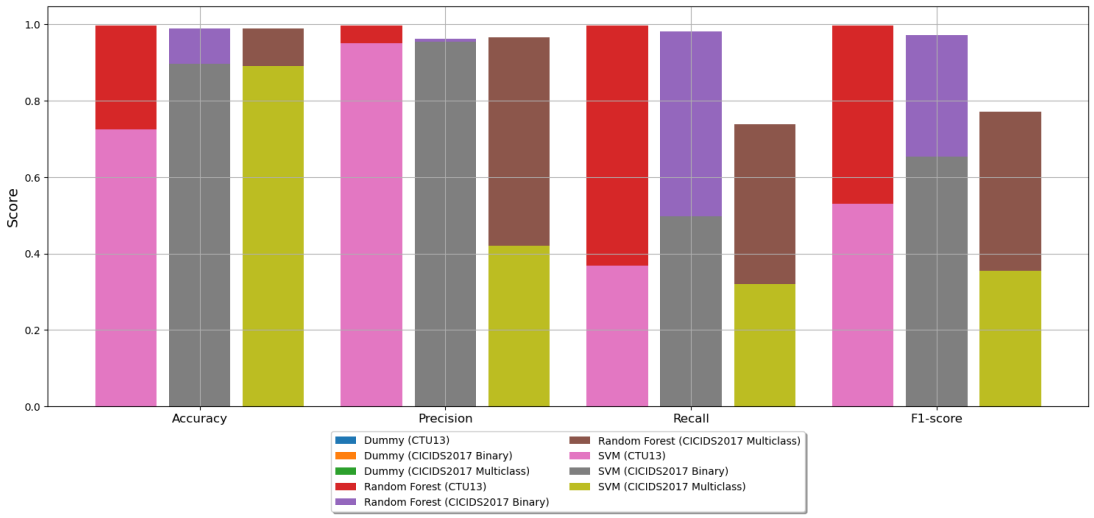

Figure 6.1: Classifiers' performance on the same datasets.

When testing the classifiers' performance on the same dataset, they were trained on; we note some promising results. The Random Forest model achieves an accuracy of 0.99, precision of 0.99, recall of 0.99, and F1 score of 0.99 on the CTU13 dataset. The model also performs well on the CICIDS2017 dataset, with an accuracy of 0.99, precision of 0.96, recall of 0.98, and F1 score of 0.97 when focusing on binary classification. These results indicate that the classifiers detect malicious traffic on their respective datasets effectively.

In the multi-class classification, the Random Forest model achieves a respectable F1 score of 0.78, with an accuracy of 0.99, precision of 0.90, and recall of 0.76. While these scores would likely need to be higher to deploy the multi-class model in a real-world scenario, they outperform the dummy classifier's baseline score of 0.66, proving that the model learns from the data.

When considering the performance of the classifiers on the same dataset, all three Random Forest classifiers outperform the dummy classifier, showing that, at least in the context of the datasets used, the classifiers are learning from the data and making accurate predictions.

The SVM classifier's performance was worse than the Random Forest's; it also ended up being worse than the dummy classifier's performance. The SVM classifier trained on CTU13 achieves an accuracy of 0.73, precision of 0.96, recall of 0.37, and an F1 score of 0.53 when tested on CTU13. The binary model trained on CICIDS2017 achieves an accuracy of 0.73, precision of 0.96, recall of 0.37, and an F1 score of 0.65 when tested on CICIDS2017. The multi-class SVM

model trained on CICIDS2017 achieves an accuracy of 0.89, precision of 0.42, recall of 0.32, and an F1 score of 0.35 when tested on CICIDS2017, indicating that the SVM classifiers are not as effective as the Random Forest classifiers at detecting malicious traffic on the datasets used, at least with the current implementation (we will discuss improvements in the conclusion chapter8).

6.1.2 Performance on Different Datasets

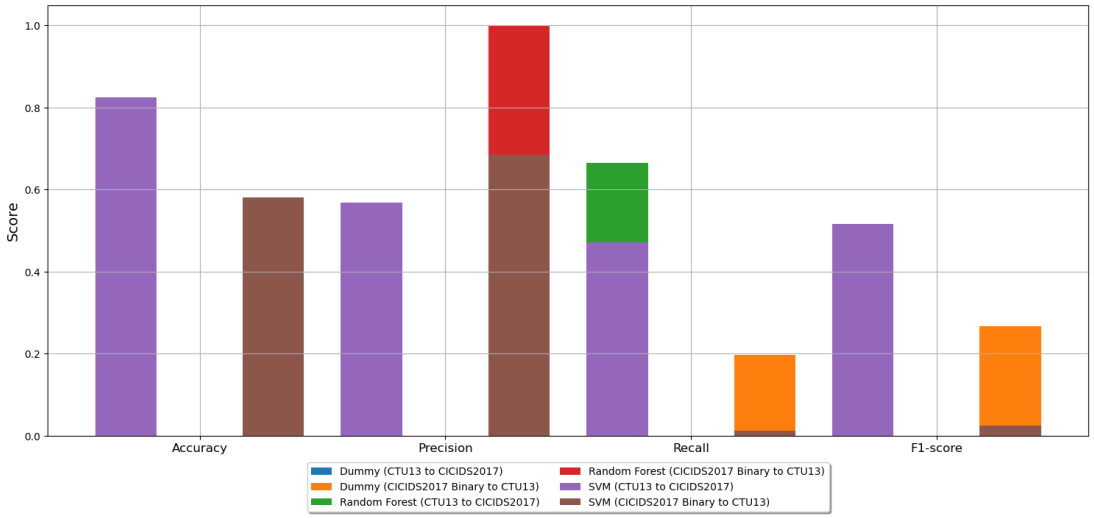

Figure 6.2: Classifiers' Performance on across datasets.

6.2 Feature Importance Analysis

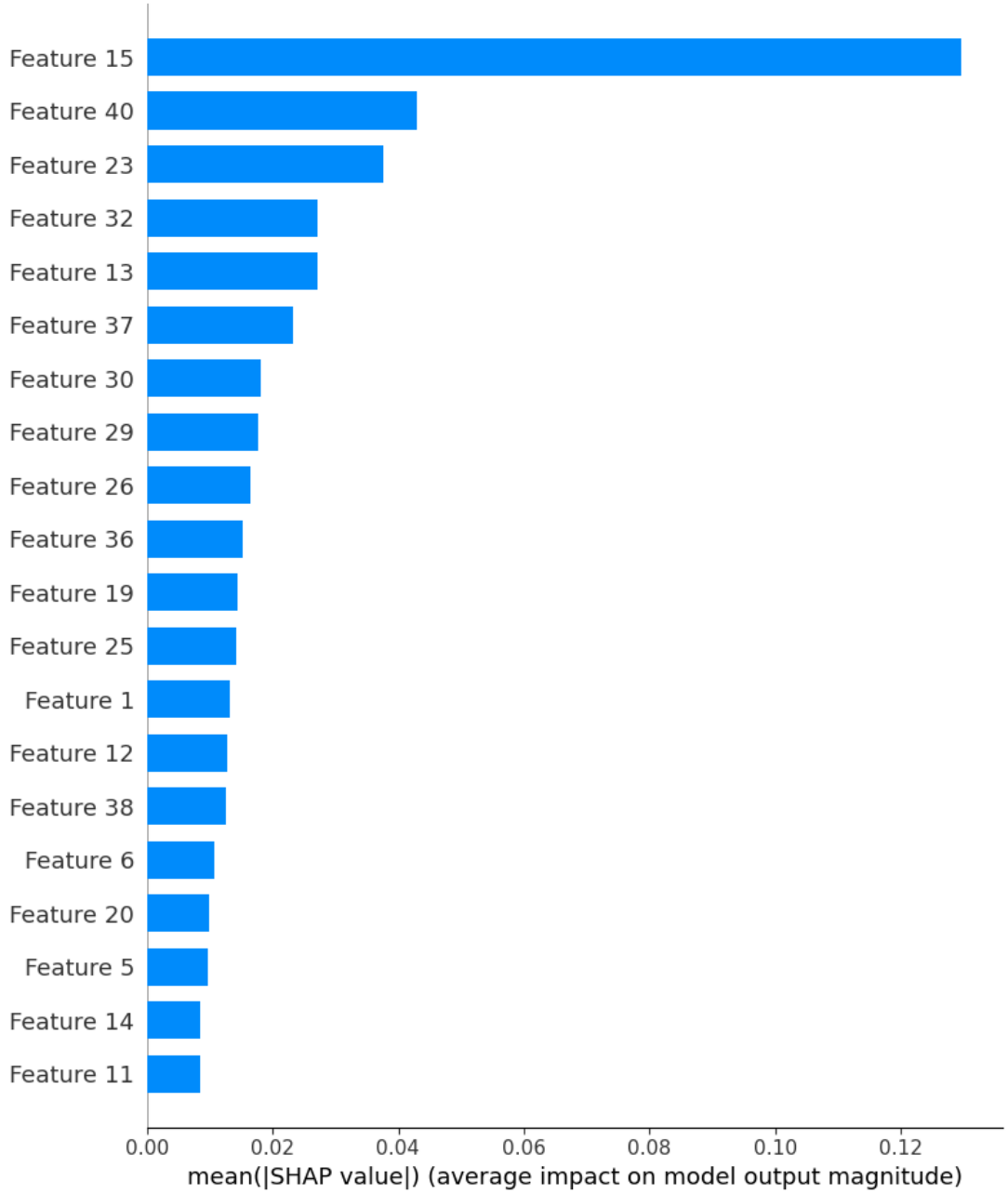

Figure 6.3: SHAP summary plot for the CTU13 random forest model tested against CTU13.

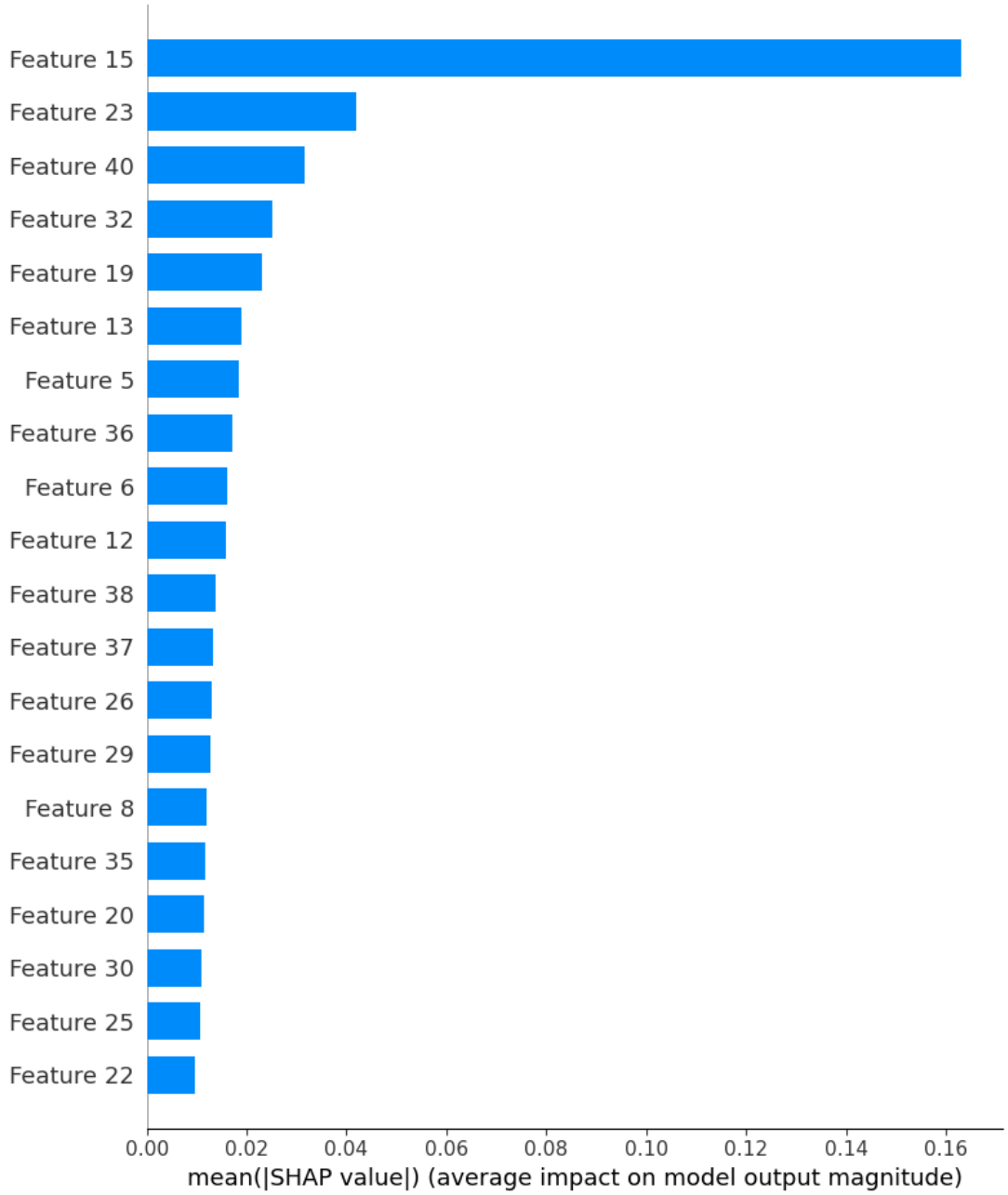

Figure 6.4: SHAP summary plot for the CTU13 random forest model tested against CICIDS2017.

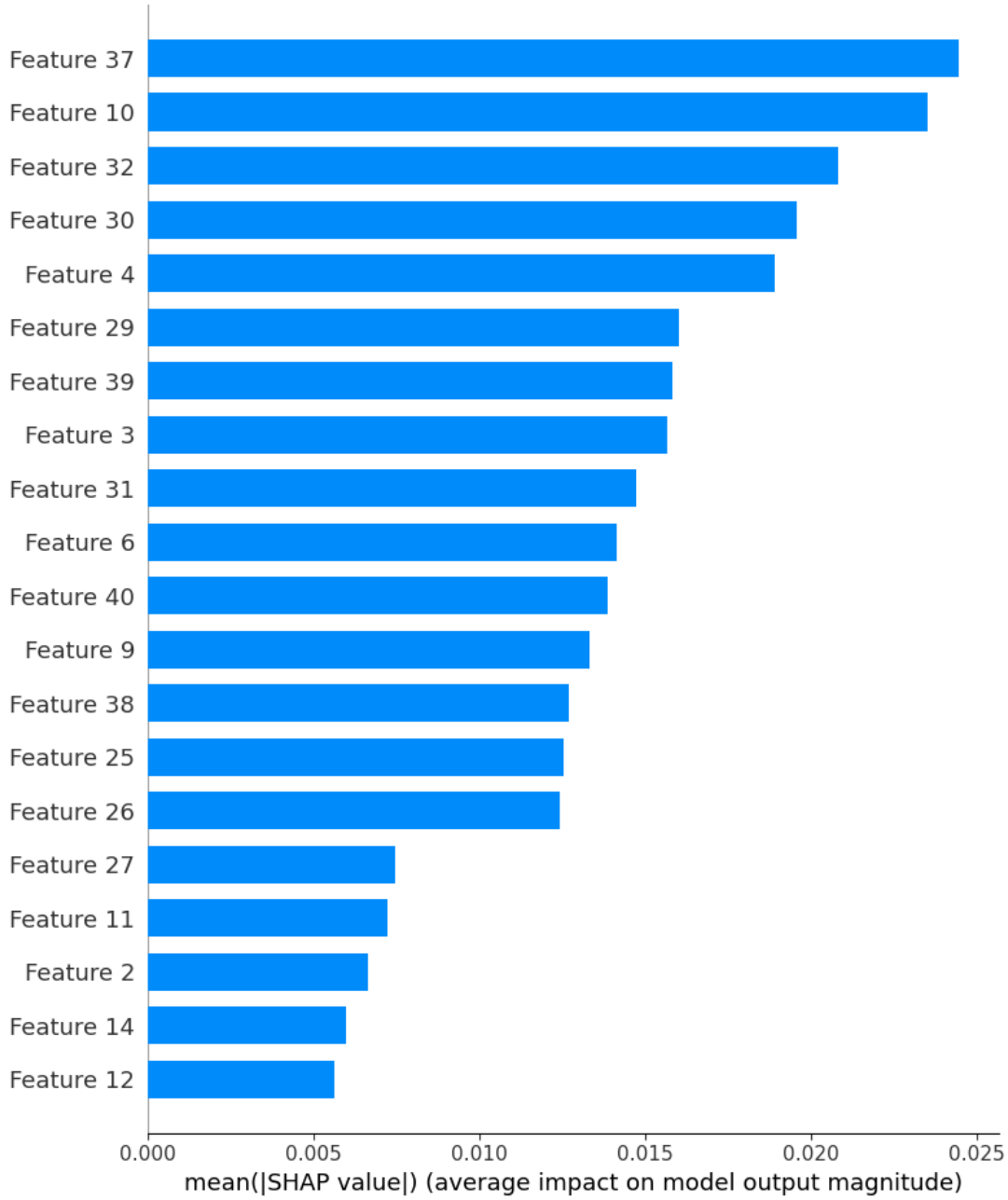

Figure 6.5: SHAP summary plot for the CICIDS2017 random forest model tested against CICIDS2017.

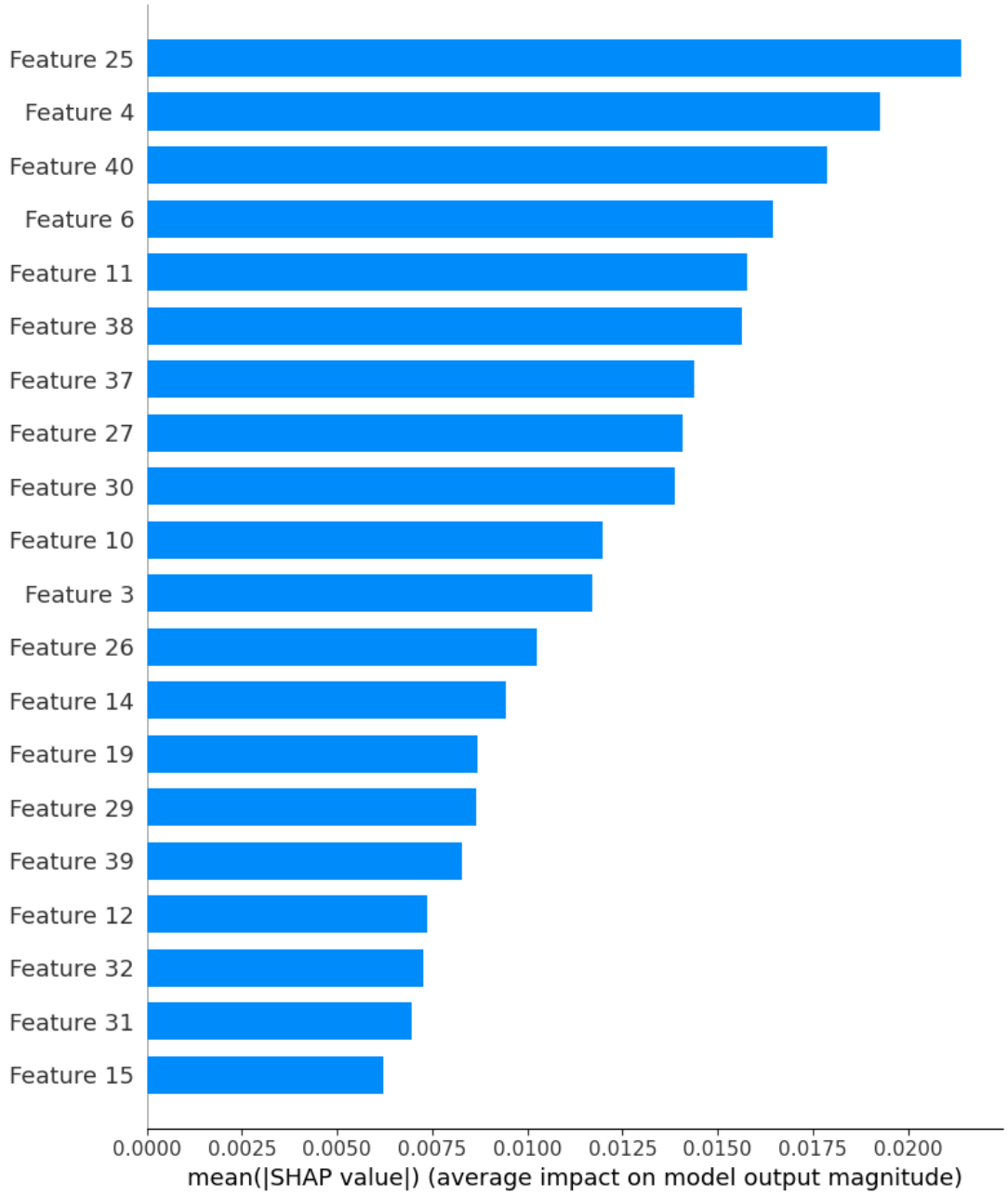

Figure 6.6: SHAP summary plot for the CICIDS2017 random forest model tested against CTU13.

We analyse the SHAP (SHapley Additive exPlanations) values to identify the most influential features in the random forest classifiers' decisions. Figure 6.3 shows that when we test the CTU13 model on CTU13 data, features like Bwd Packet Length Mean and Flow IAT Min are important. However, when applied to CICIDS2017 (Figure 6.4), the model relies more on features like Flow Duration and Flow IAT Max, indicating a shift in feature importance when

transferring to a new dataset.

The CICIDS2017 model tested on its data (Figure 6.5) attributes importance to Fwd Packet Length Max and Average Packet Size. However, when evaluated on CTU13 (Figure 6.6), the model gives more weight to Fwd Header Length and Flow Duration. These variations in feature importance underscore the challenges in directly transferring models between datasets and the need for adaptive strategies.

Chapter 7

Legal, Social, Ethical and Professional Issues

This chapter evaluates potential legal, social, ethical, and professional issues that may arise during the research process. It also explains how the research adheres to the British Computing Society's Code of Conduct.

7.1 Legal Considerations

The research utilises two publicly available datasets: CTU13[11] and CICIDS2017[25]. These datasets are accessible for research purposes and are not subject to legal restrictions. The research does not involve the use of personal data, as CICIDS2017 is a synthetic dataset, and CTU13 has excluded passive network flows that could potentially contain sensitive information. By avoiding the use of personal data, the research ensures compliance with the UK Data Protection Act 2018.

7.2 Ethical & Social Considerations

This research evaluates the transferability and generalisation of machine learning models across different datasets. The study does not involve human subjects; the datasets used are publicly available. Consequently, no direct ethical or social issues are associated with this research. However, it is vital to consider the broader implications of developing effective network intru-

sion detection systems. By enhancing the ability to detect and prevent cyber attacks, this research improves individuals' and organisations' overall security and privacy. Developing robust and transferable machine learning models for intrusion detection can help protect sensitive information, prevent data breaches, and mitigate the risks associated with malicious activities in networked environments.

On the other hand, using explainable AI techniques, such as SHAP, in network intrusion detection raises some ethical concerns. While SHAP provides valuable insights into the decision-making process of machine learning models, it can also potentially expose the most influential features for detecting specific types of attacks. If this information falls into the wrong hands, malicious actors could exploit it to evade detection by manipulating or spoofing the relevant features. Therefore, ensuring that the insights gained from SHAP are handled with utmost care and not disclosed to unauthorised parties is crucial. Proper security measures should be in place to protect the confidentiality and integrity of the explainability results.

Furthermore, developing highly effective intrusion detection systems may also have unintended consequences. For instance, it could lead to an overreliance on automated systems for security, potentially diminishing the role of human expertise and judgment. It is essential to balance leveraging machine learning models' capabilities and maintaining human oversight and intervention in the decision-making process. When considering intrusion detection systems, carefully evaluating the potential for false positives is crucial. These can cause unnecessary disruptions and result in the misallocation of resources, making it essential to take proactive steps to avoid them. Adequate safeguards and human review processes should be in place to mitigate the impact of false positives.

7.3 Professional Considerations

The research findings indicate that machine learning model transferability across different datasets is limited, which can be problematic for organisations relying on these models for cybersecurity purposes. The study suggests that organisations exercise caution when deploying machine learning models across different environments. They must ensure that the models are better generalised to avoid potential issues. However, the explainability results provide insights into the reasons behind this limitation, paving the way for future work to improve the transferability of machine learning models.

From a professional perspective, this research highlights the importance of thoroughly evaluating and validating machine learning models in network intrusion detection. It emphasises the need for organisations to carefully consider the limitations and potential biases of the datasets used for training and testing these models. The findings underscore the significance of explainable AI techniques, such as SHAP, in providing insights into the decision-making process of machine learning models. These insights can aid in identifying the most relevant features for detecting specific types of attacks and guide the development of more robust and reliable intrusion detection systems.

However, professionals should also know the potential risks of using explainable AI techniques in cybersecurity. The insights gained from techniques like SHAP should be treated as sensitive information and protected from unauthorised access. Professionals have a crucial responsibility to ensure the ethical and responsible use of explainability results and to prevent any unintentional assistance to malicious actors in evading detection. Establishing clear guidelines and protocols for handling and sharing the insights derived from explainable AI techniques is crucial to minimising the risk of misuse.

Furthermore, professionals should actively engage in ongoing research and development efforts to improve the transferability and generalisability of machine learning models in the context of network intrusion detection. Collaborative efforts within the cybersecurity community can help address the limitations identified in this study and contribute to developing more robust and adaptable intrusion detection solutions.

7.4 Societal Impact and Sustainability

The development of effective network intrusion detection systems has significant societal implications. In an increasingly interconnected world, the security and integrity of computer networks are crucial for maintaining public trust, protecting sensitive information, and ensuring the smooth functioning of critical infrastructure. This research contributes to developing more robust and transferable machine learning models for intrusion detection, which can help safeguard against cyber attacks and minimise the potential for data breaches. By enhancing the security of networked systems, this research promotes a more sustainable and resilient digital environment.

From a sustainability perspective, the research findings can guide the development of more

adaptable and scalable intrusion detection solutions. This research supports the long-term sustainability of cybersecurity measures by improving the transferability of machine learning models across different datasets and network environments. It enables organisations to leverage existing knowledge and models to detect and respond to emerging threats, reducing the need for extensive retraining and resource-intensive model development processes. This sustainability aspect is critical in the face of constantly evolving cyber attack landscapes and the increasing complexity of networked systems.

However, it is essential to acknowledge that developing advanced intrusion detection systems may also have unintended consequences for society. The increasing reliance on automated systems for cybersecurity could lead to a false sense of security and complacency. It is crucial to raise awareness among individuals and organisations about the limitations of these systems and the importance of maintaining vigilance and adopting a multi-layered approach to security. When considering intrusion detection systems, it is crucial to carefully weigh the potential societal impact of false positives they generate. False positives can cause unnecessary disruptions and erode trust in these systems, so it is crucial to minimise them, which may involve providing clear communication and redress mechanisms to help mitigate their impact on individuals and organisations.

7.5 British Computing Society Code of Conduct

This research is conducted with integrity and professionalism, following the Code of Conduct of the British Computing Society. They present the results accurately and transparently, using publicly available datasets without legal restrictions. The study does not involve any direct ethical or social issues. The researchers present the findings clearly and understandably and acknowledge the study's limitations.

The research aligns with the principles of the BCS Code of Conduct by promoting the responsible use of technology and contributing to the advancement of knowledge in the field of cybersecurity. The study adheres to the principles of honesty, integrity, and objectivity in conducting research and disseminating findings. The research also demonstrates a commitment to the public interest by addressing the critical issue of network intrusion detection and working towards developing more effective and reliable security solutions.

Furthermore, the research acknowledges the importance of professional competence and the

need for continuous learning and improvement. The study builds upon existing knowledge in the field and seeks to advance the understanding of machine learning model transferability and explainability in the context of intrusion detection. The research findings provide valuable insights that can inform future research directions and contribute to the ongoing development of cybersecurity professionals.

However, the research also recognises the potential risks and ethical considerations of using explainable AI techniques in cybersecurity. In adherence to the BCS Code of Conduct, the research emphasises the need for responsible handling and protection of the insights gained from techniques like SHAP. It highlights the importance of establishing clear guidelines and protocols to prevent malicious actors' misuse of explainability results. By addressing these ethical considerations and promoting the responsible use of AI in cybersecurity, the research demonstrates alignment with the BCS Code of Conduct principles.

Chapter 8

Conclusion and Future Work

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] 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.

[7] 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.

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

[9] Gints Engelen, Vera Rimmer, and Wouter Joosen. Troubleshooting an intrusion detection dataset: the cicids2017 case study. In *2021 IEEE Security and Privacy Workshops (SPW)*, pages 7–12. IEEE, 2021.

[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] Arash Habibi Lashkari, Andi Fitriah Abdul kadir, Hugo Gonzalez, Kenneth Mbah, and Ali Ghorbani. Towards a network-based framework for android malware detection and characterization. In *Towards a Network-Based Framework for Android Malware Detection and Characterization*, pages 233–23309, 08 2017.

[13] 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.

[14] 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.

[15] 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.

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

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

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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

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

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

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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

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.

B.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.

B.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.

B.0.3 Datasets

The dataset files are available from the following sources:

- CTU-13 dataset: [17]
- CICIDS2017 dataset: [24]

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.

B.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.