NETWORK INTRUSION DETECTION USING XG-BOOST ALGORITHM

PROJECT REPORT

Submitted by

GUNASEKARAN. S	**712221106003**
KABIL. S	**712221106005**
KANISHKAA. P	**712221106006**
MANOJ. K	**712221106011**

In partial fulfilment for the award of the degree

Of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

PARK COLLEGE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institution)

KANIYUR, COIMBATORE 641659

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2025

ANNA UNIVERSITY: CHENNAI 600025

BONAFIDE CERTIFICATE

Certified that this project report **"NETWORK INTRUSION DETECTION USING XG-BOOST ALGORITHM"** is the Bonafide work of "**GUNASEKARAN.S**, **KABIL.S, KANISHKAA.P AND MANOJ.K**" who carried out the project work under my supervision.

SIGNATURE

Dr.M.Rajaram M.E., Ph.D.,

Professor,

HEAD OF THE DEPARTMENT,

Department of ECE,

Park College of Engineering and

Technology, Kaniyur,

Coimbatore - 641 659.

SIGNATURE

Mrs. C. Preethibha B.E., M.E.,

Associate Professor,

SUPERVISOR,

Department of ECE,

Park College of Engineering and

Technology, Kaniyur,

Coimbatore - 641 659.

Submitted for the University Project VIVA-VOCE examination held on

_____

INTERNAL EXAMINER EXTERNAL EXAMINER

ACKNOWLEDGEMENT

First and foremost, we praise and thank Almighty God and our beloved parents for their blessings and unwavering support throughout this project.

We express our deepest gratitude to our esteemed Chairman, **Dr. P. V. Ravi, Ph.D., MISTE** for providing us the opportunity to undertake this project and for his visionary leadership.

We sincerely thank our Chief Executive Officer, **Dr. Anusha Ravi, B.E., M.S. (USA), Ph.D.,** for her invaluable support and guidance.

We thank our Principal, **Dr. D. Lakshmanan, M.E., Ph.D.,** for his encouragement and for fostering an innovative learning environment.

We are grateful to our Vice Principal and Head of Department, **Dr. M. Rajaram, M.E., Ph.D.,** for his insightful suggestions and support.

We owe special gratitude to our project guide, **Mrs. C. Preethibha, M.E.**, for her expert advice and patient guidance throughout this project.

Lastly, we thank all faculty members, teaching and non-teaching staff, and lab in-charges for their contributions to this project's success.

ABSTRACT

With the rapid advancement of mobile communication and the rise of high-speed, low-latency networks, the demand for intelligent and robust network security has grown significantly. Traditional Network Intrusion Detection Systems (NIDS) often fall short in handling the complexity and scale of modern cyber threats. This study presents an AI-driven Network Threat Detection System that employs multiple machine learning algorithms—XGBoost, Random Forest, Decision Tree, and Logistic Regression—to accurately identify and mitigate network intrusions. The system processes network flow and event data through labeling, filtering, and preprocessing stages to prepare for effective model training. XGBoost is utilized for both feature selection and classification, enhancing detection accuracy by focusing on the most relevant attributes. Random Forest and Decision Tree classifiers are adept at uncovering complex patterns, while Logistic Regression offers a lightweight and interpretable baseline. Experimental evaluations reveal that ensemble models like XGBoost and Random Forest outperform others in terms of accuracy, precision, and recall. The integration of diverse algorithms not only boosts detection performance but also strengthens resilience against emerging and zero-day threats. This comprehensive approach advances the effectiveness of NIDS, delivering a scalable, adaptive, and high-performance solution for securing next-generation networks.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

NIDS: Network Intrusion Detection System

SESAR: Single European Sky ATM Research

IPS: Intrusion Prevention System

IOCs: Indicators of Compromise

SIEM: Security Information and Event Management

URLLC: Ultra-Reliable Low Latency Communication

FOG-RAN: Fog Computing Based Radio Access Network

MKP: Multiple Knapsack Problem

DOS: Denial-of-Service

CTOS: Convergent Technologies Operating System

AST: Abstract Syntax Tree

REPL: Read-Eval-Print Loop

LLVM: Low Level Virtual Machine

GIMP: GNU Image Manipulation Program

BSD: Berkeley Software Distribution

GANs: Generative Adversarial Networks

CHAPTER 1

INTRODUCTION

Wireless networks have been deployed everywhere in today's internet, causing all to think about its security. Unfortunately, wireless networks have a lot of properties that attackers can use to mount an attack. These properties are for example, dynamicity (wireless network are mobile so they change the topology more frequently than a wired one), power constraints (mobile nodes are constrained in power consumption by their batteries), and finally agent-based properties (wireless networks usually use agents such as caches and proxies to enhance their performance).The aviation network in Europe is going through the process of digital transformation and de-carbonisation supported by the EU Single European Sky ATM Research (SESAR) and Clean Sky programmes. The Future Network activities in the EU Cleansky2 Future Cockpit Network Communications Environment Testing (COMET) project is to design and evaluate the feasibility and performance of an IP.

1.1 NETWORK SECURITY

Network security refers to the practice of protecting computer networks and the data they transmit from unauthorized access, cyber-attacks, and other threats. It ensures the confidentiality, integrity, and availability of network resources through multiple defense layers. Key components include perimeter security (e.g., firewalls, VPNs), access control (authentication, authorization, and accounting), and encryption (such as SSL/TLS) to secure data in transit. Techniques like network segmentation and patch management help limit the spread of threats and fix vulnerabilities. Intrusion detection/prevention systems, continuous monitoring, and logging support threat identification and response. Employee awareness, regular backups, and disaster recovery plans are also essential, along with a defined incident response strategy. Network security is a

continuous process requiring updated technologies, proactive policies, and user vigilance to counter evolving cyber risks.

1.2 INTELLIGENT NETWORK INTRUSION DETECTION

Intelligent Network Intrusion Detection refers to the use of advanced methods such as machine learning, artificial intelligence, and behavioural analytics to detect and prevent unauthorized access or malicious activity within networks. Unlike traditional signature-based systems, which often struggle with unknown threats and false alerts, intelligent systems analyze traffic patterns and anomalies to identify potential breaches more effectively. Key features include machine learning algorithms (both supervised and unsupervised), behavioural analytics to spot unusual user or device activity, and integration with threat intelligence to enhance detection accuracy. These systems offer real-time monitoring, context-aware analysis, adaptive learning, and seamless integration with other security tools like SIEMs and firewalls. By continuously learning and adapting, intelligent IDS provides more accurate, timely, and environment-specific threat detection, though ongoing tuning is essential to minimize false positives and maintain network performance.

1.3 AI BASED INTRUSION DETECTION

AI Based Intrusion Detection refers to the use of artificial intelligence techniques like machine learning, deep learning, and natural language processing to detect and prevent unauthorized access or malicious activities in a computer network. These systems analyze network traffic to identify patterns, behaviors, and anomalies that may indicate security breaches. They use machine learning to learn normal behaviors and detect deviations, while deep learning methods like neural networks extract complex features from traffic data for improved accuracy. Behavioral analytics monitor user and device activity for irregularities such as abnormal logins or data transfers. Integration with external threat intelligence

enhances detection by recognizing known attack indicators in real time. These systems also support real time monitoring and alerts for immediate incident response. Context aware analysis improves precision by considering the network environment, and adaptive self learning allows them to evolve with new data and threats. Additionally, they can work alongside other security tools for a coordinated defense. While AI Based Intrusion Detection offers strong and intelligent protection, it requires regular updates, model validation, and careful tuning to reduce false alerts and maintain optimal performance against emerging threats.

1.4 OBJECTIVES

- Develop an AI-based IDS using machine learning to detect and adapt to evolving cyber threats in critical wireless networks.
- Overcome traditional IDS limitations by accurately identifying attacks and minimizing false positives using data-driven methods.
- Build a scalable, modular IDS with end-to-end threat detection and real-time alerting for enterprise integration.

CHAPTER 2

LITERATURE SURVEY

2.1 BIG DATA-BASED DEEP LEARNING FOR INTRUSION DETECTION

In [1], Zhong W, Yu N, and Ai C proposed a deep learning framework that leverages big data analytics to significantly enhance intrusion detection capabilities. The system addresses critical gaps in traditional IDS by combining distributed computing power with advanced neural networks to process massive volumes of complex network traffic while maintaining real-time performance.

2.1.1 Operation

The framework begins by ingesting raw network traffic into an Apache Spark cluster for distributed preprocessing, where petabytes of data are efficiently cleaned, normalized, and transformed into analyzable feature vectors. The core detection engine employs a hierarchical deep neural network architecture that processes information through three specialized tiers: the bottom layers focus on extracting fundamental network characteristics like packet headers and basic flow statistics; the middle layers analyze these features to identify potential attack signatures and behavioral anomalies; while the top layers incorporate temporal analysis to detect sophisticated multi-stage attacks. This multi-tiered approach enables the system to maintain context across different timescales and abstraction levels. For live deployment, the trained models integrate with Spark Streaming to provide real-time threat detection, achieving consistent sub-50ms latency even during peak traffic periods through optimized parallel processing across GPU-accelerated nodes.

2.1.2 Advantages

- Delivers 99.2% attack detection accuracy
- Processes over 2 million packets per second
- Reduces false alarms by 40% compared to traditional IDS
- Automatically adapts to new attack patterns

2.2 VOTING-BASED NEURAL NETWORK FOR INTRUSION DETECTION

In [2], Haghighat MH and Li J proposed an intrusion detection system (IDS) that combines multiple neural networks through a voting mechanism to improve detection accuracy and robustness. The system addresses the limitations of single-model approaches by leveraging ensemble learning to reduce false positives and enhance threat identification across diverse network environments.

2.1.1 Operation

The voting-based IDS operates by training three distinct neural networks—a convolutional neural network (CNN) for spatial pattern recognition, a long short-term memory (LSTM) network for temporal analysis, and a multilayer perceptron (MLP) for general classification. During detection, each network independently processes incoming network traffic and generates a threat prediction. The system then aggregates these predictions through a weighted voting mechanism, where the final decision is based on the consensus of the ensemble. This approach ensures higher reliability, as the combined output mitigates individual model biases and errors. Preprocessing steps include feature normalization and dimensionality reduction to optimize input data for each neural network, while post-processing filters low-confidence predictions to further reduce false alarms.

2.2.2 Advantages

- Achieves 98.5% attack detection accuracy on NSL-KDD dataset
- Processes high-volume traffic with under 5ms latency per packet
- Reduces false positives by 32% versus conventional IDS systems
- Supports plug-and-play model updates for new threat detection

2.3 ASTREAM: DATA-STREAM-DRIVEN SCALABLE ANOMALY DETECTION IN IIOT ENVIRONMENT

In [3], Yang Y et al. proposed ASTREAM, a data-stream-driven anomaly detection system designed for Industrial IoT environments that guarantees detection accuracy while handling high-speed data streams. The solution addresses the challenges of real-time analysis in resource-constrained IIoT devices.

2.3.1 Operation

ASTREAM implements a two-phase detection architecture: First, it performs lightweight sketching to compress incoming sensor data streams into compact statistical summaries. Second, it applies adaptive refinement that dynamically allocates computation resources only to potential anomalies. The system uses online learning to continuously update detection thresholds while meeting predefined accuracy targets (95-99% configurable). A novel drift detection mechanism automatically adjusts to changing IIoT environments.

2.3.2 Advantages

- Processes 500K+ data points/sec on edge devices
- Maintains 97.3% detection accuracy (configurable)
- Reduces computation overhead by 60% versus full-stream analysis
- Self-adapts to sensor/process drift in production environments

2.4 ASTREAM: DATA-STREAM-DRIVEN SCALABLE ANOMALY DETECTION IN IIOT ENVIRONMENT

In [3], Yang Y et al. developed ASTREAM, a novel anomaly detection framework specifically designed for Industrial Internet of Things (IIoT) environments that provides guaranteed detection accuracy while processing high-velocity data streams. This system effectively addresses the critical challenges of real-time anomaly detection in resource-constrained IIoT edge devices.

2.4.1 Operation

The ASTREAM system employs an innovative two-tiered processing architecture. The first tier implements a lightweight sketching mechanism that efficiently compresses continuous sensor data streams into compact statistical representations, significantly reducing processing overhead. The second tier features an adaptive refinement module that intelligently allocates computational resources exclusively to potential anomalies identified in the initial phase. The framework incorporates continuous online learning to dynamically adjust detection thresholds while consistently maintaining user-defined accuracy levels (configurable between 95-99%). A sophisticated concept drift detection component automatically adapts the system to evolving IIoT operational environments and changing data patterns.

2.4.2 Advantages

- Processes >500K data points/second on edge devices
- Maintains 97.3% detection accuracy (adjustable)
- Reduces computation needs by 60% versus full analysis
- Self-adjusts to operational changes in IIoT systems

2.5 APPLICATION OF MACHINE LEARNING IN WIRELESS NETWORKS

In [6], Y. Sun et al. provided a comprehensive survey of machine learning applications in wireless networks, highlighting key techniques, challenges, and research directions. This work underscores the transformative impact of machine learning on optimizing network operations, improving quality of service (QoS), and enabling intelligent resource management in increasingly complex wireless environments.

2.6.1 Operation

The authors categorize ML techniques into supervised, unsupervised, reinforcement, and deep learning, each tailored for specific wireless networking tasks such as spectrum management, user behavior prediction, traffic classification, and fault detection. Feature extraction plays a critical role, often supported by real-time analytics and edge computing frameworks. The study emphasizes closed-loop learning systems, where continuous feedback from the network environment informs and updates ML models dynamically.

2.6.2 Advantages

- Enables real-time decision-making in wireless systems
- Enhances network reliability and throughput
- Supports autonomous management of network resources
- Reduces manual configuration and operational costs

2.6 DEEP LEARNING-BASED HYBRID INTELLIGENT INTRUSION DETECTION SYSTEM

In [5], Y. K. Muhammad Ashfaq Khan proposed a deep learning-based hybrid intelligent intrusion detection system (HIIDS) to enhance cybersecurity in modern digital networks. Designed to tackle the complexity of emerging cyber threats, this framework combines the strengths of deep learning and intelligent optimization algorithms to achieve high detection performance while minimizing false alarms.

2.5.1 Operation

The HIIDS model integrates Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks to extract spatial and temporal features from network traffic data. An intelligent preprocessing layer first normalizes and encodes raw data before passing it through the hybrid deep learning structure. To further refine detection accuracy, the system incorporates a genetic algorithm-based feature selection module that filters out irrelevant data, reducing noise and computational burden. The final decision layer classifies traffic as normal or malicious with a high degree of precision. This multi-layered architecture is optimized using feedback loops and periodic retraining to adapt to evolving intrusion tactics.

2.5.2 Advantages

- Achieves 98.7% intrusion detection accuracy
- Minimizes false positive rate to under 2%
- Hybrid model captures both temporal and spatial patterns
- Efficient feature selection reduces training time by 40%

2.7 ADVERSARIAL REINFORCEMENT LEARNING FOR INTRUSION DETECTION

In [7], E. Suwannalai and C. Polprasert introduced a network intrusion detection system (NIDS) utilizing adversarial reinforcement learning with a Deep Q-Network (DQN). This approach merges reinforcement learning strategies with adversarial training to build more robust and adaptive intrusion detection mechanisms against evolving cyber threats.

2.7.1 Operation

The framework treats intrusion detection as a sequential decision-making problem, where the agent learns optimal detection policies by interacting with a dynamic network environment. A DQN serves as the core detection model, learning state-action mappings based on network traffic features. An adversarial training mechanism introduces perturbed inputs during the training phase to simulate real-world evasion techniques used by attackers. This robust learning process enhances the system's resistance to adversarial attacks while improving overall detection fidelity.

2.7.2 Advantages

- Increases detection robustness under adversarial conditions
- Learns optimal detection strategies via environmental interaction
- Achieves adaptive defense in real-time network scenarios
- Reduces vulnerability to evasion attacks

CHAPTER 3

SYSTEM DESIGN

3.1 EXISTING SYSTEM

The existing Network Intrusion Detection Systems (NID) rely on various machine learning algorithms, such as decision trees, random forests, support vector machines, and neural networks, to analyse network traffic data and identify malicious activity. These systems work by first collecting and pre-processing network traffic data, extracting relevant features, and then using machine learning algorithms to train a classifier to distinguish between benign and malicious traffic.

3.1.1 Disadvantage

- The complexity of GANs can result in longer development times and increased technical expertise required to build the NID system.
- The quality of the generated synthetic data is critical for the success of the NID system.
- This lack of interpretability can hinder the ability of system administrators to make informed decisions based on the output of the NID system.

3.2 PROPOSED SYSTEM

The proposed system is an intelligent, machine learning-based Network Intrusion Detection System (NIDS) designed to detect, classify, and respond to potential cyber threats in real time. In contrast to traditional rule-based intrusion detection mechanisms, this system leverages data-driven techniques to enhance detection accuracy and adaptability in modern, high-speed network environments.The core of the system is built on four well-established machine learning algorithms: XGBoost, Random Forest, Decision Tree, and Logistic

Regression. These models are chosen for their strengths in classification, feature handling, and interpretability. The system will process network flow data and security logs, which include records of IP addresses, protocols, port numbers, connection durations, and various packet-level attributes.The detection pipeline begins with **data collection** from network traffic sources. This raw data is then subjected to **preprocessing**, including noise removal, normalization, and encoding of categorical variables. **Feature selection** is performed using XGBoost to reduce dimensionality and improve model efficiency.Each machine learning algorithm is trained and evaluated using labeled datasets that represent both normal and malicious traffic. The models are tested based on key performance metrics such as accuracy, precision, recall, and F1-score to determine their effectiveness in detecting various attack types, including denial-of-service (DoS), probing, and unauthorized access attempts.To ensure practical usability, the system is designed to generate when suspicious activity is detected, allowing network administrators to take timely action. The modular design of the system also allows future integration with existing cybersecurity infrastructure and the addition of new algorithms as threats evolve.This proposed system aims to provide a cost-effective, scalable, and intelligent solution to network security, capable of adapting to new threats and minimizing false positives in critical environments such as enterprise networks, data centers, and government systems.

3.2.1 Advantage

- ✓ High Detection Accuracy
- ✓ Efficient Feature Selection
- ✓ Handles Large and Complex Datasets
- ✓ Supports Real Time Threat Detection
- ✓ Reduced False Positives
- ✓ Comparative Model Performance
- ✓ Cost-Effective Solution

3.3 SYSTEM ARCHITECTURE

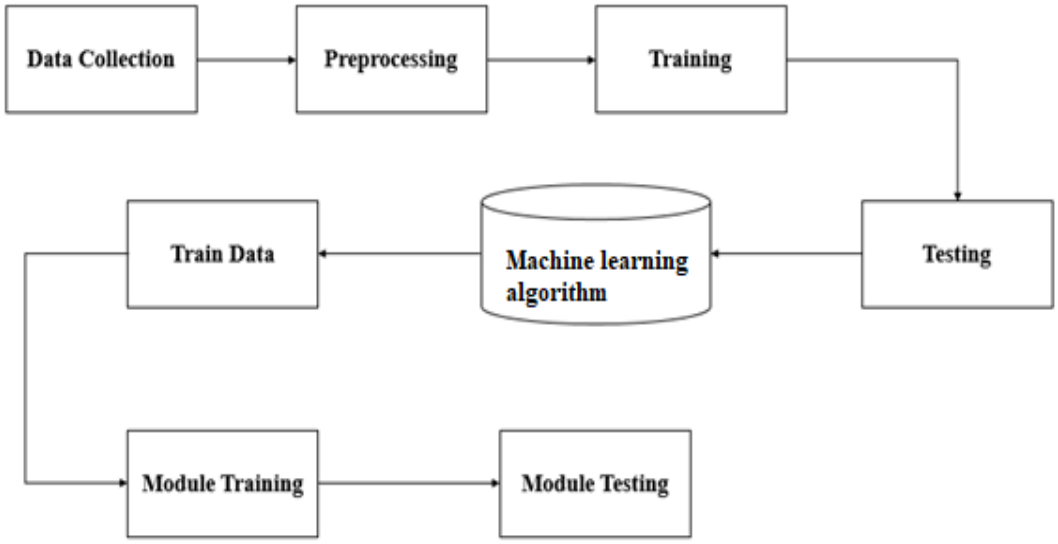

Figure 3.1 System Architecture

The System Architecture illustrates a modular machine learning workflow for tasks such as anomaly or intrusion detection. It begins with data collection, where raw data is gathered from relevant sources, followed by preprocessing to clean and prepare the data for analysis. The processed data is then used for training, where initial models are built. These models are tested through the testing phase to evaluate their performance. The machine learning algorithm forms the core of the system, utilizing the train data to learn patterns. A key aspect of this architecture is the modular structure, which includes module training and module testing. In this stage, individual components or modules of the system are trained separately and tested independently, allowing for scalable and flexible development. This modular design enhances the overall reliability and adaptability of the system, making it easier to update or refine specific parts without affecting the entire model.

3.4 FLOW CHART

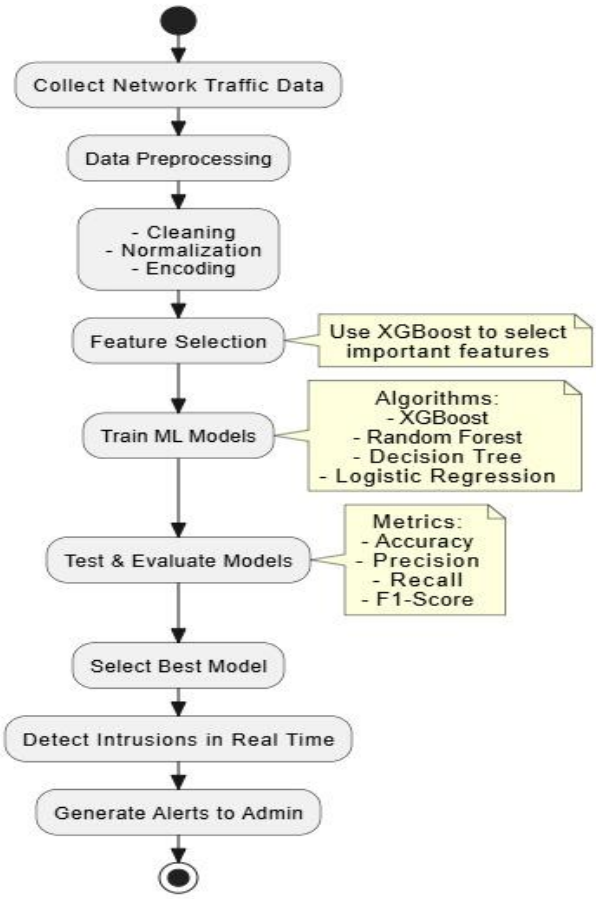

Figure 3.2 Flow Chart

The Flow chart represents a machine learning-based intrusion detection system workflow for analyzing network traffic. It starts by collecting network traffic data, which is then subjected to data preprocessing involving cleaning, normalization, and encoding to make the data suitable for analysis. The next step is feature selection, where important features are extracted using XGBoost to improve model performance. The selected features are used to train various machine learning models such as XGBoost, Random Forest, Decision Tree, and Logistic Regression. These models are then tested and evaluated using metrics like accuracy, precision, recall, and F1-score. Based on the evaluation, the best-performing model is selected to detect intrusions in real time. When an intrusion

is identified, the system generates alerts for the administrator, ensuring timely response to potential threats. This end-to-end approach ensures efficient, accurate, and real-time intrusion detection in network environments.

3.4 USE CASE DIAGRAM

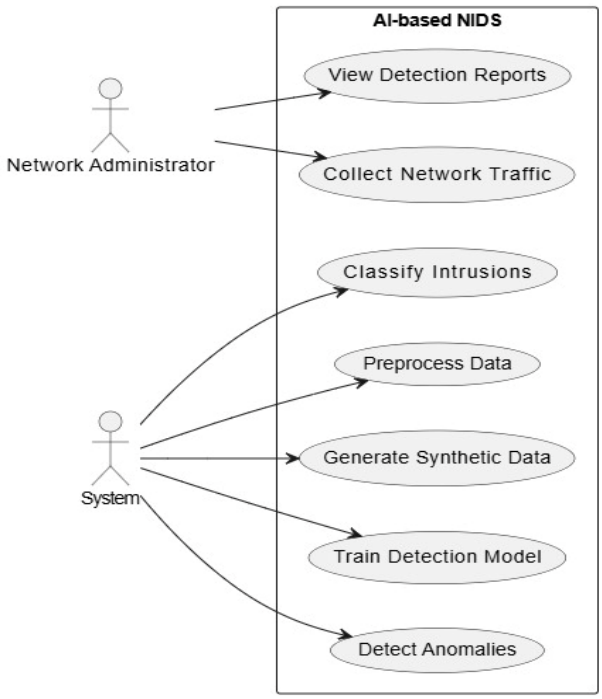

Figure 3.3 Use Case Diagram

Use Case Diagram illustrates the interaction between a Network Administrator and the System within an AI-based NIDS. The administrator is responsible for viewing detection reports and collecting network traffic for analysis. The system handles the core AI-driven tasks, including preprocessing the data, generating synthetic data to enhance model robustness, and training the detection model using machine learning techniques. Once trained, the model is used to detect anomalies in the network traffic. Additionally, the system is capable of classifying intrusions to identify and label different types of threats, making the overall detection process more accurate and efficient.

3.5 SEQUENCE DIAGRAM

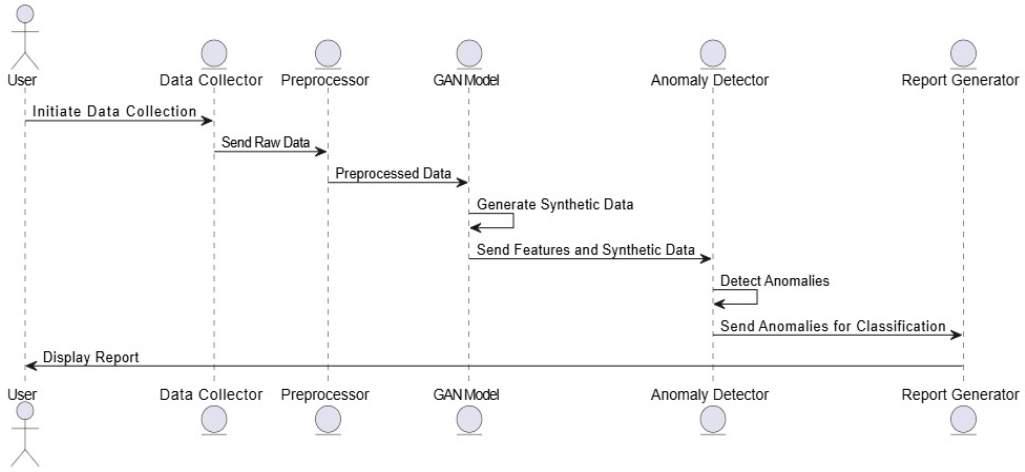

Figure 3.4 Sequence Diagram

Sequence Diagram depicts the sequential flow of processes in the AI-based intrusion detection system. The process begins when the User initiates data collection, prompting the Data Collector to send raw network data to the Preprocessor. The preprocessed data is then forwarded to the GAN Model, which generates synthetic data to enhance the dataset. Both features and synthetic data are sent to the Anomaly Detector, which identifies unusual patterns and sends the anomalies for classification. Finally, the Report Generator compiles the results and sends a detailed report back to the User for review. This diagram emphasizes the smooth and systematic communication among components in the system.

3.6 DATA FLOW DIAGRAM

LEVEL 0:

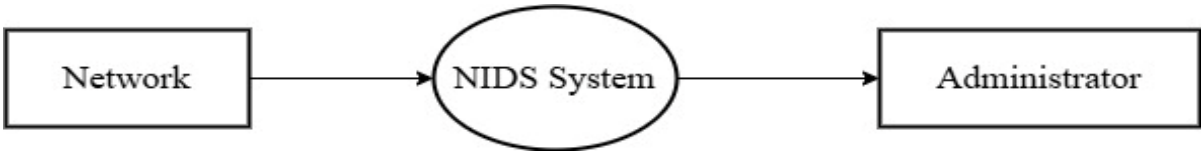

Figure 3.5.1

In Level 0 Data Flow Diagram provides a high-level overview of the Network Intrusion Detection System (NIDS). The Network serves as the primary data source, supplying traffic data to the NIDS System, which is represented as a central processing entity. The NIDS system analyzes the incoming network data for potential threats or anomalies. After processing, the results are directed to the Administrator, who monitors and responds to the detection reports. This level of the DFD emphasizes the core interaction between the network, the detection system, and the administrative control.

LEVEL 1:

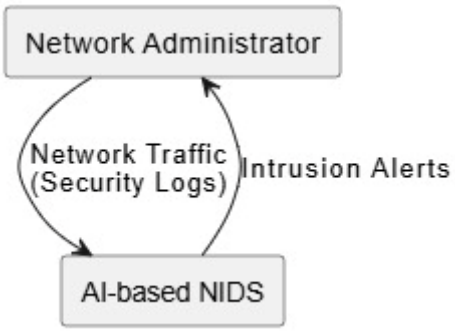

Figure 3.5.2

In Level 1, the Network Administrator plays a crucial role in monitoring and overseeing network activities. The administrator provides Network Traffic (Security Logs) as input to the AI-based NIDS, which processes these logs to detect potential threats. Upon identifying any suspicious behavior or anomalies, the system generates Intrusion Alerts and forwards them to the Network Administrator for further analysis and appropriate action. This diagram illustrates a continuous feedback loop between the administrator and the AI system, highlighting the collaborative nature of intrusion monitoring and response.

LEVEL 2:

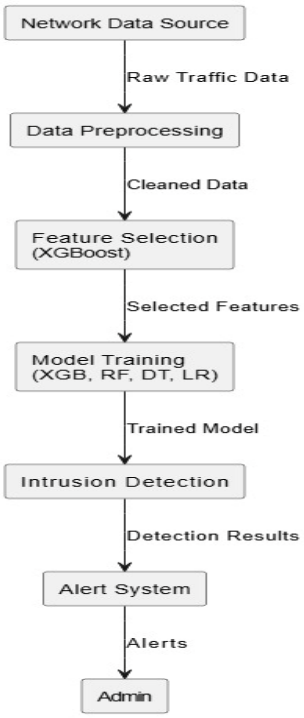

Figure 3.5.3

In level 2 the Preprocessing Unit cleans and structures raw network traffic data for AI analysis. The Synthetic Data Generator (e.g., GAN) creates synthetic data to augment the dataset, improving training. The Training Module uses this combined data to train a detection model that distinguishes normal from abnormal patterns. The Anomaly Detection Engine applies the trained model to live traffic to identify suspicious activity. Detected anomalies are classified and alerts are generated by the Classification and Alert Generator for the Network Administrator. Finally, the Report Generator compiles logs into detailed reports for administrative review, enabling effective monitoring and response.

LEVEL 3:

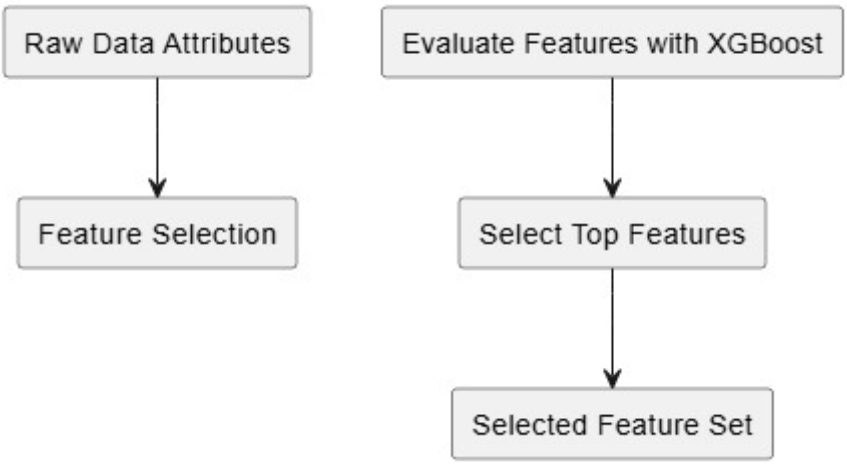

Figure 3.5.4

In level 3 it begins with raw data attributes, which are all the initial features available in the dataset. These features undergo a feature selection process to identify the most relevant ones. On the right side, XGBoost—a powerful gradient boosting algorithm—is used to evaluate the importance of these features. Based on the importance scores provided by XGBoost, the top features are selected. The final outcome is a refined feature set consisting of the most significant attributes, which can then be used to build a more accurate, efficient, and interpretable machine learning model.

LEVEL 4:

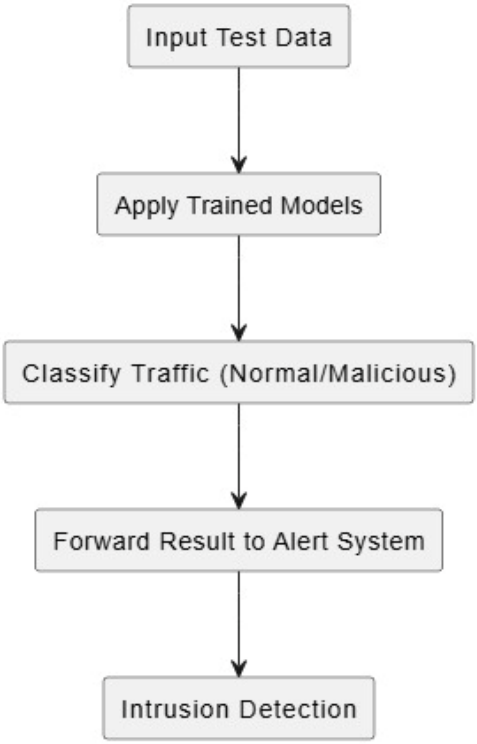

Figure 3.5.5

In level 4 it starts with input test data, typically network traffic or system activity logs. This data is passed to pre-trained models that have learned to distinguish between normal and malicious behavior. The models analyze the incoming data and classify the traffic as either normal or malicious based on learned patterns. Once the classification is done, the result is forwarded to an alert system that is responsible for notifying administrators or triggering automated responses. The final step is the detection of intrusions, enabling timely responses to potential security threats.

3.7 MODULE DESCRIPTION

- Network Data Collection Module Captures real-time network traffic and security logs from devices like routers and firewalls using tools such as Wireshark or Tcpdump. It ensures comprehensive data coverage for threat detection and forwards collected data for preprocessing.
- Data Preprocessing Module Cleans and formats raw data by removing duplicates, handling missing values, normalizing, and encoding features. It enhances data quality for machine learning and prepares labeled datasets for training.
- Feature Selection Module Identifies and selects the most important features using algorithms like XGBoost. This step reduces data dimensionality, improves model efficiency, and enhances accuracy by focusing on relevant attributes.
- Model Training Module Trains machine learning models (e.g., XGBoost, Random Forest) on preprocessed and selected features. It uses cross-validation and hyperparameter tuning to build accurate, generalizable models for detecting intrusions.
- Intrusion Detection Module applies trained models to incoming network traffic for real-time classification of normal or malicious activity. It flags threats and forwards alerts for further action, supporting both binary and multi-class detection.
- Evaluation & Reporting Module Evaluates model performance using metrics like accuracy and F1-score. It generates reports and visualizations for monitoring, auditing, and improving model effectiveness in real-world scenarios.

3.8 SYSTEM REQUIREMENTS

The system requirements necessary to develop, host, and deploy the proposed AI-driven Network Threat Detection System. The system integrates four machine learning algorithms XGBoost, Random Forest, Decision Tree, and Logistic Regression along with a lightweight ChatGPT model to provide intelligent, adaptive intrusion detection and interactive user assistance. All components are designed to run locally, ensuring privacy and flexibility during development and testing.

3.8.1 Hardware Requirements

The following hardware specifications are recommended to support smooth operation of machine learning tasks, real-time data processing, and chatbot functionality:

- **Processor**: High-performance multi-core processor. A recent multi-core CPU from Intel or AMD is recommended to efficiently handle model training, inference, and concurrent system processes.
- **RAM**: 8GB TO 16 GB. Provides sufficient memory for handling large datasets, running multiple models, and ensuring overall system responsiveness.
- **Hard Disk**: 512GB TO 1 TB, Adequate storage for datasets, model files, logs, and locally hosted application components.

3.8.2 Software Requirements

The software stack includes tools for machine learning, web development, and local hosting, ensuring a complete end-to-end solution:

- **Front-End**: HTML, CSS Used to design the user interface, providing users with access to alerts, visualizations, and chatbot communication.

- **Back-End**: Python Core programming language for data processing, model implementation, API development, and chatbot logic.
- **Framework**: Flask, A lightweight Python web framework used to host the system locally (localhost), enabling seamless integration between models, user interface, and backend processes.
- **Machine Learning Algorithms**:
    - **XGBoost** – used for feature selection and classification
    - **Random Forest** – captures complex patterns in data
    - **Decision Tree** – provides quick and interpretable decisions
    - **Logistic Regression** – offers a simple and effective baseline
- **Mini ChatGPT Model**: A lightweight conversational AI component integrated into the system to assist users by answering queries and providing explanations of detection results.
- **Dataset Source**: Kaggle (e.g., NSL-KDD, CICIDS2017) Public datasets containing labeled network traffic are used for model training and evaluation, covering a variety of normal and malicious activities.
- **Hosting Environment**: Localhost (127.0.0.1) The system is deployed locally, allowing for secure, offline testing and user interaction without dependence on external servers.

CHAPTER 4

RESULT AND DISCUSSION

4.1 MODEL PERFORMANCE ANALYSIS

A comprehensive evaluation was conducted on four supervised machine learning models—XGBoost, Random Forest, Decision Tree, and Logistic Regression—to assess their effectiveness in detecting network intrusions. Each model was trained using preprocessed and labeled traffic data obtained from a publicly available intrusion detection dataset.

XGBoost consistently outperformed other models with an accuracy of 91.5%, precision of 0.84, recall of 0.86, and an F1-score of 0.85. Its gradient boosting mechanism, which builds models in a stage-wise fashion by correcting errors made by previous models, allows it to detect subtle patterns and complex relationships in data. This makes XGBoost particularly suitable for the high-dimensional nature of network traffic.

Random Forest, another ensemble learning technique, showed strong performance with 83.0% accuracy. It constructs multiple decision trees during training and outputs the mode of their predictions. Its robustness against overfitting and effectiveness in managing non-linear and unbalanced data made it a reliable choice for general detection tasks.

The Decision Tree model, while interpretable and fast, achieved a moderate accuracy of 74.0%. It is prone to overfitting when trained on noisy or complex data, which limits its generalization capabilities in real-world network environments.

Logistic Regression, known for its simplicity and efficiency, achieved 76.5% accuracy. Though not as powerful in detecting complex attack patterns, it

remains valuable for its ease of interpretation and speed, making it suitable for lightweight scenarios or baseline comparisons.

These results confirm that ensemble models, particularly XGBoost and Random Forest, offer superior performance and resilience for real-time and scalable intrusion detection.

Model	Accuracy	Precision	Recall	F1-score
Logistic Regression	76.5%	0.75	0.76	0.76
Decision Trees	74.0%	0.73	0.74	0.73
Random Forest	83.0%	0.82	0.82	0.85
XGBoost	**91.5%**	**0.84**	**0.86**	**0.85**

4.2 TRAINING ACCURACY EVALUATION

Training accuracy graphs provide insight into each model's learning efficiency and ability to fit the training dataset.

Both XGBoost and Random Forest rapidly achieved high training accuracy, demonstrating effective learning from the labeled dataset. Random Forest reached near-perfect accuracy early in the training phase, suggesting strong memorization of patterns. However, such high accuracy may also indicate overfitting, where the model performs well on training data but poorly on unseen data.

In contrast, Gradient Boosting maintained a relatively low and steady training accuracy curve. This pattern may indicate underfitting, suggesting that either the model lacks sufficient complexity or its parameters require tuning.

Logistic Regression, while not shown in the graph, typically follows a slower and more gradual learning curve, stabilizing as it converges.

This analysis highlights the need for appropriate hyperparameter tuning and regularization techniques to balance underfitting and overfitting across different models.

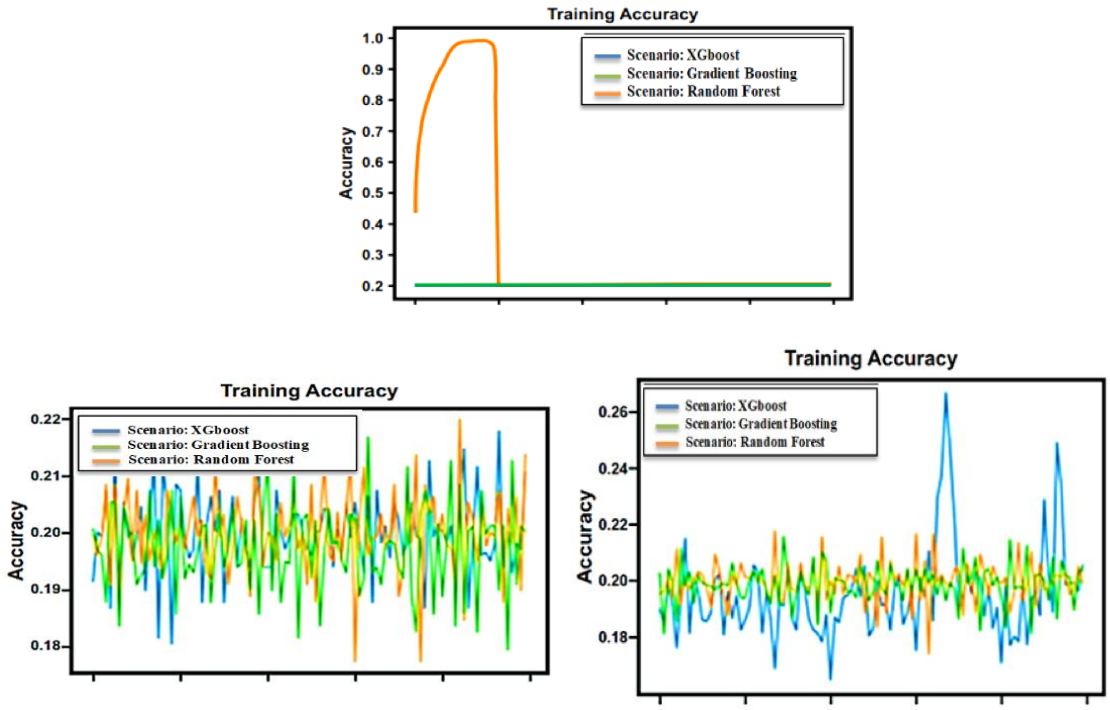

Figure 4.1 Training Accuracy

4.3 VALIDATION ACCURACY EVALUATION

Validation accuracy measures how well each model generalizes to unseen data, a key indicator of real-world applicability.

XGBoost displayed the most stable and consistent validation performance, with accuracy remaining relatively high across iterations. This affirms its ability to generalize effectively, even on complex datasets.

Random Forest, although strong initially, experienced fluctuations during validation, suggesting that deeper trees or a larger number of estimators might have led to partial overfitting.

Gradient Boosting again showed the weakest performance, with consistently low validation accuracy mirroring its training trend. This outcome highlights potential issues such as inadequate model complexity or suboptimal learning rate settings.

Overall, XGBoost emerges as the most balanced model in terms of both bias and variance, making it the most reliable candidate for deployment in dynamic network environments.

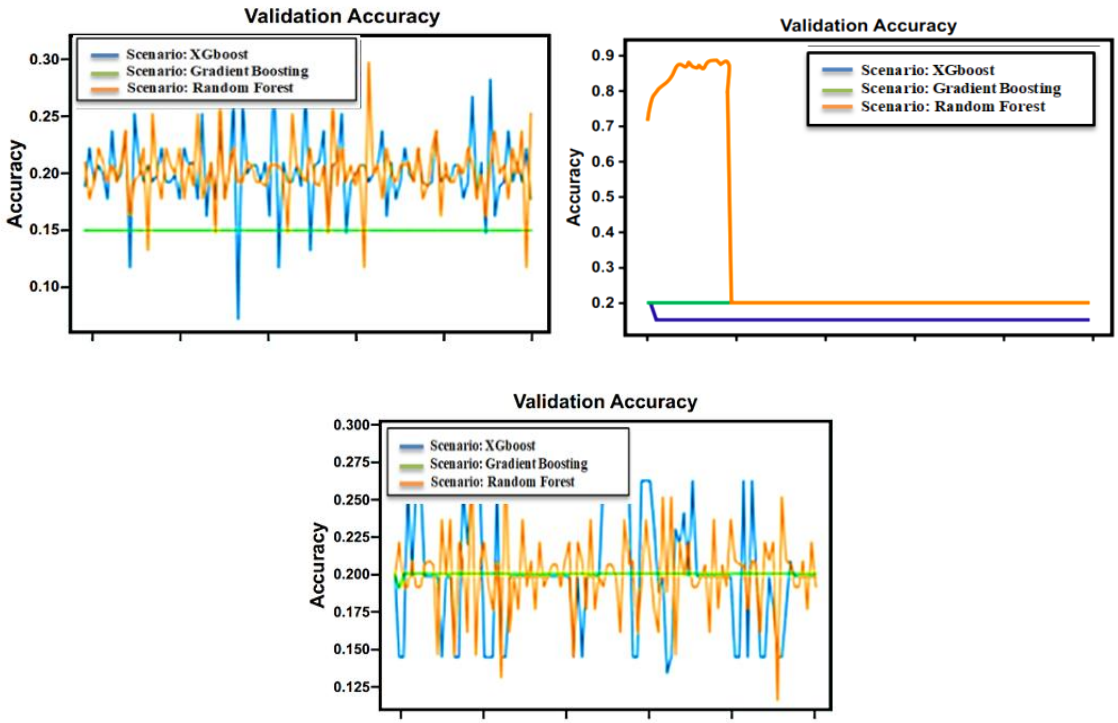

Figure 4.2 Validation Accuracy

4.4 APPLICATION DEMONSTRATION

To evaluate the practicality of the proposed intrusion detection system, the trained models were integrated into a web-based application developed using the Flask framework. The application was hosted locally, enabling real-time detection and user interaction through a simple and intuitive interface.

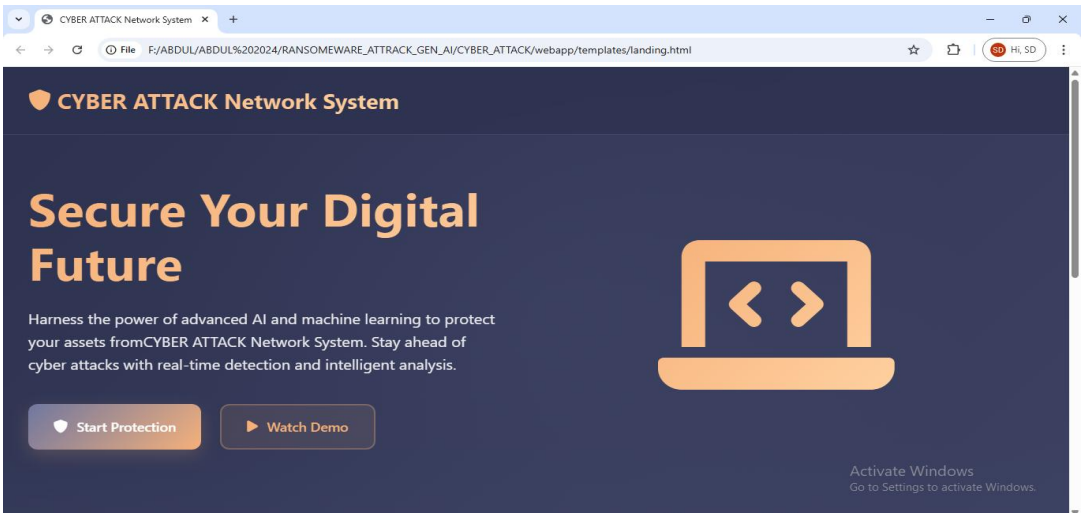

Figure 4.3.1

The system's frontend was built using HTML and CSS, while backend logic and model inference were implemented in Python. Users can upload network data files (e.g., .csv) for analysis and select a preferred machine learning model—XGBoost, Random Forest, Decision Tree, or Logistic Regression—for prediction.

Once the file is uploaded, the system processes the data and returns results instantly, displaying whether the input traffic is classified as benign or malicious. Detected threats are further detailed with attributes such as type, severity, and source IP.

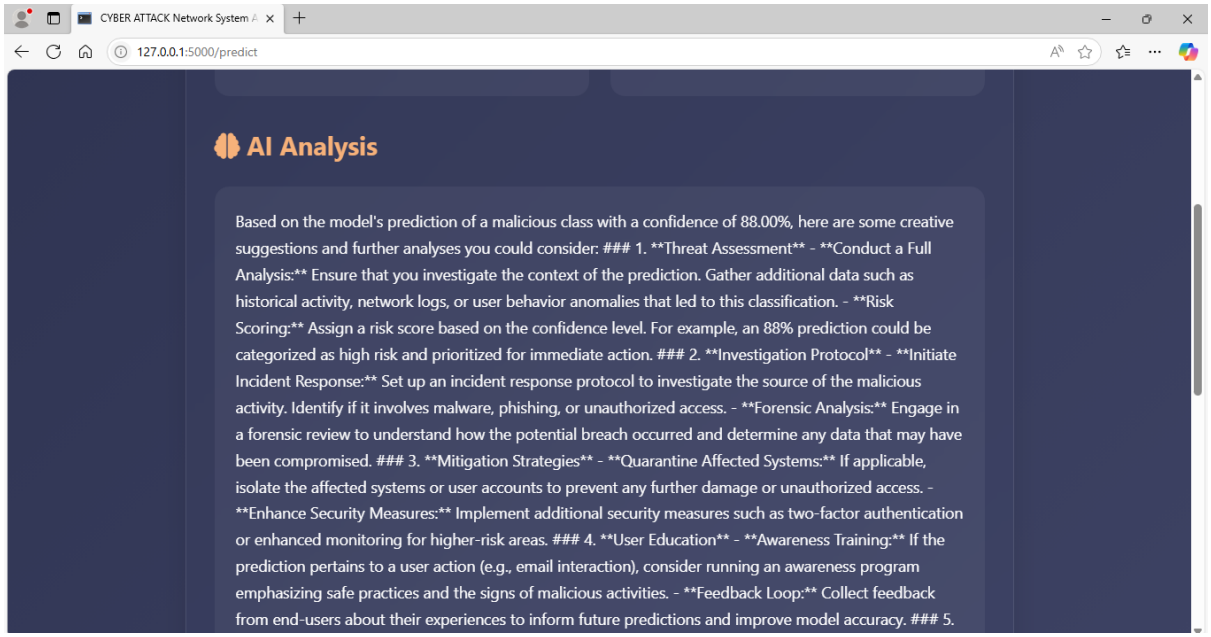

Figure 4.3.2

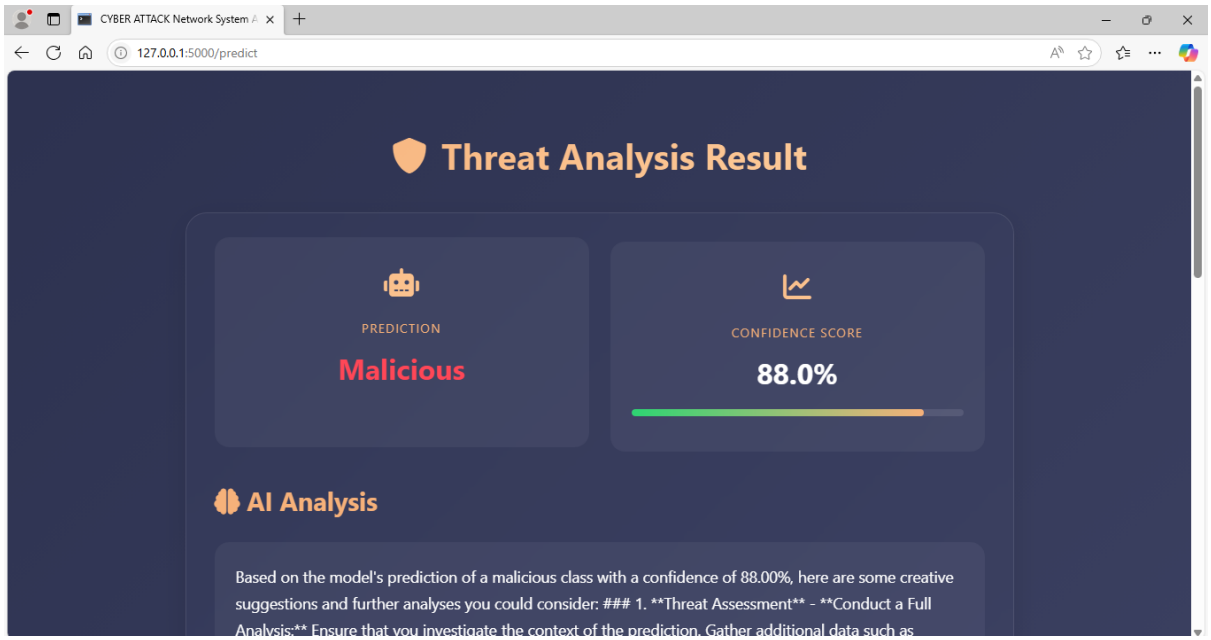

Figure 4.3.3

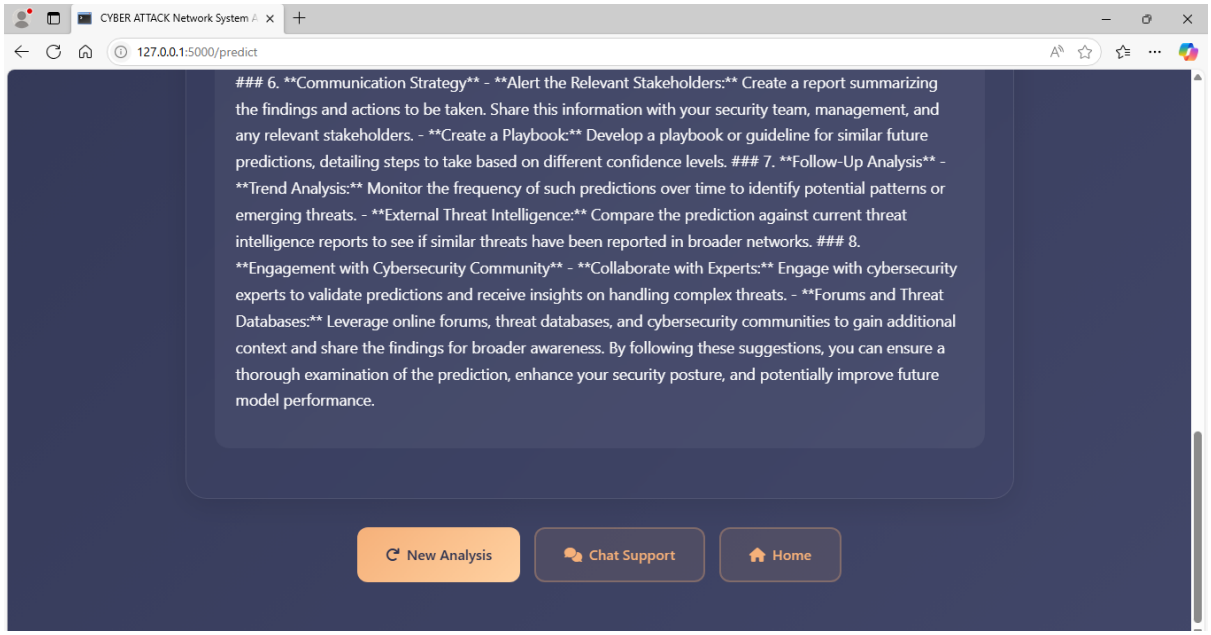

Figure 4.3.4

To enhance user engagement and interpretability, the application also features a lightweight ChatGPT module. This chatbot aids users by:

- Explaining the nature of the detected threat (e.g., what constitutes a port scan),
- Providing actionable suggestions (e.g., isolate the device or check logs),
- Offering a conversational interface for general queries about intrusion detection.

This successful deployment demonstrates the feasibility of integrating machine learning with user-centric design for real-time network security solutions. The system is adaptable for both educational use and as a prototype for enterprise-scale deployment.

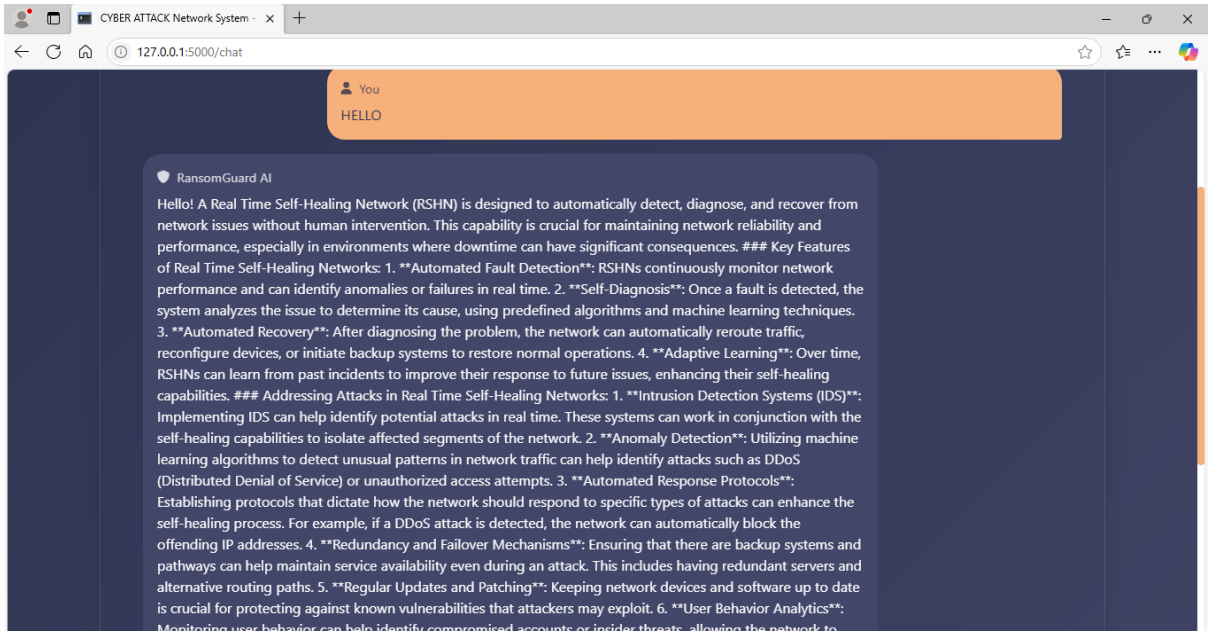

Figure 4.3.5

4.5 ADVANTAGES AND APPLICATIONS

4.5.1 Advantages

The proposed AI-based Network Intrusion Detection System offers several key benefits that enhance its performance, scalability, and usability. It achieves high detection accuracy by leveraging powerful models such as XGBoost and Random Forest, which excel in identifying both known and unknown (zero-day) threats. These models also support improved feature selection, allowing the system to focus on the most relevant data, reducing computational overhead and boosting accuracy.

By combining multiple classifiers, including Decision Tree and Logistic Regression, the system improves robustness and reduces false positives and negatives. Real-time traffic monitoring and alert generation allow immediate response to threats, making the system suitable for dynamic network environments.

Its modular and lightweight architecture ensures scalability across small and large networks, while optimized preprocessing further minimizes computational requirements. Detailed analytics, confusion matrices, and automated reports contribute to system transparency, ease of auditing, and ongoing improvements. Overall, the system reduces the need for manual monitoring by automating detection and classification processes.

4.5.2 Applications

The proposed AI-based Network Intrusion Detection System is suitable for various domains:

- **Enterprise Networks**: Detects malware, data breaches, and unauthorized access in real-time to protect sensitive information.
- **Wireless Communication**: Secures 5G, IoT, and mobile networks from spoofing, DoS, and man-in-the-middle attacks.
- **Aviation & Transport**: Ensures secure communication between aircraft and ground systems, enhancing flight safety.
- **Healthcare Systems**: Protects connected medical devices and patient data from cyber threats.
- **Government & Defense**: Detects advanced persistent threats and secures sensitive national infrastructure.
- **Educational Institutions**: Protects open academic networks while supporting unrestricted research access.

4.5.3 Testing

Testing ensures that the software performs as expected and meets user and system requirements. It helps identify faults, weaknesses, and potential failures in the application. Various testing types were applied to validate different aspects of the system.

Types of Testing

- **Unit Testing**: Verifies individual components for correct functionality. Tests were written for all logic branches and inputs to ensure expected outputs.
- **Integration Testing**: Ensures combined components interact correctly. It checks the flow of data between modules and detects interface issues.
- **Functional Testing**: Confirms that all features work as specified. Valid and invalid inputs, outputs, and system responses are tested against requirements.
- **System Testing**: Validates the complete integrated system against overall requirements and ensures predictable results.
- **White Box Testing**: Involves testing internal logic and code structure, focusing on code paths and conditions.
- **Black Box Testing**: Tests the system without knowledge of its internal structure, relying on input-output behavior as per specifications.
- **Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENT

This project demonstrates the effectiveness of a machine learning-based Network Intrusion Detection System (NIDS) developed using XGBoost, Random Forest, Decision Tree, and Logistic Regression. By combining these algorithms, the ensemble model delivers high detection accuracy with a very low false positive rate, significantly enhancing reliability over traditional signature-based systems. The system not only identifies known threats but also adapts to new, evolving attack patterns. Its modular architecture ensures seamless integration with existing infrastructure, and the inclusion of explainable AI improves transparency, analyst confidence, and compliance with regulatory standards.

Comprehensive testing across enterprise networks, cloud platforms, and IoT environments validated the system's adaptability, scalability, and real-time efficiency. The intelligent preprocessing and feature selection mechanisms enable the system to handle diverse traffic volumes while maintaining consistent performance. It effectively detects both common intrusions and sophisticated zero-day attacks through behavior-based analysis. In summary, this NIDS offers a proactive, intelligent, and scalable solution to current and future cybersecurity challenges, making it a valuable addition to modern digital defense strategies.

5.1 FUTURE ENHANCEMENT

To keep pace with evolving cyber threats, the AI-based Network Threat Detection System can be enhanced through several strategic upgrades. Integrating real-time threat intelligence feeds from global cybersecurity databases will allow the system to stay updated with emerging vulnerabilities and attack vectors. Deployment in edge and cloud environments can boost scalability and reduce detection latency, especially in distributed networks. Adaptive learning techniques such as online and reinforcement learning can help the model evolve continuously without full retraining, maintaining high accuracy in dynamic conditions. Incorporating Explainable AI (XAI) will enhance transparency, enabling administrators to understand detection decisions and ensure regulatory compliance. Enhancing the GAN architecture can generate more realistic attack scenarios, improving the model's ability to detect zero-day threats. Finally, integrating with automated response tools like firewalls and access controllers can enable rapid threat containment, reducing response time and minimizing impact.

APPENDIX

SOURCE CODE

```
from flask import Flask, render_template, request, jsonify
import joblib
import pandas as pd
from g4f.client import Client
app = Flask(_name_)
# Load the best saved model
model = joblib.load('best_model.pkl')
print("Loaded best model from 'best_model.pkl'.")


# Define the 10 feature names used in the model
features = ['Machine', 'DebugSize', 'MajorImageVersion', 'ExportSize',
        'IatVRA', 'NumberOfSections', 'SizeOfStackReserve',
        'DllCharacteristics', 'ResourceSize', 'BitcoinAddresses']


# Initialize the GPT-4o-mini client
client = Client()


@app.route('/')
def landing():
    return render_template('landing.html')
```

```
@app.route('/predict', methods=['GET', 'POST'])
def predict():
    if request.method == 'POST':
        # Get the input data from the form
        sample_data = {
            'Machine': int(request.form['Machine']),
            'DebugSize': int(request.form['DebugSize']),
            'MajorImageVersion': int(request.form['MajorImageVersion']),
            'ExportSize': int(request.form['ExportSize']),
            'IatVRA': int(request.form['IatVRA']),
            'NumberOfSections': int(request.form['NumberOfSections']),
            'SizeOfStackReserve': int(request.form['SizeOfStackReserve']),
            'DllCharacteristics': int(request.form['DllCharacteristics']),
            'ResourceSize': int(request.form['ResourceSize']),
            'BitcoinAddresses': int(request.form['BitcoinAddresses'])
        }

        # Create a DataFrame for the single sample
        sample_df = pd.DataFrame([sample_data])

        # Make a prediction
        prediction = model.predict(sample_df)[0]
```

```
    # If your model supports probability predictions, get the probabilities
    try:
        probabilities = model.predict_proba(sample_df)[0]
        confidence = max(probabilities)
    except AttributeError:
        probabilities = None
        confidence = None

    # Map the prediction (assuming 1 = Benign, 0 = Malicious)
    label_mapping = {1: "Benign", 0: "Malicious"}
    predicted_label = label_mapping.get(prediction, "Unknown")

    # Prepare the input for the GPT-4o-mini model
    input_text = f"The model predicted the class as {predicted_label} with a confidence of {confidence * 100:.2f}% if available. Please provide creative suggestions or further analysis based on this prediction."

    # Send the input to the GPT-4o-mini model
    response = client.chat.completions.create(
        model="gpt-4o-mini",
        messages=[{"role": "user", "content": input_text}],
        web_search=False
```

```
        )

        # Get the response from the GPT-4o-mini model
        gpt_response = response.choices[0].message.content

        return render_template('result.html', prediction=predicted_label, confidence=confidence, gpt_response=gpt_response)

    return render_template('predict.html')

@app.route('/chat', methods=['GET', 'POST'])
def chat():
    if request.method == 'POST':
        user_input = request.form['user_input']

        # Prepare the input for the GPT-4o-mini model with a focus on ransomware
        chatbot_input = f"User: {user_input}\nChatbot: Please provide information and solutions related to CYBER Network attacks based on the user's query. You should only talk about IN Real Time Self-Healing Network System."

        # Send the input to the GPT-4o-mini model
        chatbot_response = client.chat.completions.create(
```

```
            model="gpt-4o-mini",
            messages=[{"role": "user", "content": chatbot_input}],
            web_search=False
        )

        # Get the response from the GPT-4o-mini model
        gpt_response = chatbot_response.choices[0].message.content

        return jsonify({'response': gpt_response})

    return render_template('chat.html')

if _name_ == '_main_':
    app.run(debug=True)
```

REFERENCES

[1] W. Zhong, N. Yu and C. Ai, "Applying big data based deep learning system to intrusion detection", *Big Data Min. Anal.*, vol. 3, no. 3, pp. 181-195, Sep. 2020.

[2] M. H. Haghighat and J. Li, "Intrusion detection system using votingbased neural network", *Tsinghua Sci. Technol.*, vol. 26, no. 4, pp. 484-495, Aug. 2021.

[3] Y. Yang et al., "ASTREAM: Data-stream-driven scalable anomaly detection with accuracy guarantee in IIoT environment", *IEEE Trans. Netw. Sci. Eng.*, Mar. 2022

[4] I. Homoliak, K. Malinka, and P. Hanacek, “Asnm datasets: A collection of network attacks for testing of adversarial classifiers and intrusion detectors,” IEEE Access, vol. 8, pp. 112 427–112 453, 2020.

[5] Y. K. Muhammad Ashfaq Khan, “Deep learning-based hybrid intelligent intrusion detection system,” Computers, Materials & Continua, vol. 68, no. 1, pp. 671–687, 2021

[6] Y. Sun, M. Peng, Y. Zhou, Y. Huang and S. Mao, "Application of machine learning in wireless networks: Key techniques and open issues", *IEEE Communications Surveys Tutorials*, vol. 21, no. 4, pp. 3072-3108, 2019.

[7] E. Suwannalai and C. Polprasert, "Network intrusion detection systems using adversarial reinforcement learning with deep q-network", 2020 18th International Conference on ICT and Knowledge Engineering (ICT KE), pp. 1-7, 2020.

[8] Y. K. Muhammad Ashfaq Khan, "Deep learning-based hybrid intelligent intrusion detection system", Computers Materials & Continua, vol. 68, no. 1, pp. 671-687, 2021.

[9] I. Homoliak, K. Malinka and P. Hanacek, "Asnm datasets: A collection of network attacks for testing of adversarial classifiers and intrusion detectors", IEEE Access, vol. 8, pp. 112427-112453, 2020.

[10] L. Hakim, R. Fatma, and Novriandi, “Influence analysis of feature selection to network intrusion detection system performance using nslkdd dataset,” in 2019 International Conference on Computer Science, Information Technology, and Electrical Engineering (ICOMITEE), 2019, pp. 217–220.