
# 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 25, 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 focuses on the application of machine learning classifiers in NIDS, utilizing flow-based features extracted from network traffic. The study investigates the transferability and generalizability 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 analyzing the statistical properties and distributions of flow-based features, this research aims to uncover the key characteristics that distinguish malicious traffic from benign traffic, emphasizing 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 the detection of network intrusions. By understanding which features are crucial for accurate predictions, this study can gain insights into the underlying patterns and behaviors that indicate malicious activities. 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 25, 2024

**Acknowledgements**

Thank you to my supervisor, Dr. Fabio Pierazzi, for his guidance and support throughout the project. The supervision and support provided was very much appreciated and helped me to 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 focuses on comparing two widely used datasets for network intrusion detection: CTU13 [10] and CICIDS2017 [19]. The CTU13 dataset contains real botnet traffic mixed with normal traffic and is often used for evaluating the performance of NIDS 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 main objective of this research is to investigate the effectiveness of using machine learning models trained on the CICIDS2017 dataset to detect botnet attacks in the CTU13 dataset. Specifically, this paper trains two popular machine learning classifiers, namely Random Forest and Support Vector Machine (SVM), on the CICIDS2017 dataset and evaluates their performance on 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 important to note that machine learning-based NIDS are not without their challenges. Sommer and Paxon [20] highlighted the limitations of using machine learning for

network intrusion detection, particularly in terms of 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 normal 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 duration, packet size, inter-arrival times, and protocol usage, this paper aims to uncover the inherent characteristics that differentiate real and simulated network 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.

The inclusion of 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 back-

ground and literature review of related work in the field of 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 the 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 undertaken 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 the use of 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 further advance 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]. These datasets have been extensively utilised by researchers to develop and evaluate machine learning-based NIDS.Understanding the characteristics, strengths, and limitations of these datasets 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, which have been 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 these classifiers 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 in understanding the factors that contribute to the predictions made by the models and provide interpretable explanations for their outputs. Understanding the importance of explainable AI techniques 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 labeled 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 behaviors 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 real botnet samples, and the labeling process based on the known behavior 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, which is a more recent and comprehensive dataset designed for evaluating network intrusion detection systems. The dataset contains a wide range of modern attacks, including DoS, DDoS, brute force, XSS, SQL injection, and infiltration. The authors provide a detailed description of the dataset generation process, which involved creating a controlled lab environment that closely resembles a real-world network infrastructure. They used various tools and scripts to generate realistic benign traffic and attack scenarios. The dataset also includes a combination of manually labeled and time-based labeled data. The authors evaluate the dataset using different machine learning algorithms and demonstrate its effectiveness in detecting various types of attacks.

Several studies have utiliased these datasets for developing and evaluating 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 show-

cased the effectiveness of deep learning in capturing complex patterns and behaviors 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 present 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 in terms of 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 important 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 lack important contextual information. They emphasise the need for more realistic and representative datasets that consider the operational aspects and deployment challenges of intrusion detection systems.

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 modeling user behavior 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. This tool has been widely used in conjunction with 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 analysis of the dataset and discovered problems related to flow construction, feature extraction, and labeling. 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 the selection of 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 automatically identify malicious activities and distinguish them from benign traffic. Random Forest (RF) and Support Vector Machine (SVM) are two popular classifiers that have shown promising results in this domain.

Random Forest is an ensemble learning method that combines multiple decision trees to make predictions [12]. It constructs a large number of 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 the effectiveness of Random Forest 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 labeled 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 the effectiveness of SVM 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 machine learning classifier depends on various factors, such as the characteristics of the dataset, 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 in understanding 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 the detection of specific types of attacks.

Warnecke et al. [23] evaluated various explanation methods, including SHAP, for deep learning-based intrusion detection systems. 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 important 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 aid in validating the reliability of the trained models, identifying potential biases or limitations in the datasets, and guiding 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 the detection of specific attacks and developing more targeted defense strategies.

# Chapter 3

# Specification & Design

This chapter describes the specification and design of the experiments conducted in this paper, split into 3 sections, each detailing the experimental setup for the Dummy Classifier, Random Forest Classifier, and Support Vector Machine Classifier.

## 3.1 Dummy Classifier

**Purpose** The Dummy Classifier serves as a baseline model to establish a performance benchmark against which the more advanced Random Forest and Support Vector Machine classifiers will be compared. By evaluating the Dummy Classifier's performance, we can assess the effectiveness of the other classifiers in improving upon the baseline results.

### 3.1.1 CTU13

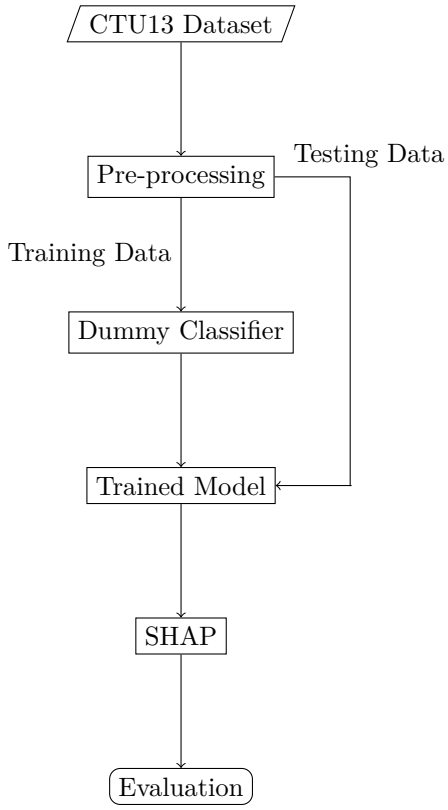


Figure 3.1: Dummy Classifier Flowchart for CTU13

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. The training data is used to train the Dummy Classifier, which is then employed to make predictions on the testing data. The predictions are evaluated using the SHAP (SHapley Additive exPlanations) library, which provides explanations for the classifier's 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

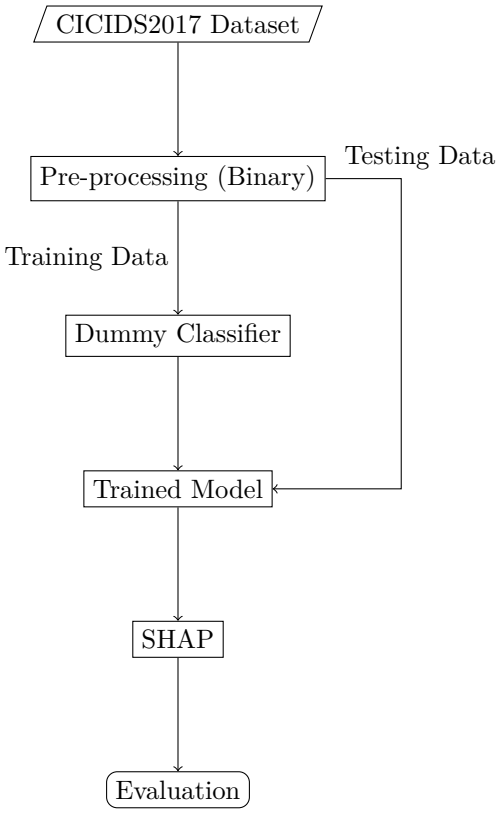


Figure 3.2: Dummy Classifier Flowchart for Binary CICIDS2017

The experiment utilizing 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. The preprocessing step, as detailed in the relabelCICIDS2017.py script, 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. The training data is used to train the Dummy Classifier, which is then employed to make predictions on the testing data. The predictions are evaluated using the SHAP library for explanations and the same set of performance metrics as in the CTU13 experiment.

### 3.1.3 Multi-class CICIDS2017

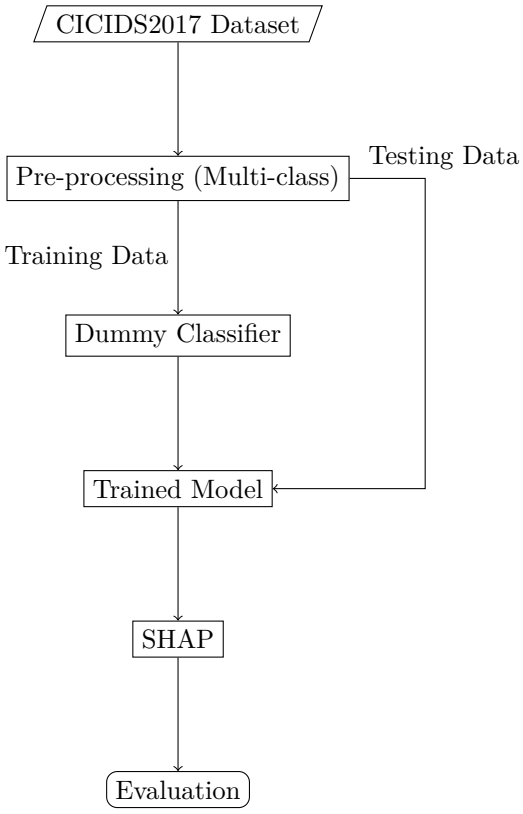


Figure 3.3: Dummy Classifier Flowchart for Multi-class CICIDS2017

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. The preprocessing step, as outlined in the relabelCICIDS2017.py script, 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. The training data is used to train the Dummy Classifier, which is then employed to make predictions on the testing data. 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 upon the performance of the Dummy Classifier by leveraging an ensemble of decision trees to make predictions. By combining multiple decision trees, the Random Forest Classifier aims to capture more complex patterns and relationships within the data, potentially leading to enhanced classification accuracy.

### 3.2.1 CTU13

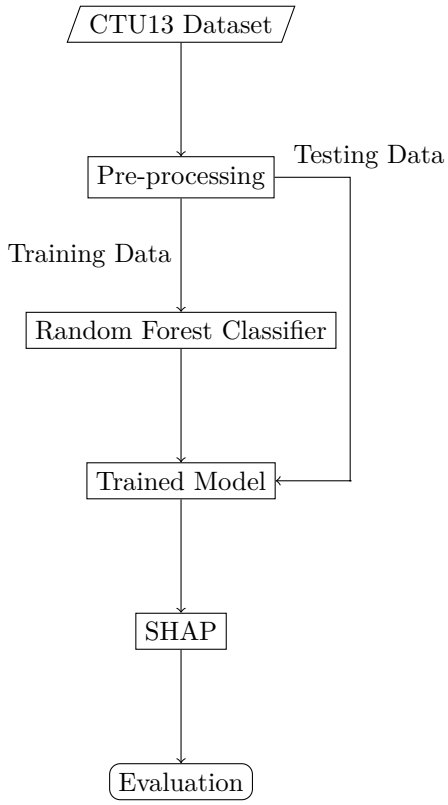


Figure 3.4: Random Forest Classifier Flowchart for CTU13

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 then split into training and testing sets. The training data is used to train the Random Forest Classifier, which is subsequently employed to make predictions on the testing data. 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

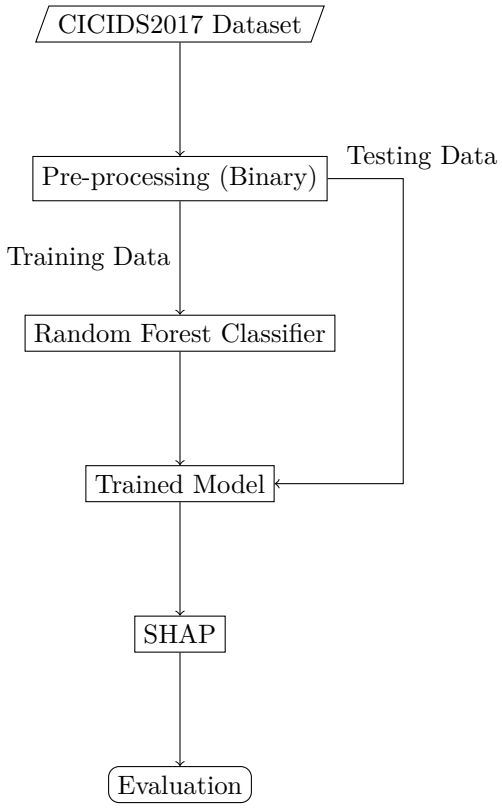


Figure 3.5: Random Forest Classifier Flowchart for Binary CICIDS2017

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 then split into training and testing sets. The training data is used to train the Random Forest Classifier, which is subsequently employed to make predictions on the testing data. 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

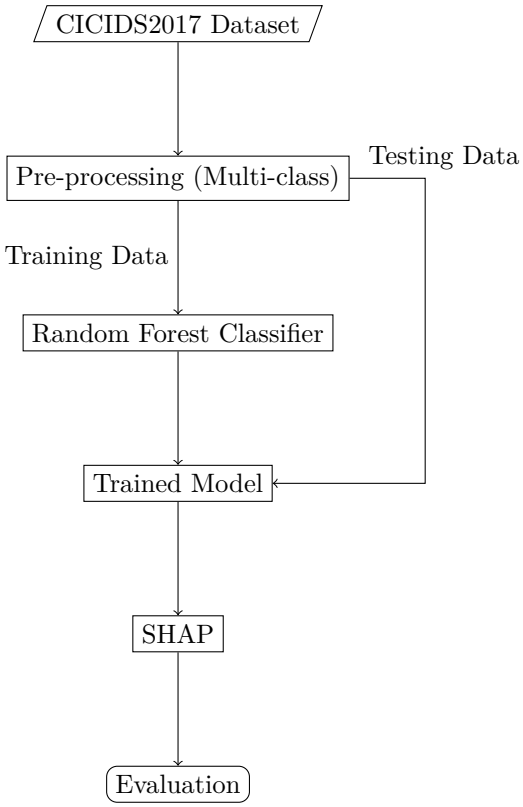


Figure 3.6: Random Forest Classifier Flowchart for Multi-class CICIDS2017

The Random Forest Classifier experiment on the multi-class CICIDS2017 dataset follows a similar process as 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 then split into training and testing sets. The training data is used to train the Random Forest Classifier, which is subsequently employed to make predictions on the testing data. 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 utilized to improve upon the performance of the Dummy Classifier 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. By employing kernel tricks, SVM can also be

extended to handle non-linearly separable data.

### 3.3.1 CTU13

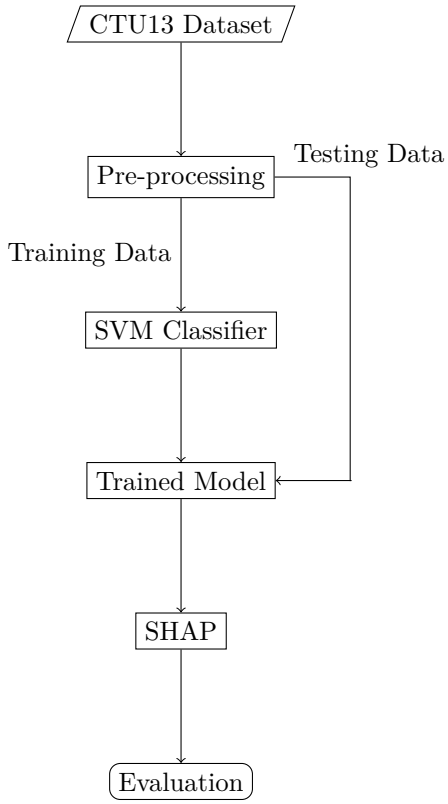


Figure 3.7: SVM Classifier Flowchart for CTU13

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 then split into training and testing sets. The training data is used to train the SVM Classifier, which is subsequently employed to make predictions on the testing data. 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

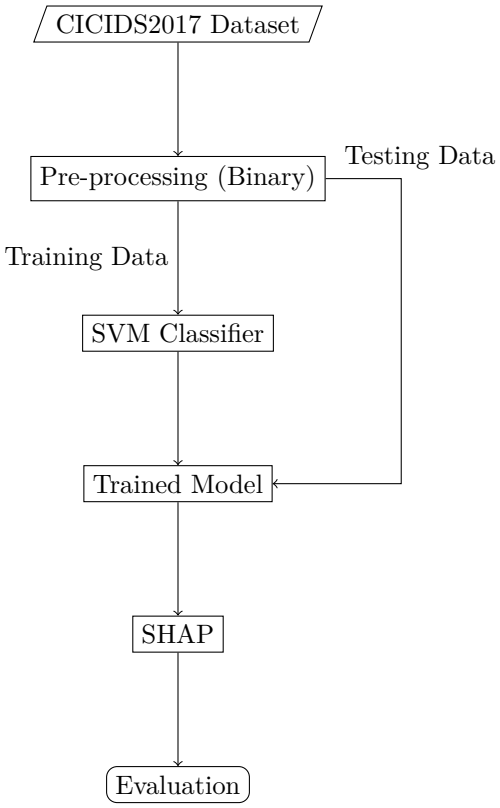


Figure 3.8: SVM Classifier Flowchart for Binary CICIDS2017

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 then split into training and testing sets. The training data is used to train the SVM Classifier, which is subsequently employed to make predictions on the testing data. 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

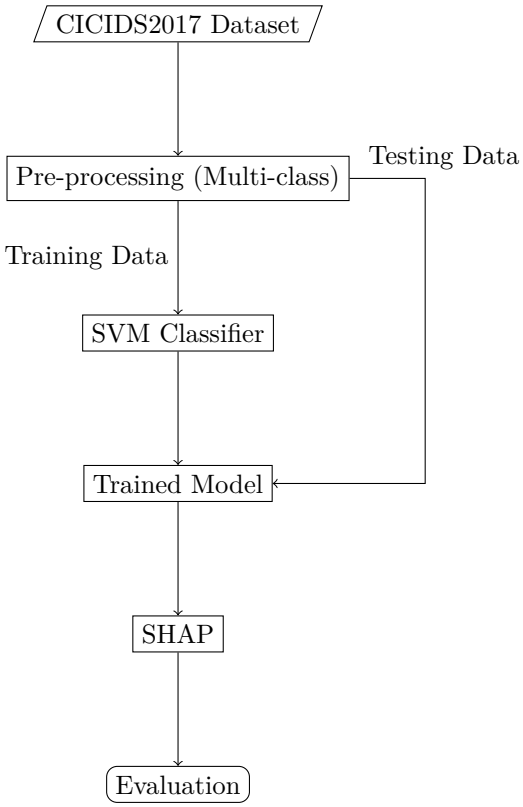


Figure 3.9: SVM Classifier Flowchart for Multi-class CICIDS2017

The SVM Classifier experiment on the multi-class CICIDS2017 dataset follows a similar process as 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 then split into training and testing sets. The training data is used to train the SVM Classifier, which is subsequently employed to make predictions on the testing data. The predictions are evaluated using the SHAP library for explanations and the same set of performance metrics as in the previous experiments.

In conclusion, this chapter provides a detailed specification and design of the experiments conducted using the Dummy Classifier, Random Forest Classifier, and Support Vector Machine Classifier on the CTU13 and CICIDS2017 datasets. The experiments are designed to evaluate the performance of these classifiers in binary and multi-class classification tasks, with a focus on preprocessing the datasets to ensure consistency and compatibility. The use of the SHAP library for model explanations and the evaluation of performance metrics provide a comprehensive

assessment of the classifiers' effectiveness in detecting network intrusions. The inclusion of flowcharts for each experiment enhances the clarity and understanding of the experimental setup and design.

# 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 concludes by explaining 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

The focus of this research is on evaluating the transferability and generalisation of machine learning models across different datasets. The study does not involve human subjects, and the datasets used are publicly available. Consequently, there are no ethical or social issues associated with this research.

## 6.3 Professional Considerations

The research findings indicate that the current state of 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 should exercise caution when deploying machine learning models across different environments, as the models may not generalise well. However, the explainability results provide insights into the reasons behind this limitation, paving the way for future work aimed at improving the transferability of machine learning models.

## 6.4 British Computing Society Code of Conduct

This research is conducted in accordance with the British Computing Society's Code of Conduct. The study is carried out with integrity and professionalism, and the results are presented accurately and transparently. The research does not involve any ethical or social issues, and the datasets used are publicly available and free from legal restrictions. The findings of the research are presented in a clear and understandable manner, and the limitations of the study are acknowledged.

# 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.