Transferability and Explainability of Machine Learning Models for Network Intrusion Detection

Final Project Report

Author: Ahmed Bedair

Supervisor: Dr. Fabio Pierazzi

Student ID: K2105577

March 26, 2024

Abstract

Machine learning-based network intrusion detection systems (NIDS) have emerged as a promising solution to detect novel and evolving cyber threats. By learning patterns and anomalies from network traffic data, algorithms such as random forests and support vector machines (SVMs) can identify previously unseen intrusions. This research applies machine learning classifiers in NIDS, utilising flow-based features extracted from network traffic. The study investigates the transferability and generalisability of learned patterns across different datasets, simulating the real-world scenario of training a model on one dataset and deploying it to detect intrusions in the wild. By analysing the statistical properties and distributions of flow-based features, this research aims to uncover the key characteristics distinguishing malicious and benign traffic, emphasising the importance of dataset representativeness and biases. Furthermore, this research employs explainable AI techniques, such as SHAP (SHapley Additive exPlanations), to identify the most relevant features contributing to detecting network intrusions. This study can gain insights into the underlying patterns and behaviours that indicate malicious activities by understanding which features are crucial for accurate predictions. This explainability aspect is essential for reasoning about the model's decisions and building trust in the NIDS. The findings of this study contribute to the development of effective and practical NIDS solutions by shedding light on the transferability of learned patterns, the impact of dataset characteristics, and the importance of explainability. By addressing these aspects, this research aims to guide the selection and generation of representative training data and inform the design of more robust and adaptable machine learning algorithms for network intrusion detection.

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

March 26, 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

In recent years, the increasing prevalence and sophistication of cyber attacks have necessitated the development of robust network intrusion detection systems (NIDS). Traditional signature-based NIDS have struggled to keep pace with the evolving threat landscape, leading researchers to explore machine learning approaches for detecting novel and previously unseen attacks [16]. However, the effectiveness of machine learning-based NIDS is heavily dependent on the quality and representativeness of the datasets used for training and evaluation [8].

This paper compares two widely used datasets for network intrusion detection: CTU13 [10] and CICIDS2017 [19]. The CTU13 dataset contains real botnet traffic mixed with passive network traffic and is often used to evaluate NIDS's performance in detecting botnets. On the other hand, the CICIDS2017 dataset is a more recent and comprehensive dataset that includes a wide range of modern attacks such as DoS, DDoS, brute force, XSS, SQL injection, and infiltration [19]. The primary goal of this study is to evaluate the efficacy of using machine learning algorithms to detect botnet attacks. The paper will leverage two datasets to achieve this objective: the CICIDS2017 dataset for training our models and the CTU13 dataset for testing their accuracy. Specifically, this paper trains two popular machine learning classifiers, Random Forest and Support Vector Machine (SVM), on the CICIDS2017 dataset and evaluates their performance detecting botnet attacks in CTU13. This approach allows the assessment of the generalisability and transferability of the learned features and patterns from one dataset to another.

However, it is essential to note that machine learning-based NIDS have challenges. Sommer

and Paxon [20] highlighted the limitations of using machine learning for network intrusion detection, particularly regarding the difficulty in obtaining representative training data and the potential for adversarial attacks. Pierazzi et al. [18] further explored the intriguing properties of adversarial attacks in the problem space of machine learning-based security systems.

This paper draws upon the recommendations of Arp et al.[3] on the dos and don'ts of machine learning for security to guide the experimental design and analysis. In addition to the Random Forest and SVM classifiers, this paper also explores the use of explainable AI techniques. Specifically, SHAP (SHapley Additive exPlanations)[14], to gain insights into the decision-making process of the trained models. SHAP provides a unified framework for interpreting predictions and helps identify the most important features contributing to the models' decisions. By applying SHAP to the trained classifiers, this paper aims to understand the key patterns and characteristics that distinguish botnet traffic from passive network traffic in both the CTU13 and CICIDS2017 datasets.

Furthermore, this paper delves into the comparative analysis of the network flow features extracted from the CTU13 and CICIDS2017 datasets. It examines the similarities and differences in the makeup of these datasets, considering that CTU13 contains real network traffic while CICIDS2017 is synthetically generated. By exploring the statistical properties and distributions of various flow features, such as flow duration, total forward packets, total backward packets, forward packet length mean, backward packet length mean, and flow inter-arrival time mean, this paper aims to uncover the inherent characteristics that differentiate actual and simulated network traffic. For instance, the analysis of flow duration and inter-arrival time mean can provide insights into the temporal characteristics of the network flows. At the same time, total forward and backward packets can shed light on the directionality and volume of the traffic.

Investigating features like forward packet length mean, backward packet length mean, and flow inter-arrival time standard deviation can reveal patterns in packet sizes and inter-arrival time variations, which may differ between actual and simulated traffic. This analysis provides valuable insights into the challenges and limitations of using synthetic datasets for training NIDS models and highlights the importance of considering dataset biases when evaluating their performance. By understanding the differences in flow characteristics between actual and simulated traffic, researchers can make informed decisions about selecting and preprocessing datasets for building robust and generalisable NIDS models.

Including the SVM classifier in this study allows for a more comprehensive evaluation of the transferability and generalisability of machine learning models across different datasets. By comparing the performance of Random Forest and SVM, this paper can assess the robustness and adaptability of these classifiers in detecting botnet attacks in a real-world scenario using the CTU13 dataset. Additionally, the application of SHAP to both classifiers enables the identification and comparison of the most influential features for each model, providing a deeper understanding of the underlying patterns and characteristics that contribute to accurate botnet detection.

The remainder of this paper is structured as follows: Section 2 provides a comprehensive background and literature review of related work in machine learning-based NIDS. It discusses the state-of-the-art approaches, their strengths, and limitations, setting the context for this research. Section 3 presents the specification and design of the experimental setup, including a detailed description of the CTU13 and CICIDS2017 datasets, their collection methodologies, attack scenarios, and feature sets. It also outlines the data preprocessing and feature engineering steps to prepare the data for machine learning. Section 4 focuses on the implementation aspects of this study, including the training and optimisation of the Random Forest and SVM classifiers, as well as the application of SHAP for model interpretability. Section 5 presents the evaluation of the proposed approach, discussing the performance metrics, comparative analysis of the classifiers, and the insights gained from SHAP explanations. It also highlights the key findings from the flow-level analysis of the datasets and their implications for NIDS development and evaluation. Section 6 explores the legal, social, ethical, and professional issues related to using machine learning in network intrusion detection, addressing concerns such as data privacy, fairness, and the potential impact on society. Finally, Section 7 concludes the paper by summarising the main contributions, discussing the limitations of this study, and outlining potential future research directions to advance further the field of machine learning-based NIDS.

Chapter 2

Background

Section 2.1 provides an overview of two widely used datasets in the field of network intrusion detection: the CTU13 dataset [10] and the CICIDS2017 dataset [19]. 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 [10] and the CICIDS2017 dataset [19].

Garcia et al. [10] 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.

Sharafaldin et al. [19] 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 that closely resembles 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.

Several studies have utilised these datasets to develop and evaluate network intrusion detection systems. Chowdhury et al. [6] 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 [17] 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. [22] 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 [1] 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 [20] discuss the challenges of using synthetic datasets for network intrusion detection. They argue that synthetic datasets may not fully 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 that consider intrusion detection systems' operational aspects and deployment challenges.

To address these limitations, Lashkari et al. [11] proposed a framework for generating representative network traffic datasets. Their approach incorporates techniques for generating realistic benign traffic and modelling user behaviour to ensure the diversity and representativeness of the datasets. They also developed the CICFlowMeter tool, which enables the extraction of flow-based features from raw network traffic captures. The tool is widely used alongside the CICIDS2017 dataset for feature extraction and analysis.

However, Engelen et al. [8] 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.

The importance of using representative and unbiased datasets for training machine learning models in network intrusion detection is emphasised by Sommer and Paxson [20]. They highlight 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 [5] 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.

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 [12]. 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 [9] 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. [4] 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 [7]. 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. [13] 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. [21] 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 [5] 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) [14] 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. [23] 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. [2] 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 [15] 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

Specification & Design

This chapter describes the specification and design of the experiments conducted in this research, split into three sections, each detailing the experimental setup for the Dummy Classifier, Random Forest Classifier, and Support Vector Machine Classifier.

The CTU13 dataset contains real botnet traffic captured in a controlled environment, consisting of 13 scenarios representing specific botnet behaviors [10]. On the other hand, the CICIDS2017 dataset is a more recent and comprehensive dataset designed for evaluating network intrusion detection systems, containing a wide range of modern attacks [19]. The motivation for using these datasets is to assess the transferability and generalisability of machine learning models trained on the CICIDS2017 dataset in detecting botnet attacks in the real-world CTU13 dataset.

The preprocessing steps, particularly the relabeling process using the relabelCTU13.py and relabelCICIDS2017.py scripts, ensure consistency and compatibility between the datasets. These scripts map the features of each dataset to a standard naming convention, enabling fair comparisons and analysis. The relabeling process also addresses the challenge of inconsistent labeling across datasets, which is a common issue in network intrusion detection research.

The evaluation metrics used in this research include the confusion matrix, accuracy, precision, recall, and F1 score. These metrics provide a comprehensive assessment of the classifiers' performance in the context of network intrusion detection. The confusion matrix allows for a detailed analysis of the classifiers' predictions, while accuracy, precision, recall, and F1 score offer a quantitative measure of their effectiveness in correctly identifying network intrusions.

The SHAP (SHapley Additive exPlanations) library is employed to interpret the trained models' predictions and provide insights into their decision-making process. SHAP assigns importance scores to each feature, indicating their contribution to the model's prediction for a specific instance. By applying SHAP to the trained classifiers, this research aims to identify the key features that contribute to detecting specific types of attacks. The insights gained from SHAP can aid in understanding the underlying patterns and behaviors that indicate malicious activities, guiding feature engineering and model development efforts.

It is worth mentioning that all train-test splits performed in this research use a 60/40 ratio. That means 60% of the data is used for training and 40% for testing. A 60/40 train/test split fairly balances the training and testing data. This balance helps to avoid overfitting and ensures fairness in testing.

3.0.1 Alternative Designs and Dataset Selection

The paper considered various sets of classifiers when designing the experiments. Neural network-based classifiers, such as Multi-Layer Perceptron (MLP) and Convolutional Neural Networks (CNN), have shown promising results in network intrusion detection tasks [17, 22]. However, the paper did not use such classifiers due to their higher computational complexity and the focus on interpretability using SHAP, which is more straightforward with tree-based and kernel-based classifiers like Random Forest and SVM.

Other classifiers considered include Decision trees and k-nearest Neighbors (k-NN), which could have been alternatives to the currently used Random Forest and SVM. Belouch et al. [4] conducted a comparative study of different classifiers on the CICIDS2017 dataset and found that Random Forest outperformed Decision Trees and k-NN in terms of accuracy and false-positive rates. The study's results suggested that Random Forest and SVM could perform better, hence the decision to use them in this research.

The paper's choice of CTU13 and CICIDS2017 was made based on each dataset's properties and makeup, respectively. CICIDS2017 contains many modern attacks, making it suitable for training classifiers to detect various intrusion types [19]. CTU13, on the other hand, focuses specifically on botnet attacks, providing a controlled environment for evaluating the transferability of models trained on CICIDS2017 to a specific attack scenario. Since the paper aims to focus specifically on botnet attacks and assess the transferability of models across datasets, the combination of CICIDS2017 and CTU13 datasets was deemed appropriate.

The CTU13 dataset is a collection of data focused on botnet attacks captured in a controlled environment [10]. The purpose of utilizing CTU13 as the testing dataset is to evaluate the effectiveness of models trained on CICIDS2017 in detecting botnet attacks in a real-world setting. By examining the performance of classifiers on CTU13, this study aims to assess the applicability of the learned patterns and features from CICIDS2017 to a specific attack scenario.

The combination of CICIDS2017 and CTU13 datasets enables a comprehensive analysis of the classifiers' ability to detect botnet attacks. It provides insights into the transferability of models across different network environments and attack scenarios. This design choice 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.

3.1 Dummy Classifier

Purpose The Dummy Classifier serves as a reference point to evaluate the efficacy of advanced classifiers like Random Forests and Support Vector Machines. If either of these classifiers fails to outperform the Dummy Classifier, it indicates that the model is not learning effectively from the data. The Dummy Classifier randomly assigns labels to the data, providing a baseline for comparison with more sophisticated models.

3.1.1 CTU13

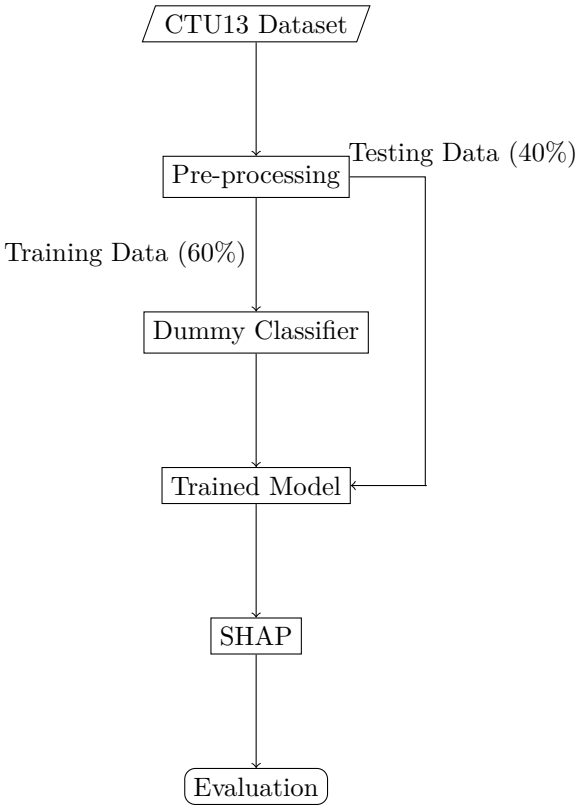

Figure 3.1: Dummy Classifier Flowchart for CTU13

The experiment begins with the raw pcap files from the CTU13 dataset, which contains only botnet attacks and benign traffic. Due to the limited scope of the dataset, a multi-class approach is not feasible. The data undergoes preprocessing to ensure compatibility with the classifier. This preprocessing step involves relabeling the dataset to maintain consistency with the CICIDS2017 dataset, as shown in the relabelCTU13.py script. The relabeling process maps the CTU13 dataset's features to match the naming convention used in the CICIDS2017 dataset, enabling fair comparisons between the two datasets.

After preprocessing, the data is split into training and testing sets using a 60/40 ratio. The training data (60% of the dataset) is used to train the Dummy Classifier, which is then employed to make predictions on the testing data (40% of the dataset). The predictions are evaluated using the SHAP (SHapley Additive exPlanations) library, which explains which features the classifier used to inform its decisions. The evaluation metrics include the confusion matrix, accuracy, precision, recall, and F1 score, offering a comprehensive assessment of the classifier's performance.

3.1.2 Binary CICIDS2017

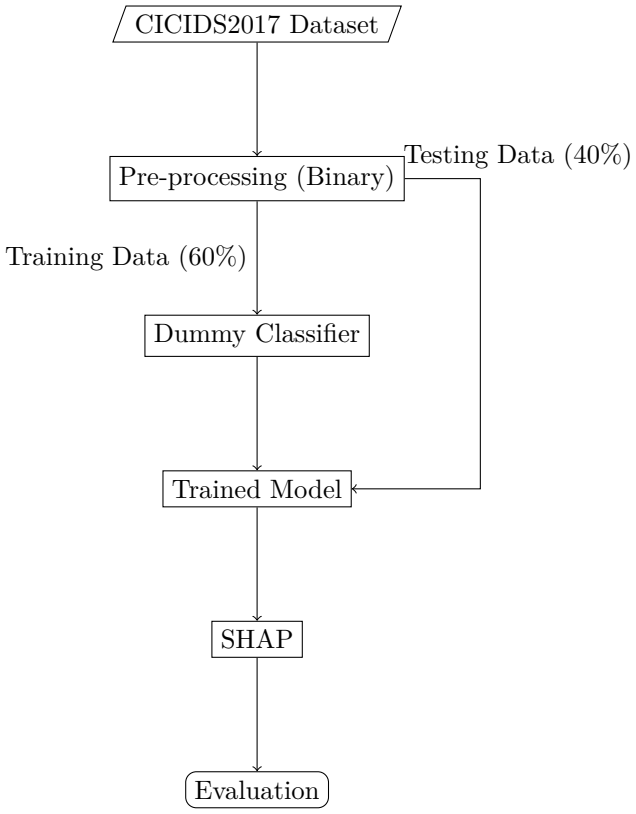

Figure 3.2: Dummy Classifier Flowchart for Binary CICIDS2017

The experiment utilising the CICIDS2017 dataset follows a similar structure to the CTU13 experiment. The raw pcap files from CICIDS2017 are preprocessed to ensure compatibility with the classifier and to convert the dataset into a binary classification problem. As detailed in the relabelCICIDS2017.py script, the preprocessing step involves relabeling the dataset to maintain consistency with the CTU13 dataset. The script maps the CICIDS2017 dataset's features to match the naming convention used in the CTU13 dataset, enabling fair comparisons between the two datasets.

After preprocessing, the data is split into training and testing sets using a 60/40 ratio. The training data (60% of the dataset) is used to train the Dummy Classifier, which is then employed to make predictions on the testing data (40% of the dataset). The predictions are evaluated using the SHAP library for explanations, and the same set of performance metrics as in the CTU13 experiment are used.

3.1.3 Multi-class CICIDS2017

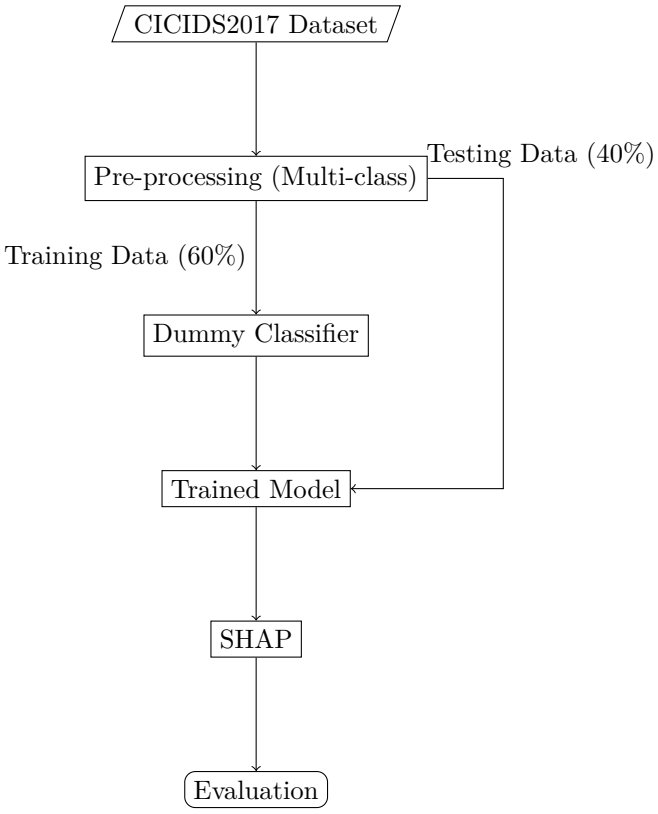

Figure 3.3: Dummy Classifier Flowchart for Multi-class CICIDS2017

In contrast to the CTU13 dataset, the CICIDS2017 dataset contains a diverse range of attack types, enabling a multi-class classification approach. The raw pcap files from CICIDS2017 are preprocessed to ensure compatibility with the classifier and to convert the dataset into a multi-class classification problem. As outlined in the relabelCICIDS2017.py script, the preprocessing step involves relabeling the dataset to maintain consistency with the CTU13 dataset, similar to the binary classification experiment.

After preprocessing, the data is split into training and testing sets using a 60/40 ratio. The training data (60% of the dataset) is used to train the Dummy Classifier, which is then employed to make predictions on the testing data (40% of the dataset). The predictions are evaluated using the SHAP library for explanations and the same set of performance metrics as in the previous experiments.

3.2 Random Forest Classifier

Purpose The Random Forest Classifier is employed to improve the Dummy Classifier's performance by leveraging an ensemble of decision trees to make predictions. Combining multiple decision trees, the Random Forest Classifier aims to capture more complex patterns and relationships within the data, potentially enhancing classification accuracy.

3.2.1 CTU13

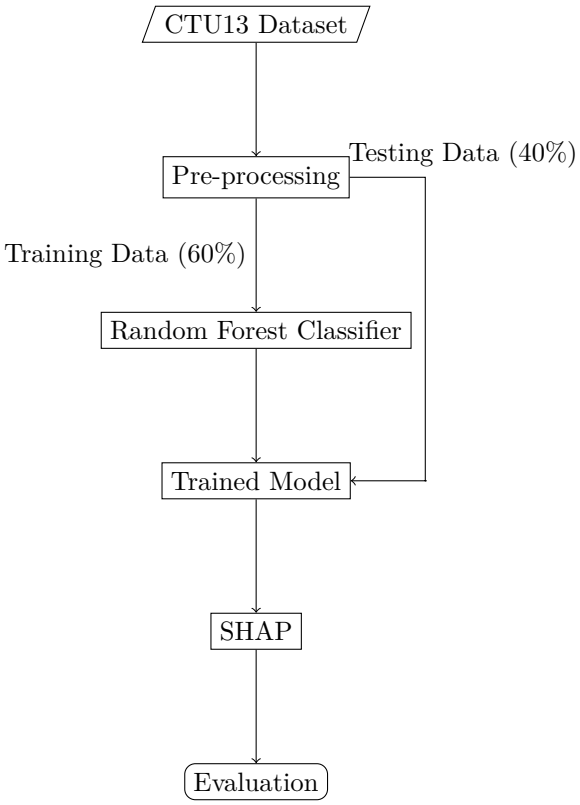

Figure 3.4: Random Forest Classifier Flowchart for CTU13

The Random Forest Classifier experiment on the CTU13 dataset follows a similar process as the Dummy Classifier experiment. The data is preprocessed using the relabelCTU13.py script to ensure consistency with the CICIDS2017 dataset. The preprocessed data is split into training and testing sets using a 60/40 ratio. The training data (60% of the dataset) is used to train the Random Forest Classifier, which is subsequently employed to make predictions on the testing data (40% of the dataset). The predictions are evaluated using the SHAP library for explanations and the same set of performance metrics as in the Dummy Classifier experiment.

3.2.2 Binary CICIDS2017

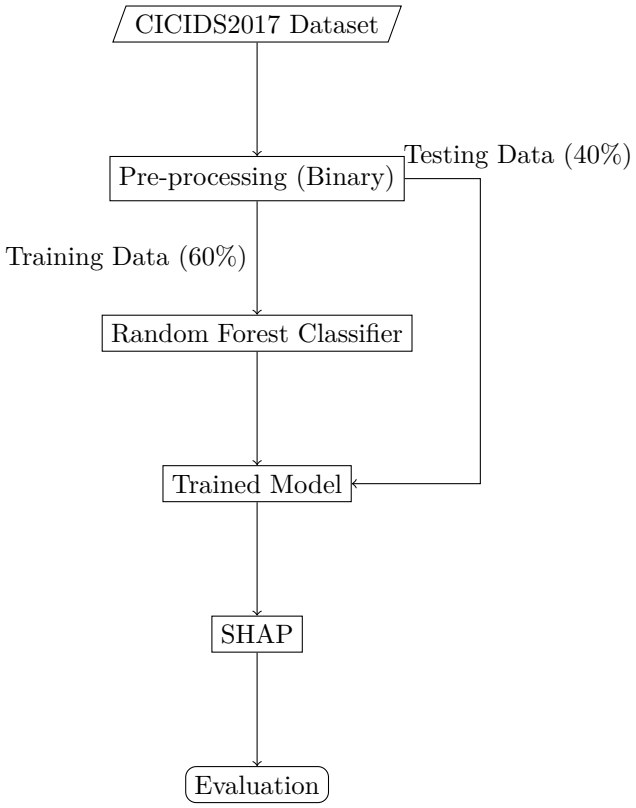

Figure 3.5: Random Forest Classifier Flowchart for Binary CICIDS2017

The Random Forest Classifier experiment on the binary CICIDS2017 dataset involves preprocessing the raw pcap files using the relabelCICIDS2017.py script to ensure consistency with the CTU13 dataset and converting the problem into a binary classification task. The preprocessed data is split into training and testing sets using a 60/40 ratio. The training data (60% of the dataset) is used to train the Random Forest Classifier, which is subsequently employed to make predictions on the testing data (40% of the dataset). The predictions are evaluated using the SHAP library for explanations and the same set of performance metrics as in the previous experiments.

3.2.3 Multi-class CICIDS2017

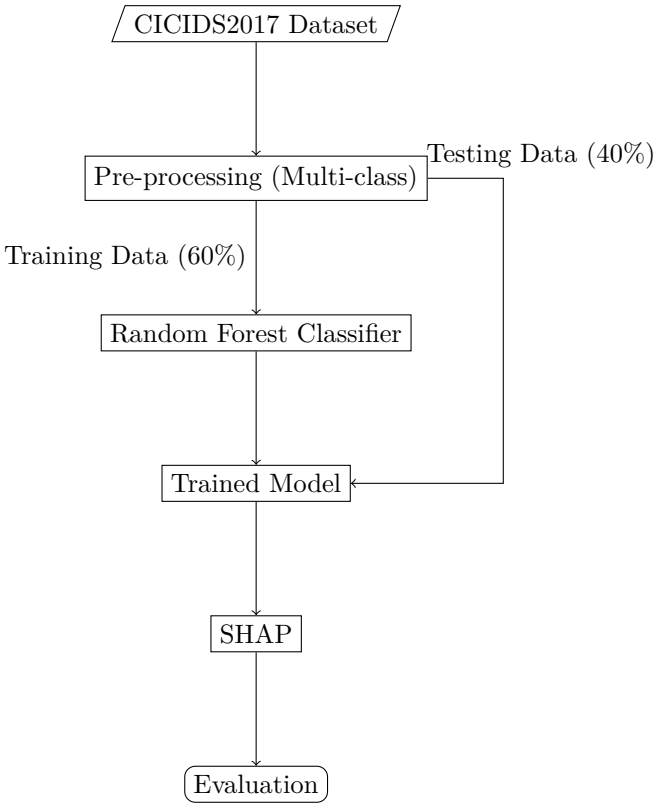

Figure 3.6: Random Forest Classifier Flowchart for Multi-class CICIDS2017

The Random Forest Classifier experiment on the multi-class CICIDS2017 dataset follows a process similar to that of the binary classification experiment. The raw pcap files are preprocessed using the relabelCICIDS2017.py script to ensure consistency with the CTU13 dataset and to convert the problem into a multi-class classification task. The preprocessed data is split into training and testing sets using a 60/40 ratio. The training data (60% of the dataset) is used to train the Random Forest Classifier, which is subsequently employed to make predictions on the testing data (40% of the dataset). The predictions are evaluated using the SHAP library for explanations and the same set of performance metrics as in the previous experiments.

3.3 Support Vector Machine Classifier

Purpose The Support Vector Machine (SVM) Classifier is utilised to improve the Dummy Classifier's performance by finding the optimal hyperplane that separates the different classes in the feature space. SVM is known for its ability to handle high-dimensional data and its effectiveness in binary classification tasks. Using kernel tricks, SVM can be extended to handle

non-linearly separable data if necessary.

3.3.1 CTU13

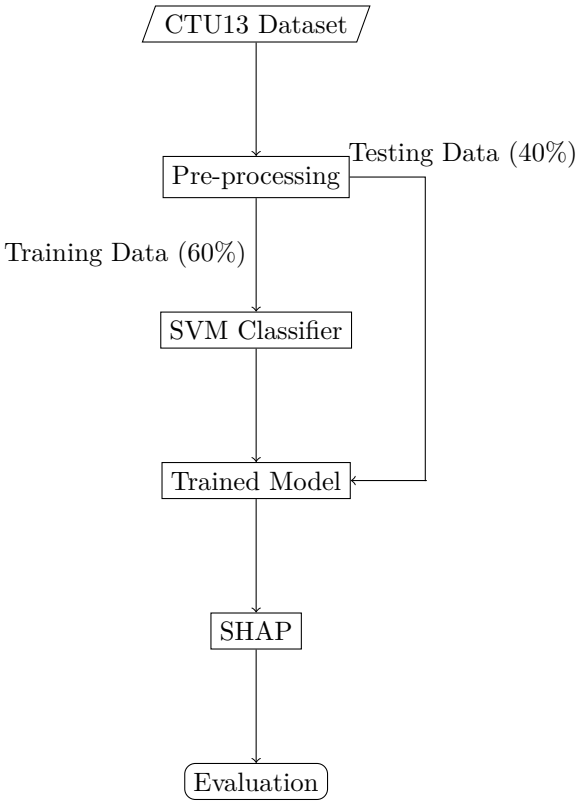

Figure 3.7: SVM Classifier Flowchart for CTU13

The SVM Classifier experiment on the CTU13 dataset follows a similar process as the Dummy Classifier experiment. The data is preprocessed using the relabelCTU13.py script to ensure consistency with the CICIDS2017 dataset. The preprocessed data is split into training and testing sets using a 60/40 ratio. The training data (60% of the dataset) is used to train the SVM Classifier, which is subsequently employed to make predictions on the testing data (40% of the dataset). The predictions are evaluated using the SHAP library for explanations and the same set of performance metrics as in the previous experiments.

3.3.2 Binary CICIDS2017

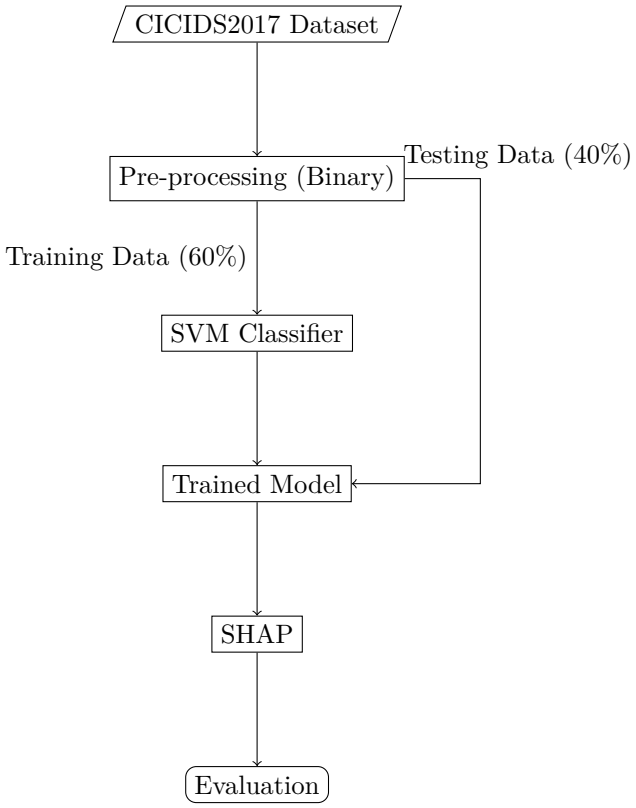

Figure 3.8: SVM Classifier Flowchart for Binary CICIDS2017

The SVM Classifier experiment on the binary CICIDS2017 dataset involves preprocessing the raw pcap files using the relabelCICIDS2017.py script to ensure consistency with the CTU13 dataset and converting the problem into a binary classification task. The preprocessed data is split into training and testing sets using a 60/40 ratio. The training data (60% of the dataset) is used to train the SVM Classifier, which is subsequently employed to make predictions on the testing data (40% of the dataset). The predictions are evaluated using the SHAP library for explanations and the same set of performance metrics as in the previous experiments.

3.3.3 Multi-class CICIDS2017

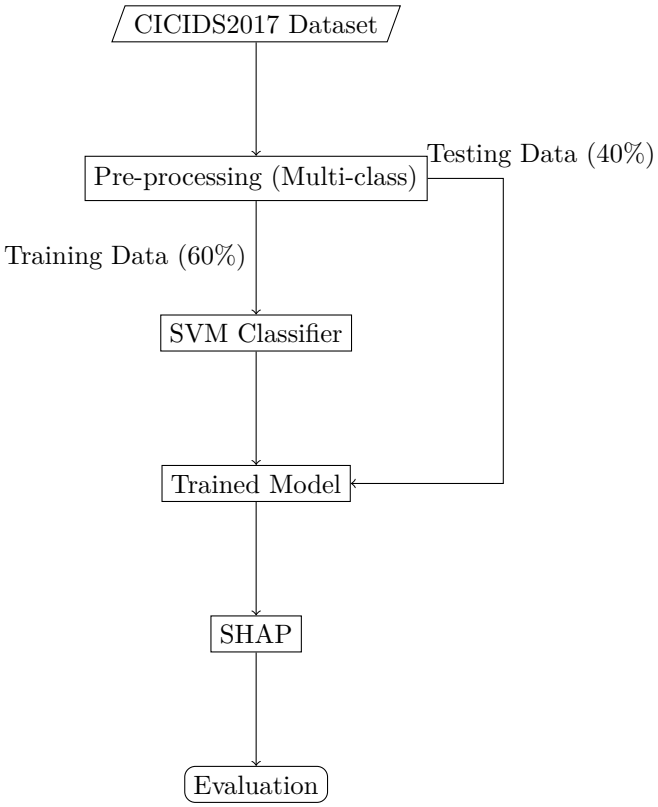

Figure 3.9: SVM Classifier Flowchart for Multi-class CICIDS2017

The SVM Classifier experiment on the multi-class CICIDS2017 dataset follows a process similar to that of the binary classification experiment. The raw pcap files are preprocessed using the relabelCICIDS2017.py script to ensure consistency with the CTU13 dataset and to convert the problem into a multi-class classification task. The preprocessed data is split into training and testing sets using a 60/40 ratio. The training data (60% of the dataset) is used to train the SVM Classifier, which is subsequently employed to make predictions on the testing data (40% of the dataset). The predictions are evaluated using the SHAP library for explanations and the same set of performance metrics as in the previous experiments.

Chapter 4

Implementation

Chapter 5

Evaluation

Chapter 6

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.

6.1 Legal Considerations

The research utilises two publicly available datasets: CTU13[10] and CICIDS2017[19]. 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.

6.2 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.

6.3 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.

6.4 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.

6.5 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 7

Conclusion and Future Work

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

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

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

[11] 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.

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

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

[16] 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.

[17] 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.

[18] Fabio Pierazzi, Feargus Pendlebury, Jacopo Cortellazzi, and Lorenzo Cavallaro. Intriguing properties of adversarial ml attacks in the problem space. In *2020 IEEE symposium on security and privacy (SP)*, pages 1332–1349. IEEE, 2020.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.