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 27, 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

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 an increasingly interconnected world, the security of computer networks is of paramount importance. However, the rapid evolution of cyber threats has made it challenging for traditional signature-based network intrusion detection systems (NIDS) to keep pace with novel and sophisticated attacks [16]. To address this issue, researchers have turned to machine learning approaches, which have the potential to detect previously unseen intrusions by learning patterns and anomalies from network traffic data [5].

The effectiveness of machine learning-based NIDS heavily relies on the quality and representativeness of the datasets used for training and evaluation [8]. Two widely used datasets in this domain are CTU13 [10], which contains real botnet traffic mixed with background traffic, and CICIDS2017 [18], a more recent dataset that includes a diverse range of modern attacks. However, the transferability and generalisability of machine learning models trained on one dataset to detect intrusions in another remain largely unexplored.

This paper aims to address this gap by investigating the efficacy of using machine learning algorithms trained on the CICIDS2017 dataset to detect botnet attacks in the CTU13 dataset. Specifically, two popular classifiers, Random Forest and Support Vector Machine (SVM), were trained on CICIDS2017 and evaluated their performance on CTU13. This approach allows us to assess the transferability of learned features and patterns across different datasets, simulating a real-world scenario where a model is trained on one dataset and deployed to detect intrusions in the wild.

To guide our experimental design and analysis, we draw upon the recommendations of Arp et al. [3] on the best practices for applying machine learning in security. Furthermore, we employ explainable AI techniques, such as SHAP (SHapley Additive exPlanations) [14], to gain insights into the decision-making process of our trained models. By identifying the most important features contributing to the detection of botnet traffic, we aim to uncover the key patterns and characteristics that distinguish malicious and benign network flows.

Additionally, we conduct a comparative analysis of the network flow features extracted from the CTU13 and CICIDS2017 datasets. By exploring the statistical properties and distributions of various features, such as flow duration, packet counts, and inter-arrival times, we seek to understand the inherent differences between actual and simulated network traffic. This analysis sheds light on the challenges and limitations of using synthetic datasets for training NIDS models and emphasises the importance of considering dataset biases in performance evaluation.

Including Random Forest and SVM classifiers in our study enables a more comprehensive assessment of model transferability and generalisability. By comparing their performance and interpreting their predictions using SHAP, we gain valuable insights into the robustness and adaptability of these algorithms in detecting botnet attacks across different datasets.

The remainder of this paper is organised as follows: Section 2 provides background information on the CTU13 and CICIDS2017 datasets, machine learning classifiers, and explainable AI techniques. Section 3 presents a literature review of relevant work in machine learning-based NIDS Section 4 describes the specification and design of our experiments, while Section 5 details the implementation aspects. Section 6 presents the evaluation results and discusses the implications of our findings. Section 7 addresses the legal, social, ethical, and professional issues related to using machine learning in network intrusion detection. Finally, Section 8 concludes the paper and outlines future research directions.

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 [18]. 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 [18].

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. [18] 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. [21] 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 [19] 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 [19]. 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. [20] 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. [22] 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

Relevant Work

This section 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 section 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. By examining these relevant works, this section provides a solid foundation for understanding the current state of research in this domain and identifying the gaps this study seeks to address.

3.1 Botnet Detection using CTU13 Dataset

Several studies have utilized the CTU13 dataset for developing and evaluating botnet 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. 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.

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. 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.

3.2 Intrusion Detection using CICIDS2017 Dataset

Many researchers have widely used the CICIDS2017 dataset to evaluate intrusion detection systems. Ustebay et al. [21] 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.

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. 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.

3.3 Random Forest for Intrusion Detection

Farnaaz and Jabbar [9] 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.

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. The authors preprocessed 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.

3.4 Support Vector Machines for Intrusion Detection

Intrusion detection tasks have extensively utilized Support Vector Machines (SVM). 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 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.

Teng et al. [20] 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.

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. [22] evaluated various explanation methods for deep learning-based intrusion detection systems, including SHAP (SHapley Additive exPlanations). 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.

Amarasinghe et al. [2] 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.

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. 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 Forest.

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 [19] 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.

Furthermore, Sommer and Paxson [19] 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 [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.

The choice of 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 emphasize 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 generalization 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.

Chapter 4

Specification & Design

This chapter outlines the design and specification of the experiments conducted in this research, focusing on three main classifiers: Dummy Classifier, Random Forest Classifier, and Support Vector Machine Classifier. Based on the literature review, the paper designed the experimental setup to address the following key issues:

1. The need for assessing the transferability and generalisability of machine learning models across different datasets in the context of network intrusion detection [5, 19].

2. The importance of considering dataset biases and the representativeness of training data when developing and evaluating intrusion detection systems [8, 19].

3. The potential of explainable AI techniques, such as SHAP, to provide insights into the decision-making process of machine learning models and identify the most relevant features for detecting specific types of attacks [2, 15, 22].

The paper chose the CTU13 and CICIDS2017 datasets as they contain specific types of attacks and possess distinct properties that make them suitable for our research.CTU13 consists of real botnet traffic captured in a controlled environment [10], while CICIDS2017 is a more recent and comprehensive dataset designed for evaluating network intrusion detection systems, containing a wide range of modern attacks [18]. By training models on CICIDS2017 and testing them on CTU13, this study aims to assess the transferability of learned patterns and features from a synthetic dataset to a real-world scenario, addressing the first identified issue.

To ensure uniformity and coherence across all datasets, we utilise the relabelCTU13.py and relabelCICIDS2017.py scripts during pre-processing. These scripts facilitate mapping dataset features to a standardised naming convention, promoting equitable comparisons and analysis. This approach effectively addresses the second concern surrounding dataset biases and representativeness.

The evaluation metrics used in this research, such as confusion matrix, accuracy, precision, recall, and F1 score, comprehensively assess the classifiers' performance in network intrusion detection. Furthermore, the SHAP library is employed to interpret the trained models' predictions and provide insights into their decision-making process. It addresses the third identified issue regarding the potential of explainable AI techniques in intrusion detection.

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. [5].

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. [4] 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.

The following sections provide detailed descriptions of the experimental setup for each classifier, including the purpose, dataset-specific considerations, and evaluation processes. By addressing the identified issues, providing the rationale for the selected approaches, and deriving the experimental design from the synthesis of relevant literature, this chapter lays the foundation for a comprehensive and rigorous investigation into the transferability and explainability of machine learning models for network intrusion detection.

4.1 Dummy Classifier

Purpose The Dummy Classifier is a reference point for evaluating 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.

4.1.1 CTU13

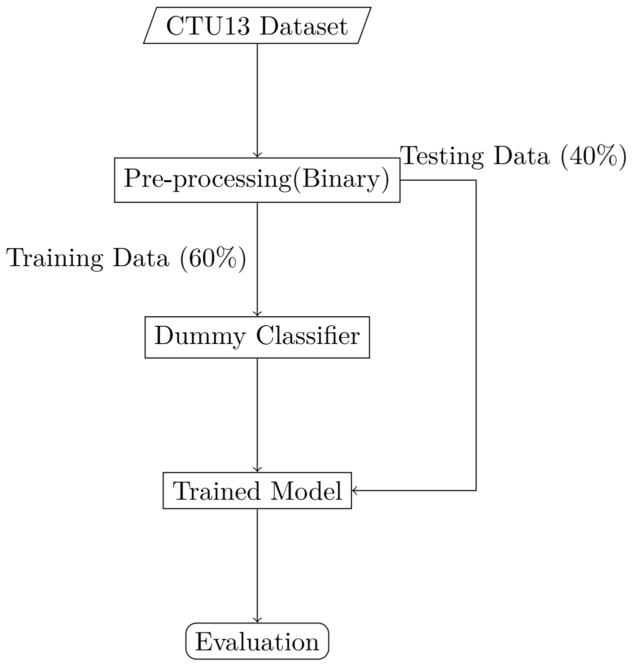

Figure 4.1: Dummy Classifier Flowchart for CTU13

The experiment commences by utilising the raw PCAP files from the CTU13 dataset, which exclusively includes botnet attacks and benign traffic. Firstly, a tool to extract flows, such as CICFlowMeter [11], converts the raw PCAP files into CSV files that the classifiers can process. Prior to analysis, the data undergoes pre-processing to ensure compatibility with CICIDS2017. This pre-processing step incorporates the relabelCTU13.py script to relabel the dataset, ensuring consistency with the CICIDS2017 dataset. The relabeling process maps the CTU13 dataset's features to match the naming convention used in CICIDS2017, enabling fair comparisons between the two datasets. Given CTU13's exclusive focus on botnet attacks, there is no need for additional relabelling steps to unify attack names for the classifiers to group them.

After pre-processing, we divide the data into training and testing sets in a 60/40 proportion. The Dummy Classifier is trained on the training data (60% of the dataset) and subsequently utilised to make predictions on the testing data (40% of the dataset). The evaluation metrics include the confusion matrix, accuracy, precision, recall, and F1 score, comprehensively evaluating the classifier's effectiveness.

4.1.2 Binary CICIDS2017

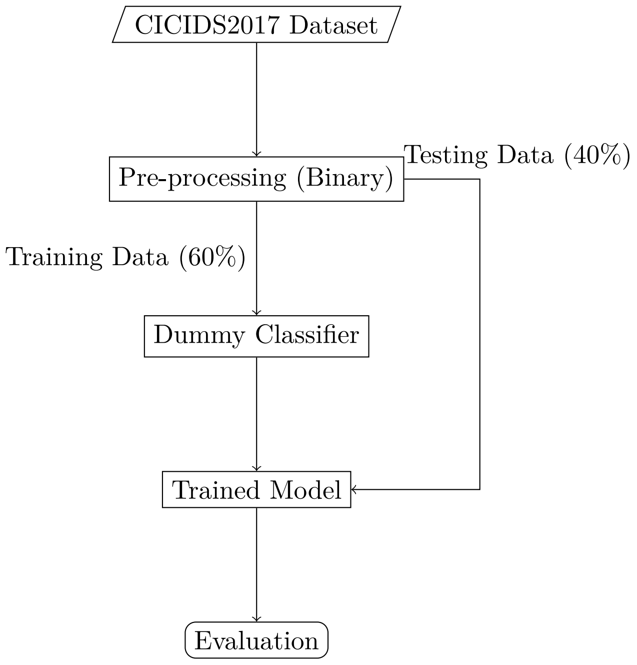

Figure 4.2: Dummy Classifier Flowchart for Binary CICIDS2017

The experiment conducted using the CICIDS2017 dataset has a structure that closely resembles the CTU13 experiment. Firstly, a tool like CICFlowMeter [11] converts the raw PCAP files from CICIDS2017 into CSV files that the classifiers can process. The CSV files undergo pre-processing to ensure compatibility with the classifier and convert the dataset into a binary classification problem. The relabelCICIDS2017.py script provides a detailed explanation of the pre-processing step, which involves relabeling the dataset to maintain consistency with the CTU13 dataset. The script enables fair comparisons between the two datasets by mapping the CICIDS2017 dataset's features to match the naming convention used in the CTU13 dataset.

After pre-processing, we divide the data proportionally into training and testing sets at a 60/40 ratio. We use the training data, which accounts for 60% of the dataset, to train the Dummy Classifier. Then, the classifier makes predictions on the testing data, representing 40% of the dataset. The paper then uses the same performance metrics as the CTU13 experiment to evaluate the predictions.

4.1.3 Multi-class CICIDS2017

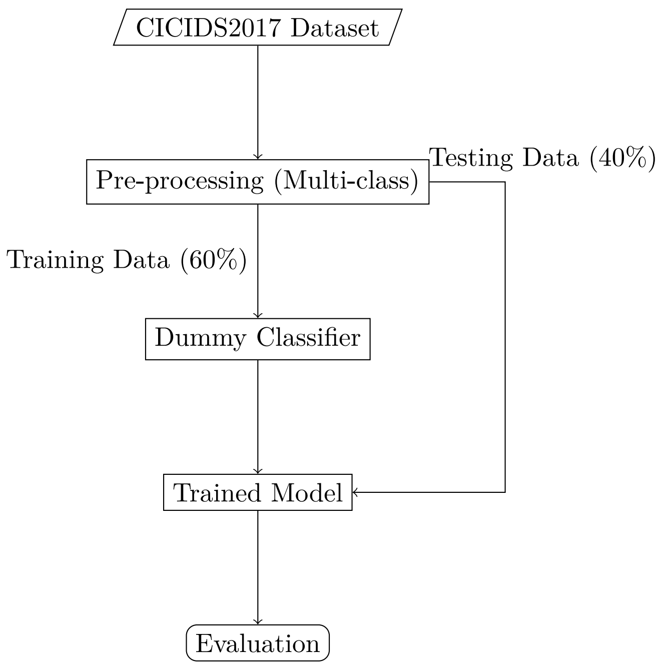

Figure 4.3: Dummy Classifier Flowchart for Multi-class CICIDS2017

Compared to the CTU13 dataset, the CICIDS2017 dataset boasts various attack types, making it an excellent choice for a multi-class classification strategy. The inclusion of the multi-class experiment aims to check if any different attack types in CICIDS2017 follow similar structures regarding their flows to botnet attacks from CTU13. If a classifier trained on CICIDS2017 can successfully discover botnet attacks in CTU13, it would suggest that some attack types in CICIDS2017 share similar flow characteristics with botnet attacks.

Firstly, a tool like CICFlowMeter [11] converts the raw PCAP files from CICIDS2017 into CSV files that the classifiers can process. The CSV files undergo pre-processing to ensure the classifier's compatibility and transform the dataset into a multi-class classification challenge. As described in the relabelCICIDS2017.py script, the pre-processing stage entails relabeling the dataset to maintain consistency with the CTU13 dataset, similar to the binary classification experiment.

After pre-processing, we divided the dataset into training and testing sets with a 60/40 ratio.

We trained the Dummy Classifier on the training data, which accounted for 60% of the dataset. Then, we applied the classifier to the testing data, representing 40% of the dataset, to generate predictions. Finally, we used the same performance metrics from previous experiments to assess these predictions.

4.2 Random Forest Classifier

4.2.1 CTU13

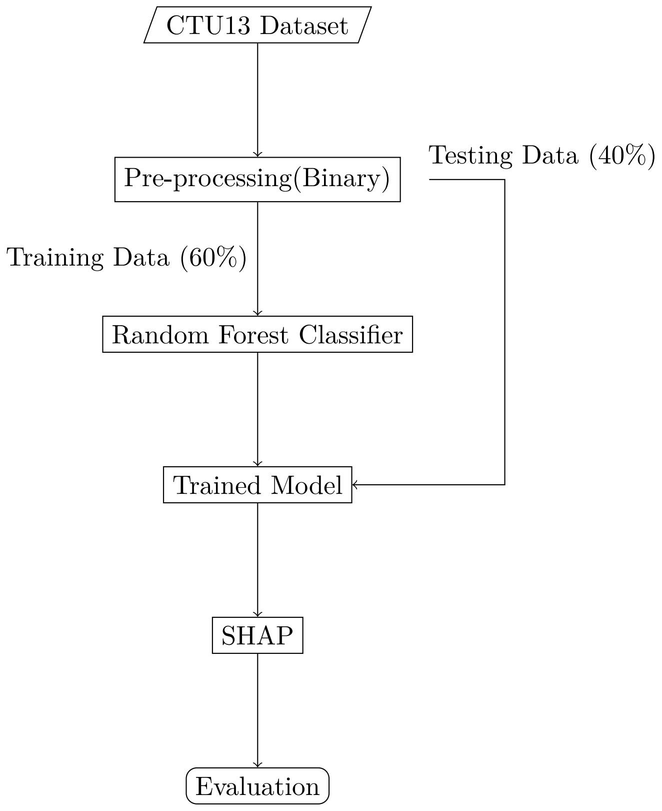

Figure 4.4: Random Forest Classifier Flowchart for CTU13

The experiment on the CTU13 dataset utilising the Random Forest Classifier follows a similar process to the previous Dummy Classifier experiment. Firstly, a tool like CICFlowMeter [11] converts the raw PCAP files from CTU13 into CSV files that the classifiers can process. To ensure consistency with the CICIDS2017 dataset, we utilise the relabelCTU13.py script for pre-processing the CSV files. Once pre-processing is complete, we split the data into training and testing sets with a 60/40 ratio. Next, the Random Forest Classifier is trained on the training data (60% of the dataset) and utilised to predict the testing data (40% of the dataset). Finally, the SHAP library for explanations and the same performance metrics used in the previous Dummy Classifier experiment are applied to evaluate the predictions.

4.2.2 Binary CICIDS2017

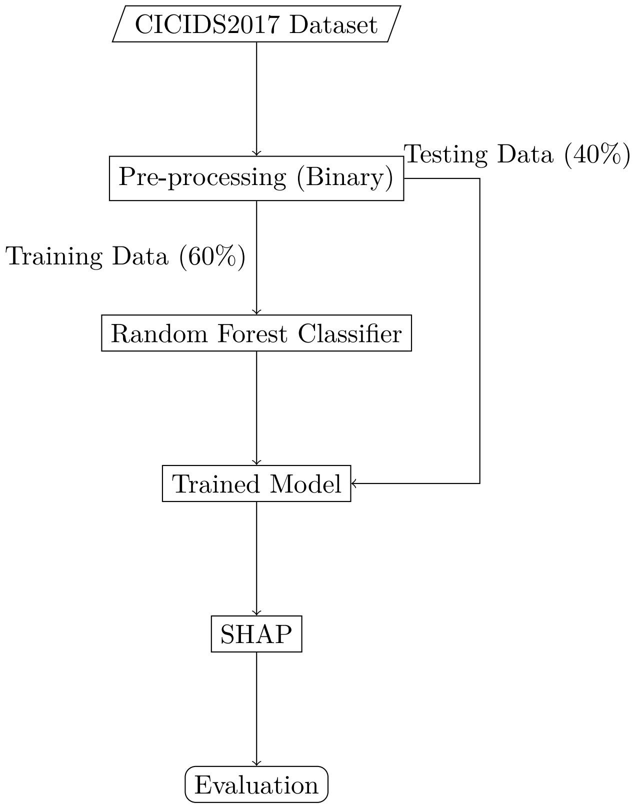

Figure 4.5: Random Forest Classifier Flowchart for Binary CICIDS2017

To execute the Random Forest Classifier experiment on the binary CICIDS2017 dataset, we first use a tool like CICFlowMeter [11] to convert the raw PCAP files from CICIDS2017 into CSV files that the classifiers can process. We then pre-process the CSV files using the relabelCICIDS2017.py script. We did this to ensure consistency with the CTU13 dataset and to convert the problem into a binary classification task. After that, we divided the pre-processed data into training and testing sets using a 60/40 ratio. We trained the Random Forest Classifier on the training data, which accounted for 60% of the dataset. We used it to make predictions on the testing data, which accounted for 40% of the dataset. Finally, we evaluated our predictions using the SHAP library for explanations and the same set of performance metrics as in the prior experiments.

4.2.3 Multi-class CICIDS2017

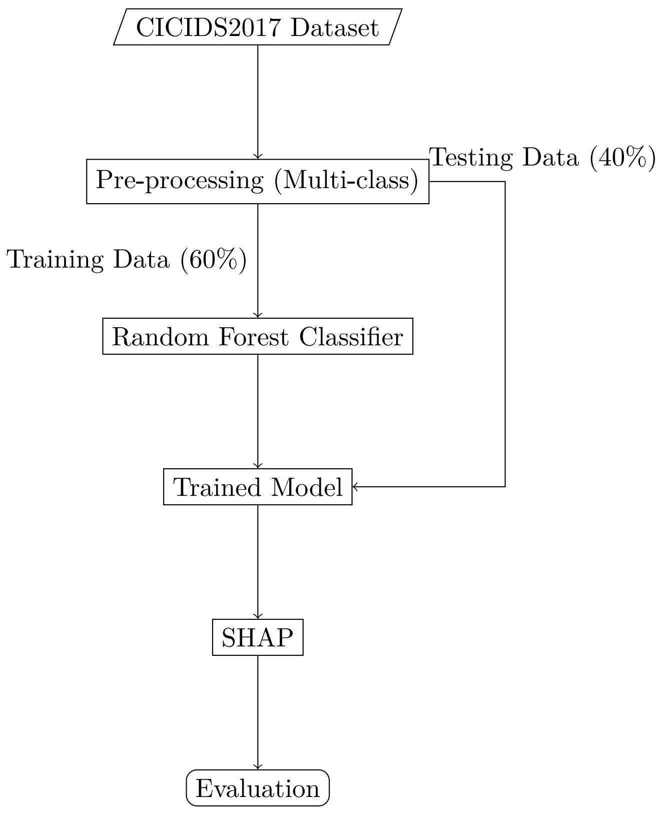

Figure 4.6: Random Forest Classifier Flowchart for Multi-class CICIDS2017

For the multi-class CICIDS2017 dataset Random Forest Classifier experiment, the goal is to investigate if the Random Forest Classifier can learn patterns from various attack types in CICIDS2017 that are similar to botnet attacks in CTU13. If the classifier can successfully detect botnet attacks in CTU13 after being trained on CICIDS2017, it would suggest that some attack types in CICIDS2017 share similar flow characteristics with botnet attacks.

We first use a tool like CICFlowMeter [11] to convert the raw PCAP files from CICIDS2017 into CSV files that the classifiers can process. We then pre-process the CSV files using the relabelCICIDS2017.py script to ensure consistency with the CTU13 dataset and convert the problem to a multi-class classification task. Then, we split the pre-processed data into training and testing sets using a 60/40 ratio. We use the training data (60% of the dataset) to train the Random Forest Classifier, which subsequently predicts the testing data (40% of the dataset). Finally, we evaluate the predictions using the SHAP library for explanations and the same set of performance metrics as in the previous experiments.

4.3 Support Vector Machine Classifier

4.3.1 CTU13

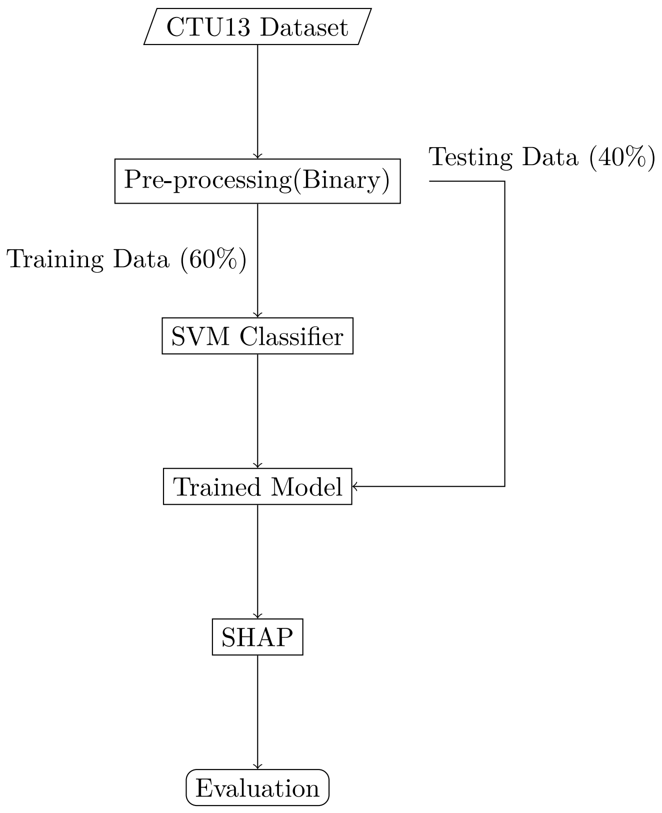

Figure 4.7: SVM Classifier Flowchart for CTU13

For the SVM Classifier experiment on the CTU13 dataset, we first use a tool like CICFlowMeter [11] to convert the raw PCAP files from CTU13 into CSV files that the classifiers can process. We then pre-process the CSV files using the relabelCTU13.py script to ensure consistency with the CICIDS2017 dataset. Then, we split the pre-processed data into training and testing sets at a 60/40 ratio. After that, we train the SVM Classifier using the training data comprising 60% of the dataset. Once we complete the classifier training, we use it to predict the testing data that constitutes 40% of the dataset. Finally, we evaluate the predictions using the SHAP library for explanations and the same set of performance metrics as in the previous experiments.

4.3.2 Binary CICIDS2017

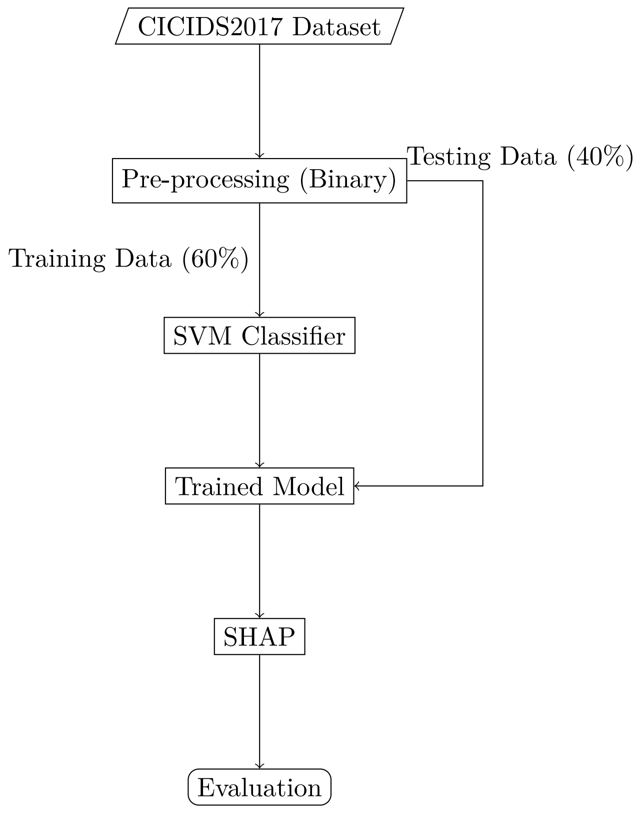

Figure 4.8: SVM Classifier Flowchart for Binary CICIDS2017

To conduct the SVM Classifier experiment on the binary dataset CICIDS2017, we first use a tool like CICFlowMeter [11] to convert the raw PCAP files from CICIDS2017 into CSV files that the classifiers can process. We then pre-process the CSV files using the relabelCICIDS2017.py script. The relabeling ensures consistency with the CTU13 dataset and transforms the problem into a binary classification task. Once we complete the pre-processing, we split the data into training and testing sets with a 60/40 ratio. We use the training data, which accounts for 60% of the dataset, to train the SVM Classifier. Afterwards, we utilise the trained model to make predictions on the testing data, which accounts for 40% of the dataset. Finally, we evaluate the predictions using the SHAP library for explanations and the same set of performance metrics as in our previous experiments.

4.3.3 Multi-class CICIDS2017

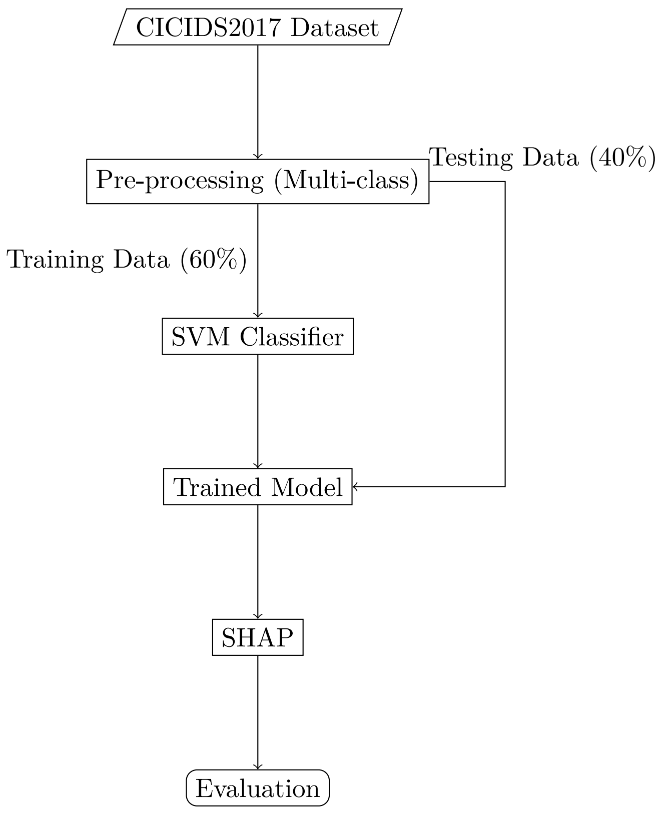

Figure 4.9: SVM Classifier Flowchart for Multi-class CICIDS2017

The SVM Classifier experiment on the multi-class CICIDS2017 dataset aims to explore if the SVM Classifier can identify similarities between the flow structures of various attack types in CICIDS2017 and botnet attacks in CTU13. If the SVM Classifier, trained on CICIDS2017, can effectively detect botnet attacks in CTU13, it would indicate that certain attack types in CICIDS2017 exhibit similar flow characteristics to botnet attacks.

We first use a tool like CICFlowMeter [11] to convert the raw PCAP files from CICIDS2017 into CSV files that the classifiers can process. We then pre-process the CSV files using the relabelCICIDS2017.py script to ensure consistency with the CTU13 dataset and transform the problem into a multi-class classification task. After pre-processing, we divided the data into training and testing sets in a 60/40 ratio. We used the training data, which comprised 60% of the dataset, to train the SVM Classifier. Next, we utilised the trained classifier to make predictions on the testing data, which accounted for 40% of the dataset. Finally, we evaluated the predictions using the SHAP library for explanations and the same performance metrics used in the previous experiments.

Chapter 5

Implementation

Chapter 6

Evaluation

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[10] and CICIDS2017[18]. 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 intrusion 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] 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] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.