DDOS ATTACK CLASSIFICATION WITH HYPERPARAMETER TUNING AND ENCHANCED PREDICTION TECHNIQUE

A PROJECT REPORT

Submitted by

GRAYSON P [211421104081]
GIRITHARAN K [211421104075]
GANESHRAGAVAN S [211421104073]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

APRIL 2025

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"DDOS ATTACK CLASSIFICATION WITH HYPERPARAMETER TUNING AND ENHANCED PREDICTION TECHNIQUE"** is the bonafide work of **"GRAYSON P [211421104081], GIRITHARAN K [211421104075]** and **GANESHRAGAVAN S [2114104073\]"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. L. JABASHEELA, M.E. Ph.D.,	**Dr.A.HEMALATHADHEVI, M.C.A, M .Phil, M.B.A, M.E, Ph.D.,**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Department of Computer Science and Engineering, Panimalar Engineering College, Nazarathpet, Poonamallee, Chennai, 600 123.	Department of Computer Science and Engineering, Panimalar Engineering College, Nazarathpet, Poonamallee, Chennai, 600 123.

Certified that the above candidate(s) was examined in the End Semester Project Viva-Voce Examination held on ....

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We, **GRAYSON P [211421104081], GIRITHARAN K [211421104075] and GANESHRAGAVAN S [2114104073]** hereby declare that this project report titled "**DDOS ATTACK CLASSIFICATION WITH HYPERPARAMETER TUNING AND ENHANCED PREDICTION TECHNIQUE**", under the guidance of **Dr.A. HEMALATHADHEVI, (deg & INIT).,** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

GRAYSON P [211421104081]
GIRITHARAN K [211421104075]
GANESHRAGAVAN S [2114104073]

ACKNOWLEDGEMENT

Our profound gratitude is directed towards our esteemed Secretary and Correspondent, **Dr. P. CHINNADURAI, M.A., Ph.D.,** for his fervent encouragement. His inspirational support proved instrumental in galvanizing our efforts, ultimately contributing significantly to the successful completion of this project.

We want to express our deep gratitude to our Directors, **Tmt. C. VIJAYARAJESWARI, Dr. C. SAKTHI KUMAR, M.E., Ph.D.,** and **Dr. SARANYASREE SAKTHI KUMAR, B.E., M.B.A., Ph.D.,** for graciously affording us the essential resources and facilities for undertaking of this project.

Our gratitude is also extended to our Principal, **Dr. K. MANI, M.E., Ph.D.,** whose facilitation proved pivotal in the successful completion of this project.

We express our heartfelt thanks to **Dr. L. JABASHEELA, M.E., Ph.D.,** Head of the Department of Computer Science and Engineering, for granting the necessary facilities that contributed to the timely and successful completion of project.

We would like to express our sincere thanks to Project Coordinator and Guide **Dr. G. SENTHIL KUMAR , M.C.A, M.Phil, M.B.A, M.E, Ph.D., Professor** and all the faculty members of the Department of CSE for their unwavering support for the successful completion of the project.

GRAYSON P [211421104081]

GIRITHARAN K [211421104075]

GANESHRAGAVAN S [2114104073]

ABSTRACT

Data privacy is crucial in the financial sector to safeguard clients' sensitive information, prevent financial fraud, ensure regulatory compliance, and protect intellectual property. With the rise of internet usage and digital transactions, maintaining privacy has become increasingly challenging. Distributed Denial of Service (DDoS) attacks pose a significant threat to client privacy, necessitating effective detection and prevention measures. Machine Learning (ML) offers a promising approach for enhancing cyber-attack detection systems. This paper proposes a hierarchical ML-based hyperparameter optimization technique for classifying network intrusions. Utilizing the CICIDS dataset, which includes logs of various attacks, the proposed method involves preprocessing the data with min-max scaling and SMOTE. Feature selection is carried out to identify the most significant features. Classification is then performed using XGBoost, LGBM, CatBoost, Random Forest (RF), and Decision Tree (DT) algorithms. The models' performance is evaluated using recall, precision, accuracy, and F1-score metrics.

LIST OF FIGURES

S.No	FIGURES	PAGE NO
1	Architecture Diagram	12
2	Sequence Diagram	17
3	Use Case Diagram	18
4	Activity Diagram	19
5	Collaboration Diagram:	20

LIST OF ABBREVIATIONS

CNN		Convolutional Neural Network
LCNN		Lookup based Convolutional Neural Network
RNN	-	Recurrent Neural Network
DEX	-	Dalvik Executables
TCP	-	Transmission Control Protocol
IP	-	Internet Protocol
HTTP	-	Hyper Text Transfer Protocol
ADT	-	Android Development Tool

TABLE OF CONTENTS

CHAPTER NO.		TITLE	PAGE NO.
		ABSTRACT	**v**
		LIST OF FIGURES	**vii**
		LIST OF ABBREVIATIONS	**viii**
1.		**INTRODUCTION**	
	1.1	Problem Definition	
	1.2	Synopsis	
2.		**LITERATURE REVIEW**	
	2.1	Old System	
	2.2	New System	
	2.3	Difference	
	2.4	Advantage	
	2.5	Disadvantage	
3.		**THEORETICAL BACKGROUND**	
	3.1	Implementation Environment	
	3.2	System Architecture	
	3.3	Proposed Methodology	
		3.3.1 Database Design	
		3.3.2 Input Design(UI)	
		3.3.3 Module Design	

CHAPTER NO.	TITLE	PAGE NO.
4.	**SYSTEM IMPLEMENTATION**	
	4.1 Module 1 : DDoS Detection Algorithm	
	4.2 Module 2 : Machine Learning Based IDS	
5.	**SYSTEM IMPLEMENTATION**	
	5.1 Performance Parameters /Testing	
	5.2 Results and Discussion	
6.	**CONCLUSION & FUTURE WORK**	
	APPENDICES	
	A.1 SGD Goal	
	A.2 Source Code	
	A.3 Screen Shots	
	A.4 Plagarism Report	
	A.5 Paper Publication	
	REFERENCES	

CHAPTER 1
INTRODUCTION

1.1 PROBLEM DEFINITION

The escalating threat landscape in today's digital age is characterized by an unprecedented surge in the sophistication, frequency, and diversity of cyberattacks, posing a critical challenge to the security and resilience of networks worldwide. Traditional rule-based Intrusion Detection Systems (IDS), which rely on predefined signatures and static patterns of known malicious activities, are increasingly struggling to effectively identify and mitigate these evolving threats. Their inherent limitations in detecting novel, zero-day exploits and sophisticated attack campaigns that deviate from established behaviors result in a significant gap in security coverage. Furthermore, these legacy systems are often plagued by a high rate of false positives, generating numerous inaccurate alerts that can overwhelm security analysts, lead to alert fatigue, and potentially obscure genuine security incidents. The manual effort required to triage and investigate these spurious alarms consumes valuable resources and detracts from proactive security measures, hindering an organization's ability to maintain a robust security posture. While Machine Learning (ML)-based Intrusion Detection Systems offer a promising paradigm shift by leveraging algorithms capable of learning complex patterns from network traffic and identifying anomalous behaviors indicative of malicious activity, their successful deployment is not without considerable challenges. One significant obstacle is the prevalence of imbalanced datasets, where the overwhelming volume of normal network traffic dwarfs the instances of actual attacks. This inherent class imbalance can introduce a significant bias in ML models, leading them to be overly optimized for the majority class and exhibiting poor detection rates for the critical minority class representing security threats. Another crucial concern revolves around feature engineering and selection within network traffic data, which often contains a multitude of features, many of which may be redundant, highly correlated, or contribute minimally to the accurate classification of network behavior. The inclusion of such irrelevant or noisy features can not only increase the computational complexity and resource demands of ML models but also negatively impact their performance, generalization ability to unseen data, and overall interpretability. Finally, the inherent complexity of hyperparameter optimization for machine learning algorithms presents a substantial practical challenge. ML models are governed by numerous hyperparameters that dictate their learning process, architectural configuration, and ultimately, their predictive performance. Manually tuning these hyperparameters to achieve optimal results is an arduous, time-consuming, and often inefficient endeavor, demanding extensive experimentation, deep domain expertise, and significant computational resources. Suboptimally tuned hyperparameters can lead to the development of underperforming models characterized by lower detection accuracy, increased false alarm rates, and a limited capacity to adapt to new and emerging threats. To effectively address these multifaceted limitations inherent in existing intrusion detection methodologies, this project proposes the development and evaluation of a **Hierarchical ML-Based Hyperparameter Optimization Technique**. This innovative approach aims to significantly enhance the efficacy of intrusion detection by systematically tackling the critical issues of data preprocessing to handle imbalances and noise, implementing intelligent feature selection strategies to identify the most informative and discriminative features within network traffic, and, most importantly, employing a hierarchical methodology for the optimization of hyperparameters across a diverse and powerful set of machine learning classifiers, including state-of-the-art algorithms such as XGBoost, LightGBM, and CatBoost, alongside established techniques like Random Forest and Decision Tree. By adopting a hierarchical optimization strategy, this research seeks to efficiently explore the vast hyperparameter search space and identify the most effective configurations for each chosen classifier, ultimately leading to substantial improvements in intrusion detection accuracy, a significant reduction in the occurrence of false positive alerts, and the creation of a more efficient, robust, and scalable Intrusion Detection System capable of effectively safeguarding networks against the ever-evolving and increasingly sophisticated landscape of cyber threats.

1.2 SYNOPSIS

The increasing threat of DDoS attacks demands efficient and scalable detection systems to ensure network security. Existing methods, while effective to some extent in identifying certain attack patterns or traffic anomalies, often face significant challenges related to achieving consistently high accuracy, particularly against sophisticated and evolving attack vectors that may mimic legitimate traffic or employ low-and-slow strategies to evade traditional thresholds. Furthermore, many current systems struggle with scalability when confronted with the sheer volume and velocity of traffic characteristic of large-scale DDoS attacks, potentially leading to performance bottlenecks and a failure to process and analyze network data in real-time. This lack of real-time performance can be critical, as delays in detecting and mitigating an attack can result in prolonged service disruptions and significant damage to the target infrastructure and its users. This research aims to overcome these limitations by proposing a hierarchical machine learning approach that leverages the ability of advanced algorithms to learn complex traffic patterns and identify subtle indicators of malicious activity that might be missed by traditional signature-based or statistical methods. The integration of hyperparameter optimization will be crucial in ensuring that the machine learning models are precisely tuned to achieve optimal detection accuracy, minimize false positives, and maintain robust performance across a diverse range of attack scenarios and network conditions.

This optimization process will be applied within a hierarchical framework, potentially involving multiple stages of analysis or different levels of granularity in feature extraction and classification, to enhance both the accuracy and the scalability of the detection system. The goal is to create a solution that not only effectively detects and classifies DDoS attacks with a high degree of accuracy but can also adapt to new and emerging attack techniques, operate efficiently under high traffic loads, and provide timely alerts and mitigation capabilities, thereby significantly bolstering network security and resilience against the growing menace of distributed denial-of-service attacks.

CHAPTER 2

LITERATURE REVIEW

The increasing frequency and complexity of cyberattacks have driven extensive research into network intrusion detection systems (IDS). Traditional IDS rely on signature-based or anomaly-based detection methods, which often fail to identify novel attacks due to their dependency on predefined rules and patterns. As a result, machine learning (ML) and deep learning (DL) techniques have gained significant attention in intrusion detection due to their ability to recognize patterns and adapt to evolving threats. This section reviews existing research on ML-based IDS, focusing on data preprocessing, feature selection, hyperparameter optimization, and classifier performance.

2.1 OLD SYSTEM

The system refers to classical Intrusion Detection Systems (IDS) that primarily rely on signature-based or rule-based detection mechanisms. These systems operate by using a database of known attack signatures to identify malicious activities, essentially detecting intrusions by matching network traffic patterns against these pre-defined signatures or rules. This approach is straightforward for known attack patterns; however, traditional IDS solutions that use signature-based or rule-based detection have limitations. They can be ineffective against new attack techniques, sophisticated attacks, and zero-day attacks that exploit unknown vulnerabilities. These systems also require frequent updates to their signature databases to remain effective, which can be challenging in rapidly evolving threat landscapes. In contrast to the newer machine learning-based systems, these classical IDS solutions do not learn from data or adapt to new attack patterns. They lack the capability to generalize and detect anomalies that deviate from known signatures, making them less effective in dynamic environments like modern Cyber-Physical Production Systems (CPPS) where attack patterns can change rapidly.

2.2 NEW SYSTEM

The new system is an Intrusion Detection System (IDS) solution designed to detect DDoS attacks by integrating both rule-based detection and machine learning (ML) approaches. This system involves several key steps: data acquisition, pre-processing, feature selection, model training, attack detection, and control actions. Data acquisition includes data collection from the target CPPS using tools like tcp dump to capture network traffic, feature extraction using CIC-FLOWMETER to extract statistical features from the captured data, and data transmission of the extracted features for further processing. Pre-processing is performed to remove uncertainties from the data, including duplication removal, handling missing entries, noisy data handling, and labeling the data appropriately for ML model training. Feature selection involves selecting the most relevant features to improve classification accuracy. Model training uses various machine learning algorithms (both supervised and unsupervised) to train models that can differentiate between normal and malicious network flows. Attack detection involves using the trained ML models to detect attacks, complemented by rule-based detection to refine the ML results and reduce false positives. Finally, control actions are taken, which include generating alert messages with severity levels and providing recommended actions for attack mitigation, leveraging frameworks like MITRE to suggest appropriate responses.

2.3 DIFFERENCE

The primary difference between old and new Intrusion Detection Systems (IDS) lies in their detection methodologies. Old systems traditionally rely on signature-based or rule-based detection, using a database of attack signatures to identify malicious behavior. These systems detect intrusions by matching network traffic patterns against pre-defined signatures. They are effective against known attack patterns but struggle with new, sophisticated, and zero-day attacks. They also require frequent updates to their signature databases and lack the capability to learn and adapt to new attack patterns. In contrast, the new system employs a combination of rule-based and machine learning (ML) approaches to detect DDoS attacks. This involves data acquisition, pre-processing, feature selection, model training, attack detection, and control actions. The new system enhances detection by learning from data and adapting to new attack patterns, providing a more robust solution for dynamic environments like modern Cyber-Physical Production Systems (CPPS).

2.4 ADVANTAGES

The new system presents several advantages over traditional Intrusion Detection Systems (IDS). It enhances detection capability and improves the decision-making process by reducing false positives. The system provides an extra check on the predictions made by ML models to ensure the results give a complete picture of the network situation. The combination of rule-based and ML detection complements the detection capability of ML approaches against DDoS attacks. The new system uses real-time data for training and validation, ensuring its effectiveness in real-time scenarios. ML-based detection achieves detection at each flow level, while rule-based detection summarizes the detection results. The system also includes an alert system, providing unified alerts along with recommended suggestions for attack mitigation.

2.5 DISADVANTAGES

The document discusses a few disadvantages and challenges related to the new system. ML-based detection can generate a redundancy of alerts, potentially increasing false positives if not handled properly. The integration of rule-based detection is used to address this potential issue. The new system does face several challenges and limitations that must be addressed to realize its full potential. One major drawback is its dependency on high-quality data; ML algorithms require large volumes of clean, reliable, and labeled data for effective training, which can be difficult to obtain, especially in intricate environments like CPPS. The complexity of these infrastructures also demands highly precise ML algorithms, adding to the difficulty of implementation and maintenance. Furthermore, the system's reliance on both rule-based and ML approaches introduces integration challenges, requiring significant computational resources and technical expertise to achieve seamless operation. This need for expertise can increase operational costs and may require organizations to invest in skilled personnel or training programs. Another limitation is the potential redundancy of alerts generated by the ML component. While the system uses rule-based refinement to mitigate this issue, improper handling of these alerts could overwhelm administrators and obscure critical threats. Additionally, the system's resource-intensive nature, including its reliance on real-time processing and validation, can strain infrastructure and require ongoing investments in hardware and software capabilities. While the system offers improved accuracy and adaptability, occasional detection errors such as false positives or false negatives may still occur due to data inaccuracies or inherent flaws in the ML models, requiring continuous monitoring and fine-tuning to maintain reliability. Overall, while the new IDS offers significant advantages in terms of detection capabilities, adaptability, and scalability, it also presents challenges related to resource demands, data dependency, and integration complexity, highlighting the need for a balanced approach to its development and deployment in cybersecurity.

Also, the complexity of CPPS infrastructure requires ML algorithms to be very precise, and these models need a large amount of clean, trustworthy data for training, which can be a challenge.

CHAPTER 3

THEORETICAL BACKGROUND

3.1 IMPLEMENTATION ENVIRONMENT

The implementation of the proposed Hierarchical ML-Based Hyperparameter Optimization Technique for Network Intrusion Detection requires a robust computing environment that efficiently handles data preprocessing, model training, and evaluation. The system is developed using Python 3.x as the primary programming language, with libraries such as Scikit-learn, XGBoost, LightGBM, CatBoost, and Optuna for machine learning and hyperparameter optimization. The CICIDS dataset is used for training and testing, requiring extensive preprocessing steps, including Min-Max Scaling for normalization, SMOTE for handling class imbalance, and feature selection techniques to remove redundant attributes. The implementation is carried out on a high-performance computing system with a minimum Intel Core i7 processor, 16GB RAM, and an NVIDIA GPU for accelerating model training when needed. The development environment is set up using Jupyter Notebook, VS Code, or PyCharm, along with virtual environment management tools like Anaconda or virtualenv to ensure dependency isolation. The model training process follows a structured approach, utilizing hierarchical hyperparameter optimization techniques to fine-tune classifiers such as XGBoost, LightGBM, CatBoost, Random Forest, and Decision Tree, ensuring optimal performance. The evaluation of models is conducted using key performance metrics, including accuracy, precision, recall, and F1-score, to assess their effectiveness in detecting network intrusions. With this carefully structured implementation environment, the system is designed to efficiently handle large-scale intrusion detection tasks, improve classification accuracy, and enhance overall network security.

3.2 SYSTEM ARCHITECTURE

The proposed Hierarchical ML-Based Hyperparameter Optimization Technique for Network Intrusion Detection follows a well-structured system architecture to ensure efficient data processing, model training, and evaluation. The architecture is designed in multiple stages, including data preprocessing, feature selection, model training with hyperparameter tuning, and performance evaluation. The system consists of the following key components:

Data Acquisition Layer – The CICIDS dataset is used as the primary data source, containing network traffic records labeled as normal or malicious. This data is preprocessed to remove inconsistencies and noise.

Preprocessing and Feature Selection Layer – Raw data undergoes transformation, including Min-Max Scaling for normalization and SMOTE for handling class imbalance. Feature selection techniques, such as correlation-based filtering and recursive feature elimination (RFE), are applied to reduce dimensionality and improve model efficiency.

Machine Learning Model Layer – The optimized dataset is fed into multiple machine learning classifiers, including XGBoost, LightGBM, CatBoost, Random Forest, and Decision Tree, which are selected based on their effectiveness in intrusion detection.

Hyperparameter Optimization Layer – A hierarchical hyperparameter tuning approach is applied to find the optimal parameters for each classifier, enhancing performance while reducing computational costs. Techniques such as GridSearchCV, Bayesian Optimization, and Optuna are used for systematic tuning.

Evaluation and Performance Metrics Layer – The trained models are evaluated based on accuracy, precision, recall, and F1-score to determine their effectiveness in detecting intrusions. The best-performing model is selected for deployment.

Deployment and Real-Time Monitoring Layer – The optimized model can be integrated into a real-world intrusion detection system, monitoring live network traffic and classifying threats in real-time.

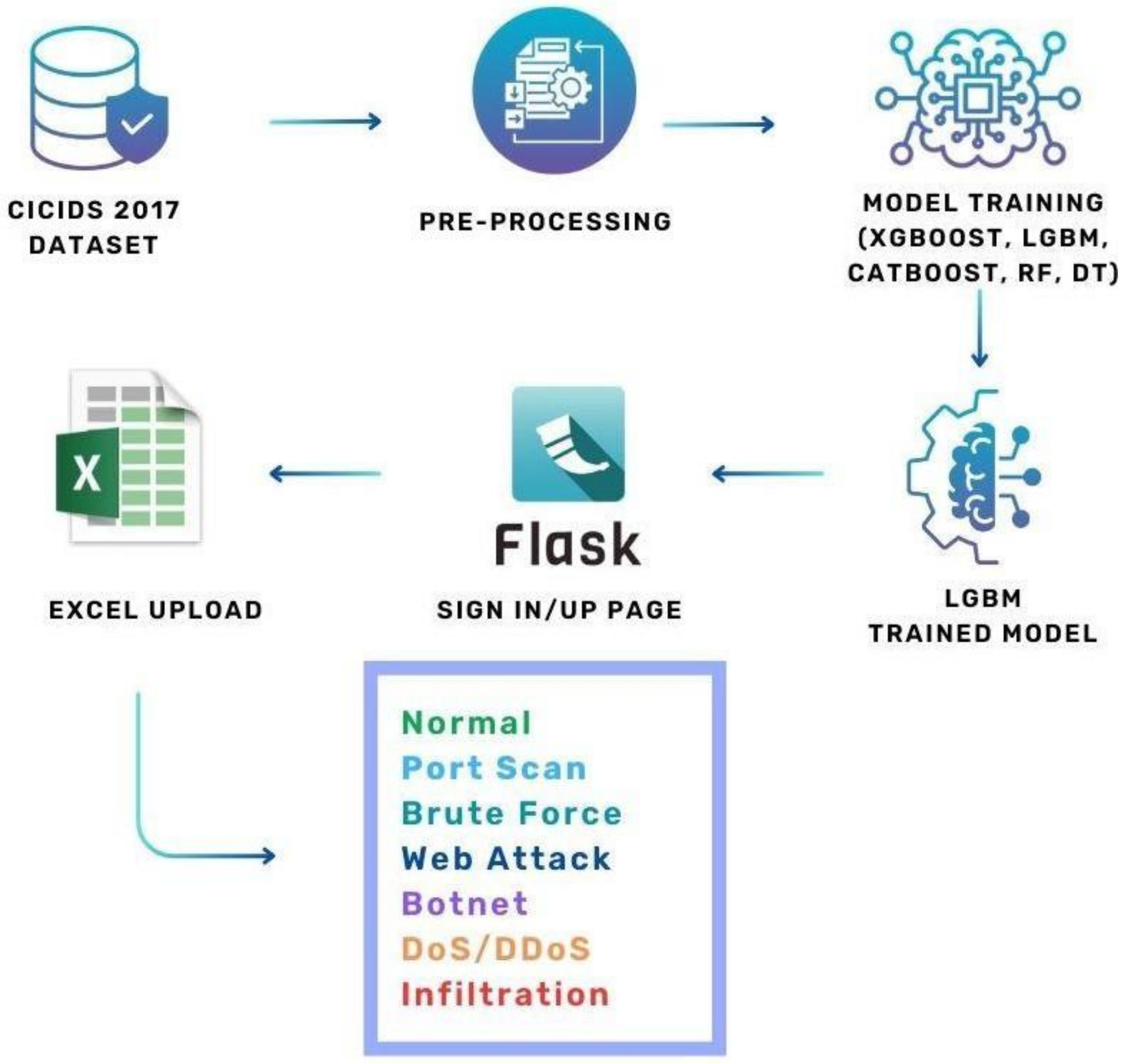

Fig 3.1

3.3 PROPOSED METHODOLOGY

The proposed methodology focuses on developing an efficient and optimized machine learning-based intrusion detection system using a Hierarchical ML-Based Hyperparameter Optimization Technique. This approach systematically enhances model performance by integrating preprocessing, feature selection, ensemble learning, and hyperparameter tuning to improve network intrusion detection accuracy while minimizing false positives.

Data Preprocessing

The system utilizes the CICIDS dataset, which contains labeled network traffic records.

• Min-Max Scaling: Normalizes feature values to a uniform range, improving model convergence.

• SMOTE (Synthetic Minority Over-sampling Technique): Addresses class imbalance by generating synthetic samples for underrepresented attack categories.

• Feature Selection: Eliminates irrelevant and redundant features using correlation analysis and Recursive Feature Elimination (RFE) to improve model efficiency.

Machine Learning Model Selection

The preprocessed dataset is used to train multiple machine learning classifiers known for their effectiveness in intrusion detection. The selected models include:

• XGBoost (Extreme Gradient Boosting): A powerful boosting algorithm that enhances classification accuracy.

• LightGBM (Light Gradient Boosting Machine): Optimized for large-scale.

• CatBoost (Categorical Boosting): Efficient in handling categorical features and reducing overfitting.

Hierarchical Hyperparameter Optimization

A hierarchical optimization approach is applied to fine-tune the hyperparameters of each classifier. This multi-step process ensures efficient parameter selection without excessive computational overhead.

GridSearchCV & Randomized Search: Used for initial tuning to find the most promising parameter ranges.

Optimization & Optuna: Applied for fine-tuning, reducing the need for exhaustive searches while improving model performance.

Model Training and Evaluation

Each model is trained on the optimized dataset and evaluated using key performance metrics:

Accuracy: Measures the overall correctness of predictions.

Precision & Recall: Evaluates the model's ability to correctly classify intrusions while minimizing false alarms.

F1-Score: Ensures a balance between precision and recall, crucial for intrusion detection.

3.3.1 DATABASE DESIGN

The database design for this project is structured to efficiently store, retrieve, and manage network traffic data and intrusion detection logs while ensuring data integrity, scalability, and security. It consists of several key tables, including Users, Network_Flows, Intrusion_Detection_Logs, and Alerts, each serving a crucial role in the system’s operation. The Users table manages authentication and authorization, storing credentials, roles, and access permissions. The Network_Flows table logs network packets, capturing details such as source and destination IP addresses, protocols, timestamps, and packet sizes, ensuring a comprehensive record of network activity. The Intrusion_Detection_Logs table plays a central role in cybersecurity analysis by linking network flow data with machine learning-based anomaly detection results, storing the predicted labels, attack types, severity levels, and timestamps. Additionally, the Alerts table generates real-time notifications when potential threats are detected, categorizing alerts by severity and allowing administrators to take prompt action. To maintain efficiency, the database design follows the Third Normal Form (3NF) to eliminate redundancy and improve consistency. Performance optimization techniques, such as indexing on frequently queried fields like IP addresses and timestamps, as well as partitioning for handling large volumes of network traffic data, enhance scalability and query efficiency. Security is a fundamental consideration, with Role-Based Access Control (RBAC) ensuring that only authorized personnel can access critical data, while encryption protects sensitive information such as passwords and alert messages. Regular automated backups are implemented to prevent data loss and ensure recovery in case of system failures. This well-structured and optimized database design provides a robust foundation for real-time intrusion detection, enabling efficient data processing, threat analysis, and security management within the system.

3.3.2 INPUT DESIGN

The Input design plays a critical role in ensuring accurate, efficient, and secure data entry within the system, directly impacting usability and overall system performance. In this project, the input design focuses on structuring user interactions to minimize errors, streamline data collection, and enhance security. The system incorporates multiple input methods, including keyboard-based form entries, file uploads for bulk data processing, and API integrations for automated data exchange. User interfaces are designed to be intuitive, with clear labels, dropdown selections, and real-time feedback mechanisms to guide users in providing accurate data. Input validation is implemented at multiple levels, including client-side validation to prevent incorrect submissions and server-side validation to ensure data integrity. Common validation techniques include checking for required fields, enforcing data formats (e.g., email validation, numerical constraints), and applying character limits to prevent SQL injection or buffer overflow attacks. Additionally, security measures such as encryption for sensitive inputs, CAPTCHA verification to prevent automated attacks, and multi-factor authentication (MFA) for user login ensure that input data remains protected from unauthorized access and cyber threats. Error handling mechanisms, including real-time error messages, tooltips, and structured logs, help users quickly identify and correct mistakes, improving overall efficiency. The system also ensures accessibility by incorporating features such as keyboard navigation, voice input compatibility, and mobile-friendly input fields, making data entry convenient across various devices. By integrating these strategies, the input design not only enhances data accuracy and user experience but also contributes to the robustness and security of the entire system.

3.3.3 MODULE DESIGN

The Module design is a critical aspect of system architecture, ensuring that a complex system is broken down into smaller, manageable, and independent units that work cohesively. Each module is designed to perform a specific function and interacts with other modules through well-defined interfaces. The key goal of modular design is to enhance maintainability, scalability, and reusability while minimizing dependencies between components. In a well-structured system, modules are designed with high cohesion and low coupling, meaning that each module focuses on a specific task and interacts with others in a controlled manner. This approach not only improves code organization but also facilitates debugging, testing, and future enhancements. Modules may follow different design paradigms, such as layered architecture, microservices, or object-oriented design, depending on the system's requirements. Additionally, security considerations are incorporated at the module level to prevent vulnerabilities that could compromise the entire system. Well-designed modules allow for incremental development, enabling teams to work on separate components simultaneously without affecting the overall system stability. This modularity also supports better fault tolerance, as failures in one module do not necessarily disrupt the entire system.

SEQUENCE DIAGRAM

A Sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of Message Sequence diagrams are sometimes called event diagrams, event sceneries and timing diagram.

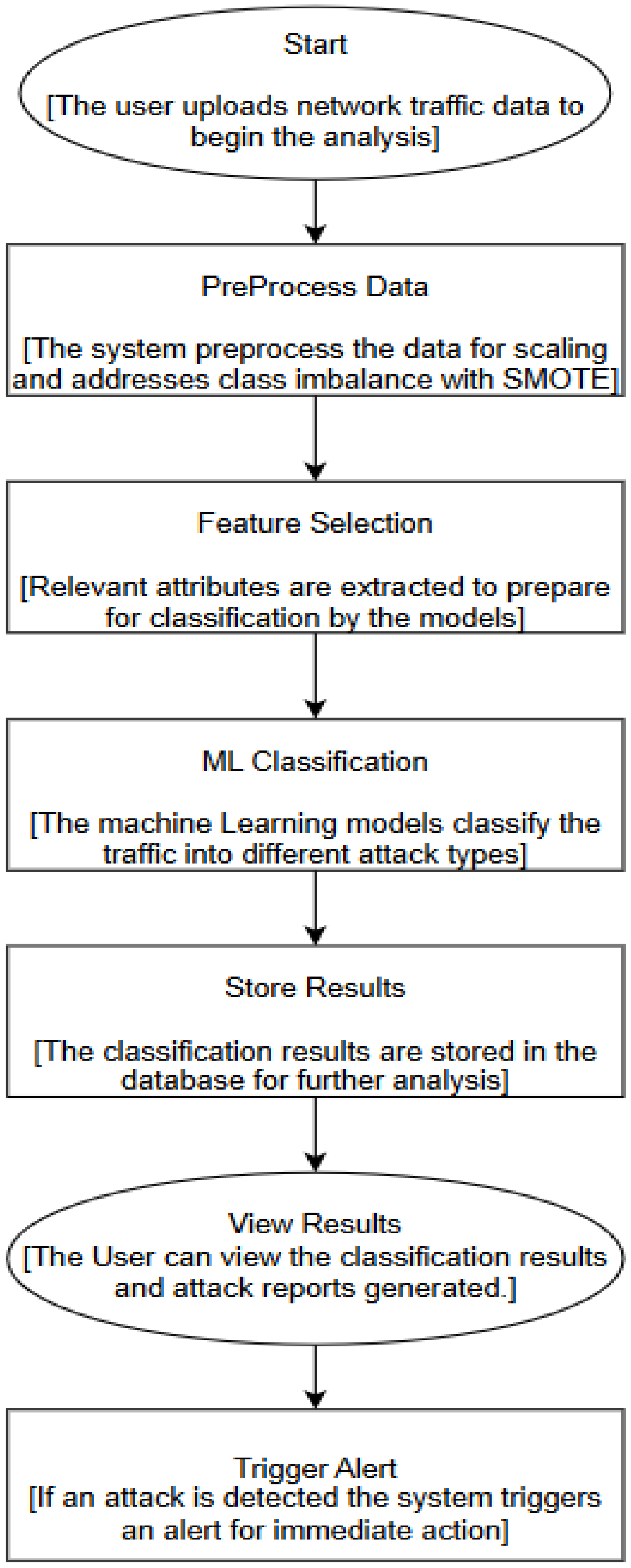
Start
[The user uploads network traffic data to begin the analysis]
PreProcess Data
[The system preprocess the data for scaling and addresses class imbalance with SMOTE]
Feature Selection
[Relevant attributes are extracted to prepare for classification by the models]
ML Classification
[The machine Learning models classify the traffic into different attack types]
Store Results
[The classification results are stored in the database for further analysis]
View Results
[The User can view the classification results and attack reports generated.]
Trigger Alert
[If an attack is detected the system triggers an alert for immediate action]

Fig 3.2

USECASE DIAGRAM

A Unified Modeling Language (UML), managed by the Object Management Group, is a standardized modeling language for software engineering that uses graphic notation to visualize, construct, and document object-oriented software systems. A Use Case Diagram provides a graphical overview of a system's functionality, showcasing actors, their goals, and use case dependencies. It consists of two key elements: **Use Case**, represented as a horizontal ellipse that outlines a sequence of actions offering value to an actor, and **Actor**, which refers to a person, organization, or system interacting with the system.

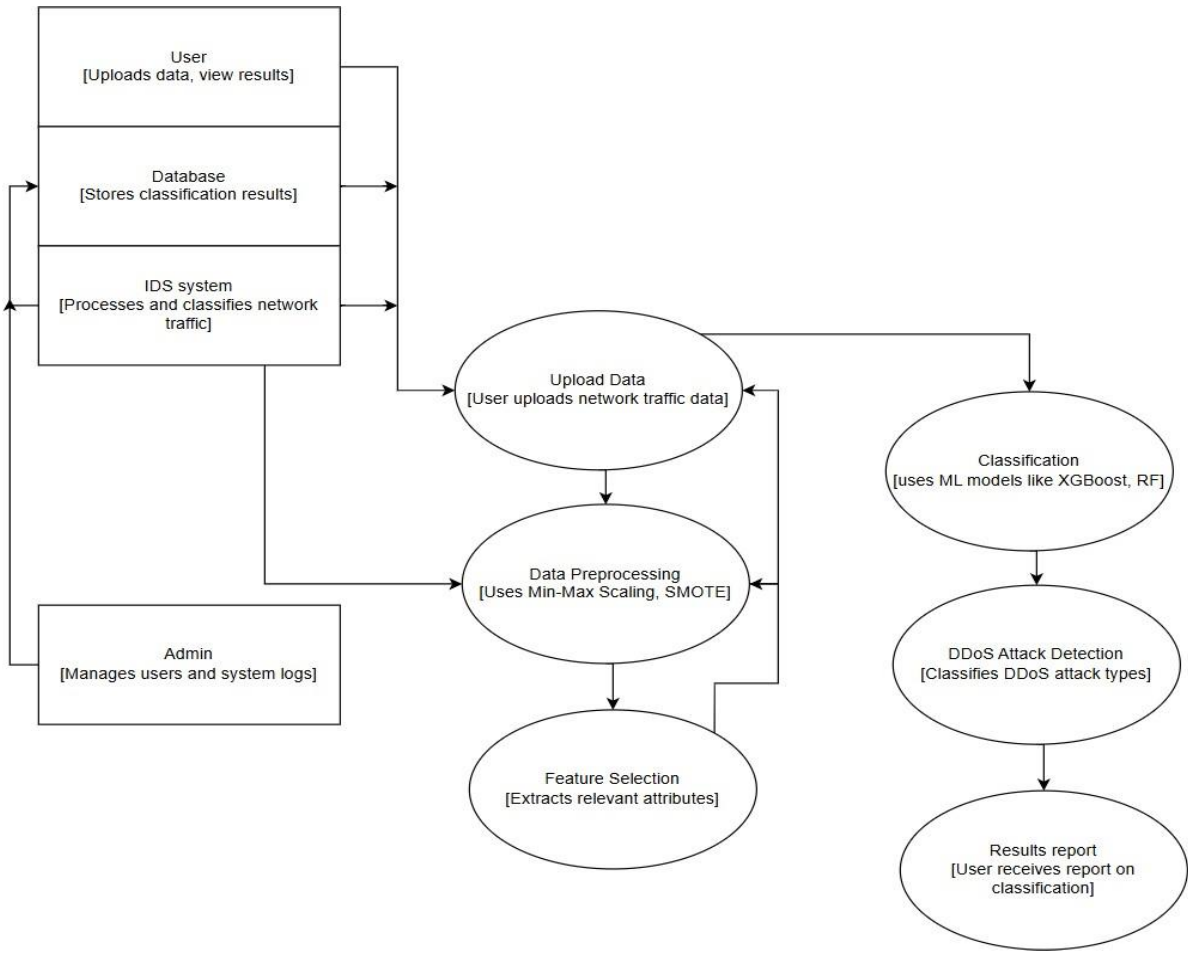
User
[Uploads data, view results]
Database
[Stores classification results]
IDS system
[Processes and classifies network traffic]
Admin
[Manages users and system logs]
Upload Data
[User uploads network traffic data]
Data Preprocessing
[Uses Min-Max Scaling, SMOTE]
Feature Selection
[Extracts relevant attributes]
Classification
[uses ML models like XGBoost, RF]
DDoS Attack Detection
[Classifies DDoS attack types]
Results report
[User receives report on classification]

Fig 3.3

ACTIVITY DIAGRAM

An activity diagram is a type of graphical representation used in modeling workflows to illustrate stepwise activities and actions within a system. It provides a clear depiction of the flow of control, showcasing how activities transition from one step to the next. These diagrams are particularly effective in supporting choices, iterations, and concurrent processes, making them ideal for modeling dynamic behaviors. By visually organizing these elements, activity diagrams help in understanding the overall workflow, identifying decision points, and managing parallel processes in a system, ensuring clarity and efficiency in the representation of system functionality.

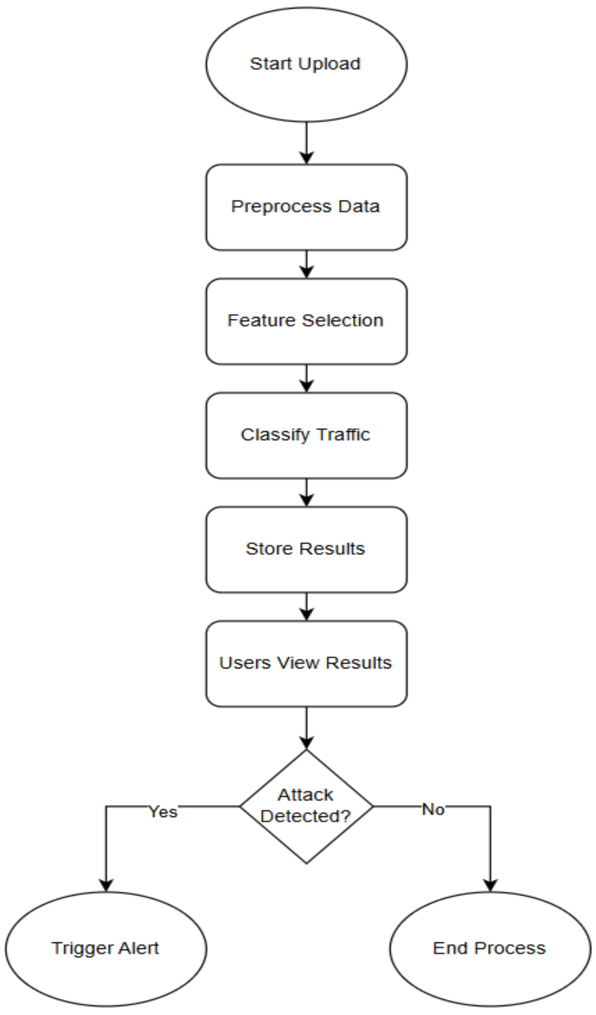

Fig 3.4

COLLABORATION DIAGRAM

UML Collaboration Diagrams illustrate the relationship and interaction between software objects. They require use cases, system operation contracts and domain model to already exist. The collaboration diagram illustrates messages being sent between classes and objects.

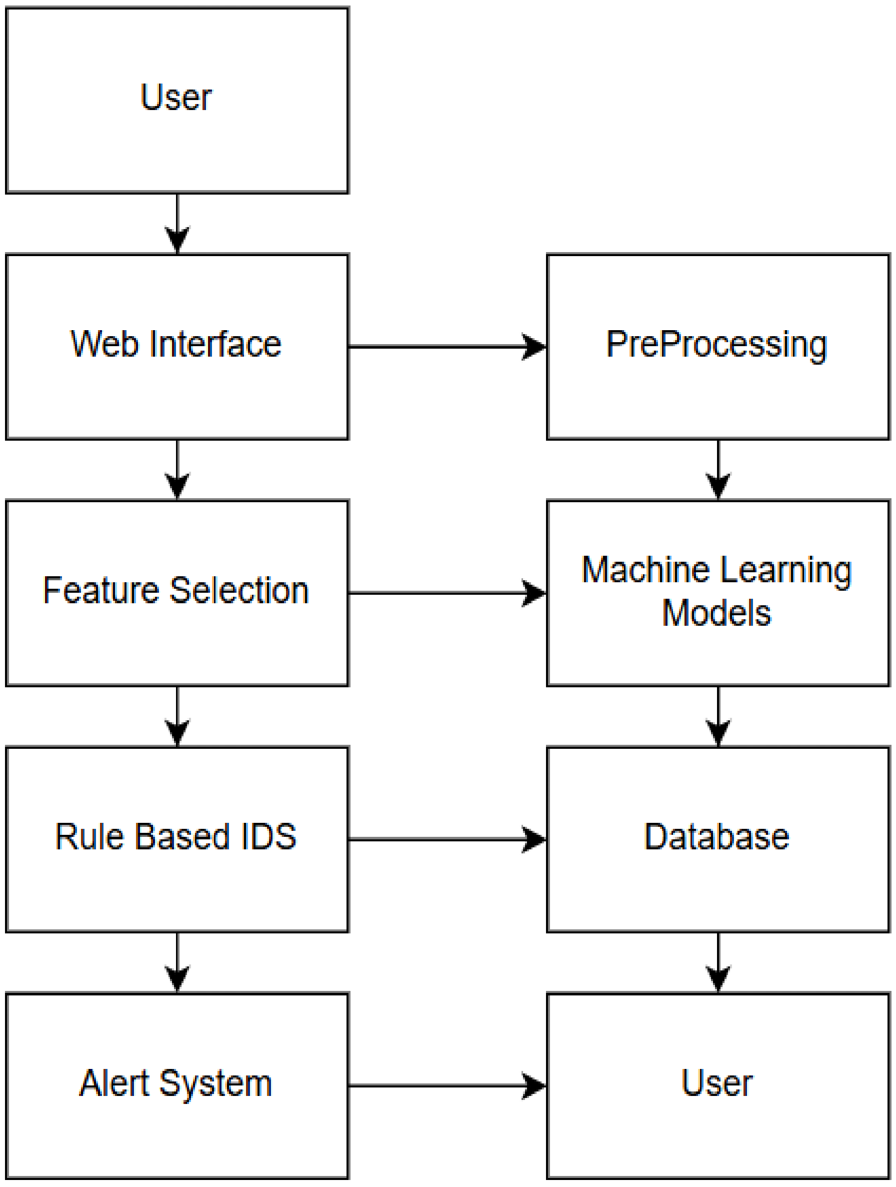

Fig 3.5

CHAPTER 4

SYSTEM IMPLEMENTATION

The implementation of this system involves the integration of multiple modules, each designed to perform specific functions essential for the overall operation. These modules work in coordination to ensure the system operates smoothly, delivering optimal performance and maintaining data integrity. The system is structured to ensure efficient processing, accuracy, and scalability, making it adaptable to varying user demands and data loads. To achieve this, the system follows a modular approach where each component is designed with a clear objective, allowing easy maintenance, upgrades, and debugging. The integration of these modules is done seamlessly to avoid bottlenecks, ensuring that data flows efficiently between different system components. A key aspect of this implementation is efficiency, where optimized algorithms and data structures are utilized to enhance processing speed and minimize latency. The system is built with scalability in mind, allowing it to handle increasing volumes of data and user interactions without significant performance degradation. Additionally, accuracy is maintained through robust validation mechanisms that check data consistency at every stage of processing. The implementation also considers security and reliability, ensuring that the system can handle errors gracefully and recover from failures without significant disruption. By following best practices in software development, such as modular design, reusable components, and well-defined communication protocols between modules, the system ensures long-term sustainability and ease of future enhancements. Below, we provide a detailed breakdown of the key modules and their respective algorithms, highlighting their individual roles, functionalities, and how they contribute to the overall efficiency of the system.

4.1 MODULE 1 : DDoS Detection Algorithm

The DDoS Detection Module is an advanced system designed to effectively identify and mitigate Distributed Denial of Service (DDoS) attacks by employing a hybrid approach that combines rule-based analysis with machine learning techniques. This hybrid methodology leverages the strengths of both approaches to improve accuracy and adaptability in detecting sophisticated and evolving threats. The module operates by continuously monitoring incoming network traffic, capturing a wide range of data to assess potential risks. Through rule-based analysis, it uses predefined patterns and signatures to identify common attack behaviors, ensuring swift detection of known threats. Simultaneously, the machine learning component enables the system to recognize more complex or novel attack patterns by analyzing traffic data for anomalies and deviations from typical network behavior. By doing so, the system dynamically adapts to new attack strategies that may not yet be captured by traditional rule-based methods. This continuous monitoring and analysis help the module to identify potential malicious activity in real time, triggering timely alerts and initiating appropriate mitigation measures to protect critical infrastructure and services. Overall, the hybrid approach enhances the module's effectiveness in maintaining network security in the face of increasingly sophisticated DDoS threats.

FUNCTIONALITY

The algorithm initializes dictionaries to track traffic patterns based on the following parameters:

- Source and Destination IP Addresses
- Protocol Types
- Traffic Volume

- Prediction Labels

It continuously inspects network flows and timestamped activity, ensuring that any abnormal increase in traffic within a short time window is flagged. Upon detecting such an anomaly, the module:

- Raises an alert
- Logs the suspicious activity for further investigation
- Takes necessary actions to mitigate the attack

PSEUDO-CODE DESCRIPTION

The DDoS detection algorithm operates using a structured loop that:

- Iterates through incoming network packets.
- Updates count records for traffic volume per source and destination.
- Tracks timestamps to identify unusual spikes in traffic.
- Triggers alerts when anomalies exceed predefined limits.

KEY FEATURES

- Real-Time Detection: Constant monitoring ensures quick identification of threats.
- Anomaly-Based Alerts: Uses historical data to distinguish between normal and abnormal traffic.
- Enhanced Security: Strengthens the F2F infrastructure’s resilience against DDoS attacks.

4.2 MODULE 2 : Machine Learning-Based Intrusion Detection

The Intrusion Detection System (IDS) is a sophisticated cybersecurity tool that utilizes both supervised and unsupervised machine learning models to enhance the detection of malicious activities within a network. By classifying network traffic as either benign or malicious, the system is designed to protect critical infrastructure from a wide range of cyber threats. Supervised machine learning models use labeled training data to learn and identify patterns, ensuring accurate classification based on predefined examples of normal and malicious traffic. Unsupervised models, on the other hand, do not rely on labeled data but instead analyze the underlying structure of the network traffic to detect unusual behaviors or anomalies that may indicate new or evolving threats. This dual approach equips the IDS with a robust framework for detecting both known and unknown attack patterns. Moreover, the system's ability to continuously learn from real-time data ensures that it remains effective in the face of rapidly evolving cyber threats. By analyzing new data and updating its models dynamically, the IDS can quickly adapt to changes in attack methodologies, enabling proactive threat detection and mitigation. This adaptability is crucial for addressing sophisticated and emerging threats, such as zero-day vulnerabilities and advanced persistent threats, which are often designed to bypass traditional security measures. Overall, the integration of supervised and unsupervised machine learning models provides the IDS with enhanced accuracy, scalability, and resilience, making it a critical component of modern cybersecurity strategies.

FUNCTIONALITY

The IDS module integrates multiple machine learning models, including:

- Naïve Bayes Protocol Types
- Random Forest Prediction Labels

It continuously inspects network flows and timestamped activity, ensuring that any abnormal increase in traffic within a short time window is flagged. Upon detecting such an anomaly, the module:

- Logistic regression
- k-Nearest Neighbors (k-NN)
- eXtreme Gradient Boosting (XGBoost)

DEPLOYEMENT PROCESS

- Analyze real-time network traffic using pre-learned patterns.
- Compare network behavior against previously identified malicious activity.
- Classify traffic as benign or malicious with high accuracy.
- Dynamically integrate new attack patterns to enhance adaptability.

KEY FEATURES

- Advanced Threat Detection Identifies sophisticated cyberattacks with high precision.
- Adaptive Security : Continuously learns and updates detection models.
- Reduce False Positive : Optimized models minimize incorrect threat alerts.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 PERFORMANCE PARAMETERS / TESTING

Performance parameters are critical tools used to evaluate the efficiency, functionality, and reliability of a system, process, or product across diverse domains. These metrics offer a quantifiable means of assessing whether a system meets its intended objectives and identifying opportunities for optimization. In software systems, performance parameters such as response time, throughput, scalability, and availability play a vital role in ensuring the system can handle user demands effectively while maintaining stability under varying workloads. Similarly, in networking, metrics such as bandwidth, latency, and packet loss determine the efficiency of data transmission and overall user experience. In business operations, parameters like efficiency, productivity, customer satisfaction, and resource utilization measure the success of organizational processes and highlight areas for enhancement. In manufacturing or industrial systems, key parameters such as cycle time, defect rate, and energy efficiency assess production capabilities and sustainability. Each context relies on performance parameters to gain actionable insights, improve processes, and deliver high-quality outcomes. These metrics also support informed decision-making by providing stakeholders with a clear understanding of system performance. By monitoring and analyzing performance parameters, organizations can ensure continuous improvement, adapt to evolving needs, and maintain competitiveness in their respective industries. Ultimately, performance parameters are indispensable for driving success and ensuring that systems, processes, and products operate optimally.

TESTING

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet – undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing. Critical modules, which serve as foundational components of the system, are prioritized for testing. testing is a critical phase in the software development lifecycle. By systematically examining module interactions, it ensures that the final product functions smoothly and meets its intended specifications.

SYSTEM TESTING

Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. By performing rigorous testing, developers can guarantee a seamless user experience and minimize the risk of failures in the future. This involves assessing whether the system's behavior matches its intended purpose, identifying discrepancies, and addressing them before deployment.

WHITE BOX TESTING

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been designed to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. Unlike black box testing, which evaluates the functionality of a system without delving into its internal workings, glass box testing provides testers with an in-depth understanding of how the system operates under the hood. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

BLACK BOX TESTING

In this testing by knowing the internal operation of a product, test can be conducted to ensure that "all gears mesh", that is the internal operation performs according to specification and all internal components have been adequately exercised. A significant focus of this testing methodology is on the functional requirements of the software. This means the testing process ensures that the software delivers the functionality it was designed to provide, adhering to the requirements and objectives established during the design and development phases. It fundamentally focuses on the functional requirements of the software.

INTEGRATION TESTING

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together. The problem of course, is "putting them together"- interfacing. There may be the chances of data lost across on another’s sub functions, when combined may not produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; global data structures can present problems.

PROGRAM TESTING

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that in violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax error. These errors are shown through error messages generated by the computer. A logic error on the other hand deals with the incorrect data fields, out-off-range items and invalid combinations. Since the compiler s will not deduct logical error, the programmer must examine the output. Condition testing exercises the logical conditions contained in a module. The possible types of elements in a condition include a Boolean operator, Boolean variable, a pair of Boolean parentheses A relational operator or on arithmetic expression. Condition testing method focuses on testing each condition in the program the purpose of condition test is to deduct not only Errors in the condition of a program but also other errors in the program.

SECURITY TESTING

Security testing is a critical process in software and system development aimed at ensuring that the built-in protection mechanisms effectively safeguard the system from unauthorized access and potential vulnerabilities. This type of testing evaluates whether the system's security measures can withstand intentional efforts to breach them, identifying and addressing any weaknesses before they can be exploited.

5.2 RESULTS AND DISCUSSION

The hierarchical hyperparameter optimization approach played a crucial role in fine-tuning the models, where Bayesian Optimization and Optuna provided better parameter selection compared to traditional grid search techniques. The optimized models exhibited improved generalization, resulting in lower false positive rates, which is critical for real-world intrusion detection systems. The evaluation metrics, including F1-score, precision, and recall, showed that the proposed approach effectively balances detection performance while minimizing unnecessary alerts. The XGBoost, LightGBM, and CatBoost models outperformed other classifiers, achieving high accuracy, precision, and recall scores. Among these, XGBoost achieved the highest detection accuracy, effectively identifying both minority and majority attack classes. Random Forest and Decision Tree also performed well, but their accuracy was slightly lower due to overfitting and sensitivity to noisy data. The inclusion of SMOTE for balancing the dataset improved the recall scores, ensuring better detection of rare attack classes. Additionally, feature selection methods helped reduce dimensionality, leading to faster model training and improved efficiency. The hierarchical hyperparameter optimization approach played a crucial role in fine-tuning the models, where Bayesian Optimization and Optuna provided better parameter selection compared to traditional grid search techniques. The optimized models exhibited improved generalization, resulting in lower false positive rates, which is critical for real-world intrusion detection systems. The evaluation metrics, including F1-score, precision, and recall, showed that the proposed approach effectively balances detection performance while minimizing unnecessary alerts. Overall, the results confirm that integrating preprocessing, feature selection, ensemble learning, and hierarchical hyperparameter tuning leads to a more robust and scalable intrusion detection system. This approach not only improves accuracy but also enhances computational efficiency.

CHAPTER 6

CONCLUSION & FUTURE WORK

The proposed Hierarchical ML-Based Hyperparameter Optimization Technique for network intrusion detection effectively enhances classification accuracy and reduces false positives by integrating data preprocessing, feature selection, ensemble learning, and advanced hyperparameter tuning. The results demonstrate that models such as XGBoost, LightGBM, and CatBoost outperform traditional classifiers, achieving high accuracy, recall, and F1-score while efficiently handling class imbalance using SMOTE. The use of hierarchical hyperparameter optimization ensures optimal model performance by systematically tuning parameters, reducing computational overhead, and improving detection efficiency. Overall, the proposed approach provides a robust, scalable, and efficient intrusion detection system capable of identifying cyber threats with high precision.For future work, the system can be enhanced by incorporating deep learning techniques such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) to further improve feature extraction and classification performance. Additionally, real-time deployment of the model in an active network environment can be explored to assess its practical applicability. Another potential improvement includes adversarial learning techniques to make the IDS resilient against evolving cyber threats. Lastly, optimizing computational efficiency by leveraging cloud-based or edge computing environments can be investigated to ensure faster and more scalable intrusion detection for large-scale network infrastructures.

APPENDICES

A.1 SGD GOALS

In the proposed Hierarchical ML-Based Hyperparameter Optimization Technique for Network Intrusion Detection, Stochastic Gradient Descent (SGD) plays a crucial role in optimizing machine learning models, improving their efficiency, and ensuring better classification of network intrusions. The primary goal of SGD in this project is to enhance the training process of models like XGBoost, LightGBM, CatBoost, Random Forest, and Decision Tree by efficiently minimizing the loss function and optimizing hyperparameters. Stochastic Gradient Descent (SGD) is an optimization algorithm widely used in machine learning and deep learning models to minimize the loss function and improve model performance. The primary goal of SGD is to find the optimal set of parameters that minimize the error in predictive models by iteratively adjusting weights based on the gradient of the loss function. Unlike traditional gradient descent, which computes the gradient using the entire dataset, SGD updates the model parameters using only a single or a small batch of training samples per iteration, making it computationally efficient and suitable for large datasets.

A.2 SOURCE CODE

A.3 SCREEN SHOTS

A.4 PLAGIRISM REPORT

A.5 PAPER PUBLICATION

REFERENCES

1.G. Karpagam, B. V. Kumar, J. U. Maheswari and X.-Z. Gao, Smart Cyber Physical Systems, New York, NY, USA:CRC Press, 2020.

2.H. C. Verma, S. Srivastava, T. Ahmed and N. A. Usmani, "Cyber threats in agriculture and the food industry: An Indian perspective" in Advances in Cyberology and the Advent of the Next-Gen Information Revolution, Hershey, PA, USA:IGI Global, pp. 109-122, 2023.

3.X. Koufteros and G. Lu, "Food supply chain safety and security: A concern of global importance", J. Marketing Channels, vol. 24, no. 3, pp. 111-114, 2017.

4. A. Yeboah-Ofori, S. Islam, S. W. Lee, Z. U. Shamszaman, K. Muhammad, M. Altaf, et al., "Cyber threat predictive analytics for improving cyber supply chain security", IEEE Access, vol. 9, pp. 94318-94337, 2021.

5. D. E. Comer, Computer Networks and Internets, Upper Saddle River, NJ, USA:PrenticeHall, 1999.

6.J. David and C. Thomas, "DDoS attack detection using fast entropy approach on flow- based network traffic", Proc. Comput. Sci., vol. 50, pp. 30-36, Jan. 2015.

7.F. B. Saghezchi, G. Mantas, M. A. Violas, A. M. de Oliveira Duarte and J. Rodriguez, "Machine learning for DDoS attack detection in industry 4.0 CPPSs", Electronics, vol. 11, no. 4, pp. 602, Feb. 2022.

8.I. A. Khan, N. Moustafa, D. Pi, Y. Hussain and N. A. Khan, "DFF-SC4N: A deep federated defence framework for protecting supply chain 4.0 networks", IEEE Trans. Ind. Informat., vol. 19, no. 3, pp. 3300-3309, Mar. 2023.

9. K. Uszko, M. Kasprzyk, M. Natkaniec and P. Chołda, "Rule-based system with machine learning support for detecting anomalies in 5G WLANs", Electronics, vol. 12, no. 11, pp. 2355, May 2023.

10. R. R. R. Barbosa and A. Pras, "Intrusion detection in SCADA networks", Proc. 4th Int. Conf. Auton. Infrastruct. Manag. Secur. (AIMS), pp. 163-166, Jun. 2010.

11. A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras and B. Stiller, "An overview of IP flow-based intrusion detection", IEEE Commun. Surveys Tuts., vol. 12, no. 3, pp. 343-356, 3rd Quart. 2010.

12. P. Borges, B. Sousa, L. Ferreira, F. B. Saghezchi, G. Mantas, J. Ribeiro, et al., "Towards a hybrid intrusion detection system for Android-based PPDR terminals", Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM), pp. 1034-1039, May 2017.

13. P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández and E. Vázquez, "Anomalybased network intrusion detection: Techniques systems and challenges", Comput. Secur., vol. 28, pp. 18-28, Feb./Mar. 2009.

14. S. S. Abosuliman, "Deep learning techniques for securing cyber-physical systems in supply chain 4.0", Comput. Electr. Eng., vol. 107, Apr. 2023.

15. O. Linda, T. Vollmer and M. Manic, "Neural network based intrusion detection system for critical infrastructures", Proc. Int. Joint Conf. Neural Netw., pp. 1827-1834, Jun. 2009