



**Department of Electrical and Computer Engineering
North South University**

Senior Design Project

**An Explainable Approach for Network Intrusion
Detection Using Machine Learning and Deep Neural
Network**

Sajan Kumer Sarker

ID# 2111131642

Mahzabeen Rahman Meem

ID# 2021300642

Arifa Akbar Srity

ID# 2021149642

Faculty Advisor:

Dr. Mohammad Ashrafuzzaman Khan (AZK)

Associate Professor

ECE Department

Spring, 2025

LETTER OF TRANSMITTAL

April, 2025

To

Dr. Mohammad Abdul Matin
Chairman,
Department of Electrical and Computer Engineering
North South University, Dhaka

Subject: Submission of Capstone Project Report on “An Explainable Approach for Network Intrusion Detection Using Machine Learning and Deep Neural Network”.

Dear Sir,

With due respect, we would like to submit our **Capstone Project Report on “An Explainable Approach for Network Intrusion Detection Using Machine Learning and Deep Neural Network”** as a part of our BSc program. The report deals with Network Security System. This project was very much valuable to us as it helped us to gain practical experience in the field and apply our knowledge in real-life applications. We tried to the maximum competence to meet all the dimensions required from this report.

We will be highly obliged if you kindly receive this report and provide your valuable judgment. It would be our immense pleasure if you find this report useful and informative to have an apparent perspective on the issue.

Sincerely Yours,

.....
Sajan Kumer Sarker
ECE Department
North South University, Bangladesh

.....
Mahzabeen Rahman Meem
ECE Department
North South University, Bangladesh

.....
Arifa Akbar Srity
ECE Department
North South University, Bangladesh

APPROVAL

Sajan Kumer Sarker (ID # 2111131642), Mahzabeen Rahman Meem (ID # 2021300642) and Arifa Akbar Srity (ID # 2021149642) from Electrical and Computer Engineering Department of North South University, have worked on the Senior Design Project titled “An Explainable Approach for Network Intrusion Detection System Using Machine Learning and Deep Neural Network” under the supervision of Dr. Mohammad Ashrafuzzaman Khan partial fulfillment of the requirement for the degree of Bachelors of Science in Engineering and has been accepted as satisfactory.

Supervisor’s Signature

.....

Dr. Mohammad Ashrafuzzaman Khan

Associate Professor

Department of Electrical and Computer Engineering

North South University

Dhaka, Bangladesh.

Chairman’s Signature

.....

Dr. Mohammad Abdul Matin

Professor

Department of Electrical and Computer Engineering

North South University

Dhaka, Bangladesh.

DECLARATION

This is to declare that this project is our original work. No part of this work has been submitted elsewhere partially or fully for the award of any other degree or diploma. All project related information will remain confidential and shall not be disclosed without the formal consent of the project supervisor. Relevant previous works presented in this report have been properly acknowledged and cited. The plagiarism policy, as stated by the supervisor, has been maintained.

Students' names & Signatures

1. Sajan Kumer Sarker

2. Mahzabeen Rahman Meem

2. Arifa Akbar Srity

ACKNOWLEDGEMENTS

The authors would like to express their heartfelt gratitude towards their project and research supervisor, Dr. Mohammad Ashrafuzzaman Khan, Associate Professor, Department of Electrical and Computer Engineering, North South University, Bangladesh, for his invaluable support, precise guidance and advice pertaining to the experiments, research and theoretical studies carried out during the course of the current project and also in the preparation of the current report.

Furthermore, the authors would like to thank the Department of Electrical and Computer Engineering, North South University, Bangladesh for facilitating the research. The authors would also like to thank their loved ones for their countless sacrifices and continual support.

ABSTRACT

An Explainable Approach for Network Intrusion Detection Using Machine Learning and Deep Neural Networks

Web applications have become essential components in various sectors, including e-commerce, banking, and healthcare, making their security a crucial priority. Cyberattacks increasingly target these applications to exploit vulnerabilities and steal users sensitive data. A robust network intrusion detection tool is essential to detect such threats. Many studies have proposed machine learning and deep learning-based approaches to build effective Intrusion Detection System (IDS) solutions. A common challenge in these studies is the imbalance in datasets, which researchers often address by generating synthetic data to improve training quality. However, how much synthetic data can be effectively used to train accurate and reliable security models remains unclear. This research investigates the impact of different proportions of synthetic data on model performance. We used the widely recognized dataset CIC-IDS2017, applied the data balancing strategy using SMOTE, and produced three different versions of the dataset—small, medium, and large—to evaluate the models performance under different levels of data complexities and sizes. Additionally, we applied XGBoost algorithm for feature selection purpose during the preprocessing phase. Our approach was evaluated using various machine learning algorithms, including Decision Tree, Random Forest, Naïve Bayes, Ada Boost, and Deep Neural Networks. Based on performance metrics, we developed an ensemble model that combines the two best-performing classifiers. To enhance transparency, we incorporated Explainable AI (XAI) techniques to make the models predictions more interpretable and transparent. The result demonstrates that the ensemble model outperformed on the medium dataset, achieving an average accuracy of 97.20%, correctly classifying types of attacks with 97.30% accuracy, and producing a low false positive rate of 0.0048.

TABLE OF CONTENTS

LETTER OF TRANSMITTAL	2
APPROVAL	4
DECLARATION	5
ACKNOWLEDGEMENTS	6
ABSTRACT.....	7
LIST OF FIGURES	11
LIST OF TABLES	12
Chapter 1 Introduction	13
1.1 Background and Motivation:	13
1.2 Purpose and Goal of the Project:	16
1.3 Organization of the Report:	16
Chapter 2 Research Literature Review	17
2.1 Existing Research and Limitations:.....	17
Chapter 3 Methodology	20
3.1 System Design:.....	20
3.1.1 Dataset Collection & Preprocessing:	21
3.1.2 Feature Selection with XGBoost:	25
3.1.3 Model Building, Training, and Evaluation	27
3.1.4 Ensemble Model	28
3.1.4 Explainable AI (XAI) for Model Interpretation.....	29
3.1.5 Real World Testing:.....	30
3.2 Software Components	31
3.3 Software Implementation	32
Chapter 4 Investigation/Experiment, Result, Analysis and Discussion.....	33

4.1	Performance Analysis on Small Dataset.....	33
4.2	Performance Analysis on Medium Dataset.....	34
4.3	Performance Analysis on Large Dataset.....	34
4.4	SHAP Plot Analysis on Medium Dataset with Ensemble Model	35
4.5	LIME Plot Analysis on Medium Dataset with Ensemble Model	36
4.6	Real-World Testing.....	37
4.7	Discussion	38
Chapter 5 Impacts of the Project.....		40
5.1	Impact of this project on Social, Health, Safety, Legal and Cultural Issues:	40
5.1.1	Social Impact:	40
5.1.2	Health Impact:.....	40
5.1.3	Safety Impact:	40
5.1.4	Legal Impact:	41
5.1.5	Cultural Impact:	41
5.2	Impact of this project on Environment and Sustainability:.....	41
5.2.1	Environmental Impacts:	41
5.2.2	Sustainability Impacts:.....	42
Chapter 6 Project Planning and Budget		43
Chapter 7 Complex Engineering Problems and Activities		44
7.1	Complex Engineering Problems (CEP):	44
7.2	Complex Engineering Activities (CEA):	45
Chapter 8 Conclusions		47
8.1	Summary:	47
8.2	Limitations:	47
8.3	Future Improvement:.....	48

References	49
------------------	----

LIST OF FIGURES

Figure 1. Network Attack Detection and Explanation System Design Process	20
Figure 2. Class Distribution of The Original Dataset	21
Figure 3. Correlation Matrix Between Numerical Features	22
Figure 4. Multiclass Distribution of Small, Medium, and Large Dataset	24
Figure 5. Feature Importance of XGBoost Algorithm	26
Figure 6. Packet Capturing Using Wireshark Tool	30
Figure 7. System Workflow Process for Network Activity Classification	32
Figure 8. SHAP Summary Plot for Multiclass Result	435
Figure 9. LIME Explanation on Benign Class Prediction	437
Figure 10. LIME Explanation on Benign Class Prediction	437
Figure 11. Real-world Network Activity Testing	438
Figure 12. Project Timeline Gantt chart	43

LIST OF TABLES

TABLE 1. LABEL ENCODING PROCESS OF SMALL DATASET	23
TABLE 2. LABEL ENCODING PROCESS OF MEDIUM AND LARGE DATASET	23
TABLE 3. BINARY LABEL ENCODING PROCESS	23
TABLE 4. SELECTED FEATURES BASED ON FEATURE IMPORTANCE SCORES	26
TABLE 5. LIST OF SOFTWARE TOOLS & TECHNOLOGIES TABLE	31
TABLE 6. CLASSIFICATION RESULTS OF SMALL DATASET	33
TABLE 7. CLASSIFICATION RESULTS OF MEDIUM DATASET	34
TABLE 8. CLASSIFICATION RESULTS OF LARGE DATASET	34
TABLE 9. ESTIMATED BUDGET TABEL	43
TABLE 10 A SAMPLE COMPLEX ENGINEERING PROBLEM ATTRIBUTES TABLE	44
TABLE 11. A SAMPLE COMPLEX ENGINEERING PROBLEM ACTIVITIES TABLE	45

Chapter 1 Introduction

1.1 Background and Motivation

In today's digital age, web applications are integral to various activities. Worldwide, there are over a billion internet users and websites at present. A big reason for the success of the internet is the simplicity and the fact that you can access the applications from anywhere. Many organizations and individuals make their websites and provide online services. With the increasing rate of this website, the risk of cyberattacks is also increasing. So, the security of web applications has become increasingly important, and a secure web environment has become a high priority for any e-business community. Cyberattacks more frequently target websites like e-commerce, banking and financial, healthcare, and government due to the value of their data, the nature of their operation, or their popularity. Cyberattacks, such as various versions of DoS and DDoS, Port Scanning, Botnet, Heartbleed, SQL injection, cross-site scripting, path traversal, and command injection, are considered the most common and dangerous threats to websites, web applications, and web users [1]. These attacks mostly require only basic knowledge of vulnerabilities and access to readily available tools. A lack of security awareness among developers and organizations often leads to unpatched vulnerabilities and poor security practices, increasing susceptibility. These attacks can assist attackers to in bypassing web systems authentication mechanisms, carrying out unauthorized modifications to web content and databases, to extract important data from web application databases, and stealing sensitive information from web servers.

Cybersecurity is becoming increasingly critical as the world moves more towards digitalization. An intrusion detection system (IDS) is used to identify unauthorized activities, attacks, or anomalies in network traffic, one of cybersecurity's most important parts. In this study, we represent a detailed process of data exploration, preprocessing, model development, and evaluation process. The study involves analyzing a vast dataset containing real-world network traffic data and applying supervised machine-learning techniques to distinguish between normal activities and various types of cyberattacks. This model detects known threats and aims to generalize well for unknown attacks, thus strengthening the defense mechanism against cyber intrusions.

An intrusion detection system (IDS) is a cybersecurity tool that monitors and analyzes network traffic for suspicious behavior or potential threats. It generates alerts when abnormal patterns or known malicious activities are detected [2]. IDS can generally be categorized into two types: signature-based and anomaly-based. Signature-based IDS relies on pre-defined patterns of known attacks to identify threats. In contrast, anomaly-based IDS identifies unusual behaviors that deviate from the normal profile using statistical or machine learning methods [3]. By allowing models to learn from past data and automatically recognize intricate attack patterns without being explicitly coded for every potential danger, machine learning has been employed more and more to increase the efficacy of intrusion detection systems. Adopting machine learning in intrusion detection systems has led to smarter, faster, and more adaptive security systems capable of handling sophisticated cyberattacks. Furthermore, the role of Explainable AI (XAI) is to provide transparency by helping the administrator understand the logic behind the models decisions. This combination not only increases trust and interpretability but also helps reduce false alarms by offering clear justifications for each alert raised.

We utilized the CIC-IDS2017 dataset developed by the Canadian Institute for Cybersecurity. The CIC-IDS2017 dataset is one of the most comprehensive publicly available resources for network intrusion detection research. The dataset offers a wide variety of network traffic scenarios captured over several days, which includes both benign traffic and a range of modern malicious traffic, such as DDoS, brute force, infiltration, web attacks, botnets, and others. This dataset was designed for intrusion detection research, which simulates real-world network traffic utilizing authentic network activity along with well-documented cyberattacks. The large size and diversity of the dataset ensure that models trained on this dataset are exposed to a wide range of network behavior and attack fingerprints.

Web attacks primarily target vulnerabilities within web servers, applications, or frameworks. An injection attack involves inserting malicious queries into the input field and taking advantage of inadequate validation, which can grant attackers complete access to databases, cross-site scripting (XSS) attacks can inject malicious scripts into otherwise trusted websites, leading to session hijacking, defacement, or redirection. Web attacks often manipulate HTTP requests and responses, and they can be detected by analyzing attributes such as request size, number of login attempts, unusual URL patterns, and sudden changes in session behavior. Machine learning models

detect web attacks by identifying these abnormalities against the baseline of normal web traffic patterns.

These attacks were simulated and recorded using Wireshark [4], a real-time packet analysis tool, and processed with CIC Flow Meter, which extracts over 80 statistical and flow-based features from the raw data [5]. These processed features serve as input for ML algorithms to detect and classify traffic patterns. The classification models in this system label traffic into multiple categories such as ‘Benign’, ‘DDoS’, or ‘Botnet’, making it a multiclass classification problem. In some use cases, the classification is simplified to a binary format: ‘benign’ vs. ‘malicious’. The models are trained using labeled data and then tested on unseen samples to predict and classify real-time network traffic.

Due to an imbalance in class distribution—where benign traffic may significantly outnumber malicious examples—the dataset is augmented using SMOTE (Synthetic Minority Over-sampling Technique) [6]. This method synthetically generates new samples for underrepresented classes to balance the training data and improve model generalization. To evaluate the system under different conditions, we test performance across three dataset sizes: Small, Medium, and Large. For feature selection, XGBoost (Extreme Gradient Boosting) is utilized due to its ability to rank feature importance during the training process. It is efficient and scalable and helps remove irrelevant features, which improves both performance and training speed.

An ensemble model was proposed in this work, which is a machine learning technique [7] that combines predictions of multiple models to improve overall performance. This approach helps to reduce biases of individual model and leverage their complementary strengths, which results in better accuracy, more stable prediction, and improved generalization, especially in common classification tasks involving both binary and multiclass. Explainable AI (XAI) methods like SHAP and LIME are implemented [8]. SHAP (SHapley Additive exPlanations) provides both important global and local features that are important by borrowing ideas from cooperative game theory. LIME (Local Interpretable Model-agnostic Explanations) approximates the decision boundary of complex models using simple interpretable models at the local level. Leveraging the XAI techniques will not only help security analysts interpret why a model detection was made but

also provide into potential misclassification. If benign traffic is mistakenly flagged as malicious, then LIME can reveal which features led to this result, aiding in debugging and refining the model.

1.2 Purpose and Goal of the Project

This research aims to explore the impact of synthetic data on model performance by generating three versions of the dataset using SMOTE. Each version contains varying degrees of synthetic samples in the minority classes. We also apply XGBoost for feature selection to enhance model efficiency. We consider the small dataset to be our primary dataset for feature selection purposes. A range of classifiers are trained and evaluated, including Decision Tree, Random Forest, Naïve Bayes, AdaBoost, and Deep Neural Networks. Furthermore, we propose an ensemble model combining the two best-performing classifiers based on evaluation metrics. To promote model transparency and trust, we incorporate Explainable AI (XAI) techniques LIME to interpret the predictions. Though this approach, we aim to provide insights into how synthetic data affects model accuracy and generalizability in intrusion detection while offering a practical and interpretable solution for real-world deployment.

1.3 Organization of the Report

- Chapter 1 represents Introduction.
- Chapter 2 represents Research Literature Review.
- Chapter 3 represents Methodology.
- Chapter 4 represents Experimental Results, Analysis, and Discussion.
- Chapter 5 represents Social and Environmental impact.
- Chapter 6 represents Project Planning and Budget.
- Chapter 7 represents Complex Engineering Problems and Activities.
- Chapter 8 represents Conclusion.

Chapter 2 Research Literature Review

2.1 Existing Research and Limitations

A novel intrusion detection presented by Callegari et al. [9] that is capable of processing raw network traffic in real-time using deep learning techniques. Performed feature extraction directly from packet-level traces using efficient probabilistic data structure. Different types of deep learning networks—MLP, CNN, RNN, LSTM, and GRU—are evaluated and compared based on detection accuracy, false alarm rate, and execution time. The CNN model achieved detection rate of 92.6% and an accuracy of 89.9% in binary classification, and around 86% for multiclass detection to identify specific attack types. The author highlights the importance of training on realistic, well-balanced datasets to avoid overly optimistic results.

Ahmed et al. [10] proposed a novel intrusion detection model HAEnID (Hybrid Adaptive Ensemble for Intrusion Detection), which integrates three ensemble methods—Stacking Ensemble (SEM), Bayesian Model Averaging (BMA), and Conditional Ensemble Method (CEM)—to enhance detection accuracy, reduce false positives, and adapt to evolving cyber threats. The model was enhanced with explainable AI techniques SHAP and LIME to make the decision transparent and interpretable. The Wednesday Attack Record subset of the CIC-IDS2017 dataset contains various types of DoS, and safe records used in this work that. The Bayesian Model Averaging (BMA) approach achieved the best 98% of accuracy on binary classification and 97.94% accuracy for multi-class classifications.

In a work of et Sajid al. [11] present a hybrid intrusion detection system that integrates XGBoost, CNN, and LSTM to enhance cybersecurity threat detection. It focuses on improving feature selection and classification using several benchmark datasets such as: CIC-IDS2017, UNSW-NB15, NSL-KDD, and WSN-DS. They combine the XGBoost and CNN for feature extraction and utilize LSTM for sequential learning, leading to better detection accuracy and reducing false positives. The results demonstrate high detection rates, with CNN-LSTM achieving 98.55% accuracy on CIC-IDS2017. The approach demonstrates robustness, scalability, and the ability to detect both known and unknown threats across diverse network environments.

A signature-based intrusion detection system was presented by Ahmed et al. [12] that integrates machine learning, deep learning, and fuzzy clustering techniques to enhance network security. They used the UNSW-NB15 dataset for binary classification of normal and malicious traffic. They integrate the data fusion, adaptive modeling, and signature-based detection with intelligent techniques. The system focuses on both knowns and unknown threats. Fuzzy clustering was applied to improve data grouping and pattern recognition. They emphasize the need for scalable, real-time IDS solutions capable of adapting of dynamic nature of cyber threats.

Talukder et al. [13] address the key challenges in IDS, including class imbalance, high dimensionality, and detection rate. Their method uses SMOTE for data balancing and XGBoost for feature selection to improve classification performance. The approach was tested on KDDCUP99 and CIC-MalMem-2022 datasets, achieving accuracy rates up to 99.99% with minimal false positives and false negatives. The model includes a pipeline of preprocessing, feature ranking, and classification using various machine learning and deep learning models.

Nirvikar et al. [14] explored how Artificial Intelligence and Machine Learning Technique can improve cybersecurity, concentrating on thread identification and response. The methodology covered various machine learning technique such as anomaly detection, malware classification, and network intrusion detection. They used real-world cybersecurity datasets, including network traffic logs and malware samples. They provide an overview of AI and ML applications in cybersecurity, including key techniques, case studies, and an examination of limitations and challenges, with notable issues being the need for large labeled datasets and the interpretability of black-box models.

Saleem et al. [15] proposes a machine learning-based intrusion detection model that analyzes web server logs to identify and classify these attacks. This technique improves the identification of unidentified attacks. Here the author utilizes two machine learning models- simple feature extraction and text-based classification, on a dataset generated using private network. The text- based model, which uses a larger set of features, outperforms the simple model in accuracy. The model has lacking on capability to handle multiclass attack detection effectively.

An unsupervised anomaly detection framework for intrusion detection system was proposed by Richa et al. [16]. The CIC-IDS2017 dataset was used to test the Performance of the

framework. They integrate One Class Support Vector Machine (OCSVM) to detect threats without prior knowledge. The proposed framework was capable of detecting various attack types, including known, unknown, and zero-day attacks.

In research of Alaoui et al. [17] explored the potential of deep learning to enhance web application security, specifically in vulnerability and attack detection through a systematic literature review of existing research. This research highlights the need for standardized real-work web attack datasets, the potential of sophisticated deep learning models like generative adversarial networks, and the importance of collaborating between machine learning and web security specialists. It highlights the lack of high-quality attack datasets and the SLR's incapacity to offer in-depth analyses of specific studies.

Ghanem et al. [18] explores using SVM techniques for enhancing network intrusion and cyber-attack detection systems. That aims to improve the accuracy of intrusion detection systems by implementing SVM and comparing its performance with other techniques. The methodology includes anomaly-based IDM and SVM with KDD9 dataset. The study acknowledges limitations including the KDD99 dataset's potential lack of real-world relevance and the high computational cost of using SVM with large datasets.

In a research Liang et al. [19] proposed a Deep Learning approached Anomaly-Based Web Attack Detection focus on Cloud Technology and Web Attacks. There are some limitations of Current Approaches: Signature-based methods, and Rule-based methods. The model has few advantages, such as, no feature selection is needed, making it adaptable and easy to apply, and the model was accurate in identifying web-based attacks and outperforms traditional Web Application Firewalls (WAFs) like MoD Security, especially when it comes to dynamic attacks and anomaly detection. It achieves 98% accuracy with high sensitivity of 97.56% and specificity of 99.21% on the CSIC dataset. The authors describe how well LSTM/GRU RNNs work in conjunction with neural network classifiers and offer possible uses for this combination in other domains where anomaly detection is needed.

Chapter 3 Methodology

3.1 System Design

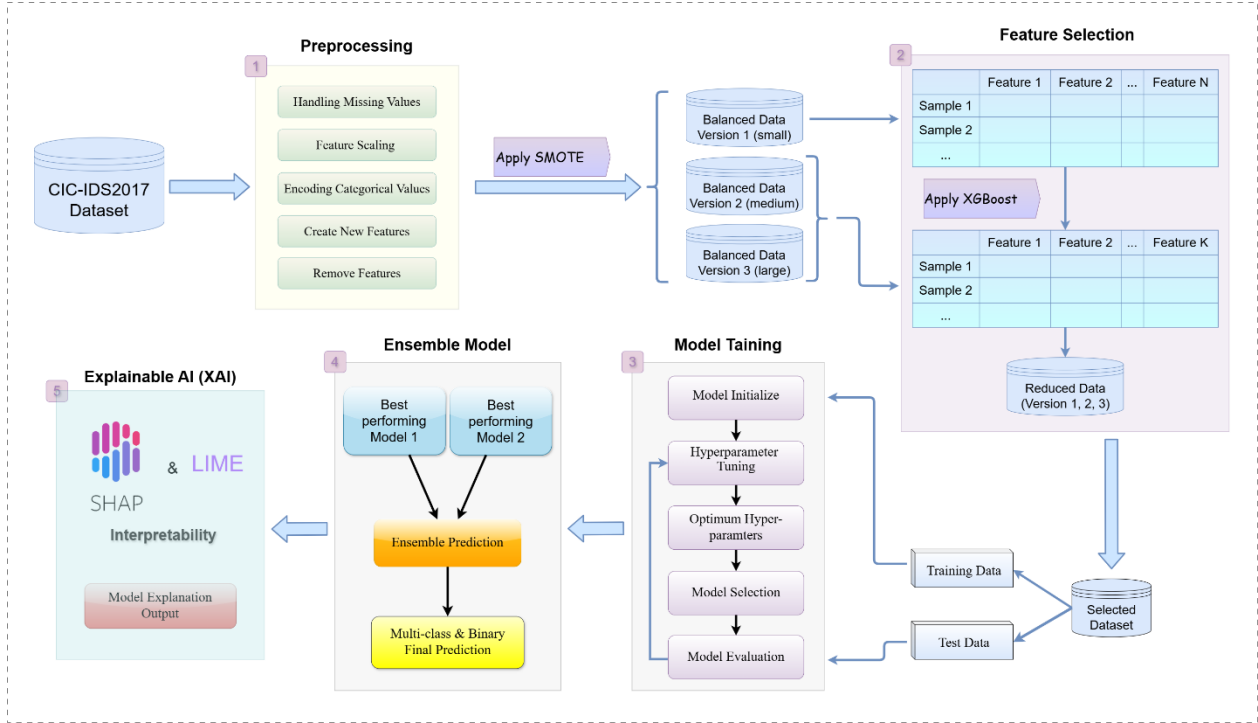


Figure 1: Network Attack Detection and Explanation System Design Process.

The study aims to investigate the impact of different proportions of synthetic data on model performance. We proposed an explainable approach to develop a Network Intrusion Detection System (IDS) framework that can analyze network traffic data to detect and classify various web attacks using machine learning and deep neural networks. To improve the accuracy, we balance the dataset using SMOTE, produced three different versions of the dataset—small, medium, and large—and select important features with the XGBoost algorithm. Additionally, we integrated Explainable AI (XAI) techniques to enhance the interpretability of the model prediction. For real-world testing, we captured network activity traffic using the Wireshark tool, which recorded raw network traffic from our device. These data were converted into flow-based features using the CIC Flow Meter tool for use in our model. The primary objective is to evaluate the influence of synthetic data on performance and enhance both the accuracy and transparency of intrusion detection in modern network environments.

3.1.1 Dataset Collection & Preprocessing:

We used the CIC-IDS2017 comprehensive dataset [20], that is widely-used dataset for intrusion detection system (IDS) research. It contains network traffic data, that captured in real-world conditions, including various malicious traffic data and normal traffic data. However, the dataset was imbalanced as shown in Figure 2, with the majority class being safe traffic data, and minority class belong to the malicious traffic.

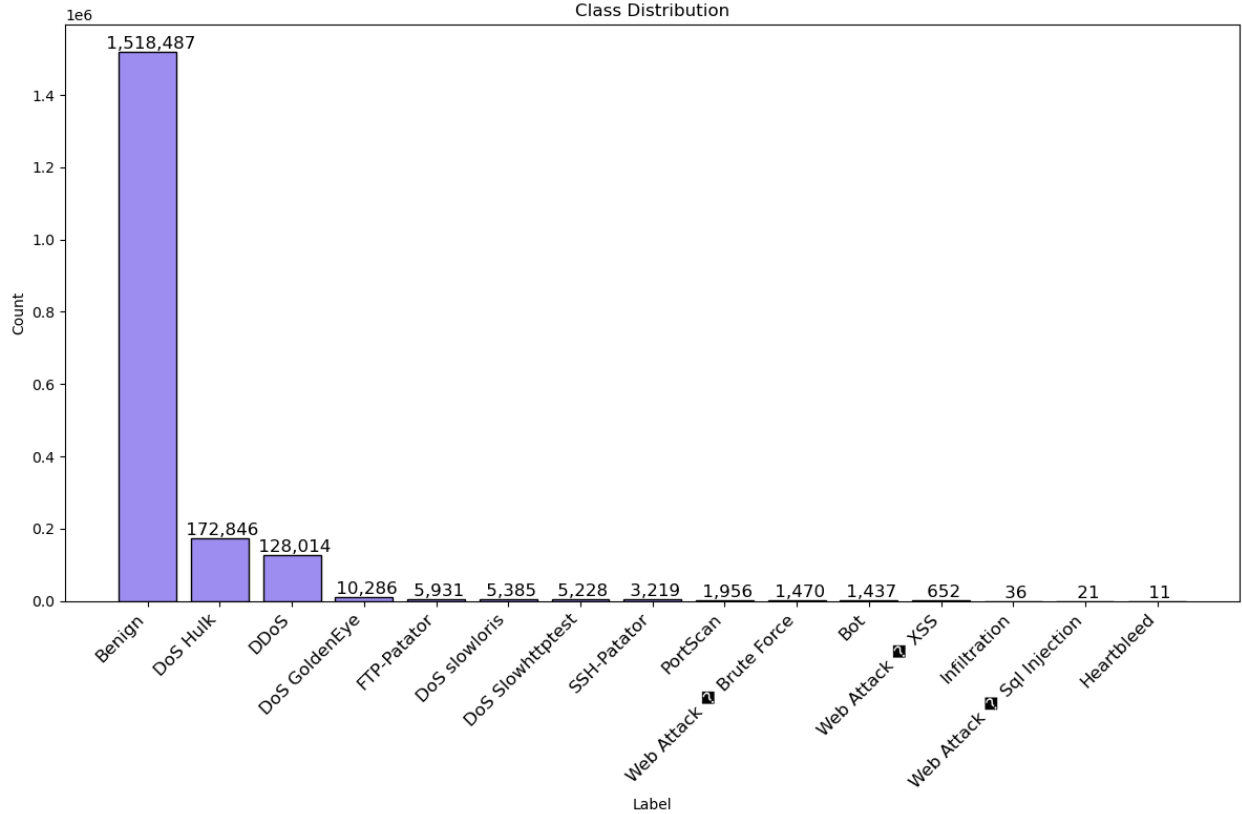


Figure 2: Class distribution of the original dataset.

We used data balancing strategies to address this issue and produced three dataset versions—small, medium, and large. The original dataset contains a total of 1,854,979, whereas the new versions are small - 75,000, medium – 1,522,380, and large containing 2,602,112 total samples. These include different types of network attack traffic such as Safe traffic, DoS Hulk, Dos Golden Ey, DoS Slowloris, DoS Slowhttptest, SSH-Patator, DDoS, Brute Force Attacks, Web Attacks XSS, Web Attacks SQLi, Port Scans, Botnet, Infiltration, and Heartbleed. To process these data for our model training, we have followed several steps:

Preprocessing Steps:

- **Data Cleaning:** Handling missing values, removing duplicate unwanted samples, and eliminating identical or low-variance features that do not meaningfully contribute to model learning.
- **Correlation Analysis:** Analyze the feature correlation to determine how strongly two features are related. The correlation matrix Figure 3 shows the pairwise correlations between all the numerical features in the dataset. The correlation matrix table helps to understand the coefficients between all pairs of features.

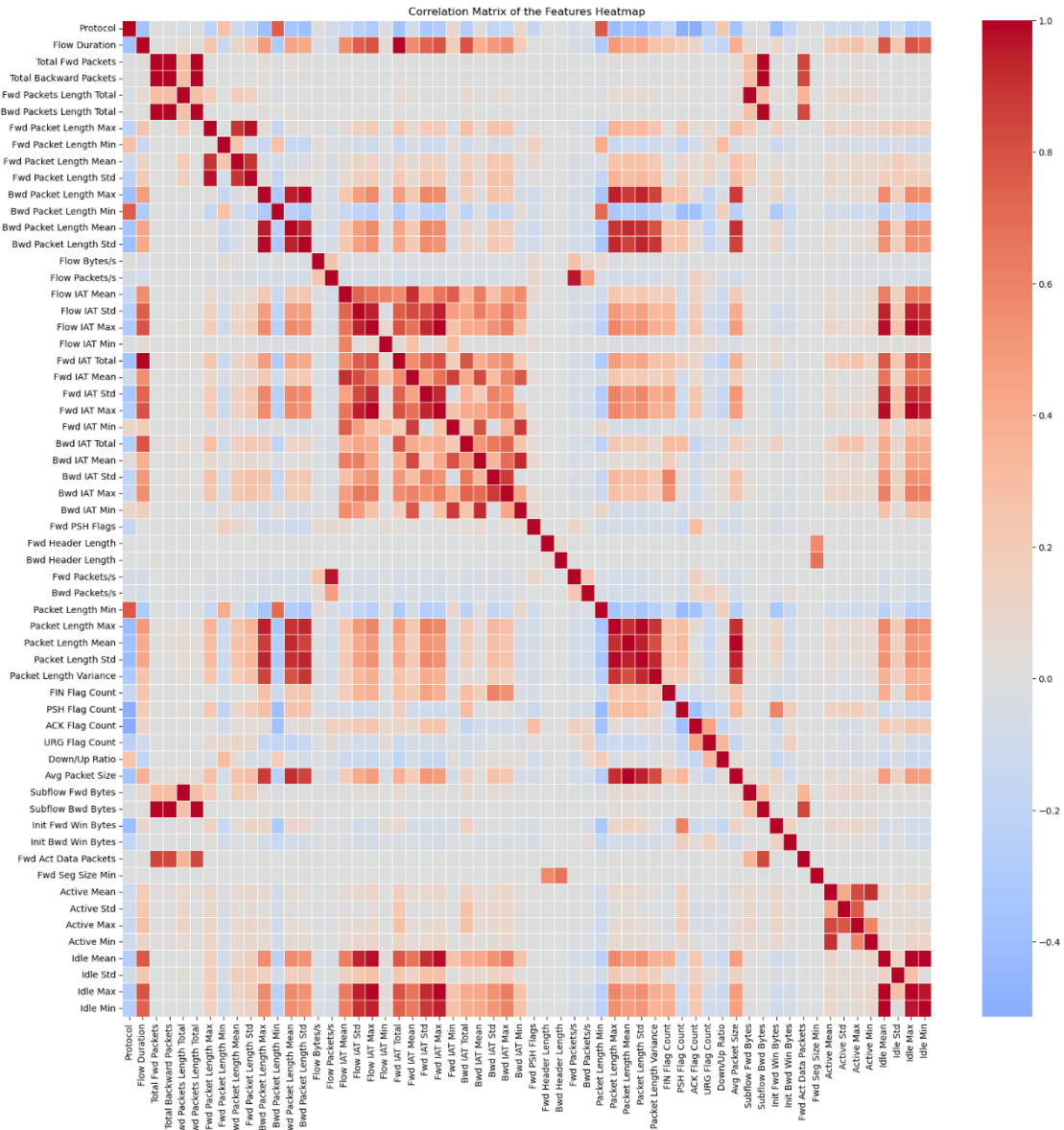


Figure 3: Correlation Matrix Between Numerical Features

- **Data Normalization:** We normalize numerical data to ensure uniform scaling across all features, which is especially crucial for our different range of features. A commonly used technique, **Standard Scaler**, was used (Equation 1) to transform the data so that each feature has a common mean of 0 and a standard deviation of 1.

$$z = \frac{x - u}{s} \quad (1)$$

This transformation ensures all features are on the same scale, making the learning process more stable and practical and often improving model performance and convergence speed.

- **Categorical Encoding:** The multiclass categorical values were encoded into numeric formats to make them suitable for machine learning models. Table 1 shows the class mapping process of the small version of the dataset, and for the medium and large versions of the datasets, the class mapping process is shown in Table 2.

TABLE 1: LABEL ENCODING PROCESS OF SMALL DATASET

Class Types	Label Encoding
Benign	0
Web Attack	1
Brute Force	2
DDoS	3
Dos	4

TABLE 2: LABEL ENCODING PROCESS OF MEDIUM AND LARGE DATASET

Class Types	Label Encoding
Benign	0
Botnet	1
Brute Force	2
DDoS	3
Dos	4
Port Scanning	5
Web Attack	6

- **Target Feature Creation:** Introduced a new target feature for multi-label classification, representing the network traffic status as either 0 (safe) or 1 (malicious), as shown in Table 3.

TABLE 3: BINARY LABEL ENCODING PROCESS

Class Types	Label Encoding
Benign	0
Malicious	1

- **SMOTE (Synthetic Minority Over-sampling Technique):** Apply the SMOTE techniques to balance the class distribution using both oversampling and undersampling methods. Using these techniques, we generated three different versions of the dataset with varying levels of data complexity and size.

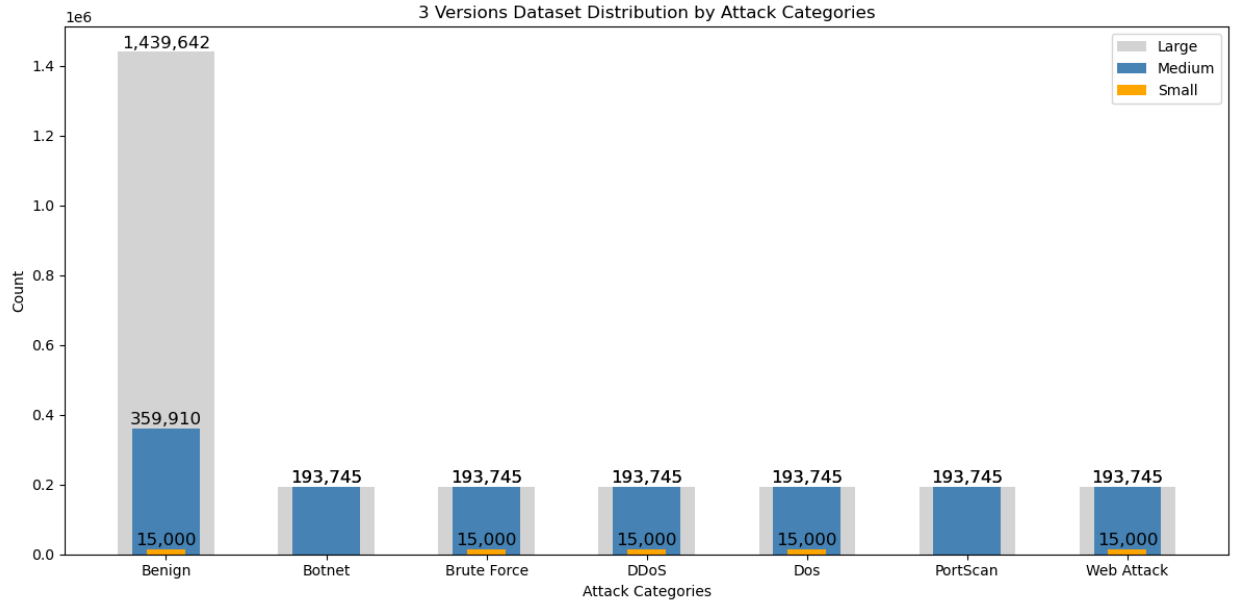


Figure 4: Multiclass Distribution of Small, Medium, and Large Dataset

Small (Version 1): This version contains 75,000 samples, with 15,000 samples for each class in the multi-class target. This dataset was treated as the primary dataset, for feature selection and model hyperparameters. During processing, we removed attack categories that contained fewer than 5,000 samples, as they were considered insufficient for effective learning. The remaining categories were balanced using SMOTE to ensure each class had equal samples. Similar attack types were grouped into broader categories to simplify the classification task and improve model generalization, as shown in Figure 4; the orange bar represents the class distribution of the small dataset. That mapping helped in reduced the complexity of the classification task while maintaining the diversity of attack types.

Medium (Version 2): This version contains 1,522,380 samples, with the benign class comprising into 25%, 359,910 samples, and each malicious class initially having approximately 193,745. To achieve a balanced dataset, the benign class was undersampled

to 25% of the total samples, while each malicious class was oversampled to match the size of the largest malicious category. As with the small version, attack types were grouped into broader categories to simplify the classification task and improve the models generalization capabilities. In Figure 4, the blue bars represent the class distribution of the medium dataset.

Large (Version 3): This dataset includes the entire set of benign samples, preserved without any modification to maintain the original distribution of benign network traffic after preprocessing. The malicious classes were balanced using oversampling techniques to match the size of the largest malicious category. Similar to the Medium version, the attack types were grouped into broader categories to simplify the classification task. The final class mapping used in this version is consistent with the one applied in the medium version, where the silver bar represents the class distribution of the large dataset in Figure 4.

3.1.2 Feature Selection with XGBoost

XGBoost, a robust gradient-boosting algorithm, is utilized for the feature selection. The XGBoost model is trained on the small version of dataset to evaluate the importance of each feature in distinguishing between safe traffic and malicious network traffic. These feature importance scores provided by the XGBoost indicate the contribution of each feature to the models decision-making process. In particular, it reflects the gain, which measures the improvement in accuracy brought by a feature to the decision tree branches it appears on—a higher gain implies higher importance. Additionally, the frequency with which the feature is used across trees further emphasizes its relevance. This technique helped identify the most influential features, enabling us to simplify the model as shown in Figure 5, which improve performance, and reduce the risk of overfitting. A threshold of 0.0030 was applied to select the top features on importance scores. Out of a total of 60 features in the processed dataset, the top 28 features were selected for model training. The Table 4 represents the selected features for our experiment.

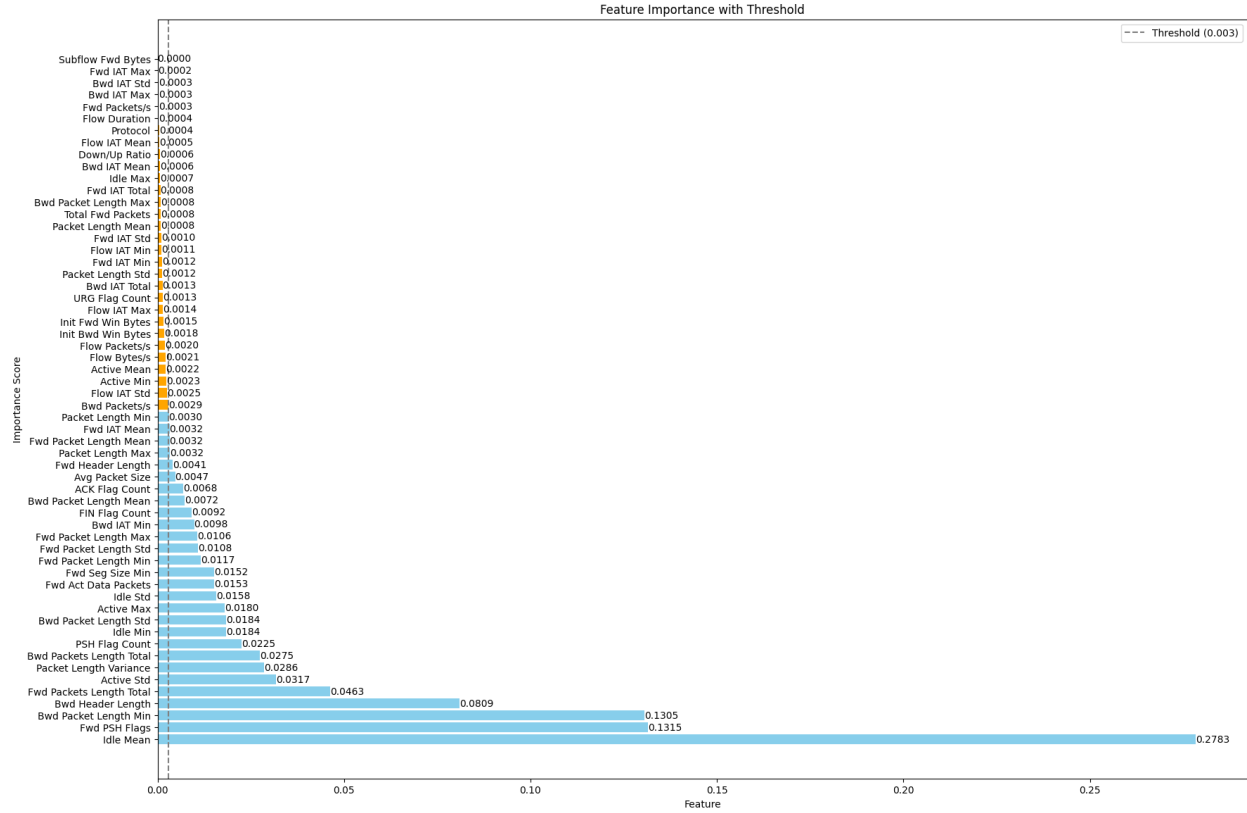


Figure 5: Feature Importance of XGBoost Algorithm

TABLE 4: SELECTED FEATURES BASED ON FEATURE IMPORTANCE SCORES

Sl. No.	Feature	Type	Sl. No.	Feature	Type
1	Idle Mean	float32	15	Fwd Seg Size Min	float32
2	Fwd PSH Flags	float32	16	Fwd Packet Length Min	float32
3	Bwd Packet Length Min	float32	17	Fwd Packet Length Std	float32
4	Bwd Header Length	float32	18	Fwd Packet Length Max	float32
5	Fwd Packets Length Total	float32	19	Bwd IAT Min	float32
6	Active Std	float32	20	FIN Flag Count	float32
7	Packet Length Variance	float32	21	Bwd Packet Length Mean	float32
8	Bwd Packets Length Total	float32	22	ACK Flag Count	float32
9	PSH Flag Count	float32	23	Avg Packet Size	float32
10	Idle Min	float32	24	Fwd Header Length	float32
11	Bwd Packet Length Std	float32	25	Packet Length Max	float32
12	Active Max	float32	26	Fwd Packet Length Mean	float32
13	Idle Std	float32	27	Fwd IAT Mean	float32
14	Fwd Act Data Packets	float32	28	Packet Length Min	float32

3.1.3 Model Building, Training, and Evaluation

In this study, multiple machine learning model was developed, trained, and evaluated on three versions of datasets: small, medium and large. These models include Decision Tree (DT), Random Forest (RF), Naïve Bayes (NB), Ada Boost (AB), and Multi-Layer Perceptron (MLP). A total of 15 models waer trained – five models across each dataset version. Additionally, an ensemble model was constructed for each dataset version using the two best-performing models, selected based on a comprehensive set of evaluation metrics.

The methodology followed a multi-step pipeline:

- **Base Model Training:** Initially, all five base models were trained on each dataset version to establish baseline performance. This step was crucial for us to identify how well each algorithm could handle the different size and complexities of data without any optimization. The initial performance also served as a reference point for further improvements through hyperparameter tuning.
- **Hyperparameter Tuning:** Hyperparameter tuning was performed to enhance the model performancce and generalization. This process was performed exclusively on the small dataset, which served as the primary dataset fro model development and tuning. We employed GridSearchCV and Optuna as the primary optimization techniques. Each model was tuned independently using a validation split of the small dataset to identify the best-performing hyperparameter configurations. Once the optimal hyperparameters for each model were identified using the small dataset, we reused this same configuration when training and evaluating the models on the medium and large datasets. This approach prevented bias brought about by re-tuning on big datasets and guaranteed a constant baseline for comparison across all dataset sizes, separating the impact of dataset scale on model performance.
- **Model Evaluation (Validation Phase):** Each model was evaluated on a validation set (a subset of the training set) to observe the overfitting and underfitting issues. To guarantee optimal performance, models with weak generalization were interactively adjusted. Only well-regularized models made it to the testing stage.
- **Final Testing:** Once we find the best-performing models for each dataset, we evaluate the model on the test set to measure its performance and generalization ability.

- **Evaluation Metrics:** Model performance was assessed using a comprehensive suite of evaluation metrics:

Accuracy: The overall correctness of the model.

$$Accuracy = \frac{(Tp + TN)}{(TP + FP + TN + FN)} \quad (2)$$

Precision: The proportion of true positive predictions among all positive predictions.

$$Precision = \frac{TP}{TP + FP} \quad (3)$$

Recall (Sensitivity): The proportion of actual positives correctly identified.

$$Recall = \frac{TP}{TP + FN} \quad (4)$$

F1-scores: The harmonic mean of precision and recall, balancing both metrics.

$$F1\ Score = \frac{3 * Precision * Recall}{Precision + Recall} \quad (5)$$

False Positive Rate (FPR): The proportion of negative instance incorrectly classified as positive.

$$FPR = \frac{FP}{TN + FP} \quad (6)$$

Confusion Matrix: A detailed view of true positive, false positive, true negative, and false negatives.

Precision-Recall Curve: Used to visualized the trade-of between precision and recall.

3.1.4 Ensemble Model

The ensemble model was designed to combine predictions from multiple machine learning models, specifically for multi-label classification task (which involves both multiclass and binary

classification). The best two performing models were selected based on their test evaluation scores. The ensemble checks how well each model performs using a scoring method called log loss (Equation 7), which tells us how confident and correct each models predictions are. If a model makes a better prediction, it gets a higher weighted value, and if a model performs poorly, it gets a lower weighted value.

$$Logloss(y, p) = -(y * \log(p) + (1 - y) * \log(1 - p)) \quad (7)$$

Then, each model weights were calculated as the inverse of the log loss (Equitation 8). During the prediction process, each model makes predictions and provides probabilistic values.

$$w_i = \frac{1}{logloss_m} \quad (8)$$

$$p_{final} = \sum_{i=1}^n w_i * P_i \quad (9)$$

These predictions are then combined. Each models prediction is given a weight, which reflects how good that model is. Then for every class, we multiply each models predicted probability by its weights (Equation 9) and sum up the weighted probabilities across all models. Next, we pick the class with the highest probability as the final prediction.

3.1.4 Explainable AI (XAI) for Model Interpretation

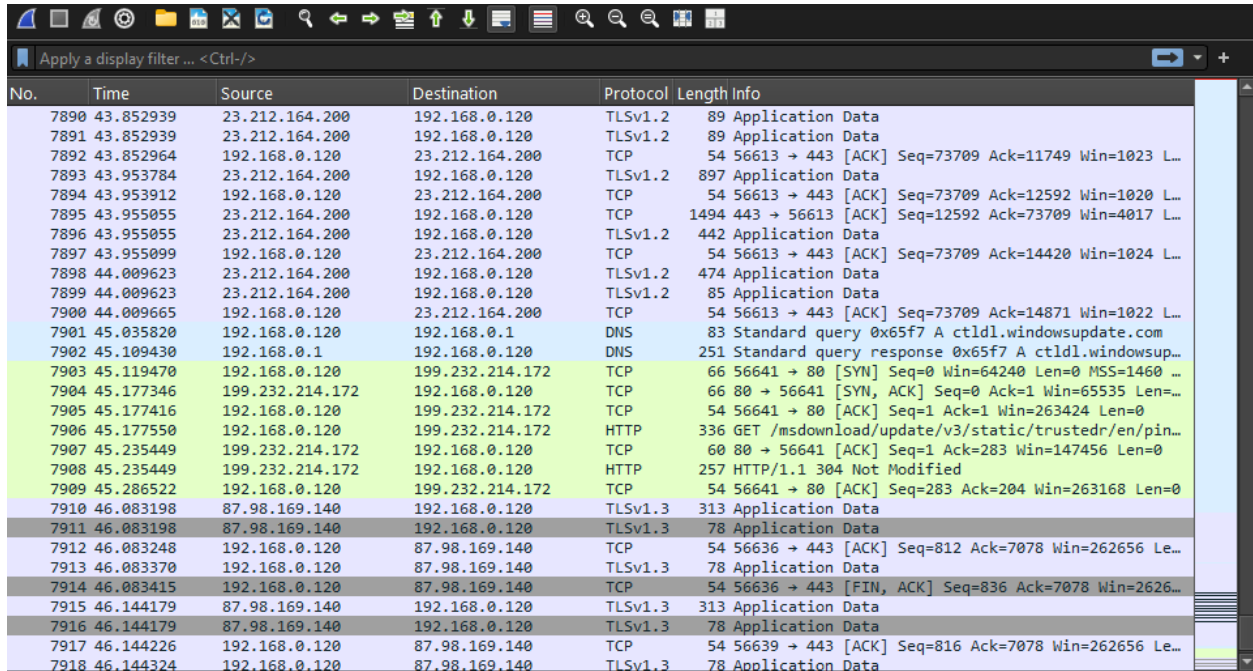
An explainable AI technique was applied to enhance the interpretability of the chosen ensemble models prediction. This is important in the domain of cybersecurity, where understanding the reason behind a mode's classification—whether network traffic is safe or malicious—is essential for building trust, supporting decision-making, and providing actionable insights to security analytics. Two widely used XAI methods were employed in this study.

- **SHAP (Shapley Additive exPlanations):** SHAP was used to provide a global and local interpretation of model predictions by quantifying the contribution of each feature to the output. It assigns each feature a value that represents its impact on the prediction, making it easier to understand how the model arrives at its decision across different data instance.

- **LIME (Local Interpretable Model-Agnostic Explanation):** Lime was used to generate local explanations for individual predictions. It works by creating a simple model just around one prediction to see how each input feature affects the results. This helps to understand why a specific sample was classified in a certain way, making the models behavior easier to understand one case at a time.

3.1.5 Real World Testing:

We have used two cybersecurity tools to test the models in real-world scenarios, including the Wireshark and CIC Flow Meter. Wireshark was utilized to capture raw network traffic packets, as illustrated in Figure 6. These packets were then converted into flow-based statistical features using the CIC Flow Meter. After further preprocessing, the extracted data was analyzed using our best-performing ensemble model.



No.	Time	Source	Destination	Protocol	Length	Info
7890	43.852939	23.212.164.200	192.168.0.120	TLSv1.2	89	Application Data
7891	43.852939	23.212.164.200	192.168.0.120	TLSv1.2	89	Application Data
7892	43.852964	192.168.0.120	23.212.164.200	TCP	54	56613 → 443 [ACK] Seq=73709 Ack=11749 Win=1023 L...
7893	43.953784	23.212.164.200	192.168.0.120	TLSv1.2	897	Application Data
7894	43.953912	192.168.0.120	23.212.164.200	TCP	54	56613 → 443 [ACK] Seq=73709 Ack=12592 Win=1020 L...
7895	43.955055	23.212.164.200	192.168.0.120	TCP	1494	443 → 56613 [ACK] Seq=12592 Ack=73709 Win=4017 L...
7896	43.955055	23.212.164.200	192.168.0.120	TLSv1.2	442	Application Data
7897	43.955099	192.168.0.120	23.212.164.200	TCP	54	56613 → 443 [ACK] Seq=73709 Ack=14420 Win=1024 L...
7898	44.009623	23.212.164.200	192.168.0.120	TLSv1.2	474	Application Data
7899	44.009623	23.212.164.200	192.168.0.120	TLSv1.2	85	Application Data
7900	44.009665	192.168.0.120	23.212.164.200	TCP	54	56613 → 443 [ACK] Seq=73709 Ack=14871 Win=1022 L...
7901	45.035820	192.168.0.120	192.168.0.1	DNS	83	Standard query 0x65f7 A ctldl.windowsupdate.com
7902	45.109430	192.168.0.1	192.168.0.120	DNS	251	Standard query response 0x65f7 A ctldl.windowsup...
7903	45.119470	192.168.0.120	199.232.214.172	TCP	66	56641 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 ...
7904	45.177346	199.232.214.172	192.168.0.120	TCP	66	80 → 56641 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=...
7905	45.177416	192.168.0.120	199.232.214.172	TCP	54	56641 → 80 [ACK] Seq=1 Ack=1 Win=263424 Len=0
7906	45.177550	192.168.0.120	199.232.214.172	HTTP	336	GET /msdownload/update/v3/static/trustedr/en/pin...
7907	45.235449	199.232.214.172	192.168.0.120	TCP	60	80 → 56641 [ACK] Seq=1 Ack=283 Win=147456 Len=0
7908	45.235449	199.232.214.172	192.168.0.120	HTTP	257	HTTP/1.1 304 Not Modified
7909	45.286522	192.168.0.120	199.232.214.172	TCP	54	56641 → 80 [ACK] Seq=283 Ack=204 Win=263168 Len=0
7910	46.083198	87.98.169.140	192.168.0.120	TLSv1.3	313	Application Data
7911	46.083198	87.98.169.140	192.168.0.120	TLSv1.3	78	Application Data
7912	46.083248	192.168.0.120	87.98.169.140	TCP	54	56636 → 443 [ACK] Seq=812 Ack=7078 Win=262656 Le...
7913	46.083370	192.168.0.120	87.98.169.140	TLSv1.3	78	Application Data
7914	46.083415	192.168.0.120	87.98.169.140	TCP	54	56636 → 443 [FIN, ACK] Seq=836 Ack=7078 Win=2626...
7915	46.144179	87.98.169.140	192.168.0.120	TLSv1.3	313	Application Data
7916	46.144179	87.98.169.140	192.168.0.120	TLSv1.3	78	Application Data
7917	46.144226	192.168.0.120	87.98.169.140	TCP	54	56639 → 443 [ACK] Seq=816 Ack=7078 Win=262656 Le...
7918	46.144324	192.168.0.120	87.98.169.140	TLSv1.3	78	Application Data

Figure 6: Packet Capturing Using Wireshark Tool

3.2 Software Components

TABLE 5: LIST OF SOFTWARE TOOLS & TECHNOLOGIES TABLE

Tool	Functions	Other similar Tools (if any)	Why selected this tool
Python3	Programming language for model development, preprocessing, and scripting	R, Java	Widely used in data science and ML, large ecosystem, easy syntax, excellent library support
Scikit-learn	ML library used for model building, training, evaluation, and preprocessing	TensorFlow	Simple API, strong support for classical ML models, integrates with Python ecosystem
PyTorch	Deep learning framework used for implementing MLP models	TensorFlow, Keras	Flexible and dynamic computation graph, widely used in academia, easier debugging and customization
Visual Studio Code	Code editor used for python development and debugging	PyCharm, Sublime Text, Atom	Lightweight, extensible, integrated terminal, excellent support for Python and Git.
Google Colab	Cloud-based notebook for writing and executing python code with GPU support	Jupyter Notebook, Kaggle Kernels	Free access to GPU for deep neural networks, easy sharing and collaboration
Git/GitHub	Version Control system for code management and collaboration	GitLab, Bitbucket	Industry standard for version control, enables collaboration, backup, and reproducible development
Flask	Lightweight web framework for building web applications and model deployment	Django, FastAPI	Minimal and easy to set up, perfect for building quick and simple for model deployment.
CICFlowMeter	Flow-based feature extraction tool for network traffic	NetFlow	Specifically designed for creating CICIDS-compatible flow features from pcap or live captured data
Wireshark	Network protocol analyzer used for capturing and analyzing network packets	Tcpdump	User-friendly GUI, powerful filtering and visualization capabilities, ideal for analyzing raw traffic data.

3.3 Software Implementations

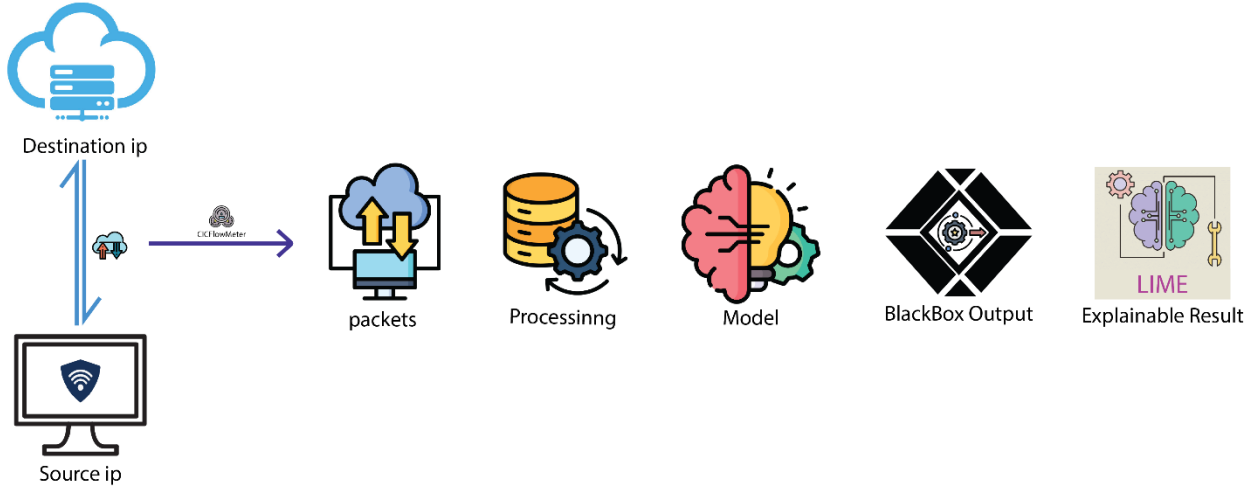


Figure 7: System Workflow Process for Network Activity Classification

The implementation was focused primarily on the software components in the cybersecurity domain, particularly the network packet capturing, packet data processing, and model development. Initially, the CIC-IDS2017 dataset was collected and preprocessed for training machine learning models. Several models were trained and evaluated, and the best two-performing models were selected based on the evaluation results. Implemented an ensemble model with the best-performing model were selected based on various evaluation metrics scores. These models were then combined into an ensemble model, which was proposed for the final prediction task to enhance performance and stability. In the real-world simulation, raw network traffic data was captured using Wireshark and saved in a 'pcap' format. These 'pcap' files were then converted into 'csv' format using the CIC Flow Meter, which extracted relevant flow-based statistical features for the traffic data packets. The resulting data was further preprocessed and fed into the classification models for multi-label prediction tasks. The model then provides the multi-label classification results in black box format. Explainable AI techniques SHAP and LIME were applied, to interpret and explain the model decisions, providing insights into the features influencing each prediction.

Chapter 4 Investigation/Experiment, Result, Analysis and Discussion

Each of these models result was analyzed based on various performance metrics to find out the impact of syntactic data and find out the best performing model. We analyzed the performance of our models based on the selected features. It shows us from Decision Trees, Random Forests, Naïve Bayes, AdaBoost, and Deep Neural Network, in every version of dataset the Decision Trees, and Random Forest Performed Better. The Ensemble model which was created with these best two performing model was performed more better compare to a single model.

4.1 Performance Analysis on Small Dataset

TABLE 6: CLASSIFICATION RESULTS OF SMALL DATASET

	Binary Classification Results					Multiclass Classification Results				
	Acc	Prec.	Rec.	F1	FPR	Acc	Prec.	Rec.	F1	FPR
Decision Trees	0.9613	0.9564	0.9972	0.9763	0.1822	0.9707	0.9709	0.9707	0.9707	0.0073
Random Forests	0.9779	0.9749	0.9980	0.9863	0.1028	0.9744	0.9750	0.9744	0.9746	0.0064
Naïve Bayes	0.7368	0.9397	0.7171	0.8134	0.1842	0.8472	0.8608	0.8472	0.8427	0.0383
AdaBoost	0.9021	0.8911	0.9999	0.9424	0.4895	0.3602	0.1645	0.3602	0.2171	0.1597
Multi-Layer Perceptron	0.9596	0.9851	0.9641	0.9745	0.0584	0.9560	0.9584	0.9560	0.9564	0.0110
Ensemble Model (DT + RF)	0.9695	0.9652	0.9978	0.9812	0.1441	0.9729	0.9731	0.9729	0.9729	0.0068

This small version contains fewer samples, which made it challenging for the model to effectively learn the patterns associated with different types of malicious activity. The result on Table 6 achieved 96.95% accuracy for binary classification and 97.29% for multiclass classification of the Ensemble Model. At the same time, the model showed a lower false positive rate of 0.0068 for multiclass classification and a higher false negative rate for binary classification with 0.1441, indicating that the model was struggled more to correctly identify malicious instances in binary classification tasks.

In this dataset version, synthetic data was generated for two types of malicious classes, while the remaining three classes contain the original real-world data. Overall, this small version of the dataset struggled to provide enough diverse and balanced information, making it difficult for the model to learn effectively and generalize well.

4.2 Performance Analysis on Medium Dataset

TABLE 7: CLASSIFICATION RESULTS OF MEDIUM DATASET

	Binary Classification Results					Multiclass Classification Results				
	Acc	Prec.	Rec.	F1	FPR	Acc	Prec.	Rec.	F1	FPR
Decision Trees	0.9682	0.9547	0.9948	0.9795	0.1187	0.9791	0.9597	0.9691	0.9692	0.0054
Random Forests	0.9766	0.9725	0.9975	0.9849	0.0919	0.9734	0.9741	0.9734	0.9735	0.0047
Naïve Bayes	0.8649	0.8554	0.9909	0.9182	0.5458	0.7000	0.8169	0.7000	0.6805	0.0491
AdaBoost	0.8826	0.8671	0.9999	0.9288	0.4995	0.3402	0.1379	0.3402	0.1897	0.1212
Multi-Layer Perceptron	0.9382	0.9836	0.9348	0.9586	0.0509	0.9172	0.9276	0.9165	0.9165	0.0148
Ensemble Model (DT + RF)	0.9719	0.9671	0.9972	0.9819	0.1107	0.9730	0.9731	0.9726	0.9726	0.0048

The Table 6 results demonstrate that the ensemble model built using Decision Tree and Random Forest classifiers, achieved the most balanced and effective performance across both binary and multiclass classification tasks. It achieved an accuracy of 97.19% for binary classification and 97.30% for multiclass classification, maintaining high precision, recall, and F1 scores while keeping a low false positive rate. Among the individual models, Random Forest showed strong performance with consistent metrics across both tasks. The Decision Tree and Multi-Layer Perceptron models offered good results compared to the other two models. The other two models failed to provide competitive results.

In this medium dataset, synthetic data was generated for four different malicious the classes in order to address class imbalance and improve model learning. Meanwhile, the benign class and the other two malicious classes contain the original samples. Although the synthetic data helped balance the dataset, the model faced difficulties in accurately classifying the artificially generated samples. The synthetic data failed to capture the complex patterns of different kinds of malicious activity. As a result, the model showed reduced performance when distinguishing between certain attack types.

4.3 Performance Analysis on Large Dataset

TABLE 8: CLASSIFICATION RESULTS OF LARGE DATASET

	Binary Classification Results					Multiclass Classification Results				
	Acc	Prec.	Rec.	F1	FPR	Acc	Prec.	Rec.	F1	FPR
Decision Trees	0.9667	0.9711	0.9539	0.9624	0.0230	0.9662	0.9665	0.9660	0.9656	0.0094
Random Forests	0.9716	0.9849	0.9511	0.9677	0.0118	0.9716	0.9721	0.9716	0.9713	0.0081
Naïve Bayes	0.6998	0.5998	0.9883	0.7465	0.5338	0.6520	0.8160	0.6502	0.6627	0.0541
AdaBoost	0.7224	0.6171	0.9999	0.7632	0.5021	0.5534	0.3055	0.5527	0.3935	0.1429
Multi-Layer Perceptron	0.8850	0.9488	0.7852	0.8593	0.0343	0.8728	0.8873	0.8724	0.8469	0.0373
Ensemble Model (DT + RF)	0.9692	0.9774	0.9533	0.9652	0.0179	0.9691	0.9697	0.9691	0.9688	0.0088

The result of the larger Dataset of Table 8 show that the Ensemble Model performed best overall achieving high accuracy, precision, and F1 scores in both binary and multiclass classification with low false positive rates. However, the large dataset contains a large number of benign (safe) samples, which influenced the model to focus more on those patterns, making it more challenging to accurately learn the complex characteristics of malicious samples. As a result, the model exhibited some bias in its predictions.

4.4 SHAP Plot Analysis on Medium Dataset with Ensemble Model

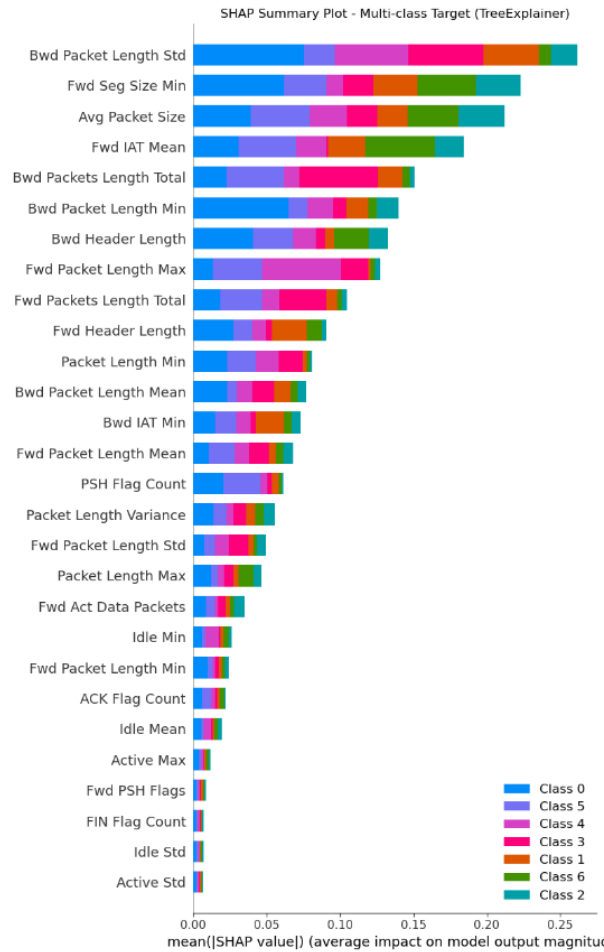


Figure 8: SHAP Summary Plot for Multiclass Result

The SHAP summary plot is a global interpretability tool that illustrates the impact of different features on the models predictions across all classes. In Figure 8, the SHAP summary plot highlights the most important features influencing the models predictions for multiclass

classifications. Features like **Bwd Packet Length Std**, **Fwd Seg Size Min**, and **Avg Packet Size** had the highest impact across multiple classes. Each color represents whether a low or high feature value pushes the model toward a particular class. This visualization emphasizes how specific network traffic characteristics contribute significantly to classifying different types of malicious activities.

4.5 LIME Plot Analysis on Medium Dataset with Ensemble Model

The LIME visualization allows for an in-depth analysis of how the model processes individual data points. Figures 9 and 10 highlight a local explanation of the probability of the predicted model. It helps to understand why the model classified a particular instance as **Benign** or other classes. The left bar chart in each plot represents the predicted probabilities for each possible class. In Figure 9, the model assigns 0.49 probability to **Benign**, making it the predicted class. Where the next highest probability is 0.34 for **Web Attack**, followed by smaller probabilities for **Brute Force** and nearly zero for other classes. This distribution shows that while **Benign** is the top prediction, the model also sees some possibility of other types.

The right side of Figure 9 highlights the feature contributions and which features most influenced the prediction toward or away from the **Benign** class. The orange bars represent those that support the prediction of **Benign**, whereas the blue bars represent features that push the prediction toward **Web Attack**. Features like **Bwd Packet Length Min** and **Fwd Seg Size Min** are strong negative contributions, pushing against the Benign prediction. On the other hand, **Bwd Packet Length Std**, **Bwd IAT Min**, and **Fwd Header Length** are influencing the model toward classifying it as **Benign**.

In Figure 10, the LIME explanation shows how the features are contributed to the models prediction for **Port Scanning**, where the model classified the instance as **Port Scanning** with a probability of 0.97. These plots help to explain the local reason for the model for single instances. While the model predicted an instance as **Benign** or **DDoS** with the highest probability, there were conflicting features influences for other classes. LIME reveals this decision boundary by showing how specific feature affect the outcome, making the models decision more interpretable.

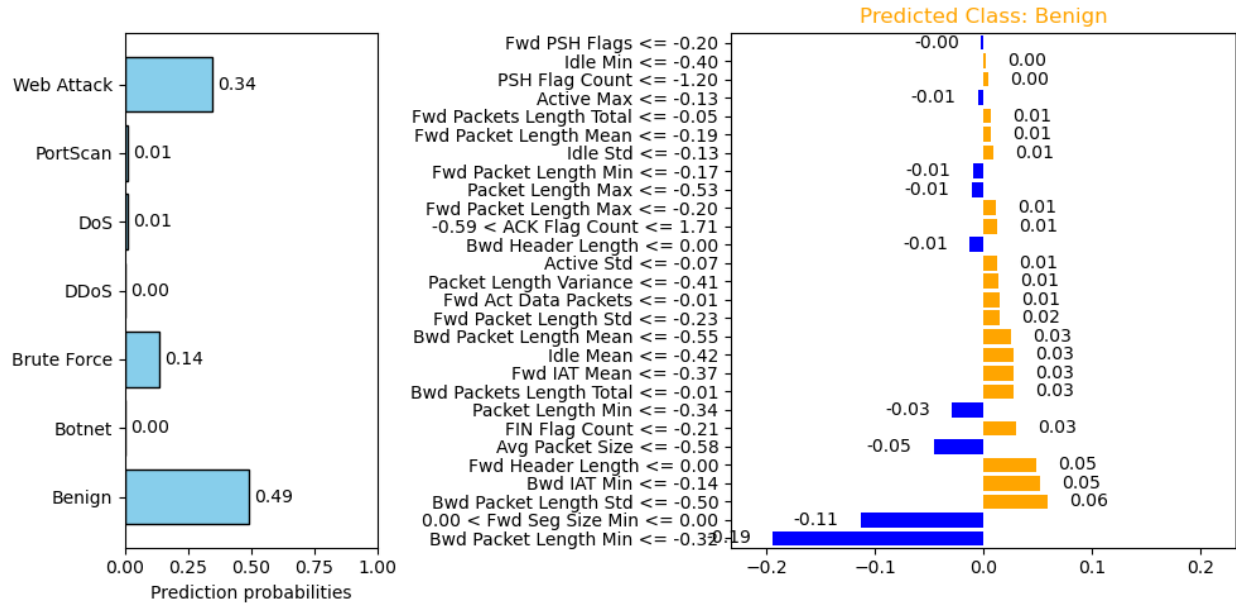


Figure 9: LIME Explanation on Benign Class Prediction

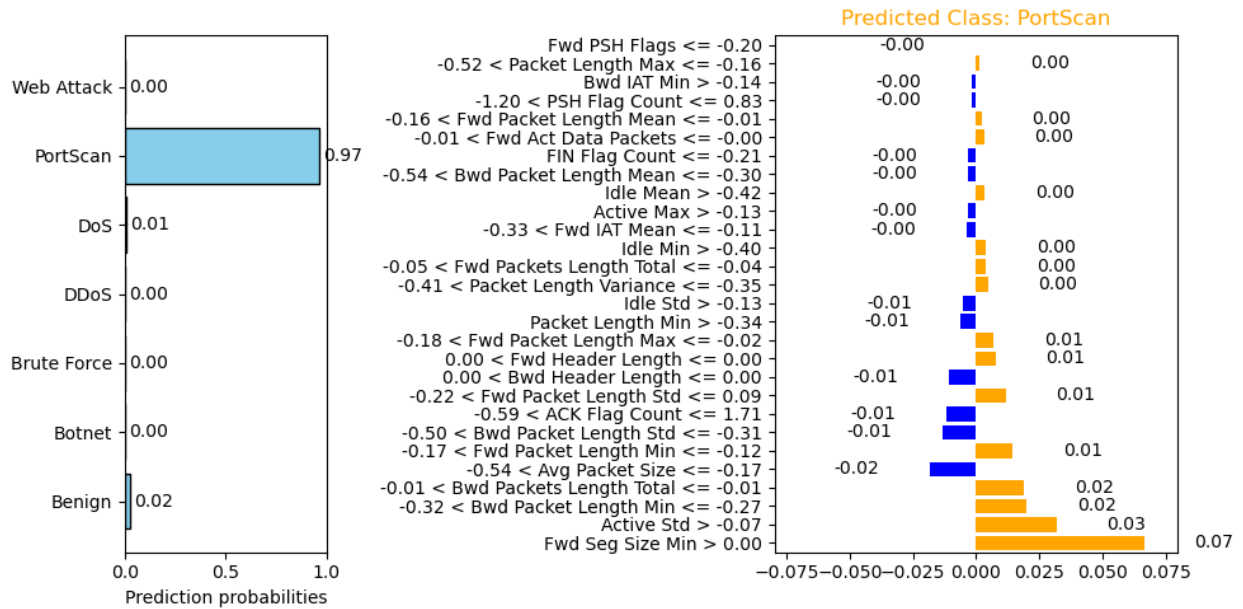


Figure 10: LIME Explanation on Benign Class Prediction

4.6 Real-World Testing

We have captured the raw network traffic packets using the Wireshark tools, allowing us to monitor our machine's live browsing activity. This included everyday browsing behavior like visiting websites, streaming videos, and checking emails—typical examples of safe or benign

network activity. Once the data was captured, we used the CIC Flow Meter to process these raw packets into flow-based statistical features, which are essential for our model. These features include the packet length, source or destination IP, flow duration, protocol type, etc., which indicate whether traffic might be malicious or safe. After preprocessing the data, we fed this data into our ensemble model to monitor our network activity types, which recognize normal behavior. As shown in Figure 11, our system correctly classified this activity as benign, irrespective of the protocol type, source or destination IP, and session. This real-world testing step was crucial to validate the practical effectiveness of our intrusion detection system.

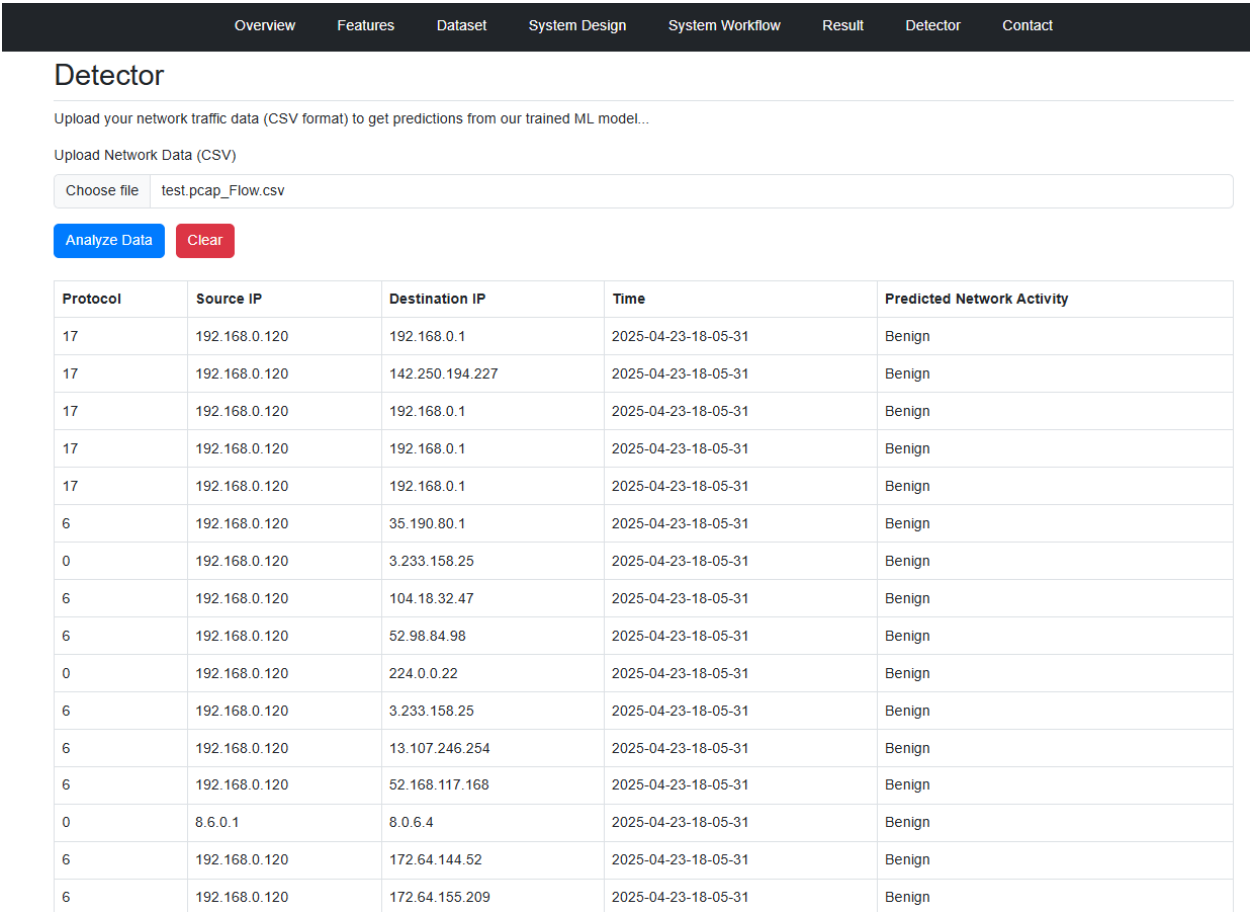


Figure 11: Real-world Network Activity Testing

4.7 Discussion

The result of the study explores the effects of different percentages of synthetic data on network intrusion detection performance. The analysis showed that combining synthetic data with

real data in a balanced way helps the model generalize better and achieve higher accuracy. In the medium dataset, where synthetic data was used moderately, the models achieved optimal performance, including that synthetic samples effectively supported learning without overwhelming the real patterns. However, in the small dataset, limited real data made it harder for the model to learn malicious patterns accurately, while in the large dataset, an excessive amount of synthetic data introduced noise, leading to biased or overfitted predictions. These findings suggest that while synthetic data can help address class imbalance and fill in gaps where data is limited, using too much of it can be a problem. If the balance isn't right, the model might miss out on learning the real patterns in the data. Our ensemble model helped manage this by combining the strengths of different algorithms, which made it better at dealing with the ups and downs of synthetic data.

Chapter 5 Impacts of the Project

5.1 Impact of this project on Social, Health, Safety, Legal and Cultural Issues:

5.1.1 Social Impact: The initiative brings a critical tool that enhances digital security and contributes to a more secure digital world. By integrating Explainable AI (XAI) in intrusion detection, it helps build a cybersecurity ecosystem that is more transparent—ensuring understanding and building trust in AI-driven decisions not just with experts but with the general public. In addition, the system increases public awareness of cyber threats and equips individuals and institutions with the necessary information to adopt proactive defense measures. Moreover, democratized access to secure systems opens the doors to small businesses and marginalized people to enjoy the use of advanced threat detection without the need to have large IT infrastructures, thus advocating for a more equitable technology.

5.1.2 Health Impact: The system, adopted to date, improves the mental health of cybersecurity professionals at the same time that it relieves them of stress. In the customary approach, the detection and classification of harmful network traffic require a systematic and detailed analysis of vast network traffic data, a process that is both time-consuming and mentally demanding. Through the automating of the detection and classification of cyber threats, the program reduces mental burdens, hence allowing personnel to concentrate their energies on more strategic and less routine duties, leading to improved job satisfaction and reduction of burnout episodes. In addition, the system protects telemedicine platforms and electronic health record systems in the healthcare sector, thus helping maintain the privacy of health information.

5.1.3 Safety Impact: In the operationally critical sectors like transportation, healthcare, and critical infrastructures, cyberattacks pose immense risks. The threat detection capability of the project allows threats to be identified and mitigated before any causing of harm. By keeping regulatory requirements and maintaining continuity of business processes, the project considerably improves both digital and physical security controls. In emergency response situations like cyber warfare or massive ransomware attacks, the competence of the system to detect and analyze anomalies in network activity helps ensure better response and incident management.

5.1.4 Legal Impact: The addition of explainability through SHAP and LIME enhances investigation processes and creates a legal pathway for intrusion incident configurations, allowing for extensive incident investigations regarding the root causes of cyberattacks, and facilitates preservation and bottling of digital evidence required for legal cases and audit compliance. Furthermore, this solution will have compliance with future legal requirements referencing accountability and AI system transparency, which will further validate the solution in a changing legal environment.

5.1.5 Cultural Impact: The inclusion of bleeding edge technologies like machine learning and explainable artificial intelligence (XAI) into cybersecurity fundamentally alters organizations and their understanding of their digital defense. This some change would encourage a positive trajectory by assisting organizations engage with more complex and automated systems, as opposed to actionable responses and rule-based systems; also concern regarding the emphasis on interpretability and understanding will help to foster a culture of transparency, which in turn will help bolster public trust in automated security systems and governance of digital regimes.

5.2 Impact of this project on Environment and Sustainability:

5.2.1 Environmental Impacts:

The project reduces the reliance on paper-based network logs and manual intrusion detection workflows that previously relied on printing and storing a large volume of separate network activity reports. Through full digitalization and cloud data storage, we eliminate paper waste and environmental impact, like deforestation, energy used to generate paper, and waste decomposition producing greenhouse gases such as methane. Digitizing the threat detection process also provides energy savings through the smaller footprint of security operations-related infrastructure (i.e. server room and filing space).

Automating document-heavy activity using AI reduces the reliance on physical infrastructure as well. This leads to lower consumption of real estate, electricity, and hardware used to store, retrieve, and analyze hard-copy logs. In the long run, the system will influence IT teams to adopt more environmentally conscious conduct around green digital workflows.

5.2.2 Sustainability Impacts:

Operational sustainability is also amplified when time-consuming processes, like data preprocessing, threat classification, and explaining anomalies, are done automatically. Cybersecurity teams are thus able to deal with more threats within a limited time without the requirement of increasing human resources, thereby boosting economic and process efficiencies. This not only reduces operational costs but also allows for reinvestment in sustainable projects, such as eco-friendlier systems and renewable infrastructures.

In a wider sense, the system supports a culture of digital sustainability at the same time that minimizes the carbon footprints that are a result of IT operations. The increased resistance to expected future cyber-attacks to ensure long-term availability and reliability of digital services supports institutional sustainable development without causing them to neglect their environmental obligations. In addition, the system's promotion of the availability of and access to security software continues to benefit small firms by enabling their sustainable digital actions, ultimately leading to a more resilient technology economy.

Chapter 6 Project Planning and Budget

Task	1 st Term			2 nd Term		3 rd Term	
	December	January	February	February	March	March	April
Project Plan							
Research Literature							
Problem Finding							
Prepare & Submit Project Proposal							
System Design							
Data Collection							
Model Development & Training							
Testing & Evaluation							
Final Demo							
Final report							
Final Presentation							

Figure-12: Project Timeline Gantt chart.

TABLE 9: ESTIMATED BUDGET TABEL

Item	Description	Approximate Cost
Software Licenses	Open-source tools are used.	0 BDT
Data Acquisition	Data is collected from publicly available source (Kaggle).	0 BDT
Hardware	Available desktop and laptop were used.	0 BDT
Computing Resources	Cloud based free open-source software are used for GPU access (Google Colab) and available core i5, 11 th gen, 16 GB RAM device was used for rest of the task.	0 BDT
Web Interface Development	Locally deployed.	0 BDT
Total Estimated Budget		0 BDT

Chapter 7 Complex Engineering Problems and Activities

7.1 Complex Engineering Problems (CEP):

TABLE 10. A SAMPLE COMPLEX ENGINEERING PROBLEM ATTRIBUTES TABLE

Attributes		Addressing the complex engineering problems (P) in the project
P1	Depth of knowledge required (K3-K8)	This project demands a diverse and in-depth knowledge base: web security and advanced cybersecurity principles (K3), data science and machine learning fundamentals (K4), and proficiency in programming with Python and frameworks such as Flask, scikit-learn, and PyTorch (K5). It also requires expertise in ensemble modeling techniques—combining multiple machine learning algorithms to improve detection accuracy—as well as the implementation of Explainable AI (XAI) methods for transparent result. Comprehensive data preprocessing and analysis skills are essential, including the use of numpy, Pandas, and matplotlib. (K6). Furthermore, the project necessitates literature review and exploration of online datasets to stay current with evolving attack patterns (K7), and the application of advanced machine learning concepts for classification, anomaly detection, and network monitoring in cybersecurity contexts (K8).
P2	Range of conflicting requirements	Key conflicts include balancing detection accuracy with real-time performance (speed vs. thoroughness), scalability to handle large-scale network traffic versus available computational resources, and the trade-off between comprehensive analysis and a user-friendly, responsive system interface.
P3	Depth of analysis required	The project necessitates sophisticated cybersecurity analysis: uncovering subtle and complex attack patterns within high-dimensional network traffic data, including detecting and classifying whether activity is safe or represents a specific attack type. It involves selecting and fine-tuning optimal machine learning models (e.g., XGBoost, Random Forest), and incorporates Explainable AI (XAI) to interpret model predictions for greater transparency. Rigorous evaluation is performed using metrics such as accuracy, precision, recall, F1-score, and confusion matrices to ensure the system's reliability, generalizability, and practical value in real-world network security contexts.
P4	Familiarity of issues	Although web attacks like DoS, DDoS, SQL Injection, XSS, Botnet, and Brute Force are well-known, their evolving signatures and obfuscation techniques require continual adaptation. Familiar tasks such as parsing and analyzing network traffic, implementing

		ML pipelines present ongoing challenges due to data complexity and adversarial tactics.
P5	Extent of applicable codes	There are no direct engineering standards; however, the project follows best cybersecurity practices (e.g., secure coding, data handling, and privacy), software engineering, and machine learning development. The use of reputable open-source libraries and adherence to ethical data usage guidelines are emphasized.
P6	Extent of stakeholder involvement	Stakeholders include developers, system administrators, Cybersecurity analyst integrating with production environments, business/end users benefiting from enhanced security, and the academic/research community contributing to and validating detection strategies and datasets.
P7	Interdependence	The system's components are tightly interdependent: machine learning models rely on the quality of preprocessed input data, the frontend depends on backend API responsiveness, and overall effectiveness is contingent on accurate logging, continuous updates, and user feedback loops.

7.2 Complex Engineering Activities (CEA):

TABLE 11. A SAMPLE COMPLEX ENGINEERING PROBLEM ACTIVITIES TABLE

Attributes		Addressing the complex engineering activities (A) in the project
A1	Range of resources	Implementation of this IDS project required a diverse set of resources, including skilled personnel in data science, cybersecurity, and software engineering. The project leveraged a suite of software tools, such as Python (with libraries like Pandas, Scikit-learn, and PyTorch), Flask for web development, and both Jupyter Notebook and Visual Studio Code (VSCode) as primary development environments for experimentation, prototyping, and code management. Computation was supported by both local machines and cloud-based platforms (e.g., Google Colab) to handle large-scale data processing and model training.
A2	Level of interactions	The project featured multi-level collaboration: internal teamwork for system design and code review, consultation with cybersecurity specialists and academic advisors for domain-specific feedback, and iterative engagement with potential end-users to refine the web interface and usability. Moreover, the team interacted with open-source communities and utilized publicly available datasets to ensure the system's robustness and real-world relevance.
A3	Innovation	The system introduces innovative use of ensemble machine learning techniques for real-time intrusion detection,

		<p>automating the classification of a wide range of cyberattacks. It combines feature engineering, advanced model selection, and explainable AI to produce actionable insights, thereby reducing dependence on manual network traffic data and improving the proactive defense capabilities of web infrastructure.</p>
A4	Consequences to society / Environment	<p>By improving the detection and classification of network attacks, the project strengthens the resilience of organizations and individuals against cyber threats. It helps prevent data breaches, financial losses, and privacy violations, contributing to a more secure digital environment. The project also fosters greater awareness of cybersecurity best practices and the importance of proactive monitoring in maintaining the integrity of digital services.</p>
A5	Familiarity	<p>The project builds upon established practices in machine learning, software engineering, and cybersecurity, but also expands the team's expertise into advanced topics such as ensemble learning, data balancing (SMOTE), and explainable AI. The work aligns with the United Nations Sustainable Development Goal (SDG) #9: Industry, Innovation, and Infrastructure, by promoting technological innovation and infrastructure resilience in the digital domain.</p>

Chapter 8 Conclusions

8.1 Summary:

This study provides an explainable and robust approach to creating a machine learning and deep neural network-based intrusion detection tool. By utilizing synthetic data generated by SMOTE, we overcome the widespread challenge of class imbalance in cybersecurity data. Through creating three variations of the dataset—small, medium, and large—we analyze how varying ratios of synthetic data influence model performance. The finding indicated that synthetic data can enhance learning, especially when the data is limited in the real world. However, its success depends on a strict balance to maintain the complexity of real network behavior.

We employed an ensemble model combining decision tree and random forest classifier predictions to improve classification accuracy and reliability. This approach successfully leveraged the strengths of both models and yielded improved results in binary and multiclass cases. We also integrated the Explainable AI (XAI) techniques, SHAP and LIME to further enhance the system's transparency and dependability. This helped us to gain a deeper understanding of the models predictions, especially when it made a mistake by showing us which features were influencing each prediction. The system was then tested with actual network traffic from the real world, which was captured using Wireshark and analyzed by CIC Flow Meter to simulate real-world environments.

In summary, the study offers the feasibility of applying ensemble learning, explainability, and synthetic data to create an improved, understandable, and cost-effective intrusion detection system that is able to provide solutions to evolving network security threats.

8.2 Limitations:

The project has several limitations; it relies on a single dataset that does not include all types of cyberattack data, which could affect the mode's ability to detect new threats. In addition, though synthetic data helps balance the dataset, it may not effectively capture rare attacks and sophisticated patterns and, therefore, affect decision accuracy. Also, while the system was built with real-world use in mind, it is not yet ready for real-time detection in live environments due to limited resources.

8.3 Future Improvement:

In future, this system could be improved by incorporating real-time detection capabilities to monitor network activities and address live cyberattacks. Expanding the dataset to include a broader range of cyberattacks, especially common threats, would enhance the models ability to recognize various types of attacks. The class misbalancing issue can be address by collecting real-world data would be more effective than generating synthetic data.

References

- [1] K. Thompson, "Most Common Website Security Attacks and How to Protect Yourself," 2024. [Online]. Available: <https://www.tripwire.com/state-of-security/most-common-website-security-attacks-and-how-to-protect-yourself>.
- [2] M. H. K. a. R. T. Patil, "A Review of Intrusion Detection System Using Machine Learning Approach," *Journal of Information Security*, vol. 6, pp. 131-137, 2015.
- [3] J. D.-V. G. M.-F. a. E. V. P. Garcia-Teodoro, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Computers & Security*, vol. 28, pp. 18-28, 2009.
- [4] K. T. Hanna, "Wireshark," TechTarget, [Online]. Available: <https://www.techtarget.com/whatis/definition/Wireshark>. [Accessed 23 4 2005].
- [5] U. o. N. Brunswick, "Research Applications," Canadian Institute for Cybersecurity, [Online]. Available: <https://www.unb.ca/cic/research/applications.html>. [Accessed 23 4 2025].
- [6] K. W. B. L. O. H. W. P. K. N. V. Chawla, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, p. 321–357, 2011.
- [7] M. H. A. M. M. T. P. S. M.A. Ganaie, "Ensemble deep learning: A review," *Engineering Applications of Artificial Intelligence*, vol. 115, 2022.
- [8] D. S. P. S. C. Gaspar, "Explainable AI for Intrusion Detection Systems: LIME and SHAP Applicability on Multi-Layer Perceptron," *IEEE Access*, vol. 12, pp. 30164-30175, 2024.
- [9] S. G. a. M. P. C. Callegari, "A real-time deep learning based approach for detecting network attacks," *Big Data Research*, vol. 36, 2024.

- [10] Z. J. A. A. S. K. M. T. S. A. A. a. A. U. R. U. Ahmed, "Explainable AI-based innovative hybrid ensemble model for intrusion detection," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 13, 2024.
- [11] K. R. M. A. A. T. S. M. A. H. K. J. T. a. A. U. R. M. Sajid, "Enhancing intrusion detection: A hybrid machine and deep learning approach," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 13, 2024.
- [12] M. N. A. S. T. A. E.-H. M. A. T. S. a. M. A. K. U. Ahmed, "Signature-based intrusion detection using machine learning and deep learning approaches empowered with fuzzy clustering," *Scientific Reports*, vol. 15, 2025.
- [13] K. F. H. M. M. I. M. A. U. A. A. M. A. Y. F. A. a. M. A. M. M. A. Talukder, "A dependable hybrid machine learning model for network intrusion detection," *arXiv*, vol. 72, p. 103405, 2023.
- [14] S. T. P. K. S. V. A. K. S. a. S. S. N. Katiyar, "AI and cyber-security: Enhancing threat detection and response with machine learning," *Educational Administration: Theory and Practice*, vol. 30, no. 4, pp. 6273-6282, 2024.
- [15] M. S. M. H. a. U. F. S. Saleem, "Web Server Attack Detection using Machine Learning," in *2020 International Conference on Cyber Warfare and Security (ICCWS)*, Islamabad, Pakistan, 2020.
- [16] G. S. Richa S., "Novel Framework for Anomaly Detection Using Machine Learning Technique on CIC-IDS2017 Dataset," *Proceedings of the 2021 International Conference on Technological Advancements and Innovations (ICTAI)*, 2021.
- [17] R. L. A. a. E. H. Nfaoui, "Deep learning for vulnerability and attack detection on web applications: A systematic literature review," *Future Internet*, vol. 14, no. 4, p. 118, 2022.

- [18] F. J. A.-N. K. G. K. S. L. a. J. A. C. K. Ghanem, "Support Vector Machine for Network Intrusion and Cyber-Attack Detection," in *2017 Sensor Signal Processing for Defence Conference (SSPD)*, London, 2017.
- [19] W. Z. a. W. Y. J. Liang, "Anomaly-based web attack detection: A deep learning approach," in *The 2017 VI International Conference*, Beijing, China, 2017.
- [20] I. H. L. A. a. G. A. A. Sharafaldin, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," *Proceedings of the 4th International Conference on Information Systems Security and Privacy - ICISSP*, pp. 108-116, 2018.