People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

Abbes Laghrour University KHENCHELA

Faculty

of Science and Technology

A dissertation for the degree of Master in Computer Science

Option: Cyber Security

An Explainable Machine Learning for Network Intrusion Detection: A Random Forest and SHAP-based Implementation

---

Presented by:

ATHMANI Fouad

N: 212134059989

Supervised by:

Dr. Mohamed Mahdi MALIK

The jury members:

President:

Examiner:

Academic Year: 2025-2026

Abstract

In this work, we propose an explainable artificial intelligence (XAI)-based approach for network intrusion detection using the CICIDS2017 dataset. The main objective is to develop an interpretable model capable of detecting multiple types of network attacks while providing clear and transparent explanations for its predictions.

A Random Forest classifier is used for multi-class classification. The dataset is preprocessed and analyzed to handle issues such as data imbalance and feature representation. The performance of the model is evaluated using standard metrics including accuracy, precision, recall, and F1-score.

In addition to performance evaluation, explainability is introduced using SHAP (SHapley Additive exPlanations). This allows a better understanding of the model's decisions by identifying the most important features influencing the predictions.

The results show that the proposed model achieves high classification performance while providing meaningful insights into its behavior, making it more reliable for real-world intrusion detection applications.

Keywords: Intrusion Detection, Machine Learning, Random Forest, Explainable AI, SHAP

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr. Mohamed Mahdi MALIK, for his continuous guidance, support, and valuable advice throughout the development of this work. His constructive feedback and recommendations played an important role in improving the quality of this project.

I would also like to thank the members of the jury for their time, effort, and valuable evaluation of this thesis.

My appreciation goes as well to all the professors of the department for their teaching and support during my academic journey. Their efforts have contributed significantly to the knowledge and skills I have acquired.

I would also like to thank the administrative staff for their assistance and for providing a supportive academic environment.

Finally, I would like to express my gratitude to everyone who contributed, directly or indirectly, to the completion of this work.

ATHMANI Fouad.

Dedicated to

My Family

for their endless support, encouragement, and patience.

My Teachers

who guided me throughout my academic journey.

Everyone who supported me

in completing this work.

Contents

List of Figures

Abbreviations

IDS	Intrusion Detection System
NIDS	Network Intrusion Detection System
ML	Machine Learning
XAI	Explainable Artificial Intelligence
RF	Random Forest
SVM	Support Vector Machine
DoS	Denial of Service
DDoS	Distributed Denial of Service
SHAP	SHapley Additive exPlanations
CICIDS2017	CICIDS2017 Dataset

Chapter 1

Introduction

1.1 Background

In recent years, the rapid growth of internet usage and networked systems has led to an increase in cyber threats [1]. Organizations rely heavily on network infrastructures to store and exchange sensitive data, which makes them a primary target for attackers [2]. As a result, ensuring the security of these systems has become a critical concern.

Intrusion Detection Systems (IDS) are security mechanisms designed to monitor network or system activities in order to detect malicious behavior, unauthorized access, or violations of security policies [2]. Unlike firewalls, which primarily focus on preventing unauthorized access, IDS are responsible for identifying suspicious activities that may indicate the presence of cyber attacks.

IDS can generally be classified into two main categories: Network-based Intrusion Detection Systems (NIDS) and Host-based Intrusion Detection Systems (HIDS). NIDS monitor and analyze network traffic across multiple devices, while HIDS focus on activities occurring within a specific host or system. These systems play an

essential role in modern cybersecurity infrastructures by helping organizations detect and respond to threats in real time [2, 3].

One of the main approaches used to protect networks is the use of Intrusion Detection Systems (IDS). These systems monitor network traffic and aim to detect malicious activities or policy violations. Traditional IDS techniques are often based on predefined rules or signatures. While effective against known attacks, they struggle to detect new or evolving threats [3].

1.1.1 Signature-Based and Anomaly-Based Detection

Signature-based intrusion detection relies on predefined patterns or signatures of known attacks. In this approach, network traffic is compared against a database of attack signatures in order to identify malicious activities. Signature-based systems are generally effective in detecting previously known threats with high accuracy and low false positive rates [3].

However, these systems have significant limitations when dealing with new or unknown attacks, since any attack that does not match an existing signature may remain undetected [3, 4]. As cyber threats continue to evolve rapidly, maintaining and updating signature databases becomes increasingly challenging [1].

To overcome these limitations, anomaly-based intrusion detection systems were introduced. Unlike signature-based methods, anomaly detection approaches focus on learning the normal behavior of a network or system and identifying deviations from this normal behavior. Such methods are more capable of detecting previously unseen attacks and zero-day threats [4].

Anomaly-based detection is commonly associated with machine learning techniques, where models learn patterns from historical data in order to distinguish between normal and malicious activities [5]. Although these approaches provide greater flexibility, they often suffer from higher false positive rates due to the difficulty of accurately defining normal behavior [4, 5].

To overcome these limitations, Machine Learning (ML) techniques have been introduced in the field of network intrusion detection. ML-based systems can learn patterns from data and identify abnormal behaviors without relying solely on predefined signatures. This makes them more flexible and capable of detecting previously unseen attacks [4].

1.1.2 Machine Learning IDS vs Deep Learning IDS

Machine learning techniques have become widely used in intrusion detection systems due to their ability to automatically learn patterns from network traffic data and identify malicious activities [4, 6]. Traditional machine learning algorithms such as Decision Trees, Support Vector Machines (SVM), and Random Forests are commonly employed because of their relatively low computational cost and good classification performance.

In recent years, Deep Learning (DL) approaches have also gained significant attention in the field of intrusion detection. Deep learning models, including Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN), are capable of learning complex representations from large-scale data [6]. These methods can achieve high detection accuracy, especially when dealing with large and complex datasets.

However, deep learning models generally require larger datasets, higher computational resources, and longer training times compared to traditional machine learning approaches [7]. In addition, many deep learning models suffer from limited interpretability, making it difficult to understand the reasoning behind their predictions [8].

In contrast, traditional machine learning models such as Random Forests provide a better balance between performance, computational efficiency, and interpretability [8, 9]. For this reason, Random Forest was selected in this work as the primary classification model.

1.1.3 Common Network Attacks

Modern networks are exposed to a wide variety of cyber attacks that can compromise the confidentiality, integrity, and availability of systems and data [1]. Intrusion detection systems are designed to identify such malicious activities and help organizations respond to potential threats effectively [10].

One of the most common attacks is Denial of Service (DoS), where an attacker attempts to overwhelm a target system or network with excessive traffic in order to disrupt its normal operation. A more advanced form of this attack is Distributed Denial of Service (DDoS), in which multiple compromised devices are used simultaneously to flood the target with malicious traffic [11].

Brute Force attacks are another common threat, where attackers repeatedly attempt different password combinations to gain unauthorized access to systems or user accounts. Similarly, Botnet attacks involve networks of infected devices controlled remotely by attackers to perform malicious activities such as spam distribution, malware propagation, or large-scale DDoS attacks [1, 5].

Other attacks include Port Scanning, where attackers probe network ports to identify vulnerable services, and Web Attacks, which target web applications through techniques such as SQL Injection and Cross-Site Scripting (XSS). Detecting these attacks is essential for maintaining secure and reliable network infrastructures [2–4].

However, most machine learning models are considered black-box models, meaning that their internal decision-making process is not easily interpretable [8]. This lack of transparency can be problematic, especially in cybersecurity, where understanding the reason behind a detection is important for trust and decision-making [8, 12].

To address this issue, Explainable Artificial Intelligence (XAI) techniques have been developed. These methods aim to make machine learning models more transparent by providing explanations for their predictions. Among these techniques, SHAP (SHapley Additive exPlanations) has gained popularity due to its ability to provide both global and local interpretability of model decisions [12].

In this work, we focus on combining machine learning with explainability techniques to improve the reliability and understanding of network intrusion detection systems.

1.2 Problem Statement

Despite the significant progress achieved using machine learning techniques in network intrusion detection, several challenges still remain [4]. One of the main issues is the imbalance of datasets, where certain attack categories contain a large number of samples while others are significantly underrepresented. This imbalance can negatively affect the performance of classification models, making them biased toward majority classes and reducing their ability to accurately detect minority attacks [13, 14].

Another important challenge is that many machine learning models used in intrusion detection are considered black-box models. Although these models can achieve high classification accuracy, they often fail to provide understandable explanations for their predictions. In cybersecurity applications, understanding why a detection decision is made is essential for trust, transparency, and effective incident response [4, 12].

In addition, certain attack categories, particularly web-based attacks, may share similar characteristics with normal network traffic. This overlap between classes makes the detection process more difficult and can negatively affect classification performance [4, 5].

Therefore, the main problem addressed in this work is how to design an intrusion detection system that achieves good classification performance while also providing interpretable and transparent explanations for its predictions through Explainable Artificial Intelligence (XAI) techniques [8, 12].

1.3 Objectives

The main objective of this work is to design an effective and explainable network intrusion detection system by combining machine learning and Explainable Artificial Intelligence (XAI) techniques.

To achieve this goal, the following objectives are defined:

- To analyze and preprocess the CICIDS2017 dataset in order to prepare it for machine learning tasks.

- To build a classification model based on Random Forest for detecting different types of network attacks.
- To evaluate the performance of the model using standard metrics such as accuracy, precision, recall, and F1-score.
- To address the issue of class imbalance and analyze its impact on the performance of the intrusion detection model.
- To apply explainable AI techniques, specifically SHAP, in order to understand the model's decisions.
- To analyze the most important features contributing to attack detection.

1.4 Contributions

The main contributions of this work can be summarized as follows:

- Development of a machine learning-based intrusion detection system using the Random Forest algorithm for multi-class attack classification.
- Analysis and preprocessing of the CICIDS2017 dataset, including handling class imbalance and improving data quality for classification tasks.
- Investigation of the impact of specific attack categories, particularly web-based attacks, on the overall performance and reliability of the detection model.
- Integration of Explainable Artificial Intelligence (XAI) techniques, specifically SHAP, in order to provide both global and local explanations for the model predictions.

- Identification and analysis of the most influential features contributing to network attack detection and classification decisions.
- Development of an interactive dashboard for visualizing prediction results and explanation outputs in a more understandable and user-friendly manner.
- Comprehensive evaluation of the proposed model using multiple performance metrics to ensure classification reliability and effectiveness.

1.5 Thesis Structure

This thesis is organized into five chapters.

Chapter 1 introduces the general context of the study, including cybersecurity challenges, intrusion detection systems, detection approaches, the problem statement, objectives, and the main contributions of this work.

Chapter 2 presents a review of the related literature and theoretical foundations, focusing on intrusion detection systems, machine learning techniques, data imbalance challenges, explainable artificial intelligence methods, and existing research works in the field.

Chapter 3 describes the methodology adopted in this work, including dataset description, data preprocessing, model development, explainability integration, and evaluation methods.

Chapter 4 presents the experimental results and analysis, including classification performance, confusion matrix analysis, feature importance, SHAP-based explanations, and discussion of the obtained results.

Finally, Chapter 5 concludes the thesis by summarizing the main findings, discussing the limitations of the proposed approach, and presenting possible directions for future work.

Chapter 2

Literature Review

2.1 Network Intrusion Detection Systems (NIDS)

Network Intrusion Detection Systems (NIDS) are designed to monitor network traffic in order to detect malicious activities and potential security threats [2, 3]. They play an important role in protecting computer networks by analyzing data packets and identifying abnormal or suspicious behavior [1].

NIDS can be generally classified into two main types: signature-based detection and anomaly-based detection. Signature-based systems rely on predefined patterns of known attacks. They are effective in detecting previously identified threats but are limited when facing new or unknown attacks [3].

On the other hand, anomaly-based systems attempt to detect deviations from normal behavior. These systems build a model of normal network activity and flag any significant deviation as a potential intrusion. While they are more capable of detecting new attacks, they often suffer from higher false positive rates due to the difficulty of accurately modeling normal behavior [4, 15].

2.1.1 Comparison Between Signature-Based and Anomaly-Based Detection

Signature-based and anomaly-based intrusion detection approaches each have their own advantages and limitations. Signature-based systems are generally more accurate when detecting previously known attacks because they rely on predefined attack patterns. In addition, they usually generate fewer false positive alerts compared to anomaly-based methods [3].

However, signature-based systems are unable to detect zero-day attacks or previously unseen threats that do not match existing signatures [3]. As cyber attacks continuously evolve, maintaining updated signature databases becomes increasingly difficult [1].

In contrast, anomaly-based systems are capable of detecting unknown attacks by identifying deviations from normal network behavior. These approaches are more flexible and adaptive, especially in dynamic network environments [4]. Nevertheless, defining normal behavior accurately remains challenging, which often leads to higher false positive rates.

For this reason, modern intrusion detection research increasingly relies on machine learning techniques to improve anomaly detection capabilities and enhance the adaptability of IDS solutions.

With the increasing complexity of network traffic and cyber threats, traditional intrusion detection systems face several challenges, including scalability, adaptability to evolving attacks, and maintaining a balance between detection accuracy and false alarms [10].

The architecture presented in Figure 2.1 illustrates the integration of Network-based Intrusion Detection Systems (NIDS) and Host-based Intrusion Detection Systems (HIDS) within a network environment.

At the network level, incoming traffic from external sources first passes through a firewall, which filters packets based on predefined security rules [1, 2]. After this initial filtering, the traffic is analyzed by the NIDS, which monitors network flows in real time to detect suspicious patterns or known attack signatures [2, 3].

The traffic is then routed through the network infrastructure via a router, distributing it to different hosts such as user machines, file servers, and web servers. At this stage, HIDS components are deployed on each host to monitor internal activities, including system calls, file access, and application behavior [2].

While NIDS provides a global view of network traffic, HIDS offers a more detailed and localized analysis at the host level. This complementary approach improves the overall detection capability by covering both external and internal threats [1].

Finally, alerts and logs generated by both NIDS and HIDS are collected and forwarded to a Security Information and Event Management (SIEM) system, where they are aggregated, correlated, and analyzed to support incident detection and response [16].

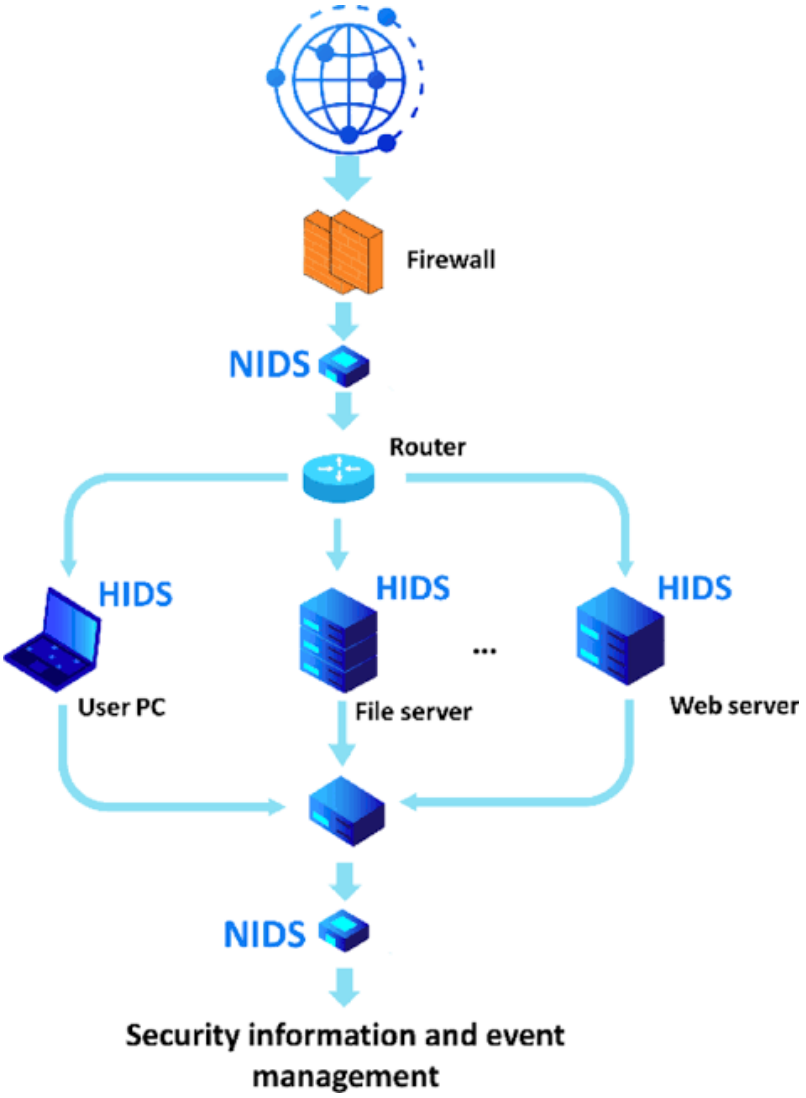


FIGURE 2.1: Example architecture of a Network Intrusion Detection System (NIDS) integrated with Host-based IDS (HIDS) in a network environment.

2.2 Machine Learning in Intrusion Detection

Machine learning has been widely adopted in network intrusion detection systems to improve their ability to detect both known and unknown attacks. Unlike traditional methods, machine learning models can learn patterns from data and make predictions based on observed behaviors.

Several machine learning techniques have been applied in this domain, including supervised and unsupervised learning approaches. Supervised learning methods, such as Decision Trees, Support Vector Machines, and Random Forest, are commonly used when labeled data is available. These models learn from historical data to classify network traffic into normal or malicious categories [4].

Among these techniques, Random Forest has gained popularity due to its robustness and ability to handle large datasets with high-dimensional features. It is an ensemble learning method that combines multiple decision trees to improve classification accuracy and reduce overfitting [6, 9].

Unsupervised learning methods, on the other hand, are used when labeled data is limited. These approaches focus on identifying anomalies or unusual patterns in network traffic without prior knowledge of attack labels [5].

2.2.1 Deep Learning Approaches in Intrusion Detection

In recent years, deep learning techniques have attracted significant attention in the field of intrusion detection due to their ability to automatically learn complex patterns from large-scale network traffic data. Deep learning models such as Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN) have been successfully applied to detect sophisticated cyber attacks [7].

Compared to traditional machine learning methods, deep learning approaches can achieve higher detection performance when dealing with complex and high-dimensional datasets. These models are particularly effective in capturing hidden relationships and temporal patterns within network traffic data [7].

However, deep learning models generally require large training datasets, high computational resources, and longer training times [7]. In addition, they often suffer from limited interpretability, making it difficult to understand the reasoning behind their predictions [8]. This limitation has increased the importance of Explainable Artificial Intelligence (XAI) techniques in cybersecurity applications [12].

Despite their advantages, machine learning-based intrusion detection systems still face several challenges. These include handling imbalanced datasets, selecting relevant features, and ensuring that the models generalize well to unseen data [4, 14].

Therefore, improving the performance of machine learning models remains an important research direction in this field [17].

2.2.2 Related Work

Several research works have explored the use of machine learning techniques for network intrusion detection. Decision Tree and Random Forest models are among the most commonly used approaches due to their good classification performance and relatively low computational complexity [9, 17].

Sommer and Paxson [4] discussed the limitations and practical challenges of applying machine learning techniques in real-world intrusion detection systems. Their work highlighted issues related to data quality, model generalization, and evolving attack patterns.

Other studies have investigated the use of deep learning models for intrusion detection. These approaches demonstrated promising results in detecting complex attacks but often required significant computational resources and large training datasets.

More recently, researchers have started integrating Explainable Artificial Intelligence (XAI) methods into intrusion detection systems in order to improve transparency and interpretability. Techniques such as SHAP and LIME are increasingly used to explain model predictions and identify the most influential features contributing to attack detection [12, 18].

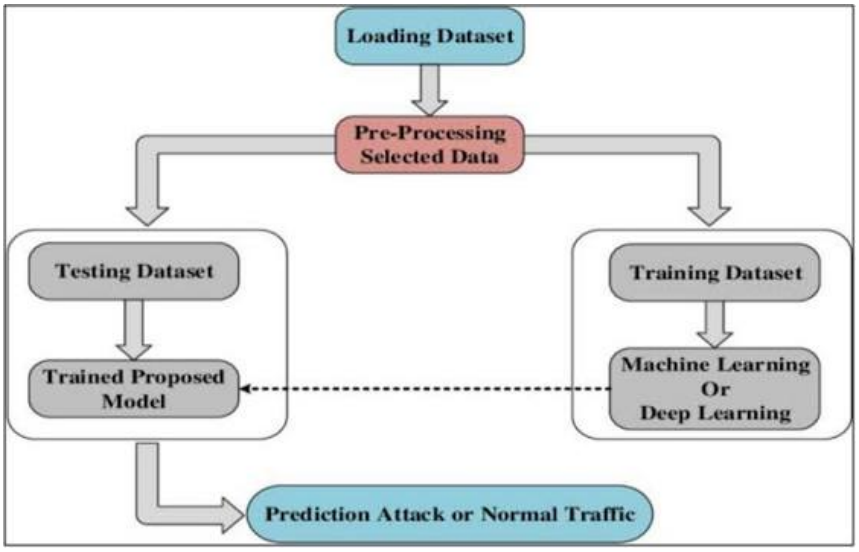


FIGURE 2.2: General workflow of a machine learning-based intrusion detection system, including data preprocessing, training, testing, and prediction phases.

2.3 Challenges of Imbalanced Data

One of the main challenges in network intrusion detection is the imbalance of datasets, where some classes are significantly underrepresented compared to others [4, 14]. In many real-world datasets, such as CICIDS2017, certain classes contain a large number of instances, while others have very few samples. This imbalance can significantly affect the performance of machine learning models [14].

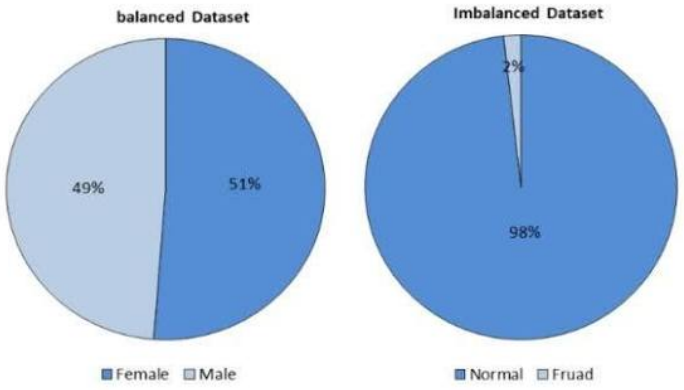


FIGURE 2.3: Comparison between balanced and imbalanced datasets. In imbalanced data, one class dominates the others, which can negatively affect the performance of machine learning models.

When trained on imbalanced data, classification models tend to favor the majority classes, as they dominate the learning process. As a result, the model may achieve high overall accuracy while performing poorly on minority classes, which are often the most critical to detect in cybersecurity contexts [4, 15].

This issue becomes more critical when dealing with rare attack types, where the number of samples is insufficient for the model to learn meaningful patterns. In some cases, these classes may even be misclassified as normal traffic due to similarities in their features.

Several techniques have been proposed to address this problem, including oversampling, undersampling, and synthetic data generation approaches such as the Synthetic Minority Over-sampling Technique (SMOTE) [13, 15]. These methods aim to balance the dataset either by increasing the number of minority class samples or by reducing the number of majority class instances in order to improve classification performance.

However, these techniques do not always guarantee improved performance, especially when the features of certain classes overlap with others [14]. In such cases, balancing the dataset may not be sufficient to achieve good classification results.

In cybersecurity applications, this issue is particularly important because minority classes may represent critical attack types that can cause severe damage if not detected correctly.

Therefore, handling imbalanced data remains a key challenge in designing reliable intrusion detection systems.

2.4 Explainable Artificial Intelligence (XAI)

With the increasing use of machine learning models in critical domains such as cybersecurity, the need for interpretability has become more important. Many high-performing models, including ensemble methods, are often considered black-box models, meaning that their internal decision-making process is not easily understandable [8, 12].

In the context of intrusion detection, this lack of transparency can be problematic. Security analysts need to understand why a certain network activity is classified as an attack in order to take appropriate actions and trust the system's decisions [8].

Explainable Artificial Intelligence (XAI) aims to address this issue by providing methods that make machine learning models more transparent and interpretable. These techniques help in understanding how input features contribute to the final prediction of a model [12].

2.4.1 Interpretability Approaches

Interpretability techniques in machine learning can generally be divided into two main categories: intrinsic interpretability and post-hoc interpretability [8, 19, 20].

Intrinsic interpretability refers to models that are naturally understandable due to their simple structure. Examples include Decision Trees, linear regression models, and rule-based systems. These models provide explanations directly through their internal structure and decision rules [8, 21].

In contrast, post-hoc interpretability techniques are applied after the model has been trained. These methods are mainly used with complex black-box models in order to explain their predictions without modifying the internal structure of the model [12, 18, 19]. Post-hoc approaches are widely used in modern machine learning applications because many high-performance models lack inherent transparency [8, 20, 21].

One of the most widely used XAI techniques is SHAP (SHapley Additive exPlanations). SHAP is based on concepts from cooperative game theory and assigns an importance value to each feature for a given prediction. It provides both global explanations, which describe the overall behavior of the model, and local explanations, which explain individual predictions [12].

Another popular XAI technique is LIME (Local Interpretable Model-agnostic Explanations), which explains individual predictions by approximating the behavior of a complex model locally using simpler interpretable models. LIME is particularly useful for understanding specific predictions and identifying the factors that influence individual classification decisions [18].

Other interpretability methods include Feature Importance analysis and Partial Dependence Plots (PDPs). Feature Importance techniques evaluate the contribution of each feature to the overall model performance, while PDPs illustrate how changes in a feature affect the model predictions. These approaches help improve transparency and provide deeper insights into model behavior [8, 22].

By using SHAP, it is possible to identify the most influential features and understand how they impact the model's decisions [8, 12]. This makes it a powerful tool for analyzing and validating machine learning models in intrusion detection systems [4, 7].

In this work, SHAP is adopted as the primary explainability technique in order to improve the interpretability of the proposed intrusion detection model [12, 19] and provide meaningful insights into the factors influencing attack detection decisions [12, 18].

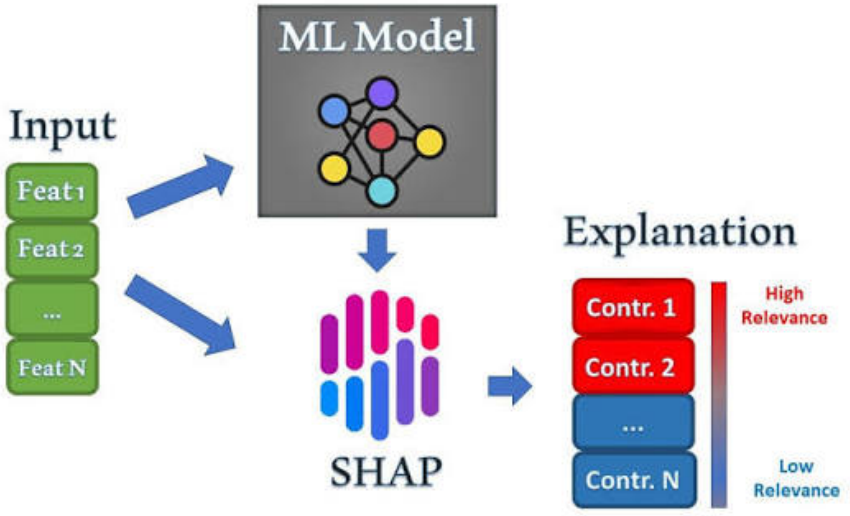


FIGURE 2.4: Illustration of the SHAP (SHapley Additive exPlanations) approach, showing how input features contribute to the model prediction by assigning importance scores to each feature.

Chapter 3

Study Methodology

3.1 Overview of the Methodology

The methodology adopted in this work aims to design and implement an explainable multi-class Intrusion Detection System (IDS) capable of accurately identifying different types of network attacks while providing interpretable insights into model decisions [10, 19]. The proposed approach integrates machine learning techniques with explainability methods to enhance both detection performance and transparency [7, 12, 19].

The overall process follows a structured pipeline that begins with data acquisition and preprocessing, followed by model training, evaluation, and finally interpretability and system deployment [6, 8]. Initially, the dataset is prepared by removing irrelevant or inconsistent classes and applying necessary preprocessing steps, including encoding categorical features and organizing the feature space [23]. These steps are essential to ensure data consistency and improve the learning capability of the model [6, 14].

Subsequently, a supervised learning approach is employed using the Random Forest algorithm, which is widely recognized for its robustness and effectiveness in handling high-dimensional data in intrusion detection tasks [9]. The dataset is divided into training and testing subsets using a stratified split to preserve class distribution. In addition, a k-fold cross-validation strategy is incorporated to provide a more reliable evaluation of the model's generalization performance and reduce the impact of data variability [6].

To address the critical need for transparency in cybersecurity systems, the proposed methodology integrates SHAP (SHapley Additive exPlanations) as an explainability technique. SHAP enables both global and local interpretation of model predictions by quantifying the contribution of each feature to the final decision [12]. This is particularly important in intrusion detection systems, where understanding the reasoning behind a prediction is essential for trust and practical deployment [8, 12].

Finally, the trained model is integrated into an interactive dashboard developed using Streamlit. This interface allows users to input data manually or upload datasets, obtain predictions, and visualize explanations in real time. The end-to-end system therefore combines data processing, machine learning, explainability, and user interaction into a unified framework.

The complete pipeline of the proposed system can be summarized as follows:

$$\textit{Data Collection} \rightarrow \textit{Preprocessing} \rightarrow \textit{Model Training} \rightarrow \textit{Evaluation} \rightarrow \textit{Explainability (SHAP)} \rightarrow \textit{Deployment (Dashboard)}$$

3.2 Dataset Description

The performance of any Intrusion Detection System (IDS) largely depends on the quality and relevance of the dataset used for training and evaluation. In this work, a network intrusion dataset derived from the CICIDS framework is utilized, which is widely adopted in the cybersecurity research community due to its realistic representation of modern network traffic and diverse attack scenarios [23].

The dataset contains a mixture of normal and malicious network traffic, where each instance is described by a set of statistical and flow-based features [23]. These features capture various characteristics of network behavior, such as packet length, flow duration, and protocol-specific attributes [17, 23]. Such feature representations enable machine learning models to effectively distinguish between benign and malicious activities [7, 17].

A key aspect of this dataset is its multi-class structure, where different types of attacks are labeled separately rather than grouped into a single malicious category [23]. This is particularly important for practical intrusion detection systems, as it allows for fine-grained identification of attack types such as Distributed Denial of Service (DDoS), PortScan, Bot, and others [10, 11]. Multi-class classification provides more actionable insights compared to binary classification, which only distinguishes between normal and attack traffic [7, 24].

During the preprocessing phase, the class labeled *WebAttack* was removed from the dataset. This decision was motivated by its relatively low representation and potential inconsistency compared to other attack categories, which could negatively impact the learning process and bias the model [13, 14]. Removing such classes helps improve the stability and reliability of the classification model [14].

Figure 3.1 illustrates the distribution of classes in the dataset before preprocessing. It can be observed that the dataset is highly imbalanced, as the BENIGN class contains a significantly larger number of samples compared to several attack classes such as WebAttack and Brute Force. Although the selected attack categories provide a suitable foundation for building a multi-class intrusion detection model, this imbalance may negatively affect the learning process by biasing the model toward majority classes and reducing its ability to correctly detect minority attacks [14].

```
Label
BENIGN         2271320
DoS             251712
PortScan        158804
DDoS            128025
Bot              15835
Brute Force       1507
WebAttack          673
Name: count, dtype: int64
```

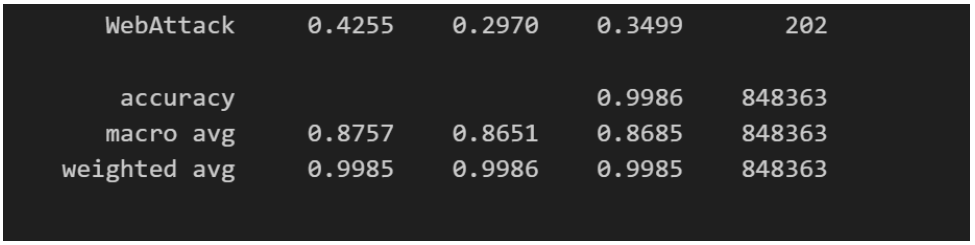
FIGURE 3.1: Distribution of classes in the dataset before preprocessing

3.3 Data Preprocessing

Data preprocessing is a crucial step in building an effective Intrusion Detection System (IDS), as raw network data often contains inconsistencies, irrelevant information, and categorical attributes that cannot be directly processed by machine learning models. Data preprocessing is a crucial step in building an effective Intrusion Detection System (IDS), as raw network traffic data often contains missing values, redundant attributes, categorical features, and inconsistencies that cannot be directly processed by machine learning models. In this work, the preprocessing phase included handling missing and infinite values, removing irrelevant features, encoding categorical attributes, and normalizing the numerical features before model

training. These operations improve data quality, enhance model performance, and increase the reliability of the obtained results [6].

3.3.1 Handling Low-Performance Classes



WebAttack	0.4255	0.2970	0.3499	202
accuracy			0.9986	848363
macro avg	0.8757	0.8651	0.8685	848363
weighted avg	0.9985	0.9986	0.9985	848363

FIGURE 3.2: Classification performance including the WebAttack class before preprocessing

As shown in Figure 3.2, the model exhibited significantly lower performance when including the *WebAttack* class, with notably reduced precision, recall, and F1-score compared to other classes.

This behavior suggests that the WebAttack class introduces noise or ambiguity into the classification task, potentially due to overlapping feature patterns with other attack types or insufficient representation in the dataset [13, 14].

To improve the stability and overall performance of the model, this class was excluded during preprocessing. This decision allowed the model to focus on more distinguishable attack categories and achieve more reliable classification results [6, 14].

In this work, several preprocessing steps were applied to prepare the dataset for training. First, the class labeled *WebAttack* was removed due to its relatively low representation and potential inconsistency compared to other attack categories [13, 14]. This step helps reduce noise in the dataset and prevents the model from being biased toward underrepresented classes [14].

Next, categorical features present in the dataset, such as Protocol, Destination Port Category, and Traffic Direction, were transformed into numerical representations using the Label Encoding technique. Since machine learning algorithms such as Random Forest require numerical input, this transformation is essential for enabling the model to process all available features [6, 9]. Label Encoding assigns a unique integer value to each category by converting textual labels into numerical form, which facilitates their use in machine learning models [6]. For example, the protocol values *TCP*, *UDP*, and *ICMP* may be encoded as 0, 1, and 2, respectively. Each categorical feature was encoded independently, and the corresponding encoders were stored for later use during the prediction phase in order to ensure consistency between training and deployment [6].

After encoding, the dataset was divided into input features and target labels. The feature set includes all network-related attributes, while the target variable represents the class label associated with each instance [23]. To maintain consistency between the training phase and the deployment phase, the list of features used by the model was saved and reused during inference [6].

Finally, the dataset was split into training and testing subsets using a stratified sampling approach [25]. This ensures that the distribution of classes remains consistent across both subsets, which is particularly important in multi-class classification problems [13, 14]. Maintaining class distribution helps the model generalize better and provides a more reliable evaluation of its performance [6, 25].

Overall, these preprocessing steps transform the raw dataset into a clean and structured format suitable for machine learning, forming a solid foundation for the subsequent model training phase [6, 17].

Figure 3.3 summarizes the main preprocessing steps applied to the dataset before the training phase. The workflow includes data cleaning, handling missing values, feature selection, categorical encoding, and dataset preparation for machine learning.

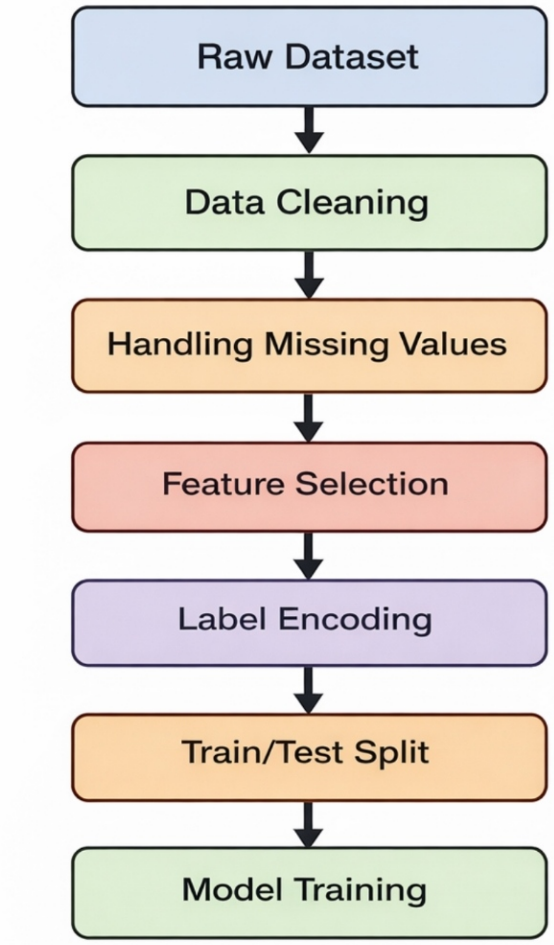


FIGURE 3.3: Overview of the data preprocessing pipeline used in this work.

3.4 Model Selection and Training

The selection of an appropriate machine learning model is a critical step in the design of an effective Intrusion Detection System (IDS). In this work, the Random Forest

algorithm was chosen due to its robustness, high performance in classification tasks, and ability to handle high-dimensional data efficiently [9].

Random Forest is an ensemble learning method that constructs multiple decision trees during training and combines their outputs to improve predictive accuracy and reduce overfitting [6, 9]. This characteristic makes it particularly suitable for intrusion detection scenarios, where data can be complex, noisy, and highly variable [7, 9, 17]. Additionally, Random Forest provides an inherent measure of feature importance, which is beneficial for later interpretability analysis [9, 12].

The model was trained using a set of hyperparameters selected to balance performance and computational efficiency. Specifically, the number of trees was set to 150, and the maximum depth of each tree was limited to 25. These values were chosen empirically to ensure that the model captures complex patterns in the data without overfitting.

To evaluate the model's ability to generalize, the dataset was divided into training and testing subsets using a stratified split, ensuring that the class distribution remains consistent across both sets. The model was trained on the training set and evaluated on the test set using standard classification metrics.

In addition to the traditional train-test split, a 3-fold cross-validation strategy was applied to further assess the robustness of the model. Cross-validation allows the model to be trained and evaluated on multiple subsets of the data, providing a more reliable estimate of its performance and reducing the impact of random variations in the dataset [6].

Finally, the trained model was saved and later integrated into the deployed system for real-time prediction. This ensures consistency between the training phase and the operational phase of the intrusion detection system.

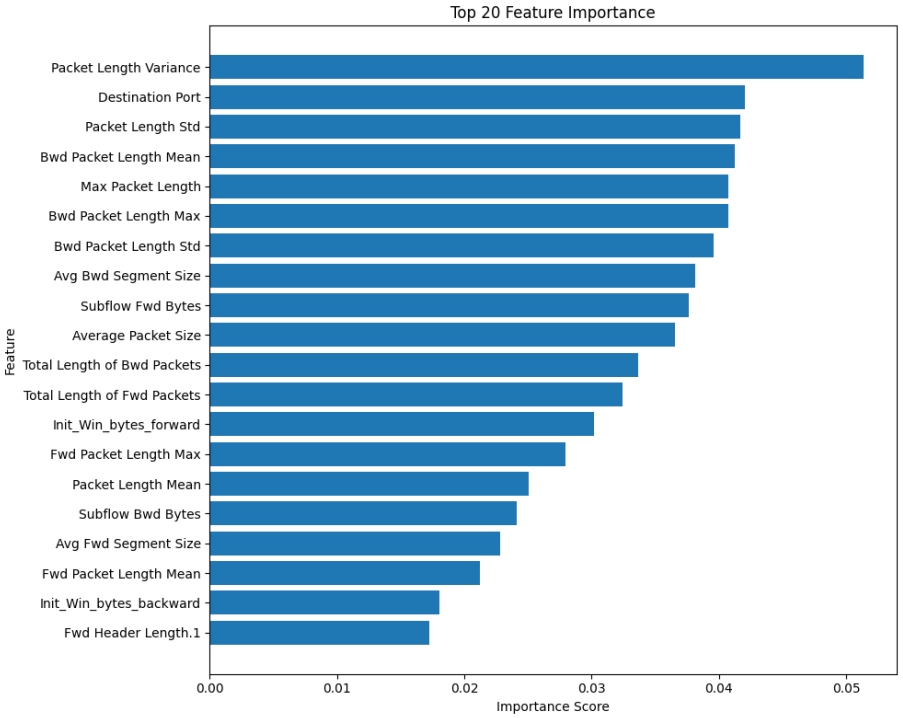


FIGURE 3.4: Top 20 most important features according to the Random Forest model

3.5 Evaluation Strategy

Evaluating the performance of an Intrusion Detection System (IDS) is essential to ensure its effectiveness in detecting and classifying different types of network attacks [7, 10]. In this work, a comprehensive evaluation strategy was adopted, combining both a train-test split and cross-validation techniques to obtain reliable and robust performance estimates [6, 25].

Initially, the dataset was divided into training and testing subsets using a stratified split, with 70% of the data used for training and 30% for testing [25]. Stratification ensures that the distribution of classes remains consistent across both subsets, which is particularly important in multi-class classification problems where class imbalance may exist [13, 14, 25].

To further strengthen the evaluation, a 3-fold cross-validation approach was employed. In this method, the dataset is partitioned into three subsets, and the model is trained and evaluated three times, each time using a different subset as the validation set while the remaining subsets are used for training. This process provides a more reliable estimate of the model's generalization performance and reduces the risk of biased results due to a single data split [6].

The performance of the model was assessed using standard classification metrics, including precision, recall, and F1-score [6, 24]. Precision measures the proportion of correctly predicted positive instances among all predicted positives, while recall evaluates the model's ability to correctly identify actual positive instances [24]. The F1-score provides a balanced measure by combining both precision and recall, making it particularly suitable for multi-class classification scenarios [14, 24].

In addition to these metrics, a confusion matrix was used to provide a detailed visualization of the model's classification performance [6, 26]. The confusion matrix highlights the number of correct and incorrect predictions for each class, allowing for a deeper analysis of misclassification patterns and model behavior across different attack types [7, 26].

Overall, this evaluation strategy ensures a comprehensive and reliable assessment of the model, combining quantitative metrics with visual analysis tools to validate its effectiveness in intrusion detection tasks [7, 10, 25].

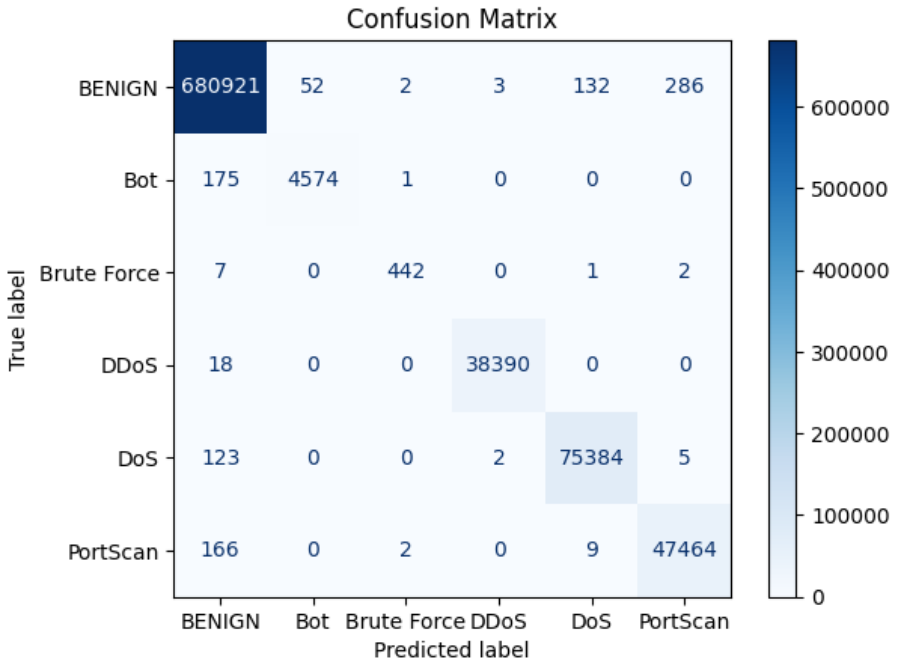


FIGURE 3.5: Confusion matrix of the Random Forest model on the test set

```
===== CROSS VALIDATION (3-FOLD) =====
CV Scores: [0.97611845 0.99856961 0.99061228]
Mean CV Accuracy: 0.9884334446447601
(1979042, 78)
(848161, 78)
rf model saved successfully

===== FINAL CLASSIFICATION REPORT =====
              precision    recall  f1-score   support

      BENIGN     0.9993    0.9993    0.9993    681396
         Bot     0.9888    0.9629    0.9757      4750
 Brute Force     0.9888    0.9779    0.9833       452
        DDoS     0.9999    0.9995    0.9997     38408
         DoS     0.9981    0.9983    0.9982     75514
    PortScan     0.9939    0.9963    0.9951     47641

    accuracy                         0.9988    848161
   macro avg     0.9948    0.9890    0.9919    848161
weighted avg     0.9988    0.9988    0.9988    848161
```

FIGURE 3.6: Classification report showing precision, recall, and F1-score for each class

3.6 Explainability using SHAP

While machine learning models can achieve high accuracy in intrusion detection tasks, they are often considered black-box systems due to their lack of interpretability. In cybersecurity applications, understanding the reasoning behind a model's prediction is crucial for building trust and enabling effective decision-making. To address this limitation, this work integrates SHAP (SHapley Additive exPlanations) as an explainability technique [8, 12].

SHAP is based on cooperative game theory and assigns each feature an importance value representing its contribution to the final prediction [8, 12]. This method provides a unified framework for interpreting complex models such as Random Forest, allowing both global and local explanations of model behavior [12, 19].

At the global level, SHAP is used to identify the most influential features across the entire dataset [12]. This helps in understanding which network characteristics play a significant role in distinguishing between different types of attacks [12, 18]. Such insights are valuable for validating the model and gaining domain knowledge about intrusion patterns [7, 19].

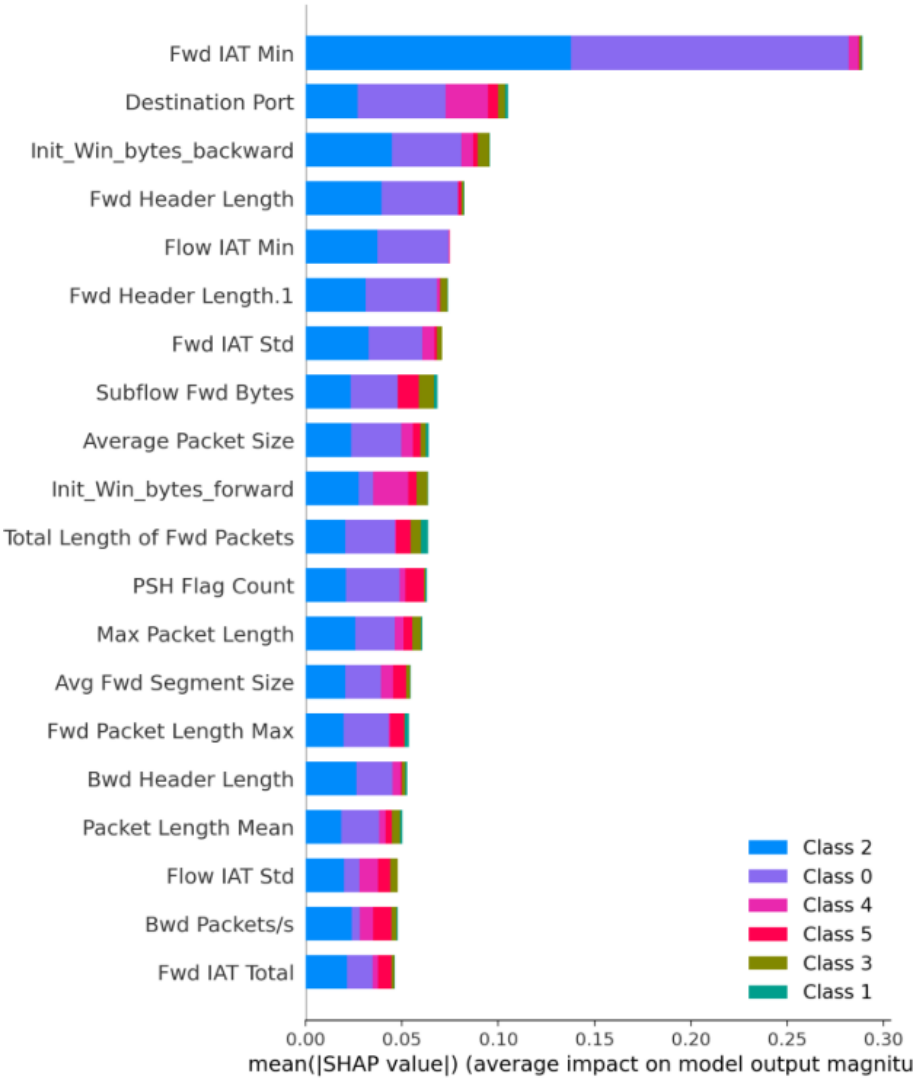


FIGURE 3.7: Global feature importance using SHAP summary plot

At the local level, SHAP explains individual predictions by showing how each feature contributes to a specific classification decision [12, 18]. This is particularly important in intrusion detection systems, where analysts need to understand why a specific traffic instance was classified as a certain type of attack [4, 19]. The local explanation provides a detailed breakdown of feature contributions, indicating which factors pushed the prediction toward or away from a given class [8, 12].

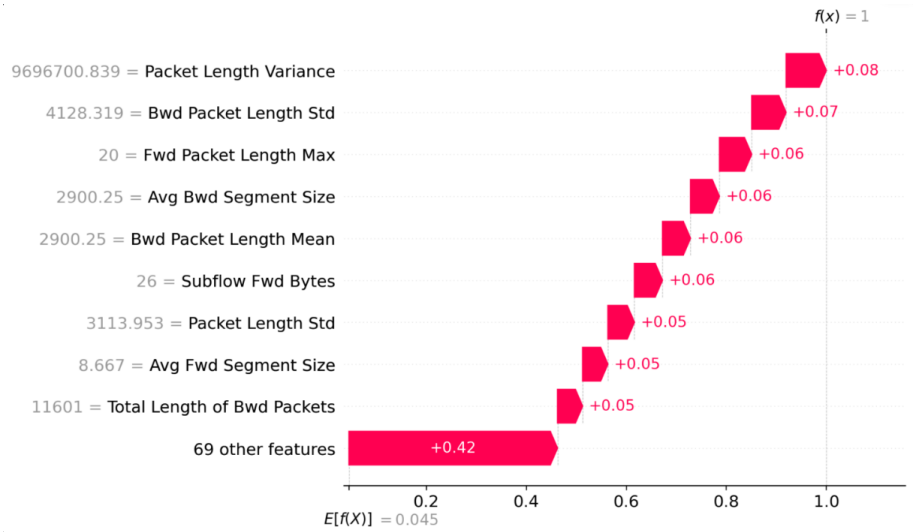


FIGURE 3.8: Local explanation of a single prediction using SHAP waterfall plot

In this work, SHAP was integrated into the system to provide both summary visualizations and detailed explanations for individual predictions [12, 19]. This dual-level interpretability enhances the transparency of the model and supports more informed analysis of network traffic behavior [8, 20].

Overall, the integration of SHAP transforms the proposed IDS from a purely predictive system into an interpretable and trustworthy solution, which is essential for real-world cybersecurity applications [4, 12, 19].

3.7 System Implementation

To make the proposed Intrusion Detection System (IDS) practically usable, the trained model was integrated into an interactive web-based dashboard developed using Streamlit [27]. This interface allows users to interact with the system dynamically, providing both predictions and interpretability features in a user-friendly environment [8, 19].

The implemented system supports two modes of input. The first mode allows users manually enter network traffic feature values, such as protocol type, flow duration, packet length, and connection statistics, through an intuitive interface in order to generate intrusion detection predictions, enabling precise control over the input data [27]. The second mode enables users to upload a CSV file containing multiple network traffic instances. In this case, a random sample is automatically selected from the uploaded dataset for prediction, simplifying the interaction and improving usability [6].

Once the input data is provided, the system processes it using the same preprocessing pipeline applied during training, ensuring consistency in feature representation [6, 23]. The trained Random Forest model is then used to generate predictions, identifying the type of network traffic (e.g., Benign, DDoS, PortScan, etc.) [7, 9].

Figure 3.9 illustrates the prediction workflow of the proposed intrusion detection system. The process starts with user-provided network traffic features, followed by data preprocessing and feature transformation. The processed data is then passed to the trained Random Forest model to generate the final prediction and corresponding explainability results.

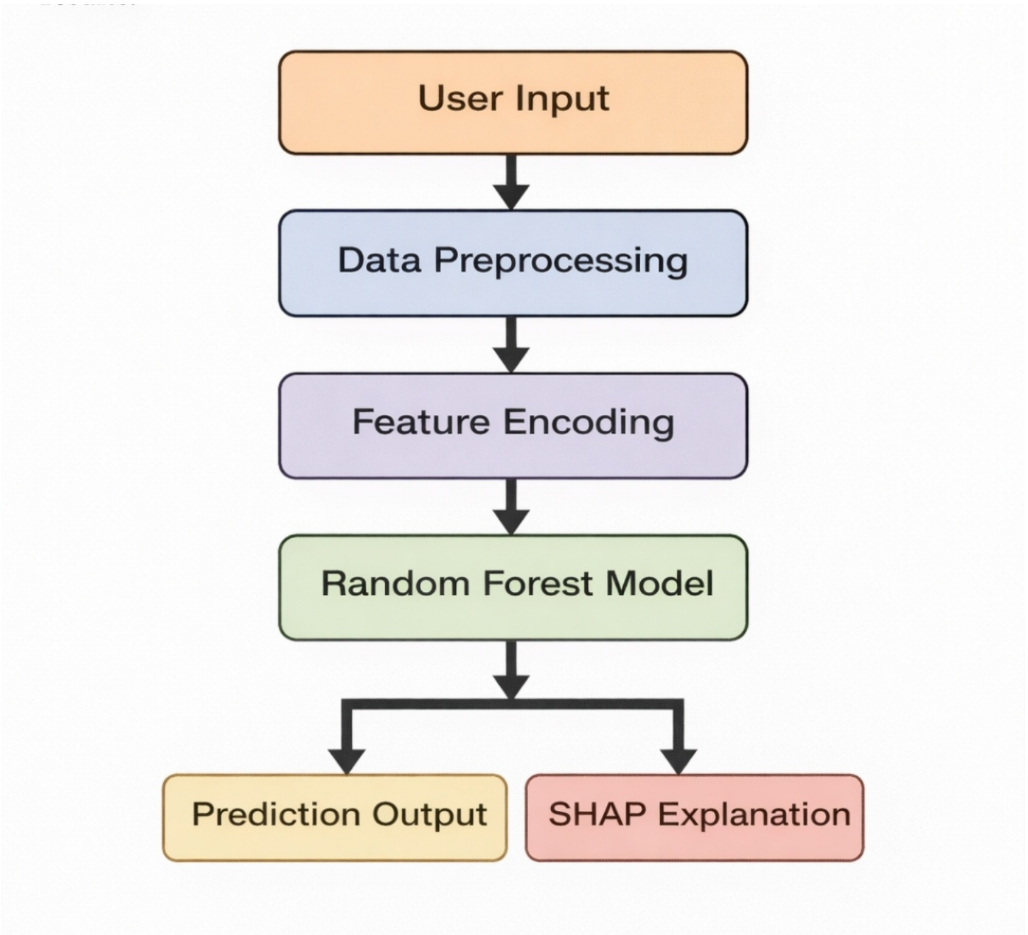


FIGURE 3.9: Prediction workflow of the proposed intrusion detection system.

In addition to prediction, the dashboard includes an explainability module based on SHAP [12]. Users can visualize both global feature importance and local explanations for individual predictions [12, 19]. This allows users not only to see the predicted class but also to understand the factors influencing the decision [8, 18].

The interface is organized into multiple sections, including prediction results, input data summary, and explainability visualizations [27]. This structured layout improves readability and provides a seamless user experience [19].

Overall, the implementation of the system as an interactive dashboard bridges the

gap between theoretical modeling and practical application [27], making the proposed IDS accessible, interpretable, and ready for real-world usage [4, 19].

The developed intrusion detection system was implemented using a Streamlit-based interactive interface. The interface allows users to manually provide network traffic features or upload CSV files for prediction and analysis. In addition to attack classification, the system also integrates SHAP-based explainability to help users understand the reasoning behind the model predictions.

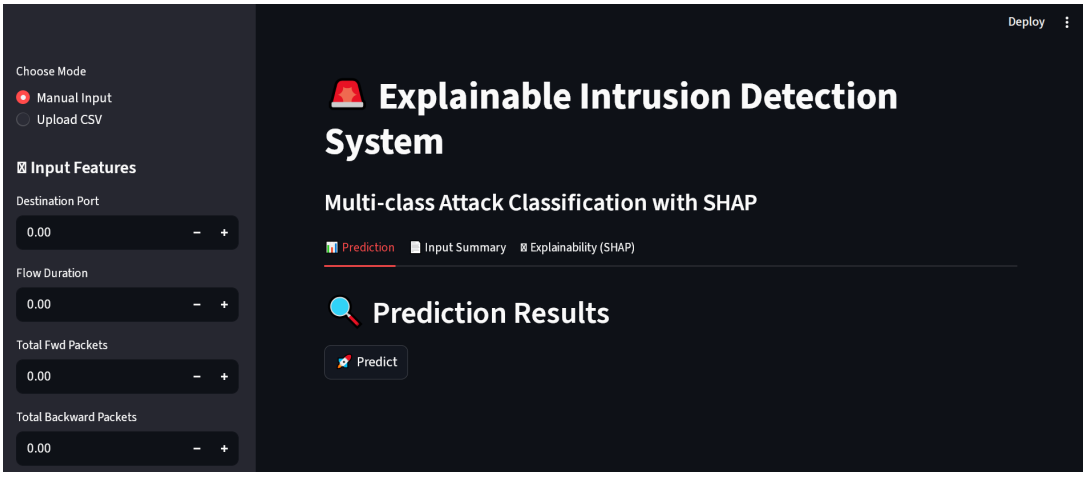


FIGURE 3.10: Main interface of the proposed Explainable Intrusion Detection System implemented using Streamlit.

Figure 3.11 presents the CSV-based prediction mode of the developed intrusion detection system. In this mode, users can upload a CSV file containing network traffic feature values instead of entering them manually. After preprocessing the uploaded data, the trained Random Forest model generates predictions for the detected traffic classes.

The interface also displays a summary of the uploaded features and the corresponding prediction results in a structured and user-friendly format.

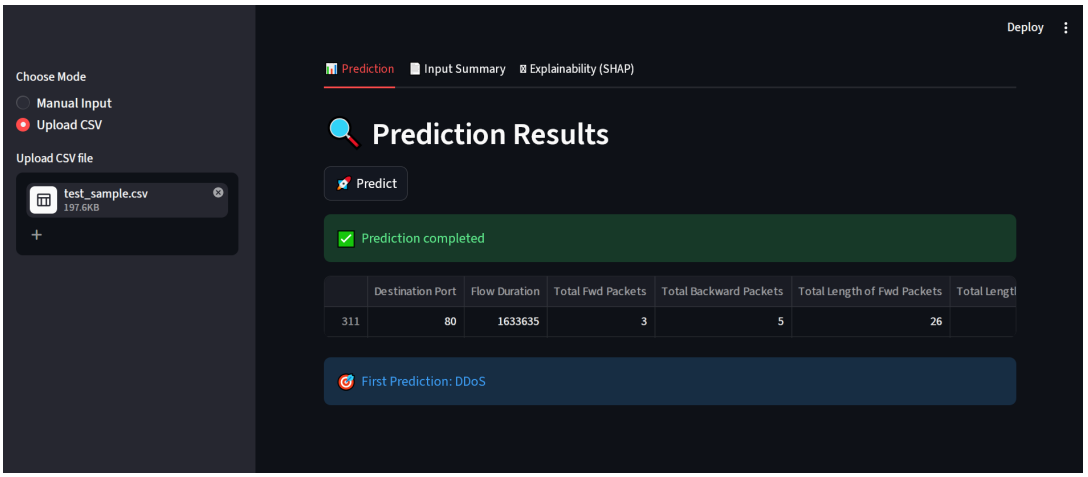


FIGURE 3.11: The prediction interface of the proposed intrusion detection system showing uploaded traffic data and generated attack classification results.

Chapter 4

Results and Discussion

4.1 Introduction to Results

This chapter presents and analyzes the results obtained from the implementation of the proposed Intrusion Detection System (IDS). The objective is not only to evaluate the performance of the model but also to provide a detailed interpretation of its behavior across different types of network traffic.

The analysis focuses on several key aspects, including classification performance, feature importance, and explainability. These components are essential for assessing both the effectiveness and reliability of the system in real-world scenarios [7, 10].

First, the overall performance of the model is examined using standard evaluation metrics and visual tools such as the confusion matrix [26]. This allows for a detailed understanding of how well the model distinguishes between different classes of network traffic.

Next, the stability of the model is analyzed through cross-validation results [25], providing insight into its generalization capability. The contribution of individual features is then explored using feature importance analysis, highlighting the most influential factors in the classification process [9].

Finally, the interpretability of the model is investigated using SHAP-based explanations, offering both global and local insights into the model's decision-making process [12, 19].

Through this comprehensive analysis, this chapter aims to validate the effectiveness of the proposed approach and identify both its strengths and potential limitations.

4.2 Model Performance Analysis

The performance of the proposed Intrusion Detection System (IDS) was evaluated using a combination of quantitative metrics and visual analysis tools [10, 26]. The confusion matrix and classification report provide a comprehensive understanding of the model's ability to correctly classify different types of network traffic [26].

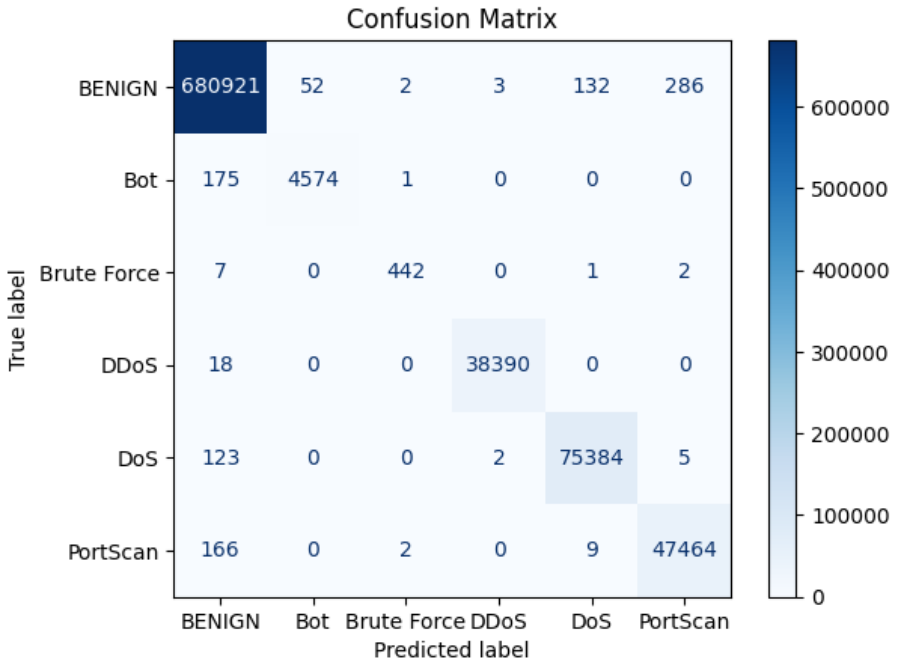


FIGURE 4.1: Confusion matrix of the Random Forest model on the test dataset

The confusion matrix in Figure 4.1 shows that the model achieves a high level of accuracy across all classes. The majority of instances are correctly classified, as indicated by the strong diagonal values.

The model performs exceptionally well in detecting *Benign*, *DDoS*, *PortScan* and *DoS* traffic, with very high true positive rates and minimal misclassifications. This indicates that the model is highly effective in distinguishing normal traffic from large-scale attack patterns.

For the *Bot* class, a small number of instances are misclassified as *Benign*. This suggests that some bot traffic shares characteristics with normal network behavior, making it slightly more challenging for the model to distinguish between the two.

Similarly, minor misclassifications are observed in the *Brute Force* classe. These errors are relatively limited and may be attributed to overlapping feature patterns between certain attack types [15, 17].

Overall, the confusion matrix demonstrates that the model maintains strong classification performance, with only minimal confusion between classes.

```
===== CROSS VALIDATION (3-FOLD) =====
CV Scores: [0.97611845 0.99856961 0.99061228]
Mean CV Accuracy: 0.9884334446447601
(1979042, 78)
(848161, 78)
rf model saved successfully

===== FINAL CLASSIFICATION REPORT =====
              precision    recall  f1-score   support

      BENIGN     0.9993    0.9993    0.9993    681396
         Bot     0.9888    0.9629    0.9757      4750
 Brute Force     0.9888    0.9779    0.9833       452
        DDoS     0.9999    0.9995    0.9997     38408
         DoS     0.9981    0.9983    0.9982     75514
    PortScan     0.9939    0.9963    0.9951     47641

    accuracy                         0.9988    848161
   macro avg     0.9948    0.9890    0.9919    848161
weighted avg     0.9988    0.9988    0.9988    848161
```

FIGURE 4.2: Classification report showing precision, recall, and F1-score for each class

The classification report in Figure 4.2 further confirms the performance of the model . Most classes achieve precision, recall, and F1-score values close to 1.00, indicating highly accurate predictions.

The overall accuracy of the model is extremely high, reaching nearly perfect performance on the test dataset. Both macro and weighted averages also reflect strong and consistent performance across all classes. The macro average computes the metric independently for each class and then takes the unweighted mean, treating all classes equally regardless of their size. In contrast, the weighted average accounts for class imbalance by weighting each class's metric according to its proportion in the dataset [14, 26].

However, slightly lower recall for the *Bot* class indicates that a small number of bot instances are not correctly identified. This aligns with the observations from the confusion matrix and highlights a minor limitation of the model.

In general, the results demonstrate that the proposed model is highly effective for multi-class intrusion detection, achieving excellent performance while maintaining low misclassification rates across different attack categories.

4.3 Cross-Validation Results

To further evaluate the generalization capability of the proposed model, a 3-fold cross-validation strategy was employed [25]. This approach provides a more reliable estimate of the model's performance by evaluating it across multiple data splits [6, 25].

The cross-validation results are summarized as follows:

- Fold 1 Accuracy: 0.9761
- Fold 2 Accuracy: 0.9986
- Fold 3 Accuracy: 0.9906
- Mean Accuracy: 0.9884

These results indicate that the model maintains consistently high performance across different subsets of the dataset. Although slight variations can be observed between folds, the overall accuracy remains close to 1.00, demonstrating strong stability.

The lower accuracy observed in the first fold may be attributed to variations in the distribution of samples, particularly for classes with fewer instances [13, 14].

However, this variation is minimal and does not significantly impact the overall performance of the model.

The high mean accuracy confirms that the model generalizes well and is not overly dependent on a specific train-test split [25]. This is particularly important in intrusion detection systems, where the model must perform reliably on unseen data [4, 7].

Overall, the cross-validation results reinforce the effectiveness of the proposed approach and provide additional confidence in the model's ability to maintain high performance in real-world scenarios [7, 10].

4.4 Feature Importance Analysis

To better understand the behavior of the proposed model, feature importance analysis was conducted using the inherent capabilities of the Random Forest algorithm [9]. This analysis highlights the most influential features contributing to the classification process [8, 12].

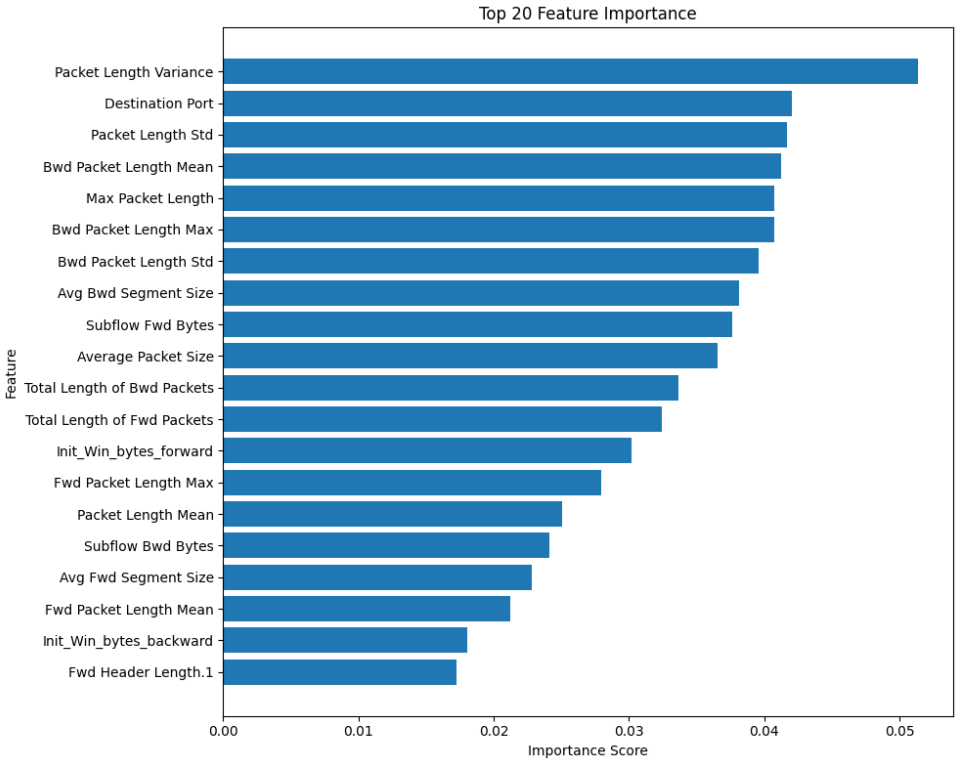


FIGURE 4.3: Top 20 most important features based on the Random Forest model

Figure 4.3 presents the top 20 features ranked by their importance scores. It can be observed that several features related to packet length and traffic volume play a dominant role in the model's decision-making process.

Among the most significant features, *Packet Length Variance* stands out as the most influential. This indicates that variations in packet size are a strong indicator for distinguishing between different types of network traffic [5, 15]. Similarly, features such as *Destination Port*, *Packet Length Standard Deviation*, and *Maximum Packet Length* also contribute significantly to the classification [23].

Additionally, features related to forward and backward packet statistics, such as *Bwd Packet Length Mean* and *Subflow Fwd Bytes*, further enhance the model's ability to capture traffic patterns associated with various attack types [17, 23].

The prominence of these features is consistent with domain knowledge in network security, where anomalies in packet size, flow behavior, and communication patterns are key indicators of malicious activity [3, 5, 15].

Overall, the feature importance analysis confirms that the model relies on meaningful and relevant attributes, which strengthens its interpretability and supports its effectiveness in intrusion detection.

4.5 SHAP-Based Interpretation

To enhance the interpretability of the proposed intrusion detection system, SHAP (SHapley Additive exPlanations) was employed. This method provides both global and local explanations by quantifying the contribution of each feature to the model's predictions [12].

4.5.1 Global Interpretation

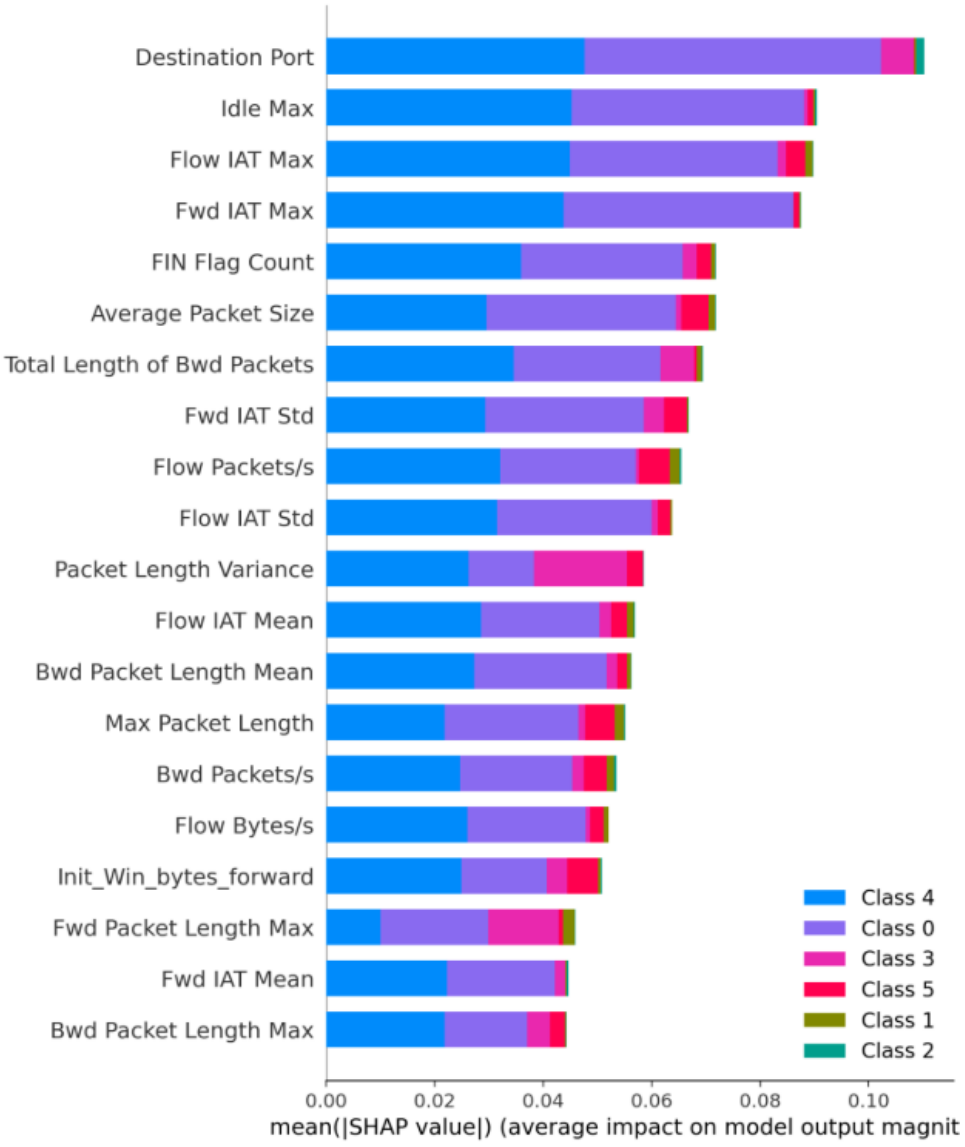


FIGURE 4.4: Global SHAP feature importance across all classes

Figure 4.4 presents the global SHAP summary plot, which highlights the most influential features across all classes [12, 19]. Each bar represents the average absolute SHAP value of a feature, while the colors indicate contributions to different classes.

It can be observed that features such as *Destination Port*, *Idle Max*, *Flow IAT Max*, and *Fwd IAT Max* have the highest impact on the model's predictions. These

features are strongly related to network timing and communication behavior, which are critical indicators in detecting malicious traffic [5, 15].

The color distribution within each bar shows how different classes are influenced by the same feature. For instance, certain features contribute more significantly to specific attack types, reflecting the model's ability to distinguish between multiple intrusion patterns [12, 23].

4.5.2 Local Interpretation (DoS Example)

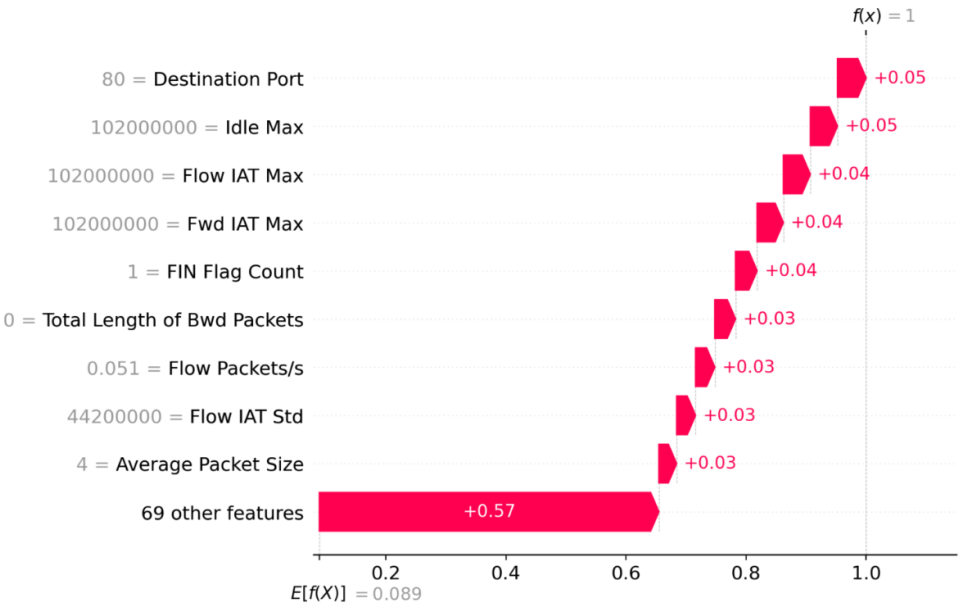


FIGURE 4.5: SHAP waterfall plot for a DoS prediction

Figure 4.5 illustrates the SHAP waterfall plot for a single instance classified as a DoS attack. This plot explains how individual feature contributions lead to the final prediction [12, 18].

The baseline value $E[f(X)]$ represents the average model output, while each feature either increases or decreases the prediction score [12]. In this case, several features

such as *Destination Port*, *Idle Max*, *Flow IAT Max*, and *Fwd IAT Max* contribute positively toward the DoS classification [5].

Notably, the aggregation of multiple smaller contributions (represented as "69 other features") also plays a significant role in pushing the prediction toward the final output [12]. This demonstrates that the model does not rely solely on a single feature but rather on a combination of traffic characteristics [8, 9].

The SHAP analysis confirms that the model's decisions are based on meaningful and interpretable patterns, particularly those related to traffic intensity, timing intervals, and packet behavior, which are well-known indicators of DoS attacks [5, 12, 15].

Overall, the integration of SHAP enhances the transparency of the system and provides valuable insights into how and why specific predictions are made [12, 19].

4.6 Discussion

The results obtained from the experimental evaluation highlight both the effectiveness and the practical relevance of the proposed intrusion detection system. By leveraging a Random Forest classifier combined with SHAP-based explainability, the system achieves a balance between predictive performance and interpretability, which is often lacking in traditional machine learning-based IDS approaches [4, 7, 19].

From a performance perspective, the model demonstrates near-perfect classification accuracy across all classes. The confusion matrix shows a strong concentration of predictions along the diagonal, indicating that the classifier successfully distinguishes between benign traffic and multiple attack categories [26]. However, a closer examination reveals minor misclassifications, particularly between *BENIGN* and certain

attack types such as *DoS*. This phenomenon can be attributed to the inherent similarity in traffic patterns, where high-volume benign traffic may resemble low-intensity attack behavior. Such ambiguity reflects a known limitation in anomaly-based intrusion detection systems [4].

The cross-validation results further strengthen the reliability of the model. With a mean accuracy of approximately 98.84%, the classifier exhibits strong generalization across different subsets of the data [25]. The relatively low variance between folds suggests that the model captures stable and consistent patterns rather than overfitting to specific samples [9, 25]. This is particularly important in cybersecurity contexts, where models must remain robust against unseen and evolving threats [7, 17].

The feature importance analysis provides valuable insights into the decision-making process of the model. Features related to packet length, flow duration, and inter-arrival times emerge as dominant contributors [23]. These findings are consistent with established domain knowledge, where abnormal traffic behavior—such as irregular packet sizes or unusual timing patterns—serves as a key indicator of malicious activity [5, 15]. The prominence of these features confirms that the model is learning meaningful representations rather than relying on spurious correlations [9, 12].

The integration of SHAP significantly enhances the interpretability of the system [12, 19]. At a global level, the SHAP summary plot reveals how different features contribute across multiple classes, highlighting the multi-dimensional nature of the classification problem [12]. At a local level, the waterfall plot provides a transparent explanation for individual predictions [12, 18]. For example, in the DoS case, features such as *Destination Port*, *Flow IAT Max*, and *Idle Max* contribute positively to the final decision [5]. This aligns with the expected behavior of DoS attacks, which are characterized by abnormal traffic bursts and timing irregularities [11, 15].

An important observation is that the model relies on a combination of features rather than a single dominant attribute. The cumulative contribution of multiple features, as illustrated in the SHAP waterfall plot, indicates a distributed decision-making process [12]. This reduces the risk of bias and enhances the robustness of the classifier, as decisions are not overly dependent on isolated variables [8, 9].

Despite these strengths, it is important to acknowledge that the model operates within a controlled dataset environment. Real-world network traffic may introduce additional variability, noise, and previously unseen attack patterns [4, 15]. Therefore, while the current results are promising, further validation on real-time or live network data would be necessary to fully assess the deployment readiness of the system [7, 17].

In summary, the proposed approach demonstrates that combining ensemble learning techniques with explainability methods can lead to highly accurate and interpretable intrusion detection systems [9, 19]. The ability to justify predictions through SHAP not only improves transparency but also increases trust in the model, which is a critical requirement in cybersecurity applications [12, 19, 20].

4.7 Limitations

Despite the strong performance achieved by the proposed intrusion detection system, several limitations should be acknowledged.

First, the model was trained and evaluated on a preprocessed and labeled dataset. While this allows for controlled experimentation, it does not fully reflect the complexity of real-world network environments. In practice, network traffic is often

noisy, incomplete, and may contain previously unseen attack patterns. As highlighted in [4], machine learning models in intrusion detection may struggle when deployed outside the closed-world assumption.

Second, the dataset used in this study may exhibit class imbalance and redundancy, which can influence the learning process [13, 14]. Although techniques such as stratified splitting were applied, the presence of dominant classes (e.g., *BENIGN*) may still bias the model toward majority patterns [14].

Third, the current implementation focuses on offline analysis rather than real-time detection. The system processes static data samples and does not account for streaming data or temporal dependencies, which are essential in operational cybersecurity systems [10, 17].

Another limitation concerns the exclusion of certain attack categories, such as *WebAttack*, from the dataset during preprocessing. This decision was made to simplify the classification problem and improve model stability [14]. However, it limits the model's ability to detect web-based attacks, which are common in real-world scenarios [5, 15]. As a result, the current system is not fully comprehensive and may require further extension to cover a broader range of attack types [10].

Additionally, while SHAP provides valuable interpretability, it introduces computational overhead, particularly when explaining individual predictions [8, 12]. This may limit its scalability in real-time applications where rapid decision-making is required [19, 20].

Finally, the model relies on a fixed set of engineered features. Any changes in network protocols, traffic behavior, or attack strategies may reduce the effectiveness of these features over time, requiring periodic retraining and feature updates [7, 17].

These limitations highlight the need for further research and improvements before deploying the system in real-world scenarios [4, 10].

4.8 Conclusion

This chapter presented a comprehensive analysis of the results obtained from the proposed intrusion detection system. The evaluation demonstrated that the Random Forest model achieves high classification performance across multiple attack categories, with minimal misclassification and strong generalization capability [7, 9].

The use of cross-validation confirmed the performance and stability of the model, ensuring that the obtained results are not dependent on a specific data split [25]. In addition, feature importance analysis revealed that the model relies on meaningful network traffic characteristics, such as packet length and flow behavior, which are consistent with domain knowledge in cybersecurity [5, 23].

Furthermore, the integration of SHAP significantly enhanced the interpretability of the system [12, 19]. Both global and local explanations provided valuable insights into the model's decision-making process, allowing for a better understanding of how different features contribute to specific predictions [12, 18].

Despite these strengths, several limitations were identified, including the reliance on a static dataset, the exclusion of certain attack categories, and the absence of real-time processing capabilities [4, 10]. These aspects highlight opportunities for further improvement and future research.

Overall, the results confirm that the proposed approach is effective for multi-class intrusion detection and offers a balanced combination of accuracy and interpretability [7, 19], making it a promising solution for practical cybersecurity applications [12, 20].

Chapter 5

General Conclusion and Future Work

5.1 General Conclusion

This work addressed the problem of network intrusion detection by proposing an intelligent and interpretable system based on machine learning techniques. The main objective was to develop a solution capable of accurately classifying multiple types of network traffic while providing clear insights into the decision-making process of the model [12, 19].

To achieve this, a Random Forest classifier was employed due to its robustness and effectiveness in handling complex datasets [9]. In addition, SHAP was integrated as an explainability method to overcome the limitations of black-box models and enhance the transparency of the system [12, 20].

The experimental results demonstrated that the proposed approach achieves high classification performance across multiple attack categories, with strong generalization capability [7, 9]. The system was able to effectively distinguish between benign

traffic and various types of attacks, confirming its suitability for intrusion detection tasks.

Beyond performance, the integration of explainability played a crucial role in understanding the behavior of the model [12, 19]. By providing both global and local interpretations, the system offers valuable insights into how different features influence predictions [12, 18]. This significantly improves trust and usability, which are essential in cybersecurity applications [19, 20].

Furthermore, the development of an interactive dashboard enabled practical interaction with the model, allowing users to perform predictions and visualize explanations in a user-friendly environment. This bridges the gap between theoretical modeling and real-world application [19].

In summary, this work demonstrates that combining ensemble learning with explainability techniques can lead to effective and transparent intrusion detection systems [9, 19]. The proposed solution not only achieves high accuracy but also provides meaningful interpretations, making it a promising approach for modern cybersecurity challenges.

5.2 Contributions

This work presents several key contributions in the field of network intrusion detection, combining performance, interpretability, and practical usability:

- **Multi-class Intrusion Detection Model:** A classification model based on Random Forest was developed to detect and distinguish between multiple types of network attacks, rather than limiting the problem to binary classification [7, 9].

- **Integration of Explainable AI (SHAP):** SHAP was incorporated to provide both global and local interpretability, enabling a clear understanding of how individual features influence the model's predictions [12, 19].
- **Comprehensive Feature Analysis:** An in-depth analysis of feature importance was conducted, identifying the most relevant network traffic characteristics and linking them to known intrusion patterns [5, 9, 23].
- **Interactive Dashboard Implementation:** A user-friendly interface was developed using Streamlit, allowing users to perform predictions, upload data, and visualize model explanations in real time.
- **Combination of Performance and Interpretability:** The proposed system achieves high classification accuracy while maintaining transparency, addressing a key limitation of traditional black-box intrusion detection models [4, 19, 20].

These contributions collectively demonstrate the feasibility of building intrusion detection systems that are not only accurate but also interpretable and accessible for practical use [7, 12].

5.3 Future Work

While the proposed system demonstrates strong performance and interpretability, several extensions can be considered to further enhance its capabilities and practical impact.

First, future work may include the integration of additional attack categories, such as *WebAttack*, which were excluded during preprocessing. Incorporating these classes

would allow the system to cover a broader spectrum of real-world cyber threats and improve its comprehensiveness [10, 15].

Second, the system can be extended toward real-time intrusion detection by enabling the analysis of live network traffic [17]. Instead of relying on static datasets, the model could process streaming data captured directly from network interfaces, allowing for immediate detection of malicious activities as they occur [4, 7].

In this context, the proposed solution could be integrated with widely used network monitoring tools such as Wireshark, which provides packet-level traffic inspection [28]. By leveraging packet capture mechanisms (e.g., PCAP files or live traffic streams), the system could operate alongside such tools to deliver intelligent, real-time threat detection combined with explainable insights [17, 19].

Furthermore, future improvements may explore the use of advanced machine learning techniques, including deep learning models, to capture more complex patterns in network traffic [29]. In parallel, optimizing SHAP computations or exploring alternative explainability methods could enhance the scalability of the system in real-time environments [12, 20].

From an application perspective, these extensions open the possibility of transforming the proposed system into a deployable cybersecurity solution. A real-time, explainable intrusion detection platform capable of integrating with existing monitoring tools could provide significant value for enterprise networks, security operations centers (SOCs), and system administrators [10, 19].

Overall, these future directions highlight the potential of evolving the current academic prototype into a practical and economically valuable cybersecurity solution [7, 19].

References

[1] W. Stallings, "Network security essentials: Applications and standards," *Pearson Education*, 2017.

[2] National Institute of Standards and Technology (NIST), "Guide to intrusion detection and prevention systems (idps)," 2012.

[3] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on Software Engineering*, vol. 13, no. 2, pp. 222–232, 1987.

[4] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," *IEEE Symposium on Security and Privacy*, pp. 305–316, 2010.

[5] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Macia-Fernandez, and E. Vazquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Computers & Security*, vol. 28, no. 1-2, pp. 18–28, 2009.

[6] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, 2019.

[7] G. Kim *et al.*, "A survey of intrusion detection systems based on machine learning and deep learning," *IEEE Access*, vol. 7, pp. 41 589–41 605, 2019.

[8] C. Molnar, "Interpretable machine learning," *Lulu.com*, 2020.

[9] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[10] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[11] J. Mirkovic and P. Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.

[12] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[13] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[14] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[15] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer Networks*, vol. 51, no. 12, pp. 3448–3470, 2007.

[16] K. Kent and M. Souppaya, "Guide to computer security log management," *National Institute of Standards and Technology (NIST)*, 2006.

[17] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[18] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?: Explaining the predictions of any classifier," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144, 2016.

[19] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado *et al.*, "Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Information Fusion*, vol. 58, pp. 82–115, 2020.

[20] Z. C. Lipton, "The mythos of model interpretability," *Queue*, vol. 16, no. 3, pp. 31–57, 2018.

[21] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," in *arXiv preprint arXiv:1702.08608*, 2017.

[22] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[23] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *Proceedings of the International Conference on Information Systems Security and Privacy*, pp. 108–116, 2018.

[24] M. Fernandez-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *Journal of Machine Learning Research*, vol. 15, pp. 3133–3181, 2014.

[25] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 14, 1995, pp. 1137–1145.

[26] D. M. W. Powers, "Evaluation: From precision, recall and f-measure to roc, informedness, markedness and correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

[27] S. Inc., "Streamlit: The fastest way to build and share data apps," https://streamlit.io, 2023, accessed: 2024.

[28] Wireshark Foundation, "Wireshark: The world's foremost network protocol analyzer," https://www.wireshark.org, 2023, accessed: 2024.

[29] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing*, vol. 22, pp. 949–961, 2019.