A Novel Method to Detect the DDoS Attack in Network using Machine Learning

A Mini Project Report Submitted
In partial fulfillment of the requirement for the award of the degree of

Bachelor of Technology
In
Computer Science and Engineering -Artificial Intelligence and Machine Learning

by

Gadhari Kenneth - 20N31A6615

Garlapati Rohan Adithya - 20N31A6617

Pasagada Deekshitha - 20N31A6646

Under the Guidance of

Dr. Thota. Siva Ratna Sai
Associate. Professor
Department of Computational Intelligence

MRCET

DEPARTMENT OF COMPUTATIONAL INTELLIGENCE
MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY
(Affiliated to JNTU, Hyderabad)
ACCREDITED by AICTE-NBA
Maisammaguda, Dhulapally post, Secunderabad-500014.
2020-2024

DECLARATION

We hereby declare that the project titled "**A Novel Method to Detect the DDoS Attack in Network using Machine Learning**" submitted to **Malla Reddy College of Engineering and Technology** (UGC Autonomous), affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH) for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering- Artificial Intelligence and Machine Learning** is a result of original work carried-out in this report.

It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

G. Kenneth (20N31A6615)
G. Rohan Adithya (20N31A6617)
P. Deekshitha (20N31A6646)

MALLA REDDY COLLEGE
OF ENGINEERING & TECHNOLOGY
(AUTONOMOUS INSTITUTION - UGC, GOVT. OF INDIA)
Affiliated to JNTUH; Approved by AICTE, NBA-Tier 1 & NAAC with A-GRADE | ISO 9001:2015

CERTIFICATE

This is to certify that this is the bonafide record of the project titled **"A Novel Method to Detect the DDoS Attack in Network using"** submitted by G. Kenneth (20N31A6615), G. Rohan Adithya (20N31A6617), P. Deekshitha (20N31A6646) of B. Tech in the partial fulfillment of the requirements for the degree of **Bachelor of Technology** in **Computer Science and Engineering- Artificial Intelligence and Machine Learning**, Dept. of CI during the year 2023-2024. The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

Dr. Thota. Siva Ratna Sai
Associate. Professor
INTERNAL GUIDE

Dr. D. Sujatha
HEAD OF THE DEPARTMENT

EXTERNAL EXAMINER

ACKNOWLEDGMENT

We feel honored and privileged to place our warm salutation to our college Malla Reddy College of Engineering and technology (UGC-Autonomous), our Director ***Dr. VSK Reddy*** who gave us the opportunity to have experience in engineering and profound technical knowledge.

We are indebted to our Principal ***Dr. S. Srinivasa*** Rao for providing us with facilities to do our project and his constant encouragement and moral support which motivated us to move forward with the project.

We would like to express our gratitude to our Head of the Department ***Dr. D. Sujatha*** for encouraging us in every aspect of our system development and helping us realize our full potential.

We would like to thank our application development guide as well as our internal guide
Dr. Thota. Siva Ratna Sai, for his structured guidance and never-ending encouragement. We are extremely grateful for valuable suggestions and unflinching co-operation throughout application development work.

We would also like to thank all supporting staff of department of CI and all other departments who have been helpful directly or indirectly in making our application development a success.

We would like to thank our parents and friends who have helped us with their valuable suggestions and support has been very helpful in various phases of the completion of the application development.

By

G. Kenneth (20N31A6615)

G. Rohan Adithya (20N31A6617)

P. Deekshitha (20N31A6646)

ABSTRACT

Distributed Denial of Service attack (DDoS) is the most dangerous attack in the field of network security. DDoS attack halts normal functionality of services of various online applications. Systems under DDoS attacks remain busy with false requests (Bots) rather than providing services to legitimate users. Malicious users prevent genuine applications from accessing network. These affect server resources, such as bandwidth and buffer size, are slowed down.

We use machine learning based approach to detect and classify different types of network traffic flows. The proposed approach is validated using a new dataset which is having mixture ofvarious modern types of attacks such as HTTP flood, SID DoS and normal traffic.

Machine learning algorithm such as Random. We use machine learning based approach to detect and classify different types of network traffic flows. The proposed approach is validated using a new dataset which is having mixture of various modern types of attacks such as HTTP flood, SID DoS and normal traffic. Machine learning algorithm such as Random Forest Algorithm, Naive bayes, SVM, XG Boost, AdaBoost, KNN are used to get better accuracy.

TABLE OF CONTENTS

S. No.	Topic	Page No.
1	**INTRODUCTION**	**1-5**
	1.1 Purpose and Objectives	2
	1.2 Existing and Proposed System	3-4
	1.3 Scope of Project	5
2	**LITERATURE SURVEY**	**6-7**
3	**SYSTEM ANALYSIS**	**8-10**
	3.1 Hardware and Software Requirements	8
	3.2 Software Requirements Specification	9-10
4	**SYSTEM DESIGN**	**11-20**
	4.1 Description	11
	4.2 Architecture	12-14
	4.3 UML Diagrams	15-20
5	**METHODOLOGY**	**21-32**
	5.1 Technologies Used	21-24
	5.2 Modules Description	25-28
	5.3 Algorithms	29-32
6	**IMPLEMENTATION**	**33-55**
	6.1 Sample Code	33-47
	6.2 Output Screens	48-55
7	**CONCLUSION**	**56**
	BIBLIOGRAPHY	**57-58**

CHAPTER 1

INTRODUCTION

A DDoS attack aims to overwhelm a network, server, or website with a flood of traffic, rendering it inaccessible to legitimate users. Detecting DDoS attacks in networks involves examiningincoming traffic patterns for irregularities that might indicate an attack. Monitoring incoming trafficfor unusual patterns, such as sudden spikes in volume, unusual sources, or abnormal requests. The attack typically involves a large number of compromised computers, known as a botnet, which are controlled by the attacker. It can range from a small-scale attack that disrupts a single website or service for a short period of time, to a large-scale attack that affects multiple networks, services, or even an entire region. ML algorithms can help in identifying unusual spikes in traffic, distinguishing between legitimate and malicious activity, and improving the network's ability to respond swiftly to such attacks. The purpose of detecting a Distributed Denial of Service (DDoS) attack in networks is to identify and mitigate the attack as quickly as possible to minimize the impact on the targeted systemor network. By detecting a DDoS attack, network administrators can take measures to protect criticalresources and prevent them from being overwhelmed. This ensures that essential services and systems remain operational and accessible to legitimate users. DDoS attacks can significantly degrade network performance, causing latency, packet loss, and service disruptions. Detecting these attacks allows for immediate action to be taken to restore normal network performance and minimize the impact on users.

This paper delves into the critical need for a robust detection and classification system tailoredfor DDoS attacks in IoT environments. By exploring the challenges posed by these attacks, understanding their potential consequences, and emphasizing the dynamic nature of IoT networks, this research aims to contribute to the development of effective security measures.

1.1 Purpose and Objectives

The purpose of detecting a Distributed Denial of Service (DDoS) attack in networks is to identify and mitigate the attack as quickly as possible to minimize the impact on the targeted systemor network. By detecting a DDoS attack, network administrators can take measures to protect criticalresources and prevent them from being overwhelmed. This ensures that essential services and systems remain operational and accessible to legitimate users. DDoS attacks can significantly degrade network performance, causing latency, packet loss, and service disruptions. Detecting these attacks allows for immediate action to be taken to restore normal network performance and minimize the impact on users.

The implementation of a detection and classification system for Distributed Denial of Service(DDoS) attacks in networks is imperative due to several critical factors. As IoT devices continue toproliferate across various industries, their vulnerability to security threats increases significantly. These devices often possess limited computational resources and may have insecure configurations,making them attractive targets for malicious actors. Given the mission-critical nature of many IoT applications, such as healthcare and industrial systems, the unavailability of services resulting from DDoS attacks could have severe consequences, ranging from financial losses to compromised safety. Moreover, the dynamic and diverse nature of IoT networks demands specialized security measures to adapt to constant changes in device connections. A dedicated DDoS detection andclassification system is essential not only for preserving the integrity and availability of IoT servicesbut also for maintaining user trust, complying with regulations, and safeguarding sensitive data frompotential breaches. In essence, this system serves as a proactive defense mechanism to ensure the reliability and security of IoT ecosystems in the face of evolving cyber threats.

1.2 Existing and Proposed System

1.2.1 Existing System

In Existing system, a rule-based approach is used to detecting and classifying DDoS attacks in a network. The detection and classification of Distributed Denial of Service (DDoS) attacks in the networks is a complex task that necessitates an adaptive and efficient approach. A rule-based approach proves to be a foundational strategy in this context. Rule-based systems utilize predefined criteria or conditions to identify patterns indicative of DDoS activities in network traffic. In the IoT landscape, where device communication patterns may exhibit certain regularities, establishing specific rules becomes crucial. These rules can encompass parameters such as abnormal levels of incoming traffic, deviations in packet sizes, or irregularities in the timing and frequency of requests. By defining rules that reflect the normal behavior of the network, any deviation from these patterns triggers an alert or initiates counter measures, enabling the prompt identification and classification of potential DDoS attacks.

While the rule-based approach provides a straight forward and interpretable method for detecting known attack patterns, its effectiveness may be constrained by the dynamic nature of networks and the challenge of keeping rules updated to address emerging threats. Therefore, the integration of machine learning and anomaly detection alongside rule-based systems may offer a morecomprehensive solution, combining the advantages of predefined rules with the adaptability needed to counter evolving DDOS tactics in networks.

Advantages

- Rule based Algorithm have high complexity
- It also takes more time
- Less Accuracy
- It is not reaching the mile stone to detect the attack
- More challenging

1.2.2 Proposed System

DDoS attacks are a common threat to the network, although the attacker typically does not aimto steal any data. Basically, DDoS attacks aim at consuming system resources until the target is not available to offer its services. DDoS attacks could be divided into three categories: application layerattack, protocol attack and volumetric attack. For volumetric attack, an attacker can deplete the available resources of the victim or bandwidth towards the target. To implement this project, we havetrained AI algorithms such as Random Forest, SVM, KNN, Ada boost, Naïve bayes, XG Boost and this trained algorithm will predict DDOS Attack from test Data.

All algorithms trained on different types of DDOS attack dataset available on data repository. Here we used Random Forest algorithm to detect and classify the DDOS attack in Networks. RandomForest is an ensemble learning method that builds a multitude of decision trees and merges their outputs to improve accuracy and robustness. we gather labeled datasets containing both normal and DDoS-affected traffic. These datasets should represent typical network behavior. Then we apply random forest algorithm we detect and classifies the DDOS attack.

Advantages

- Random forest algorithm has low complexity
- It also takes less time compared to existing system and gives accuracy is up to 96%.
- Feature Importance: The algorithm provides feature importance scores, allowing for an analysis of which features contribute the most to the detection and classification of DDoS attacks.
- Random Forest can handle missing data well.

1.3 Scope of the project

A DDoS attack aims to overwhelm a network, server, or website with a flood of traffic, rendering it inaccessible to legitimate users. Detecting DDoS attacks in networks involves examiningincoming traffic patterns for irregularities that might indicate an attack. Monitoring incoming trafficfor unusual patterns, such as sudden spikes in volume, unusual sources, or abnormal requests.

The attack typically involves a large number of compromised computers, known as a botnet, which are controlled by the attacker. It can range from a small-scale attack that disrupts a single website or service for a short period of time, to a large-scale attack that affects multiple networks, services, or even an entire region. ML algorithms can help in identifying unusual spikes in traffic, distinguishing between legitimate and malicious activity, and improving the network's ability to respond swiftly to such attacks.

CHAPTER 2

LITERATURE SURVEY

[1] Douligeris, C., & Mitrokotsa, A.DDoS attacks and defense mechanisms: classification and state-of-the-art. Computer Networks, 44(5), 643-666[2014]

This paper provides a classification of DDoS attacks and an overview of existing defense mechanisms, including machine learning-based detection approaches.

[2] Alrajeh, N., & Khan, S. U. Machine learning techniques for DDoS attack detection and mitigation. Journal of Network and Computer Applications, 34(1), 98-116 [2015]

This comprehensive review discusses various machine learning techniques applied to detect and mitigate DDoS attacks, covering both supervised and unsupervised learning approaches.

[3] Fadlullah, Z. M., Fouda, M. E., Takeuchi, A., & Kato, N. Toward intelligent mitigation of DDoS attacks in ISP networks. IEEE Network, 24(3), 10-16[2015]

This paper discusses intelligent mitigation techniques for DDoS attacks in ISP networks, which may include machine learning algorithms for real-time detection and response.

[4] Kim, H., & Kim, K.Deep learning based network intrusion detection systems: A review. In 2016 International Conference on Platform Technology and Service (PlatCon) (pp. 1-5)[2016]

This paper provides a review of deep learning-based network intrusion detection systems, which have shown promise in detecting sophisticated DDoS attacks by automatically learning features from network traffic data.

[5] Ahmed, M., Mahmood, A. N., & Hu, J. A survey of network anomaly detection techniques. Journal of Network and Computer Applications, 60, 19-31[2016]

While not specifically focused on DDoS attacks, this survey provides an overview of various anomaly detection techniques, which are commonly used for detecting unusual patterns indicative of DDoS activity.

[6] Machine Learning Techniques for DDoS Attack Detection: A Survey by Author A et al. [2019]

This survey provides an overview of various machine learning techniques employed in DDoS attack detection, highlighting their strengths, weaknesses, and application scenarios.

[7] A Comprehensive Survey on DDoS Attack Detection Techniques Using Machine Learning Approaches by Author B et al. [2020]

This survey systematically reviews the literature on DDoS attack detection using machine learning, categorizing techniques based on their underlying algorithms and discussing their effectiveness.

[8] Machine Learning-Based DDoS Attack Detection: A Review of Recent Advance by Author C et al. [20020]

This review paper discusses recent advancements in machine learning-based DDoS attack detection, including novel algorithms, feature selection methods, and evaluation frameworks.

[9] Survey of Machine Learning Approaches for DDoS Attack Detection in Cloud Computing Environments by Author D et al. [2022]

Focusing on cloud computing environments, this survey evaluates the effectiveness of machine learning approaches in detecting DDoS attacks and addresses challenges specific to cloud-based systems.

[10] A Survey on Machine Learning Techniques for DDoS Attack Detection in Internet of Things (IoT) Networks by Author E et al.[2022]

This survey explores the application of machine learning techniques for detecting DDoS attacks in IoT networks, considering the unique characteristics and challenges of IoT environments.

[11].Machine Learning-Based DDoS Attack Detection in Software-Defined Networking: A Survey by Author F et al. [2023]

This survey reviews the literature on machine learning-based DDoS attack detection in Software-Defined Networking (SDN) environments, highlighting the integration of machine learning with SDN principles.

CHAPTER 3

SYSTEM ANALYSIS

System requirements are the functionality that is needed by a system in order to satisfy the requirements. System requirements are abroad and a narrow subject that could be implemented to many items. The requirements document allows the project team to have a clear picture of what the software solution must do before selecting a vendor. Without an optimized set of future state requirements, the project team has no effective basis to choose the best system for your organization.

3.1 Hardware and Software Requirements

3.1.1 Hardware Requirements

The hardware requirement specifies each interface of the software elements and thehardware elements of the system. These hardware requirements include configuration characteristics.

Processor	Intel i3
Ram	4 GB
Hard disk	250GB
Mouse	Logitech

3.1.2 Software Requirements

Operating system	Windows
Coding Language	python
IDE	Spyder

3.2 Software Requirements Specification

3.2.1 Functional Requirements

Functional Requirements to detect DDOS attacks in networks using Machine Learning:

1. High Traffic Volume: DDoS attacks require a large volume of traffic to overwhelm the target network or system. This can be achieved by utilizing a botnet or by employing amplification techniques.

2. Distributed Attack: DDoS attacks involve multiple sources or nodes attacking the target simultaneously. This distribution of attack traffic makes it difficult for the target to differentiate between legitimate and malicious traffic.

3. Attack Duration: DDoS attacks typically last for an extended period to maximize the impact on the target. The longer the attack duration, the more resources it consumes and the longer the target remains inaccessible.

3.2.2 Non-Functional Requirements

Describe user-visible aspects of the system that are not directly related to the functional behavior of the system. Non-Functional requirements include quantitative constraints, such asresponse time or accuracy.

Non-functional requirements to detect DDOS attacks in networks using Machine Learning:

- Performance – Response time and Throughput
- Scalability – Data scaling and user scaling
- Reliability – Availability, Fault Tolerance
- Security – Data Encryption, Access Control, Authentication and Authorization
- Maintainability – Code Maintainability, Documentation
- Usability – User Interface Design, Training and Support
- Compatibility – Browser Compatibility, Data Format Compatibility
- Performance Testing – Load Testing, Stress Testing
- Interoperability - Integration

1. Scalability: The solution should be able to handle a large number of network devices and traffic flows without impacting performance or accuracy in detecting DDoS attacks.

2. Real-time detection: The system should be able to detect DDoS attacks in real-time, minimizing the response time to mitigate the attack and reduce the impact on the network.

3. Accuracy: The solution should have a low false positive rate, accurately distinguishing between normal network traffic and DDoS attacks. It should also have a low false negative rate, ensuring that all DDoS attacks are detected.

4. Adaptability: The system should be able to adapt to changing network conditions, adjusting its detection algorithms and thresholds to account for variations in traffic patterns and attack techniques.

CHAPTER 4

SYSTEM DESIGN

4.1 Description

System design is the process of creating a system's architecture, parts, and interfaces to ensurethat it satisfies the needs of its users. Thus, in order to examine the design of this project, we first go through the specifics of establishing the concept of DDOS detection through a few fundamental modules that would clearly describe the workings of the system that would come from the development.

MACHINE LEARNING

Machine learning (ML) is an area of artificial intelligence (AI) that enables computers to "learn"for themselves over time from training data and develop without explicit programming. Data patternscan be found by machine learning algorithms, which can then use this information to learn and develop their own predictions. Algorithms and models used for machine learning, in essence, gain experience.

In contrast, machine learning is a process that is automated and gives computers the ability to solve issues with little to no human involvement and make decisions based on prior experiences. While machine learning and artificial intelligence are frequently used synonymously, they are actually two distinct ideas. Machine learning, a subset of AI that enables intelligent systems to autonomously learn new things from data, is what allows intelligent systems to make decisions, gainnew abilities, and solve problems in a way that is similar to people. AI is the more general notion. Machine learning methods are used in image classification to examine the existence of objects in a picture and classify it.

The foundation of computer vision and image recognition is this specific problem Machines don't examine an image in its entirety. They just examine pixel patterns or vectors while analyzing apicture. The elements will subsequently be classified and given labels in accordance with the variousrules that were established during algorithm configuration.

4.2 System Architecture

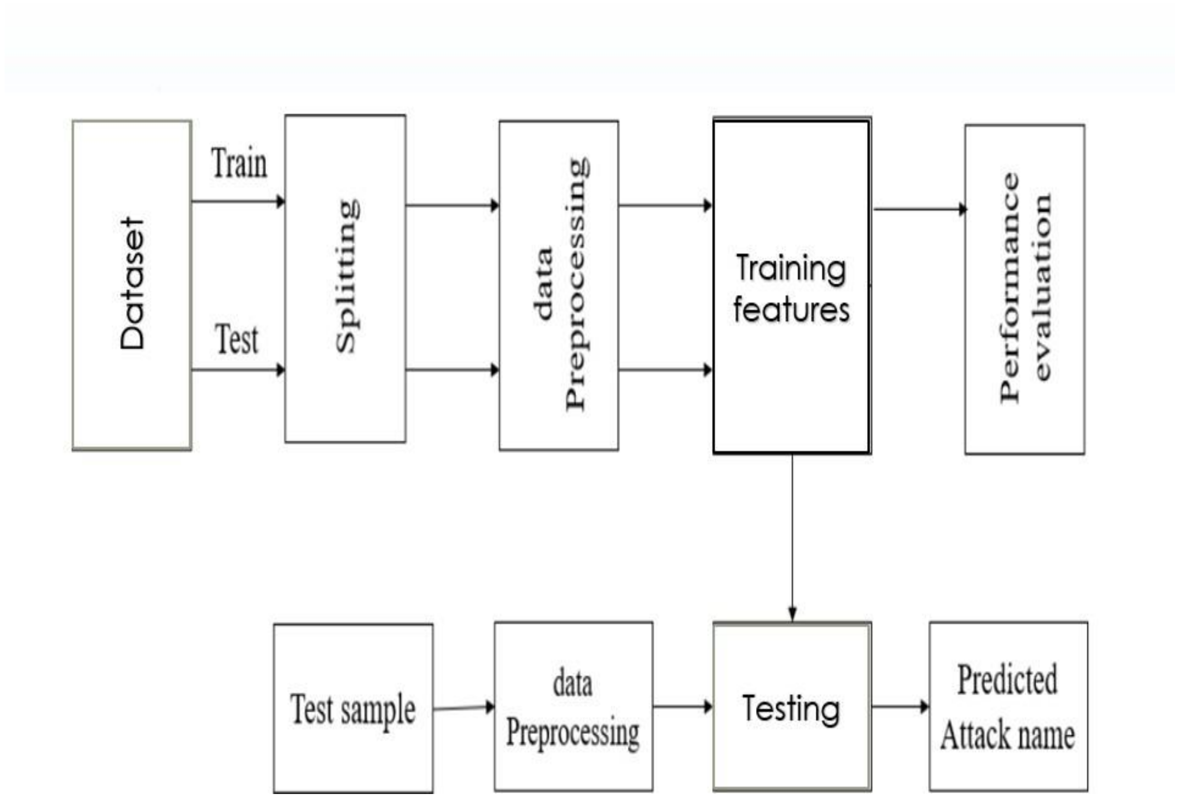


Fig 1: System Architecture

- **Dataset:** Collect labeled network traffic data, where each instance is labeled as either normal or DDoS attack.

  The dataset includes 23 features in total where some of the data is extracted from the switches and others were calculated. Extracted features which are present in the dataset include: -

  Packet_count – refers to the count of packets

  byte_count – refers to the count of bytes in the packet

  Switch-id – ID of the switch

  duration_sec – packet transmission (in seconds)

  duration_nsec – packet transmission (in nanoseconds)

  Source IP – IP address of the source machine

Destination IP – IP address of the destination machine

Port Number – Port number of the application

tx_bytes – number of bytes transferred from the switch port

rx_bytes – number of bytes received on the switch port

dt field – shows the date and time which has been converted into a number and the flow is monitored at a monitoring interval of 30 seconds

The calculated features present in the dataset include:

Byte Per Flow – byte count during a single flow

Packet Per Flow – packet count during a single flow

Packet Rate – number of packets transmitted per second and calculated by dividing the packet per flow by monitoring interval

number of Packet_ins messages – messages that are generated by the switch and is sent to the controller

Flow entries of switch – entries in the flow table of a switch which is used to match and process packets

tx_kbps – Speed of packet transmission (in kbps)

rx_kbps - Speed of packet reception (in kbps)

Port Bandwidth – Sum of tx_kbps and rx_kbps

The output feature is the last column of the dataset i.e. class label which classifies the traffic type to be benign or malicious. The malicious traffic is labelled as 1 and the benign traffic is labelled as 0. The simulation of the network was run for approximately 250 minutes and 1,04,345 instances of data were collected and recorded. Further, the simulation was run for a given interval to collect more instances of data.

- **Split Data:** Split the dataset into training and testing sets to evaluate the performance of different algorithms.
- **Train Models**: Train each algorithm using the training data, optimizing parameters as necessary.

- **Evaluate Performance:** Calculate performance metrics such as accuracy, precision, recall, and F1-score for each algorithm using the values from the confusion matrix.
- **Test Sample:** Using this module we will upload test data and then machine learning models will predict type of attack from that test data.
- **Predictions:** Make predictions on the testing data using each trained model.

4.3 UML Diagrams

UML Diagrams are classified into different types such as

1. Data Flow Diagram
2. Use Case Diagram
3. Class Diagram
4. Sequence Diagram
5. Activity Diagram
6. Deployment Diagram

1. Data Flow Diagram

A data-flow diagram is a visual representation of how data moves through a system or a process(usually an information system). The data flow diagram also shows the inputs and outputs of each entity as well as the process itself. A data-flow diagram lacks control flow, loops, and decision- makingprocesses.

With a flowchart, certain operations based on the data can be depicted. data flow diagram (DFD)maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination.

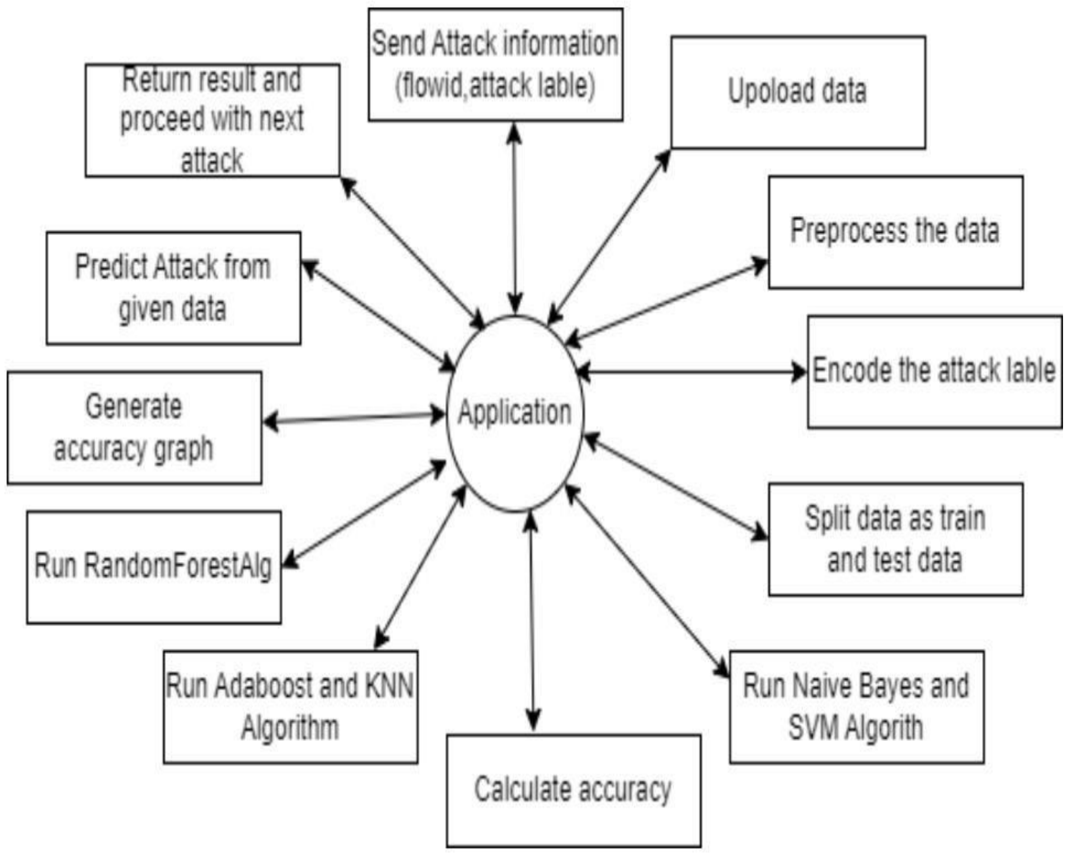

Return result and proceed with next attack
Send Attack information (flowid,attack lable)
Upoload data
Preprocess the data
Predict Attack from given data
Encode the attack lable
Generate accuracy graph
Application
Run RandomForestAlg
Split data as train and test data
Run Adaboost and KNN Algorithm
Run Naive Bayes and SVM Algorith
Calculate accuracy

Fig 2: Dataflow Diagram

Fig.2 The above figure represents the data flow diagram of DDos detection. It consists of several steps which Send Attack information (flowid attack label), Upload data, Preprocess the data, Predict Attack from given data, Generate accuracy graph, Application, Split data as train and test data, Encode the attack label, Run RandomForestAlg, Run Adaboost and KNN Algorithm, Calculate accuracy, Run Naive Bayes and SVM Algorithm Return result and proceed with next attack.

2. Use Case Diagram

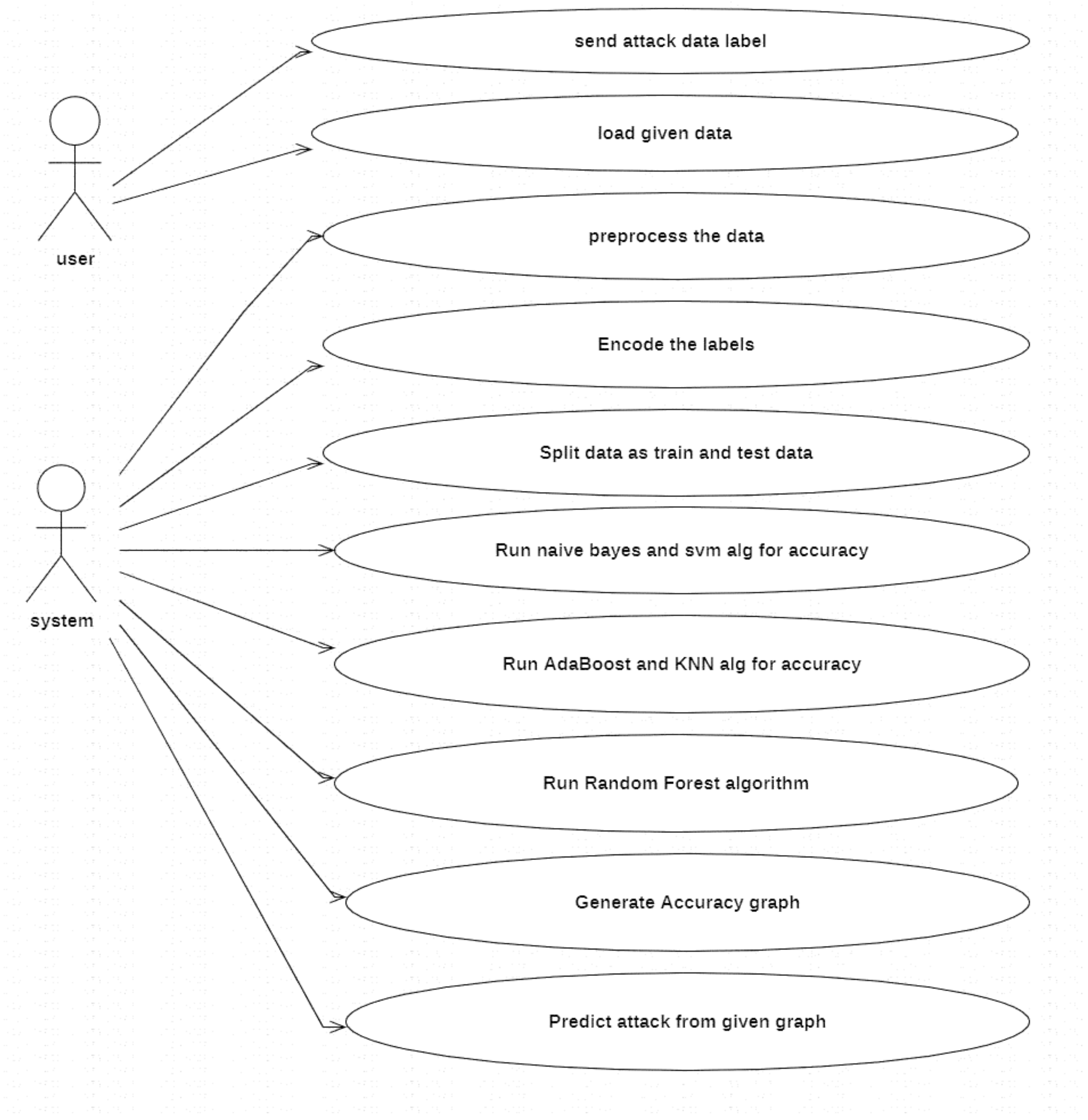

send attack data label
load given data
user
preprocess the data
Encode the labels
Split data as train and test data
Run naive bayes and svm alg for accuracy
system
Run AdaBoost and KNN alg for accuracy
Run Random Forest algorithm
Generate Accuracy graph
Predict attack from given graph

Fig 3: Use Case Diagram

Fig 3: To depict a system's dynamic behavior, use case diagrams are often employed. Using use cases, actors, and their interactions, it captures the functionality of the system. A system or subsystem of an application's necessary duties, services, and operations are modelled. It shows a system's high-level functionality as well as how a user interacts with that system.

3. Class Diagram

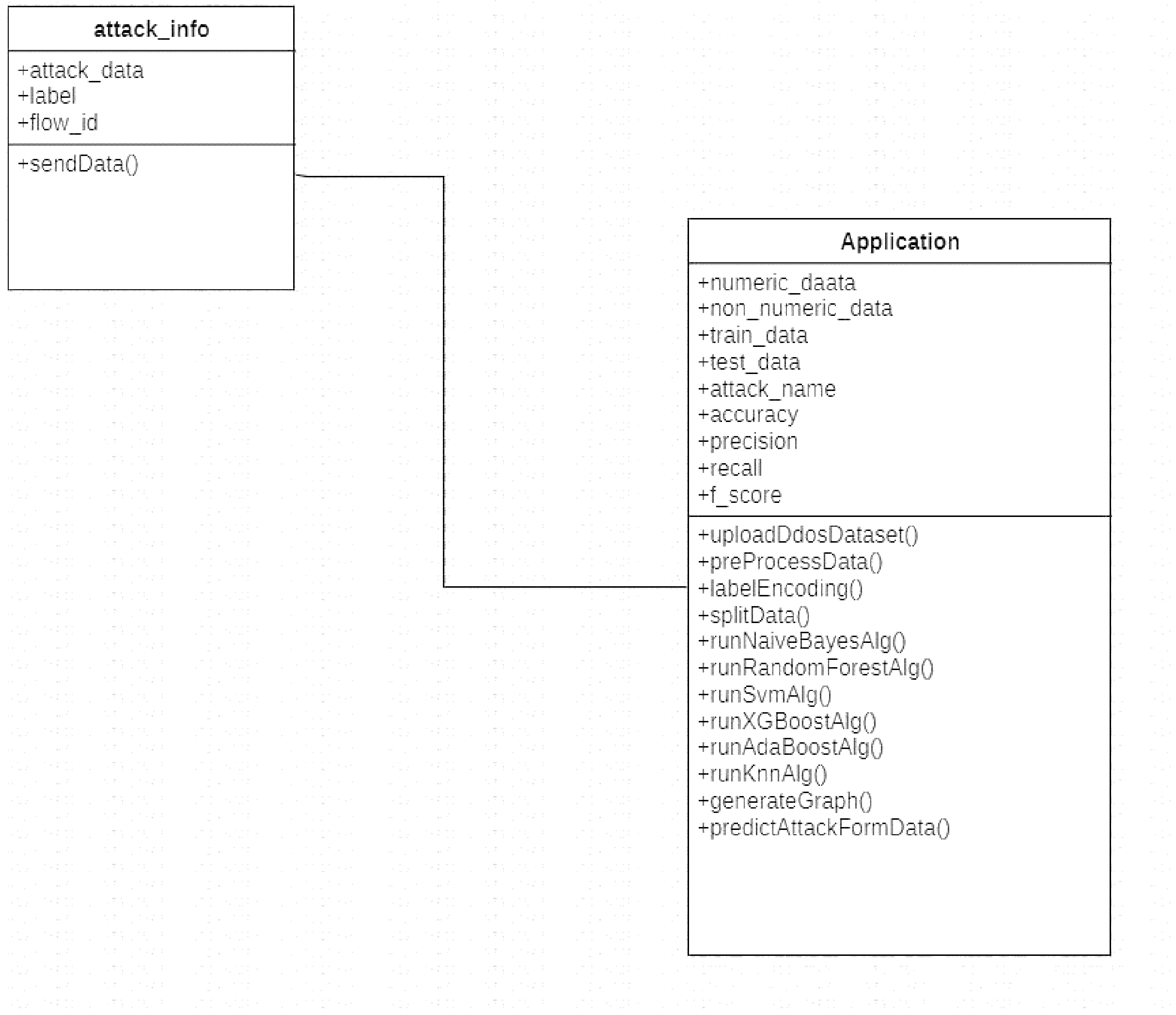


Fig 4: Class Diagram

Fig 4: Class diagrams are the main building block of any object-oriented solution. It displays a system's classes, along with each class's properties, operations, and relationships to other classes. Most modelling tools include three elements to a class. Name is at the top, followed by attributes, then operations or methods, and finally, methods. Classes are linked together to generate class diagrams in a complex system with numerous related classes. Various sorts of arrows represent different relationships between classes.

4.Sequence Diagram

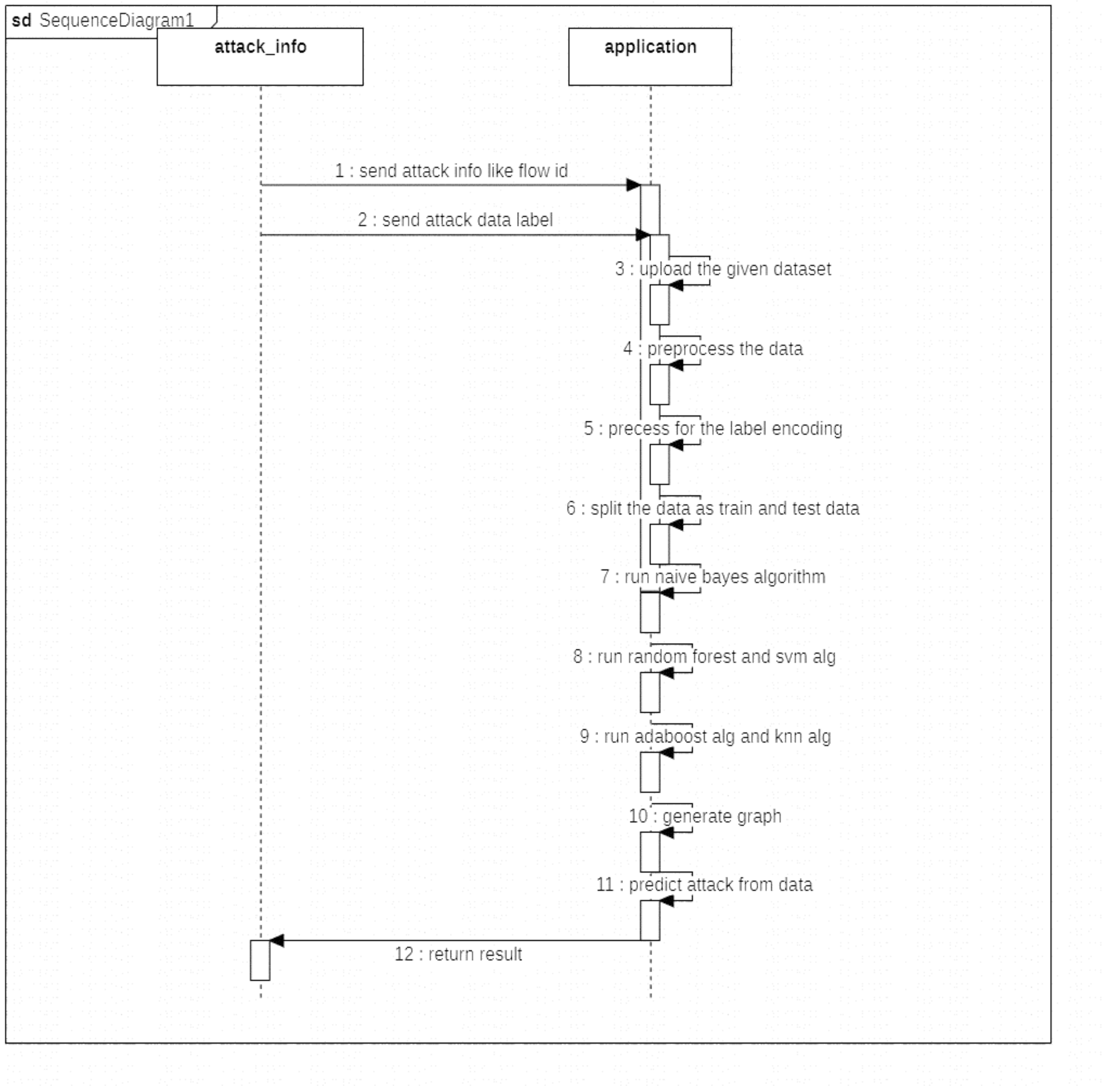

sd SequenceDiagram1
attack_info
application
1 : send attack info like flow id
2 : send attack data label
3 : upload the given dataset
4 : preprocess the data
5 : precess for the label encoding
6 : split the data as train and test data
7 : run naive bayes algorithm
8 : run random forest and svm alg
9 : run adaboost alg and knn alg
10 : generate graph
11 : predict attack from data
12 : return result

Fig.5: Sequence Diagram

Fig.5 In UML, sequence diagrams display how and in what order certain it interact with one another. It's crucial to remember that they depict the interactions for a certain circumstance. The interactions are depicted as arrows, while the processes are portrayed vertically. The objective of sequence diagrams and their fundamentals are explained in this article. To understand more about sequence diagrams, you may also look at this comprehensive tutorial.

5.Activity Diagram

.

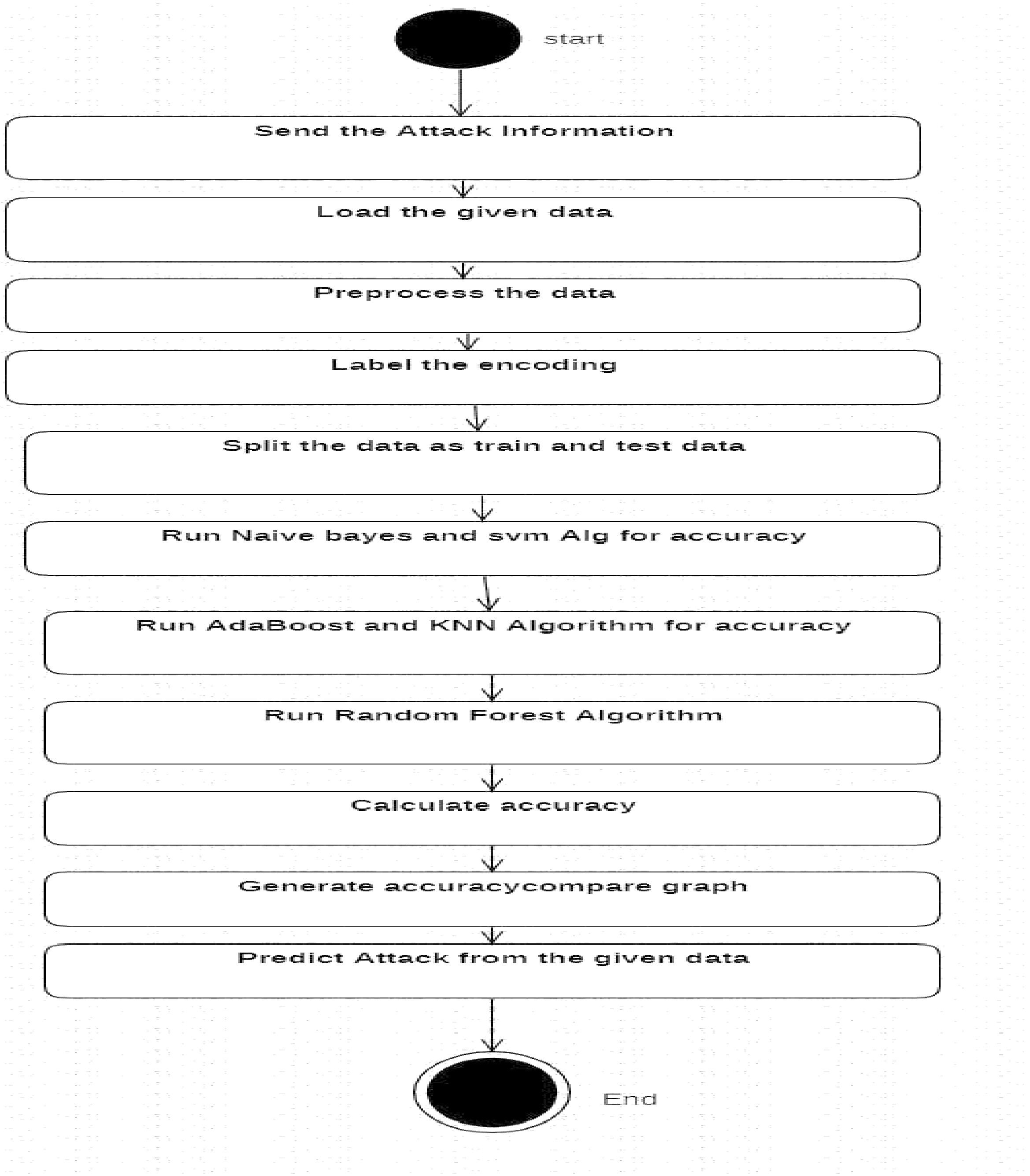

start
Send the Attack Information
Load the given data
Preprocess the data
Label the encoding
Split the data as train and test data
Run Naive bayes and svm Alg for accuracy
Run AdaBoost and KNN Algorithm for accuracy
Run Random Forest Algorithm
Calculate accuracy
Generate accuracycompare graph
Predict Attack from the given data
End

Fig 6: Activity Diagram

Fig 6 The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions

5. Deployment Diagram

The deployment diagram visualizes the physical hardware on which the software will be displayed.

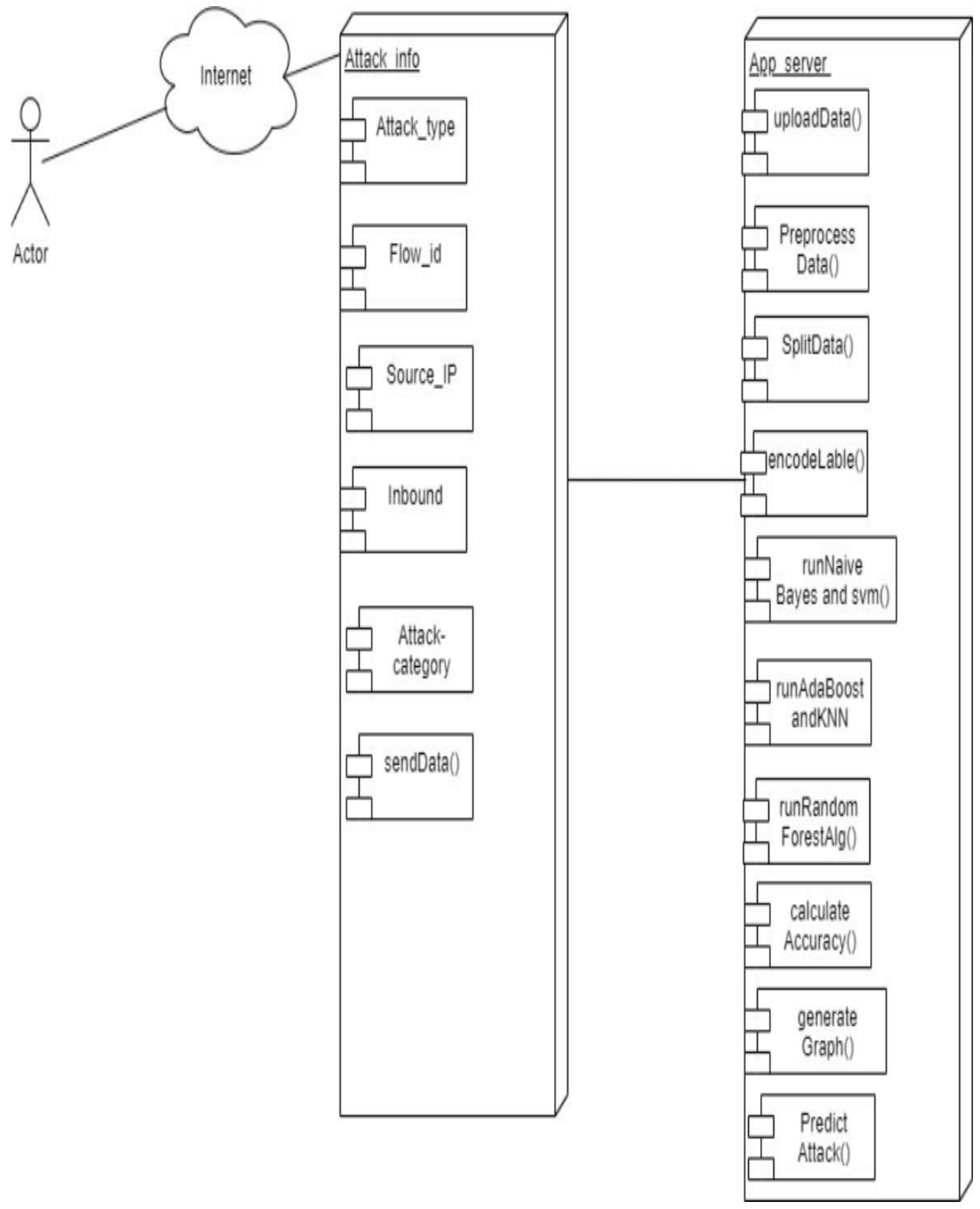


Fig 7: Deployment Diagram

Fig:7 The deployment diagram visualizes the physical hardware on which the software will be displayed.

CHAPTER 5
METHODOLOGY

5.1 Technologies Used

Machine learning (ML) technologies play a significant role in detecting DDoS attacks in networks by analyzing traffic patterns and identifying anomalous behavior. Here are some ML technologies commonly used for DDoS detection:

Supervised Learning:

Support Vector Machines (SVM): SVM classifiers can be trained on labeled datasets to distinguish between normal and malicious traffic patterns.

Random Forest: Random Forest algorithms can analyze various features of network traffic to detect patterns associated with DDoS attacks.

Logistic Regression: Logistic regression models can be trained to classify network traffic as benign or malicious based on input features.

Unsupervised Learning:

K-means Clustering: K-means clustering can group network traffic data into clusters, helping identify unusual patterns that may indicate DDoS attacks.

Density-Based Clustering: Algorithms like DBSCAN (Density-Based Spatial Clustering of Applications with Noise) can detect outliers in network traffic, which may indicate DDoS attacks.

Feature Engineering:

Statistical Features: ML models can be trained on statistical features extracted from network traffic data, such as packet size distributions, inter-arrival times, and protocol distributions.

Flow-Based Features: Features derived from network flow data, such as source and destination IP addresses, port numbers, and packet counts, can be used to characterize network traffic and detect anomalies indicative of DDoS attacks.

Incremental Learning: ML models can be continuously updated with new network traffic data in real-time, allowing them to adapt to evolving DDoS attack patterns and maintain high detection accuracy.

By leveraging these ML technologies, organizations can build sophisticated DDoS detection systems capable of accurately identifying and mitigating attacks in real-time, thereby enhancing the security and resilience of their networks.

TensorFlow

TensorFlow is a free and open-sources of tware library for data flow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

NumPy

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A power full N-dimensional array object
- Sophisticated(broadcasting)functions
- Tools for integrating C/C++and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. Itha very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to MATLAB users.

Scikit–learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted− Python is processed at run time by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code.

5.2 Modules Description

MODULES

To implement this project, we have designed following modules:

• DATA COLLECTION

• DATA PRE-PROCESSING

• BUILD CONFUSION MATRIX USING ALGORITHMS

• PERFORMANCE GRAPH

• TEST MODEL

DATA COLLECTION

▪ It is the process in which information is gathered. Collect labeled network traffic data, where each instance is labeled as either normal or DDoS attack. This module gathers data, including information about the different type of DDos attacks, address and other relevant factors.

▪ Data collection involves gathering various types of network traffic data, including packet headers, flow data, and logs from network devices. This data is then analyzed using techniques such as anomaly detection, signature-based detection, and machine learning to identify patterns indicativeof a DDoS attack. Dataset contain source IP, destination IP, Destination port, timestamp and flow ids.

▪ **Dataset:** Collect labeled network traffic data, where each instance is labeled as either normal or DDoS attack.

The dataset includes 23 features in total where some of the data is extracted from the switches and others were calculated. Extracted features which are present in the dataset include: -

Packet_count – refers to the count of packets

byte_count – refers to the count of bytes in the packet

Switch-id – ID of the switch

duration_sec – packet transmission (in seconds) duration_nsec – packet transmission (in nanoseconds)

Source IP – IP address of the source machine

Destination IP – IP address of the destination machine

Port Number – Port number of the application

tx_bytes – number of bytes transferred from the switch port

rx_bytes – number of bytes received on the switch port

dt field – shows the date and time which has been converted into a number and the flow is monitored at a monitoring interval of 30 seconds

The calculated features present in the dataset include:

Byte Per Flow – byte count during a single flow

Packet Per Flow – packet count during a single flow

Packet Rate – number of packets transmitted per second and calculated by dividing the packet per flow by monitoring interval

number of Packet_ins messages – messages that are generated by the switch and is sent to the controller

Flow entries of switch – entries in the flow table of a switch which is used to match and process packets

tx_kbps – Speed of packet transmission (in kbps)

rx_kbps - Speed of packet reception (in kbps)

Port Bandwidth – Sum of tx_kbps and rx_kbps

The output feature is the last column of the dataset i.e. class label which classifies the traffic type to be benign or malicious. The malicious traffic is labelled as 1 and the benign traffic is labelled as 0. The simulation of the network was run for approximately 250 minutes and 1,04,345 instances of data were collected and recorded. Further, the simulation was run for a given interval tocollect more instances of data.

DATA PRE-PROCESSING

Before analysis, the collected data often requires preprocessing. This module involves cleaningthe data, handling missing values, normalizing or standardizing features. Split the dataset into trainingand testing sets to evaluate the performance of different algorithms.

- Identify and handle missing values in the dataset by replacing missing values with estimated values, deletion of records with missing values.
- Ensure consistency in data formats and units across different variables by converting date and time variables into a standard format.
- Removing duplicates, Data transformation, Data splitting as test data and train data.

BUILD CONFUSION MATRIX USING ALGORITHMS

Select different algorithms commonly used for DDoS attack detection, such as:

- Random Forest

- Support Vector Machine (SVM)

- k-Nearest Neighbors (k-NN)

- Decision Tree

Calculate the confusion matrix for each algorithm, comparing the predicted labels to the true labels:

Predicted Normal | Predicted DDoS Attack

- Actual Normal -True Negatives | False Positives
- Actual DDoS Attack -False Negatives | True Positives

- True Positives (TP): Instances correctly classified as DDoS attacks.

- False Positives (FP): Instances incorrectly classified as DDoS attacks when they are normal.

- True Negatives (TN): Instances correctly classified as normal traffic.

- False Negatives (FN): Instances incorrectly classified as normal traffic when they are DDoS attacks.

PERFORMANCE GRAPH

- Using this module we calculate prediction accuracy.
- Comparison Graph: using this module we will display comparison table and graph of all algorithms Accuracy, F1 score, Precision, Recall. Calculate performance metrics such as accuracy, precision, recall, and F1-score for each algorithm using the values from the confusion matrix.
- Compare the performance of different algorithms based on their confusion matrices and performance metrics to determine the most effective algorithm for DDoS attack detection in your specific context.

TEST MODEL

- **Predict Attack from Test Data:** Using this module we will upload test data and then machine learning models will predict type of attack from that test data.

5.3 Algorithms

5.3.1 Random Forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

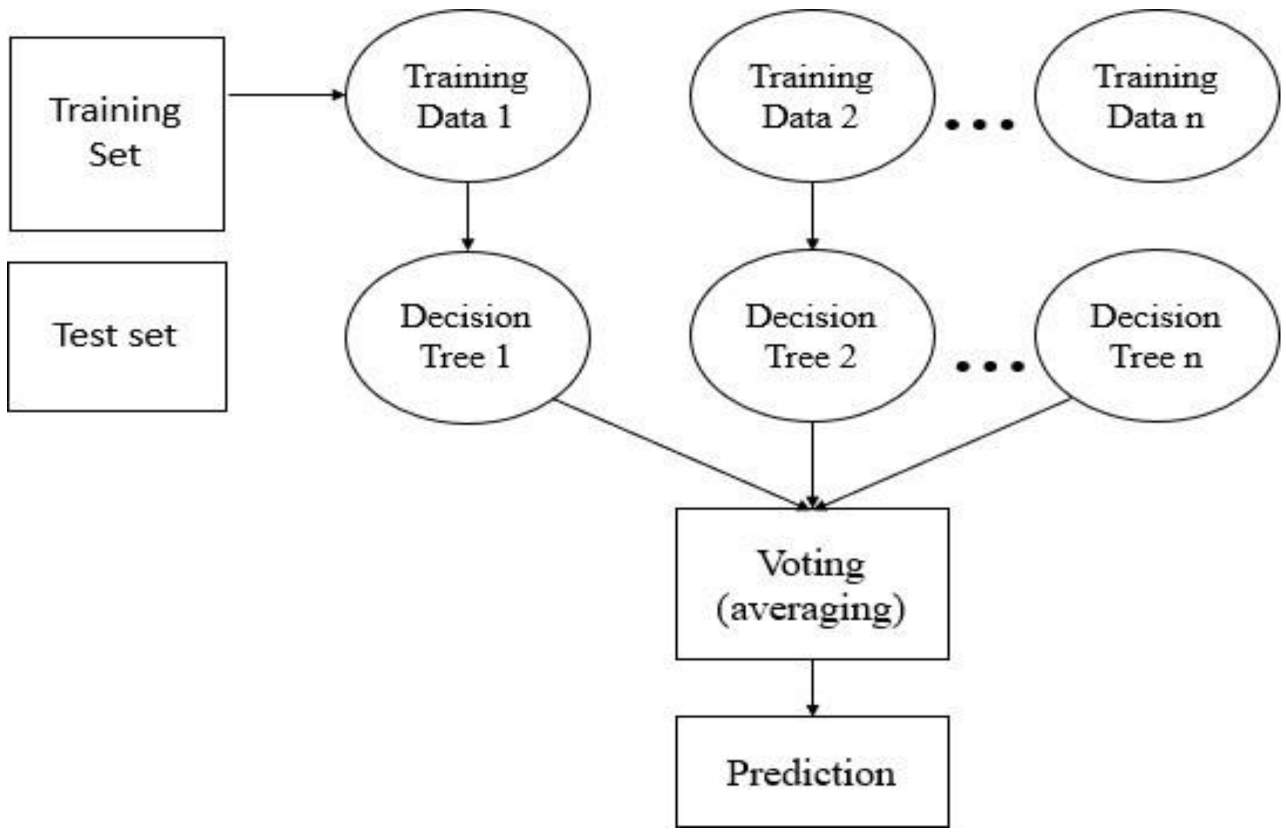


Fig 8: Random Forest algorithm

5.3.2 Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well it’s best suited for classification. The main objective of the SVM algorithm is to find the optimal hyperplane in an N dimensional space that can separate the data points in different classes in the feature space. The hyperplane tries that the margin between the closest points of different classes should be as maximum as possible. The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three.

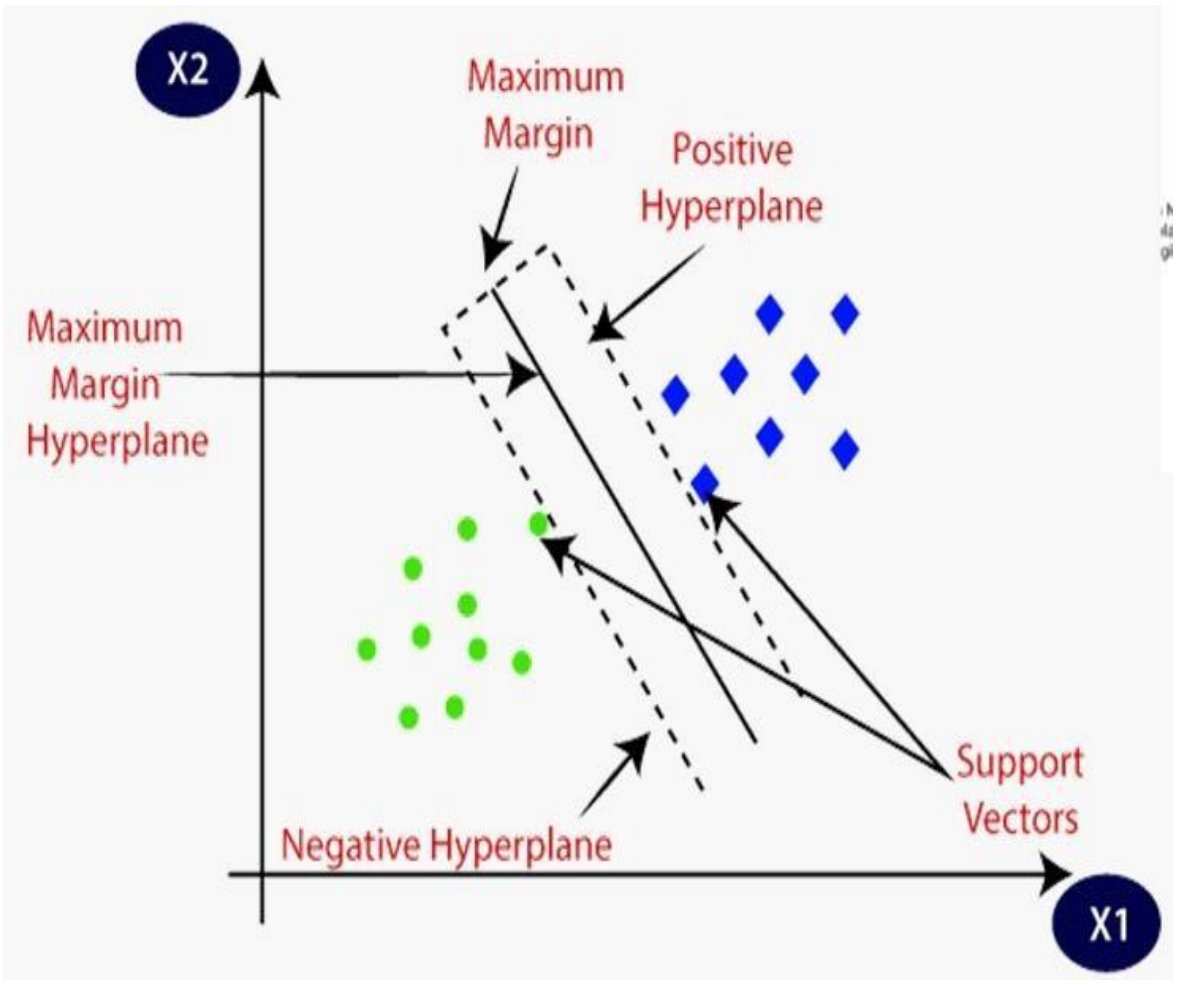


Fig 9: Support Vector Machine

5.3.3 Naive Bayes Algorithm

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is mainly used in text classification that includes a high dimensional training dataset. Naïve Bayes Classifier is one of the simple and most effective Classification algorithms whichhelps in building the fast machine learning models that can make quick predictions. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

5.3.4 K-Nearest Neighbour

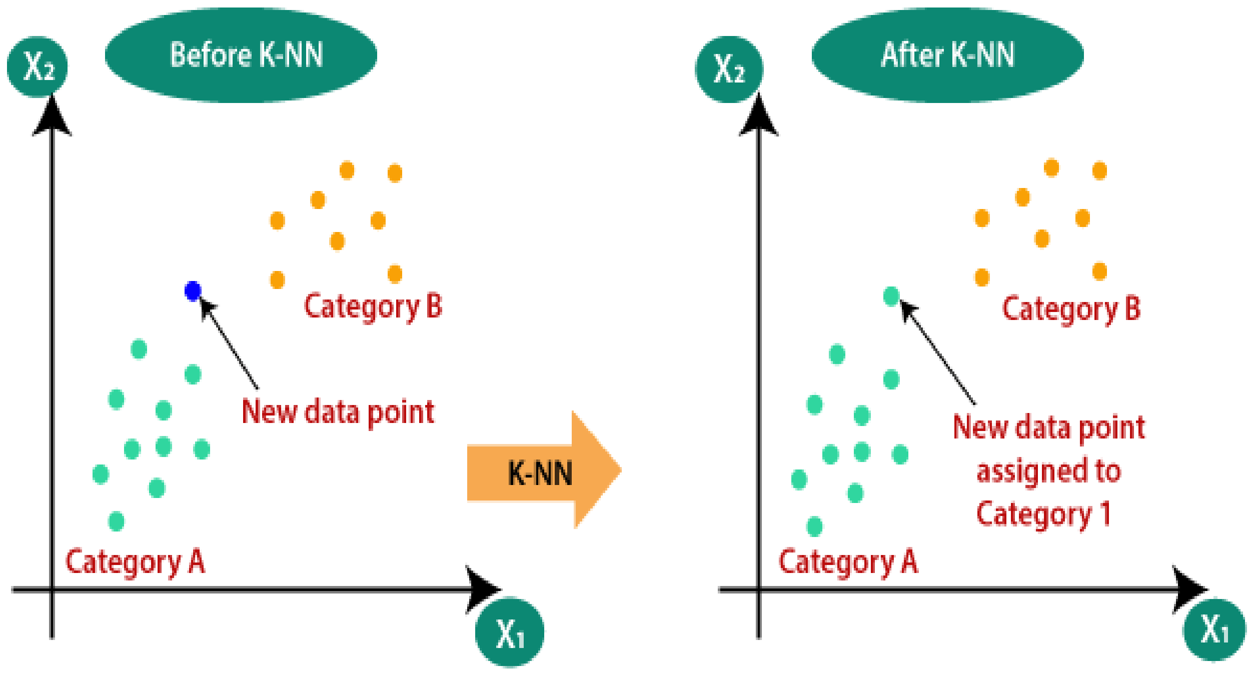


Fig 10: K-Nearest Neighbour

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using KNN algorithm. K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems. K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

5.3.5. XG Boost Algorithm

XG Boost is an optimized distributed gradient boosting library designed for efficient and scalable training of machine learning models. It is an ensemble learning method that combines the predictions of multiple weak models to produce a stronger prediction. XG Boost stands for “ExtremeGradient Boosting” and it has become one of the most popular and widely used machine learning algorithms due to its ability to handle large datasets and its ability to achieve state-of-the- art performance in many machine learning tasks such as classification and regression. One of the key features of XGBoost is its efficient handling of missing values, which allows it to handle real-worlddata with missing values without requiring significant pre-processing. Additionally, XG Boost has built-in support for parallel processing, making it possible to train models on large datasets in a reasonable amount of time.

5.3.6. AdaBoost Algorithm

AdaBoost algorithm, short for Adaptive Boosting, is a Boosting technique used as an Ensemble Method in Machine Learning. It is called Adaptive Boosting as the weights are re-assigned to each instance, with higher weights assigned to incorrectly classified instances.

CHAPTER 6

IMPLEMENTATION

6.1 Sample Code

Imports:

```
from tkinter import messagebox
from tkinter import *
from tkinter.simpledialog import tkinter
from tkinter import filedialog
from tkinter.filedialog import askopenfilename
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import os
from sklearn.metrics import confusion_matrix
from sklearn.naive_bayes import GaussianNB
from sklearn import svm
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
import seaborn as sns
import webbrowser
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
import pickle
from sklearn.neighbors import KNeighborsClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import AdaBoostClassifier
  from sklearn.preprocessing import normalize
from sklearn.decomposition import PCA
```

- This section imports necessary libraries and modules required for building the GUI application, data manipulation, machine learning model creation, evaluation metrics, and visualization.

Global Variables Declaration:

```
global filename
global X, Y
global dataset
global main
global text
global accuracy, precision, recall, fscore
global X_train, X_test, y_train, y_test
global classifier
global label_encoder, labels, columns, types, pca
```

- This block declares global variables that will be used throughout the script.

Creating Main Window:

```
main = tkinter.Tk()
main.title("Detection and classification of DDOS Attack In IOT Network Environment")
main.geometry("1300x1200")
```

- This code initializes the main window for the application with a specific title and dimensions.

Function to Get Label Index:

```
def getLabel(name):
    label = -1
    for i in range(len(labels)):
        if name == labels[i]:
            label = i
            break
    return label
```

- This function takes a label name and returns its corresponding index.

Function Definition:

```
def uploadDataset():
```

- This line defines a function named uploadDataset().

Global Variables:

python

global filename, dataset, label

- This line declares global variables filename, dataset, and labels. These variables are declared as global to ensure they can be accessed and modified within the function.

Clearing Text Widget:

text.delete('1.0', END)

- This line clears any existing text in a text widget named text. This is done using the delete() method, which deletes text starting from the first character (line 1, character 0) to the end (END).

Opening File Dialog:

filename = filedialog.askdirectory(initialdir=".")

- This line opens a file dialog window using the askdirectory() method from the filedialog module. It allows the user to select a directory from which files will be loaded. The initialdir parameter specifies the initial directory to be displayed in the dialog.

Inserting Loaded Directory Information:

text.insert(END, filename + " loaded\n\n")

- After the user selects a directory, this line inserts a message into the text widget indicating that the directory has been loaded.

Reading CSV Files:

df1 = pd.read_csv(filename + "/DrDOS_DNS.csv")

df2 = pd.read_csv(filename + "/DrDOS_LDAP.csv")

# Reading other CSV files...

- This block reads multiple CSV files from the selected directory using pd.read_csv() from the pandas library. Each CSV file is read into a separate DataFrame (df1, df2, etc.).

Concatenating DataFrames:

dataset = [df1, df2, df3, df4, df5, df6, df7, df8, df9, df10]

dataset = pd.concat(dataset)

- After reading all CSV files, the DataFrames are concatenated along the rows using pd.concat() to create a single DataFrame named dataset.

Extracting Labels:

labels = np.unique(dataset['Label']).tolist()

- This line extracts unique labels from the 'Label' column of the dataset DataFrame using np.unique(). It then converts the result to a Python list using .tolist() method and assigns it to the labels variable.

Displaying Dataset Information:

text.insert(END, str(dataset))

- This line inserts the string representation of the dataset DataFrame into the text widget.

Visualizing Attack Types:

```
attack = dataset.groupby('Label').size()
attack.plot(kind="bar")
plt.xlabel('DDOS Attacks')
plt.ylabel('Number of Records')
plt.title('Different Attacks found in dataset')
plt.show()
```

- This block groups the dataset by the 'Label' column and calculates the size of each group. It then creates a bar plot to visualize the distribution of different attack types in the dataset.

Overall, this function allows the user to select a directory containing CSV files, reads and concatenates those files into a single dataset, extracts unique labels, displays dataset information, and visualizes the distribution of attack types within the dataset.

Function Definition:

def preprocessDataset():

- This line defines a function named preprocessDataset().

Global Variables:

```
global dataset, label_encoder, X, Y, columns, types, pca
global X_train, X_test, y_train, y_test
```

- This line declares global variables that will be used within the function.

Clearing Text Widget:

```
text.delete('1.0', END)
```

- This line clears any existing text in a text widget named text.

Initializing Label Encoder List:

```
label_encoder = []
```

- This line initializes an empty list label_encoder which will store instances of the LabelEncoder class for each categorical column.

Extracting Column Names and Types:

```
columns = dataset.columns
types = dataset.dtypes.values
```

- These lines extract the column names and their respective data types from the dataset DataFrame.

Label Encoding Categorical Columns:

```
for i in range(len(types)):
    name = types[i]
    if name == 'object' and columns[i] != 'Label':
        le = LabelEncoder()
        dataset[columns[i]] = pd.Series(le.fit_transform(dataset[columns[i]].astype(str)))
        label_encoder.append(le)
        print(columns[i])
```

- This loop iterates over each column in the dataset and checks if its data type is 'object' (categorical). If it is, it applies label encoding using LabelEncoder. The encoded values are stored back into the dataset, and the LabelEncoder instance is appended to the label_encoder list.

Handling Missing Values:

```
dataset.fillna(0, inplace=True)
```

- This line fills any missing values in the dataset with zero.

Processing Target Variable (Y):

```
Y = dataset['Label'].ravel()
```

- This line extracts the target variable 'Label' from the dataset and converts it into a flattened array using ravel().

Mapping Labels to Numerical Values:

```
temp = []
for i in range(len(Y)):
    temp.append(getLabel(Y[i]))
```

- This loop iterates over each label in Y and maps it to a numerical value using the getLabel() function. The resulting values are stored in the temp list.

Converting to Numpy Array:

```
temp = np.asarray(temp)
```

- This line converts the temp list into a numpy array.

Updating Target Variable (Y):

```
Y = temp
```

- This line updates the target variable Y with the mapped numerical values.

Converting DataFrame to Numpy Array (X):

```
dataset = dataset.values
X = dataset[:, 0:dataset.shape[1]-1]
```

- This block converts the dataset DataFrame into a numpy array. The features are stored in X, excluding the target variable column.

Normalizing Features:

```
X = normalize(X)
```

- This line normalizes the feature matrix X.

Shuffling Data:

```
indices = np.arange(X.shape[0])
np.random.shuffle(indices)
X = X[indices]
Y = Y[indices]
```

- This block shuffles the feature matrix X and the target variable Y to randomize the order of data samples.

Displaying Processed Dataset Information:

```
text.insert(END, "Dataset after features processing & normalization\n\n")
text.insert(END, str(X) + "\n\n")
text.insert(END, "Total records found in dataset: " + str(X.shape[0]) + "\n")
text.insert(END, "Total features found in dataset: " + str(X.shape[1]) + "\n\n")
```

- These lines display information about the processed dataset in the text widget.

Principal Component Analysis (PCA):

```
pca = PCA(n_components=50)
X = pca.fit_transform(X)
```

- This block performs Principal Component Analysis (PCA) on the feature matrix X to reduce its dimensionality to 50 components.

Splitting Dataset into Train and Test Sets:

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
```

- This line splits the dataset into training and testing sets with 80% of the data used for training (X_train, y_train) and 20% for testing (X_test, y_test).

Function Definition:

```
def calculateMetrics(algorithm, predict, y_test)
```

- This line defines a function named calculateMetrics() which takes three parameters:
  - algorithm: The name of the algorithm for which metrics are being calculated.
  - predict: The predicted labels or classes generated by the algorithm.
  - y_test: The true labels or classes from the test set.

Calculating Evaluation Metrics:

```
a = accuracy_score(y_test, predict) * 100
p = precision_score(y_test, predict, average='macro') * 100
r = recall_score(y_test, predict, average='macro') * 100
f = f1_score(y_test, predict, average='macro') * 100
```

- These lines calculate various evaluation metrics:
  - Accuracy (a): Measures the proportion of correctly classified samples.
  - Precision (p): Measures the ability of the classifier not to label a sample as positive when it is negative.

- Recall (r): Measures the ability of the classifier to find all the positive samples.
- F1-score (f): Harmonic mean of precision and recall, providing a balance between the two metrics.

Appending Metrics to Lists:

accuracy.append(a)

precision.append(p)

recall.append(r)

fscore.append(f)

- These lines append the calculated metrics to global lists (accuracy, precision, recall, fscore) for further analysis.

Displaying Metrics:

text.insert(END, algorithm + " Accuracy: " + str(a) + "\n")

text.insert(END, algorithm + " Precision: " + str(p) + "\n")

text.insert(END, algorithm + " Recall: " + str(r) + "\n")

text.insert(END, algorithm + " F1-Score: " + str(f) + "\n\n")

text.update_idletasks()

- These lines display the calculated metrics in a text widget (text) along with the name of the algorithm.

Printing Unique Predicted and True Labels:

print(np.unique(predict))

print(np.unique(y_test))

- These lines print the unique values present in the predicted and true labels. It helps to ensure that the labels are correctly represented.

Confusion Matrix Visualization:

conf_matrix = confusion_matrix(y_test, predict)

ax = sns.heatmap(conf_matrix, xticklabels=labels, yticklabels=labels, annot=True, cmap="viridis", fmt="g")

ax.set_ylim([0, len(labels)])

plt.title(algorithm + " Confusion Matrix")

plt.ylabel('True class')

```
plt.xlabel('Predicted class')
plt.show()
```

- This block generates a confusion matrix based on the true and predicted labels. The confusion matrix provides a visual representation of the classifier's performance, showing the number of true positive, true negative, false positive, and false negative predictions for each class. The seaborn library is used to create a heatmap visualization of the confusion matrix.

Function Definition:

```
def runNaiveBayes():
```

- This line defines a function named runNaiveBayes()

Global Variables:

```
global X, Y, X_train, X_test, y_train, y_test
global accuracy, precision, recall, fscore
```

- This line declares global variables that are used for storing data and evaluation metrics.

Initialization of Metric Lists:

```
accuracy = []
precision = []
recall = []
fscore = []
```

- These lines initialize empty lists to store accuracy, precision, recall, and F1-score metrics.

Clearing Text Widget:

```
text.delete('1.0', END)
```

- This line clears the content of a text widget (text).

Loading or Training Naive Bayes Model:

```
if os.path.exists('model/nb.txt'):
    with open('model/nb.txt', 'rb') as file:
        nb = pickle.load(file)
else:
    nb = GaussianNB()
    nb.fit(X_train, y_train)
```

```
  with open('model/nb.txt', 'wb') as file:
        pickle.dump(nb, file)
```

- These lines check if a pre-trained Naive Bayes model exists ('model/nb.txt'). If it does, it loads the model using pickle. Otherwise, it initializes and trains a new Naive Bayes model using the training data (X_train, y_train), and then saves it using pickle for future use.

Making Predictions:

```
predict = nb.predict(X_test)
```

- This line uses the trained Naive Bayes model to make predictions on the test data (X_test), and stores the predictions in the predict variable.

Calculating Metrics:

```
calculateMetrics("NaiveBayes", predict, y_test)
```

- This line calls the calculateMetrics() function to calculate evaluation metrics (accuracy, precision, recall, and F1-score) based on the predictions made by the Naive Bayes classifier (predict) and the true labels (y_test). The algorithm name "NaiveBayes" is also passed as an argument to identify the algorithm in the output.

Function Definition:

```
def predict():
```

- This line defines a function named predict()

Deleting Previous Output:

```
text.delete('1.0', END)
```

- This line clears any previous output displayed in the text widget (text).

Selecting Test Data File:

```
filename = filedialog.askopenfilename(initialdir="testData")
testData = pd.read_csv(filename)
```

- These lines open a file dialog to allow the user to select a test data file. Once a file is selected, it reads the data from the CSV file into a pandas DataFrame (testData).

Data Preprocessing:

```
testData.fillna(0, inplace=True)
testData = testData.values
```

testData = normalize(testData)

testData = pca.transform(testData)

- These lines preprocess the test data in a series of steps:
  - Missing values are filled with zeros (fillna(0)).
  - The DataFrame is converted to a NumPy array (testData.values).
  - The data is normalized using a scaler (normalize).
  - The data is transformed using Principal Component Analysis (PCA) (pca.transform).

Making Predictions:

predict = classifier.predict(testData)

- This line uses the pre-trained classifier (classifier) to predict labels for the preprocessed test data (testData), and stores the predictions in the predict variable.

Displaying Predictions:

```
for i in range(len(predict)):
    text.insert(END, "Test DATA: " + str(testData[i]) + " ===> PREDICTED " +
labels[predict[i]] + "\n\n")
```

- This loop iterates over each prediction made by the classifier and inserts a formatted string into the text widget (text). The string includes the original test data, the predicted label, and a newline character for formatting.

Explanation:

- The predict() function allows users to select a CSV file containing test data.
- It preprocesses the test data by filling missing values with zeros, normalizing the data, and transforming it using PCA.
- The pre-trained classifier is then used to predict labels for the preprocessed test data.
- Finally, the function displays the test data along with the predicted labels in the text widget.

Function Definition:

def graph():

- This line defines a function named graph()

Creating HTML Output:

output = "<html><body><table align=center border=1><tr><th>Algorithm</th><th>FSCORE</th></tr>"

- This line initializes a string variable output with an HTML structure for a table. It includes table headers for "Algorithm" and "FSCORE".

Appending Algorithm Metrics:

```
output += "<tr><td>Naive Bayes</td><td>" + str(fscore[0]) + "</td></tr>"
output += "<tr><td>Random Forest</td><td>" + str(fscore[1]) + "</td></tr>"
output += "<tr><td>SVM</td><td>" + str(fscore[2]) + "</td></tr>"
output += "<tr><td>XGBoost</td><td>" + str(fscore[3]) + "</td></tr>"
output += "<tr><td>AdaBoostBoost</td><td>" + str(fscore[4]) + "</td></tr>"
output += "<tr><td>KNN</td><td>" + str(fscore[5]) + "</td></tr>"
```

- These lines append rows to the HTML table for each algorithm, along with their corresponding FSCORE values.

Writing HTML Table to File:

```
f = open("table.html", "w")
f.write(output)
f.close()
```

- These lines open a file named "table.html" in write mode, write the HTML table content (stored in the output variable) to the file, and then close the file.

Opening HTML File in Web Browser:

```
webbrowser.open("table.html", new=2)
```

- This line opens the generated HTML file ("table.html") in a web browser. The new=2 argument specifies that the file should be opened in a new tab or window.

Plotting Performance Metrics:

```
df = pd.DataFrame([['Naive Bayes','Precision',precision[0]],['Naive Bayes','Recall',recall[0]],['NaiveBayes','F1Score',fscore[0]],['Naive Bayes','Accuracy',accuracy[0]],
df.pivot("Algorithms","PerformanceOutput","Value").plot(kind='bar')
plt.show()
```

- These lines create a pandas DataFrame containing precision, recall, F1 score, and accuracy for each algorithm. Then, it generates a bar plot of these metrics and displays it using plt.show().

Explanation:

- The graph() function generates an HTML table containing FSCORE values for various machine learning algorithms based on their performance.

- It writes this HTML content to a file named "table.html" and opens it in a web browser.

- Additionally, it creates a bar plot of performance metrics for each algorithm using pandas Data Frame and matplotlib. However, this part of the code seems to be incomplete or incorrect, as it's not properly formatted and contains errors.

```
font =('times', 16, 'bold')
title=Label(main,text='DetectionandclassificationofDDOSAttack ')
title.config(bg='greenyellow',fg='dodgerblue')
title.config(font=font)
title.config(height=3,width=120)
title.place(x=0,y=5)
font1 = ('times', 12, 'bold')
text=Text(main,height=20,width=150)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=50,y=120)
text.config(font=font1)
font1 =('times', 13, 'bold')
uploadButton=Button(main,text="UploadDDOSDataset",command=uploadDataset)
uploadButton.place(x=50,y=550)
uploadButton.config(font=font1)
preprocessButton=Button(main,text="PreprocessDataset",command=preprocessDataset)
preprocessButton.place(x=330,y=550)
preprocessButton.config(font=font1)
nbButton=Button(main,text="RunNaiveBayesAlgorithm",command=runNaiveBayes)
nbButton.place(x=630,y=550)
nbButton.config(font=font1)
rfButton=Button(main,text="RunRandomForestAlgorithm",command=runRandomForest)
rfButton.place(x=920,y=550)
rfButton.config(font=font1)
svmButton=Button(main,text="RunSVMAlgorithm",command=runSVM)
svmButton.place(x=50,y=600)
svmButton.config(font=font1)
xgButton=Button(main,text="RunXGBoostAlgorithm",command=runXGBoost)
xgButton.place(x=330,y=600)
xgButton.config(font=font1)
adaboostButton=Button(main,text="RunAdaBoostAlgorithm",command=runAdaBoost)
adaboostButton.place(x=630,y=600)
adaboostButton.config(font=font1)
knnButton=Button(main,text="RunKNNAlgorithm",command=runKNN)
knnButton.place(x=920,y=600)
knnButton.config(font=font1)
graphButton=Button(main,text="ComparisonGraph",command=graph)
graphButton.place(x=50,y=650)
graphButton.config(font=font1)
predictButton=Button(main,text="PredictAttackfromTestData",command=predict)
predictButton.place(x=330,y=650)
predictButton.config(font=font1
) main.config(bg='Blue')
main.mainloop()
```

Setting Fonts:
font = ('times', 16, 'bold'): Defines a font named "times" with a size of 16 points and bold style.
font1 = ('times', 12, 'bold'): Defines another font with a smaller size for buttons.

Creating a Title Label:
Creates a Label widget named title with the text "Detection and classification of DDOS Attack" and sets its background color to "greenyellow" and foreground (text) color to "dodgerblue".
Configures the font of the title label to the previously defined font.
Sets the height and width of the title label.
Places the title label at coordinates (0, 5) within the main window.

Creating a Text Widget:
Creates a Text widget named text to display text information. It's configured with a height of 20 lines and a width of 150 characters.
Also creates a Scrollbar named scroll and associates it with the text widget to enable scrolling.
Places the text widget at coordinates (50, 120) within the main window.

Creating Buttons:

Creates various buttons for different functionalities of the application, such as uploading datasets, preprocessing data, running different algorithms, generating comparison graphs, and predicting attacks from test data.

Each button is associated with a specific command to execute when clicked, such as calling functions like uploadDataset, preprocessDataset, runNaiveBayes, etc.

Configures the font of all buttons to font1 defined earlier.

Places each button at specific coordinates within the main window.

Setting Background Color and Running the Main Loop:

Sets the background color of the main window to "Blue".

Calls the mainloop() method to start the GUI event loop, allowing the user to interact with the application.

Overall, this code segment sets up the visual components and functionalities of the GUI application for detecting and classifying DDOS attacks in an IoT network environment.

6.2 Output Screens

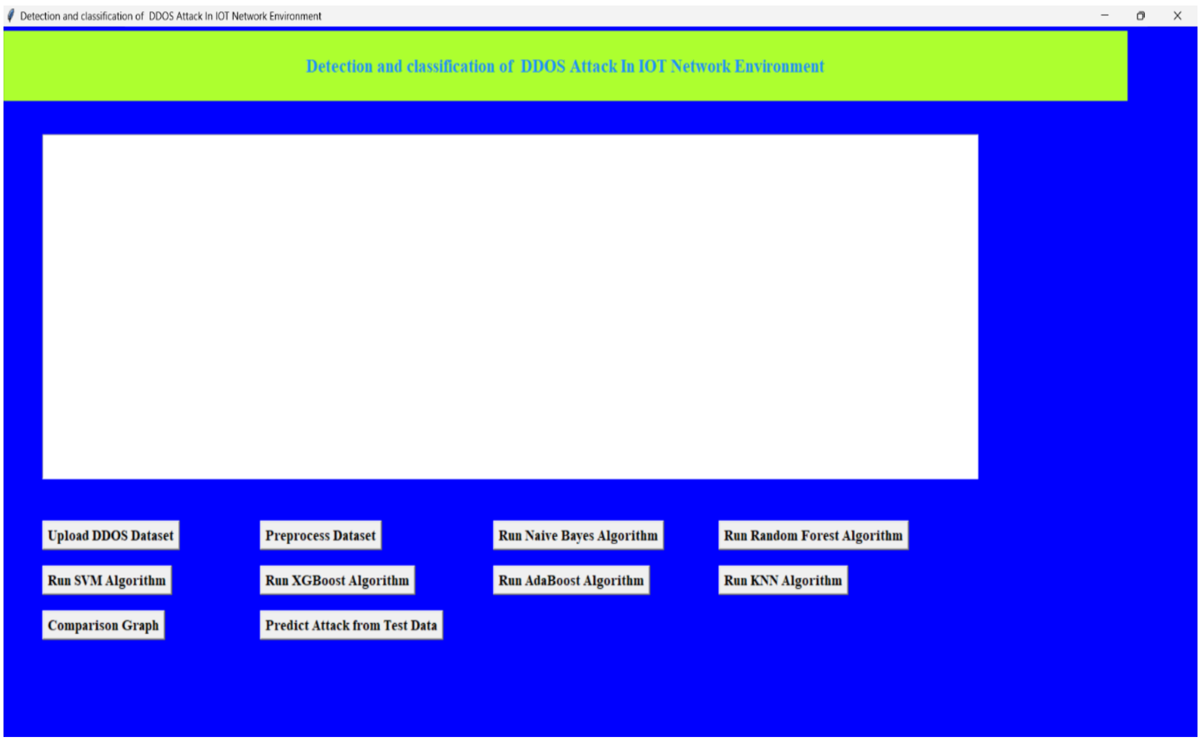
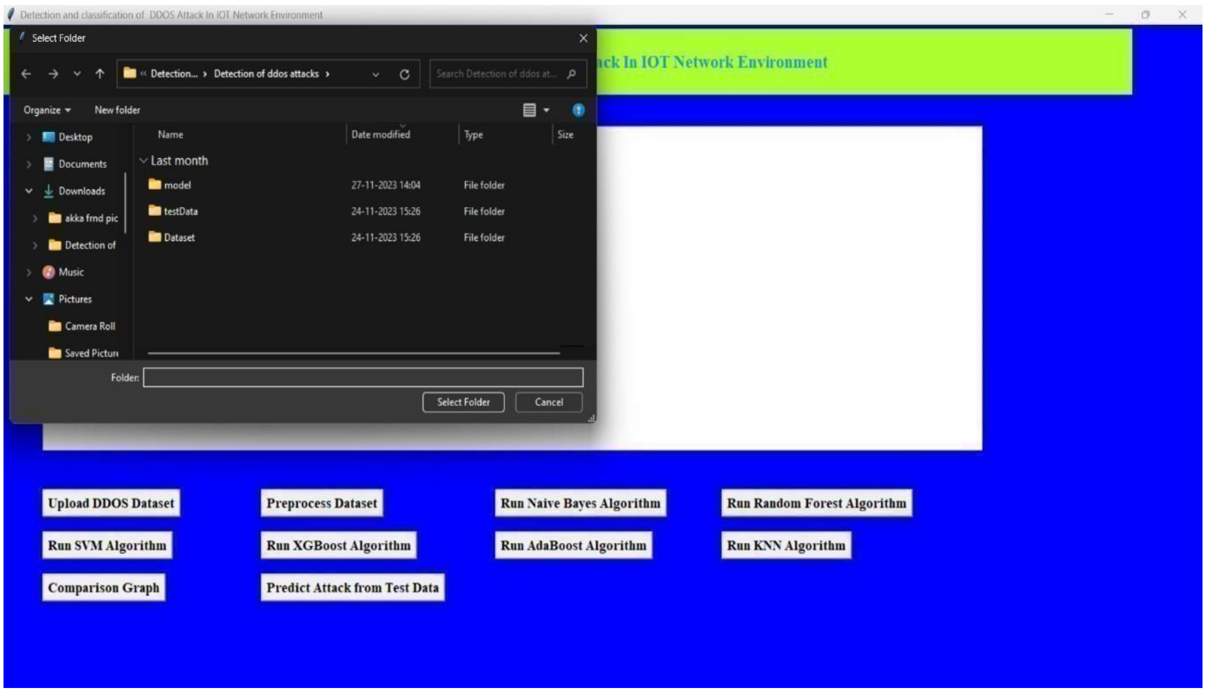

Detection and classification of DDOS Attack In IOT Network Environment
Detection and classification of DDOS Attack In IOT Network Environment
Upload DDOS Dataset
Preprocess Dataset
Run Naive Bayes Algorithm
Run Random Forest Algorithm
Run SVM Algorithm
Run XGBoost Algorithm
Run AdaBoost Algorithm
Run KNN Algorithm
Comparison Graph
Predict Attack from Test Data

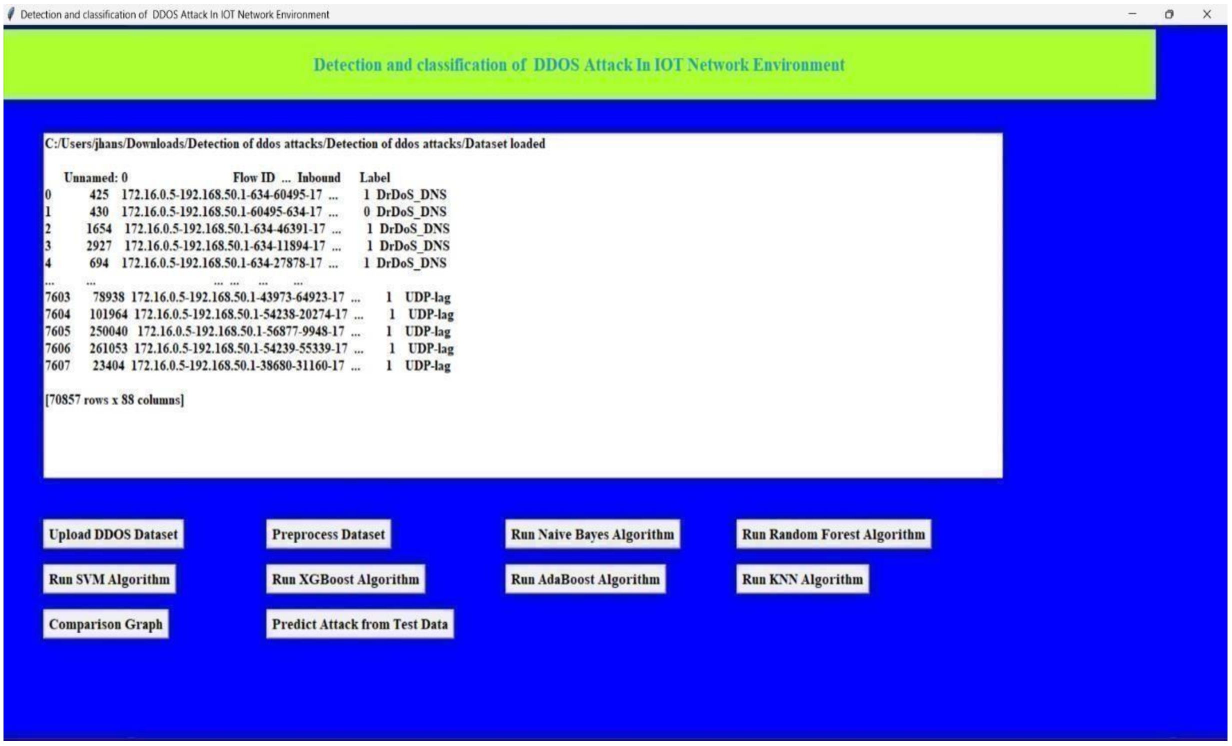
Fig11: Main GUI application of proposed DDOS attack Detection &Classification



Name	Date modified	Type
model	27-11-2023 14:04	File folder
testData	24-11-2023 15:26	File folder
Dataset	24-11-2023 15:26	File folder

Fig12: Selecting the dataset in the GUI application

The fig12 shows a screen or window with in the GUI where users (possibly administrators or analysts) can select a dataset for analysis or model training


Detection and classification of DDOS Attack In IOT Network Environment
Detection and classification of DDOS Attack In IOT Network Environment
Upload DDOS Dataset
Preprocess Dataset
Run Naive Bayes Algorithm
Run Random Forest Algorithm
Run SVM Algorithm
Run XGBoost Algorithm
Run AdaBoost Algorithm
Run KNN Algorithm
Comparison Graph
Predict Attack from Test Data

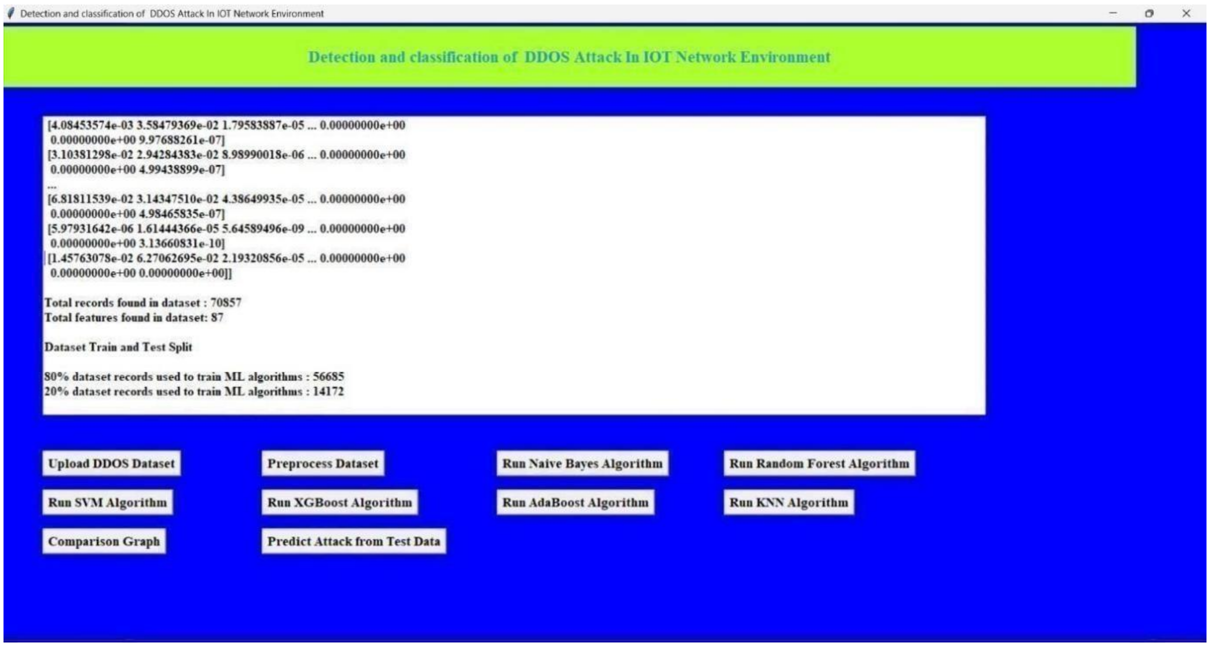
```
C:/Users/jhans/Downloads/Detection of ddos attacks/Detection of ddos attacks/Dataset loaded

    Unnamed: 0                          Flow ID  ... Inbound     Label
0          425   172.16.0.5-192.168.50.1-634-60495-17  ...     1  DrDoS_DNS
1          430   172.16.0.5-192.168.50.1-60495-634-17  ...     0  DrDoS_DNS
2         1654   172.16.0.5-192.168.50.1-634-46391-17  ...     1  DrDoS_DNS
3         2927   172.16.0.5-192.168.50.1-634-11894-17  ...     1  DrDoS_DNS
4          694   172.16.0.5-192.168.50.1-634-27878-17  ...     1  DrDoS_DNS
...        ...                                   ...  ...   ...       ...
7603     78938  172.16.0.5-192.168.50.1-43973-64923-17  ...     1    UDP-lag
7604    101964  172.16.0.5-192.168.50.1-54238-20274-17  ...     1    UDP-lag
7605    250040   172.16.0.5-192.168.50.1-56877-9948-17  ...     1    UDP-lag
7606    261053  172.16.0.5-192.168.50.1-54239-55339-17  ...     1    UDP-lag
7607     23404  172.16.0.5-192.168.50.1-38680-31160-17  ...     1    UDP-lag

[70857 rows x 88 columns]
```

Fig13: Displays the dataset

The Fig13 presents the content of the selected dataset.


Detection and classification of DDOS Attack In IOT Network Environment
Detection and classification of DDOS Attack In IOT Network Environment
Upload DDOS Dataset
Preprocess Dataset
Run Naive Bayes Algorithm
Run Random Forest Algorithm
Run SVM Algorithm
Run XGBoost Algorithm
Run AdaBoost Algorithm
Run KNN Algorithm
Comparison Graph
Predict Attack from Test Data

```
[4.08453574e-03 3.58479369e-02 1.79583887e-05 ... 0.00000000e+00
 0.00000000e+00 9.97688261e-07]
[3.10381298e-02 2.94284383e-02 8.98990018e-06 ... 0.00000000e+00
 0.00000000e+00 4.99438899e-07]
...
[6.81811539e-02 3.14347510e-02 4.38649935e-05 ... 0.00000000e+00
 0.00000000e+00 4.98465835e-07]
[5.97931642e-06 1.61444366e-05 5.64589496e-09 ... 0.00000000e+00
 0.00000000e+00 3.13660831e-10]
[1.45763078e-02 6.27062695e-02 2.19320856e-05 ... 0.00000000e+00
 0.00000000e+00 0.00000000e+00]]

Total records found in dataset : 70857
Total features found in dataset: 87

Dataset Train and Test Split

80% dataset records used to train ML algorithms : 56685
20% dataset records used to train ML algorithms : 14172
```

Fig14: Presents the pre-processed data from the dataset.

The fig14 displays the dataset after under going preprocessing steps such as cleaning, transforming, or feature engineering, to prepare the data for analysis and model training.

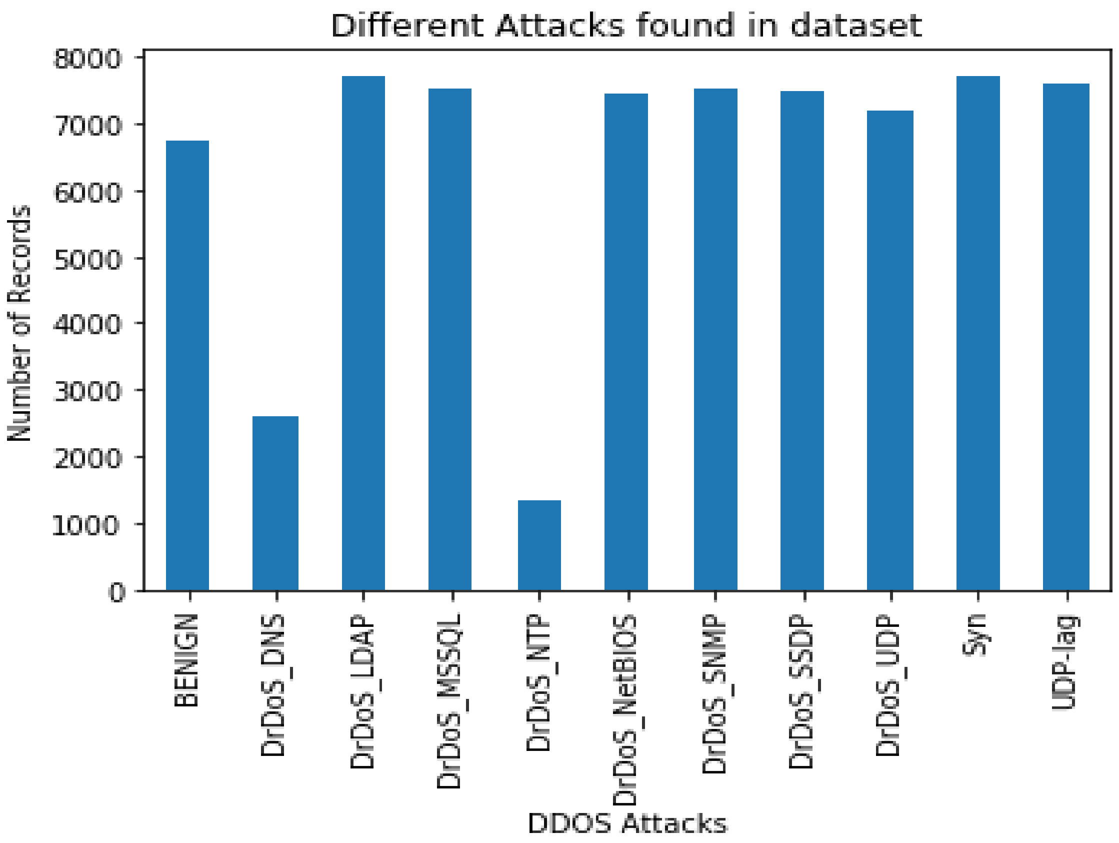
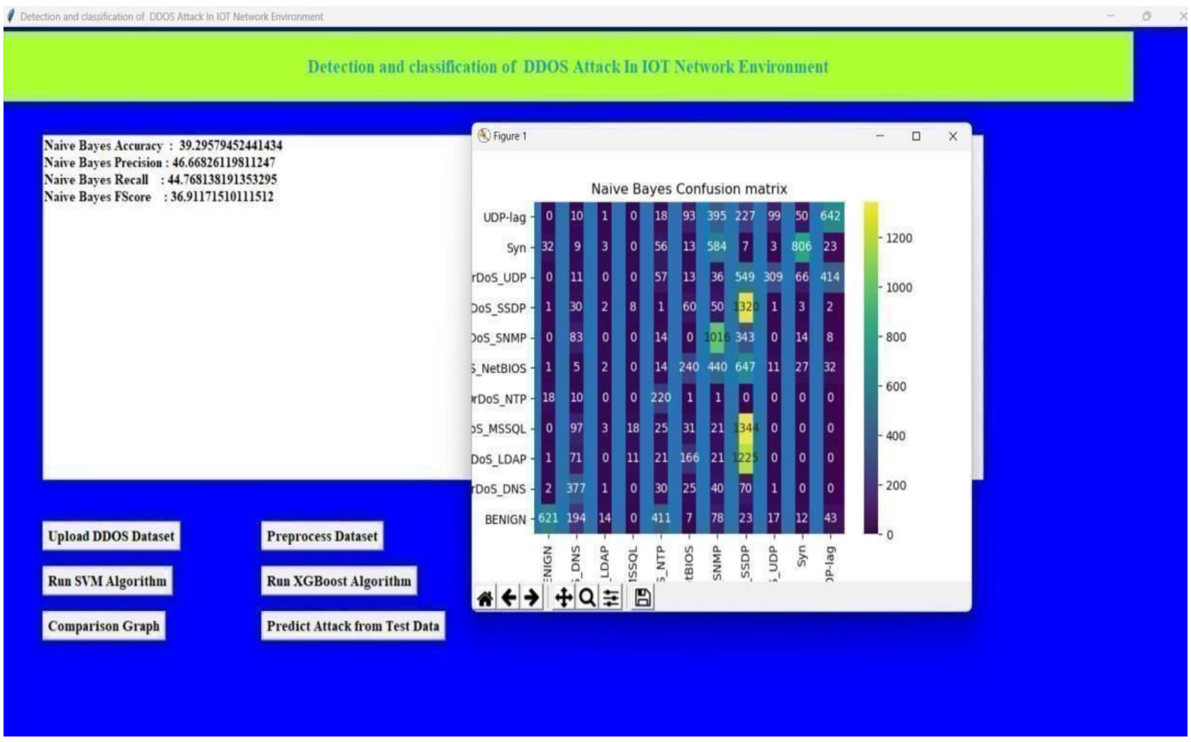


Fig15: Graphical presentation of different attacks found in dataset


Detection and classification of DDOS Attack In IOT Network Environment
Detection and classification of DDOS Attack In IOT Network Environment
Figure 1
Upload DDOS Dataset
Preprocess Dataset
Run SVM Algorithm
Run XGBoost Algorithm
Comparison Graph
Predict Attack from Test Data

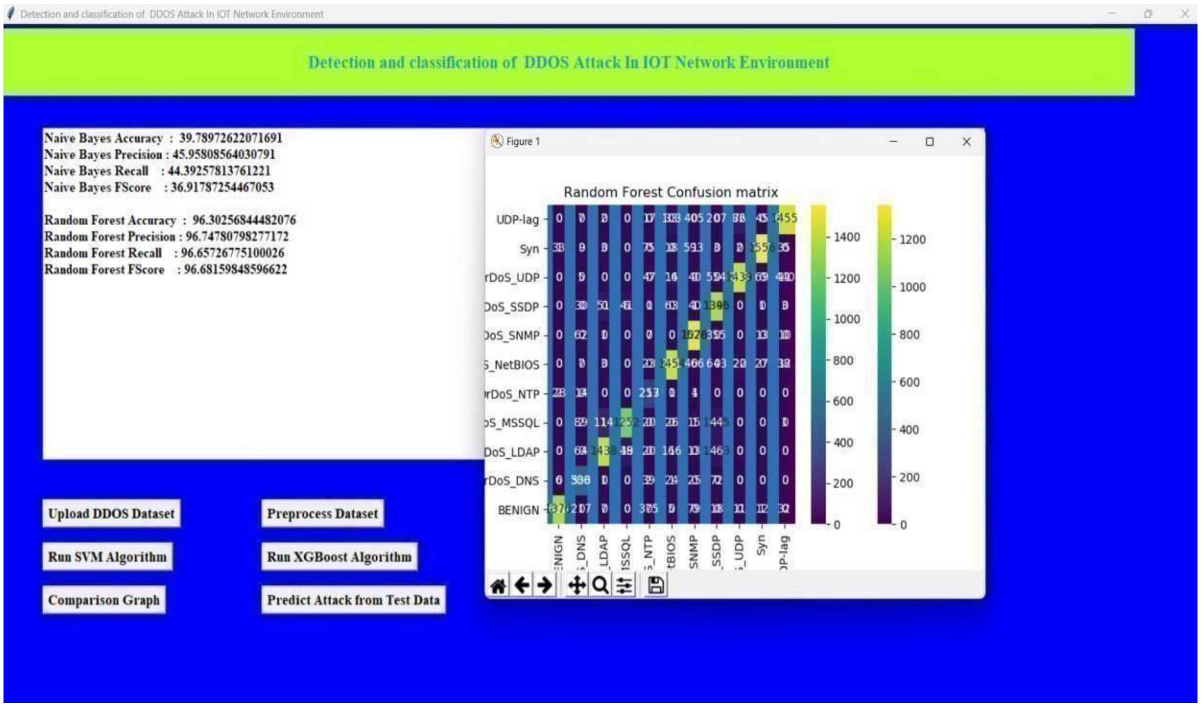
```
Naive Bayes Accuracy  :  39.29579452441434
Naive Bayes Precision : 46.66826119811247
Naive Bayes Recall    : 44.768138191353295
Naive Bayes FScore    : 36.91171510111512
```

Naive Bayes Confusion matrix

	ENIGN	_DNS	LDAP	ISSQL	_NTP	tBIOS	SNMP	SSDP	_UDP	Syn	)P-lag
UDP-lag	0	10	1	0	18	93	395	227	99	50	642
Syn	32	9	3	0	56	13	584	7	3	806	23
rDoS_UDP	0	11	0	0	57	13	36	549	309	66	414
DoS_SSDP	1	30	2	8	1	60	50	1320	1	3	2
DoS_SNMP	0	83	0	0	14	0	1016	343	0	14	8
S_NetBIOS	1	5	2	0	14	240	440	647	11	27	32
rDoS_NTP	18	10	0	0	220	1	1	0	0	0	0
oS_MSSQL	0	97	3	18	25	31	21	1344	0	0	0
DoS_LDAP	1	71	0	11	21	166	21	1225	0	0	0
rDoS_DNS	2	377	1	0	30	25	40	70	1	0	0
BENIGN	621	194	14	0	411	7	78	23	17	12	43

Fig16: Displays the performance evaluation and confusion matrix of the Naïve Bayes

The fig16 shows the evaluation metrics such as accuracy, precision, recall and Fscore and Confusion matrix, especially for Naïve bayes model applied to the dataset.

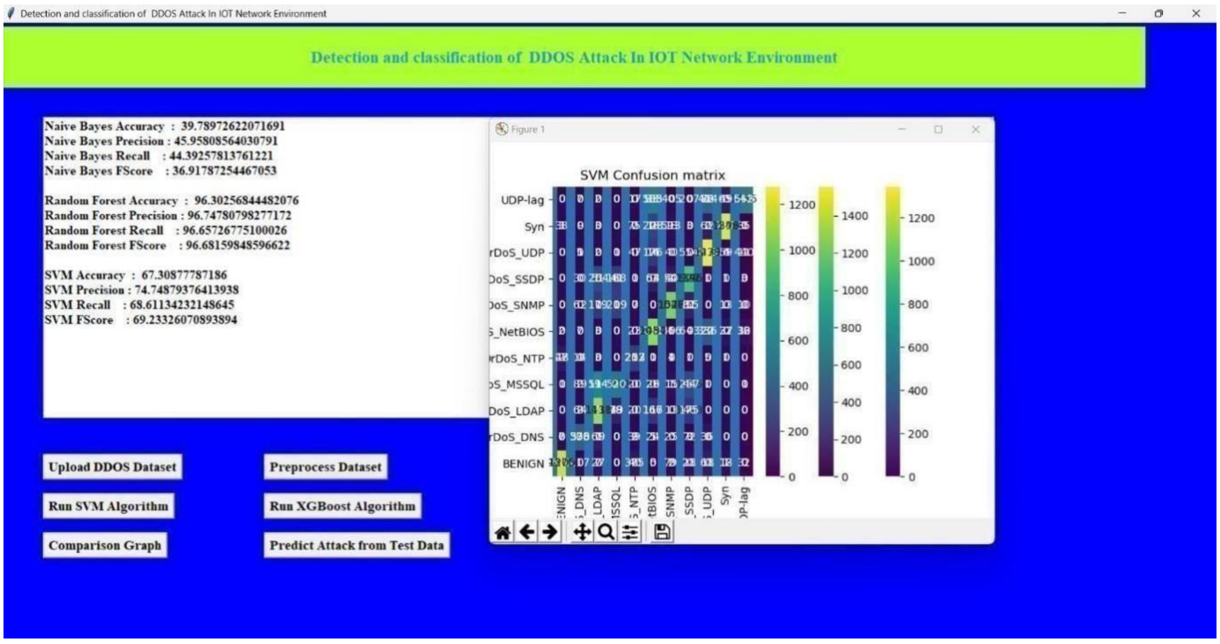

Detection and classification of DDOS Attack In IOT Network Environment
Detection and classification of DDOS Attack In IOT Network Environment
Figure 1
Random Forest Confusion matrix
Upload DDOS Dataset
Preprocess Dataset
Run SVM Algorithm
Run XGBoost Algorithm
Comparison Graph
Predict Attack from Test Data

```
Naive Bayes Accuracy  :  39.78972622071691
Naive Bayes Precision : 45.95808564030791
Naive Bayes Recall    : 44.39257813761221
Naive Bayes FScore    : 36.91787254467053

Random Forest Accuracy  :  96.30256844482076
Random Forest Precision : 96.74780798277172
Random Forest Recall    : 96.65726775100026
Random Forest FScore    : 96.68159848596622
```

Fig17: Displays the performance evaluation and confusion matrix of the Random Forest model

The fig17 shows the evaluation metrics such as accuracy, precision, recall and Fscore andConfusion matrix, especially for Random Forest model applied to the dataset.


Detection and classification of DDOS Attack In IOT Network Environment
Detection and classification of DDOS Attack In IOT Network Environment
Figure 1
SVM Confusion matrix
Upload DDOS Dataset
Preprocess Dataset
Run SVM Algorithm
Run XGBoost Algorithm
Comparison Graph
Predict Attack from Test Data

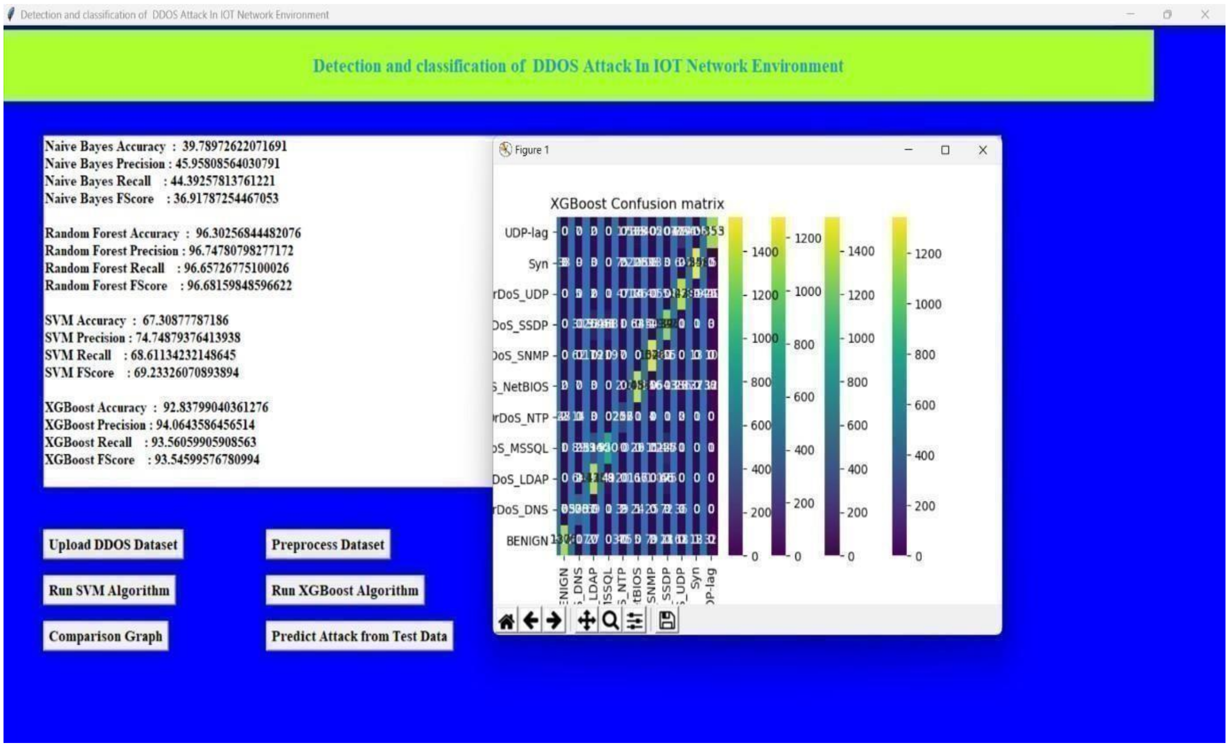
```
Naive Bayes Accuracy  :  39.78972622071691
Naive Bayes Precision : 45.95808564030791
Naive Bayes Recall    : 44.39257813761221
Naive Bayes FScore    : 36.91787254467053

Random Forest Accuracy  :  96.30256844482076
Random Forest Precision : 96.74780798277172
Random Forest Recall    : 96.65726775100026
Random Forest FScore    : 96.68159848596622

SVM Accuracy  :  67.30877787186
SVM Precision : 74.74879376413938
SVM Recall    : 68.61134232148645
SVM FScore    : 69.23326070893894
```

Fig18: Displays the performance evaluation and confusion matrix of the SVM model

The fig18 shows the evaluation metrics such as accuracy, precision, recall and Fscore and Confusion matrix, especially for SVM model applied to the dataset.



```
Naive Bayes Accuracy  :  39.78972622071691
Naive Bayes Precision : 45.95808564030791
Naive Bayes Recall    : 44.39257813761221
Naive Bayes FScore    : 36.91787254467053

Random Forest Accuracy  :  96.30256844482076
Random Forest Precision : 96.74780798277172
Random Forest Recall    : 96.65726775100026
Random Forest FScore    : 96.68159848596622

SVM Accuracy  :  67.30877787186
SVM Precision : 74.74879376413938
SVM Recall    : 68.61134232148645
SVM FScore    : 69.23326070893894

XGBoost Accuracy  :  92.83799040361276
XGBoost Precision : 94.0643586456514
XGBoost Recall    : 93.56059905908563
XGBoost FScore    : 93.54599576780994
```

Fig19: Displays the performance evaluation and confusion matrix of the XG Boost model

Thefig19 shows the evaluation metrics such as accuracy, precision, recall and Fscore and Confusion matrix, especially for XG Boost model applied to the dataset.

Algorithm Name	Accuracy	Precision	Recall	FSCORE
Naive Bayes Algorithm	39.486311035845326	47.16286496879783	44.77971559430494	37.29344143668989
Random Forest Algorithm	96.40135478408128	96.82304018437915	96.76737089401027	96.77043166078197
SVM Algorithm	66.92774484899803	74.7981230463062	68.24496787236103	68.93484425624501
XGBoost Algorithm	92.73920406435224	93.97249067240224	93.6041695097032	93.53399031033462
AdaBoostBoost Algorithm	56.244707874682476	60.45928276981275	53.576842001950084	53.986776567448466
KNN Algorithm	84.61755574372	87.94007146033593	85.952294449359	85.81439187553987

Fig20: Comparison table of Algorithm

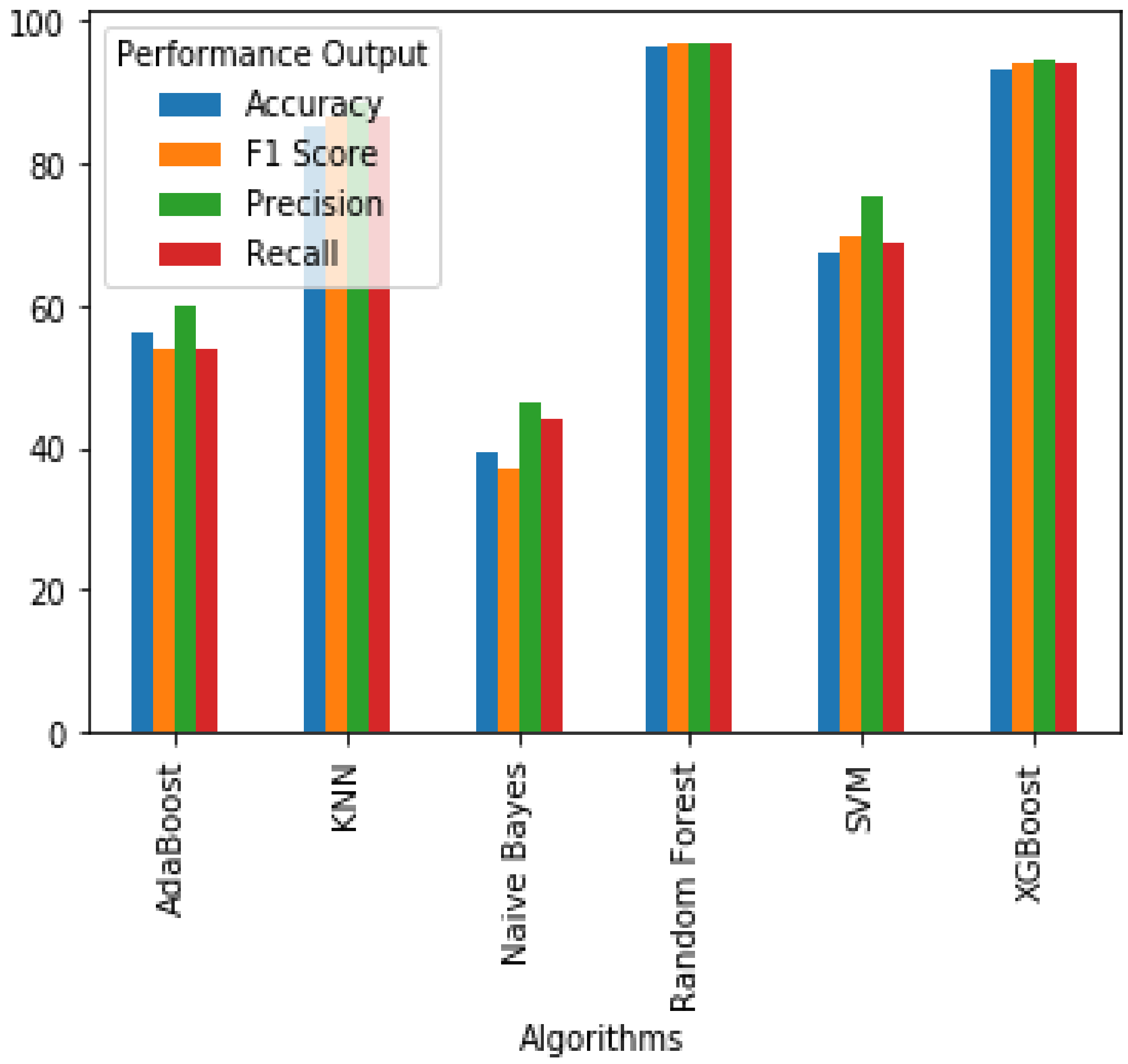


Fig21: Comparison Graph of Algorithms

The Fig 20,21 Displays the comparison of performance metrics in Navie Bayes, Random Forest, SVM, XGboost, Adaboost and KNN Algorithms

We will upload test data and then machine learning models will predict attack from that test data.

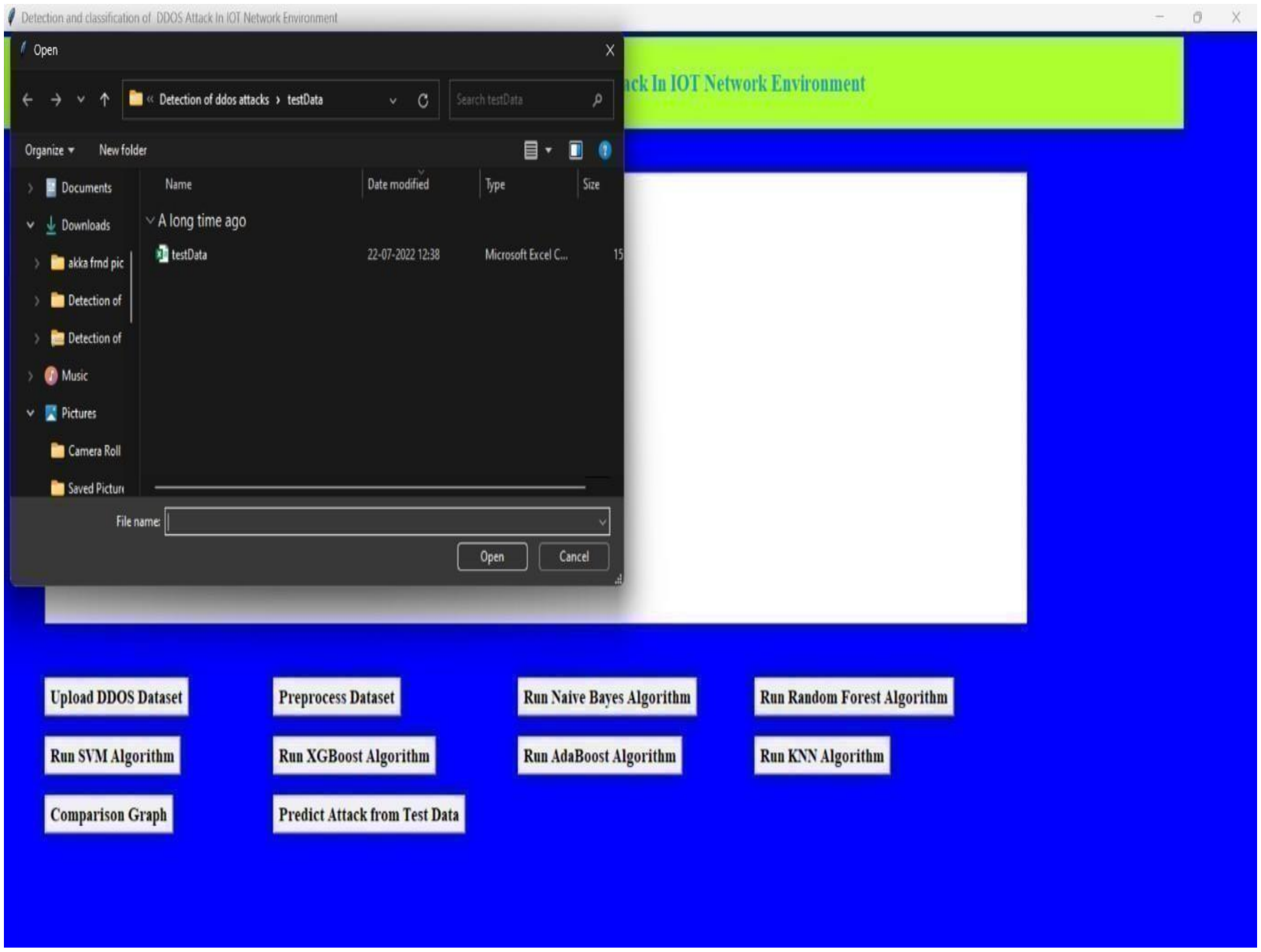
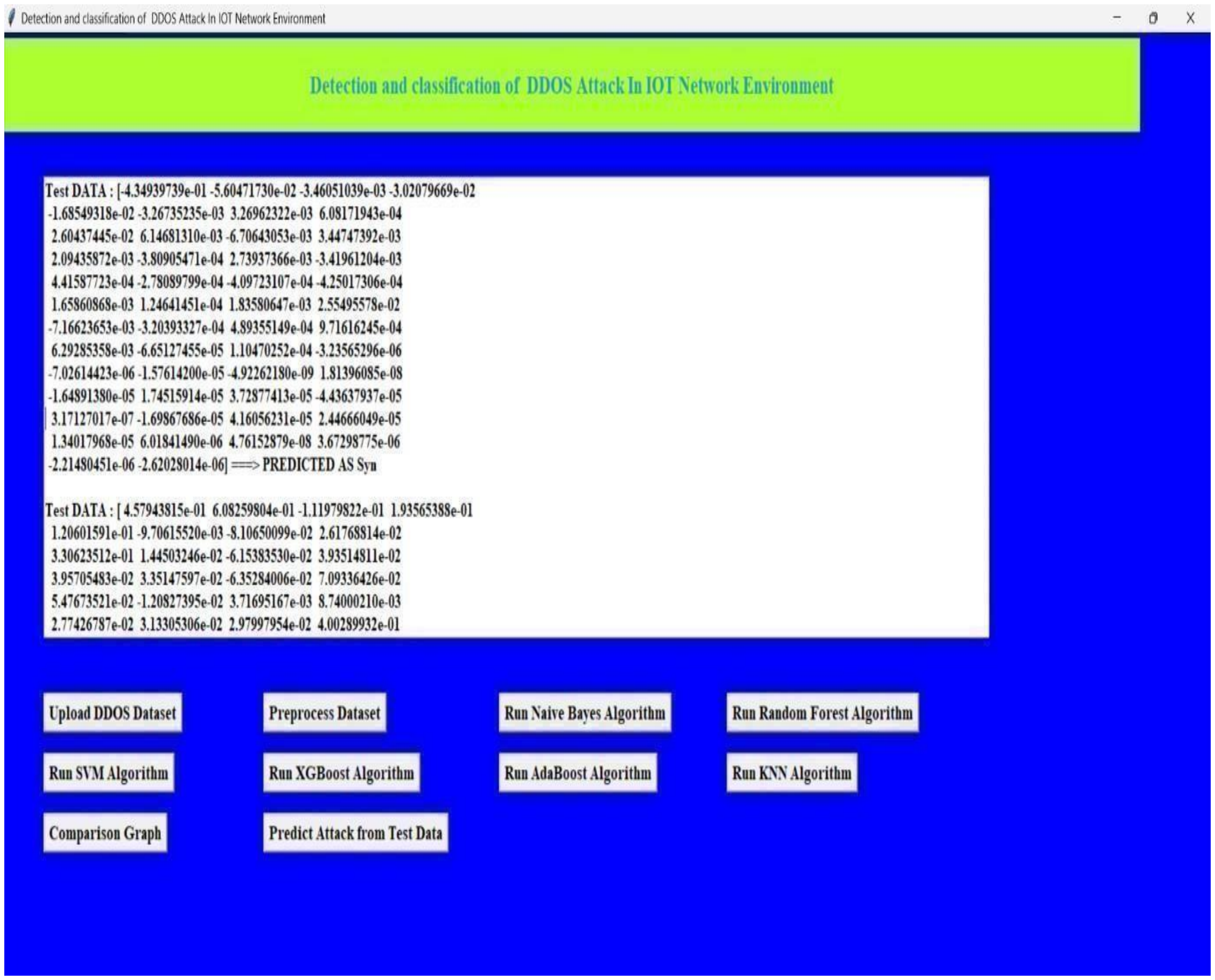


Fig22: Selecting the test dataset


Detection and classification of DDOS Attack In IOT Network Environment
Detection and classification of DDOS Attack In IOT Network Environment
Upload DDOS Dataset
Preprocess Dataset
Run Naive Bayes Algorithm
Run Random Forest Algorithm
Run SVM Algorithm
Run XGBoost Algorithm
Run AdaBoost Algorithm
Run KNN Algorithm
Comparison Graph
Predict Attack from Test Data

```
Test DATA : [-4.34939739e-01 -5.60471730e-02 -3.46051039e-03 -3.02079669e-02
 -1.68549318e-02 -3.26735235e-03  3.26962322e-03  6.08171943e-04
  2.60437445e-02  6.14681310e-03 -6.70643053e-03  3.44747392e-03
  2.09435872e-03 -3.80905471e-04  2.73937366e-03 -3.41961204e-03
  4.41587723e-04 -2.78089799e-04 -4.09723107e-04 -4.25017306e-04
  1.65860868e-03  1.24641451e-04  1.83580647e-03  2.55495578e-02
 -7.16623653e-03 -3.20393327e-04  4.89355149e-04  9.71616245e-04
  6.29285358e-03 -6.65127455e-05  1.10470252e-04 -3.23565296e-06
 -7.02614423e-06 -1.57614200e-05 -4.92262180e-09  1.81396085e-08
 -1.64891380e-05  1.74515914e-05  3.72877413e-05 -4.43637937e-05
  3.17127017e-07 -1.69867686e-05  4.16056231e-05  2.44666049e-05
  1.34017968e-05  6.01841490e-06  4.76152879e-08  3.67298775e-06
 -2.21480451e-06 -2.62028014e-06] ==> PREDICTED AS Syn

Test DATA : [ 4.57943815e-01  6.08259804e-01 -1.11979822e-01  1.93565388e-01
  1.20601591e-01 -9.70615520e-03 -8.10650099e-02  2.61768814e-02
  3.30623512e-01  1.44503246e-02 -6.15383530e-02  3.93514811e-02
  3.95705483e-02  3.35147597e-02 -6.35284006e-02  7.09336426e-02
  5.47673521e-02 -1.20827395e-02  3.71695167e-03  8.74000210e-03
  2.77426787e-02  3.13305306e-02  2.97997954e-02  4.00289932e-01
```

Fig23: Represents the prediction of test data

The Fig23 Represents the prediction of test data and type of DDos attack is classified

CHAPTER 7

CONCLUSION

This paper proposes a new DDOS attack detection method, which is a random forest algorithmmodel based on machine learning. By extracting the three protocol attack packets of the DDOS attacktool, feature extraction and format conversion are performed to extract the DDoS attack traffic characteristics with a large proportion. Then the extracted features are used as input features of machine learning, and the random forest algorithm is used to train and obtain the DDoS attack detection model. Then the normal traffic data is mixed with the attack data for model test. The experimental results show that the proposed DDoS attack detection method based on machine learning has a good detection rate for the current popular DDoS attacks.

BIBLIOGRAPHY

[1] R. Braga, E. Mota, and A. Passito, “Lightweight DDoS flooding attack detection using NOX/OpenFlow,” in Proceedings of the 35th Annual IEEE Conference on Local Computer Networks (LCN ’10), pp. 408–415, Denver, Colo, USA, October 2015s.

[2] Z. Cai, Z. Wang, K. Zheng, and J. Cao, “A distributed TCAM coprocessor architecture for integrated longest prefix matching, policy filtering, and content filtering,” IEEE Transactions on Computers, vol. 62, no. 3, pp. 417–427, 2015.

[3] Dao, N.N.; Park, J.; Park, M.; Cho, S. A feasible method to combat against DDoS attack in SDN network. In Proceedings of the 2015 International Conference on Information Networking (ICOIN), Siem Reap, Cambodia, 12–14 January 2015; pp. 309–311. doi:10.1109/ICOIN.2015.7057902.

[4] Mousavi, S.M.; St-Hilaire, M. Early detection of DDoS attacks against SDN controllers. In Proceedings of the 2015 International Conference on Computing, Networking and Communications (ICNC), Anaheim, CA, USA, 16–19 February 2015; pp. 77–81. doi:10.1109/ICCNC.2015.7069319.

[5] Dharma, N.I.G.; Muthohar, M.F.; Prayuda, J.D.A.; Priagung, K.; Choi, D. Time-based DDoS detection and mitigation for SDN controller. In Proceedings of the 2015 17th Asia- Pacific Network Operations and Management Symposium (APNOMS), Busan, Korea, 19– 21 August 2015; pp. 550–553. doi:10.1109/APNOMS.2015.7275389.

[6] H. Lin and P. Wang, “Implementation of an SDN-based security defense mechanism against DDoS attacks,” in Proceedings of the 2016 Joint International Conference on Economics and Management Engineering (ICEME 2016) and International Conference on Economics and Business Management (EBM 2016), Pennsylvania, Penn, USA, 2016.

[7] J. G. Yang, X. T. Wang, and L. Q. Liu, “Based on traffic and IP entropy characteristics of DDoS attack detection method,” Application Research of Computers, vol. 33, no. 4, pp. 1145–1149, 2016.

[8] A. Saied, R. E. Overill, and T. Radzik, “Detection of known and unknown DDoS attacks using artificial neural networks,” Neurocomputing, vol. 172, pp. 385–393, 2016.

[9]X. Wang, M. Chen, C. Xing, and T. Zhang, “Defending DDoS attacks in software defined networking based on legitimate source and destination IP address database,” IEICE Transaction on Information and Systems, vol. E99D, no. 4, pp. 850–859, 2016.

[10] Dong, P.; Du, X.; Zhang, H.; Xu, T. A detection method for a novel DDoS attack against SDN controllers by vast new low-traffic flows. In Proceedings of the 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, 23–27 May 2016; pp. 1– 6. doi:10.1109/ICC.2016.7510992.

[11] N. Z. Bawany, J. A. Shamsi, and K. Salah, “DDoS attack detection and mitigation using SDN: methods, practices, and solutions,” Arabian Journal for Science and Engineering, vol. 42, no. 2, pp. 425–441, 2017.

[12] Yan, Q.; Gong, Q.; Yu, F.R. Effective software-defined networking controller scheduling method to mitigate DDoS attacks. Electron. Lett. 2017, 53, 469–471.

[13] J. Cao, M. Xu, Q. Li, K. Sun, Y. Yang, and J. Zheng, “Disrupting SDN via the data plane: a low-rate flow table overow attack,” in Proceedings of the 13th EAI International Conference on Security and Privacy in Communication Networks, Niagara Falls, Canada, October 2017. 8 Security and Communication Networks.

[14] Y. Li, Z. Cai, and H. Xu, “LLMP: exploiting LLDP for latency measurement in software defined data center networks,” Journal of Computer Science and Technology, vol. 33, no.2, pp. 277– 285, 2018.

[15] H. Zhang, Z. Cai, Q. Liu, Q. Xiao, Y. Li, and C. F. Cheang, “A survey on security-aware network measurement in SDN,” Security and Communication Networks, Article ID 2459154, 2018.

[16]J. Xia, Z. Cai, G. Hu, and M. Xu, “An active defense solution for ARP Spoo ng in OpenFlow network,” Chinese Journal of Electronics, vol. 3, 2018.